# Erkennung beliebiger Bälle durch den humanoiden Roboter NAO in der RoboCup Standard Platform League

Bachelorarbeit

Felix Thielke

Matrikelnummer: 2909855

30. September 2016


Fachbereich Mathematik / Informatik
Studiengang Informatik

1. Gutachter: Dr. Thomas Röfer
2. Gutachter: Prof. Dr. Christoph Lüth

## Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig angefertigt und nicht anderweitig zu Prüfungszwecken vorgelegt habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel verwendet. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Bremen, den 30. September 2016

.....

(Felix Thielke)

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

Die vorliegende Arbeit entstand im Kontext des Teams B-Human, einem studentischen Projekt der Universität Bremen in Kooperation mit dem Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI). Seit 2009 tritt B-Human bei den internationalen Wettbewerben der RoboCup Federation in der Standard Platform League (SPL) an, in welcher Roboter gegeneinander Fußball spielen. Das Team gewann siebenmal die RoboCup German Open, fünfmal den RoboCup und einmal die RoboCup European Open und ist somit das erfolgreichste Team der SPL. [1]

## 1.1 Motivation

In der SPL finden neben dem regulären Fußballwettbewerb jährlich auch sogenannte Technical Challenges statt. Dies sind Wettbewerbe mit speziellen Herausforderungen, in denen die teilnehmenden Teams einzeln antreten. Im Jahr 2015 war einer dieser Zusatzwettbewerbe die *Realistic Ball Challenge* (RBC), bei der die Aufgabe darin bestand, mit einem Roboter statt des in den Regeln des Fußballwettbewerbs vorgeschriebenen Balls mit definierter Farbe und Größe (siehe [2]) fünf beliebige Bälle auf dem Feld zu finden und in die Tore zu schießen (vgl. [3]). Hierbei befanden sich neben den Bällen und dem spielenden Roboter zusätzlich noch zwei weitere, stehende Roboter auf dem Feld, welche entsprechend der Regeln der SPL nicht umgeworfen werden durften.

## 1.2 Ziel und Aufbau der Arbeit

Ziel dieser Arbeit ist die Entwicklung eines Bildverarbeitungsmoduls, welches beliebige Bälle auf dem Feld der SPL erkennt und somit in der RBC zur Lokalisierung der zu spielenden Bälle genutzt werden kann.

Zum Verständnis der Arbeit wird im nächsten Kapitel 2 zunächst näher auf den RoboCup, die SPL und insbesondere auf die RBC sowie auf die in dieser Arbeit verwendete Hardwa-

replattform, den Roboter *NAO*, eingegangen. Hierauf folgt ein Kapitel 3, in dem bestehende Ansätze zur Erkennung von Bällen beziehungsweise Kreisen sowohl im RoboCup als auch in anderen Kontexten vorgestellt und für die Nutzung im vorliegenden Anwendungsfall diskutiert werden.

In den Kapiteln 4 und 5 wird dann nach Erläuterung von für die Implementierung relevanten Grundlagen der Mathematik und Bildverarbeitung die Implementierung der Erkennung und Selektion des zu spielenden Balls beschrieben und anschließend in Kapitel 6 hinsichtlich der Zielsetzung der Arbeit evaluiert.

# Kapitel 2

# Hintergrund

In diesem Kapitel wird ein kurzer Überblick über den Kontext der Arbeit im RoboCup gegeben. Insbesondere erläutert es die in der Standard Platform League 2015 ausgetragene Realistic Ball Challenge sowie den in der SPL verwendeten Roboter *NAO*.

## 2.1 RoboCup

Der RoboCup ist ein internationaler Forschungswettbewerb zur Förderung der Forschung in den Bereichen Robotik und künstliche Intelligenz, welcher 1993 ins Leben gerufen wurde und 1997 zum ersten Mal stattfand. [4] Er dient der Forschung in den genannten Bereichen als Standardproblem, in welchem Theorien, Algorithmen und Architekturen evaluiert werden können und hat dabei, um die Forschung voranzutreiben, das Ziel, im Jahr 2050 ein Team aus vollständig autonomen humanoiden Robotern vorweisen zu können, welches in einem Fußballspiel mit den offiziellen Regeln der FIFA gegen den dann amtierenden menschlichen Fußballweltmeister gewinnt. [5]

Der RoboCup findet jährlich mit Wettbewerben in mehreren Ligen statt, in denen mit jeweils eigenen Regeln gespielt wird. Die SPL ist dabei eine Liga, in welcher die verwendeten Roboter vorgeschrieben sind, welche von den Teilnehmern fertig gekauft werden und nicht modifiziert werden dürfen, sodass es sich bei den Turnieren in dieser Liga um reine Softwarewettbewerbe handelt. In jeder Liga gibt es ein Technical Commitee, das die jährlich erfolgenden Regeländerungen sowie etwaige Technical Challenges festlegt, in denen einzelne Probleme des Fußballs erprobt und für zukünftige Regeländerungen evaluiert werden. Hierdurch sollen sich die Regeln der einzelnen Ligen schrittweise an die Regeln der FIFA annähern.

Im Jahr 2015 spielten in der SPL je fünf Roboter der Marke *NAO* der Firma Aldebaran Robotics in zwei Teams gegeneinander. Die Spiele fanden auf einem grünen Feld der Größe 9 m $\times$ 6 m mit weißen Toren und Feldlinien statt, wobei zusätzlich zu den das Feld umrandenden Linien analog zu den Fußballregeln der FIFA ein Mittelkreis, Strafstoßpunkte und ein Strafraum existierten.

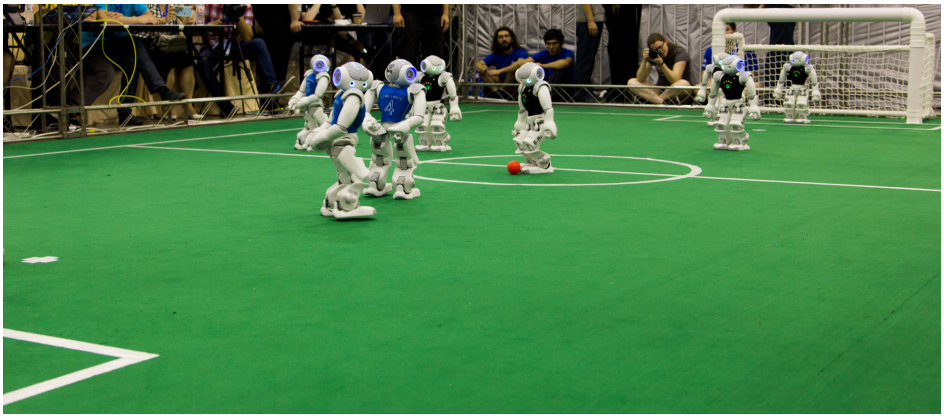

**Abbildung 2.1** Ein Fussballspiel in der Standard Platform League beim RoboCup 2015.

Die Spieler trugen von den jeweiligen Teams gestaltete einheitliche Trikots, welche jeweils zu mindestens 80% eine nicht weiße, orangene, rote, grüne oder hellgraue Grundfarbe hatten. Auch der verwendete Ball war fest definiert als ein „Mylec orange street hockey ball“, welcher einfarbig orange war und eine feste Größe von 65 mm sowie ein festes Gewicht von 55 g hatte. [2]
Eine Momentaufnahme eines Spiels in der SPL während des RoboCup 2015 ist in Abbildung 2.1 zu sehen.

## 2.2 Realistic Ball Challenge

In der Realistic Ball Challenge, welche beim RoboCup 2015 stattfand und die Motivation dieser Arbeit bildet, wurden fünf beliebige, von Teams mitgebrachte und vom Technical Committee der SPL ausgewählte Bälle sowie zwei Roboter mit Trikots beliebiger Farbe innerhalb eines Quadrates von 4m Seitenlänge um den Mittelpunkt des Felds platziert.

Nacheinander wurde dann jeweils ein Roboter der antretenden Teams an einem der Schnittpunkte der Mittellinie mit den Seitenlinien platziert und hatte drei Minuten Zeit, um die Bälle in Richtung der Tore zu schießen. Punkte wurden dabei für die absichtliche Bewegung eines Balls in Richtung eines der Tore sowie für das tatsächliche Erzielen eines Tors vergeben. Hätte ein Team es geschafft, sämtliche Bälle in die Tore zu schießen, so hätte es zusätzliche Punkte abhängig von der übrigen Zeit erhalten. Um gleiche Voraussetzungen für alle Teilnehmer zu gewährleisten, wurden die Bälle und Roboter für jedes Team an denselben Positionen platziert. [3]

Zweck der RBC war, zu evaluieren, ob in einem Spiel der SPL beliebige und insbesondere Fußbällen der FIFA ähnliche Bälle von Teams erkannt werden können. Als Ergebnis der Challenge wurde in den Regeln des Jahres 2016 der orangene Ball durch einen größeren Ball mit einer schwarz-weißen Textur ersetzt. [6]

## 2.3 NAO

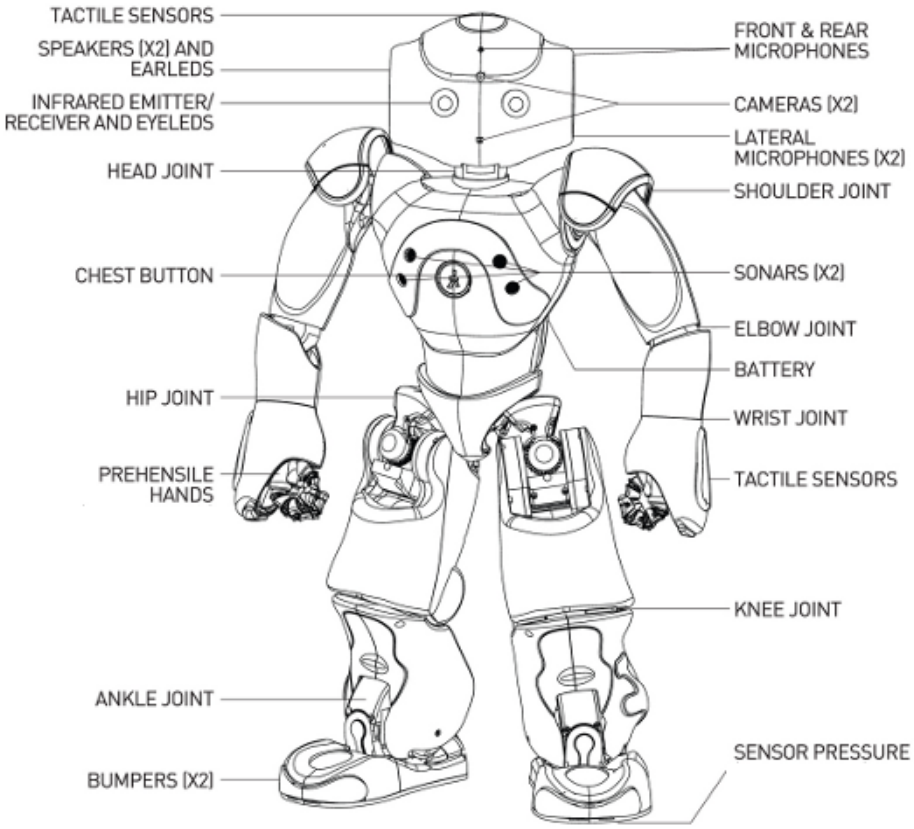


**Abbildung 2.2** Sensoren und Gelenke des *NAO*. (Abbildung aus [10])

Grundlage dieser Arbeit ist die aktuelle fünfte Version des in der SPL verwendeten humanoiden Roboters *NAO*, welche 574mm groß ist und 5,4kg wiegt. [7] Er verfügt über 26 Gelenke, die bis auf die beiden Hüftgelenke, welche von einem einzigen Motor angesteuert werden, über jeweils einen zugehörigen Motor verfügen und somit unabhängig voneinander bewegt werden können. [8] Die aktuelle Ausrichtung aller Gelenke wird dabei von Sensoren erkannt und kann von Software ausgelesen werden. Einen Überblick über die Komponenten des *NAO* gibt Abbildung 2.2.

Über zahlreiche weitere Sensoren kann der *NAO* seine Umgebung wahrnehmen. Dies sind einerseits zwei im Kopf montierte Kameras, welche jeweils bis zu 30 Bilder pro Sekunde mit einer maximalen Auflösung von $1280 \times 960$ Pixeln im Format YUV422 liefern. Dabei beträgt der Öffnungswinkel der Kameras horizontal 60,9° und vertikal 47,6°. [9] Außerdem befinden sich in der Brust des *NAO*s ein Gyroskop, ein Beschleunigungsmesser sowie Ultraschallsensoren, mit welchen er Entfernungen zu bis zu 80 cm entfernten Objekten messen kann. Weiterhin sind im Kopf vier Mikrophone und unter den Füßen jeweils vier Drucksensoren eingebaut. [10]

Die den *NAO* steuernde Software läuft auf einem im Kopf verbauten Embedded-PC mit einer 1,6GHz schnellen Intel Atom-CPU und einem Gigabyte Arbeitsspeicher. [11]

# Kapitel 3

# Verwandte Arbeiten

Dieses Kapitel gibt einen kurzen Überblick über bestehende Ansätze zur Erkennung von Bällen in Bildern.

## 3.1 Bisherige Ballerkennung im B-Human-System

Im bisherigen System des Teams B-Human werden Bälle in einem zweistufigen Prozess gefunden. Zunächst werden in festgelegtem horizontalen Abstand jeweils vertikale Bildregionen anhand des im B-Human-System vorher berechneten Rasters basierend auf ihrer Farbe gebildet und orangene Regionen als sogenannte *BallSpots* gespeichert. [12] Diese werden dann jeweils als Bälle validiert, indem zunächst zu kleine Regionen sowie solche, welche ins Feld projiziert zu weit entfernt wären, ausgefiltert werden. Daraufhin werden je *BallSpot* weitere zum Ball gehörende Punkte gesucht, indem umliegende Punkte mit geringer Abweichung der U- und V-Komponenten im Bild zur Region hinzugefügt werden, sofern ihre Entfernung den erwarteten Ballradius nicht zu sehr übersteigt. Nachdem dann der Mittelpunkt und der Radius des Balls im Bild bestimmt wurden, wird geprüft, ob in der Umgebung des Balls keine Punkte ähnlicher Farbe sind sowie ob der Ball innerhalb der zuvor im B-Human-System berechneten Feldgrenze im Bild liegt und ob er nicht zu viele Pixel enthält, welche als Bestandteil eines roten oder magentafarbenen Trikots erkannt werden könnten. [13] Da dieses Verfahren nur aufgrund der bekannten Farbe und Größe des Balls funktioniert, ist es für den Fall beliebiger Bälle nicht anwendbar.

Im Jahr 2009 fand bereits eine der RBC ähnliche Technical Challenge in der SPL mit dem Name *Any Ball Challenge* statt, in welcher ebenfalls fünf beliebige Bälle in Richtung der Tore geschossen werden sollten. Der damals im Team B-Human verfolgte Ansatz zur Erkennung der Bälle sah vor, runde Regionen im Bild zu finden, welche nicht als grün, gelb oder blau klassifiziert wurden und diese als Bälle anzunehmen. [14] Hierbei wird nicht zwischen Bällen und Robotern unterschieden, da die *Any Ball Challenge* nicht vorsah, dass Roboter auf dem Feld standen. Daher kann dieses Verfahren nicht direkt für die RBC übernommen werden.

## 3.2 Verwandte Arbeiten zur farbunabhängigen Ballerkennung

Hanek et al. stellten 2002 in [15] den sogenannten Contracting Curve Density-Algorithmus zur Erkennung von beliebigen Bällen im Kontext des RoboCup vor. Hierzu wird ausgehend von einer Schätzung der aktuellen Ballposition das Modell der Ballkurve basierend auf lokalen Statistiken im RGB-Bild iterativ angepasst auf die tatsächliche Position des Balls im Bild. Weil dieses Verfahren stets nur einen Ball im Bild annimmt, ist zu erwarten, dass bei Bildern mit mehreren Bällen nur ein einziger Ball gesehen wird, sodass entweder nicht alle Bälle gefunden werden können oder nicht der dem Roboter nächste Ball gespielt wird. Im Kontext der zeitlich beschränkten RBC wäre dies ein Nachteil.

In [16] stellten Coath und Musumeci 2003 einen Algorithmus zur Erkennung von Bällen basierend auf Kanteninformationen vor. Dieser verfolgt die Konturen von Kanten im Bild, um Kreisbögen zu bilden und findet Bälle, indem jeweils die Schnittpunkte der Linien durch die Kreismittelpunkte adjazenter Kreisbögen bestimmt werden und solche Punkte, an denen sich diese Schnittpunkte häufen, als Kreismittelpunkte von Bällen im Bild angesehen werden.

Martins et al. evaluierten 2008 im Kontext der Middle Size League des RoboCup in [17] verschiedene Algorithmen zur Kantendetektion in Bildern, nämlich den Sobel-, Laplace- und Canny-Operator (vgl. Kapitel 4.3), und wandten auf das von diesen gebildete Kantenbild eine Hough-Transformation für Kreise (vgl. Kapitel 4.4.3) an, um beliebige Bälle in Kamerabildern von omnidirektionalen Kameras zu erkennen. Zur Verminderung von Falscherkennungen schlossen sie dabei Kanten, welche auf die Ebene des Felds projiziert zu weit vom Roboter entfernt wären, aus und evaluierten gefundene Bälle mithilfe der im Ball gefundenen Farben.

# Kapitel 4

# Grundlagen

In diesem Kapitel werden einige Grundlagen der Mathematik und Bildverarbeitung beschrieben, welche in den nachfolgenden Kapiteln vorausgesetzt werden.

## 4.1 Lineare Faltung und lineare Filter

Die mathematische Operation der linearen Faltung mit dem Operator $*$ beschreibt die Verknüpfung zweier Funktionen gleicher Dimensionalität und ist für zweidimensionale, diskrete Funktionen $I, H$ wie in Gleichung 4.1 gezeigt definiert. Die Faltungsoperation ist kommutativ, linear und assoziativ. [18]

$$I' = I * H \Leftrightarrow \forall u, v \colon I'(u, v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u - i, v - j) \cdot H(i, j) \tag{4.1}$$

Lineare Filter sind in der Bildverarbeitung Operationen, welche eine zweidimensionale Bildfunktion $I$ mit einer diskreten, die Operation definierenden Filtermatrix $H$ falten. Hierbei wird für die Faltung davon ausgegangen, dass Komponenten außerhalb des Definitionsbereichs der Filtermatrix null sind, sodass die Filterregion $R$, welche die Indizes aller Komponenten der Filtermatrix enthält, die nicht null sind, für jeden linearen Filter definiert ist und die Anwendung eines linearen Filters bei horizontal und vertikal gespiegelter Filtermatrix wie folgt beschrieben werden kann. [18]

$$I' = I * H \Leftrightarrow \forall u, v \colon I'(u, v) = \sum_{(i,j) \in R} I(u + i, v + j) \cdot H(i, j) \tag{4.2}$$

## 4.2 Glättungsfilter

Glättungs- oder Weichzeichnungsfilter sind lineare Filter, die zum Ziel haben, das gegebene Bild weichzuzeichnen und damit Bildrauschen zu entfernen.

### 4.2.1 Box-Filter

Ein einfacher Glättungsfilter ist der Box-Filter, welcher den Farbwert eines Pixels als den Mittelwert der Farbwerte aller Pixel innerhalb eines bestimmten quadratischen Bereichs um ihn herum bestimmt. Für einen Bereich von $3 \times 3$ Pixel besitzt er damit die Filtermatrix $H^B$.

$$H^B = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{4.3}$$

### 4.2.2 Gauß-Filter

Ein weiterer, in der Literatur häufig genutzter Glättungsfilter ist der Gauß-Filter. Dieser glättet das Bild mit einer Filtermatrix, deren Einträge der Diskretisierung der zweidimensionalen Gauß-Funktion $g(\sigma, x, y)$ (vgl. Gleichung 4.4) entsprechen, allerdings normalisiert sind, sodass die Summe der Einträge der Matrix 1 ist. Die Glättung durch diesen Filter wird beeinflusst von der gewählten Größe der Filtermatrix sowie durch den gewählten Wert für die Standardabweichung $\sigma$.

Ein Beispiel für eine Filtermatrix der Größe $3 \times 3$ mit $\sigma = 1$ ist $H_1^G$ in Gleichung 4.5, deren Einträge auf sieben Nachkommastellen gerundet wurden.

$$g(\sigma, x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.4}$$

$$H_1^G = \begin{pmatrix} 0{,}0751136 & 0{,}1238414 & 0{,}0751136 \\ 0{,}1238414 & 0{,}20418 & 0{,}1238414 \\ 0{,}0751136 & 0{,}1238414 & 0{,}0751136 \end{pmatrix} \tag{4.5}$$

## 4.3 Kantenerkennung

Um Kanten in Graustufenbildern zu finden, wird der Gradientenvektor $\nabla I(u, v)$ an jeder Stelle $(u, v)$ des Bildes $I$ gebildet. Dessen Komponenten sind die partiellen Ableitungen $I_x$ und $I_y$ der Bildfunktion $I$ entlang der $x$- beziehungsweise $y$-Koordinate des Bildes. [18]

$$\nabla I(u, v) = \begin{pmatrix} I_x \\ I_y \end{pmatrix} = \begin{pmatrix} \frac{\partial I}{\partial x}(u, v) \\ \frac{\partial I}{\partial y}(u, v) \end{pmatrix} \tag{4.6}$$

Die Stärke $E$ der Kante an der Stelle $(u, v)$ lässt sich dann durch die Länge des Gradientenvektors und die Richtung $\Phi$ dieser Kante durch seine Richtung bestimmen. $\Phi$ zeigt dabei in Richtung der Normalen einer Geraden durch die Kante.

$$E(u,v) = \sqrt{I_x^2(u,v) + I_y^2(u,v)} \tag{4.7}$$

$$\Phi(u,v) = \text{atan2}(I_y(u,v), I_x(u,v)) \tag{4.8}$$

Die atan2-Funktion ist dabei entsprechend der Definition in [19] festgelegt:

$$\text{atan2}(y,x) := \begin{cases} \arctan\frac{y}{x}, \text{ falls } y \geq 0 \\ \arctan\frac{y}{x} + 180°, \text{ falls } y < 0 \\ 0°, \text{ falls } x = y = 0 \\ 90°, \text{ falls } x = 0 \wedge y > 0 \\ -90°, \text{ falls } x = 0 \wedge y < 0 \end{cases} \tag{4.9}$$

Das zweikanalige Bild $\nabla_I$ mit $\nabla_I(u,v) = (E(u,v), \Phi(u,v))$, welches als Komponenten also Stärke und Richtung der Gradientenvektoren des Bilds $I$ hat, heißt Gradientenkarte. [19]

### 4.3.1 Bestimmung der partiellen Ableitungen

Um die für die Ermittlung der Gradientenvektoren nötigen partiellen Ableitungen der Bildfunktionen zu erhalten, werden diese analog zur Annäherung der Ableitung einer kontinuierlichen Funktion durch die Steigungen der Sekanten durch die beiden zur betrachteten Stelle in Richtung der Ableitung adjazenten Pixel angenähert. Dies ist notwendig, da für die Bildfunktion als diskrete Funktion keine Ableitung definiert ist. [18]

$$\frac{\partial I}{\partial x}(u,v) = \frac{I(u+1,v) - I(u-1,v)}{u+1-u-1} = \frac{I(u+1,v) - I(u-1,v)}{2} \tag{4.10}$$

$$\frac{\partial I}{\partial y}(u,v) = \frac{I(u,v+1) - I(u,v-1)}{v+1-v-1} = \frac{I(u,v+1) - I(u,v-1)}{2} \tag{4.11}$$

### 4.3.2 Lineare Filter zur Kantenerkennung

Die Bestimmung der partiellen Ableitungen der Bildfunktion – wie im vorherigen Abschnitt genannt – kann durch lineare Filter erfolgen. Eine naheliegende Operation, welche genau die Steigungen der Sekanten ermittelt, nutzt dabei die Filtermatrizen $H_x^D$ und $H_y^D$ zur Berechnung von $I_x$ beziehungsweise $I_y$. [18]

$$H_x^D = \begin{pmatrix} -0\,5 & 0 & 0\,5 \end{pmatrix} \tag{4.12}$$

$$H_y^D = \begin{pmatrix} -0\,5 \\ 0 \\ 0\,5 \end{pmatrix} \tag{4.13}$$

Übliche Verfahren nutzen jedoch Filter, welche zur Verringerung von erkannten Kanten in Bildrauschen nicht nur die Steigung an jeweils einer Stelle betrachten, sondern die jeweils umliegenden Stellen ebenfalls berücksichtigen. Klassische Verfahren sind hierbei der Prewitt-Operator mit den Filtermatrizen $H_x^P$ und $H_y^P$ und der Sobel-Operator mit den Filtermatrizen $H_x^S$ und $H_y^S$. [18]

$$H_x^P = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \tag{4.14}$$

$$H_y^P = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \tag{4.15}$$

$$H_x^S = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{4.16}$$

$$H_y^S = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \tag{4.17}$$

### 4.3.3 Canny-Operator

Canny wählte 1986 in [20] den Ansatz, Kriterien für eine gute Kantenerkennung mathematisch zu formulieren, sodass die Güte von Verfahren zur Kantenerkennung messbar ist. Diese Kriterien sind eine gute Detektion und eine gute Lokalisierung von Kanten. Hierfür stellte er ein Verfahren vor, welches diese Kriterien möglichst gut erfüllt sowie nur eine Antwort je Kante, also nur ein Pixel breite Kanten, liefert.

Hierzu schlägt er vor, die Gradientenkarte des Bildes durch dessen Faltung mit Filtermatrizen $H_x^{G'}, H_y^{G'}$ zu berechnen, die der Diskretisierung der partiellen ersten Ableitungen $g'_x$ und $g'_y$ der zweidimensionalen Gauß-Funktion (siehe Gleichungen 4.18, 4.19) entsprechen, und potentielle Kantenpunkte als Nullübergänge im Ergebnis der Faltung des Bildes mit der partiellen zweiten Ableitung $g''$ dieser Gauß-Funktion (siehe Gleichung 4.20) zu erhalten, um nur solche Kantenpunkte in Betracht zu ziehen, die an lokalen Maxima der Gradientenlängen liegen.

Ausgehend von diesen potentiellen Kantenpunkten werden dann mittels zwei Hysterese-Schwellwerten solche Punkte als Kantenpunkte markiert, die entweder bereits potentielle Kantenpunkte sind und deren Gradientenvektoren eine größere Länge als der höhere Schwellwert haben oder die eine bestehende markierte Kante entsprechend ihrer Gradientenrichtung fortführen würden und deren Gradientenvektor eine größere Länge als der niedrigere Schwellwert haben.

$$g'_x(\sigma, x, y) = \frac{\partial}{\partial x} g(\sigma, x, y) = -\frac{x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.18}$$

$$g'_y(\sigma, x, y) = \frac{\partial}{\partial y} g(\sigma, x, y) = -\frac{y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.19}$$

$$g''(\sigma, x, y) = \frac{\partial^2}{\partial x^2} g(\sigma, x, y) + \frac{\partial^2}{\partial y^2} g(\sigma, x, y) = \left( \frac{x^2+y^2}{2\sigma^2} - 1 \right) \frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{\pi\sigma^4} \tag{4.20}$$

Da die Faltungsoperation assoziativ ist und lineare Filter daher separierbar sind, lassen sich die Faltungen mit $H_x^{G'}$ beziehungsweise $H_y^{G'}$ als Faltungen des Bildes mit einem Gauß-Filter $H^G$ und anschließenden Faltungen mit den partiellen Ableitungsfiltern $H_x^D$ beziehungsweise $H_y^D$ durchführen.

Außerhalb der Arbeit von Canny wurden verschiedene Verfahren etabliert, die ebenfalls Canny-Operatoren genannt werden, aber lediglich eine Annäherung des Verfahrens von Canny darstellen. Ein solches Verfahren verwendet nach einer Glättung des Bildes durch Faltung mit einem Gauß-Filter den Sobel-Operator zur Berechnung der Gradientenkarte, aus welcher daraufhin mittels einer Nicht-Maximum-Unterdrückung potentielle Kantenpunkte gefunden werden (siehe [19]).

Dieses Verfahren liefert zwar lediglich eine Annäherung an die von Canny angestrebte optimale Kantendetektion, allerdings ist seine Anwendung schneller, da zur Anwendung der Sobel-Filter im Gegensatz zu den Ableitungen der Gauß-Funktion lediglich Rechenoperationen mit Ganzzahlen notwendig sind und die für Cannys Verfahren mit dem Bild zu faltenden Filtermatrizen je nach gewünschter Genauigkeit der Diskretisierung größer als die $3 \times 3$-Matrizen der Sobel-Filter sind.

Ein Vergleich des vom Canny-Operator gelieferten Bildes mit den Impulsantworten der Prewitt- und Sobel-Filter ist in Abbildung 4.1 zu sehen.

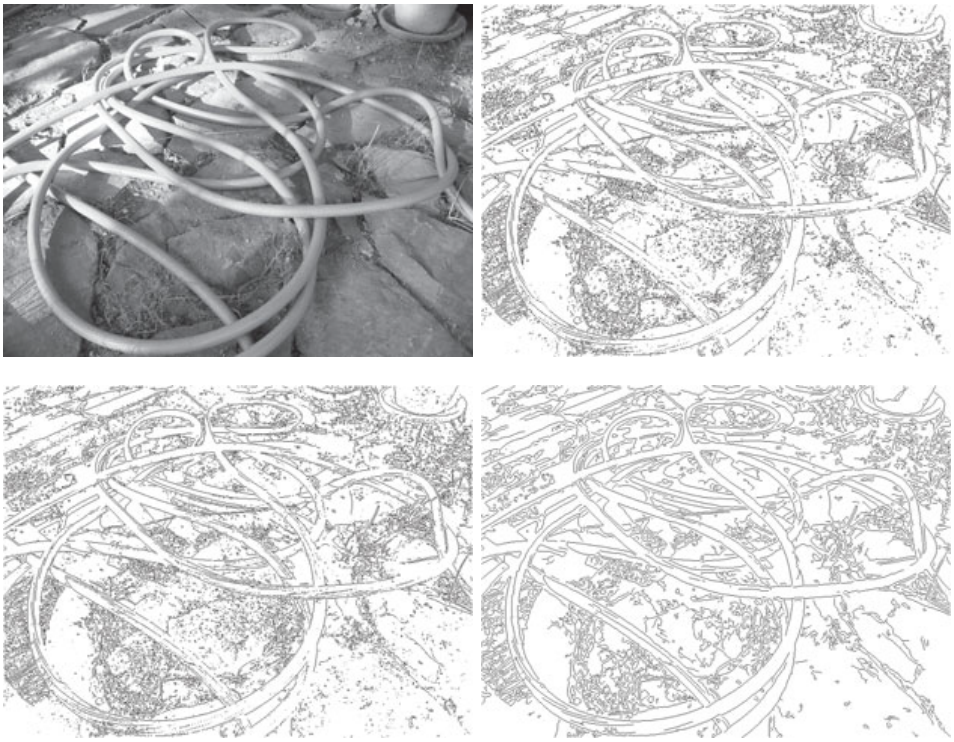

**Abbildung 4.1** Vergleich der beschriebenen Verfahren zur Kantenerkennung. (Abbildungen aus [18])
*Obere Reihe:* Ausgangs-Graustufenbild und mit Prewitt-Filter gefaltetes Bild.
*Untere Reihe:* Mit Sobel-Filter gefaltetes Bild und vom Canny-Operator berechnetes Kantenbild.

## 4.4 Kreiserkennung

Um Kreise in Konturbildern zu erkennen, welche lediglich Pixel enthalten, die mithilfe eines Verfahrens zur Kantenerkennung als Kantenpixel klassifiziert wurden, existieren mehrere Verfahren, von denen insbesondere die Hough-Transformation häufig angewendet wird (vgl. [19]).

Mathematisch ist ein Kreis über seinen Mittelpunkt $(x_m, y_m)$ und Radius $r$ definiert, welche durch die Kreiserkennung bestimmt werden müssen. Punkte $(x, y)$, welche auf dem Kreis liegen, erfüllen dabei die in Gleichung 4.21 gezeigte Standardkreisgleichung (vgl. [19]).

$$(x - x_m)^2 + (y - y_m)^2 = r^2 \tag{4.21}$$

### 4.4.1 Berechnung eines Kreises aus drei Punkten

Sind drei nicht kolineare Punkte P, Q, R gegeben, so lässt sich zeigen, dass es stets einen Kreis gibt, auf dem diese drei Punkte liegen.

Hierbei ist der Mittelpunkt $(x_m, y_m)$ des Kreises – wie in Abbildung 4.2 zu sehen – der Schnittpunkt der Mittelsenkrechten der Strecken $\overline{PQ}$ und $\overline{QR}$, während der Radius entsprechend der Definition eines Kreises die Länge der Strecke vom Mittelpunkt zu jedem der Punkte

$\mathrm{P} = (x_1, y_1), \mathrm{Q} = (x_2, y_2), \mathrm{R} = (x_3, y_3)$ ist (vgl. [21]).

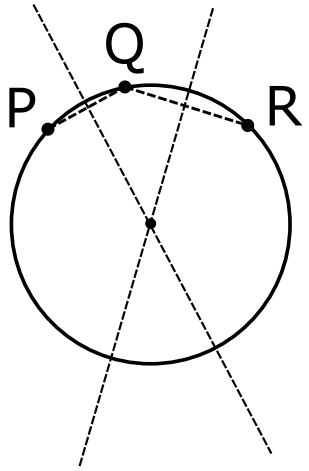


**Abbildung 4.2** Berechnung des Kreises durch drei nicht kolineare Punkte.

Um den Mittelpunkt des Kreises zu finden, kann die in [22] beschriebene Methode angewandt werden. Hierzu werden zunächst die Gleichungen von $\overline{PQ}$ und $\overline{QR}$ abhängig von $x_m$ aufgestellt:

$$y_a = m_a(x_m - x_1) + y_1 \tag{4.22}$$

$$y_b = m_b(x_m - x_2) + y_2 \tag{4.23}$$

Deren Steigungen $m_a = \frac{y_2 - y_1}{x_2 - x_1}, m_b = \frac{y_3 - y_2}{x_3 - x_2}$ sind dabei durch die Definition der Strecken bekannt. Die Gleichungen der Mittelsenkrechten sind dann:

$$y'_a = -\frac{1}{m_a}\left(x_m - \frac{x_1 + x_2}{2}\right) + \frac{y_1 + y_2}{2} \tag{4.24}$$

$$y'_b = -\frac{1}{m_b}\left(x_m - \frac{x_2 + x_3}{2}\right) + \frac{y_2 + y_3}{2} \tag{4.25}$$

Somit kann der Mittelpunkt des Kreises – wie in Gleichung 4.26 gezeigt – bestimmt werden, indem die Gleichungen der Mittelsenkrechten nach $x_m$ aufgelöst werden. $y_m$ wird dann durch Einsetzen von $x_m$ in die Gleichung einer der Mittelsenkrechten erhalten.

$$x_m = \frac{m_a m_b(y_1 - y_3) + m_b(x_1 + x_2) - m_a(x_2 + x_3)}{2(m_b - m_a)} \tag{4.26}$$

### 4.4.2 Kreisfindung durch Ausgleichsrechnung

Eine einfache Methode zur Findung von Kreisen in Bildern ist die Minimierung der in Gleichung 4.27 gezeigten Summe der quadrierten Abstände aller $n$ gefundenen Konturpunkte $(u_i, v_i)$ zum zu findenden Kreis. Hierzu wird jedem Konturpunkt eine initial auf 1 gesetzte Gewichtung $w_i$ gegeben. Daraufhin wird iterativ immer wieder $\epsilon^2$ berechnet und die Gewichtung aller Konturpunkte so angepasst, dass $\epsilon$ minimiert wird (vgl. [23]). Eine Vereinfachung

dieses Verfahrens besteht darin, lediglich die Werte 1 und 0 für Gewichte zu verwenden, also iterativ Konturpunkte auszuschließen, bis ein lokales Minimum für $\epsilon$ gefunden ist.

$$\epsilon^2 = \sum_{i=1}^{n} w_i \left( \sqrt{(u_i - x_m)^2 + (v_i - y_m)^2} - r \right)^2 \tag{4.27}$$

Dieses Verfahren ist sehr aufwändig, weil sämtliche Konturpunkte betrachtet werden und für komplexe Bilder viele Iterationen zum Ausschluss irrelevanter Punkte notwendig sind. Außerdem wird in Bildern mit mehreren Kreisen nur einer gefunden, sofern nicht durch vorherige Segmentierung der Konturpunkte diese in Untermengen aufgeteilt wurden, die zu jeweils einem potentiellen Kreis gehören. Daher wird es in dieser Arbeit nicht verwendet.

### 4.4.3 Hough-Transformation für Kreise

Im Gegensatz zu Verfahren, die wie die Ausgleichsrechnung nach einzelnen geometrischen Formen in gegebenen Bildern suchen, stellt die Hough-Transformation eine vollständige Suche da, die Evidenzen für jedes mögliche Vorkommen einer Form im Bild findet. Hierzu wird ein Raum für alle möglichen Parametrisierungen einer die jeweilige Form beschreibenden Gleichung erstellt. Die Einträge dieses Raums sind Akkumulatoren, deren Wert aussagt, wie viele Evidenzen für eine Form mit den jeweiligen Parametern im Bild gefunden wurden (vgl. [19]).

Im Fall der Hough-Transformation für Kreise ist der Parameterraum dreidimensional, da die Standardkreisgleichung die drei Parameter $x_m, y_m, r$ enthält. Jeder Konturpunkt im Bild stellt eine Evidenz für all diejenigen Kreise dar, von deren Mittelpunkt er genau ihren Radius entfernt liegt. Dies bedeutet, dass für jeden Konturpunkt für jede mögliche Belegung von $r_m$ in der entsprechenden Ebene im Parameterraum alle Einträge erhöht werden, die auf einem Kreis des jeweiligen Radius $r_m$ um den Punkt liegen (vgl. Abbildung 4.3).

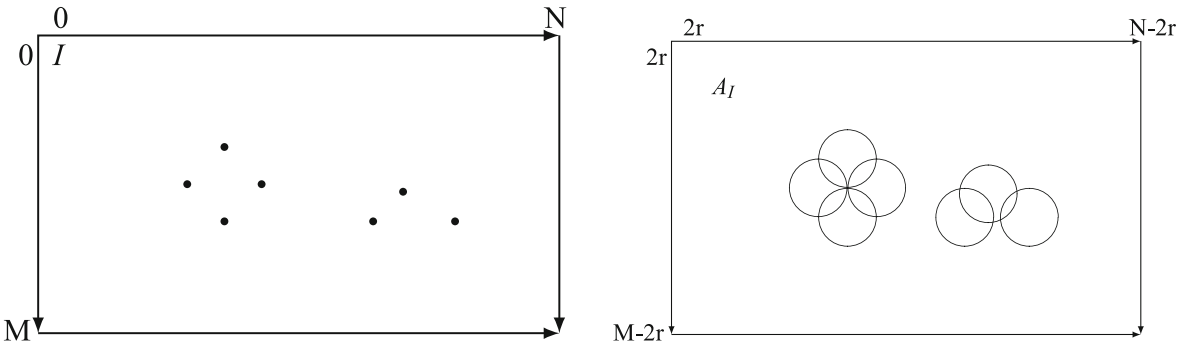


**Abbildung 4.3** Ein Konturbild mit sieben Punkten (links) und eine Ebene des zugehörigen Parameterraums mit einem bestimmten $r_m$ (Abbildungen aus [19]).

Der so erstellte Parameterraum sagt durch die Akkumulation der Evidenzen nun aus, welche Kreise im Bild mit jeweils einer bestimmten relativen Wahrscheinlichkeit existieren. Um daraus eine Aussage über tatsächlich vorhandene Kreise treffen zu können, wird auf den Parameterraum hinterher in der Regel eine Nicht-Maximum-Unterdrückung angewendet, wo-

raufhin mithilfe eines Schwellwerts für die Akkumulatoreinträge tatsächliche Kreise im Bild identifiziert werden.

Die Laufzeit der an sich aufwendigen Hough-Transformation für Kreise lässt sich noch durch verschiedene Variationen optimieren, etwa indem nicht nur ein Konturbild, sondern die Gradientenkarte des Bilds betrachtet und nur diejenigen Akkumulatoren erhöht werden, die in Richtung des Gradientenvektors am jeweiligen Konturpunkt liegen. Andere Variationen wie das in dieser Arbeit genutzte Verfahren betrachten nur eine Teilmenge der Konturpunkte zur Akkumulation von Kreiskandidaten.

## 4.5 Transformation von Koordinaten zwischen Bild- und Feldebene

Zur Transformation von Koordinaten eines Punktes im Bild zu zum *NAO* relativen Koordinaten auf der Ebene des Spielfelds oder einer dazu parallelen Ebene werden in dieser Arbeit die im B-Human-System implementierten Funktionen verwendet.

Diese nutzen zur Projektion das Lochkameramodell, welches von einer unendlich kleinen Kameralinse ausgeht. Dadurch verlaufen alle Lichtstrahlen, welche die Bildebene treffen, durch einen Punkt und der Strahlensatz kann verwendet werden, um die Gerade durch einen Punkt auf der Bildebene und alle möglichen dazugehörigen Punkte im dreidimensionalen Raum zu bestimmen. Schneidet man diese Gerade mit der Feldebene oder einer dazu parallelen Ebene, erhält man die gesuchten Koordinaten relativ zur Position der Kamera. Um die Koordinaten relativ zum Ursprung im Mittelpunkt der Füße des Roboters zu erhalten, werden die inverse Rotation und inverse Translation der Kamera auf die erhaltenen Koordinaten angewandt. Rotation und Translation der Kamera sind dabei im B-Human-System durch Folgen der kinematischen Kette aller Gelenkwinkel vom Koordinatenursprung bis zur Kamera stets bekannt.

Die Projektionsfunktionen des B-Human-Systems sind detaillierter unter anderem in [24] beschrieben.

## 4.6 B-Human Framework

Der im Kontext dieser Arbeit geschriebene Programmcode ist eingebettet in die B-Human-Framework genannte Softwarearchitektur des Teams B-Human, auf welche hier kurz eingegangen wird.

Das B-Human Framework stellt eine Rahmenarchitektur für die Programmierung von Robotern mit limitierten Ressourcen dar. Es ist darauf ausgelegt, durch Verzicht auf komplexe externe Bibliotheken und Verwendung eines einheitlichen Compilers auf beliebigen Betriebssystemen zu funktionieren. Im Framework laufen parallel drei Prozesse, nämlich *Cognition*,

welcher auf aufgenommene Kamerabilder reagiert, *Motion*, welcher auf gelieferte Daten der Gelenke reagiert und neue Gelenkwinkel setzt und *Debug*, welcher zu Zwecken der Fehlersuche und als Entwicklungshilfe eine Netzwerkverbindung zu einem externen Computer aufbauen kann. Die Funktionalität der Prozesse liegt in sogenannten Modulen, welche in Form von einzelnen C++-Klassen vorliegen und Daten durch sogenannte Repräsentationen, die ebenfalls durch C++-Strukturen oder -Klassen definiert sind, austauschen. Da die Schnittstellen der Module so über die Repräsentationen fest definiert sind, lassen sich die Module leicht austauschen.

In [25] ist das B-Human Framework inklusive eines Vergleichs zum populären Robot Operating System genauer beschrieben.

Im Kontext dieser Arbeit wurde – wie im nächsten Kapitel 5 beschrieben – ein einziges im Prozess *Cognition* laufendes Modul entwickelt.

# Kapitel 5

# Implementierung

In diesem Kapitel werden die für die RBC implementierten Programmteile beschrieben.

Wie in Abschnitt 4.6 genannt, bilden Module und Repräsentationen die Grundlage der Software des Teams B-Human. Für die RBC wurde, um möglichst viel des bestehenden Systems nutzen zu können, beschlossen, ein Modul zu schreiben, welches die bestehende Repräsentation *BallPercept* füllen soll. Diese enthält die Position des im letzten Bild gesehenen Balls und wird im B-Human-System verwendet, um die Position des Balls relativ zum Roboter zu modellieren.

Dies bedeutet, dass wie im normalen Spiel in der Challenge stets nur ein einzelner Ball gespielt werden kann. Da allerdings mehrere Bälle vorhanden sind, wird, damit der Roboter nicht ständig zwischen mehreren zu spielenden Bällen wechselt, vom Modul stets ein bestimmter gesehener Ball zur Füllung der Repräsentation selektiert.

Die im Folgenden zugrunde gelegte Idee zur Erkennung von Bällen ist, ähnlich dem von Martins et al. gewählten Ansatz (vgl. [17] beziehungsweise Abschnitt 3.2) aus dem Kamerabild zunächst ein Konturbild zu erstellen, um anschließend in diesem Kreise zu finden und als gesehene Bälle anzunehmen, sofern sie bestimmte Kriterien erfüllen.

Die vorliegende Implementierung entspricht dabei nicht exakt der 2015 in der RBC verwendeten Version. Für diese Arbeit wurde der Programmcode, welcher zuvor auf zwei Module aufgeteilt war, von denen eines Bälle erkannte und das andere aus diesen den zu spielenden Ball selektierte, in einem einzigen Modul zusammengefasst. Ferner wurde der Quelltext neu strukturiert und hinsichtlich der Laufzeit des Moduls und der Korrektheit seiner Ergebnisse optimiert.

## 5.1 Ein- und Ausgaben

Das Modul erhält als Eingabe eine Reihe von Repräsentationen, welche im folgenden beschrieben sind. Zu diesen gehören zunächst ein Graustufenbild $Y$ mit einem Wertebereich

von [0,255], das aus dem Y-Kanal des YUV422-Kamerabildes besteht, sowie ein Farbklassenbild $C$, welches jedem Pixel des Kamerabildes die Farbklasse „schwarz“, „weiß“, „grün“ oder „keine“ zuordnet (beide enthalten in *ECImage*). Außerdem erhält das Modul die Kameramatrix (*CameraMatrix*), welche die Translation und Rotation der Kamera zum Torso des *NAO* beschreibt, Informationen über die das aktuelle Bild liefernde Kamera (*CameraInfo*) wie zum Beispiel Brennweite und Dimensionen des Bildes, den Versatz eines am Horizont ausgerichteten Koordinatensystems zum tatsächlichen Bild (*ImageCoordinateSystem*), die Koordinaten des im Bild erkannten Feldrands (*FieldBoundary*), die im Bild gefundenen Hindernisse (*PlayersPercept*), zu denen insbesondere Roboter gehören, sowie die aus den bekannten Positionen der Körperteile des *NAO*s berechnete Kontur des Roboterkörpers im Kamerabild (*BodyContour*). Dazu kommen die für die Selektion des zu spielenden Balls benötigten Repräsentationen *FrameInfo* und *RobotPose*, welche den Zeitstempel des Bildes relativ zum Start der Software beziehungsweise die aktuell vermutete Position des Roboters auf dem Feld relativ zu dessen Mittelpunkt enthalten.

Ausgabe des Moduls ist die Repräsentation *BallPercept*, die die Position des gefundenen Balls sowohl in der Bild- als auch in der Feldebene relativ zu den Füßen des Roboters enthält.

## 5.2 Kantenerkennung

Um potenzielle Bälle zu finden, wird entsprechend der in Kapitel 4.3 vorgestellten Methoden zunächst aus dem Graustufenbild ein binäres Konturbild $K$ erstellt. Hierbei ist $\{false, true\}$ die Wertemenge von $K$.

Um durch Bildrauschen entstandene Konturpunkte im Konturbild zu vermeiden, wird zunächst analog zum Vorgehen des Canny-Operators ein Gauß-Filter auf das Graustufenbild angewendet. Damit die Glättung eine möglichst geringe Laufzeit hat, hat der verwendete Gauß-Filter eine Filtermatrix $H^G$ der Größe $3 \times 3$, die bei einer Standardabweichung von $\sigma \approx 0{,}79788$ zur Faltung nur ganzzahlige Operationen benötigt.

$$H^G = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{5.1}$$

Als nächstes wird mithilfe des Sobel-Operators aus dem Graustufenbild $Y$ eine Gradientenkarte $\nabla Y$ erstellt. Um die Werte der Einträge von $Y_x$ und $Y_y$ in jeweils 8 Bit speichern zu können, sind hierbei die Sobel-Filter um den Faktor $\frac{1}{8}$ skaliert.

$$Y_x := \frac{1}{8} H_x^S * H^G * Y \tag{5.2}$$

$$Y_y := \frac{1}{8} H_y^S * H^G * Y \tag{5.3}$$

$$\nabla Y(u,v) := (E_Y(u,v), \Phi_Y(u,v)) = \left(\sqrt{Y_x^2(u,v) + Y_y^2(u,v)}, \operatorname{atan2}(Y_y(u,v), Y_x(u,v))\right) \tag{5.4}$$

Da eine sowohl echtzeitfähige als auch korrekte Kantenerkennung gewünscht ist, wurden mehrere Ansätze zur Erstellung eines Konturbilds zur Evaluation implementiert. Diese im Anschluß näher erläuterten Verfahren sind ein einfaches Schwellwertverfahren, eine Nicht-Maximum-Unterdrückung der Gradientenkarte mit einem anschließendem Schwellwertverfahren sowie eine Abwandlung des von Topal und Akinlar entwickelten Edge Drawing-Verfahrens (vgl. [26]). In der genannten Reihenfolge nimmt einerseits die Komplexität der Verfahren und damit ihre erwartete Laufzeit zu, während andererseits auch die erwartete Güte der Kantendetektion im Hinblick auf die von Canny definierten Kriterien (siehe Kapitel 4.3.3) und somit auch im Hinblick auf die Tauglichkeit ihrer Ergebnisse für die Erkennung von Kreisen steigt.

### 5.2.1 Schwellwertverfahren

Das einfache Schwellwertverfahren erstellt aus der Gradientenkarte ein Konturbild, indem all jene Punkte $(u, v)$ als Konturpunkte betrachtet werden, deren Gradientenlänge mindestens einen definierten Schwellwert $t$ beträgt.

$$K(u, v) := E_Y(u, v) \geq t \tag{5.5}$$

Da für dieses Verfahren lediglich die Länge der Gradienten benötigt wird, wird, um Laufzeit zu sparen, $\Phi_Y$ nicht berechnet.

### 5.2.2 Schwellwertverfahren mit Nicht-Maximum-Unterdrückung

Das zweite implementierte Verfahren zur Bildung eines Konturbilds nutzt ebenfalls diejenigen Punkte als Konturpunkte, deren Gradientenlänge größer oder gleich einem definierten Schwellwert $t$ ist. Es schließt dabei allerdings analog zu der in [19] beschriebenen Variante des Canny-Operators all jene Punkte aus, deren Gradientenlängen keine Maxima in Richtung der entsprechenden Gradienten bilden. Dadurch wird wie von Canny von einem optimalen Kantenerkenner gefordert pro Kante im Graustufenbild nur eine Antwort geliefert.

Um die Nicht-Maximum-Unterdrückung durchführen zu können, werden zunächst – wie in Gleichung 5.6 gezeigt – im Bild der eingeschränkten Gradientenrichtungen $\Phi'_Y$ die Richtungen der Gradientenkarte auf die Hauptrichtungen $\uparrow, \downarrow, \leftarrow, \rightarrow, \nearrow, \searrow, \swarrow$ und $\nwarrow$ eingeschränkt, welche weiter zu den Richtungen $\updownarrow, \leftrightarrow, \nearrow\!\!\swarrow$ und $\nwarrow\!\!\searrow$ zusammengefasst werden. Bei der Klassifizierung in die genannten Richtungen ist dabei zu beachten, dass im Bildkoordinatensystem die $y$-Achse nach unten zeigt, sodass die in $\Phi_Y$ berechneten Winkel nicht wie üblich ausgehend

von der $x$-Achse gegen, sondern mit dem Uhrzeigersinn verlaufen.

$$\Phi'_Y(u,v) := \begin{cases} \leftrightarrow, \text{ falls } 0° \leq (\Phi_Y(u,v) + 22{,}5°) \bmod 180° < 45° \\ \nwarrow\!\!\searrow, \text{ falls } 45° \leq (\Phi_Y(u,v) + 22{,}5°) \bmod 180° < 90° \\ \updownarrow, \text{ falls } 90° \leq (\Phi_Y(u,v) + 22{,}5°) \bmod 180° < 135° \\ \swarrow\!\!\nearrow, \text{ falls } 135° \leq (\Phi_Y(u,v) + 22{,}5°) \bmod 180° < 180° \end{cases} \tag{5.6}$$

Nach der Reduktion der Gradientenrichtungen kann – wie in Gleichung 5.7 gezeigt – das Konturbild $K$ durch Nicht-Maximum-Unterdrückung berechnet werden. Hierzu wird je Punkt, der, da seine Gradientenlänge mindestens $t$ ist, ein Kandidat für einen Konturpunkt ist, geprüft, ob einer der beiden in Richtung seiner in $\Phi'_Y$ berechneten Hauptrichtung angrenzenden Punkte eine größere Gradientenlänge hat. Nur wenn dies nicht der Fall ist, ist ein Kandidat ein gültiger Konturpunkt.

$$K(u,v) := E_Y(u,v) \geq t \wedge$$
$$E_Y(u,v) \geq \begin{cases} \max(E_Y(u-1,v), E_Y(u+1,v)), \text{ falls } \Phi'_Y(u,v) = \leftrightarrow \\ \max(E_Y(u,v-1), E_Y(u,v+1)), \text{ falls } \Phi'_Y(u,v) = \updownarrow \\ \max(E_Y(u-1,v-1), E_Y(u+1,v+1)), \text{ falls } \Phi'_Y(u,v) = \nwarrow\!\!\searrow \\ \max(E_Y(u-1,v+1), E_Y(u+1,v-1)), \text{ falls } \Phi'_Y(u,v) = \swarrow\!\!\nearrow \end{cases} \tag{5.7}$$

### 5.2.3 Edge Drawing

Ein weiteres implementiertes Verfahren zur Erstellung eines Konturbilds aus der Gradientenkarte entspricht dem Edge Drawing (ED) genannten Verfahren, welches von Topal und Akinlar 2012 vorgestellt wurde. Dieses hat zum Ziel, bei kürzerer Laufzeit bessere Ergebnisse als der Canny-Operator, insbesondere dessen Implementierung in OpenCV, zu liefern [26]. Der von ED verfolgte Ansatz ist dabei, ausgehend von einer Menge von Ankerpunkten, die definitiv Konturpunkte darstellen, die im Bild vorhandenen Kanten „nachzuzeichnen“, das heißt jeweils die Kantenzüge, auf denen die Ankerpixel liegen, zu verfolgen und die dabei passierten Pixel als Konturpunkte zu markieren.

Hierzu werden zunächst ebenfalls die Richtungen der Gradientenkarte eingeschränkt, allerdings lediglich auf horizontale ($\leftrightarrow$) und vertikale ($\updownarrow$) Richtungen. Im Gegensatz zur Einschränkung der Gradientenrichtungen im Bild $\Phi'_Y$ im zuvor vorgestellten Verfahren werden dabei nicht die Richtungen der Gradienten, sondern die der durch die entsprechenden Punkte verlaufenden Kantenzüge klassifiziert. Da die Gradienten jeweils senkrecht auf diesen Kantenzügen stehen, werden somit horizontale Gradienten als $\updownarrow$ und vertikale Gradienten als $\leftrightarrow$ klassifiziert. Das Bild der Kantenzugrichtungen $D_Y$ ist damit ausgehend von der Gradientenkarte analog zum Bild $\Phi'_Y$ wie in Gleichung 5.8 definiert. Da dies allerdings zur Berechnung von $D_Y$ wie in Gleichung 5.9 beschrieben nur mithilfe von $Y_x$ und $Y_y$ äquivalent ist, wird $D_Y$

im Programmcode stattdessen auf letztere Weise ermittelt und $\Phi_Y$ für dieses Verfahren nicht berechnet.
Dies entspricht auch der von Topal und Akinlar beschriebenen Methode (vgl. [26]).

$$D_Y(u,v) := \begin{cases} \updownarrow, \text{ falls } 0^\circ \leq (\Phi_Y(u,v) + 45^\circ) \bmod 180^\circ < 90^\circ \\ \leftrightarrow, \text{ falls } 90^\circ \leq (\Phi_Y(u,v) + 45^\circ) \bmod 180^\circ < 180^\circ \end{cases} \tag{5.8}$$

$$D_Y(u,v) = \begin{cases} \updownarrow, \text{ falls } |Y_x| \geq |Y_y| \\ \leftrightarrow, \text{ falls } |Y_x| < |Y_y| \end{cases} \tag{5.9}$$

Außerdem werden von ED, wie schon von den anderen implementierten Verfahren, nur solche Pixel des Bildes betrachtet, deren Gradienten länger als ein definierter Schwellwert $t$ sind. Hierzu verwendet das Verfahren ein modifiziertes Bild $E'_Y$, welches nur solche Gradientenlängen enthält, die mindestens $t$ betragen.

$$E'_Y(u,v) := \begin{cases} 0, \text{ falls } E_Y(u,v) < t \\ E_Y(u,v) \text{ sonst} \end{cases} \tag{5.10}$$

Nachdem nun die Bilder $D_Y$ und $E'_Y$ berechnet wurden, werden als nächstes die Ankerpunkte bestimmt, von denen aus die Kantenzüge des Bildes nachgezeichnet werden sollen. Topal und Akinlar schlagen hierzu vor, das Bild in festgelegten horizontalen und vertikalen Intervallen zu durchlaufen und dabei all jene besuchten Pixel als Ankerpunkte zu verwenden, deren Gradientenlänge um mindestens einen definierten Schwellwert $a$ größer als die Gradientenlängen der beiden senkrecht zur Richtung des Kantenzugs adjazenten Pixel ist (vgl. [26]).
Gleichung 5.11 zeigt die Funktion $\text{anch}(u,v)$, welche auf die beschriebene Weise bestimmt, ob ein bestimmter Punkt $(u,v)$ ein Ankerpunkt ist.

$$\text{anch}(u,v) := E'_Y(u,v) - a \geq \begin{cases} \max\left(E'_Y(u,v-1), E'_Y(u,v+1)\right), \text{ falls } D_Y(u,v) = \leftrightarrow \\ \max\left(E'_Y(u-1,v), E'_Y(u+1,v)\right), \text{ falls } D_Y(u,v) = \updownarrow \end{cases} \tag{5.11}$$

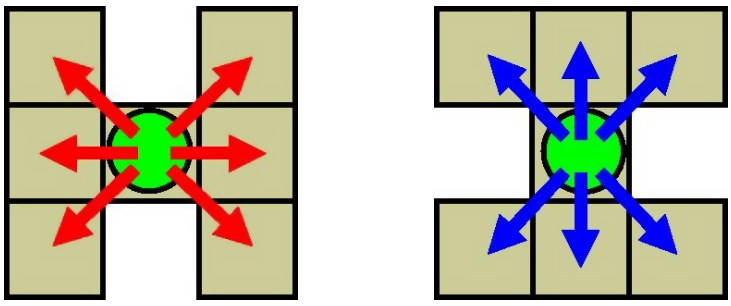

**Abbildung 5.1** Mögliche Fortsetzungen des Kantenzugs beim Zeichnen der Kantenzüge durch das ED-Verfahren. (Abbildung aus [26])
Links ist für den aktuellen Punkt $(x,y)$ $D_Y(x,y) = \leftrightarrow$, rechts $D_Y(x,y) = \updownarrow$.

Das Nachzeichnen der Kantenzüge zur Konstruktion des Konturbilds $K$ funktioniert dann

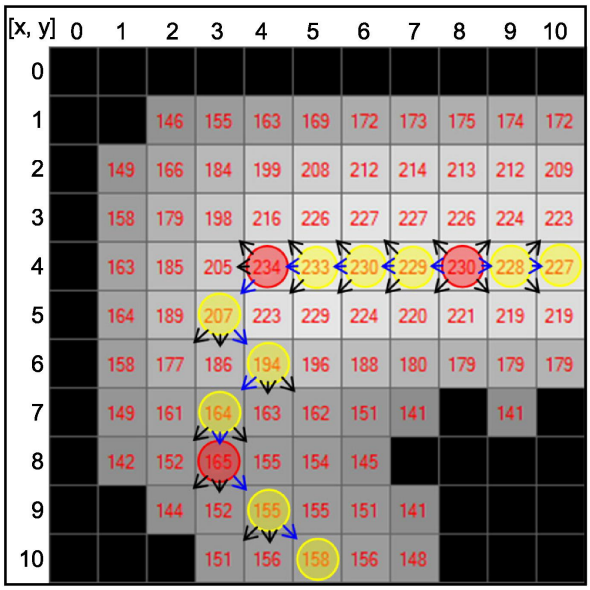


**Abbildung 5.2** Das Zeichnen der Kantenzüge durch das ED-Verfahren. (Abbildung aus [26]) Hierbei wurde vom Ankerpunkt (8,4) ausgegangen, die Zahlen in den Feldern stellen die Werte $E'_Y(x, y)$ und die roten Kreise die Ankerpunkte dar.

derart, dass ein jeweils aktueller Punkt $(x, y)$ als Konturpunkt markiert wird und aus dessen Nachbarn zwei nächste Punkte gewählt werden, welche in beide Richtungen des Kantenzugs durch $(x, y)$ diesen Kantenzug am Besten fortsetzen. Begonnen wird dabei bei den Ankerpunkten. Ist der Kantenzug durch den aktuellen Punkt horizontal, so wird als nächster Punkt zur rekursiven Fortführung des Kantenzugs zunächst derjenige der drei Punkte $(x-1, y-1), (x-1, y), (x-1, y+1)$ gewählt, der in $E'_Y$ die größte Gradientenlänge hat. Danach wird analog hierzu der Punkt aus den drei rechts angrenzenden Punkten mit der größten Gradientenlänge als nächster Punkt gewählt. Entsprechend funktioniert das Verfahren bei Punkten, durch die vertikale Kantenzüge verlaufen, mit den drei oben beziehungsweise unten angrenzenden Punkten (siehe Abbildung 5.1).

Dieses rekursive Verfahren wird unterbrochen, wenn die Gradientenlänge des aktuellen Punktes 0 ist, wobei er dann nicht als Konturpunkt markiert wird, oder wenn ein zuvor bereits als Konturpunkt markierter Punkt erreicht wird. Auf diese Weise werden nach und nach sämtliche Kantenzüge im Bild, auf denen ein Ankerpunkt liegt, mit einer Breite von einem Pixel in das Konturbild $K$ übertragen. Ein Beispiel hierzu zeigt Abbildung 5.2.

### 5.2.4 Optimierungen

Da Echtzeitfähigkeit und damit eine geringe Laufzeit ein wichtiges Kriterium für das implementierte Modul ist, wurden die beschriebenen Algorithmen bei der Implementierung noch weiter optimiert.

Da die zu findenden Bälle auf dem Spielfeld und damit unterhalb der Kameras des *NAO* liegen, ist die Suche nach Bällen oberhalb des Horizonts der bildgebenden Kamera nicht notwendig.

Um Zeit zu sparen, wird daher die Gradientenkarte und alle weiteren Bilder nur ab der Y-Koordinate des Ursprungs des gegebenen, am Horizont ausgerichteten Koordinatensystems abwärts berechnet. Da der *NAO* aufrecht steht und sein Kopf während der RBC horizontal stets ungefähr parallel zum Boden ausgerichtet ist, wird somit genau der Bereich oberhalb des Horizonts von der Suche nach Bällen ausgeschlossen.

Bei den Faltungen des Bildes $Y$ mit $\frac{1}{8}H_x^S$ beziehungsweise $\frac{1}{8}H_y^S$ zur Ermittlung der Gradienten sowie zuvor mit $H^G$ zur Glättung des Bildes wurde ausgenutzt, dass die linearen Filter durch die Assoziativität der Faltungsoperation – wie in den Gleichungen 5.12 bis 5.14 gezeigt – separierbar sind. Stattdessen wurde das Bild daher jeweils erst mit den $1 \times 3$- und danach mit den $3 \times 1$-Filtermatrizen gefaltet.

$$H^G = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \end{pmatrix} \tag{5.12}$$

$$\frac{1}{8}H_x^S = \frac{1}{8}\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \frac{1}{2}\begin{pmatrix} -1 & 0 & 1 \end{pmatrix} \tag{5.13}$$

$$\frac{1}{8}H_y^S = \frac{1}{8}\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \end{pmatrix} \tag{5.14}$$

Weiterhin existieren die Bilder $Y_x, Y_y$ und $\Phi_Y$ im Programmcode nur implizit, da die weiteren Schritte der beschriebenen Verfahren direkt auf die Ergebnisse der Berechnungen einzelner Pixel dieser Bilder angewandt werden.

Um die Laufzeit stark zu senken, finden außerdem alle Berechnungen, bei denen dies möglich ist, nämlich die komplette Berechnung der Gradientenkarte $\nabla Y$ und der Bilder $\Phi'_Y, D_Y$ und $E'_Y$ sowie $K$ im Schwellwertverfahren, für je 16 Pixel gleichzeitig statt. Dies ist möglich durch die Verwendung von Instruktionen aus dem Streaming SIMD Extensions (SSE)-Befehlssatz für x86-Prozessoren, welcher von der CPU des *NAO* bis zur Version SSSE3 unterstützt wird (vgl. [11], [27]). Zur Einbindung der SSE-Instruktionen in den C++-Quelltext des Moduls werden die von der Intel Corporation spezifizierten intrinsischen Funktionen benutzt. Eine Übersicht der intrinsischen Funktionen findet sich in [28].

Überdies ist die Berechnung der Quadratwurzel mithilfe von SSE-Instruktionen sehr aufwändig, sodass diese bei der Erzeugung des Bildes $E_Y$ und damit auch des Bildes $E'_Y$ im Programmcode nicht berechnet wird, sondern diese Bilder tatsächlich jeweils die quadrierte Länge der Gradientenvektoren enthalten. Die tatsächliche Länge eines Vektors wird im weiteren Programmcode nur dann berechnet, wenn sie für eine weitere Berechnung notwendig ist. Weil die Quadratwurzelfunktion streng monoton wächst und somit für beliebige $a, b \in \mathbb{R}$ Gleichung 5.15 gilt, ist dies allerdings nur bei der Bestimmung der Ankerpunkte für das Edge

Drawing-Verfahren mittels der Funktion anch$(u, v)$ (siehe Abschnitt 5.2.3) notwendig.

$$a \leq b \Leftrightarrow \sqrt{a} \leq \sqrt{b} \tag{5.15}$$

## 5.3 Kreiserkennung

Zum Finden von Kreisen im erhaltenen Konturbild wurde das von Chiu et al. entwickelte Verfahren Fast Randomized Hough Transform (FRHT) (vgl. [29]) verwendet. Dieses findet wie die Hough-Transformation alle Kreise im gegebenen Konturbild, durchsucht dazu aber nicht das gesamte Bild, sondern nur einige zufällig ausgewählte Pixel.

Hierzu werden aus allen gefundenen Konturpunkten zufällig einzelne Punkte ausgewählt. Die Idee des FRHT ist nun, für jeden dieser Punkte festzustellen, ob er sich auf einem Kreis befindet. Dafür wird davon ausgegangen, dass zwei Konturpunkte, welche denselben Abstand zum gewählten Punkt $d_s$ haben, mit diesem einen Kreis definieren (siehe Abbildung 5.3). Nach solchen Punkten wird innerhalb eines Suchfensters mit festgelegter Größe um $d_s$ gesucht. [29]

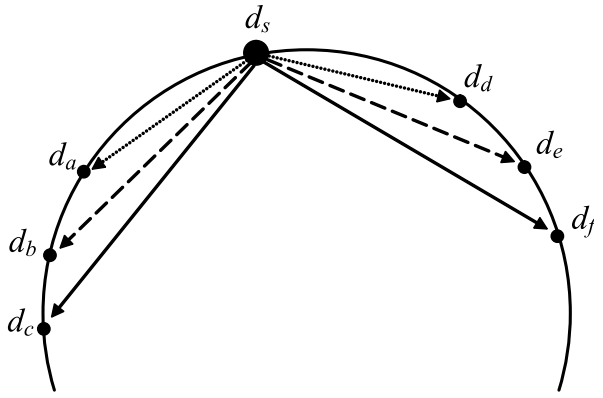


**Abbildung 5.3** Paare von Konturpunkten $(d_a, d_d), (d_b, d_e), (d_c, d_f)$ mit jeweils demselben Abstand von $d_s$, welche mit $d_s$ denselben Kreis bilden.
(Abbildung aus [29])

Die Suche funktioniert dabei wie von Chiu et al. vorgeschlagen derart, dass zunächst je Ausgangspunkt $d_s$ eine initial leere Liste angelegt wird. Wenn nun im Suchfenster ein Konturpunkt $d_a$ gefunden wird, so wird dessen ganzzahliger Abstand $l = \lfloor |d_a - d_s| \rfloor$ zu $d_s$ berechnet. Ist der Eintrag der Liste an der Stelle $l$ leer, so wird $d_a$ an dieser Stelle eingetragen. Im anderen Fall, dass der Eintrag $l$ der Liste bereits ein anderer Konturpunkt $d_b$ ist, so wird, sofern $d_s, d_a$ und $d_b$ nicht kolinear sind, der durch die drei Punkte verlaufende Kreis mittels der in Abschnitt 4.4.1 beschriebenen Methode berechnet und als Kreiskandidat gespeichert (vgl. [29]).

Die auf diese Weise gefundenen Kreiskandidaten werden ähnlich der Kreisfindung durch die Hough-Transformation akkumuliert, sodass der Akkumulator eines Kreiskandidaten desto höher ist, je mehr Paare von Konturpunkten auf diesem gefunden wurden. Der Akkumulator

des Kreiskandidaten in Abbildung 5.3, auf welchem drei Paare von Konturpunkten gefunden wurden, hätte demnach zum Beispiel den Wert 3. Da davon ausgegangen wird, dass der gewählte Punkt $d_s$ jeweils nur auf maximal einem Kreis im Bild liegt, wird nach dem Durchsuchen des gesamten Suchfensters der Kreiskandidat mit dem höchsten Akkumulator ausgewählt, sofern sein Akkumulator mindestens einen definierten Schwellwert beträgt.

Um diesen Kandidaten mit dem Mittelpunkt $(x_c, y_c)$ und dem Radius $r_c$ als Kreis zu verifizieren, wird anschließend – wie in Gleichung 5.16 gezeigt – die Menge aller Konturpunkte $D_c$, die auf diesem Kreis liegen, bestimmt. $\epsilon$ ist dabei ein festgelegter Parameter, welcher bestimmt, in welchem Abstand um den Kreisbogen Punkte noch zum Kreis gezählt werden. Dann wird geprüft, ob für einen definierten Schwellwert $m$ die Ungleichung $\frac{|D_c|}{2\pi r_c} > m$ gilt, also die relative Abweichung der Kardinalität von $D_c$ zum Umfang des Kreises größer als $m$ ist. [29]

$$D_c = \left\{ (x,y) \middle| \left| \sqrt{(x-x_c)^2 + (y-y_c)^2} - r_c \right| \leq \epsilon, K(x,y) = true \right\} \tag{5.16}$$

Sämtliche Kreiskandidaten, die auf diese Weise verifiziert wurden, werden als im Bild vorhandene Kreise und somit als Ballkandidaten angenommen.

Die Performanz dieses Algorithmus sowohl hinsichtlich seiner Laufzeit als auch hinsichtlich der Häufigkeit richtiger Kreiserkennungen ist abhängig von der Auswahl einerseits der Wahrscheinlichkeit der Wahl eines Konturpunkts als Ausgangspunkt und andererseits der Größe des Suchfensters. Chiu et al. geben keine Empfehlungen für die Wahl dieser Größen, stellen allerdings in der Evaluation des Algorithmus heraus, dass, wenn ein zu findender Kreis aus $k$ Konturpunkten besteht, aus den $n$ Konturpunkten mindestens $\frac{n}{k}$ als Ausgangspunkte gewählt werden sollten, um einen Ausgangspunkt, der auf dem Kreis liegt, zu erhalten (vgl. [29]). Um dies zu erreichen, wird zur Auswahl der Ausgangspunkte über alle Konturpunkte des Bildes iteriert und jeder Punkt mit einer unterschiedlichen Wahrscheinlichkeit $p$ gewählt, die abhängig von der erwarteten Anzahl der Konturpunkte eines Balls im Bild an der jeweiligen Position ist. Hierzu wird vereinfachend angenommen, dass ein betrachteter Punkt im Bild jeweils dem Fußpunkt eines Balls auf dem Feld entspricht. Dann wird ausgehend von einer fest definierten erwarteten Ballgröße der ins Bild projizierte Radius $r$ eines Balls mit diesem Fußpunkt auf dem Feld ermittelt. Die erwartete Anzahl der Konturpunkte $k$ des Balls im Bild bestimmt sich dann aus Gleichung 5.17 und die Wahrscheinlichkeit $p$ zur Wahl des betrachteten Balls als Konturpunkt aus Gleichung 5.18.

$$k = \pi \cdot 2r \tag{5.17}$$

$$p = \frac{1}{k} \tag{5.18}$$

Dieses Verfahren geht davon aus, dass Kanten im Konturbild genau einen Pixel breit sind. Falls zur Berechnung des Konturbildes das einfache Schwellwertverfahren verwendet wurde, sind allerdings gefundene Kanten in der Regel drei Pixel breit. Um dies zu kompensieren, wird in diesem Fall $k$ entsprechend mit dem Faktor 3 multipliziert.

Wurde ein Punkt als Ausgangspunkt gewählt, so wird davon ausgegangen, dass der Kreis, auf dem dieser Punkt liegt, korrekt erkannt wird, wenn der Radius des Suchfensters mindestens dem Radius des Kreises entspricht. Da dieser zuvor bereits als $r$ berechnet wurde, wird diese Größe als Radius des Suchfensters genutzt.

Um darüber hinaus eine bessere Performanz zu erhalten, wird der Schwellwert, den der Akkumulator eines Kreiskandidaten mindestens erreichen muss, abhängig von der erwarteten Anzahl der Konturpunkte des Kreises im jeweiligen Suchfenster gewählt. Da für das Suchfenster derselbe Radius wie der eines darin erwarteten Kreises festgelegt wurde, wird davon ausgegangen, dass mindestens ein Viertel der Konturpunkte des Kreises im Suchfenster liegt. Weil außerdem je zwei Konturpunkte des Kreises den Akkumulator des richtigen Kreiskandidaten um 1 erhöhen, ist der Schwellwert somit als $\frac{k}{8}$ festgelegt.

## 5.4 Plausibilitätsprüfung

Da die Kreiserkennung durch den FRHT-Algorithmus viele Kreise in Bildern findet, die tatsächlich keine Bälle sind, wird nach der Ballerkennung noch eine Plausibilitätsprüfung durchgeführt, mit der sichergestellt werden soll, dass Ballkandidaten tatsächliche Bälle sind und als Rückgabe des Moduls verwendet werden können. Diese Validierung besteht aus mehreren Schritten, wobei zunächst die gefundenen Ballkandidaten im Bild validiert werden. Anschließend werden die Koordinaten und Radien der Bälle auf dem Feld relativ zum Roboter bestimmt. In einem letzten Schritt schließlich werden die Ballkandidaten basierend auf diesen ermittelten Feldkoordinaten und Radien erneut auf bestimmte Eigenschaften geprüft.

### 5.4.1 Validierung in Bildkoordinaten

Der erste Validierungsschritt in Bildkoordinaten prüft zunächst, ob der Mittelpunkt eines gefundenen Kreises innerhalb des Bildes und innerhalb der im Bild erkannten Feldgrenze sowie außerhalb der im Bild erwarteten Kontur des Roboters liegt. Ist dies der Fall, wird als Nächstes der Ballkandidat ausgeschlossen, sollte er sich innerhalb eines im Bild gesehenen Hindernisses befinden. Als letztes wird sichergestellt, dass nicht fälschlicherweise Bälle im Teppich gefunden werden, aber andererseits vollständig grüne Bälle dennoch erkannt werden können, indem geprüft wird, ob der Anteil der grünen Pixel im gefundenen Kreis im Bild entweder geringer als ein gegebener Schwellwert oder höher als ein anderer Schwellwert ist. Hierzu werden ausgehend vom Mittelpunkt des Kreises alle Punkte horizontal, vertikal und diagonal bis zum Kreisbogen betrachtet.

### 5.4.2 Projektion in Feldkoordinaten

Nach der Validierung der gefundenen Kreise im Bild als Ballkandidaten werden für weitere Plausibilitätsprüfungen sowie für die spätere Ballselektion die Positionen der gefundenen Bäl-

le relativ zum Roboter benötigt. Hierzu werden Position und Radius jedes Ballkandidaten durch Projektion der Bild- in Feldkoordinaten bestimmt. Weil die tatsächlichen Radien der gesuchten Bälle im Vorhinein nicht bekannt sind, kann diese Bestimmung nur näherungsweise geschehen. Zur Bestimmung des Radius eines Ballkandidaten wird zunächst vereinfachend davon ausgegangen, dass die im Bild horizontal am weitesten vom Mittelpunkt des Kreises entfernten Punkte des Balls im Raum auf einer zum Boden parallelen Ebene liegen, deren Abstand zum Boden dem Ballradius entspricht, und sich auf der Kugel des Balls genau gegenüberliegen, also den Durchmesser des Balls voneinander entfernt sind. Weil der Ballradius noch nicht bekannt ist, wird für diesen zunächst ein erwarteter Wert genutzt. Basierend darauf werden die beiden Punkte auf die zum Boden parallele Ebene projiziert. Der Abstand der so erhaltenen Punkte in Feldkoordinaten entspricht somit näherungsweise dem Durchmesser des Balls. Da somit nun der Ballradius bekannt ist, wird als nächstes die Position des Balls in Feldkoordinaten bestimmt. Diese Position beschreibt den Fußpunkt des Balls. Da dies genau derjenige Punkt ist, der auf der Feldebene liegt und vom Mittelpunkt des Balls genau den Radius des Balls entfernt ist, kann er durch Bestimmung des Ballmittelpunkts erhalten werden. Wird nun vereinfachend davon ausgegangen, dass der Mittelpunkt des Balls der Projektion des Kreismittelpunkts im Bild auf die Ebene entspricht, deren Abstand von der Feldebene gleich dem Ballradius ist, so kann mithilfe dieser Projektion die Ballposition bestimmt werden.

Die Ermittlung von Position und Radius eines Ballkandidaten in Feldkoordinaten ist somit aufgrund einiger vereinfachender Annahmen ungenau, aber ausreichend genau, um den Radius eines Balls auf verschiedene Entfernungen zu bestimmen.

Zusätzlich zu den Positionen der Ballkandidaten relativ zum Roboter werden im Folgenden auch ihre absoluten Positionen relativ zum Mittelpunkt des Felds genutzt. Um diese zu bestimmen, wird die aktuell vom Roboter geschätzte eigene Rotation und Translation auf dem Spielfeld auf die ermittelten Ballpositionen angewendet.

### 5.4.3 Validierung in Feldkoordinaten

Nachdem der ungefähre Radius eines Ballkandidaten auf dem Feld bekannt ist, wird zunächst sichergestellt, dass dieser innerhalb eines festgelegten erwarteten Bereichs liegt. Weiterhin wird sichergestellt, dass Bälle nur innerhalb des Felds gefunden werden, indem die absolute Ballposition mit der Größe des Felds verglichen wird. Um ferner zu vermeiden, dass Bälle fälschlicherweise an Stellen erkannt werden, an welchen sich Linienkreuzungen oder Strafstoßpunkte befinden, muss entweder der Radius des erkannten Balls sich um mindestens einen festgelegten Schwellwert von der Breite einer Linie unterscheiden oder der erkannte Ball muss mindestens einen definierten Mindestabstand von den basierend auf der aktuell geschätzten Roboterposition vermuteten Feldlinien und Strafstoßpunkten haben.

## 5.5 Selektion

Nachdem durch die vorangegangenen Schritte die im Kamerabild vorhandenen Bälle sowie ihre Positionen auf dem Feld relativ zum *NAO* bestimmt wurden, wird zuletzt derjenige gesehene Ball ausgewählt, welcher vom implementierten Modul zurückgegeben werden soll. Hierbei wird der Ansatz verfolgt, dass bevorzugt ein bereits zuvor gesehener und damit zum Spielen verwendeter Ball weiter vom restlichen B-Human-System gesehen werden sollte, sodass der Roboter nicht, während er eine Bewegung ausführt, um einen Ball zu spielen, zu einem anderen Ball wechselt oder sich wiederholt zwischen mehreren Bällen umentscheidet, ohne dabei einen dieser Bälle zu spielen.

Zur Umsetzung dieses Verhaltens wird stets die letzte Position eines selektierten Balls relativ zum Mittelpunkt des Felds zusammen mit dem Zeitstempel, zu dem dieser Ball gesehen wurde, gespeichert. Wenn dann neue Bälle gesehen wurden, so wird, sofern nicht mindestens eine festgelegte Zeitspanne seit der letzten Sichtung verstrichen ist, stets derjenige Ball selektiert, welcher dem zuletzt gesehenen Ball am nächsten ist. Ist andernfalls zu viel Zeit seit der letzten Sichtung vergangen, wird der dem *NAO* nächste Ball ausgewählt.

Aufgrund der Nutzung der geschätzten Roboterpose sowohl für die Ballselektion als auch zuvor zur Validierung ist die Ballerkennung stark abhängig von der Lokalisierung des Roboters auf dem Feld. Dieses Vorgehen wurde gewählt, da erwartet wird, dass die Selbstlokalisierung des B-Human-Systems während der RBC durchgängig ausreichend korrekt ist, weil die Challenge bis auf den spielenden *NAO* in einer statischen Umgebung stattfindet.

# Kapitel 6

# Evaluation

In diesem Kapitel wird die implementierte Software hinsichtlich ihrer Tauglichkeit für die Anwendung in der RBC evaluiert.

Um tauglich zu sein, muss die Software echtzeitfähig und korrekt sein. Echtzeitfähigkeit bedeutet dabei, dass ihre Laufzeit in allen in der RBC vorkommenden Situationen ausreichend gering sein muss, um den Rest der Software nicht zu blockieren und oft genug gesehene Bälle zu liefern, dass der Roboter sinnvoll zu diesen gesteuert werden kann. Auf der anderen Seite ist die Software ausreichend korrekt, wenn die Korrektheit der gelieferten Ergebnisse genügt, um in der RBC die richtigen Bälle schnell zu finden und zu spielen.

Neben dem gesamten entwickelten Modul sollen zunächst die verschiedenen implementierten Verfahren zur Kantenerkennung hinsichtlich ihrer Echtzeitfähigkeit und der Qualität ihrer Ergebnisse evaluiert werden, um aus diesen das beste Verfahren zur Verwendung in der RBC auszuwählen. Daher findet ein zweistufiges Testverfahren statt, indem im ersten Schritt die Verfahren zur Kantenerkennung verglichen werden und im zweiten Schritt das gesamte Modul evaluiert wird. Im zweiten Schritt wird dabei dasjenige Verfahren, das sich im ersten Schritt als das beste herausgestellt hat, verwendet.

Um zur Beurteilung der Echtzeitfähigkeit der einzelnen Programmteile deren Laufzeiten zu messen, wird das im B-Human-System vorhandene STOPWATCH-Makro verwendet. Dieses misst die während eines gewählten Programmteils vergangene Prozessorzeit und ermöglicht es, diese einerseits direkt an einen per TCP verbundenen Computer zu senden und andererseits in Log-Dateien zur späteren Auswertung zu speichern. Außerdem bietet das B-Human-System die Möglichkeit, die minimalen und maximalen sowie die durchschnittlichen Laufzeiten innerhalb eines bestimmten Zeitfensters anzuzeigen. Dies erfolgt im Cognition-Prozess, in welchem das implementierte Modul läuft, getrennt je nach Kamera, von der das verarbeitete Bild stammt.

## 6.1 Kantenerkennung

Zunächst werden die verschiedenen implementierten Verfahren zur Bildung eines Konturbilds evaluiert. Dies sind das einfache Schwellwertverfahren, die Nicht-Maximum-Unterdrückung und das Edge Drawing-Verfahren, welche jeweils mit einer Gradientenkarte arbeiten, die entweder direkt aus dem Graustufenbild oder aus einem zuvor mit dem Gauß-Filter geglätteten Bild erzeugt wurde.

### 6.1.1 Testverfahren

Um die Methoden zur Kantenerkennung zu evaluieren, wird ein mit der Software bespielter *NAO* so ausgerichtet, dass seine obere Kamera ein komplexes Bild aufnimmt, in welchem sich entsprechend der Situation in der RBC neben einem weiteren Roboter mehrere beliebige Bälle und einige Feldlinien befinden (siehe Abbildung 6.1). Damit möglichst der hinsichtlich der Laufzeit schlechteste Fall verglichen wird, wird der Kopf des *NAO* so ausgerichtet, dass sich das Kamerabild vollständig unter dem Horizont befindet und somit das gesamte Bild in ein Konturbild umgewandelt wird. Die Erzielung des schlechtesten Falls ist auch der Grund für die Verwendung der oberen Kamera, da die Auflösung derer Bilder doppelt so groß ist wie die der unteren Kamera.

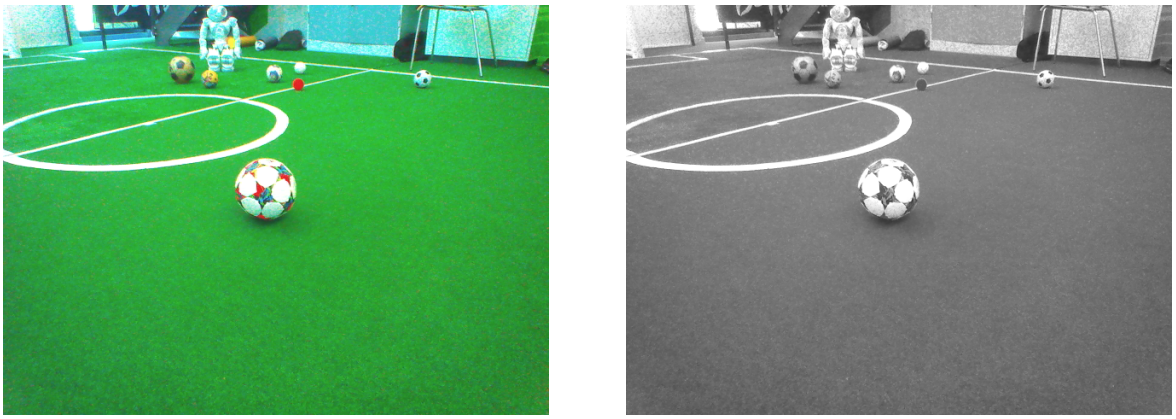

**Abbildung 6.1** Das für die Evaluation der Verfahren zur Kantenerkennung verwendete Kamerabild und das zugehörige für die Bildung des Konturbilds verwendete Graustufenbild.

Nun werden nacheinander die einzelnen Verfahren zur Bildung des Konturbilds aktiviert. Um bestmögliche Ergebnisse zu erhalten, werden für jedes der Verfahren die Parameter der Kantenerkennung (vgl. Abschnitt 5.2) so angepasst, dass ein möglichst gutes Konturbild erzielt wird. Dann werden die für das obere Kamerabild benötigten minimalen, maximalen und durchschnittlichen Laufzeiten notiert und die erzielten Konturbilder zum Vergleich gespeichert. Da weiterhin die Weichzeichnung des Graustufenbilds mit einem Gauß-Filter vor der Erstellung der Gradientenkarte evaluiert werden soll, wird diese für jedes Verfahren ebenfalls für je eine Messung aktiviert und für eine weitere Messung deaktiviert.

### 6.1.2 Ergebnisse

Die durch die einzelnen Verfahren erzielten Ergebnisse zeigt Abbildung 6.2, die Laufzeiten der jeweiligen Komponenten sind in Tabelle 6.1 zu sehen. Die Laufzeiten der Anwendung des Gauß-Filters sind dabei separat aufgelistet, da dessen Berechnung unabhängig von der Generierung des Konturbilds ist.

| Angewandtes Verfahren | minimal | maximal | durchschnittlich |
|---:|:---:|:---:|:---:|
| Gauß-Filter | 0,286 ms | 0,639 ms | 0,39 ms |
| Schwellwertverfahren | 0,866 ms | 1,906 ms | 1,13 ms |
| Nicht-Maximum-Unterdrückung | 6,492 ms | 8,016 ms | 7,02 ms |
| Edge Drawing | 10,208 ms | 13,119 ms | 10,8 ms |

**Tabelle 6.1** Laufzeiten der Verfahren zur Bildung des Konturbilds sowie zur Weichzeichnung des Graustufenbilds.

Es zeigt sich, dass die Weichzeichnung des Bilds mit dem Gauß-Filter bei allen Verfahren unerwünschtes Rauschen in den Konturbildern vermeidet, die Erkennung korrekter Kanten jedoch nicht, also eine durchweg positive Auswirkung auf die erzielten Ergebnisse hat. Da die für die Glättung benötigte Laufzeit zudem im Verhältnis zu den Laufzeiten der Konturbilderstellung gering ist, wird die Weichzeichnung im Modul als sinnvoll erachtet und für die weitere Evaluation benutzt.

Von den Verfahren zur Konturbilderstellung ist das Schwellwertverfahren das schnellste mit einem deutlichem Abstand von mindestens 4,5 ms zur Nicht-Maximum-Unterdrückung. Somit erfüllt es am ehesten das Kriterium der Echtzeitfähigkeit. Jedoch sind wie erwartet die Kantenzüge im erhaltenen Konturbild jeweils mindestens drei Pixel breit, während die anderen beiden Verfahren jeweils ein Pixel breite Kantenzüge liefern. Dies führt in der implementierten Kreiserkennung mit dem FRHT-Algorithmus dazu, dass entsprechend im durch das Schwellwertverfahren gebildeten Konturbild mindestens die dreifache Anzahl an Konturpunkten zur Bildung von Kreiskandidaten betrachtet wird und somit die Laufzeit des FRHT-Algorithmus deutlich größer ist, ohne dass mehr tatsächliche Kreise gefunden werden.

Die anderen beiden Verfahren liefern sehr ähnliche Ergebnisse, wobei das Edge Drawing-Verfahren weniger Details als das Verfahren der Nicht-Maximum-Unterdrückung liefert, damit aber auch etwas weniger unerwünschtes Rauschen als Konturpunkte ausgibt. Allerdings ist die Laufzeit des Edge Drawing-Verfahrens stets um mindestens 2 ms, also 25% der maximalen Laufzeit der Nicht-Maximum-Unterdrückung, länger als die der Nicht-Maximum-Unterdrückung.

Insgesamt stellt sich somit also das zweite implementierte Verfahren, nämlich die Nicht-Maximum-Unterdrückung, als das geeignetste heraus, da es einen guten Kompromiss zwischen kurzer Laufzeit und guter Qualität der Ergebnisse bietet. Daher wird dieses Verfahren für die weitere Evaluation des gesamten Moduls verwendet.

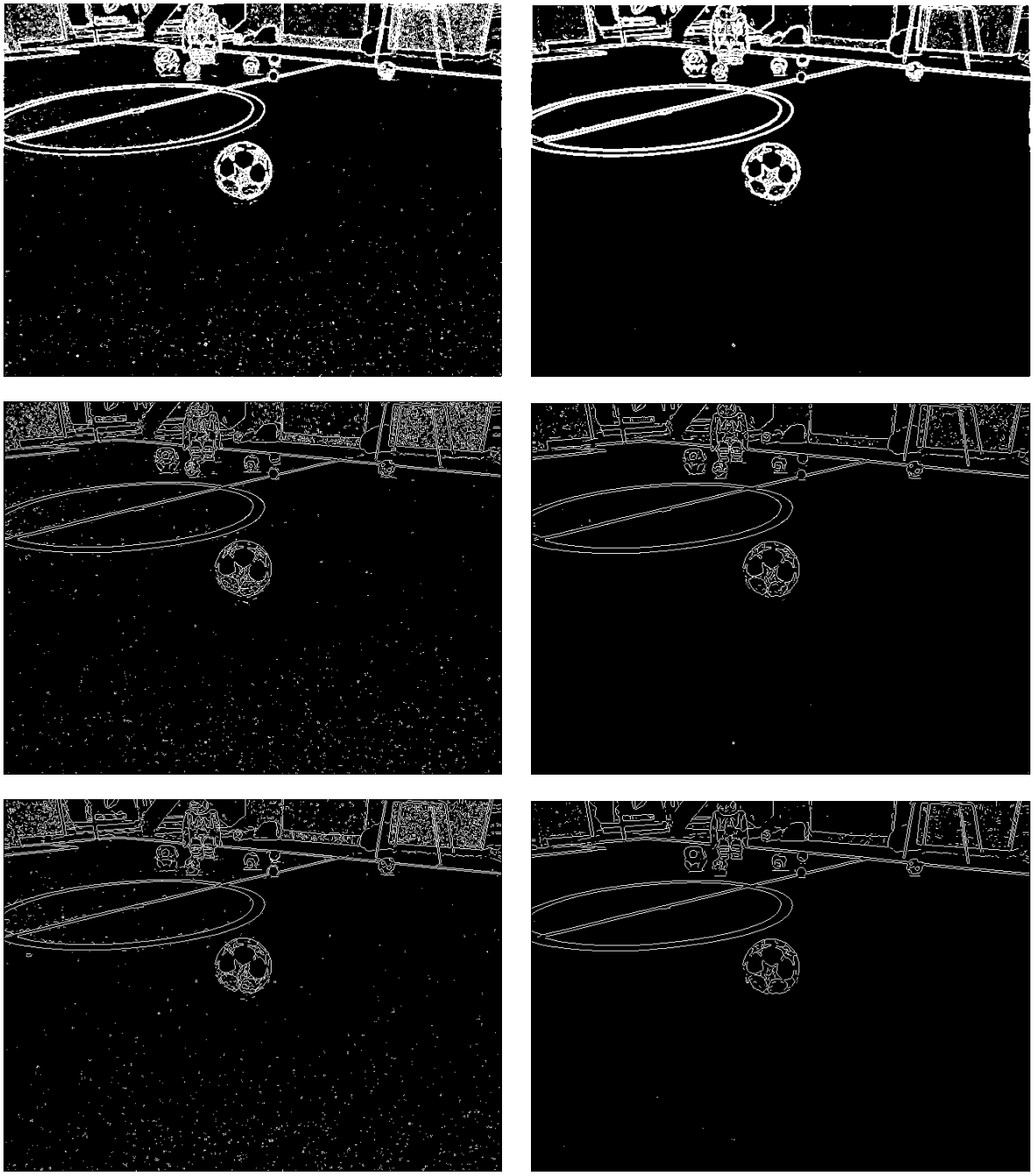

**Abbildung 6.2** Die von den einzelnen implementierten Verfahren aus dem Graustufenbild berechneten binären Konturbilder.
In der ersten Zeile sind Ergebnisse des Schwellwertverfahrens zu sehen, in der zweiten Ergebnisse der Nicht-Maximum-Unterdrückung und in der dritten Zeile Ergebnisse des Edge Drawing-Verfahrens.
Die Bilder in der linken Spalte wurden durch Anwendung der Verfahren auf das gegebene Graustufenbild erhalten, während für die Bilder in der rechten Spalte das Graustufenbild zunächst mit dem Gauß-Filter weichgezeichnet wurde.

## 6.2 Gesamtes Modul

Nachdem nun ein Verfahren zur Kantenerkennung zur Verwendung im Modul ausgewählt und die Nutzung der Weichzeichnung des Graustufenbilds bestätigt wurde, wird im Folgenden die Performanz des Moduls im Hinblick auf seine Korrektheit und seine Echtzeitfähigkeit geprüft.

### 6.2.1 Testverfahren

Zur Evaluation des implementierten Moduls wird ein mit der Software bespielter *NAO* angelehnt an die Regeln der RBC drei Minuten lang auf dem SPL-Feld laufen gelassen, auf welchem zuvor zwei *NAO*s und fünf beliebige Bälle platziert wurden. Die verwendeten Bälle sind in Abbildung 6.3 gezeigt. Das Verhalten des Roboters ist dabei nicht repräsentativ für seine Performanz in der RBC, da das normale Fußballverhalten des Teams B-Human statt des 2015 eigens für die RBC entwickelten Verhaltens zur Steuerung des Roboters verwendet wird. Überdies wird das Experiment nicht auf dem 2015 verwendeten Feld der Größe 9 m × 6 m, sondern auf dem bis zum Jahr 2012 verwendeten Feld der Größe 6 m × 4 m durchgeführt.

**Abbildung 6.3** Die für die Evaluation verwendeten beliebigen Bälle.

Danach wird manuell die bei diesem Experiment vom B-Human-System geschriebene Log-Datei angesehen, in welcher neben den Laufzeiten des Moduls und seiner einzelnen Programmteile auch dessen Ergebnisse stehen, nämlich die im Bild gefundenen Bälle und der von diesen für die Repräsentation *BallPercept* gewählte Ball sowie um ein Viertel ihrer Auflösung verringerte Versionen der vom *NAO* gesehenen Graustufenbilder.

Hierbei werden die vom Modul gesehenen und durch manuelle Betrachtung des Bilds bestätigten tatsächlichen Bälle (*true positives*), die gesehenen, aber nicht wirklich vorhandenen, Bälle (*false positives*) und die nicht gesehenen, aber im Bild vorhandenen, Bälle (*false negatives*) gezählt. Entscheidend bei der Zählung sind dabei nicht die vom Modul selektierten Ballperzepte, sondern alle gesehenen Bälle vor dem Selektionsschritt. Das Modul kann dann als ausreichend korrekt angesehen werden, wenn die Anzahl der *true positives* möglichst hoch

im Verhältnis zu denen der *false positives* und der *false negatives* ist. Ein Beispiel für die Klassifikation eines in der Logdatei gespeicherten Bilds zeigt Abbildung 6.4.

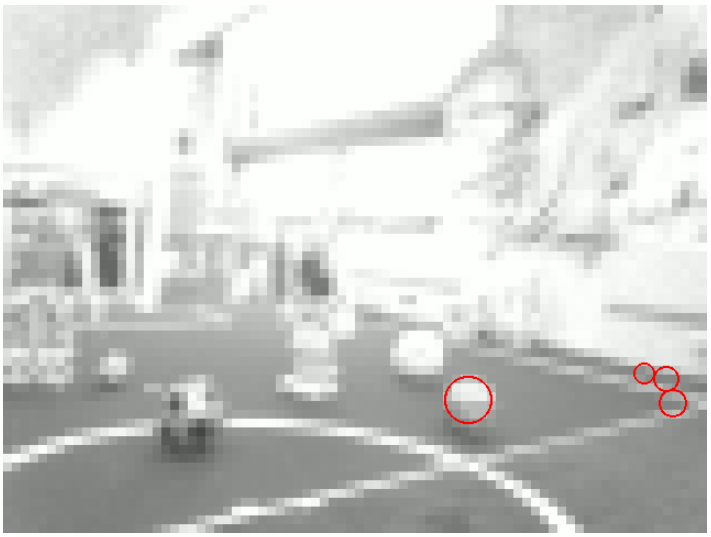

**Abbildung 6.4** Beispiel für das Vorgehen in der Evaluation: dieses Bild zeigt 1 *true positive* und 3 *false negatives* sowie am rechten Bildrand 3 *false positives*.

Die Echtzeitfähigkeit wird dadurch geprüft, dass die Laufzeiten des Moduls während des Experiments angesehen werden und dabei die absolut minimalen, maximalen und durchschnittlichen Laufzeiten berechnet werden. Außerdem wird die Bildwiederholrate des Cognition-Prozesses, also die Anzahl der von diesem pro Sekunde verarbeiteten Bilder, geprüft. Im Idealfall liegt diese bei 60 Hz, da jede Kamera pro Sekunde 30 Bilder liefert und somit in diesem Fall jedes Bild verarbeitet werden kann.

### 6.2.2 Ergebnisse

Tabelle 6.2 zeigt die Anzahl der *true positives*, *false negatives* und *false positives*, die während eines dreiminütigen Testlaufs in 5686 Bildern gesehen wurden. Dabei wird deutlich, dass das Verhältnis von tatsächlich korrekt gesehenen Bällen zu den vorhandenen Bällen sowie auch zu den fälschlicherweise gesehenen Bällen recht gering ist. Die Erkennungsrate von Bällen beträgt nur $\frac{246}{246+2489} \approx 9\%$ und es werden ungefähr 5,67 mal so viele falsche Bälle wie korrekte Bälle gesehen. Das Modul kann demnach nicht als korrekt angesehen werden.

| | |
|---|---|
| **true positives** | 246 |
| **false negatives** | 2489 |
| **false positives** | 1396 |

**Tabelle 6.2** Ergebnisse der Analyse der Logdatei hinsichtlich der Korrektheit des entwickelten Moduls.

Die falsch erkannten Bälle liegen zum überwiegenden Teil einerseits in mehrere Meter entfernten Feldlinien und andererseits in vom B-Human-System nicht korrekt erkannten Robotern.

Es werden allerdings die meisten durch die Ballerkennung gefundenen *false positives*, da sie in der Regel weit vom zuletzt gesehenen Ball entfernt sind, durch die verwendete Methode zur Ballselektion nicht in das *BallPercept* eingetragen und haben damit keinen negativen Effekt. Umgekehrt werden *false positives* im Gegensatz zu echten Bällen selten mehrmals an denselben Orten gesehen, sodass ein einmal gesehener und selektierter *false positive* die Ballfindung nur kurzzeitig behindert. Ein negativer Effekt der *false positives* ist aber durchaus deutlich erkennbar.

Bei der Analyse der Logdatei hat sich davon abgesehen herausgestellt, dass große Bälle häufig nicht gesehen wurden, da sie vom B-Human-System als Hindernisse (siehe Abbildung 6.5) erkannt wurden und sie daher, sofern die Ballerkennung sie erkannt hätte, von der Plausibilitätsprüfung ausgeschlossen wurden (vgl. 5.4). Überdies werden in Bällen mit komplexer Textur oft mehrere Bälle gesehen (vgl. Abbildung 6.6). Vermutlich liegt dies daran, dass die Rundungen in der Textur vom FRHT-Algorithmus als einzelne Kreise erkannt werden.

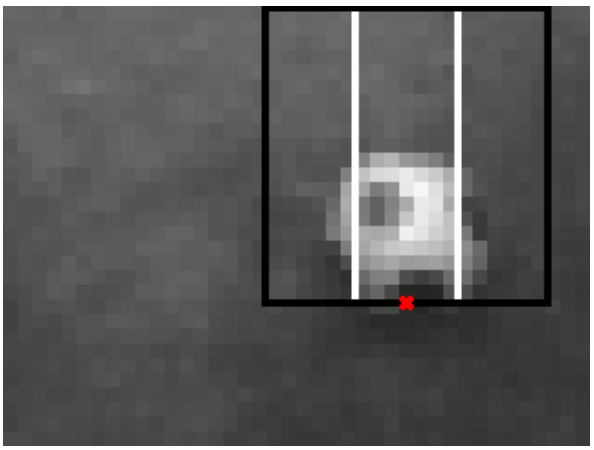

**Abbildung 6.5** Große Bälle werden oft als Hindernisse erkannt. Das schwarze Rechteck zeigt ein vom B-Human-System erkanntes Hindernis.

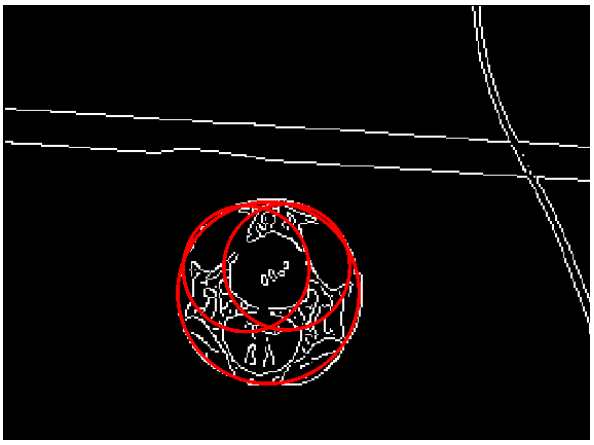

**Abbildung 6.6** In Bällen mit komplexer Textur werden oft mehrere Bälle gesehen. Ausschnitt eines Konturbilds mit rot eingezeichneten Kreisen an den Stellen, an denen Bälle erkannt wurden.

| | **minimal** | **maximal** | **durchschnittlich (ungefähr)** |
|---|---|---|---|
| Laufzeit obere Kamera | 1,558 ms | 84,903 ms | 15-30 ms |
| Laufzeit untere Kamera | 1,259 ms | 22,931 ms | 2,5-6 ms |
| Bildwiederholrate | 23,5 Hz | 43,6 Hz | |

**Tabelle 6.3** Laufzeiten der Ballerkennung während des Testlaufs.

Die Laufzeiten des Moduls während des Testlaufs zeigt Tabelle 6.3. Es zeigt sich, dass zu keinem Zeitpunkt die ideale Bildwiederholrate von 60 Hz erreicht wird, sondern in der Regel nur jedes zweite von einer Kamera eingefangene Bild verarbeitet werden kann. Die Laufzeit der Ballerkennung steigt mit der Komplexität des Bildes, also der Anzahl und Länge von Kanten im Bild. Dabei treten die höchsten Werte auf, wenn andere *NAO*s groß im Bild zu sehen sind.

Basierend auf einer durchschnittlichen Laufzeit von 15 ms kann die Ballerkennung im Idealfall bis zu $\frac{1\text{ s}}{15\text{ ms}} \approx 66{,}67$ Bilder pro Sekunde verarbeiten. Die niedrigere Bildwiederholrate liegt daran, dass neben dem implementierten Modul auf der Rest des Cognition-Threads ausgeführt wird. Demnach ist das implementierte Modul zur Ballerkennung nicht echtzeitfähig im Sinne des B-Human-Systems. Die Frequenz, mit der Bilder verarbeitet werden, ist aber ausreichend, um Bälle finden und verfolgen zu können.

## 6.3 Realistic Ball Challenge

Die RBC fand wie geplant beim RoboCup 2015 in Hefei statt und das Team B-Human trat mit einem Roboter an, auf welchem das im Kontext dieser Arbeit entwickelte Modul zur Ballerkennung lief. Während der vorgegebenen drei Minuten wurden in der RBC vom *NAO* des Teams B-Human vier der fünf Bälle gesehen und in Richtung eines der Tore bewegt sowie einer dieser Bälle in ein Tor geschossen, also gemäß der Regeln insgesamt sieben Punkte erzielt. Damit erreichte B-Human den zweiten Platz hinter den Nao Devils Dortmund, welche neun Punkte erzielten, und vor den Teams WrightOcean und Austrian Kangaroos mit jeweils zwei Punkten.

Das Verhalten des Roboters von B-Human während der RBC wurde gefilmt und kann unter `https://www.youtube.com/watch?v=V0tVxfKCspw` angesehen werden. Eine Momentaufnahme kurz vor dem erfolgreichen Torschuss ist in Abbildung 6.7 zu sehen.

Obwohl die vorliegende Implementierung wie zu Beginn von Kapitel 5 beschrieben leicht von der in der RBC verwendeten abweicht, ist durch dieses Ergebnis gezeigt, dass die in dieser Arbeit beschriebene Lösung durchaus geeignet bezüglich des Ziels der Arbeit ist.

**Abbildung 6.7** Der Roboter des Teams B-Human (mittig) während der RBC.

# Kapitel 7

# Zusammenfassung und Ausblick

Ziel der vorliegenden Arbeit war ein Modul zur Erkennung beliebiger Bälle durch den humanoiden Roboter *NAO*. Hierzu wurde ein Modul im B-Human-Framework geschrieben, das Bälle im Kamerabild findet und einen dieser Bälle zur Verwendung im restlichen System zurückgibt. Dafür erstellt das Modul zunächst aus dem Kamerabild ein Konturbild und findet anschließend mithilfe des FRHT-Algorithmus Kreise in diesem. Aus den gefundenen Kreisen werden durch mehrere Validierungsschritte potentielle Bälle gefunden, von denen derjenige selektiert wird, welcher entweder nahe dem zuvor verfolgten Ball ist oder dem *NAO* am Nächsten liegt.

Zur Erstellung eines Konturbilds aus dem Kamerabild wurden drei verschiedene Methoden entwickelt und evaluiert. Bei der Evaluation zeigte sich, dass die besten Ergebnisse für den gegebenen Anwendungsfall erzielt werden, wenn das in Graustufen vorliegende Kamerabild zunächst mit einem Gauß-Filter geglättet wird, dann daraus durch Anwendung des Sobel-Operators eine Gradientenkarte erstellt wird und anschließend mithilfe einer Nicht-Maximum-Unterdrückung aus den Punkten mit den stärksten Gradienten das Konturbild gebildet wird.

Das entwickelte Modul wurde hinsichtlich seiner Echtzeitfähigkeit und seiner Korrektheit überprüft. Hierbei stellte sich bezüglich der Laufzeit heraus, dass diese in der Regel derart lang ist, dass das B-Human-System nicht mehr sämtliche von den Kameras gelieferten 60 Bilder pro Sekunde verarbeiten kann. Dadurch wird das B-Human-System weniger reaktiv. In einem Spiel der SPL wäre dies von Nachteil.

Bezüglich der Korrektheit wurde gezeigt, dass das Modul eine recht schlechte Erkennungsrate von Bällen hat und überdies weit mehr Falscherkennungen als echte Bälle liefert. Es gibt jedoch durch die gewählte Methode der Ballselektion trotz dieser Probleme ausreichend gute Informationen heraus, um ein einfaches Spielverhalten zu ermöglichen.

Insgesamt ist das entwickelte Modul also nicht für den Einsatz in einem normalen Fußballspiel der SPL geeignet. Allerdings bietet es, wie auch die Performanz bei der RBC auf dem RoboCup 2015 zeigt, durchaus eine akzeptable Lösung für die konkrete Problemstellung, die die Motivation dieser Arbeit bildet.

Die RBC wurde wie in 2.2 benannt veranstaltet, um die Einführung eines vom bisher verwendeten einfarbig orangenen abweichenden Balls zu evaluieren. Demnach stellt sich die Frage, wie das entwickelte Modul für einen solchen Fall angepasst werden könnte. Die verwendeten Verfahren zur Kreiserkennung, Validierung und Selektion treffen viele Annahmen bezüglich der Größe der verwendeten Bälle und würden demnach bessere Ergebnisse liefern, wenn bereits im Vorhinein die Größe eines zu erkennenden Balls bekannt wäre. Überdies wählt das FRHT-Verfahren bisher zufällige Konturpunkte als Ausgangspunkte der Kreiserkennung. Ist die Textur des zu findenden Balls bekannt, so können ähnlich dem für den bisherigen Ballerkenner des Teams B-Human verwendeten Verfahren bereits in einem ersten Schritt potentielle Bälle basierend auf ihrer Textur im Bild gefunden und deren Konturpunkte bevorzugt als Ausgangspunkte für die Kreiserkennung verwendet werden. Eine angepasste Version zur Erkennung eines festgelegten Balls der Größe FIFA Size 1 wurde implementiert, da ein solcher zunächst als Ball für die Spiele in der SPL im Jahr 2016 im Gespräch war. Diese Version lieferte deutlich bessere Ergebnisse als das für diese Arbeit implementierte Modul.

Für die Erkennung des letztendlich im Jahr 2016 genutzten Balls wurde jedoch keine Version des beschriebenen Verfahrens genutzt, da dieses für den verwendeten Ball schlechtere Ergebnisse als der letzthin genutzte Ballerkenner sowohl hinsichtlich der Laufzeit als auch der Korrektheit lieferte.

# Literaturverzeichnis

[1] B-Human: *B-Human | RoboCup Standard Platform League. Offizielle Webseite des Teams B-Human.* https://www.b-human.de/?lang=de, . – Abgerufen am 03.02.2016

[2] RoboCup Technical Committee: *RoboCup Standard Platform League (NAO) Rule Book (2015 rules, as of July 1, 2015).* 2015. – Online verfügbar unter: http://www.informatik.uni-bremen.de/spl/pub/Website/Downloads/Rules2015.pdf

[3] RoboCup Technical Committee: *RoboCup Standard Platform League (NAO) Technical Challenges.* 2015. – Online verfügbar unter: http://www.informatik.uni-bremen.de/spl/pub/Website/Downloads/Challenges2015.pdf

[4] The Robocup Federation: *A Brief History of RoboCup.* http://www.robocup.org/about-robocup/a-brief-history-of-robocup/, . – Abgerufen am 16.06.2016

[5] The Robocup Federation: *Objective.* http://www.robocup.org/about-robocup/objective/, . – Abgerufen am 16.06.2016

[6] RoboCup Technical Committee: *RoboCup Standard Platform League (NAO) Rule Book (2016 rules, as of June 9, 2016).* 2016. – Online verfügbar unter: http://www.informatik.uni-bremen.de/spl/pub/Website/Downloads/Rules2016.pdf

[7] Aldebaran Robotics: *H25 - Construction — Aldebaran 2.1.4.13 documentation.* http://doc.aldebaran.com/2-1/family/nao_h25/dimensions_h25.html, . – Abgerufen am 16.06.2016

[8] Aldebaran Robotics: *H25 - Joints — Aldebaran 2.1.4.13 documentation.* http://doc.aldebaran.com/2-1/family/nao_h25/joints_h25.html, . – Abgerufen am 16.06.2016

[9] Aldebaran Robotics: *NAO - Video camera — Aldebaran 2.1.4.13 documentation.* http://doc.aldebaran.com/2-1/family/robots/video_robot.html, . – Abgerufen am 20.06.2016

[10] Aldebaran Robotics: *NAO H25 — Aldebaran 2.1.4.13 documentation.* http://doc.aldebaran.com/2-1/family/nao_h25/index_h25.html, . – Abgerufen am 20.06.2016

[11] Aldebaran Robotics: *Motherboard — Aldebaran 2.1.4.13 documentation.* http://doc.aldebaran.com/2-1/family/robots/motherboard_robot.html, . – Abgerufen am 20.06.2016

[12] Röfer, Thomas ; Laue, Tim ; Müller, Judith ; Schüthe, Dennis ; Böckmann, Arne ; Jenett, Dana ; Koralewski, Sebastian ; Maass, Florian ; Maier, Elena ; Siemer, Caren ; Tsogias, Alexis ; Vosteen, Jan-Bernd: *B-Human Team Report and Code Release 2014.* 2014. – Only available online: http://www.b-human.de/downloads/publications/2014/CodeRelease2014.pdf

[13] Röfer, Thomas ; Laue, Tim ; Müller, Judith ; Bartsch, Michel ; Batram, Malte J. ; Böckmann, Arne ; Böschen, Martin ; Kroker, Martin ; Maass, Florian ; Münder, Thomas ; Steinbeck, Marcel ; Stolpmann, Andreas ; Taddiken, Simon ; Tsogias, Alexis ; Wenk, Felix: *B-Human Team Report and Code Release 2013.* 2013. – Only available online: http://www.b-human.de/downloads/publications/2013/CodeRelease2013.pdf

[14] Röfer, Thomas ; Laue, Tim ; Müller, Judith ; Bösche, Oliver ; Burchardt, Armin ; Damrose, Erik ; Gillmann, Katharina ; Graf, Colin ; Haas, Thijs J. ; Härtl, Alexander ; Rieskamp, Andrik ; Schreck, André ; Sieverdingbeck, Ingo ; Worch, Jan-Hendrik: *B-Human Team Report and Code Release 2009.* 2009. – Only available online: http://www.b-human.de/downloads/bhuman09_coderelease.pdf

[15] Hanek, Robert ; Schmitt, Thorsten ; Buck, Sebastian ; Beetz, Michael: Towards RoboCup without Color Labeling. In: *RoboCup 2002: Robot Soccer World Cup VI*, 2002, 179–194

[16] Coath, Genevieve ; Musumeci, Phillip: Adaptive arc fitting for ball detection in robocup. In: *APRS Workshop on Digital Image Analysing*, 2003, S. 63–68

[17] Martins, Daniel A. ; Neves, António JR ; Pinho, Armando J.: Real-time generic ball recognition in RoboCup domain. In: *Proc. of the 3rd International Workshop on Intelligent Robotics, IROBOT*, 2008, S. 37–48

[18] Burger, Wilhelm ; Burge, Mark J.: *Digitale Bildverarbeitung: eine algorithmische Einführung mit Java.* 3., vollst. überarb. und erw. Aufl. Berlin [u.a.] : Springer Vieweg, 2015 (X.media.press). – ISBN 9783642046032

[19] Priese, Lutz: *Computer Vision: Einführung in die Verarbeitung und Analyse digitaler Bilder.* Berlin : Springer Vieweg, 2015 (eXamen.press). http://dx.doi.org/10.1007/978-3-662-45129-8. – ISBN 9783662451298

[20] Canny, John: A computational approach to edge detection. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1986)

[21] Snyder, Wesley E. ; Qi, Hairong: *Machine vision.* Cambridge : Cambridge Univ. Press, 2004. – ISBN 9780521830461

[22] Bourke, Paul: *Equation of a Circle from 3 Points (2 dimensions).* http://paulbourke.net/geometry/circlesphere/, 1990. – Abgerufen am 28.07.2016

[23] Steger, Carsten ; Ulrich, Markus ; Wiedemann, Christian: *Machine vision algorithms and applications.* Weinheim : Wiley-VCH-Verl., 2008 (Wiley-VCH textbook). – ISBN 9783527407347

[24] Böckmann, Arne: *Freiraum- und Robotererkennung mit dem humanoiden Roboter NAO*, Universität Bremen, Bachelorarbeit, 2013

[25] Röfer, Thomas ; Laue, Tim: On B-Human's Code Releases in the Standard Platform League - Software Architecture and Impact. In: *RoboCup 2013: Robot Soccer World Cup XVII*, Springer, 2014 (Lecture Notes in Artificial Intelligence)

[26] Topal, Cihan ; Akinlar, Cuneyt: Edge Drawing: A combined real-time edge and segment detector. In: *Journal of Visual Communication and Image Representation* 23 (2012), Nr. 6, 862 - 872. http://dx.doi.org/10.1016/j.jvcir.2012.05.004. – DOI 10.1016/j.jvcir.2012.05.004. – ISSN 1047–3203

[27] Intel Corporation: *Intel® Atom™ Processor Z530 (512K Cache, 1.60 GHz, 533 MHz FSB) Spezifikationen.* – Online verfügbar unter: http://ark.intel.com/de/products/35463/Intel-Atom-Processor-Z530-512K-Cache-1_60-GHz-533-MHz-FSB

[28] Intel Corporation: *Intel Intrinsics Guide.* – Online verfügbar unter: https://software.intel.com/sites/landingpage/IntrinsicsGuide/#techs=SSE,SSE2,SSE3,SSSE3

[29] Chiu, Shih-Hsuan ; Liaw, Jiun-Jian ; Lin, Kuo-Hung: A FAST RANDOMIZED HOUGH TRANSFORM FOR CIRCLE/CIRCULAR ARC RECOGNITION. In: *International Journal of Pattern Recognition and Artificial Intelligence* 24 (2010), Nr. 03, 457-474. http://dx.doi.org/10.1142/S0218001410007956. – DOI 10.1142/S0218001410007956