Technische Universität Hamburg-Harburg
Vision Systems

Prof. Dr.-Ing. R.-R. Grigat

Lokalisierung von Torpfosten bei variablen Hintergrundtexturen auf dem NAO Robotik System

Projektarbeit

Niklas Hollstegge

15. September 2017

*Eidesstattliche Erklärung

Ich, Niklas Hollstegge, geboren am 16.09.1993 in Münster, versichere hiermit an Eides statt, dass ich diese von mir bei der Technischen Universität Hamburg-Harburg (TUHH) vorgelegte Projektarbeit selbstständig verfasst habe. Ich habe ausschließlich die angegebenen Quellen und Hilfsmittel benutzt.

Ort und Datum

Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

TUHH Technische Universität Hamburg

HULKs Hamburg Ultra Legendary Kickers

SPL Standard Platform League

SIFT Scaling Invariant Feature Transform

SURF Speeded Up Robust Features

RE Steigende Flanke, von *Rising Edge*

FE Fallende Flanke, von *Falling Edge*

TP True Positive

FP False Positive

1 Einleitung

Diese Arbeit ist im Rahmen der Hamburg Ultra Legendary Kickers (HULKs) Robotik AG an der Technischen Universität Hamburg-Harburg (TUHH) geschrieben worden. Diese verfolgt den Aufbau einer Fußballmannschaft humanoider Roboter für die Teilnahme am internationalen RoboCup.

1.1 Motivation

Die Positionsbestimmung ist eine der essentiellen Aufgaben in der humanoiden Robotik und von enormer Bedeutung für Navigation und Aktion autonom agierender Systeme. Der größte Informationsfluss der Sensorik des Roboters entspricht typischerweise der visuellen Wahrnehmung der Umgebung durch Kameras, weshalb die Bildverarbeitung ein fundamentaler Bestandteil der Robotik ist.

Eine absolute Positionsbestimmung ist immer nur an Hand statischer Objekte möglich, zu denen eine Referenz hergestellt und Bildpunkte zu Weltkoordinaten transformiert werden können. Auf einem Fußballfeld sind nur sehr wenige statische Merkmale vorhanden. Die aktuelle Selbstlokalisierung erfolgt durch Detektion der Feldlinien. Diese sind einerseits nicht fortlaufend sichtbar und lassen andererseits nur selten auf eine exakte Position auf dem Spielfeld rückschließen. Neben den Feldlinien ist lediglich das Tor als statisches Objekt auf der Spielfläche vorhanden und stellt somit eine Möglichkeit zur Unterstützung der Selbstlokalisierung dar. Eine erfolgreiche Pfostendetektion stabilisiert somit die Selbstlokalisierung und relativiert Fehler der Feldlinienerkennung.

Bis zum Jahr 2015 waren Tore mit gelber und blauer Farbe markiert, wodurch eine Torpfostendetektion durch farbbasierte Filterung mit wenig Aufwand implementiert werden konnte. Nachdem die Regularien seitdem von farbigen Toren absehen, wurde im HULKs-Framework sowie beim Großteil aller Mitbewerber keine Tordetektion mehr verwendet.

Beliebig zugelassene Hintergrundtexturen und offenere Regularien für Umwelt- und Lichteinflüsse erfordern eine detaillierte und aufwändigere Entwicklung eines Verfahrens zur Torlokalisierung, mit welchem letztendlich ein Wettbewerbsvorteil gegenüber anderen Teams erreicht und in den zukünftigen Turnieren Einfluss auf den Ausgang des Spiels genommen wird.

1.2 Der RoboCup

Der RoboCup ist ein jährlich stattfindendes Turnier, bei dem weltweit oder national Teams teilnehmen, um ihre Robotikforschung in verschiedenen Schwerpunkten zu präsentieren. Die Standard Platform League (SPL) ist ein Teil des RoboCups, bei der humanoide Roboter gegeneinander Fußball spielen. Abbildung 1.1 zeigt die Umgebungsbedingungen während der *German Open 2017* und das bereits große Interesse seitens der Zuschauer. Im Rahmen dieses Wettbewerbs steht allen Teams die gleiche Hardware zur Verfügung. Die Roboter dürfen keinem menschlichen Einfluss oder dezentralen Berechnungen unterliegen. Das Ziel des RoboCups ist es bis 2050 ein Roboterteam zu entwickeln, das gegen den menschlichen Fußballweltmeister gewinnen kann[23].

Abbildung 1.1: Umgebungsbedingungen bei den RoboCup German Open 2017 [4]

1.3 Ziele und Struktur

Das Ziel dieser Arbeit ist die Entwicklung eines echtzeitfähigen Verfahrens zur Lokalisierung von Torpfosten für aktuelle und zukünftige Wettbewerbe des RoboCups. Dabei sollen variable (vor allem weiße) Hintergrundtexturen und unscharfe Objekte sowie Aufnahmen ohne Querlatte erfolgreich verarbeitet werden können. Folgende drei quantitative Ziele werden in dieser Arbeit verfolgt:

- Eine **Erkennung von mindestens 25%** der aus der tornahen Spielfeldhälfte aufgenommenen Torpfosten ist wünschenswert. Zur Lokalisation ist lediglich das untere Pfostenende zu bestimmen.
- Da dem NAO in jedem Zyklus 10 ms für alle Module der Bildverarbeitung zur Verfügung stehen, sollte die **Laufzeit des Moduls 2 ms** nicht übersteigen, um den nachfolgenden Modulen ausreichend Rechenzeit zu erhalten. Eine höhere Rechenzeit ist akzeptabel, wenn auch die Erkennungsrate größer ist. In diesem Fall kann eine Verarbeitung von nur jedem zweiten oder drittem Bild in Betracht gezogen werden.
- Zieldefinierend ist ebenso die Wahrscheinlichkeit von Falschaussagen. Auf Grund der probabilistischen Lokalisierungsfilter des NAOs führen falsch generierte Informationen nicht implizit zu einer falschen Positionsbestimmung. Dennoch sollten **falsch lokalisierte Torpfosten in maximal 3% der Aufnahmen** vorkommen.

Die Anforderung nach einer pixelgenauen Positionsangabe der Torpfosten innerhalb des Bildes ist nicht gefordert, da dieser Fehler gegenüber der Fehler der nachfolgenden Transformationen von Bildkoordinaten in die Weltkoordinaten des Roboters zu vernachlässigen ist. Diese Ermittlung der globalen Roboterposition aus den Bildpixeln ist einem bestehenden Modul zu entnehmen und nicht Bestandteil dieser Arbeit.

Die hier vorgelegte Arbeit ist in drei Hauptbestandteile gegliedert (vgl. Abbildung 1.2). Die **Generierung** dient der Reduzierung der Bilddaten, indem nur Teile der Aufnahme ermittelt werden, in denen Torpfosten liegen können. Die Aufgabe der **Lokalisierung** besteht aus der Detektion des unteren und oberen Pfostenendes innerhalb dieser Bereiche. Anschließend werden für jeden Pfostenkandidaten in der **Bewertung** mehrere Kriterien geprüft und eine Gesamtbewertung ermittelt. An Hand dieser werden die Kandidaten endgültig als (kein) Torpfosten klassifiziert.

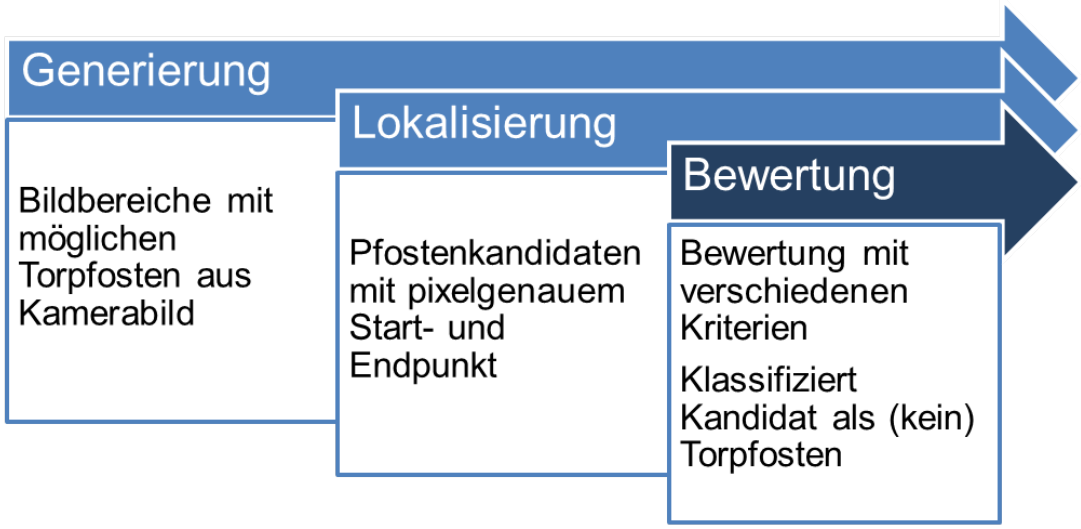

Abbildung 1.2: Teilbereiche des in dieser Arbeit entwickelten Verfahrens

2 Grundlagen

2.1 Der NAO Roboter

Der NAO ist ein humanoider Roboter des französischen Unternehmens *Aldebaran*. Er wurde für vielfältige Einsatzzwecke, insbesondere für menschliche Interaktion, entwickelt und wird u.a. seit zehn Jahren in der SPL für den RoboCup eingesetzt [22].

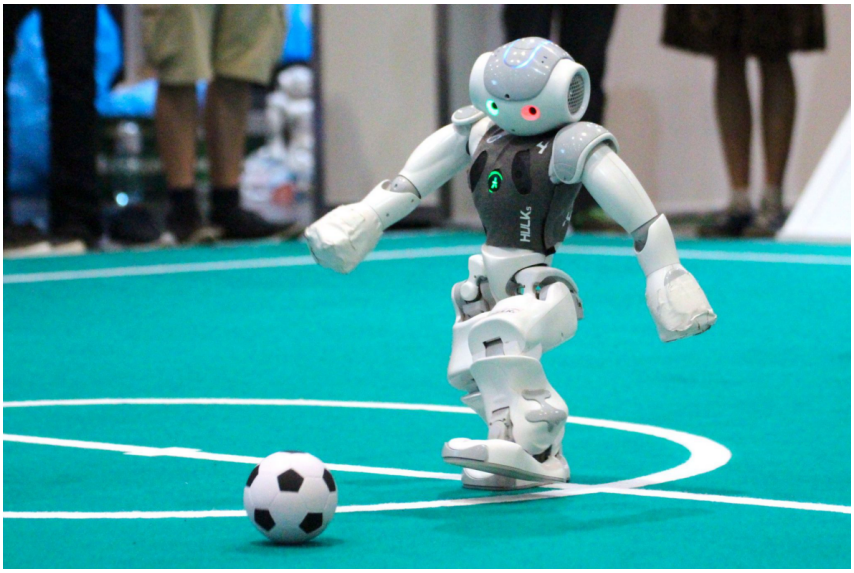

Abbildung 2.1: NAO Roboter beim RoboCup 2016 in Leipzig [20]

Die in dieser Arbeit verwendete Version des NAOs (5.0) ist 57,3 cm hoch und wiegt 5,2 kg. Es liegen 25 Freiheitsgrade und eine Ausstattung mit diverser Hardware vor (u.a. Gyroskop, Beschleunigungssensor, Ultraschallsensoren, Drucksensoren). Im Kopf befindet sich ein Intel ATOM Z530 mit 1,6 GHz, sowie 1 GB Arbeitsspeicher. Die Kommunikation mit dem NAO erfolgt über Ethernet oder Wireless LAN [24].

2.2 Kamera

Der NAO besitzt zwei unterschiedlich ausgerichtete Kameras, jeweils eine im Stirn- und Mundbereich. Hiermit ist sichergestellt, dass sowohl weiter entfernte Objekte als auch der Bereich vor den Füßen wahrgenommen wird. Da sich die Sichtfelder nur in geringem Maße überschneiden, ist eine stereoskopische Bildverarbeitung nicht möglich. Aus beiden Kameras wird lediglich die Obere verwendet, da nur diese hoch genug ausgerichtet ist, um das Tor zu erfassen.

Die Kameras ermöglichen eine Auflösung von bis zu 1280x960 Pixeln (1,22 Megapixel) und eine Bildrate von 29 bis 30 fps. Derzeit wird im HULKs-Framework eine Auflösung von 640x480 Pixeln verarbeitet. Die Bilder werden im *Electronic Rolling Shutter*-Modus aufgenommen, wodurch die Aufnahmen bei gleichzeitiger Bewegung Unschärfe enthalten [11, S.402].

Die Rohdaten der Kamera werden im *YUV422*-Farbmodell ausgegeben [24]. Dabei beschreibt der Y-Wert die Helligkeit, bzw. Luminanz eines Pixels, während die U- und V-Werte die Farbigkeit, bzw. Chrominanz festlegen. U gibt den Verlauf von Gelb zu Blau, V von Türkis zu Rot an. Jede der drei Werte liegt im *uint8-Datentyp* zwischen den Werten 0 und 255. *YUV422* ist ebenso bekannt als *YCbCr 4:2:2*-Farbraum, welcher eine skalierte und digitalisierte Version von *YUV* repräsentiert [1]. Im Folgenden werden daher die Termini Y-, Cb- und Cr-Kanal verwendet. Der Vorteil gegenüber additiven Farbmodellen wie RGB liegt darin, dass naturelle Beleuchtungsänderungen meist nur Auswirkungen auf die Helligkeit, jedoch nicht auf die Chrominanz haben. Somit ist die Detektion einer Farbe mathematisch besser einzugrenzen. Im Rahmen dieser Arbeit besitzt die Bestimmung bestimmter Farbtöne keine Relevanz [19].

Abbildung 2.2 zeigt das Farbmodell in Abhängigkeit der beiden Chrominanzwerte Cb und Cr bei konstanter Helligkeit Y. Es ist zu erkennen, dass ein maximaler Helligkeitswert nicht zwingend ein Weißton impliziert, sondern u.a. auch Farbtöne wie Gelb oder Grün inkludiert. In dieser Arbeit kann die Bildverarbeitung auf Grund der Anforderung nach Farbunabhängigkeit bis auf das Kapitel der Generierung auf die Luminanz Y beschränkt werden. Diese Information reicht zur Unterscheidung von dunklen und hellen Elementen im Bild.

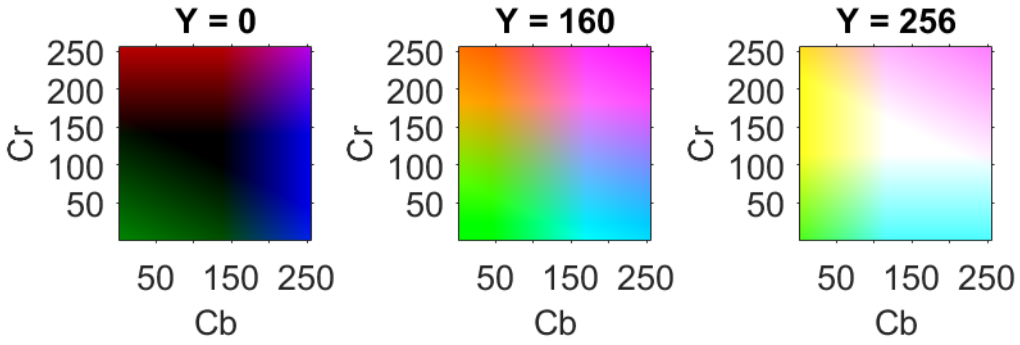

Abbildung 2.2: Ebenen des YCbCr-Farbmodells bei konstanter Luminanz Y

2.3 Relevante Regularien

Die Regularien für das Jahr 2017 [21] beinhalten erstmals natürliche und wechselnde Lichtverhältnisse als Bestandteil des RoboCups zu akzeptieren. Eine Beleuchtungsstärke von 300 Lux sollte dabei nicht unterschritten werden. Das Verhältnis des hellsten gegenüber des dunkelsten Punktes des Spielfeldes sollte maximal 10:1 betragen. Jedoch wird explizit darauf hingewiesen, dass eine homogene Beleuchtung nicht garantiert wird. Dies inkludiert ebenso die Änderung von Sonnenlicht wie auch Ausfallen von künstlicher Beleuchtung.

Das Feld ist 10,4 m lang und 7,4 m breit.

Das in dieser Arbeit zu detektierende Tor ist in Abbildung 2.3 dargestellt. Die Torpfosten besitzen einen Durchmesser von 100 mm, sind 800 mm hoch und die Innenseiten besitzen einen Abstand von 1500 mm. Der Durchmesser der Querlatte beträgt ebenso 100 mm. Während die Tore in früheren Jahren in gelb und blau angemalt waren, sieht das Regelwerk seit 2015 weiße Tore vor. Die Stützbalken zur Fixierung des Tornetzes sind 800 mm hoch und orthogonal, bzw. parallel zu Torpfosten und Querlatte. Dieses Gerüst kann im Gegensatz zu Pfosten und Querlatte auch grau oder schwarz sein.

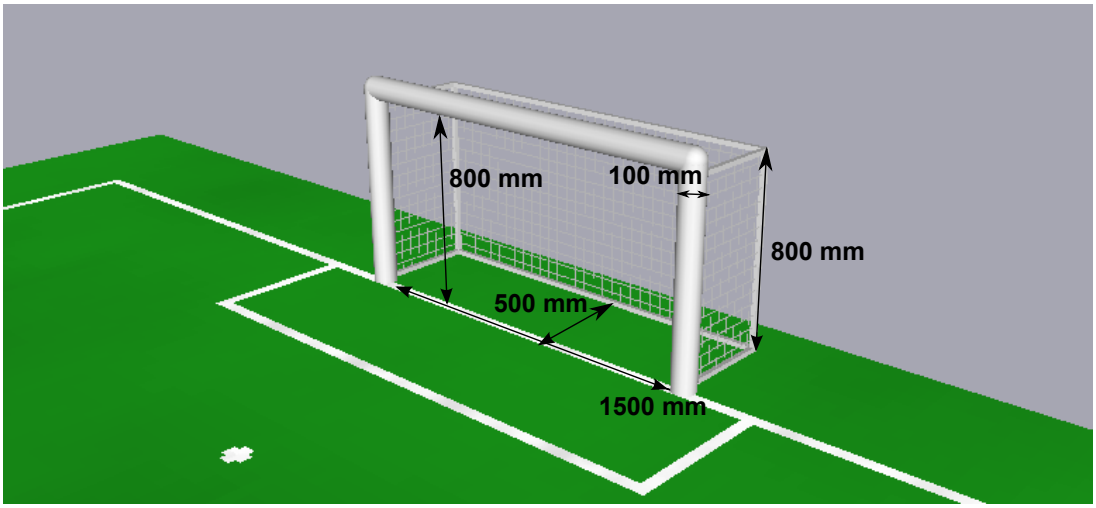

Abbildung 2.3: Dimensionen des Tores nach den SPL RoboCup Regularien für 2017

Der Bereich hinter dem Tor unterliegt keinen Bestimmungen und kann daher jegliche Helligkeits- und Farbwerte annehmen. Während offiziellen Turnieren stehen meist Zuschauer sowie Werbebanden im Hintergrund des Tores (vgl. Abbildung 1.1).

3 Generierung von Bereichen mit Pfostenkandidaten

3.1 Dateneingänge

Im Folgenden sollen die für diese Arbeit potentiell hilfreichen Dateneingänge neben den Kameradaten vorgestellt werden.

Dies sollten Informationen über Objekte oder Elemente sein, die statisch zum Torpfosten positioniert sind, um eine bessere und recheneffizientere Torpfostenerkennung zu gewährleisten. Die Voraussetzung dafür ist jedoch ein zuverlässiger Informationsgehalt, sodass eine Fehlerfortpflanzung innerhalb des Systems vermieden wird.

Die Feldlinien sind ein Beispiel solcher Elemente. Jedoch stützt sich die aktuell implementierte Lokalisierung des NAOs bereits auf diese Daten, wodurch eine Nutzung dieser Informationen zu verlorener Redundanz führt.

Eine weitere Möglichkeit zur Unterstützung der Tordetektion stellt die Feldgrenze[1] dar, welche farbbasiert die Grenze der grünen Spielfläche um das Feld herum ermittelt. Nach Betrachtung von Videosequenzen aus dem Versuchslabor stellt sich heraus, dass die Feldgrenze in verschiedenen Orientierungen und bei Gegenlicht zum Teil dauerhaft falsche Informationen generiert.

Als zuverlässiger stellt sich der Horizont heraus, welcher sich aus Gyroskop- und Gelenkdaten ermitteln lässt. Wird angenommen, dass die Fußsohle des NAOs die gleiche Ebene wie das Spielfeld darstellt, kann über die Gelenkwinkel errechnet werden, wo aus Sicht der Kamera sich diese Ebene im Unendlichen erstrecken würde. Auch bei Bewegungen des NAOs zeigt sich, das sich der Horizont immer über der Spielfeldgrenze und unter der Querlatte befindet, womit eine hilfreiche Unterstützung gegeben ist. Der Horizont kann und wird in dieser Arbeit als Referenz zur Suche der Torpfosten dienen.

3.2 Möglichkeiten zur Kandidatengenerierung

In diesem Kapitel sind die Ansätze anderer Teams, Dokumentationen und gängige Algorithmen zu nicht farbbasierter Objekterkennung zusammengefasst. Dabei ist die Zielsetzung mindestens die Generierung von Teilbereichen des Kamerabildes, in denen die realen Pfosten enthalten sind. Ansätze werden zunächst evaluiert und in Matlab getestet, um anschließend eine finale Implementierung auf dem NAO vorzunehmen.

[1] Das Feld erstreckt sich weiter als die weißen Außenlinien.

Kantendetektion mit Sobel Die Kanten sind ein markantes Merkmal der Torpfosten und geben Information über den Umriss des Tores. Der Sobelfilter ist ein weit verbreiteter Filter, der auf einer oder mehreren Faltungen beruht. Aus dem Originalbild (Abbildung 3.1, links) resultiert ein Gradientenbild (mitte). Der Sobelfilter liefert lediglich Grundinformationen und beinhaltet weiterhin viele ungewollte Merkmale [15]. Eine Nachbearbeitung (z.B. minimaler Schwellwert des Gradienten) reduziert den Informationsgehalt des Gradientenbildes und lässt sich dann zum Beispiel mit einer Hough-Transformation weiterverarbeiten. Diese untersucht in vorgegebenen Distanzen zum Ursprung sowie Orientierungen Liniensegmente und liefert anschließend Informationen über die Ausprägung einer Kante, sodass die grünen Linien (Abbildung 3.1, rechts) resultieren[2].

Abbildung 3.1: Originalbild (links) nach Sobel-Filterung (mitte) und Liniendetektion nach Hough (rechts)

Das Bremer Team „B-Human“ nutzte bis 2015 die in Kapitel 3.1 erläuterte Feldgrenze und suchte helle Regionen unterhalb dieser. Anschließend werden die Regionen als Roboter oder Pfosten klassifiziert, da andere Objekte nicht hinreichend groß seien. Dazu werden an diesen Stellen mit genannter Methode Linien erkannt und auf Parallelität und eingeschlossene Fläche geprüft[5]. Seit 2016 wurde von einer Tordetektion ohne Angabe von Gründe abgesehen [6].

Die „Nao Devils“ der TU Dortmund suchen mit einer nicht benannten, aber vermutlich ähnlichen Methode nach Kanten im ganzen Bild, um anschließend Höhe, Breite und einheitliche Farbe zwischen diesen zu bewerten. Als Unterstützung dienen die erkannten Feldlinien, die am unteren Pfostenende zu sehen sind [17]. Die Dokumentation zeigt auf, dass das Verfahren Mängel bei weißem Hintergrund aufweist.

Die Bachelorarbeit von Karl Tarvas [16] enthält ebenso Informationen über die Klassifizierung verschiedener Objekte im Rahmen des RoboCups mit Hilfe der hier erläuterten Kantendetektionen. Für die Erkennung von Toren werden Länge, Winkel und Anzahl der gefundenen Kanten bewertet. Anschließend wird eine Querlatte gesucht, die zwei Pfostenkandidaten verbindet und somit das Tor klassifiziert. Die dargestellten Bilder zeigen, dass ein farblich homogener Hintergrund für die Evaluation angenommen wurde und eine Querlatte immer im Bildbereich sein muss.

[2]Weitere Informationen zur Liniendetektion mit Hough finden sich in [15]

Spaltenhistogramm Ein Histogramm ist eine grafische Darstellung von Merkmalen und deren Häufigkeitsverteilung. Die Merkmale sind in dieser Methode die Pixel in einer gewissen Spalte, dessen Luminanz Y einen Schwellwert überschreitet (160 in Abbildung 3.2). Die Maxima der resultierenden Histogrammwerte[3] beschreiben Spalten, die besonders viele Weißwerte und demnach potentiell Bestandteil eines Torpfostens sind.

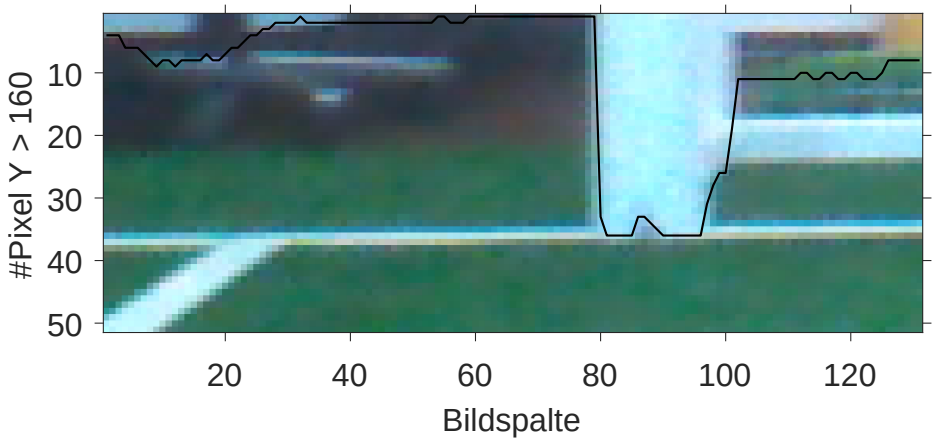

Abbildung 3.2: Spaltenhistogramm mit Originalbild im Hintergrund
Die vertikale Achse steigt nach unten und beschreibt die Anzahl der Pixel, dessen Luminanz über 160 liegt.

Dies wird u.A. in der Bachelorarbeit von Anastasia Bolotnikova genutzt[3]. Dort wird ebenso ein Zeilenhistogramm erstellt, jeweils große Steigungen als Kante klassifiziert und anschließend geschlossene Bereiche im Bild als Pfosten in Betracht gezogen. Die Feldgrenze dient als maximale Höhe für den Beginn des Pfostens. Die Autorin zeigt auf, dass hierbei oftmals andere Roboter oder die Feldlinien als Torpfosten resultieren. Es bedürfe weitere generelle Änderungen, um die Klassifizierung weiter einzugrenzen. Histogrammbasierte Tordetektionen waren bis 2015, vor Verabschiedung der farbigen Tore, weit verbreitet. Sie werden u.a. für die sogenannte *Four-Legged RoboCup League* [25] oder in Kombination mit einer Kantendetektion über Hough für die SPL [13] vorgestellt.

[3]Die Werte der vertikalen Achse der in dieser Arbeit gezeigten Spaltenhistogramme steigen in die untere Richtung

SIFT und SURF David G. Lowe beschreibt in seinem Paper [9] ein Verfahren, welches sich *Scaling Invariant Feature Transform* (SIFT) nennt und rotations- und skalierungsinvariant die geometrischen Merkmale zweier Bilder detektiert und anschließend aufeinander transformiert. In seiner Einleitung verspricht Lowe eine gewisse Affinität für dreidimensionale Projektionen, welche für den Fall der Torerkennung aus verschiedenen Positionen notwendig ist. Vorstellbar wäre ebenso ein Vergleich des Kamerabildes mit idealen Torabbildungen, die aus verschiedenen Positionen aufgenommen wurden.

Der zwei Jahre später entwickelte *Speeded Up Robust Features* (SURF)-Algorithmus verspricht eine recheneffizientere Merkmalsdetektion, welche sich mit den Ergebnissen des SIFT-Verfahrens messen könne. Abbildung 3.3 zeigt in Form von Linien die vermeintlich acht besten übereinstimmenden Merkmale beider Bilder. Detailliertere Informationen zu diesem Verfahren befinden sich in der Primärliteratur [14] und grundlagenorientierter in [15]. Ein Matlab-Code zum Austesten des Algorithmus und Generierung von Abbildung 3.3 findet sich in [10].

Abbildung 3.3: Merkmalssuche des *SURF*-Algorithmus

Machine Learning Für Aufgaben, für die ein statischer Algorithmus zu komplex oder gar unmöglich scheint, eignet sich *Machine Learning*. Hierbei wird kein analytischer Ansatz vom Benutzer implementiert, sondern eine „Black Box“ kreiert, die anhand vieler Trainingsdaten lernt, aus welchen Inputs welche Outputs folgen. Es ist denkbar das Kamerabild oder Teile dessen zur Klassifizierung als (kein) Tor zu verwenden. Hierbei ist die Entscheidungsfindung des Programms für den Benutzer nicht mehr nachvollziehbar, kann aber dennoch mit großen Datenmengen zum Lernen eine bessere Entscheidung treffen als ein analytischer Algorithmus [2]. Ein Versuch zur Objektklassifizierung mit Hilfe von *Machine Learning* auf dem NAO wurde im Rahmen der HULKs-AG von Olaf Lüders [18] durchgeführt.

3.3 Evaluierung und Bewertung der Kandidatengenerierung

Im Folgenden werden die vier erläuterten Verfahren zur Detektion von Pfostenbereichen evaluiert.

Die Echtzeitansprüche dieser Arbeit erfordern schnelle und effektive Algorithmen. Die Laufzeitmessungen des SIFT-Verfahrens [9] basieren auf einem Desktop-Computer von 1992, weswegen hier neuere Quellen betrachtet werden. Ein alternatives Paper [12] führt Rechenzeiten eines beschleunigten SIFT-Algorithmus für Echtzeitdetektion eines Videostreams auf. Diese liegen für SD-Auflösung, welcher dem Anwendungsfall dieser Arbeit nahe kommt, bei 246 ms. Da dies die Zielsetzung der durchschnittlichen Laufzeit von 3 ms weit übersteigt und außerdem die verwendete CPU weitaus leistungsstärker ist als die des NAOs, wird von SIFT aus Laufzeitgründen abgesehen.

Der SURF-Algorithmus erfordert bei gleicher Auflösung eine Rechenzeit von circa 120 ms (Pentium IV, 3 GHz)[14]. Auch dies ist, selbst bei Verarbeitung von Teilen des Kamerabildes, ungeeignet für die Echtzeitverarbeitung des NAO-Rechners und wird folglich verworfen.

Machine Learning wird zurzeit immer wieder zu verschiedenen Zwecken der Bilderkennung erfolgreich getestet. Dabei darf nicht unbeachtet bleiben, dass eine erfolgreiche Klassifizierung auf wiederkehrenden Details innerhalb des Bildes beruht. Ein Fußballtor hat im Vergleich zu anderen Anwendungsfällen nur wenige markante Details, wie z.B. Höhe, Breite, das Tornetz oder die Feldlinien davor. Unter der Annahme, dass diese Merkmale von der Kamera überhaupt erfasst sind, entspricht dies einer übersichtlichen Anzahl an Merkmalen, die auch analytisch verarbeitet und bewertet werden kann. *Machine Learning* lässt sich dieses Basiswissen nicht übergeben, da Entscheidungsfindungen zu komplex sind, um sie nachzuvollziehen. Zu allerletzt stehen mir im Rahmen dieser Arbeit ungefähr 200 archivierte Bilder von Toren der Vorjahresturniere zur Verfügung. Lüders [18] verwendet circa 1750 Bilder für eine gute Objekterkennung auf dem NAO-System. Weitere Aufnahmen können nur im aktuellen Versuchslabor vorgenommen werden, was zu kontingenten Zusammenhängen führt. Dies bedeutet, dass sich das Modul ungewollt an Details aus dem Labor orientiert und nicht nur am Tor selbst. Eine Umsetzung mit Hilfe von *Machine Learning* ist bei größeren Beständen an Bilddaten möglich, jedoch nicht zwingend notwendig und wird deshalb im Rahmen dieser Arbeit nicht weiter verfolgt.

Sowohl die Veröffentlichungen über Kantendetektion als auch der Spaltenhistogramme zeigen auf, dass nach aktuellem Stand der Technik nur homogene, meist farblich vom Tor abgrenzende, Texturen eine zuverlässige Torerkennung ermöglichen. Keines der Paper beinhaltet Aussagen über Tordetektionen aus einem realen Spiel des RoboCups. Drei der vier vorgestellten Implementierungsarten bedienen sich der Feldgrenzen oder Feldlinien, eines benötigt zwingend die Querlatte im Bild, was nach Kapitel 1.3 in dieser Arbeit nicht vorausgesetzt werden soll. Zusammenfassend lässt sich sagen, dass eine Kopie dieser Algorithmen nicht ohne Weiteres zum Erfolg führt. Über die grundlegende Funktionalität der Algorithmen in realen Einsatzbedingungen kann dennoch keine Aussage getroffen werden. Eine Implementierung der beiden Verfahren wird im nachfolgenden Kapitel 3.4 beschrieben.

3.4 Kantendetektion und Histogrammfilter im Test

Im Rahmen dieses Kapitels werden beide Verfahren in Matlab zur Evaluation implementiert. Dabei wird zunächst auf bereits vorhandene Matlab-Bibliotheken zurückgegriffen und von einer Nutzung des Horizonts abgesehen. Der erfolgreichere Ansatz wird anschließend modifiziert und recheneffizienter auf dem NAO implementiert.

Implementierung der Kantendetektion Die aus Kapitel 3.2 erwähnte Filterung nach Sobel wird für vertikale Kanten durchgeführt, um lediglich Torpfosten und keine horizontalen Objekte zu detektieren. Das Festlegen eines Schwellwerts für die Absolutwerte nach der Faltung resultiert in einem Bild mit lediglich binären Pixeln. Auf diese Weise lassen sich Kanten des realen Bildes erkennen [15]. Ein Beispiel hierfür aus dem HULKs-Versuchslabor findet sich in Abbildung 3.4 (links).

Mittels einer Hough-Transformation werden Linien im Bereich von $\pm$ 20° um die vertikale Achse gesucht. Nach Aussortieren von Linien unter einer Mindestlänge verbleiben die in Abbildung 3.4 (rechts) grün dargestellten Kanten. Darunter befinden sich beide Kanten des linken, eine des rechten Pfosten, sowie zwei falsch detektierte Bereiche. Der zu diesem Abschnitt zugehörige Matlab-Code befindet sich in Anhang A.1.

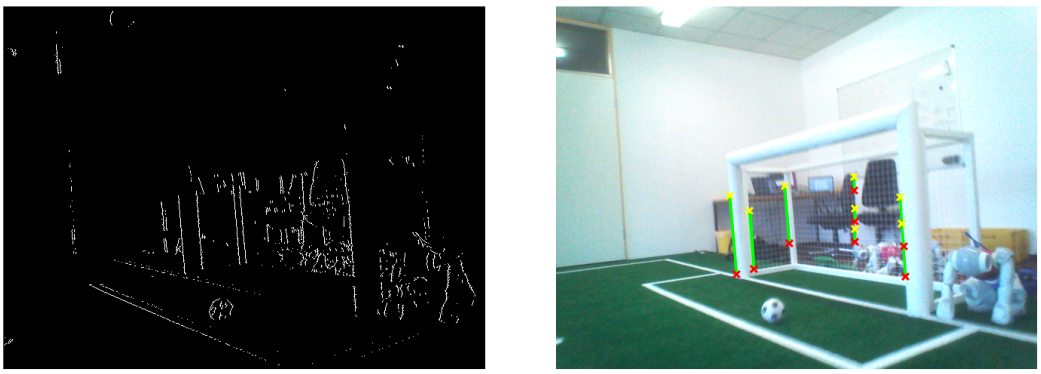

Abbildung 3.4: Sobel-Filterung (links) und gefundene Linien mit Hough (rechts)

Implementierung des Histogrammfilters Der Histogrammfilter sucht im Bild spaltenweise nach der Anzahl an Pixeln, die einen Helligkeitswert überschreiten. Um viele Bildpunkte der Tore, aber möglichst wenige des Hintergrunds zu erfassen, wird der Helligkeitswert, bei dem ein Pixel zum Spaltenhistogramm zählt, zunächst auf 160 festgelegt. Dies lässt sich aus Abbildung 3.5 ableiten, in der in zehn untersuchten Bildern aus dem Versuchslabor sowie verschiedener Turniere die Helligkeitswerte (Y) der Pfosten und Umgebung dargestellt sind.

Die weißen Torpfosten befinden sich idealerweise in denjenigen Spalten, in denen das Histogramm (siehe Abbildung 3.6) lokale Maxima besitzt, also die meisten hellen Pixel gefunden wurden. Da durch Rauschen, Abtastung und Einwirkungen des Hintergrunds

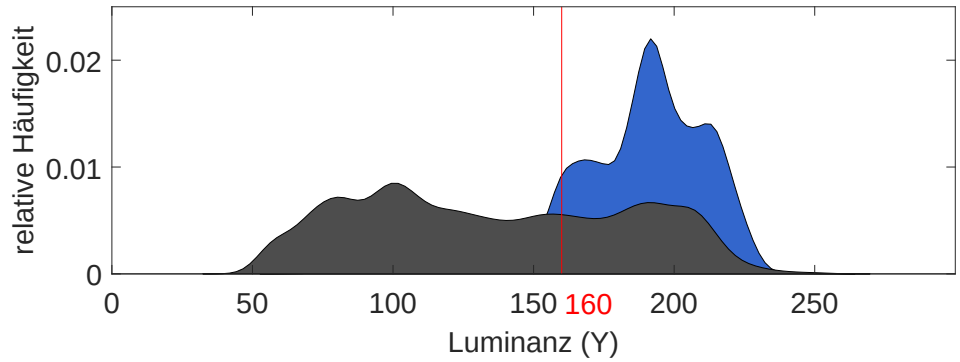

Abbildung 3.5: Historamm der Luminanz (0 bis 256) von Torpfosten (blau) und Hintergrundtexturen (grau)

sehr viele lokale Maxima entstehen, werden nur jene ausgewählt, die gleichzeitig auch den maximalen Wert in einer bestimmten Umgebung besitzen. Dazu wird bis zu einem Abstand von 20 Spalten geprüft, ob nicht noch höhere Histogrammwerte vorliegen.

Es resultieren die in Abbildung 3.6 dargestellten schwarzen und roten Stellen, wobei die Farbgebung im Folgenden erläutert wird.

Nach Betrachtung verschiedener Histogramme stellt sich heraus, dass an den Positionen der realen Pfosten sichtbar steilere Steigungen um das Extremum auftreten. Daher wird für jedes Extremum in beide Richtungen nach einer minimalen Steigung gesucht (hier 15). In Abbildung 3.6 sind die verbleibenden Maxima in rot gekennzeichnet. Der zugehörige Matlab-Code ist dem Anhang A.2 zu entnehmen.

Abbildung 3.6: Spaltenhistogramm eines Bildausschnitts, German Open 2017. Alle Punkte beschreiben lokale Maxima des Histogramms, die in einer festgelegten Umgebung globale Maxima beschreiben. Nur die rot dargestellten Punkte verbleiben nach einer Filterung, die die Steigung in der Umgebung einbezieht.

Resultate beider Verfahren Für die Evaluierung eines geeigneten Ansatzes wurde ein Datenset aus 41 Bildern zusammengestellt, die insgesamt 54 Pfosten beinhalten. Dabei wurden hier bereits anspruchsvolle Daten mit weißem Hintergrund und Unschärfe ausgewählt. Die Bilder enthalten einen, zwei oder gar keinen Pfosten und wurden aus möglichst verschiedenen Positionen aufgenommen. 24 Bilder stammen aus dem HULKs Labor und 17 aus den Turnieren des *German Open* 2017 und *Iran Open* 2017.

Die beiden Verfahren werden auf drei Kriterien überprüft (vgl. Tabelle 1). Angegeben ist die Anzahl der richtig detektierten Bereiche, die tatsächlich einen Torpfosten enthalten, als auch die falsch detektierten Bereiche, in denen keine Torpfosten enthalten sind. Für den Histogrammfilter bedeutet ein richtiger Bereich, dass die ausgewählte Spalte des Bildes durch das untere Ende des Torpfostens läuft. Bei der Kantendetektion ist ein Bereich als richtig klassifiziert, wenn mindestens ein Teil der Kante von einer Seite des Pfostens detektiert werden konnte. Außerdem wird die durchschnittliche Laufzeit der jeweiligen Matlab-Codes[4] ausgewertet.

Es zeigt sich, dass der Histogrammfilter mit 39 richtigen Bereichen (72%) deutlich mehr Torpfosten erkennt als die Kantendetektion (59%). Zudem ist der Informationsgehalt deutlich höher, da eine einzelne Kante noch keine Aussage darüber treffen kann, auf welcher Seite der Kante sich der Torpfosten befindet. Somit würden noch mehr als 75 falsch klassifizierte Bereiche überprüft werden müssen. Der Histogrammfilter weist mit 187 falsch detektierten Bereichen eine noch höhere Fehlerquote auf. Der Ansatz müsste hinsichtlich dieses Kriteriums für eine Nutzung weiter optimiert werden. Der Rechenaufwand liegt dafür bei nur einem Achtel gegenüber der Kantendetektion. Dabei gibt die absolute Rechenzeit einen qualitativen Vergleich beider Verfahren, jedoch keine absolute Einschätzung über die tatsächliche Rechenzeit auf dem NAO (andere Hardware und Programmiersprache).

Tabelle 1: Vergleich zwischen Histogrammfilter und Kantendetektion als Ansatz zur Generierung von Pfostenkandidaten

	Histogrammfilter	Kantendetektion
Richtige Bereiche	39	26
Falsche Bereiche	187	75
Rechenzeit (Matlab)	16,5 ms	134,1 ms

Im Folgenden wird daher der Ansatz des Histogrammfilters zur Generierung von Pfostenbereichen verwendet. Dabei ist nochmals zu betonen, dass die Torpfosten für dieses Verfahren den definierten Helligkeitswert überschreiten müssen. Bei starkem Gegenlicht kann der automatische Weißabgleich des NAOs zur Verdunklung und somit einer Nichterkennung der Torpfosten führen.

[4]Hierzu wurde die Matlab-interne *TicToc*-Funktion mit einem Intel Core i7-6600U, 2,6 GHz und 8 GB RAM verwendet.

3.5 Finale Implementierung auf dem NAO

Um die Resultate des provisorischen Matlab-Codes zu optimieren, wird der Algorithmus weiter modifiziert und anschließend auf dem NAO implementiert.

- Für das Kapitel der Generierung und Lokalisierung wird die Qualität der Bilddaten reduziert, um eine möglichst geringe Rechenzeit zu gewährleisten. Abbildung 3.7 zeigt beispielhaft Aufnahmen in Kameraauflösung (links) und mit einem Sampling-Intervall von 5 (mitte) und 10 (rechts) Pixeln in horizontaler und vertikaler Richtung.

Abbildung 3.7: Bildqualität und Auflösung bei Beachtung (a) jeder (b) jeder fünften (c) jeder zehnten Spalte und Zeile

  Wie in Tabelle 2 zu sehen, stellt sich heraus, dass eine Auswertung jeder fünften oder zehnten Spalte und Zeile die Rechenzeit um mehr als 3 ms reduziert. Dabei werden zwar 3, bzw. 7 Torpfosten weniger erkannt, aber die Fehldetektionen sinken von 187 auf 79, bzw. 45. Eine Reduzierung der Datenmenge wird somit als hilfreich angesehen. Um die Erkennung der Torpfosten jedoch auch von der Mittellinie oder weiter zu erhalten, sowie eine qualitativ gute Ermittlung der Pfostenbreite möglich zu machen, sollte der Abtastabstand nicht auf zehn festgelegt werden. In diesem Fall kommt es vor, dass die Torpfosten horizontal nur noch durch ein bis zwei Pixel repräsentiert werden. Für die Generierung von Bereichen mit Torpfostenkandidaten wird daher eine Abtastung jeder fünften Zeile und Spalte festgelegt.

Tabelle 2: Richtig und falsch detektierte Bereiche bei 54 Torpfosten und einer Bildabtastung von jedem, jedem fünften und jedem zehnten Pixel horizontal und vertikal

Abtastabstand [px]	1	5	10
Richtig Bereiche	39	36	32
Falsch Bereiche	187	79	45
Rechenzeit (Matlab)	16,5 ms	13,3 ms	13,1 ms

- In Abschnitt 2.2 wurde bereits erwähnt, dass selbst ein hoher Wert des Y-Kanals nicht zwingend auf einen Weißton schließt. Insbesondere bei Luminanzen unter dem Wert 200 sind lediglich die Grautöne potentielle Farbtöne eines Torpfostens. Für

die Generierung des Histogramms werden daher auch die Chrominanzen Cb und Cr beachtet. Die in Abbildung 3.8 dargestellten Rechtecke und deren innere Fläche zeigen die Farbtöne, die in das Histogramm einfließen. Unter einem Y-Wert von 200 ist der rechteckige Bereich klein gewählt, da nur die grauen Pixel in der Mitte des Farbmodells potentiell dem Farbton des Pfostens entsprechen. Über einem Wert von 200 sind die Weißtöne großflächiger verteilt. Die mathematische exakte Beschreibung ist dem Anhang A.4 zu entnehmen.

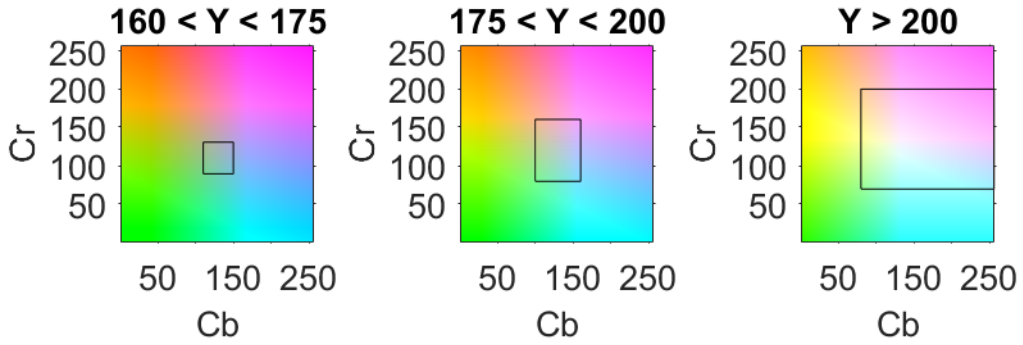

Abbildung 3.8: Numerische Eingrenzung der Cb/Cr-Werte für einen mögliche Torpixel Die drei zur Visualisierung hinterlegten Farbebenen entsprechen den Y-Werten 167, 187 und 200.

- Um Störungen der Umgebung für das Spaltenhistogramm noch weiter zu reduzieren, werden nur die Bilddaten unterhalb einer spezifizierten Höhe untersucht. Diese liegt 50 Pixel über dem in Abschnitt 3.1 erläuterten Horizont, sodass immer ein Großteil des Pfostens beachtet, der Bereich über dem Tor aber außer Acht gelassen wird. Somit ergibt sich ein Histogramm, aus dem die realen Pfosten ersichtlicher und damit leichter analytisch zu detektieren sind. Sollte der Horizont unter dem Bildrand liegen, ist das untere Ende eines Torpfostens nicht zu sehen und der Algorithmus dieser Arbeit wird aus Gründen der Recheneffizienz beendet.

- Im Folgenden ist außerdem eine grobe Abschätzung der Pfostenbreite wünschenswert. Dabei wurde bereits in Kapitel 3.2 (Abschnitt Spaltenhistogramm) zu beiden Seiten des Maximums nach einer Steigung gesucht. Jetzt wird außerhalb dieser ebenso jede weitere Spalte miteinbezogen, sofern der Schwellwert der Steigung nicht unterschritten wird. Abbildung 3.9 visualisiert ein exemplarisches Histogramm und zwei daraus abgeleitete blau hinterlegte Bereiche mit potentiellen Torpfosten. Sollte auf einer oder beiden Seiten innerhalb einer gewissen Distanz keine entsprechende Steigung erkannt werden, so gilt das Maximum nicht als Pfostenkandidat.

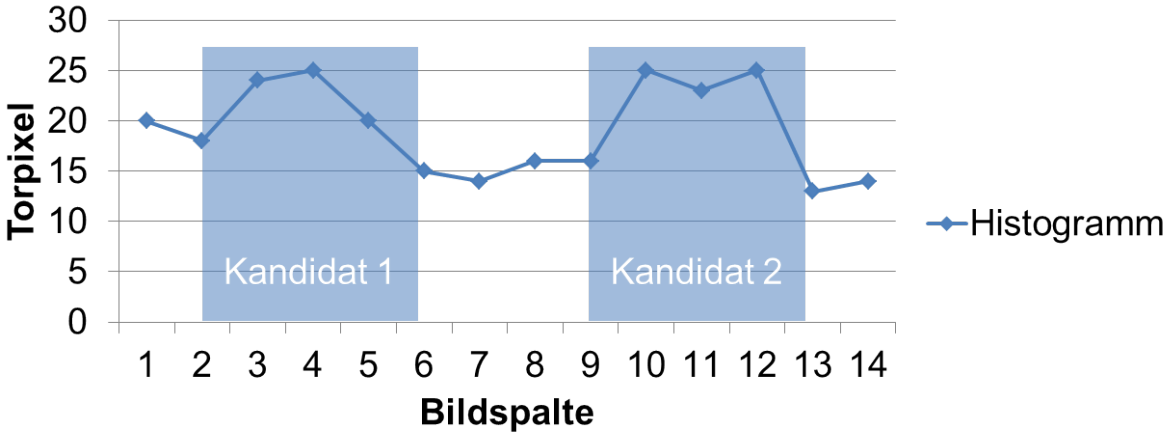

Abbildung 3.9: Abschätzung der Pfostenbreite im Spaltenhistogramm
Die minimal notwendige Steigung beträgt hier vier, sodass z.B. die Segmente 3-4, 10-11 oder 11-12 keine Pfostenenden darstellen können. Das Segment 4-5 hingegen entspricht einem möglichen Ende, das jedoch um ein weiteres Segment erweitert wird, da die Steigung 5-6 den Wert vier nicht unterschreitet.

Schlussendlich werden aus den in Tabelle 1 bereits evaluierten Daten 37 richtige und nur noch 66 falsche Bereiche detektiert. Zudem ist eine ungefähre Abschätzung der Pfostenbreite möglich. Die Rechenzeit der finalen Ausführung beträgt auf dem NAO über 3800 Iterationen hinweg durchschnittlich 0,51 ms. Das Maximum einer Iteration liegt bei 0,69 ms. Diese kann auf Grund verschiedener Hardware und Programmiersprache nicht mit der Rechenzeit der Matlab-Codes verglichen werden.

Tabelle 3: Auswertung der finalen Implementierung des Histogrammfilters auf dem NAO

	Erweiterter Histogrammfilter
Richtige Bereiche	37
Falsche Bereiche	66
Rechenzeit (NAO)	0,51 ms

4 Lokalisierung der Pfostenkandidaten

In diesem Kapitel soll eine Methode gefunden werden, mit dessen Hilfe mindestens ein Merkmal des Torpfostens bestimmt wird, dessen Höhe in Weltkoordinaten bekannt ist. Nur somit kann von erkannten Bildpunkten auf die Roboterposition in Weltkoordinaten geschlossen werden. Ebenso ist es wünschenswert, dass von der Generierung falsch detektierte Bereiche weiter isoliert werden.

4.1 Evaluierung einer Methode

Bolotnikova [3] generiert zusätzlich zu dem im letzten Kapitel erstellten Spaltenhistogramm ein Zeilenhistogramm. In diesem kann die Zeile der Querlatte detektiert werden, welche die oberen Endpunkte der Pfostenkandidaten beschreibt. Da die Höhe der Torpfosten aus den Regularien bekannt ist, kann hiermit auf eine Position in Weltkoordinaten zurückgeschlossen werden. Dieses Verfahren setzt jedoch voraus, dass die Querlatte von der Kamera erfasst wird. Mit Hilfe dieser Methode würden außerdem keine der falsch detektierten Bereiche ausgeschlossen werden, die in Tabelle 3 festgestellt wurden.

Cano [8] nutzt vom Spaltenhistogramm ermittelten Spalten und die Höhe der Feldgrenze, indem von dessen Schnittpunkt aus in beiden Richtungen vertikal nach einem Sprung von hoher (Torpfosten) zu niedriger Luminanz (Feldfarbe, Hintergrund) gesucht wird. Die Feldgrenze, welche in Kapitel 3.1 erläutert wurde, soll jedoch auf Grund von Fehlerfortpflanzungen nicht genutzt werden.

Dennoch ist dieses Vorgehen auch ohne eine Referenzhöhe möglich, indem die gesamte Spalte vertikal auf Helligkeitssprünge untersucht wird. Diese Methode wurde noch für die farbbasierte Torlokalisierung beim RoboCup veröffentlicht[7]. Da weiterhin das im Abstand von fünf Pixeln abgetastete Bild verwendet wird, sind maximal 96 Bildpunkte pro Kandidat zu überprüfen. Die Rechenzeit ist somit gegenüber dem vorherigen Kapitel zu vernachlässigen, welche $96 \cdot 128 = 12288$ Bildpunkte zur Histogrammgenerierung verarbeitet und anschließend Maxima in diesem ermittelt. Eine Isolierung falscher Bereiche ist gegeben, indem ein unteres Pfostenende zwingend gefunden werden muss und ebenso die Distanz zum eventuell vorhandenen oberen Ende betrachtet werden kann. Da das obere Pfostenende nicht zwingend zu sehen ist, liegt der Fokus mehr auf der korrekten Lokalisierung des unteren Endes.

Im folgenden Abschnitt wird die Implementierung auf dem NAO System für eine vertikale Suche nach Start- und Endpunkt des Torpfostens erläutert.

4.2 Implementierung

Die Implementierung der ausgewählten Methode wird an Hand der Abbildung 4.1 erläutert, welche ein Ausschnitt der in der Generierung behandelten Abbildung 3.6 beschreibt. Die bereits durch den Histogrammfilter geschätzte Breite der Kandidaten ist durch die blauen Linien am oberen Rand gekennzeichnet. Die Spalte mittig dieses Bereichs wird im Folgenden auf mögliche Anfangs- und Endpunkte der Kandidaten geprüft.

Abbildung 4.1: Lokalisierung der Pfostenkandidaten innerhalb einer Spalte.
Die Nummerierung kennzeichnet Sonderfälle, die in diesem Kapitel erläutert sind. Die weiße Markierung links kennzeichnet den Horizont, die blauen Markierung am oberen Bildrand geben die zu untersuchenden Bereiche an.

Zur Erkennung von Objektübergängen wird **von unten nach oben** die Helligkeitsdifferenz zweier Pixel $Y_i - Y_{i-1}$ betrachtet. Ein Übergang von dunkleren zu helleren Konturen spiegelt sich in einem positiven Sprung der Luminanz wieder (Abbildung 4.1 in grün), andersherum negativ (in blau)[5]. Der Gleichung 4.1 zufolge werden die Übergänge im Folgenden als steigende Flanke (RE, von *Rising Edge*) und fallende Flanke (FE, von *Falling Edge*) bezeichnet. Diese beiden Schwellwerte unterscheiden sich, da der Übergang von

[5]Das Bild wurde hier in vollauflösender Qualität abgebildet. Der Algorithmus verarbeitet nur jede fünfte Zeile und Spalte, weswegen die grün und blau dargestellten Flanken unpräzise lokalisiert sein können.

Feld zu Pfosten leichter zu ermitteln ist als von Pfosten zu variablem Hintergrund.

$$\text{Pixelübergang} = \begin{cases} \text{RE} & Y_i - Y_{i-1} \geq 25 \\ \text{FE} & Y_i - Y_{i-1} \leq -15 \\ \text{false} & \text{sonst} \end{cases} \tag{4.1}$$

Dabei sollen kurze Kandidaten (< 80 Pixel in Originalauflösung) nicht beachtet werden (entspricht den mit einer 1 gekennzeichneten Stellen). Diese Mindestlänge erfüllen die realen Pfosten in allen innerhalb des Spielfelds aufgenommenen Bildern. Alle kürzeren Kandidaten können somit aussortiert werden. Somit besteht für den linken Kandidaten in Abbildung 4.1 nur noch eine mögliche Kombination aus Start- und Endpunkt. Sollte die Distanz zwischen RE und FE sogar weniger als 20 Pixel betragen, wird die grüne RE entfernt (Kennzeichnung durch eine 2), sodass diese auch nicht anderweitig als Startpunkt dienen kann. Dies hängt damit zusammen, dass diese kurzen Intervalle durch Linien, andere NAOs oder den Ball hervorgerufen werden. Der Pseudocode im Anhang A.3 stellt diese Vorgehensweise zur Bestimmung von Start- und Endpunkt der möglichen Pfosten ergänzend dar.

Zur Hilfe genommen wird hier zudem der in Kapitel 3.1 erläuterte Horizont. Die Höhe des Horizonts wird durch die weiß gestrichelte Linie am linken Bildrand dargestellt. Im Folgenden werden REs, also das potentielle untere Pfostenende, nur unterhalb des Horizonts und FEs, oberes Pfostenende, nur oberhalb des Horizonts akzeptiert[6]. Hiermit können pfostenähnliche Objekte an oder unter der Decke, auf dem Spielfeld oder Feldlinien nicht fälschlicherweise als Pfosten in Betracht gezogen werden, da diese ganz unter oder über dem Horizont liegen.

Um einerseits Roboter oder andere Hindernisse zu ignorieren und andererseits den Rechenaufwand gering zu halten, wird eine Spalte nach Finden von acht Flanken als Pfostenkandidat verworfen (Kennzeichnung 3). Zudem werden oberhalb des Horizonts lediglich zwei FE gesucht. Dies reduziert die Anzahl der zu bewertenden Kandidaten und begrenzt somit die Rechenzeit. Es sollte idealerweise die erste FE bereits das Ende des Pfostens sein. Bei verschmutzten Torpfosten oder Schattenübergängen kann dies jedoch abweichen. Sollten weniger als zwei FE über dem Horizont gefunden werden, wird eine künstliche FE 150 Pixel über dem Horizont, aber maximal am oberen Bildrand gesetzt (Kennzeichnung 4). Diese Art von FE wird in dieser Arbeit als *gesetzt* gekennzeichnet, während alle anderen FE als *erkannt* oder *detektiert* bezeichnet werden. Eine korrekte Angabe des oberen Endes ist somit nicht gewährleistet, jedoch können hierdurch Pfosten evaluiert werden, dessen oberes Ende nicht im Bild liegt.

Alle übrig gebliebenen Möglichkeiten für unteren und oberen Punkt des Pfostens sind mit grünen Linien gekennzeichnet und werden im nachfolgenden Kapitel bewertet[7]. Die Ergebnisse (ausgegebene richtig lokalisierte sowie falsch lokalisierte Pfostenpunkte) dieses Kapitels werden in Kapitel 6 evaluiert.

[6] Abbildung 4.1 zeigt dennoch auch die FE unterhalb des Horizonts, um das soeben beschriebene Löschen einer RE (Kennzeichnung 2) zu visualisieren

[7] Da für die beiden rechten Kandidaten zwei mögliche blaue FE in Frage kommen, werden insgesamt fünf Kandidaten ermittelt

5 Bewertung der Pfostenkandidaten

5.1 Bewertungskriterien

Die nun lokalisierten Kandidaten enthalten weiterhin falsche Objekte. Mit Hilfe dieses dritten und letzten Abschnitts des Verfahrens dieser Arbeit sollen alle falschen Kandidaten aussortiert und gleichzeitig möglichst alle korrekten Kandidaten beibehalten werden. Ein bis hierhin mit dieser Arbeit vergleichbarer Ansatz wurde nicht veröffentlicht, weswegen eine Evaluation verschiedener Ansätze entfällt. Dieses Kapitel ist das Resultat aus langfristigen Beobachtungen über richtig und falsch bewertete Kandidaten und eine dahingehend selektierte Auswahl an Prüfkriterien sowie Optimierung der jeweiligen Gewichtung.

Insgesamt werden alle Kandidaten an Hand von sieben Kriterien bewertet, gewichtet und eine Gesamtbewertung von bis zu 100 ermittelt. Das achte, sogenannte Stützpfosten-Kriterium, ermöglicht einen zusätzlichen Bonus von 15. Anschließend kann eine finale Klassifizierung als (kein) Torpfosten getroffen werden.

Dabei wird weiterhin Bezug auf die fünf in Kapitel 4 ermittelten Kandidaten genommen, die in Abbildung 5.1 nummeriert sind. Da nur noch auf spezielle Bereiche des Bildes zugegriffen werden muss, wird von hier an das vollauflösende Kamerabild mit 640x480 Pixeln verwendet.

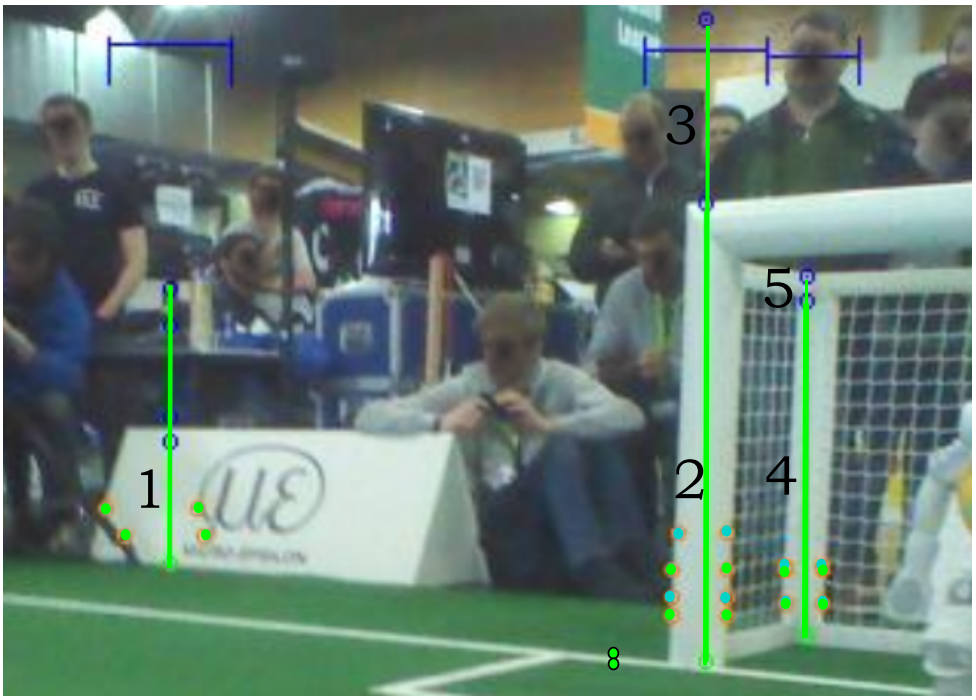

Abbildung 5.1: Fünf von der Lokalisierung ausgegebene Torpfostenkandidaten
Kandidat 1 erstreckt sich vom grünen Start- zum obersten blauen Endpunkt, da vorherige Endpunkte in einer zu kurzen Pfostenlänge resultieren. Für die mittlere und rechte Spalte kommen zwei verschiedene Endpunkte in Frage.

Kriterium *Torlinie* Das erste Kriterien ist die beidseitige Suche nach einer hellen Linie, die der Torlinie entsprechen soll. Dazu wird am untersten Punkt des Kandidaten zehn Pixel außerhalb der vom Histogramm geschätzten Breite vertikal nach einer RE und darauffolgenden FE gesucht. Für jeweils eine erkannte Seite wird die Bewertung dieses Kriteriums um $0,5$ inkrementiert (vgl. Gleichung 5.1). Außerdem wird in Betracht gezogen, dass der Pfosten am Bildrand und eine Seite nicht prüfbar (n.p.) ist.

$$\text{Score}_\text{A} = \begin{cases} 1 & \text{beide Seiten erkannt} \\ 1 & \text{eine Seite erkannt, eine n.p.} \\ 0,5 & \text{eine Seite erkannt} \\ 0 & \text{sonst} \end{cases} \tag{5.1}$$

Die schwarz umkreisten Punkte in Abbildung 5.1 kennzeichnen gefundene Torlinien. Nur für die Kandidaten 2 und 3 kann auf der linken Seite eine Linie erkannt werden, womit sich Tabelle 4 ableiten lässt.

Tabelle 4: Torlinien-Kriterium für Torpfostenkandidaten aus Abbildung 5.1

Kandidat	1	2	3	4	5
Score_A	0	0,5	0,5	0	0

Kriterium *Luminanz im Durchschnitt* Um die Bewertung des Pfosten zu ergänzen, wird der durchschnittliche Helligkeitswert W im Bereich des Kandidaten ermittelt. Dabei werden die Luminanzen aller Pixel vertikal von unterem zu oberem Ende, sowie horizontal bis zur geschätzten Breite betrachtet. Gleichung 5.2 zeigt das Resultat in Abhängigkeit des Helligkeitswerts pro Pixel innerhalb des durchsuchten Bereichs. Dabei soll die Bewertung bei Helligkeitswerten von 190 bis 155 linear auf Null fallen und darunter gegebenenfalls negativ werden. Diese Bewertung fällt positiver aus als nachfolgendes Kriterium, da somit Torpfosten erhalten bleiben, dessen oberes Ende nicht erkannt und daher ein Teil des Hintergrund mit einbezogen wurde.

$$\text{Score}_\text{B}(\text{W}) = \begin{cases} 1 & W \geq 190 \\ \frac{W-190+35}{35} & \text{sonst} \end{cases} \tag{5.2}$$

Für die Kandidaten dieses Kapitel ergeben sich die für Tabelle 5 gerundeten Werte. Dabei erhält Kandidat 2 richtigerweise eine höhere Bewertung als Kandidat 3, da Letzterer auch Teile des Publikums beinhaltet.

Tabelle 5: Luminanz-Kriterium für Torpfostenkandidaten aus Abbildung 5.1

Kandidat	1	2	3	4	5
W	144	165	153	169	169
Score_B	0,17	0,55	0,34	0,61	0,62

Kriterium *Luminanz im Durschnitt (unten)* Innerhalb des vorherigen Kriteriums wird die durchschnittliche Luminanz bei 15% der erreichten Pfostenhöhe gespeichert. Diese dient der Bewertung auf gleicher Weise wie zuvor, vergibt jedoch bereits bei $W = 170$ eine Bewertung von Null, da im unteren Pfostenbereich keine dunkleren Hintergrundtexturen auftreten dürfen. Es dient vor allem der Unterscheidung der Kandidaten innerhalb einer einzelnen Spalte sowie der Vermeidung von Torraumlinien oder anderen Objekten als Startpunkt unterhalb des Pfostens. Da dieser Fall in der beispielhaften Abbildung 5.1 nicht vorliegt, erhalten die Kandidaten ähnliche Bewertungen in Tabelle 6.

$$\text{Score}_\text{C}(\text{W}) = \begin{cases} 1 & W \geq 190 \\ \frac{W-190+20}{20} & \text{sonst} \end{cases} \tag{5.3}$$

Tabelle 6: Luminanz-Kriterium (unten) für Torpfostenkandidaten aus Abbildung 5.1

Kandidat	1	2	3	4	5
W	209	177	171	175	175
Score_C	1	0,67	0,52	0,63	0,62

Kriterium *Histogrammbreite* Das bereits generierte Spaltenhistogramm des Bildes (Abbildung 3.6) ist mit dessen geschätzter Breite B der Kandidaten auch ein Bestandteil der Gesamtbewertung. Auch wenn die Pfostenbreite je nach Roboterposition unterschiedlich groß erscheint, ist ein großes Objekt tendenziell zu bevorzugen. Somit werden Ausreißer im Histogramm und insbesondere die dünneren Stützpfosten des Tornetzes schlechter bewertet. Gleichung 5.4 beschreibt diese Wertung für Tabelle 7.

$$\text{Score}_\text{D}(\text{B}) = \begin{cases} 1 & \text{B} > 30 \\ 0 & \text{sonst} \end{cases} \tag{5.4}$$

Tabelle 7: Bewertung der Histogrammbreite für Torpfostenkandidaten aus Abbildung 5.1

Kandidat	1	2	3	4	5
Breite	40	40	40	30	30
Score_D	0,5	0,5	0,5	0	0

Kriterium *Flanken* Betrachtet man den möglichen Pfosten von unten nach oben, ist die Luminanz im Normalfall konstant. Sollten zwischen Pfostenstart und -endpunkt Flanken (REs oder FEs) erkannt werden, verschlechtert sich die Wertung. Es schwächt Kandidaten, die fälschlicherweise durch das Histogramm erkannt wurden, wie z.B. andere NAOs, da diese viele Flanken innerhalb des Objektes enthalten. Folgende Gleichung kann negativ in die Gesamtbewertung eingehen:

$$\text{Score}_{\text{E}}(\text{RE}, \text{FE}) = 1 - 0,5 \cdot \sum (\text{RE+FE}) \tag{5.5}$$

Dabei werden auch die REs und FEs betrachtet, die in Kapitel 4 gelöscht wurden, da sie dennoch die Textur des Kandidaten beschreiben. Für die exemplarischen Kandidaten aus Abbildung 5.1 ergibt sich die Bewertung nach Tabelle 8.

Tabelle 8: Flankenkriterium für Torpfostenkandidaten aus Abbildung 5.1

Kandidat	1	2	3	4	5
Score_{E}	-0,5	1	0,5	1	0,5

Kriterium *Oberes Torpfostenende* Mit Hilfe dieses Kriteriums werden selbst gesetzte (vgl. Kapitel 4) Pfostenenden innerhalb einer Spalte benachteiligt, da sie nur für den Fall richtig sind, dass das korrekte Pfostenende nicht erkannt werden konnte.

Ist das Pfostenende (PE) detektiert worden, wird eine Eins ausgegeben (vgl. Gleichung 5.6). Die Bewertung liegt bei Null, wenn das Ende sich am oberen Bildrand befindet. Somit können auch aus dem Bild herausgehende Pfosten noch erkannt werden. Im Falle eines manuell gesetzten Endpunkts innerhalb des Bildes wird eine -1 ausgegeben, da es nur unwahrscheinlich nahe des korrekten oberen Endes liegt.

$$\text{Score}_{\text{F}}(\text{PE}) = \begin{cases} -1 & \text{PE unterhalb von Bildrand } \textit{gesetzt} \\ 0 & \text{PE an Bildrand } \textit{gesetzt} \\ 1 & \text{sonst} \end{cases} \tag{5.6}$$

Es resultiert eine Bewertung der exemplarischen Kandidaten nach Tabelle 9.

Tabelle 9: Bewertung des oberen Endes für Torpfostenkandidaten aus Abbildung 5.1

Kandidat	1	2	3	4	5
Score_{F}	1	1	0	1	1

Kriterium *Höhe zu Breite Verhältnis* Das auf die Gesamtbewertung einflussreichste Kriterium wird durch das Verhältnis von Höhe zu Breite des Pfostens (α) ermittelt. Die Länge ist durch den unteren und oberen Pfostenpunkt bereits gegeben. Kapitel 3.4 liefert bereits eine grobe Schätzung der Breite. Da diese auf Grund der Abtastung jeder fünften Spalte jedoch dementsprechend ungenau ist, wird an vorher festgelegten Stellen nochmals pixelgenau nach dem seitlichen Ende des Pfostens gesucht. Dies wird hier horizontal in beide Richtung bei Erreichen einer Höhe von 10% sowie 20% der Pfostenlänge überprüft und ist durch die grünen und blauen Punkte bei den Kandidaten in Abbildung 5.1 gekennzeichnet. Weiter oben ist der Übergang zwischen Pfosten und Hintergrund nicht zwingend erkennbar und liefert vor allem bei hellen Hintergrundtexturen unzuverlässige Ergebnisse.

Aus den jeweils zwei ermittelten Breiten resultieren die Verhältnisse α_1 und α_2. Das wahre α (Höhe dividiert durch Breite des Torpfostens) beträgt nach Kapitel 2.3 $\frac{800mm+100mm}{100mm} = 9$. Aus Gleichung 5.7 lässt sich aus den beiden ermittelten Breiten jeweils eine Bewerung Score_{G1} und Score_{G2} ermitteln. Wie diese in Tabelle 10 zu Score_{G} verrechnet werden ist dem Anhang A.5 zu entnehmen.

$$\text{Score}_{\text{G}} = \begin{cases} -1 & \text{Flanke bedseitig nicht erkannt} \\ 0,15 & \text{Flanke nur einseitig erkannt} \\ 1 & 7,5 \leq \alpha \leq 10,5 \\ 1 - 0,3 \cdot (\alpha - 9) & \alpha > 10,5 \\ 1 - 0,25 \cdot (9 - \alpha) & \alpha < 7,5 \text{ und PE } \textit{erkannt} \\ 1 - 0,1 \cdot (9 - \alpha) & \alpha < 7,5 \text{ und PE } \textit{gesetzt} \end{cases} \tag{5.7}$$

Tabelle 10: Höhe zu Breite-Kriterium für Torpfostenkandidaten aus Abbildung 5.1

Kandidat	1	2	3	4	5
α_1	3,46	8,3	11,67	9,17	9,83
α_2	3	8,3	13,13	9,17	9,83
Score_{G}	-0,40	1	-0,06	1	1

Dieses Kriterium stellt sich zur Filterung als sehr effektiv heraus, da die sonst auftretenden Fehldetektionen nur selten das gleiche Verhältnis α wie ein Torpfosten aufweisen. Zudem erweist es sich im Labor und bei den Iran Open als eine effektive Möglichkeit die Stützpfosten des Tornetzes von den realen Pfosten zu unterscheiden. Diese sind schmaler und haben deswegen ein größeres α. Ist das Verhältnis zu hoch, hat dies in Gleichung 5.7 daher schlechtere Bewertungen zur Folge als bei zu niedrigem α. Bei den German Open, zu dem die hier erläuterte Abbildung 5.1 gehört, sind die Stützpfosten breiter und werden somit nicht wie gewünscht schlechter bewertet. Sollte der erkannte Pfosten breiter erscheinen (niedrigeres α), wird geprüft, ob die Oberkante des Pfostens *erkannt* oder aber *gesetzt* wurde (vgl. Kapitel 4). Befindet sich die Querlatte nicht mehr auf dem Bild, so soll der Pfosten dennoch erkannt werden und bekommt daher bei gleicher Abweichung einen höheren Score.

Stützpfosten-Bonus Da es wie in dem hier demonstrierten Beispiel trotz der hohen Gewichtung von Score_G vorkommen kann, dass die weißen Tornetzstützen am Gerüst eine hohe Bewertung erreichen, wird ein Bonus $\text{Score}_\text{Stuetz}$ eingeführt. Liegen zwei Torpfosten in einem Abstand von 100 Pixeln und liegen vertikale Anfangs- und Endkoordinate eines dieser Pfosten zwischen denen des anderen Pfostens, handelt es sich bei dem kleineren Kandidaten um einen Stützpfosten. Dies kommt in Abbildung 5.1 zur Anwendung. Dabei ist dieser Bonus, bedingt durch Ablauf des Programmcodes, ungewichtet und geht im Folgenden Kapitel als absoluter Bonus in die Gesamtbewertung ein. Im Falle eines erkannten Stützpfostens erhöht sich die Bewertung des korrekten Torpfosten um 15 Bonuspunkte und dekrementiert die des Stützpfostens um 30.

5.2 Gesamtscore

Um die endgültige Gesamtbewertung des Pfostenkandidaten zu berechnen, ist die Gewichtung der sieben Kriterien in Abbildung 5.2 dargestellt.

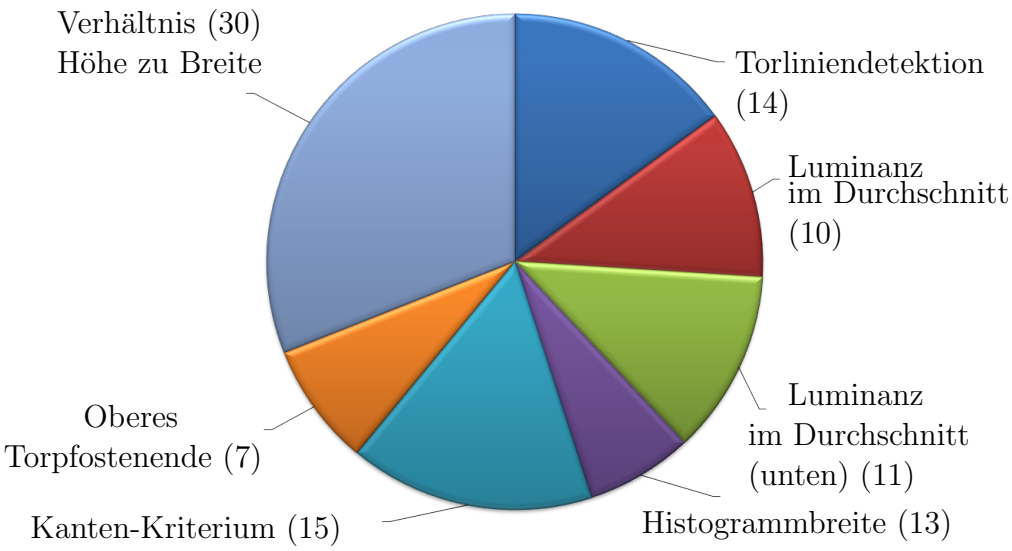

Abbildung 5.2: Gewichtung der in diesem Kapitel beschriebenen Kriterien

Tabelle 11 fasst alle mit der jeweiligen Gewichtung multiplizierten Bewertungen zusammen. Innerhalb einer Spalte wird nur der beste Kandidat weiter betrachtet, weshalb Kandidat 3 und 5 nach Vergleich der Zwischensumme wegfallen. Die Gesamtbewertung errechnet sich dann nach Gleichung 5.8. Sein maximaler Wert beträgt 100 (+15 bei Stützpfosten).

$$\text{Gesamtbewertung} = \text{Score}_{\text{Stuetz}} + \sum_{i=A}^{G} \text{Score}_{\text{i}} \cdot \text{Gewichtung}_{\text{i}} \tag{5.8}$$

Tabelle 11: Analyse der gewichteten Bewertungen und Gesamtbewertung der Torpfostenkandidaten aus Abbildung 5.1

Kandidat	1	2	3	4	5
$\text{Score}_{\text{A}} \cdot \text{Gewichtung}_{\text{A}}$	0	7	7	0	0
$\text{Score}_{\text{B}} \cdot \text{Gewichtung}_{\text{B}}$	1,7	5,5	3,4	6,1	6,2
$\text{Score}_{\text{C}} \cdot \text{Gewichtung}_{\text{C}}$	11	8,36	5,72	6,93	6,82
$\text{Score}_{\text{D}} \cdot \text{Gewichtung}_{\text{D}}$	6,5	6,5	6,5	0	0
$\text{Score}_{\text{E}} \cdot \text{Gewichtung}_{\text{E}}$	-7,5	15	7,5	15	7,5
$\text{Score}_{\text{F}} \cdot \text{Gewichtung}_{\text{F}}$	7	7	0	7	7
$\text{Score}_{\text{G}} \cdot \text{Gewichtung}_{\text{G}}$	-12	30	-1,8	30	30
Zwischensumme	6,7	79,36	28,32	65,03	57,52
$\text{Score}_{\text{Stuetz}}$	0	15	-	-30	-
Gesamtbewertung	6,7	94,36	-	35,03	-

Folglich ist der korrekte Torpfosten 2 mit einer Gesamtbewertung von 94 hoch bewertet, während die falschen Kandidaten deutlich schlechtere Ergebnisse erzielen.

5.3 Klassifizierung der Torpfostenkandidaten

In diesem Abschnitt werden die Gesamtbewertungen falscher und richtiger Kandidaten analysiert, um einen geeigneten Schwellwert für die Klassifizierung als Torpfosten zu erhalten.

Für diese Ermittlung werden Datensätze mit insgesamt 165 zu erkennenden Torpfosten genutzt. Darunter fallen Torpfosten, deren unteres Pfostenende sichtbar ist und dessen Farbwert in den unter Kapitel 3.5 definierten Bereichen fällt.

16 Torpfosten sind statisch aus zufällig ausgewählten Positionen der tornahen Spielfeldhälfte mit Blickrichtung auf das Tor aufgenommen worden. 30 Torpfosten wurde sind entstanden, während der NAO von der Mittelfeldlinie aus auf das Tor zulief. Bilddaten mit 119 Torpfosten wurden aus Testspielen vom 31.06.2017 und 06.07.2017 im HULKs-Labor entnommen. Somit sind viele Ansichten des Tores, unscharfe Bilder, aber auch Einflüsse anderer NAOs oder menschlicher Personen im Testset enthalten. Ein exemplarischer Bildausschnitt aus dem Labor, bei dem der linke Pfosten erkannt wird, zeigt Abbildung 5.3.

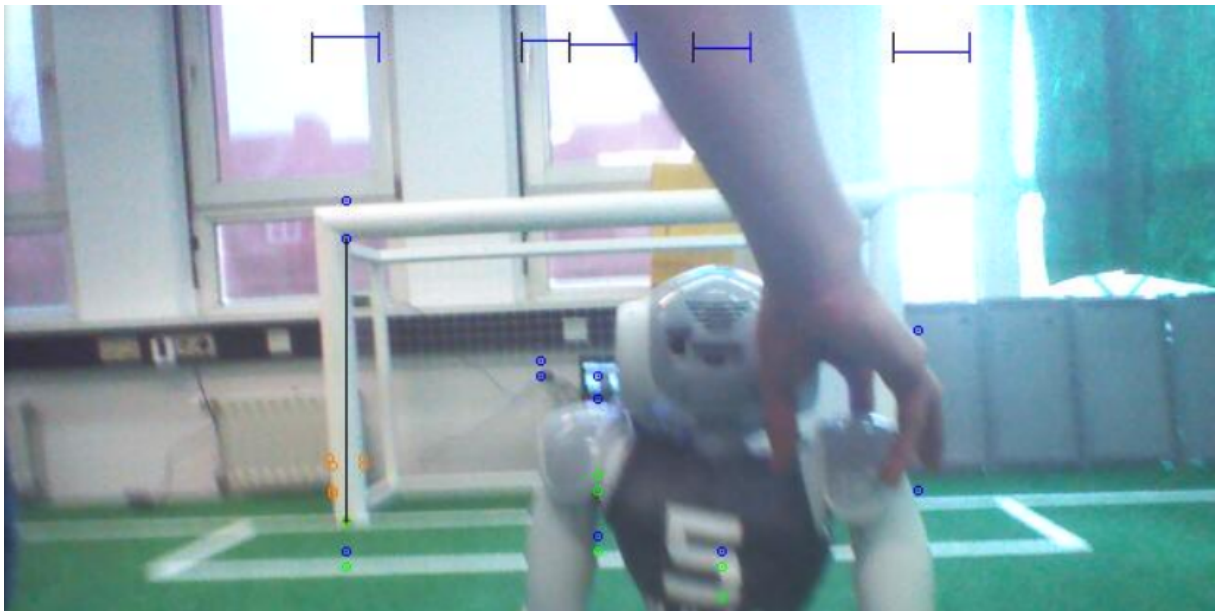

Abbildung 5.3: Detektion des linken Torpfostens im HULKs-Labor aus dem in diesem Kapitel verwendeten Testset

Die Abbildung 5.4 zeigt die nach dieser Arbeit bewerteten Kandidaten. Die grünen Balken repräsentieren korrekte Torpfosten, während die roten Balken Falschdetektionen sind. Je nach Zielsetzung des Anwenders kann nun ein Schwellwert festgelegt werden, über dem ein Pfostenkandidat als Torpfosten klassifiziert und ausgegeben wird. Besteht der Wunsch, keine einzige Falschaussage zu treffen, liegt der Schwellwert hier bei circa 66 (am weitesten rechts liegender roter Balken). Sind falsch detektierte Pfosten für den Nutzer weniger relevant, ist ein Schwellwert von 45 denkbar.

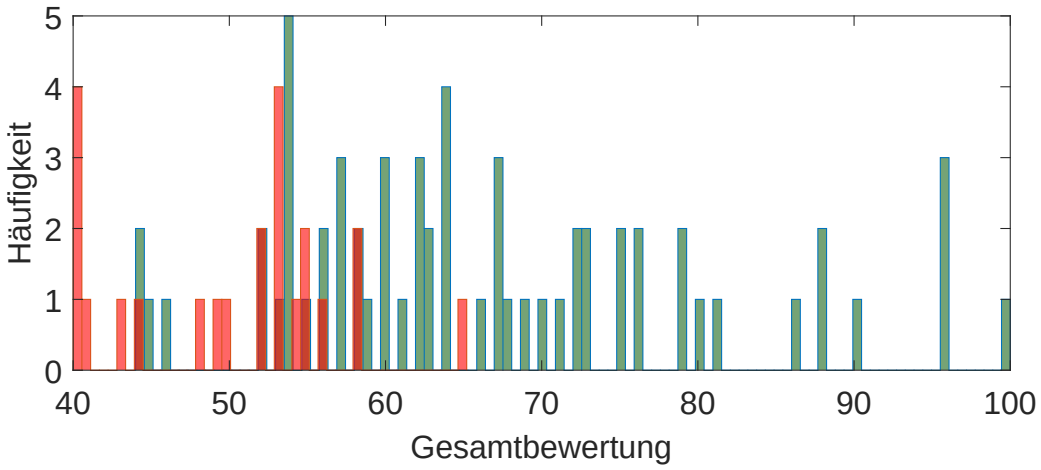

Abbildung 5.4: Gesamtbewertung richtig (grüner) und falsch (roter) erkannter Torpfostenkandidaten aus dem in diesem Kapitel verwendeten Testset

Mit einem Schwellwert von 58 lassen sich nach Abbildung 5.5 46 der 165 (28%) Torpfosten erkennen, welche im Folgenden als *True Positive* (TP) bezeichnet werden. Nur ein falscher Kandidat (0,9% unter 116 Bildern) wird fälschlicherweise als Torpfosten klassifiziert und ist somit *False Positive* (FP). Die TP-Rate ist auf die Zahl der gesamten Torpfosten bezogen, die FP-Rate ist auf die Zahl der verarbeiteten Bilder bezogen. Die hier genannten Raten entsprechen den Zielen dieser Arbeit von mindestens 25% TP-Rate und maximal 3% FP-Rate. Der Wert 58 wird in den Folgenden Analysen als geeigneter Schwellwert angesehen. Dem Nutzer des hier entwickelten Verfahrens ist es freigestellt bei anderer Zielsetzung den Schwellwert und die somit folgende Statistiken zu verändern.

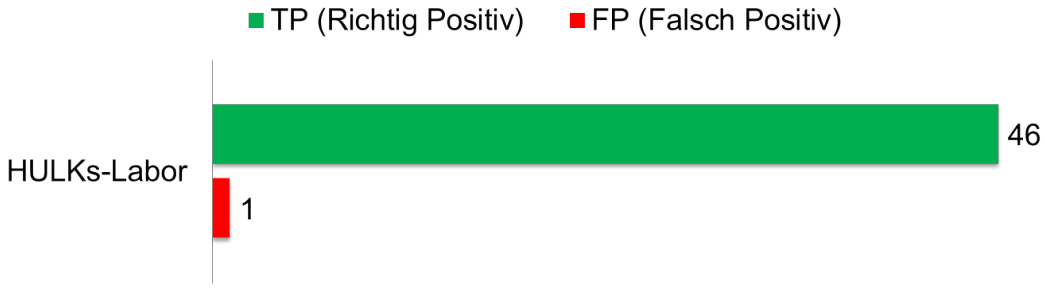

Abbildung 5.5: Anzahl der TP und FP unter 165 Torpfosten bei einem Schwellwert von 58 zur Klassfizierung als richtig

5.4 Ausgabe der Torpfosten

Ein einzelner Torpfosten Ist nur ein Pfosten lokalisiert worden, ist es nicht möglich eine Aussage darüber zu treffen, ob es sich um den linken oder rechten Torpfosten handelt. Sowohl das Tornetz als auch das Gestänge hinter dem Tor kann je nach Sichtweise sowohl rechts- als auch linksseitig der beiden Pfosten sein. Der für das menschliche Auge bestimmende Faktor hierzu stellt die Querlatte dar. Da die Querlatte nur in eine Richtung vom Pfosten abzweigt, wurde ein Algorithmus implementiert, der diese sucht. Dazu wird sowohl links, als auch rechts vom oberen Pfostenende vertikal nach einer zunächst steigenden und anschließend fallenden Flanke des Helligkeitswerts gesucht. Auf einer Seite sollte die Querlatte detektiert werden, auf der anderen Seite nicht. Der Abstand dieser Flanken ist wenig aussagekräftig, da oftmals bereits der Lichtschatten auf der Querlatte eine vorzeitige fallende Flanke hervorruft. Wird eine Querlatte nur auf einer Seite detektiert, kann der Pfosten als rechter oder linker Torpfosten klassifiziert werden. Beobachtungen im Rahmen dieser Arbeit ergeben, dass diese Suchart fehlerhaft Flanken detektiert, da neben dem Pfosten beliebige Hintergrundtexturen liegen. Ebenso werden bei hellem Hintergrund vorhandene Querlatten nicht erkannt oder aber die Querlatte ist gar nicht im Sichtbereich der Kamera.

Eine analytische Klassifizierung eines einzelnen Torpfostens als rechts oder links ist im Rahmen dieser Arbeit nicht gegeben. Dennoch sind auch aus einem einzelnen Torpfosten Positionsinformationen zu gewinnen. Zuerst lässt sich der Abstand ermitteln, womit nur noch Positionen kreisförmig um alle vier realen Torpfosten in Frage kommen. Zusätzlich ist eine Klassifizierung unter Zuhilfenahme der aktuellen Roboterlokalisierung möglich. Hierbei werden einzelne Torpfosten in Weltkoordinaten markiert. Sollte ein weiterer Torpfosten in nachfolgenden Zyklen erkannt werden, kann dieser mit Beachtung der zwischenzeitlichen Bewegung ebenso markiert werden. Befindet dieser sich nicht an einer ähnlichen Position wie der erste, sind beide Torpfosten klassifizierbar. Diese Methode stellt eine zeitliche Betrachtung der Pfostenlokalisierung dieser Arbeit dar und ist im HULKs-Framework bereits vorhanden.

Zwei Torpfosten Im Falle von zwei lokalisierten Torpfosten wird der weiter links (rechts) positionierte Torpfosten als links (rechts) klassifiziert. Hierbei soll erwähnt sein, dass die Länge der Torpfosten nicht zwingend der wahren Länge entspricht. Lediglich das untere Ende des Pfosten ist zur Lokalisation essentiell und korrekt. Es werden ebenso Torpfosten ermittelt, die den oberen Bildrand verlassen oder deren oberes Ende nicht korrekt detektiert werden konnte.

Mehr als zwei Torpfosten Sollten mehr als zwei Torpfosten auf einem Bild erkannt werden, handelt es sich um mindestens eine Fehldetektion. Um das Risiko von Falschaussagen zu minimieren, werden in diesem Fall alle Torpfosten verworfen.

6 Evaluation

Die in dieser Arbeit entworfene Pfostendetektion wird im Folgenden mit Datensets aus zwei aktuellen Wettbewerben des RoboCups verifiziert. Dabei werden Rechenzeit und die Ergebnisse der drei Hauptbestandteile dieser Arbeit, Generierung, Lokalisierung und Bewertung, hinsichtlich ihrer Anzahl richtiger und falscher Klassifizierungen analysiert. Die Klassifizierung des Bewertungsteils entspricht dem finalen Output dieser Arbeit.

German Open 2017 Abbildung 6.1 zeigt eine Aufnahme der *German Open 2017* in Magdeburg. Die Annahme variabler Hintergrundtexturen ist hier durch Zuschauer, Banden und anderen Objekten gegeben.

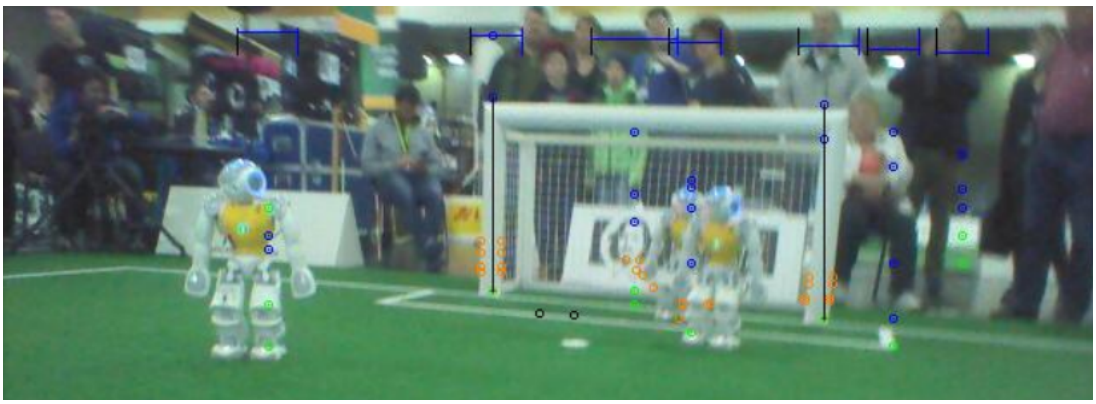

Abbildung 6.1: Exemplarische Detektion zweier Torpfosten bei den German Open 2017

Abbildung 6.2 (Erklärungen entsprechen denen aus Abbildung 5.4) zeigt eine nahezu perfekte Seperation von falschen und korrekt erkannten Pfosten. Der bestmögliche Schwellwert zur Klassifizierung liegt hier bei 48. Da eine manuelle Änderung in jeder Einsatzbedingung jedoch nicht gewollt ist, wird weiterhin die in der Validierung ermittelte Grenze von 58 verwendet.

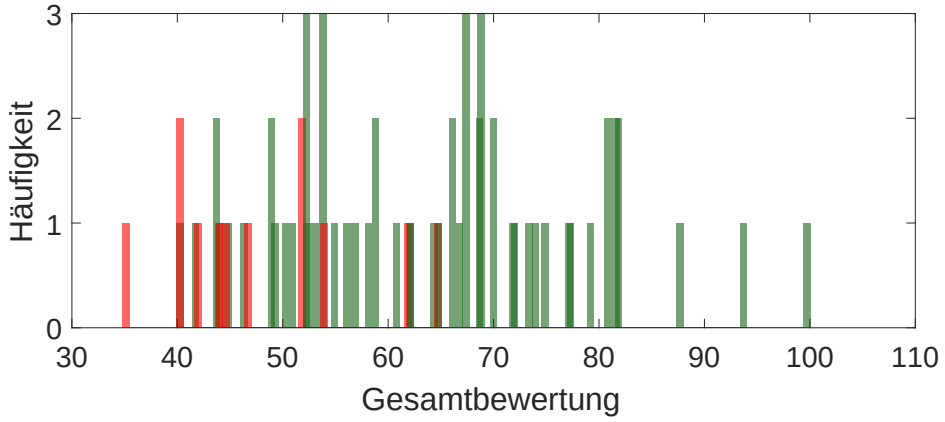

Abbildung 6.2: Gesamtbewertung richtig (grün) und falsch (rot) detektierter Torpfosten

Die Auswertung in Abbildung 6.3 zeigt eine Detektionsrate des Spaltenhistogramms (Generierung) mit 61 von 68 Torpfosten (89,7%, TP). Dabei werden insgesamt auch 163 Spalten zu den Kandidaten gezählt, obwohl sie keinen Torpfosten erhalten (FP). Diese Zahl übersteigt das in Tabelle 3 prognostizierte Verhältnis deutlich. Ausschlaggebend sind hier die umfangreichen Details im Hintergrund, die weitaus mehr Maxima im Histogramm hervorrufen als bei größtenteils homogener Hintergrundfläche. Es zeigt sich, dass die Lokalisierung diesen Effekt kompensieren kann. Denn nach der Lokalisierung sind noch 52 (76%) TPs erhalten, während die FP-Zahl auf 51 sinkt. Die nach der Bewertung über 58 entsprechenden Gesamtbewertungen resultieren in einer TP-Rate von 29,4% (20 aus 68) und keinem FP. Somit wird mindestens jeder vierte als sichtbar angenommene Torpfosten von dem Algorithmus dieser Arbeit erkannt. Die Ziele dieser Arbeit (vgl. 1.3) von 25% Erkennungsrate und maximal 3% falschen Positiven sind für diesen Wettbewerb erfüllt.

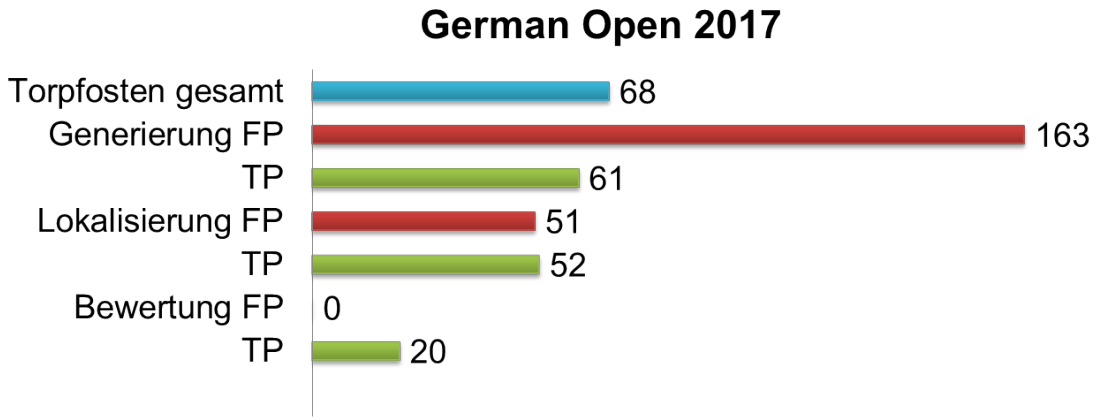

Abbildung 6.3: True Positives TP und False Positives FP nach der Generierung, Lokalisierung und Bewertung aller Torpfosten bei den German Open 2017

Sowohl bei der Generierung von Pfostenbereichen als auch bei der Lokalisierung und der Bewertung entweichen dem Algorithmus TPs. Beachtet man, dass die Torpfosten sowohl verschwommen sein, schief im Bild liegen und vor gleichfarbigem Hintergrund auftreten können, gibt es kaum analytische Kriterien, die alle drei Herausforderungen einbezieht und die Umgebung isoliert. Das Vorgehen dieser Arbeit, mehrere Kriterien zu überprüfen, hat den Vorteil, dass jeder dieser drei Fälle mit einbezogen wird. Der Nachteil ergibt sich jedoch dadurch, dass nur selten alle Kriterien erfüllt werden können. Da auch falsche pfostenähnliche Objekte die Kriterien teilweise erfüllen können, erreichen sie oftmals ähnliche Gesamtbewertungen wie die korrekten Torpfosten. Der Wunsch falsche Klassifizierungen auszuschließen impliziert somit auch ein Wegfall von vielen richtigen Pfosten.

Möchte man Ursachen für die TP-Verluste der Generierung, Lokalisierung und Bewertung konkret benennen, ist eine Betrachtung der Bilder erforderlich. Dabei lassen sich folgende Hauptursachen herausstellen. Abbildung 6.4 demonstriert korrekte Torpfosten, die einen Schwellwert von 58 nicht erreichen.

- *Generierung:* Verluste deuten darauf hin, dass der Torpfosten sich in seiner Helligkeit mit Betrachtung der Spalten nicht vom Hintergrund abheben kann oder die Luminanz des Torpfosten den Schwellwert nicht überschreitet und somit im Histogramm kein Maximum darstellt (Abbildung 6.4, links).
- *Lokalisierung:* Die Ursache für Fehler der Lokalisierung liegt darin, dass das obere oder untere Pfostenende nicht erkannt wird. Dies geschieht bei verschwommenen Bildern oder schief im Bild liegenden Torpfosten (Abbildung 6.4, mitte).
- *Bewertung:* Die Fehlerursachen des Bewertungsteils sind vielfältig, können jedoch oftmals auf Unschärfe zurückgeführt werden. Bei unscharfen Bildern kann die Pfostenbreite nicht exakt ermittelt werden, wodurch das hoch gewichtete *Höhe zu Breite*-Kriterium nicht zur Anwendung kommen kann (Abbildung 6.4, rechts).

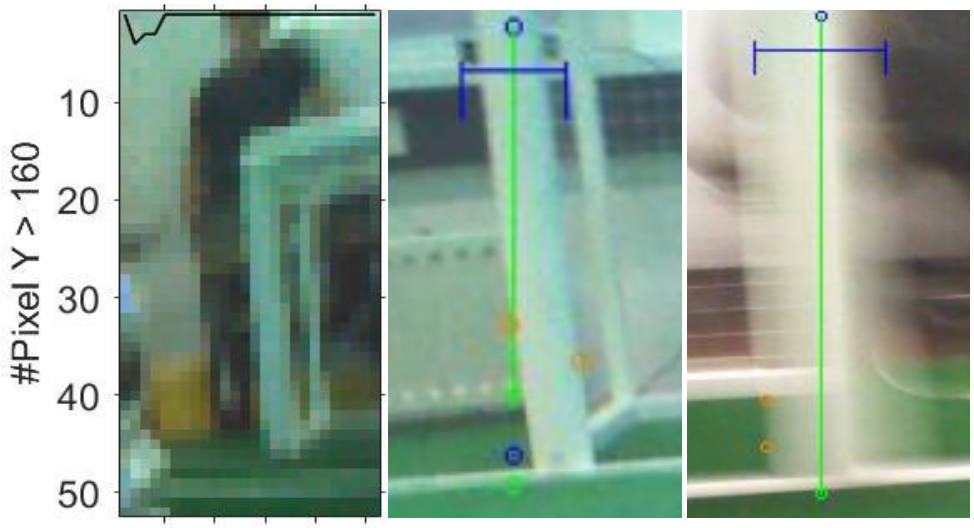

Abbildung 6.4: Fehlerursachen bei Generierung (links, Torpfosten zu dunkel), Lokalisierung (mitte, Schieflage) und Bewertung (rechts, keine Breitendetektion)

Iran 2017 Eine Demonstration der Hintergrundumgebung bei den *Iran Open 2017* zeigt Abbildung 6.5. Eine durchgehend weiße Bande hinter dem Tor erschwert die Isolierung der Torpfosten von der Umgebung. Zudem ist dies ein Beispiel für die oftmals verschwommenen Kameraaufnahmen des NAOs, welche trotz dieses Umstands durch die Methoden dieser Arbeit korrekt klassifiziert werden können.

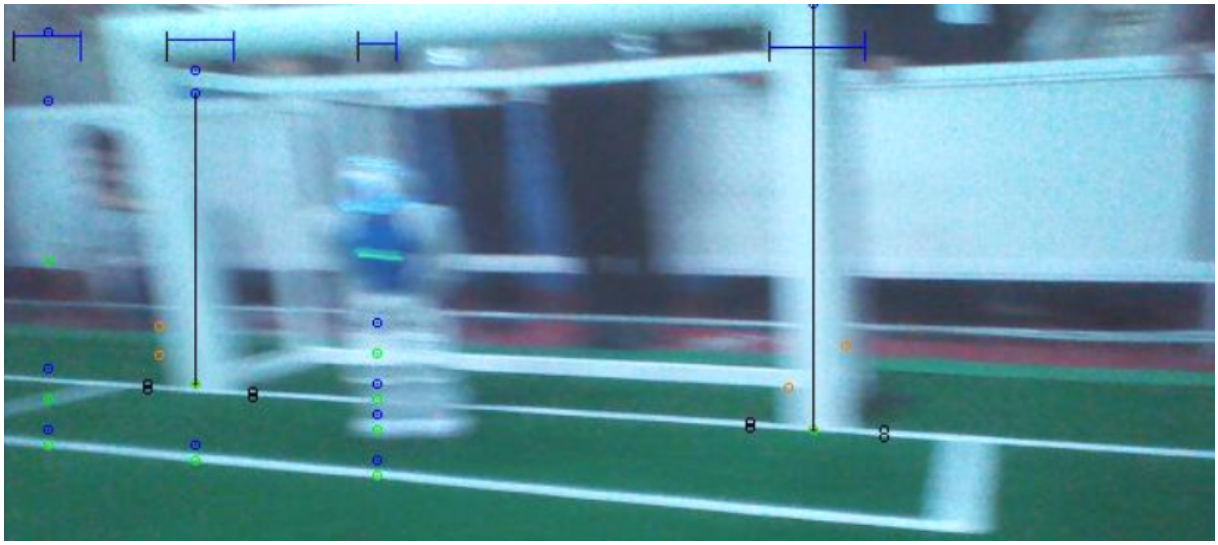

Abbildung 6.5: Exemplarische Detektion zweier Torpfosten bei den Iran Open 2017

Die archivierten Bilddaten der Iran Open 2017 enthalten 48 Torpfosten. Abbildung 6.6 ist zu entnehmen, dass der festgelegte Schwellwert von 58 in zwei FPs resultiert, da zwei rote Balken rechtsseitig dieser Gesamtbewertung liegen.

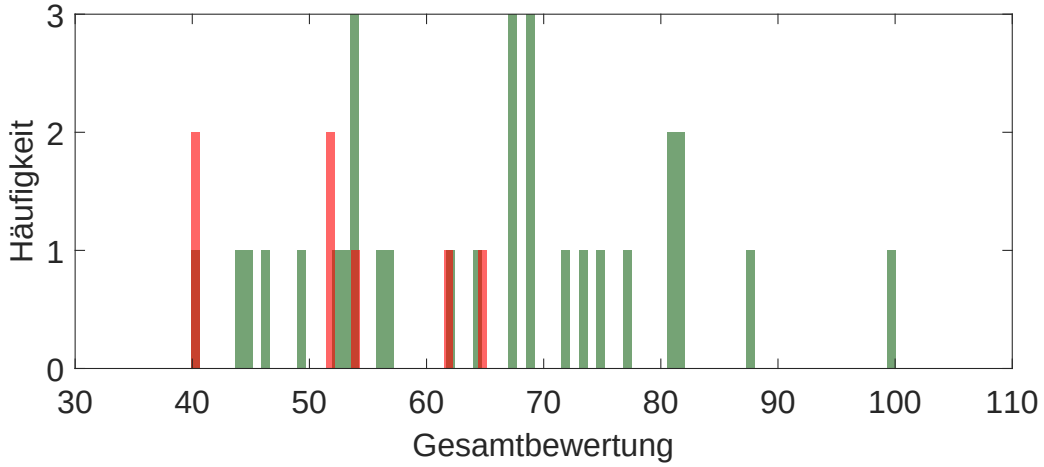

Abbildung 6.6: Gesamtbewertung richtig (grün) und falsch (rot) detektierter Torpfosten

Die Generierung erkennt 42 der 48 Torpfosten korrekt (Detektionsrate 88%) und erzeugt währenddessen 109 falsche Kandidatenbereiche (vgl. Abbildung 6.7). Diese FPs können während der Lokalisierung auf 45 reduziert werden, während vier korrekte Kandidaten

nicht erkannt werden. Die Abbildung zeigt zudem, dass der Gesamtalgorithmus dieser Arbeit 18 der 48 Torpfosten (37,5%) korrekt klassifiziert. Während dies die Zielerfüllung sogar um 12% übertrifft, liegt die FP-Rate unter 39 Bildern bei 5% und somit höher als gewünscht. Die Ursachen werden in Kapitel 6.1 benannt.

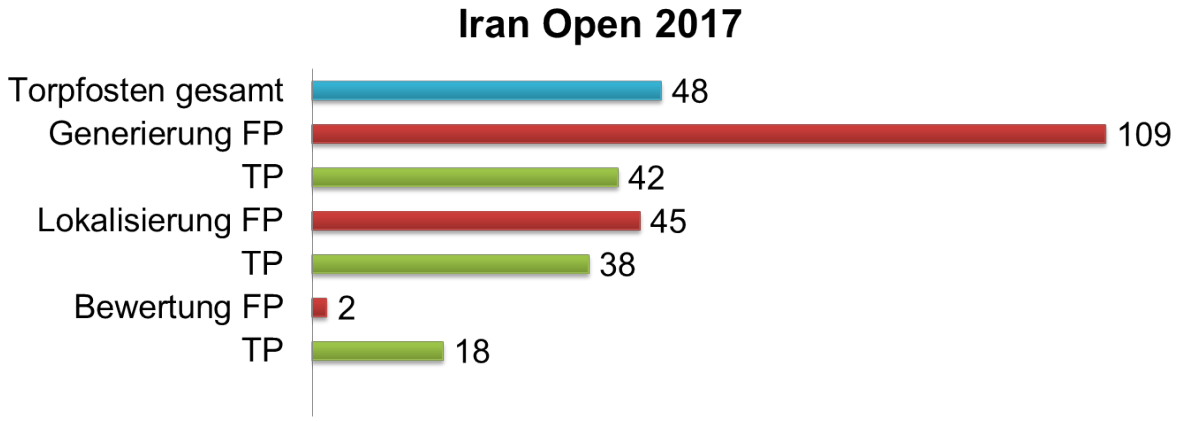

Abbildung 6.7: True Positives TP und False Positives FP nach der Generierung, Lokalisierung und Bewertung aller Torpfosten bei den Iran Open 2017

6.1 Falsche Positive

Eine analytische Objektdetektion impliziert immer eine begrenzte Zahl an Überprüfungen. Treten Umrisse auf, die einem Torpfosten qualitativ ähneln und die implementierten Kriterien erfüllen, kann eine Fehldetektion nicht ausgeschlossen werden. Abbildung 6.8 zeigt drei Beispiele falscher Positive (FP) aus den Testsets dieser Arbeit. Die schwarzen, bzw. grüne Linie stellen die falsch detektierten Torpfosten dar. Der linke Teil der Abbildung entspricht dem FP bei den Iran Open mit einer Gesamtbewertung von 65 (vgl. Abbildung 6.6 und 6.7). Auch der zweite FP dieses Wettbewerbes ist auf die weiße Bande zurückzuführen, da diese bei allen generierten Spalten implizit Startpunkte und Endpunkte beschreibt, die mit Beachtung des Horizonts gültig sind. Wird dann fälschlicherweise wie mit den orangenen Kreisen dargestellt eine passende Breitendetektion durchgeführt, wird dieser Kandidat als Torpfosten klassifiziert.

Die in Abbildung 6.8 mittig und rechts gezeigten Fälle zeigen Situationen, in denen Personen (meist nur temporär) einen pfostenähnlichen Umriss bilden. Auch hier treffen die analytischen Kriterien (z.B. die durch die orangenen Kreise gekennzeichnete Breitendetektion) dieser Arbeit fälschlicherweise zu.

Des Weiteren kommt es trotz des in Kapitel 5.1 eingeführten Stützpfostenkriteriums zu falschen Klassifizierung von Stützpfosten. Sollte der korrekte Torpfosten nicht detektiert werden, kann auch der Stützpfosten nach diesem Kriterium nicht als solcher erkannt werden. Wird dann auf Grund von Unschärfe oder Schieflage im Bild eine geringere Höhe ermittelt, gleicht der Kandidat einem realen Pfosten.

Alle FPs haben gemeinsam, dass das Höhe zu Breite Verhältnis einem richtigen Torpfosten ähnlich ist und auch der Helligkeitswert den Ansprüchen des Verfahrens genügt.

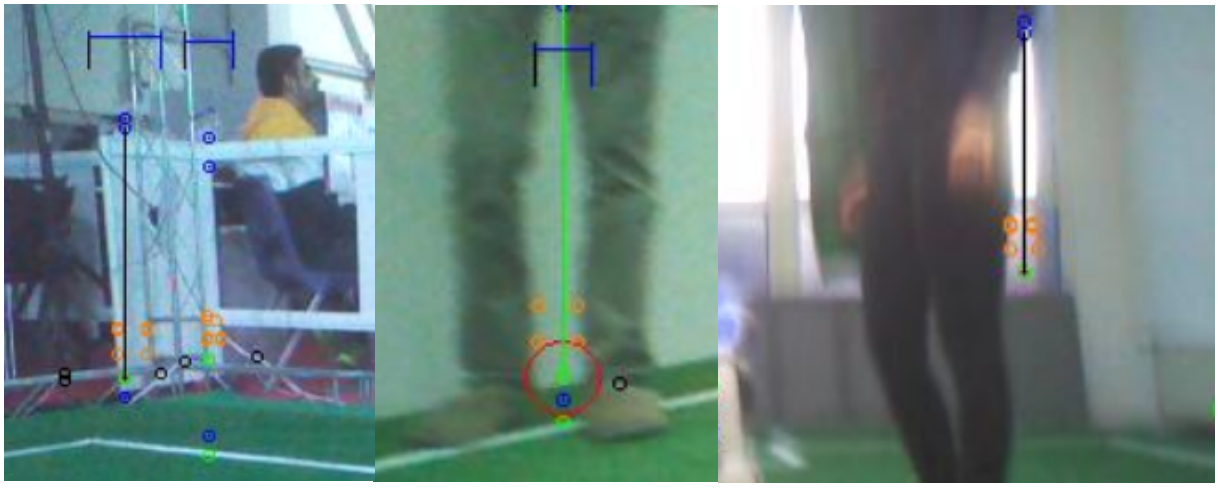

Abbildung 6.8: Exemplarische False Positives (FPs) aus den in dieser Arbeit verarbeiteten Datensätzen

6.2 Rechenaufwand

In diesem Kapitel wird der Rechenaufwand des entwickelten Algorithmus evaluiert (vgl. Abbildung 6.9). Während der Zeitmessung läuft der NAO im Versuchslabor auf ein Tor zu, wodurch in den Aufnahmen fortlaufend Torpfosten enthalten sind.

Der Abschnitt der Generierung, also das Erstellen des Spaltenhistogramms und die verbundene Suche der Maxima, benötigt durchschnittlich 0,42 ms und im Maximum 0,84 ms. Diese große Abweichung von Maximum zu Durchschnittswert ist darauf zurückzuführen, dass je nach Höhe des Horizonts unterschiedlich viele Bildpunkte für das Histogramm untersucht werden müssen (lediglich der Bildbereich bis 50 Pixel über dem Horizont wird betrachtet, vgl. Kapitel 3.5).

Die Lokalisierung und Bewertung der Pfostenkandidaten entsprechen mit einer Laufzeit von 0,2 ms im Durchschnitt und 0,35 ms maximal lediglich einem Drittel der Gesamtlaufzeit. Sie werden aus diesem Grund, und weil übergreifende Programmcodes eine unpräzise Laufzeitmessung verursachen, nicht einzeln betrachtet. Würde das Kamerabild eine weiße Wand abbilden, werden keine Kanten und keine Torpfostenkandidaten ermittelt. Die Rechenzeit geht in diesen Fällen gegen Null. Mit Sicht auf andere NAOs oder Zuschauer erhöht sich der Rechenaufwand.

Die Gesamtlaufzeit wurde separat ermittelt und weicht daher in geringem Maße von der Summe aller drei Bestandteile ab. Die Echtzeitanforderungen dieser Arbeit erfordern eine maximale Rechenzeit von 2,0 ms. Über alle Iterationen ergibt sich ein durchschnittlicher Rechenaufwand des gesamten Verfahrens von 0,68 ms und ein Maximum von 1,11 ms, was die Zielvorstellungen dieser Arbeit übertrifft.

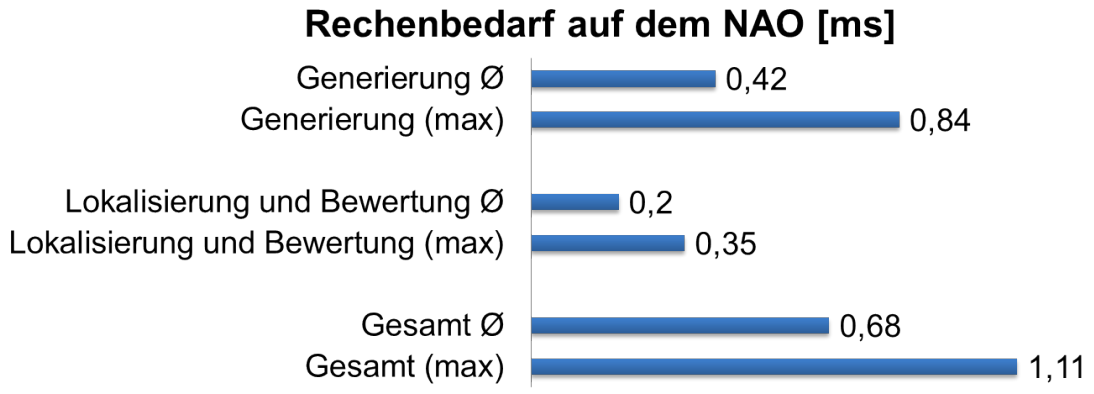

Abbildung 6.9: Durchschnittliche und maximale Rechenzeit der Teilabschnitte sowie des Gesamtmoduls dieser Arbeit

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein analytisches Verfahren zur Lokalisierung von weißen Torpfosten in einem Kamerabild entwickelt. Das wichtigste Merkmal stellt hierbei das untere Pfostenende dar, mit dessen Information die derzeit implementierte Selbstlokalisierung des Roboters unterstützt werden soll. Dabei dienten, im Gegensatz zu derzeit veröffentlichten Arbeiten, die realen Einsatzbedingungen des RoboCups zur Entwicklung und Evaluierung.

Das entwickelte Verfahren beruht auf drei Teilmodulen. Zunächst werden Teilbereiche mit möglichen Torpfosten generiert, anschließend das untere Pfostenende exakt bestimmt und schlussendlich eine Gesamtbewertung an Hand von unterschiedlichen Kriterien ermittelt. Eine Empfehlung der Gesamtbewertung, über der ein Torpfosten als solcher klassifiziert wird, wurde in dieser Arbeit ermittelt (58).

Mit Hilfe der Aufzeichnungsdaten aus zwei Wettbewerben des Jahres 2017 konnte ermittelt werden, dass die Lokalisierung zwischen 29, bzw. 38% aller Torpfosten am unteren Ende korrekt lokalisiert hat. Dies entspricht mindestens jedem vierten Bild und erfüllt somit die Zielvorstellung dieser Arbeit von 25%. Dabei enthalten die Daten entgegen aller bisherigen Veröffentlichungen im Rahmen des RoboCups verschwommene, unscharfe Aufnahmen der Torpfosten.

Eine weitere zuvor nicht zufriedenstellend gelöste Herausforderungen bestand in der Torpfostenlokalisierung bei gleichfarbigen Hintergrundtexturen. Auch unter diesen Umständen konnte eine Detektionsrate von 28% im Versuchslabor nachgewiesen werden.

Falsch klassifizierte Bereiche ohne reale Torpfosten wurden in den Wettbewerbsdaten in 0, bzw. 5%, im Labor in 0,9% aller Bilder festgestellt. Die Zielsetzung von maximal 3% ist wettbewerbesübergreifend erfüllt, kann jedoch in bestimmten Spielfeldumgebungen bei Vorhandensein von pfostenähnlichen Strukturen im Hintergrund überschritten werden. Dem Nutzer dieses Verfahrens steht die Möglichkeit zur Verfügung, den Schwellwert zur Klassifizierung als Torpfosten zu verändern. Somit kann der Anteil falsch klassifizierter Torpfosten den Ansprüchen des Nutzers angepasst werden.

Der Rechenzeit auf der Zielhardware dieser Arbeit beträgt durchschnittlich 0,68 ms und im Maximum 1,11 ms. Die Echtzeitfähigkeit zur Integration in das HULKs-Framework ist demnach gegeben.

Diese Arbeit bietet neben seiner Gesamtfunktionalität auch die Möglichkeit zur Nutzung von Teilmodulen. Auf Grund der unabhängigen Auswahl von Generierungs-, Lokalisierungs- und Bewertungsmethodik können Abschnitte dieser Arbeit mit wenig Aufwand erweitert, ersetzt oder für andere Zwecke genutzt werden.

7.2 Ausblick

Das in dieser Arbeit entwickelte Verfahren kann im HULKs-Framework implementiert und genutzt werden. Um ein möglichst synergisches Zusammenspiel zwischen der derzeitigen Roboterlokalisierung und der Torpfostenlokalisierung dieser Arbeit zu gewährleisten, ist die Schnittstelle zwischen diesen festzulegen. Dazu zählt zum Beispiel der Schwellwert zur Klassifizierung als Pfosten. Es ist ebenso denkbar alle bewerteten Torpfostenkandidaten mit der zugehörigen Gesamtbewertung auszugeben. Anschließend kann jeder Kandidat nach seiner Bewertung gewichtet in den Lokalisierungsfilter eingehen.

Es ist denkbar, dass Modifizierungen des hier entwickelten Verfahrens zu besseren Ergebnissen führen. Eine mögliche Modifizierung stellt ein neuer Algorithmus zur Breitendetektion dar, der auch am oberen Pfostenende bei hellen Hintergrundtexturen zuverlässige Ergebnisse liefert. Dieser könnte sowohl die Rate der True Positives erhöhen, als auch die Rate der False Positives reduzieren. Letzteres kann auch durch eine andere oder erweiterte Isolierung der Stützpfosten geschehen.

Des Weiteren stellt sich in dieser Arbeit die Rechts-/Links-Klassifizierung eines einzelnen Torpfostens als nicht zuverlässig heraus. Eine korrekte Klassifizierung reduziert die möglichen Roboterpositionen und würde zu robusterer und schnellerer Ermittlung dessen führen.

Literatur

[1] Yuv and luminance considered harmful. In *Poynton (Hg.) 2012*, pages 567–572. URL http://www.sciencedirect.com/science/article/pii/B9780123919267500667.

[2] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2014. ISBN 0262028182.

[3] Anastasia Bolotnikova. *Melioration of color calibration, goal Melioration and self-localization systems of NAO humanoid robots*. Bachelorarbeit, University of Tartu, 2015. URL http://icv.tuit.ut.ee/images/nana/thesis.pdf.

[4] Andreas Lander. Robocup german open 2017, 5.-7. mai 2017: https://www.robocupgermanopen.de/, 2017. URL https://www.robocupgermanopen.de/sites/default/files/styles/ddslide/public/slideshow/RoboCup-German-Open-2017_131_Foto_Andreas_Lander_SPL.png?itok=6irG_fQ1.

[5] B-Human Team. Team report and code release 2015, 2015. URL https://www.b-human.de/downloads/publications/2015/CodeRelease2015.pdf.

[6] B-Human Team. Team report and code release 2016, 2016. URL https://github.com/bhuman/BHumanCodeRelease/raw/master/CodeRelease2016.pdf.

[7] José María Cañas Plaza, Eduardo Perdices García, Tomás González Sánchez, and Domenec Puig Valls. Recognition of standard platform robocup goals, 2010. URL https://rua.ua.es/dspace/bitstream/10045/13291/1/JoPha_4_1_03.pdf.

[8] Pablo Cano, Yoshiro Tsutsumi, Constanza Villegas, and Javier Ruiz-del Solar. Robust detection of white goals. In Luis Almeida, Jianmin Ji, Gerald Steinbauer, and Sean Luke, editors, *RoboCup 2015: Robot World Cup XIX*, pages 229–238. Springer International Publishing, Cham, 2015. ISBN 978-3-319-29339-4. doi: 10.1007/978-3-319-29339-4_19. URL https://doi.org/10.1007/978-3-319-29339-4_19.

[9] David G. Lowe. Object recognition from local scale-invariant features, 1999. URL http://www.cs.ubc.ca/~lowe/papers/iccv99.pdf.

[10] Dirk-Jan Kroon. Opensurf (including image warp): Surf (speeded up robust features) image feature point detection / matching, 2010. URL http://de.mathworks.com/matlabcentral/fileexchange/28300-opensurf--including-image-warp-.

[11] Gerhard Lakemeyer, Elizabeth Sklar, Domenico G. Sorrenti, and Tomoichi Takahashi. *RoboCup 2006: Robot Soccer World Cup X*. Springer, 2007. ISBN 0302-9743.

[12] Hannes Fassold and Jakub Rosner. A real-time gpu implementation of the sift algorithm for large-scale video analysis tasks, 2015. URL http://hgpu.org/papers/13457_20150212031803.pdf.

[13] Alexander Härtl, Ubbo Visser, and Thomas Röfer. Robust and efficient object recognition for a humanoid soccer robot. In Sven Behnke, Manuela Veloso, Arnoud Visser, and Rong Xiong, editors, *RoboCup 2013: Robot World Cup XVII*, pages 396–407. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-662-44468-9. doi: 10.1007/978-3-662-44468-9_35.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: speeded up robust features, 2008. URL http://www.sciencedirect.com/science/article/pii/S1077314207001555?via%3Dihub.

[15] Joachim Hertzberg, Kai Lingemann, and Andreas Nüchter. *Mobile Roboter: Eine Einführung aus Sicht der Informatik*. SpringerLink Bücher. Springer Vieweg, Berlin, Heidelberg, 2012. ISBN 9783642017261. URL http://dx.doi.org/10.1007/978-3-642-01726-1.

[16] Karl Tarvas. *Edge information based object detection and classification*. Bachelorarbeit, University of Tartu, 2016. URL https://www.tuit.ut.ee/sites/default/files/tuit/atprog-courses-bakalaureuset55-loti.05.029-karl-tarvas-text-20160520.pdf.

[17] Nao Devils Dortmund. Team report 2016, 2016. URL https://github.com/NaoDevils/CodeRelease2016/blob/master/TeamReportNaoDevils2016.pdf.

[18] Nathapon Olaf Lüders. *Object Localization on the Nao Robotic System Using a Deep Convolutional Neural Network and an Image Contrast Based Approach*. Masterarbeit, Technische Universität Hamburg, Hamburg, 2016-08-15. URL http://www.hulks.de/_files/MA_Olaf_Lueders.pdf.

[19] Pascal Loth. Implementierung und evaluation einer robusten echtzeitkantendetektion auf dem humanoiden nao-robotiksystem: Bachelorarbeit. URL http://www.hulks.de/_files/BA_Pascal-Loth.pdf.

[20] Finn Poppinga. *Implementation and Evaluation of Audio Based Methods for Robust Inter-Robot Communication*. Projektarbeit, Helmut Schmidt Universität, Hamburg, 2016-07-29. URL http://www.hulks.de/_files/PA_Finn-Poppinga.pdf.

[21] RoboCup Technical Committe. Robocup standard platform league (nao) rule book, 2017. URL http://spl.robocup.org/wp-content/uploads/downloads/Rules2017.pdf.

[22] Softbank Robotics. Robots: Who is nao?, . URL https://www.ald.softbankrobotics.com/en/cool-robots/nao.

[23] Softbank Robotics. Robocup 2017, . URL https://www.ald.softbankrobotics.com/en/robocup-2017.

[24] Softbank Robotics Corp. Nao datasheet. URL https://www.ald.softbankrobotics.com/sites/aldebaran/files/nao_datasheet.pdf.

[25] Souzana Volioti and Michail G. Lagoudakis. Histogram-based visual object recognition for the 2007 four-legged robocup league, 2008. URL https://link.springer.com/chapter/10.1007/978-3-540-87881-0_28.

A Anhang

A.1 Matlab Code zur Kantendetektion mit Sobel und Hough

```
%Bild im YUV-Format
I = imread('top_image_xxxx.png');

%Erzeuge Schwarz-Weiß-Bild
Igrey = I(:, :, 1);
%Falls Unterabtastung gewünscht (hier 5 vert. und horiz.)
Igrey=Igrey(1:5:end,1:5:end,:);
Irgb =Irgb(1:5:end,1:5:end,:);

%Binäres Bild erzeugen
BW = edge(Igrey,'sobel',0.05,'vertical');
%Hough Transformation zur Liniendetektion
[H,T,R] = hough(BW, 'Theta', -20:1:20);
%Maxima der Hough-Transformation auslesen
P  = houghpeaks(H,8,'threshold',ceil(0.5*max(H(:))));
x = T(P(:,2)); y = R(P(:,1));
%Linien auch mit Unterbrechung detektieren
lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',15);

%Plotte gefundene Linien
max_len = 0;
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

   % Plot Anfang und Ende der Linie
   plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
   plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');

   % Bestimmte den Endpunkt des längsten Segments
   len = norm(lines(k).point1 - lines(k).point2);
   if ( len > max_len)
      max_len = len;
      xy_long = xy;
   end
end
```

A.2 Matlab Code zur Generierung eines Spaltenhistogramms

```
%Bild im YUV-Format
I = imread('top_image_xxxx.png');
%Definiere Horizont (ganzzahlig), wenn notwendig, sonst 1
horizon = 1;
%Schwarz-Weiß Bild, sowie Cb (U) und Cr (V) Kanal
Igrey = I(:, :, 1);
IU = I(:, :, 2);
IV = I(:,:, 3);

%Unterabtastung, wenn notwendig (hier 10 horiz. und 5 vert.)
Igrey=Igrey(1:5:end,1:10:end,:);
IU=IU(1:5:end,1:10:end,:);
IV=IV(1:5:end,1:10:end,:);

%Zur Visualisierung des Bildes
Irgb = ycbcr2rgb(I);
%Unterabtastung, wenn notwendig, auch hier übernehmen
Irgb = Irgb(1:5:end,1:10:end,:);

%Größe festlegen
Hist = 1:length(Igrey(1,:));

%Für alle Spalten
for col = 1:length(Igrey(1,:))
    %Zählt die hellen Pixel
    counter = 1;
    %Für alle Zeilen
    for row = horizon:length(Igrey(:,1))
        %Option 1 - Konstanten Y-Wert als Schwellwert
       %if Igrey(row,col) > 160  %Option1
        %Option 2 - Farbmodelleingrenzung
        if (Igrey(row,col) > 200 && IU(row,col) > 80 && IU(row,col) < 256 && ...
           IV(row,col) > 70 && IV(row,col) < 190)
            counter = counter + 1;
        elseif (Igrey(row,col) > 175 && IU(row,col) > 100 && IU(row,col) < 160 && ...
           IV(row,col) > 80 && IV(row,col) < 160)
            counter = counter + 1;
        elseif (Igrey(row,col) > 160 && IU(row,col) > 110 && IU(row,col) < 150 && ...
           IV(row,col) > 90 && IV(row,col) < 130)
        %Nächste Zeile immer einkommentiert lassen
            counter = counter + 1;
        end
    end
    Hist(col) = counter;
end

%Abstand, in der das lokale Maximum ein globales Maximum sein muss
persistence = round((5/128)*length(Hist));
%Findet alle Extrema unter Beachtung der persistence
%findmax wurde selbst geschrieben und ist keine bereits vorhandene Fkt
extremaIndices = findmax(Hist, persistence);
```

```
%Zur Visualisierung abspeichern
persistentIndices = extremaIndices;

%%Auf beiden Seiten nach einer fallenden Flanke suchen
%Maximaler Suchbereich
inner_edge_max = (8/128)*length(Hist);
found_left_ = false;
found_right_ = false;
h_last_ = 0;
h_ = 0;
%Muss angepasst werden (Für Sampleabstand(1) = 2, (5) = 4, (10) = 8)
%beschreibt die minimale Steigung zur Klassifizierung als Flanke
h_diff_min_ = 4;
mydata = Hist;

for k = 1:length(extremaIndices)
    %Suche links
    found_left_ = false;
    h_last_ = mydata(extremaIndices(k));
    for subit = 1:inner_edge_max
        search_it = extremaIndices(k) - subit;
        if(search_it ≤ 0)
            break;
        end
        h_ = mydata(search_it);
        if(h_last_ - h_ ≥ h_diff_min_)
            found_left_ = true;
            break;
        end
        h_last_ = h_;
    end
    %Suche rechts
    found_right_ = false;
    h_last_ = mydata(extremaIndices(k));
    for subit = 1:inner_edge_max
        search_it = extremaIndices(k) + subit;
        if(search_it ≥ length(mydata))
            break;
        end
        h_ = mydata(search_it);
        if(h_last_ - h_ ≥ h_diff_min_)
            found_right_ = true;
            break;
        end
        h_last_ = h_;
    end
    %Wenn nur auf einer Seite gefunden, dann mit 0 kennzeichnen
    if(found_left_ == false || found_right_ == false)
        persistentIndices(k) = 0;
    end
end

%Sortiere 0 aus
persistentIndices = persistentIndices(persistentIndices≠0);
```

```
104 % Plot all features

106 imshow(Irgb);
107 hold on;
108 plot(mydata,'LineWidth',1,'color','black');
109 ylabel('#Pixel Y > 160')
110 xlabel('Bildspalte')
111 set(gca,'XTickLabel',[]);

113 % Markierungen (Punkte) im Histogramm hinzufügen
114 markers = mydata(extremaIndices);
115 scatter(extremaIndices, markers,100,'black','fill');
116 markers = mydata(persistentIndices);
117 scatter(persistentIndices, markers,100,'red','fill');
118 axis on;

120 set(gca,'color','none');
```

A.3 Pseudocode für Pfostenlokalisierung innerhalb einer Spalte

Algorithm 1 Pfostenlokalisierung innerhalb einer Spalte

Input: *Spalte*

Output ← *Kandidaten*

for every 5th *Pixel* **in** *Spalte* **do**
  if *RisingEdge* **then**
    if $Differenzzurletzten RisingEdge < 10$ **then**
      Lösche letzte Rising Edge
    end if
    AlleRisingEdges ← *RisingEdge*
  else if *FallingEdge* **then**
    for *AlleRisingEdges* **do**
      if $Distanz(Falling - Rising) < 20$ **then**
        Lösche Falling und Rising Edge
        Break
      else if $Distanz(Falling - Rising) < 40$ **then**
        Lösche Falling Edge
        Break
      else*Kandidaten* ← $RisingundFallingEdge$
      end if
    end for
  end if
end for

A.4 Eingrenzungen des Suchbereichs im YCbCr-Farbmodell für das Spaltenhistogramm

$$\text{Inkrement} = \begin{cases} 1 & Y \geq 200 \;\&\; 80 \leq \text{Cb} \leq 256 \;\&\; 70 \leq \text{Cr} \leq 190 \\ 1 & 175 \leq Y < 200 \;\&\; 100 \leq \text{Cb} \leq 160 \;\&\; 80 \leq \text{Cr} \leq 160 \\ 1 & 160 \leq Y < 175 \;\&\; 110 \leq \text{Cb} \leq 150 \;\&\; 90 \leq \text{Cr} \leq 130 \\ 0 & \text{sonst} \end{cases}$$

A.5 Ermittlung von $Score_G$ aus α_1 und α_2

Input: α_1, α_2
1: $Output \leftarrow Score_G$
2:
3: Ermittlung von $Score_{G1}$ und $Score_{G2}$
4: aus Gleichung 5.7
5:
6: $Score_G = Score_{G1} + Score_{G2}$
7: ($Score_G$ im Folgenden maximal 2)
8:
9: Wenn die beiden Breiten sich ähneln...
10: **if** $\alpha_1/\alpha_2 < 1.25$ und $\alpha_1/\alpha_2 > 0.8$ **then**
11: **if** $\alpha_1 < 0$ oder $\alpha_2 < 0$ **then**
12: Negativer Score darf nicht quadriert werden
13: $Score_G = 2 \cdot Score_G$
14: **else**
15: $Score_G = Score_G \cdot Score_G$
16: $Score_G$ im Folgenden maximal vier
17: **end if**
18: Wenn die beiden Breiten sich stark unterscheiden...
19: **else if** $\alpha_1/\alpha_2 < 0.65$ und $\alpha_1/\alpha_2 > 1.5$ **then** $Score_G = -4$
20: **else**$Score_G = Score_G$
21: **end if**
22: $Score_G = Score_G/4$
