Hochschule für Technik, Wirtschaft und Kultur Leipzig
Fakultät Informatik, Mathematik und Naturwissenschaften

Masterarbeit

Kalibrierungsfreie Bildverarbeitungsalgorithmen zur echtzeitfähigen Objekterkennung im Roboterfußball

vorgelegt von

B. Sc. Thomas Reinhardt

Leipzig, Oktober 2011

Betreuender Professor: Prof. Dr. rer. nat. habil. Karl-Udo Jahn

Selbstständigkeitserklärung

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

Leipzig, den 21.10.2011

_____

Kurzfassung

Die autonome Handlungsfähigkeit von Robotern beruht auf der Wahrnehmung und Verarbeitung von Informationen aus ihrer Umwelt. Die Signalerfassung erfolgt über integrierte Sensoren, die Signalverarbeitung durch effiziente Programme, die auf die gewünschten Verhaltens- und Handlungsmuster ausgerichtet sind. Dem humanoiden Fußballroboter "NAO" stehen hierfür unter anderem zwei integrierte Kameras zur Verfügung. Wie die so erzeugten Bilder durch speziell entwickelte Algorithmen analysiert werden und wie es dem Roboter damit möglich ist, Fußbälle, Tore oder das Spielfeld und dessen Begrenzungslinien zu erkennen, ist Thema der vorliegenden Arbeit. Die größte Herausforderung bei dieser Aufgabenstellung sind wechselnde Lichtverhältnisse, die es erschweren oder unmöglich machen, Objekte ausschließlich anhand ihrer Farbe zu klassifizieren. In dieser Arbeit wurden echtzeitfähige Algorithmen entwickelt, die durch Hinzunahme von Wissen über Objektformen auch unter variablen bzw. unbekannten Lichtverhältnissen robuste Ergebnisse liefern und deren Anwendung im Gegensatz zu vielen anderen Verfahren keiner Farbkalibrierung bedarf.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Die folgende Tabelle beinhaltet sowohl einen Überblick über in dieser Arbeit häufig verwendete Symbole als auch die jeweiligen Seitenverweise für deren erste Erwähnung.

Y, Cb, Cr	Farbwerte im Bereich [0..255]	2.3.3
$Y_{field}, Cb_{field}, Cr_{field}$	konkrete Werte der Feldfarbe	3.4.3
$Y_{line}, Cb_{line}, Cr_{line}$	konkrete Werte der Feldlinienfarbe	3.5.5
$Y_{ball}, Cb_{ball}, Cr_{ball}$	konkrete Werte der Ballfarbe	3.6.4
$Y_{goal}, Cb_{goal}, Cr_{goal}$	konkrete Werte der Torfarbe	3.7.2
n_B	Anzahl der Testbilder	3.1
s_{Raster}	Pixelabstand von benachbarten Scanlines	3.4.3
T_y, T_{cb}, T_{cr}	Schwellenwerte bei der Feldfarbenerkennung	3.4.4
P	ein Punkt mit den Koordinaten P_x und P_y	3.2
p_r, p_o, p_c	Wahrscheinlichkeiten	3.6.5
V	binäre Funktion zur Auswertung des Vorzeichens	3.4.1
d_{line}	Breite der Spielfeldlinien in mm	3.5.1
d_{ball}	Durchmesser des Spielballs in mm	3.6.1
h_{cam}	Höhe der Kamera über dem Spielfeldboden in mm	3.5.1
α_{fov}	Öffnungswinkel der Kamera	3.5.1
g	Gradientenvektor	3.5.3

1 Einleitung

Komplexe Bildverarbeitungssysteme finden breiten Einsatz in Industrie, Medizin, Astronomie, zur Verkehrsanalyse und in vielen weiteren Bereichen. So wird heutzutage die Qualitätssicherung in der Produktherstellung oft von kamerabasierten Systemen unterstützt. Als Beispiel sei die Solarzellenfertigung erwähnt, bei der produktionsbedingte lokale Defekte wie Kratzer oder Dellen auf den Solarmodulen in einem automatisierten Prozess visuell erkannt werden können. In der Medizin dienen eine Reihe von bildgebenden Verfahren wie MRT und CT der Diagnostik und greifen auf verschiedene Bildverarbeitungsalgorithmen zur Erstellung, Filterung oder Korrektur der Messdaten zurück. In der Automobilbranche helfen Fahrerassistenzsysteme straßenverkehrsbedingte Risiken zu minimieren und den Fahrkomfort zu erhöhen. Am Kraftfahrzeug angebrachte Kameras ermöglichen unter Verwendung geeigneter Bildverarbeitungsmethoden Verkehrszeichen wiederzuerkennen, die Fahrspur zu verfolgen oder den toten Winkel zu überwachen und beim Registrieren von Anomalien dem Fahrer eine Warnmeldung auszugeben. Noch einen Schritt weiter geht man bei der Entwicklung komplett autonomer Fahrzeuge, die mit Kameras und zahlreichen weiteren Sensoren ausgestattet sind und das Geschehen im Straßenverkehr erfassen und analysieren können. Bei einer solchen computergesteuerten Navigation eines Fahrzeuges soll das Verhalten eines menschlichen Fahrers nachgeahmt werden. So beschäftigt sich ein Teilgebiet der Informatik mit der Erforschung von künstlicher Intelligenz, dem Versuch der Entwicklung eines automatisierten, menschenähnlichen Intelligenzverhaltens zur Lösung von Problemen. Dabei können im Allgemeinen nur abgegrenzte Problemstellungen betrachtet werden.

Etwa 50 Jahre lang wurde das Schachspiel als ein zentrales Standardproblem der künstlichen Intelligenz angesehen. Die Idee bestand darin, ein Computerprogramm zu entwickeln, welches besser Schach spielen kann als der Mensch. Weil Schach ein Spiel mit „perfekter Information" ist, also keine unvorhersehbaren Ereignisse auftreten können, lassen sich Algorithmen entwickeln, die deterministisch und effizient den durch die Spielregeln vorgegebenen Spielbaum nach einem Folgezug mit einer möglichst hohen Gewinnwahrscheinlichkeit durchsuchen.

Aufgrund dieser mathematisch klar definierten Problemwelt ist Schach jedoch zur Erforschung der künstlichen Intelligenz ungeeignet. Denn im Gegensatz zu jenem substanzlosen Modell benötigt der Mensch zur Entfaltung seiner kognitiven Fähigkeiten einen Körper und eine Umgebung, mit welcher er interagieren kann. Die Wahrnehmung dieser Umgebung ist keinesfalls klar definiert, sondern oft unvorhersehbar und verrauscht in einer sich ständig verändernden Welt, auf die angemessen reagiert werden muss.

Als im Jahre 1997 der IBM Supercomputer "Deep Blue" [CHH02] den Schachweltmeister Garri Kasparow in einem Turnier besiegt hatte und damit der Höhepunkt des Computerschachs als Forschungsgebiet der künstlichen Intelligenz erreicht war, verlagerte sich der Fokus des wissenschaftlichen Interesses auf Roboterfußball als dynamisches Spiel in einer komplexen Umgebung. Ziel ist die Entwicklung einer vollkommen autonomen, humanoiden Robotermannschaft, die 2050 gegen die amtierenden menschlichen Weltmeister ein Fußballspiel gewinnen soll.

Die Gründung des Robocups, einer internationalen Forschungsinitiative, die sich mit der Entwicklung neuer Methoden und Verfahren für zukünftige Robotikanwendungen auseinandersetzt, eröffnet neue Perspektiven in verschiedenen Teilgebieten der künstlichen Intelligenz, wie beispielsweise dem maschinellen Sehen, Lernen und Planen.

Um Ergebnisse zu vergleichen, Lösungen auszutauschen und die Entwicklung für das gemeinsame Ziel voranzutreiben, treffen sich jährlich Teams aus der ganzen Welt zu den im Rahmen der Robocup-Wettbewerbe ausgetragenen Fußballturnieren, an denen sich auch das im Jahre 2009 gegründete Nao-Team [nao11] der HTWK Leipzig in der Standard-Plattform-Liga beteiligt.

1.1 Motivation

Die Grundvoraussetzung für einen kompetitiven Fußballroboter in der Standard-Plattform-Liga ist eine möglichst schnelle und genaue Wahrnehmung seiner Umgebung. Dafür stehen ihm verschiedene Sensoren zur Verfügung, wobei die visuelle Objekterkennung mittels integrierter Kameras eine zentrale Rolle spielt. Aus diesem Grunde stellt die echtzeitfähige Bildverarbeitung, d.h. die schnelle und zuverlässige Extraktion von Informationen über Objekte und Strukturen im Bildmaterial, einen wichtigen Forschungsbereich im Robocup dar.

In einer farbcodierten Welt, die auch in der Standard-Plattform-Liga im Robocup vorherrscht, liegt es nahe, die Objekterkennung auf Basis einer Farbsegmentierung vorzuneh-

men. Ein gängiger Standard sind daher so genannte Lookup- oder Farbtabellen, in denen gleichsam jeder mögliche Farbwert einem Objekt zugeordnet wird. Dies erfolgt in einem meist relativ zeitaufwendigen, oft manuellen Kalibrierungsprozess. Im späteren Spielverlauf kann mit dieser Tabelle das Kamerabild sehr schnell segmentiert werden, so dass bei konstanten Lichtverhältnissen eine relativ zuverlässige Objekterkennung möglich ist.

Jedoch werden die Anforderungen innerhalb des Wettbewerbes jährlich erhöht, mit dem Ziel, schrittweise auf die standardisierte Beleuchtung des Spielfeldes zu verzichten. Als Beispiel hierzu seien die German Open 2009 bis 2011 erwähnt, deutschlandweite Wettbewerbe der Robocup-Initiative, bei denen sich aufgrund der Sonneneinstrahlung durch große Deckenfenster die Lichtverhältnisse auf dem Spielfeld mehrfach änderten, so dass Bildsegmentierungsverfahren auf Basis von Farbtabellen in Folge der Helligkeitsunterschiede nicht mehr zuverlässig funktionierten. Der Wechsel von Kunstlicht zu Sonnenlicht verursachte des Weiteren bei den empfindlichen Kameras der Roboter teils starke Farbverschiebungen, so dass beispielsweise ein orangener Ball auf seiner Oberseite teilweise violett im Kamerabild erschien. Bei der Robocup-Weltmeisterschaft 2010 in Singapur war der störende Einfluss von Tageslicht zwar ausgeschlossen, jedoch existierten unterschiedlich farbige Bälle. Waren die Roboter beispielsweise nicht auf die Farbe des aktuellen Spielballs kalibriert, wurde dieser häufig nicht mehr als solcher erkannt. Jene Beispiele verdeutlichen, dass hier der Einsatz von Lookup-Tabellen aufgrund der starren Zuordnung von Objekten zu Farbwerten an seine Grenzen stößt.

1.2 Zielsetzung und Abgrenzung

Eine rein farbbasierte Objekterkennung scheint aus den genannten Gründen nicht ausreichend, so dass die Hinzunahme von Wissen über Objektformen angestrebt werden sollte. Ziel dieser Arbeit ist die Entwicklung von Bildverarbeitungsalgorithmen, die auf der Nutzung von Wissen über Objekteigenschaften und -formen basieren, die trotz der hohen Echtzeitanforderungen im kompetitiven Fußballspiel und trotz der eingeschränkten Ressourcen der mobilen Systeme robuste Objekterkennungsresultate liefern. Besondere Relevanz hat hierbei die Erkennungsleistung unter sich ändernden bzw. unbekannten Lichtverhältnissen, wobei im Gegensatz zu anderen Verfahren auf die Verwendung einer mitunter zeitaufwendigen Farbkalibrierung verzichtet wird.

Erkannt werden sollen das grüne Spielfeld in Verbindung mit den weißen Feldlinien, der rot-/orangefarbene Ball und die beiden unterschiedlich farbigen Tore. Zur Validierung der

Erkennungsleistung der Algorithmen und zur Optimierung von Parametern erfolgt die Prüfung der Ergebnisse durch eine Testbilddatenbank.

Nicht Bestandteil dieser Abhandlung ist die Weiterverarbeitung von 2D-Pixelkoordinaten der Objekte in 3D-Koordinaten und im Zusammenhang hiermit die Selbstlokalisierung des Roboters anhand der erkannten Linien oder Torpfosten.

1.3 Aufbau der Arbeit

Einführend werden im Kapitel 2 die unterschiedlichen Ligen des Robocup-Wettbewerbes vorgestellt und die in dieser Arbeit verwendete „NAO"-Standard-Plattform entsprechend eingeordnet sowie technische Details des Roboters und nützliche Informationen zur Standard-Plattform-Liga aufgeführt. Des Weiteren werden im Abschnitt 2.4.1 etablierte, insbesondere im Rahmen des Robocup verwendete Standardverfahren des maschinellen Sehens beschrieben und deren Vor- und Nachteile diskutiert.

Der Hauptteil vorliegender Abhandlung (siehe Kapitel 3) beinhaltet zunächst eine Beschreibung der erstellten Testbild-Datenbank (Abschnitt 3.1), die eine zentrale Rolle in Entwicklung der in dieser Arbeit angewandten Algorithmen einnimmt. Anschließend werden zur visuellen Objekterkennung nützliche robuste Eigenschaften definiert (Abschnitt 3.2), die in weiteren Abschnitten regelmäßig für die zu erkennenden Objekte evaluiert und anhand derer schließlich Kriterien für die zu entwickelnden Objekterkennungsverfahren festgelegt werden. Nach einer in Abschnitt 3.2 definierten Reihenfolge werden nun schrittweise objektspezifische Problemstellungen betrachtet und Algorithmen zu deren Lösung entwickelt (siehe Abschnitte 3.4, 3.5, 3.6 und 3.7).

Im Kapiel 4 werden zusammenfassend die Verarbeitungszeiten sowie die Erkennungs- bzw. Falsch-Positiv-Raten der vorgestellten Verfahren aufgelistet. Abschließend gibt das Kapitel 5 einen Ausblick für zukünftige Entwicklungen.

2 Grundlagen

Der Robocup ermöglicht die Erforschung der künstlichen Intelligenz parallel in unterschiedlichen Bereichen, wie Sensorik, Motorik, Selbstlokalisierung, sowie Strategieentwicklung, Handlungsplanung und weiteren. Neben dem Roboterfußball (Robocup-Soccer) gibt es weitere Kategorien, in denen die Teams mit ihren Forschungsarbeiten neue Zukunftsperspektiven auf dem Gebiet der künstlichen Intelligenz erschließen. Von großem Interesse ist beispielsweise die Entwicklung von leistungsfähigen Rettungsrobotern (Robocup-Rescue) für Such- und Bergungsaufgaben, für die effiziente Koordination von Einsatzkräften oder für Entscheidungsunterstützungssysteme, wie sie heute bereits genutzt werden. Relativ neu ist außerdem die Kategorie der Haushaltsroboter (RoboCup-@home), in der in einer nachgestellten Wohnumgebung demonstriert wird, wie die Roboter Personen erkennen, Kommandos verstehen oder Gegenstände finden und greifen können. Des Weiteren sei die Nachwuchsförderung durch den Schülerwettbewerb „Robocup-Junior" erwähnt.

2.1 Robocup-Ligen

Im Folgenden wird kurz auf die Forschungsschwerpunkte der verschiedenen Ligen innerhalb des Robocup-Soccer Wettbewerbes eingegangen und erläutert, welche Aufgaben und Probleme hierbei vornehmlich auf dem Gebiet der Bildverarbeitung zu lösen sind.

2D-Simulation In dieser Liga spielen auf Basis einer zweidimensionalen Simulationsumgebung (Abbildung 2.1a) zwei Mannschaften mit jeweils elf virtuellen Spielern, auch Agenten genannt, gegeneinander. Jeder dieser programmierten Agenten steht in Verbindung mit einem Soccer-Server und erhält hierdurch zum einen Informationen über seine Umgebung und sendet zum anderen Kommandos, wie beispielsweise auszuführende Bewegungsvektoren, an diesen Server zurück, die dieser folglich ausführt. Aufgrund der im Vergleich zu den anderen Ligen hohen Anzahl an Agenten konzentriert sich das Forschungsinteresse hier besonders auf den Bereich des kooperativen Verhaltens. Eine Bildverarbeitung zum Zweck

der Wahrnehmung von Objekten ist hierbei jedoch nicht erforderlich, sondern vielmehr werden verrauschte Positionsdaten auf direktem Weg vom Simulationsserver zu den Clients übermittelt.

3D-Simulation Ähnlich wie in der 2D-Simulationsliga spielen auch hier virtuelle Roboter in einer physikalisch simulierten Umgebung, wobei jedoch die dritte Dimension mit einbezogen wird, wie in Abbildung 2.1b dargestellt. Die Agenten kommunizieren ausschließlich mit einem Server, der sämtliche Sensordaten zur Verfügung stellt sowie Steuerbefehle entgegennimmt, überprüft und ausführt [Rob10d].

Da keine teure Robotikhardware benötigt wird, liegt in beiden Simulationsligen der Schwerpunkt ausschließlich auf der Softwareentwicklung. Ideen und Theorien zur Lösung von Standardproblemen wie maschinelles Lernen, Koordinierung oder Gegnermodellierung, können so zeit- und kosteneffizient erforscht und getestet werden. Aufgrund der Art der durch den Server zur Verfügung gestellten Sensordaten ist auch in dieser Liga keine Bildverarbeitung nötig.

(a) (b)

Abb. 2.1: Die Abbildung zeigt je einen Screenshot der GUI des Simulationsservers für die 2D-Simulationsliga (a) [lig11a] und die 3D-Simulationsliga (b) [lig11b].

Small-Size Es spielen zwei Teams mit je fünf auf Rädern fahrenden Robotern (siehe Abbildung 2.2a) auf einem 6×4 m^2 großem Spielfeld. Diese werden per Funk von einem zentralen Computer gesteuert. Dieser Server ist mit einer über dem Spielfeld montierten Kamera verbunden und kann anhand der farbigen Markierungen auf den Robotern deren Position ermitteln [Rob10a]. Die hierfür angewandte Bildverarbeitungssoftware muss neben den Spielern auch die Feldbegrenzungen sowie einen orangefarbenen Golfball, der als Fußball dient, erkennen.

Middle-Size In dieser Liga spielen fünf autonome Roboter je Mannschaft auf einem 12×18 m^2 großem Spielfeld. Sie werden von den teilnehmenden Forschungsteams selbst entwickelt und unterscheiden sich daher hinsichtlich Aufbau, Antriebsart und Sensorik, sind jedoch im Vergleich zu den Robotern der Small-Size-Liga mit bis zu 80cm Höhe wesentlich größer und mit bis zu 40 kg wesentlich schwerer (siehe Abbildung 2.2b). Zur Umgebungswahrnehmung dürfen in die Roboter verschiedene Sensoren, wie zum Beispiel Kameras, Lage- und Odometriesensoren eingebaut werden. Zur Kommunikation sind die Roboter eines Teams über WLAN miteinander vernetzt. Sie agieren autonom, d.h. der Programmablauf darf nicht durch externe Einflüsse modifiziert werden. Aufgrund der Größe und des hohen Gewichtes lassen sich Notebooks in die Roboter verbauen, so dass auch komplexe Berechnungen, besonders im Bereich der Bildverarbeitung, durchgeführt werden können. Sehr beliebt sind omnidirektionale Kameras (siehe Abbildung 2.2c), die mit einer Aufnahme nahezu jeden Bereich des Spielfeldes visuell erfassen, so dass die Roboter auf dem Feld sehr robust ihre Position ermitteln können. Eine Herausforderung für Bildverarbeitungssysteme besteht darin, dass die Tore nicht farbig markiert, sondern wie auf einem realen Fußballfeld weiß gefärbt sind [Rob10b] und dass zukünftig auch die Farbe des Fußballs nicht mehr als orange definiert ist [Rob11]. Die Weiterentwicklung der visuellen Objekterkennung zu einer robusten Funktionsweise auch unter natürlichen und wechselnden Licht- und Farbverhältnissen ist eine der Forschungsinteressen in dieser Liga. Neben der Bildverarbeitung ist auch die Entwicklung adaptiver, lernender und kooperativer Verhaltensweisen relevant, da die Leistungsfähigkeit einer Robotermannschaft grundlegend von der Fähigkeit, im Team zu interagieren, abhängt.

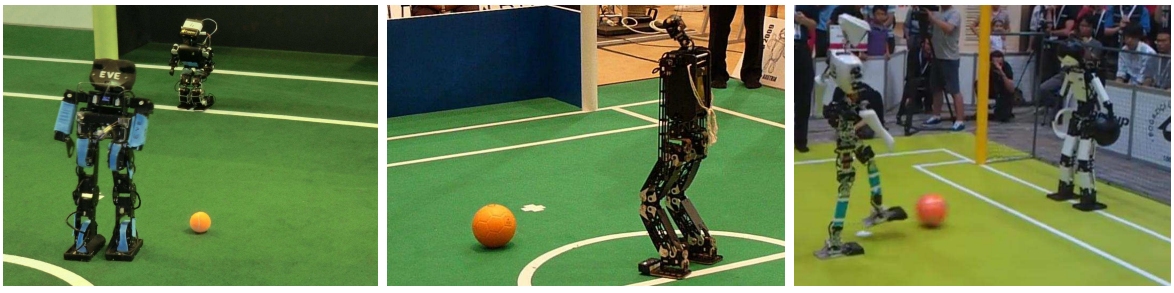
(a) (b) (c)

Abb. 2.2: In Abbildung (a) sind die kleinen Roboter der Small-Size-Liga und in (b) die der Middle-Size-Liga dargestellt. Bild c zeigt eine Nahaufnahme einer omnidirektionalen Kamera eines Middle-Size-Roboters.

Humanoid Anders als in den bisher vorgestellten Ligen werden hier Roboter menschenähnlicher Gestalt mit aufrechtem bipedem Gang konstruiert, die auf einem 6×4 m großen Feld in Dreierteams antreten. Unterschieden wird die Kidsize-Liga mit Robotern bis zu einer Höhe von 60 cm (Abbildung 2.3a), die Teensize-Liga mit 100-120 cm (Abbildung 2.3b) und die Adultsize-Liga ab einer Höhe von 130cm (Abbildung 2.3c). Sie agieren autonom, dürfen jedoch - ähnlich wie in der Middle-Size-Liga - untereinander sowie mit einem Schiedsrichtercomputer über WLAN kommunizieren.

Da die bipede Fortbewegung jedoch Risiken des Gleichgewichtsverlustes birgt, ist eines der grundlegenden Forschungsziele deren komplikationsloser Ablauf, zu dessen Realsierung Beschleunigungs- und Winkelsensoren eingesetzt werden. Kommt es dennoch zum Gleichgewichtsverlust, ist die selbstständige Wiederaufrichtung des Roboters Voraussetzung für eine Weiterführung der aktiven Teilnahme am Spielgeschehen.

Zur Orientierung auf dem Spielfeld dienen den Robotern farbige Markierungen [Rob10c]: die Diskrimination zwischen eigenem und gegnerischem Tor erfolgt anhand eines blauen bzw. gelben Farbtons. Des Weiteren fungieren zwei farbige Säulen an den Rändern der Mittellinie als Landmarken. Auch die Extremitäten der Roboter sind entweder magenta- oder cyanfarben markiert, was eine Unterscheidung der antretenden Teams ermöglicht. Gespielt wird mit einem orange gefärbten Standard-Tennisball. Auch in dieser Liga liegt aufgrund der Farbcodierung der Spielfeldumgebung eine kamerabasierte Objekterkennung nahe. Oft kommen dabei Objektive mit einem relativ großem Öffnungswinkel von bis zu ca. 180 Grad zum Einsatz, so dass im Kamerabild möglichst viele Informationen des Spielgeschehens erfasst werden.



(a) (b) (c)

Abb. 2.3: Die drei unterschiedlichen Klassen der Humanoid-Liga

Standard Plattform Liga In den bisher vorgestellten Ligen zielten die Forschungsschwerpunkte entweder auf die reine Softwareentwicklung in einer simulierten Umgebung oder auf eine Kombination aus Software- und Hardwareentwicklung an realen Robotern ab. Jedoch

wird die Konstruktion leistungsfähiger Roboterhardware durch die jeweils zur Verfügung stehenden finanziellen Mittel unterschiedlich stark limitiert, so dass kein „fairer" Vergleich von Programmierlösungen möglich ist. So wird beispielsweise die Chance eines Programms, einen Spielsieg zu erwirken, um so geringer, je langsamer und instabiler sich der Roboter - etwa aufgrund leistungsschwacher Servomotoren - fortzubewegen vermag.

Unter anderem aus diesem Grund wurde die Standard-Plattform-Liga eingeführt, in der alle Teams die gleichen Roboter nutzen und sich somit ausschließlich auf die Erforschung neuer Ideen und Ansätze im Softwarebereich konzentrieren können. Durch gleiche Ausgangsbedingungen begünstigt werden insbesondere der Vergleich von Entwicklungsarbeiten, der schnelle Einstieg für neue Teams in diese Liga, der Austausch zwischen den Teams und schließlich auch eine Einsparung von Entwicklungskosten. Zu den wichtigsten zu untersuchenden Gesichtspunkten gehören die dynamische bipede Fortbewegung, die visuelle Wahrnehmung der Spielsituation, die Selbstlokalisierung innerhalb des Spielfeldes, die Ballmanipulation und die Koordination des Mannschaftsspiels.

Viele Jahre diente in der früher auch als „Four-Legged" bezeichneten Liga der vierbeinige Roboterhund Aibo (Abbildung 2.4a) als Standardroboter. Aufgrund seiner Produktionseinstellung im Jahre 2008 wurde dieser durch den humanoiden Roboter „NAO" (Abbildung 2.4b) der Firma Aldebaran-Robotics ersetzt, ein wichtiger Schritt für die Entwicklung des humanoiden Roboterfußballs [Ald11].

Da die Entwicklung der in dieser Arbeit vorgestellten Algorithmen auf einem Roboter in dieser Liga erfolgte, werden in folgendem Abschnitt einige Details über das Spielfeld, die Spielregeln und den verwendeten Roboter vorgestellt.

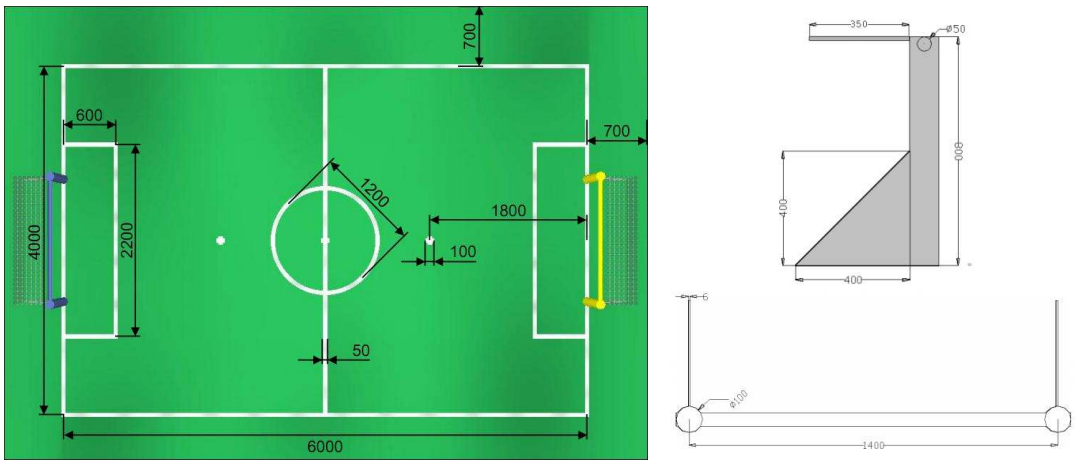
(a) (b)

Abb. 2.4: Der bis einschließlich 2007 als Standardplattform eingesetzte Roboter namens „Aibo" (a) wurde ein Jahr später durch den humanoiden Roboter „NAO" (b) ersetzt.

2.2 Standard-Plattform-Liga

In dieser Liga spielen jeweils vier dieser humanoiden Roboter auf einem 6×4 m^2 großen Feld. Das Wissen über den Spielfeldaufbau, insbesondere über die verwendeten Farben, ist Grundlage der späteren Betrachtungen in dieser Arbeit und wird in folgendem Abschnitt genauer beschrieben.

2.2.1 Spielfeldumgebung



(a) (b)

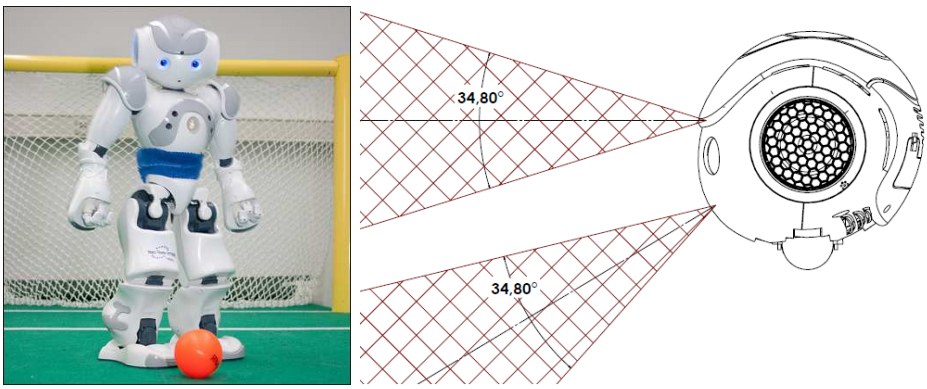
Abb. 2.5: Maße des Spielfelds der Standard-Plattform-Liga (Distanzen in mm).

Als Spielfeldboden dient - wie auch in anderen Ligen - ein grüner Teppich, dessen Farbe von hinreichender Helligkeit sein sollte und auf dem weiße, fünf Zentimeter breite Spielfeldlinien die jeweiligen Areale voneinander abgrenzen. Die Außengrenzen des Feldbodens liegen laut Regelwerk allseits genau 70cm jenseits der äußeren Spielfeldlinien (siehe Abbildung 2.5a). Dies hat besondere Relevanz für die in Abschnitt 3.5 beschriebene Linienerkennung. Zugunsten der Annäherung an realitätsgetreue Spielbedingungen wurde in der Standard-Plattform-Liga auf seitlich der Mittellinie positionierte Landmarken verzichtet. Die Tore bestehen aus zwei 80 cm hohen zylindrischen Pfosten mit einem Durchmesser von 10 cm, die durch einen 140 cm langen, gleichfarbigen Querbalken miteinander verbunden sind. Die Tore der jeweiligen Spielfeldseite werden anhand der Farben „himmelblau" und „gelb" unterschieden. Zur Stabilisierung der Pfosten dient je ein rechtwinkliges, weißes Holzdreieck, das an der Rückseite montiert ist (siehe Abbildung 2.5b). Gespielt wird mit einem 6,5 cm großen, rotfarbenen Ball. Von Bedeutung für die in Abschnitt 3.6 beschriebene

visuelle Ballerkennung ist dessen glänzende Oberfläche, die durch daraus resultierende Lichtreflexion die Objekterkennung erschweren kann. Die Roboter können voneinander durch unterschiedlich farbige Rumpf-Bänder unterschieden werden. So tragen die Roboter des „blauen Teams" ein blaufarbenes, die des „roten Teams" ein magentafarbenes Band [Rob10e].

2.2.2 Standardplattform „NAO"

„NAO" (siehe Abbildung 2.6a) ist weltweit der am häufigsten für akademische Zwecke genutzte humanoide Roboter. Er ist 58 cm groß, wiegt ca. 4,3 kg und hat 21 Freiheitsgrade in der „Robocup Edition" bzw. 25 Freiheitsgrade in der „Academics Edition" [Ald11]. Zur Informationsverarbeitung ist im Kopfteil des Roboters ein AMD Geode Prozessor mit 500 MHz und 512 MB Arbeitsspeicher integriert. Ein USB-Stick dient als Speichermedium und bietet ausreichend Speicherplatz für ein „echtzeit gepatchtes" Linux-Betriebssystem. Ein WLAN-Stick ermöglicht die Kommunikation mit der Außenwelt, was insbesondere in Fußballturnieren zur Teamkommunikation sowie zur Reaktion auf einen Schiedsrichtercomputer Grundvoraussetzung ist. Zur Interaktion mit der Umgebung stehen eine Reihe von Sensoren, wie zum Beispiel Abstandssensoren, Mikrofone und verschiedene Buttons zur Verfügung. Beschleunigungs- und Winkelsensoren dienen der Messung der Oberkörperneigung und Drucksensoren an den Füßen der Schwerpunktberechnung. Zwei mittig in der Frontseite des Kopfteils installierte Farbkameras gem. Abbildung 2.6b ermöglichen die Verarbeitung visueller Informationen bis zu einer Auflösung von 640x480 Pixeln bei 30 Bildern pro Sekunde und liefern die Ausgangsdaten für die in dieser Arbeit entwickelten Algorithmen.



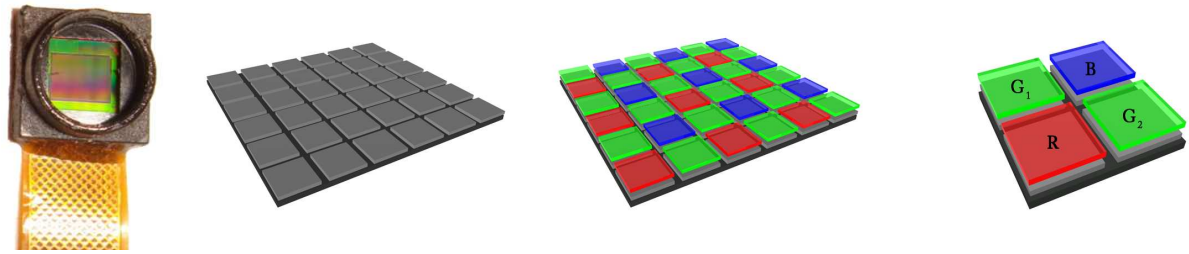
(a) (b)

Abb. 2.6: Dem humanoiden Roboter NAO (a) stehen zur Umgebungswahrnehmung zwei im Kopfteil integrierte Farbkameras (b) [AR11] zur Verfügung.

2.3 Pixeldaten

2.3.1 CMOS-Bildsensoren

Ein Bildsensor besteht üblicherweise aus einer Matrix von Halbleiterdetektoren (Complementary Metal Oxide Semiconductors - CMOS), sogenannte „Active Pixel Sensoren" (Schematisch in Abbildung 2.7b dargestellt). Diese messen die Menge an Licht, die in einer bestimmten Zeit auf ihre Fläche trifft und erzeugen eine Spannung, die proportional zur gemessenen Lichtintensität ist. Diese Spannung kann nun durch einen Analog-Digital-Wandler in einen Binärwert konvertiert und zur digitalen Weiterverarbeitung genutzt werden. Zur Gewinnung von Farbinformationen dient ein Farbfilter, so dass jeweils nur definierte Anteile des Lichtes auf bestimmte Detektoroberflächen auftreffen. Häufig findet hierfür eine Bayerfiltermatrix Anwendung, die für jeden einzelnen Detektor entweder rotes, grünes oder blaues Licht gemäß Abbildung 2.7c passieren lässt. Weil das menschliche Auge besonders gut Grüntöne differenzieren kann, werden grüne Elemente doppelt so häufig wie blaue oder rote verwendet, um durch Mittelung von G_1 und G_2 einen hoffentlich weniger verrauschten Grünwert $G := \frac{G_1+G_2}{2}$ zu erhalten. Zusammen mit den übrigen zwei Messwerten erhält man so eine RGB-Farbinformation für einen Pixel des Kamerabildes (siehe Abbildung 2.7d).



(a) (b) (c) (d)

Abb. 2.7: Die Abbildungen zeigen einen Active-Pixel-Sensor (a), die Anordnung der Detektoren in einer Matrix (b), eine Bayer-Filtermatrix (c) und ein aus vier Detektoren bestehendes Pixel (d).

2.3.2 RGB-Farbraum

Der RGB-Farbraum beschreibt Farben durch die additive Mischung der drei Komponenten Rot (700 nm), Grün (546,1 nm) und Blau (435,8 nm) [Str05]. Sie spannen einen dreidimensionalen Farbraum auf, der als Würfel wie in Abbildung 2.8 dargestellt werden kann. In der Regel werden 8 Bit für einen Farbkanal (256 verschiedene Intensitätsstufen) verwendet, so dass die Farbe eines Pixels durch $3 * 8 = 24$ Bit dargestellt werden kann.

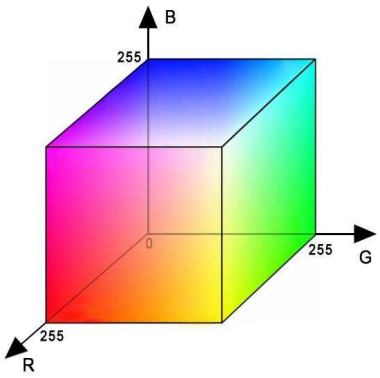


Abb. 2.8: RGB-Würfel für 256 verschiedene Intensitätsstufen eines jeden Farbkanals

2.3.3 Von RGB nach YCbCr

Die im NAO verbaute Kamera nutzt zur Abbildung von Farbinformationen der Pixelrohdaten statt der RGB-Farben das YCbCr-Farbmodell [1], einen gebräuchlichen Standard für die digitale Bild- und Videoaufzeichnung.

Ein Bildpunkt wird im YCbCr-Modell durch seine Grundhelligkeit (Luma) und seine Farbigkeit (Chroma) beschrieben. Durch eine konvexe Linearkombination der R, G und B Komponenten, die sich an der farbabhängigen Helligkeitsempfindung des menschlichen Auges orientiert, führt die Formel

$$Y := 0,299 * R + 0,587 * G + 0,114 * B$$

zur ersten Komponente des YCbCr-Farbmodells. Die beiden übrigen Komponenten sind Maße für die Abweichung der Farbigkeit von einem neutralen Grau, wobei Cb die Abwei-

[1] Häufig wird das YCbCr-Farbmodell mit dem YUV-Farbmodell für das analoge Farbfernsehen verwechselt. Trotz der anscheinenden Ähnlichkeit dieser sei in [Poy03b] weiterführend auf die korrekte Unterscheidung hingewiesen .

chung in Richtung Blau/Gelb und Cr in Richtung Rot/Türkis darstellen. Sie werden durch die Formel

$$Cb := 128 - 0,168736 * R - 0,331264 * G + 0,5 * B$$

und

$$Cr := 128 + 0,5 * R - 0,418688 * G - 0,081312 * B$$

berechnet. Die Cb- und Cr-Komponente spannen gemeinsam eine Ebene auf, die in Abbildung 2.9 für drei verschiedene Y-Werte beispielhaft dargestellt wird.

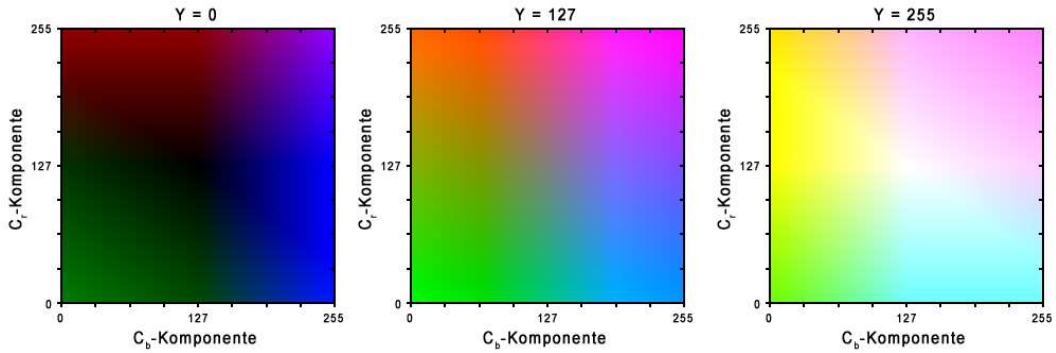


Abb. 2.9: Cb-Cr-Farbebenen für drei unterschiedliche Helligkeitsstufen

Die Trennung der Grundhelligkeit von der Farbigkeit im YCbCr-Farbmodell bietet einige Vorteile gegenüber dem RGB-Farbraum: so können die Cb- und Cr-Kanäle unterabgetastet werden, ohne dass dies mit einem für den Menschen erkennbaren Verlust der Bildqualität einhergeht. Dies ist hauptsächlich begründet durch die Anpassung des menschlichen Sehsinns an Farb- und Helligkeitsverteilungen in der Natur: viele Informationen, wie Schattierungen und Reflexionen, führen hauptsächlich zu einer Helligkeitsänderung ohne wesentliche Farbtonabweichung. Mit der Unterabtastung erreicht man auf einfachen Wege, fast ohne merklichen Qualitätsverlust eine Komprimierung des Bildes, so dass dieses schneller übertragen werden kann (Chroma-Sub-Sampling [Poy03a]).

Die Verwendung dieses Farbmodells bietet zudem für Anwendungen im Robocup, speziell für die Standard-Plattform-Liga, einen weiteren Vorteil: Da nach dem Regelwerk [Rob10e] den Objekten dieser Liga definierte Farbbereiche gem. Abbildung 2.10 zugeordnet sind, lassen sich diese teils allein durch die Cb- bzw. Cr-Komponente unterscheiden. Darüber hinaus können einzig anhand der Cr-Komponente Ball- und Spielfeldfarbe voneinander

abgegrenzt werden, so dass sich insbesondere während der visuellen Verfolgung des Spielballs Rechenzeit in der Bildverarbeitung einsparen lässt. Ebenso verhält es sich mit der Cb-Komponente, anhand der die beiden Torfarben (gelb und blau) voneinander unterscheidbar sind.

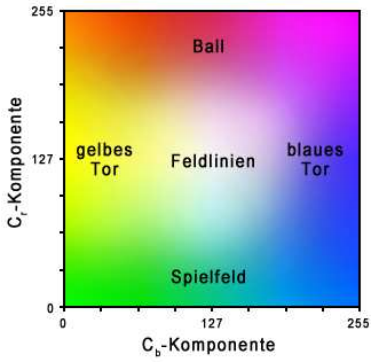


Abb. 2.10: Gezeigt wird eine grobe Einteilung von (Cb, Cr)-Tupeln zu Objekten der Robocup-Umgebung. Die entsprechenden Y-Werte wurden für diese Grafik zur besseren Erkennung der Farben auf diese abgestimmt.

Neben dem $YCbCr$-Farbmodell sind für die Bildverarbeitung auch andere Modelle, wie zum Beispiel das HSI- oder Lab-Modell, geeignet [WWW99]. Dennoch überwiegt ein weiterer Vorteil in Bezug auf die Verarbeitungsgeschwindigkeit: da die Kameras des Roboters bereits $YCbCr$-Farbwerte als Rohdaten liefern, wird keine zusätzliche Rechenzeit für die Konvertierung in ein anderes Farbmodell benötigt. Messungen ergaben beispielsweise eine Vervierfachung der Verarbeitungszeiten unter Verwendung eines sogenannten HSI-Farbmodells gegenüber dem „YUV422"-Format ($YCbCr$-Farbmodell mit Chroma-Sub-Sampling) auf einer NAO-Plattform [Sán09].

2.4 Standardverfahren zur Bildverarbeitung im Robocup

Im Folgenden wird ein Überblick über einige etablierte Bildverarbeitungsverfahren gegeben, die Anwendung in der Segmentierung und Objekterkennung - insbesondere im Bereich Roboterfußball - finden.

2.4.1 Farb- und Objektzuordnung

Häufig erfolgt als erster Verarbeitungsschritt in der Farb- und Objektzuordnung eine farbliche Klassifizierung der Pixel des Kamerabildes. Voraussetzung hierbei ist eine bestehende

Zuordnung von Farben zu Objekten. Dieser Schritt kann durch die Verwendung einer Farb-Lookup-Tabelle, in der einmalig im Voraus zu jeder möglichen Farbe eine zugehörige Klassifikation definiert wird, effizient realisiert werden. Jener Zuordnungsvorgang wird auch als Farbkalibrierung bezeichnet und kann beispielsweise vor einem Roboter-Fußballturnier in Form der Aufnahme und Analyse mehrerer Testbilder aus unterschiedlichen Positionen stattfinden.

Die einzelnen Farbwerte werden hierbei als Indizes der Farbtabelle interpretiert: ist eine Farbe aus den drei 8-Bit Komponenten Y, Cb und Cr gegeben, so kann durch Anwendung der Formel 2.1 ein 24-Bit-Tabellenindex errechnet und die Farb-Objekt-Zuordnung direkt in die Tabelle gespeichert oder ausgelesen werden:

$$index_{LUT} := (Y << 16) + (Cb << 8) + Cr \tag{2.1}$$

Gewöhnlich handelt es sich bei den Elementen innerhalb der Farbtabelle um Bytes, so dass für eine gegebene Farbe bis zu 256 potenzielle Objekt-Klassen existieren. Für die Anwendung im Robocup werden üblicherweise die in Tabelle 2.1 aufgezählten Klassen unterschieden.

Objekt
Spielfeldboden
Ball
Linien
gelbes Tor
blaues Tor
magenta farbener Teammarker
blau farbener Teammarker
Hintergrund

Tab. 2.1: Typische Farbklassen im Robocup

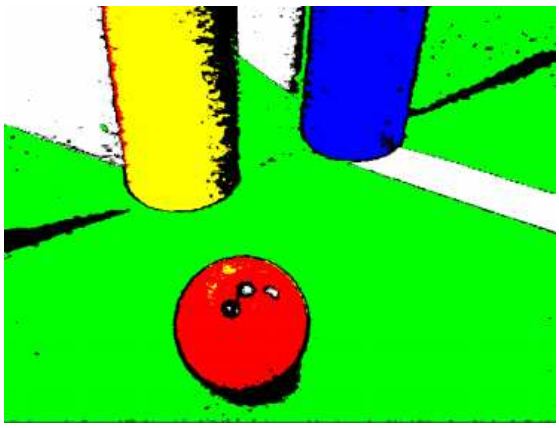
Der Speicherbedarf einer aus diesen Elementen erstellbaren Tabelle betrüge $256^3 Bytes = 16 MBytes$. In Realität kann jedoch deren Größe durch Quantisierung des Farbraumes reduziert werden [Bau08].

Im Robocup-Wettbewerb haben sich Farbtabellen als Standard etabliert, nicht zuletzt aufgrund der relativ leichten Implementierbarkeit und der hohen Geschwindigkeit der Klassifizierung eines Pixels, bei der lediglich zwei Additionen und zwei Shifts zur Berechnung des Tabellenindexes nötig sind (siehe Formel 2.1). Nachteile sind jedoch die

verhältnismäßig hohe Speicherplatzbelegung und der zeitliche Aufwand der im Voraus nötigen Farbkalibrierung.

In Abbildung 2.11 wird beispielhaft eine Objektzuordnung der Pixel vorgenommen, die verdeutlicht, dass Überlappungen der Farben für unterschiedliche Objekte auftreten können. Die Lichtreflexionen auf der Balloberfläche werden beispielsweise aufgrund der Ähnlichkeit zur Linienfarbe falsch klassifiziert.

(a)



(b)

Objekt	Farbe
Spielfeldboden	
Ball	
Linien	
gelbes Tor	
blaues Tor	
Hintergrund	

(c)

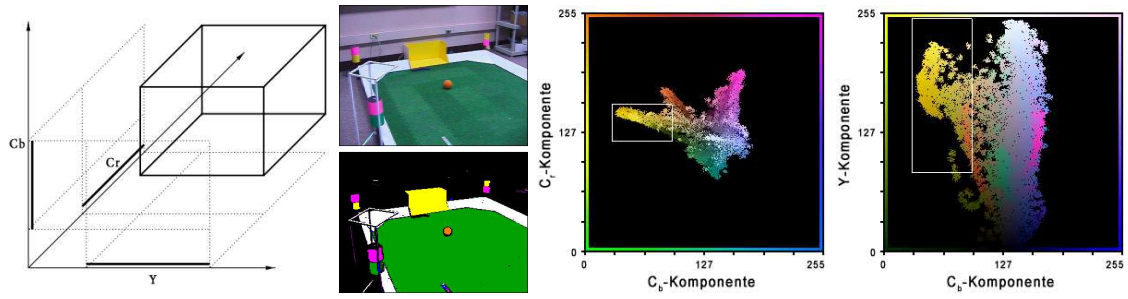
Abb. 2.11: Die Pixel eines Kamerabildes (a) werden durch Anwendung eines Klassifikators mittels einer Farb-Lookup-Tabelle verschiedenen Objekten zugeordnet (b) [FFM10]. Durch Überschneidungen der Farben unterschiedlicher Objekte kann es dabei zu Fehlklassifizierungen kommen. Der Zusammenhang zwischen Objekten und Klassenfarben ist in Tabelle (c) definiert.

Ein Speicherplatz-effizienteres Verfahren wird in [BBV00] vorgestellt. Dabei wird eine Zuordnung von $YCbCr$-Farben zu einem Objekt durch sechs Schwellenwerte gemäß dem nachfolgenden Algorithmus 2.1 realisiert.

Algorithm 2.1 Zugehörigkeit zu einer Farbklasse color_class überprüfen

```
if ((Y ≥ Y_lowerthresh)
&& (Y ≤ Y_upperthresh)
&& (Cb ≥ Cb_lowerthresh)
&& (Cb ≤ Cb_upperthresh)
&& (Cr ≥ Cr_lowerthresh)
&& (Cr ≤ Cr_upperthresh))
        pixel_color := color_class;
```

Sinngemäß beschränken die Schwellenwerte den Farbbereich, der einem bestimmten Objekt zugeordnet werden soll, auf einen Quader im $YCbCr$-Raum, wie in Abbildung 2.12 dargestellt. Das Verfahren scheint zunächst im Vergleich zu einer Farbtabelle rechenintensiver zu sein, denn zur Klassifizierung eines Pixels bei beispielsweise acht möglichen Objektklassen wären bis zu 48 Vergleiche durchzuführen, da der Algorithmus 2.1 für jede Objektklasse erneut ausgeführt werden müsste. Jedoch kann der Rechenaufwand, wie in [BBV00] beschrieben, für bis zu 32 verschiedene Objekte durch geschickte Verwendung von drei Lookup-Tabellen mit einer Gesamtgröße von lediglich 3 kB auf nur zwei logische „AND" Operationen reduziert werden. Nachteile hierbei sind sowohl die fest vorgegebene Quaderform von Farbbereichen als auch die Inflexibilität dieses auf statischen Schwellenwerten basierenden Verfahrens bei Veränderungen der Lichtverhältnisse.



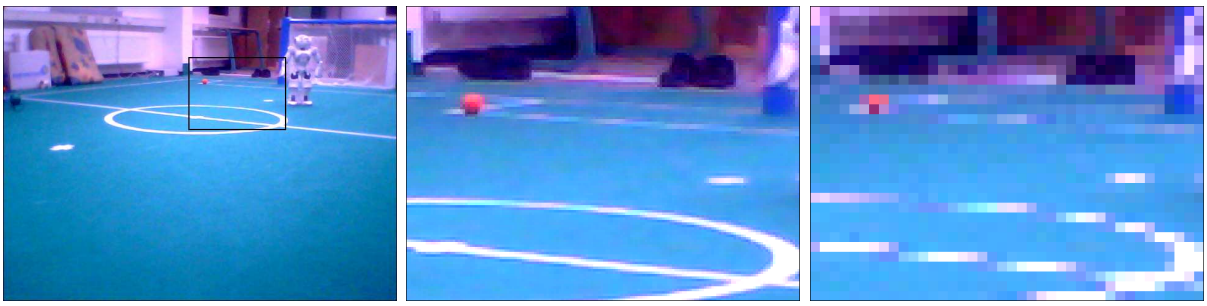
(a) (b) (c) (d)

Abb. 2.12: Durch sechs Schwellenwerte wird der Farbraum für ein bestimmtes Objekt auf einen Quader (a) beschränkt. Abbildung (c) und (d) stellt die Lage der Farben aller Pixel des Beispielbildes (b) und die Lage des Quaders für das Objekt „gelbes Tor" dar.

2.4.2 Effizientes Sampling

Für eine echtzeitfähige Verarbeitung des Kamerabildes ist eine hinreichend zeiteffiziente Objektklassenzuordnung der Pixel nötig, jedoch reicht womöglich die Rechenleistung der NAO-Plattform zur vollständigen Analyse eines Kamerabild mit 640x480 Pixeln nicht aus. Dieser Problematik wird mit diversen Methoden begegnet, in denen zum Zweck der Einsparung von Rechenzeit lediglich ein ausgewählter Teil der Bildpunkte analysiert wird.

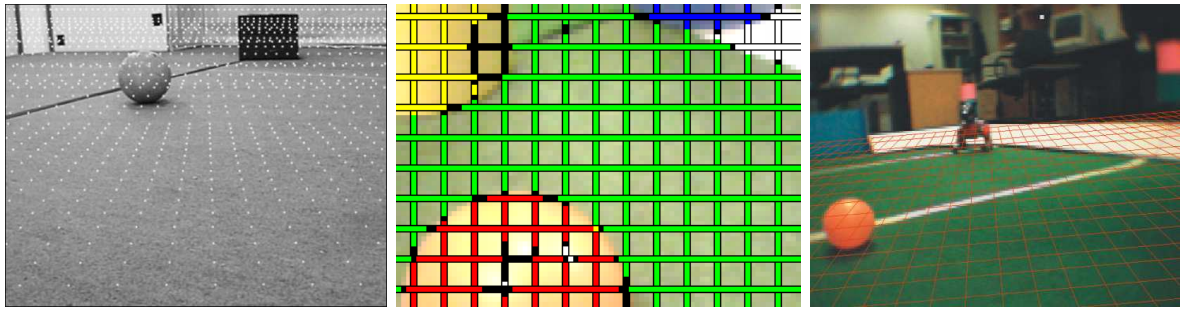
Ein einfaches Verfahren, in dem beispielsweise nur 25% bzw. 6% der Pixel klassifiziert werden brauchen, ist die Unterabtastung des Bildes mit 320x240 bzw. 160x120 Pixeln, was jedoch mit einem erheblichen Informationsverlust, wie in Abbildung 2.13 zu sehen, einhergeht.



(a) (b) (c)

Abb. 2.13: Zum Vergleich der Bildqualität bei unterschiedlichen Auflösungsstufen wurde der in (a) markierte Auschnitt ohne Unterabtastung (b) und mit Unterabtastung (c) dargestellt.

Da der hiermit verbundene Qualitätsverlust besonders für kleine, weit entfernte Objekte verhältnismäßig große Auswirkungen haben kann, wird in [JSM+02] eine Weiterentwicklung vorgestellt, die unter Beachtung der perspektivischen Verzerrung des Spielfeldes ein flexibles Punktraster zur Unterabtastung des Bildes nutzt. Dabei liegen gemäß Abbildung 2.14a Abtastpunkte für nahe Objekte weiter auseinander, für entfernte Objekte jedoch dichter beieinander, so dass eine relativ ausgeglichene Überdeckung der perspektivisch entzerrten Spielfeldebene durch die Abtastpunkte resultiert. Die Höhe und Neigung der Kamera müssen für dieses Verfahren im Vorfeld bekannt sein und können beispielsweise anhand von Wissen über die Kinematik, die Gelenkpositionen und die Lagesensoren des Roboters errechnet werden [JSM+02, BJ02].



(a) (b) (c)

Abb. 2.14: Durch ein flexibles Punktraster (a), ein starres Scanlinegitter (b) oder ein flexibles Scanlinegitter (c) kann eine effiziente Unterabtastung zur Reduktion des Berechnungsaufwandes erreicht werden.

Ein anderer, häufig angewandter Ansatz ist der Einsatz sogenannter Scanlines [ÁMO05], mittels derer lediglich einige Spalten und/oder Zeilen, wie in Abbildung 2.14b verdeutlicht, im Bild klassifiziert werden. Für jedes Pixel auf einer derartigen horizontalen oder vertikalen Scanline ist zur Objektzuordnung eines der im Abschnitt 2.4.1 vorgestellten Klassifikationsverfahren üblich. Nach der Objektzuordnung jedes der Pixel entlang der Scanlines können die auf den Grenzen benachbarter Objektklassen liegenden Bildpunkte bestimmt werden. In Abbildung 2.14b kann beispielsweise der Umriss des Balls durch Erkennen des Übergang zwischen „Ball"- und „Feldboden"-Klassen ermittelt werden. Die Scanline-Methode ist folglich besonders zur Erkennung von Konturen geeignet und bietet im Vergleich zur oben genannten Methode der gleichverteilten Unterabtastung des Bildes eine höhere Präzision, da entlang der Scanline keine Auflösungsreduktion erfolgt. Dennoch können kleine Objekte, die in den nicht erfassten Freiräumen zwischen den Scanlines liegen, „übersehen" werden. Dieser Problematik kann mit der Anwendung eines flexiblen Scanlinegitters begegnet werden. Analog zur zuvor geschilderten Methode der Unterabtastung durch ein flexibles Punktraster wird hierbei eine Projektion der Scanlines in die Spielfeldebene vollzogen, so dass Objekte in der Ferne aufgrund der dort bestehenden höheren Anzahl an Abtastpunkten detaillierter im Bild analysiert werden. In Abbildung 2.14c ist ein diesbezüglicher Ansatz der Humboldt Universität [BJ02] zur Verdeutlichung dargestellt.

Es sei auch auf weitere Methoden zur Anordnung von Scanlines [Nor05, JHL04, Jün04] und von Abtastpunkten [Fas09] hingewiesen, auf die jedoch in dieser Abhandlung nicht näher eingegangen wird.

3 Robuste Objekterkennung

Zunächst soll der im Folgenden verwendete Begriff der „Robustheit" im Rahmen dieser Arbeit definiert werden: „Robustheit ist die Fähigkeit eines Systems, seine Funktion auch bei Schwankung der Umgebungsbedingungen aufrecht zu erhalten, ohne dass Anpassungen am System nötig sind." [rob05]

Für die zugrunde liegende Aufgabenstellung der Objekterkennung in digitalem Bildmaterial mittels Bildverarbeitungsalgorithmen liegt der Fokus dementsprechend auf deren Unempfindlichkeit gegenüber globalen Helligkeits- oder Farbveränderungen der zu erfassenden Umgebung.

3.1 Datengrundlage

Zur Bewertung von Bildverarbeitungsalgorithmen hinsichtlich ihrer Erkennungsleistung und Güte dient als Datengrundlage eine für diese Arbeit erstellte Testbilddatenbank [Rei11]. Sie umfasst $n_B := 600$ Einzelbilder, welche mit der Kamera des in Abschnitt 2.2.2 vorgestellten Roboters in typischen Anwendungsszenarien - hauptsächlich während der German Open 2009, 2010 und der WM in Singapur 2010 - aufgenommen wurden. Die Bilder umfassen typische Standardsituation, wie sie in Wettkampfspielen vorkommen können, sowie ausgewählte Spezial- und Extremfälle. Die in dieser Arbeit beschriebenen Algorithmen zur Objekterkennung werden anhand dieser Teilmenge an Spielsituationen optimiert mit der Intention, dass sie auch für neue Spielszenen korrekte Ergebnisse liefern. Um eine möglichst hohe Anzahl an unterschiedlichen Ausgangssituationen zu erfassen, wurden die Testszenen aus diversen Positionen, Neigungswinkeln und Lichtverhältnissen aufgenommen. Die Häufigkeitsverteilung für bestimmte Objekte und Szenen innerhalb der entstandenen Datenbank ist der folgenden Tabelle zu entnehmen.

Sichtbare Objekte	Häufigkeit
Spielball (insgesamt)	40,2 %
Spielball (partiell verdeckt)	7,8 %
Blaues Tor (beide Pfosten)	13,3 %
Blaues Tor (nur ein Pfosten)	18,2 %
Gelbes Tor (beide Pfosten)	11,3 %
Gelbes Tor (nur ein Pfosten)	21,2 %
mindestens eine Linie sichtbar	89,0 %
kein Spielfeldboden sichtbar	0,3 %

Tab. 3.1: Häufigkeitsverteilung der Objekte im Datenbestand

Für jedes der Bilder wurden die in Tabelle 3.2 aufgelisteten Ground-Truth-Daten manuell erstellt, die sich vielfältig - beispielsweise zur automatisierten Qualitätsprüfung der entwickelten Algorithmen oder zur Generierung von Statistiken - verwenden lassen. Zunächst dienen sie jedoch als Hilfestellung zur Ermittlung robuster Objekteigenschaften.

Objekt	Form der Daten
Feldbegrenzung	Approximation durch zwei Geraden (siehe Abschnitt 3.5.2)
Feldfarbe	$(\bar{Y}, \bar{Cb}, \bar{Cr})$-Tripel und Information über Sichtbarkeit
Linien	mehrere Listen von auf Linienkanten liegenden Punkten
Spielball	Koordinate des Mittelpunktes, Radius und Information über Verdeckung
Tore	Koordinaten und Breite der Torpfosten-Fußpunkte

Tab. 3.2: Gezeigt werden die in der Datenbank abgespeicherten Informationen für Objekte im Testbildmaterial. Das Tripel $(\bar{Y}, \bar{Cb}, \bar{Cr})$ sei definiert als das arithmetische Mittel der Farbwerte aller feldfarbenen Pixel in einem Bild.

3.2 Robuste Eigenschaften und Erkennungsmerkmale

In den folgenden Beispielen werden potenzielle Auswirkungen von Helligkeits- oder Farbveränderungen auf die Objektdifferenzierung erläutert. Zur Vereinfachung werden hierbei ausschließlich die Pixel einer einzigen Zeile des Beispielbildes in Abbildung 3.1a betrachtet. Für alle Pixel entlang dieser Zeile sind in den Abbildungen 3.1b-d die jeweiligen Intensitätswerte für die Y-, Cb- und Cr-Kanäle in Abhängigkeit von der x-Koordinate des Ursprungsbildes dargestellt. Jedes dieser Diagramme stellt für die Abbildung 3.1a ermittelte Farbkanalwerte in jeweils drei unterschiedlichen Belichtungssituationen dar. Sie sollen auch dazu dienen, robuste Eigenschaften für Objekte festzustellen und von nicht robusten Eigenschaften abzugrenzen.

Für die weiße Spielfeldlinie zwischen den Punkten P_1 und P_2 im Diagramm 3.1b ist beispielsweise ein deutlicher Anstieg des Y-Wertes, also der Helligkeit in diesem Bereich, erkennbar. Analog dazu steigt zwischen den Punkten P_3 und P_4 der Wert der Cr-Komponente aufgrund des Grün-Orange-Kontrastes zwischen Feldboden und Ball an. Am Torpfosten, zwischen den Punkten P_5 und P_6 , erfolgt im Vergleich zum Umfeld sowohl ein Anstieg des Cb-Wertes im Diagramm 3.1c als auch einen Helligkeitsabfall in 3.1b. Diese lokalen Wertveränderungen eines bestimmten Farbkanals zwischen Objekten und ihrer Umgebung sind nahezu unabhängig von der jeweiligen Beleuchtungssituation und können folglich als robuste Eigenschaften definiert werden. Hingegen variieren die absoluten Y-, Cb-, und Cr-Werte dieser Objekte je nach Lichtverhältnissen deutlich und stellen demnach keine zuverlässige Eigenschaft für die Objekterkennung dar.

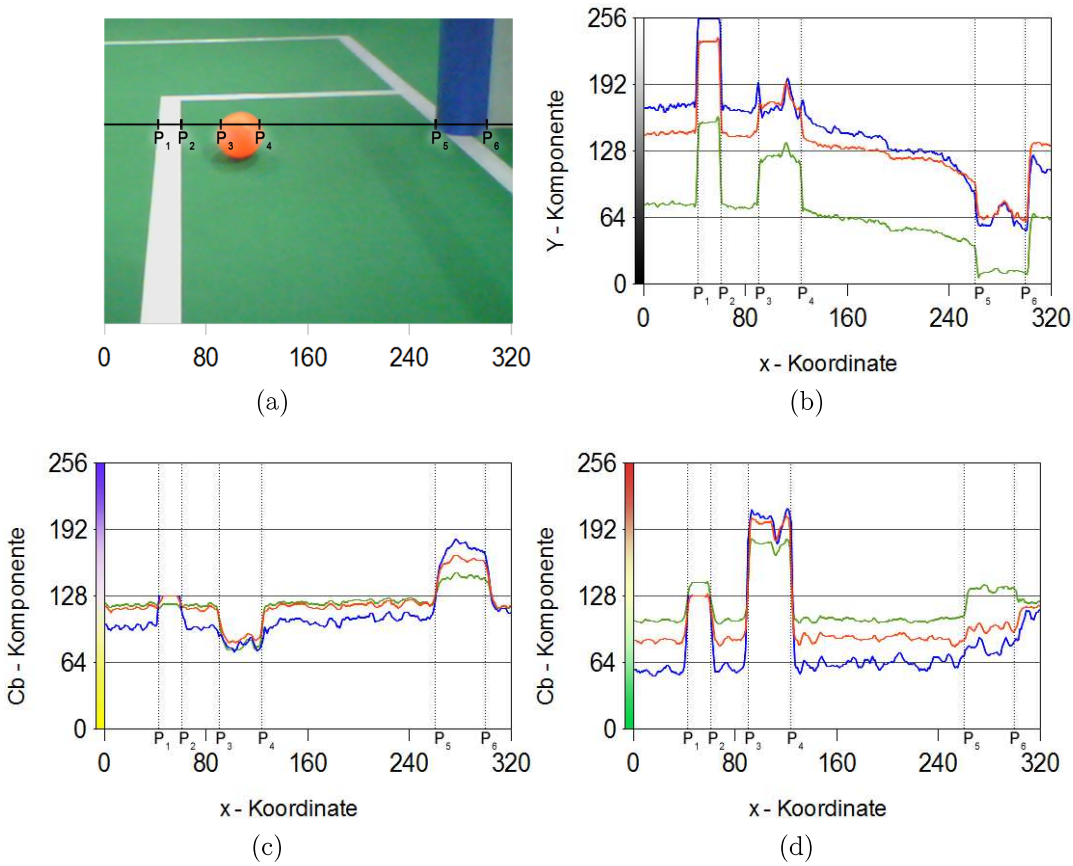


Abb. 3.1: Für das abgebildete Kamerabild werden für die auf der markierten Profillinie liegenden Pixel (a) die Farbwerte für den Y-Kanal (b), den Cb-Kanal (c) und den Cr-Kanal (d) für jeweils drei unterschiedliche Belichtungssituationen dargestellt.

In den Diagrammen 3.1b-d wird des Weiteren deutlich, dass für die drei Messreihen die auf den Objektkanten liegenden Positionen P_1 bis P_6 unverändert bleiben. Aus der äußeren

Kontur der Objekte resultiert deren Form, die als ein weiteres robustes Erkennungsmerkmal dient, da sie bei Helligkeits- oder Farbverschiebungen im Allgemeinen unverändert bleibt. Objektformen seien definiert als: „Shape is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object." [Ken84]. Als Beispiel sei hier der Spielball erwähnt, der, unabhängig von seiner Position, Größe oder Rotation im Kamerabild, eine Kreisform aufweisen sollte. Je nach Blickwinkel kann die Form bestimmter Objekte zwar aufgrund der perspektivischen Verzerrung variieren, jedoch nach wohlbekannten Regeln, auf welche in den Abschnitten Linien- und Torerkennung konkret eingegangen wird. Des Weiteren lassen sich Größe, Position und Rotation der Objekte einschränken und stellen ein nützliches Kriterium zur Filterung von Fehlerkennungen dar. Die genannten Eigenschaften werden im Folgenden als Basis zur Objekterkennung verwendet und sind zusammenfassend in Tabelle 3.3 nochmals aufgelistet.

Es sind durchaus weitere Eigenschaften, wie beispielsweise die Farbvarianz, Entropie oder Textureigenschaften denkbar, diese werden jedoch aufgrund von einigen empirischen Tests am Bildmaterial nicht verwendet. Gründe hierfür sind unter anderem das variable Rauschverhalten bei unterschiedlichen Schutterzeiten der Kamera, die die Farbvarianz oder Entropie möglicherweise wesentlich beeinflussen, sowie potenzielle Lichtreflexionen und Spiegelungen an den Objekten, die eine Texturanalyse erschweren. Vielmehr konzentriert sich diese Abhandlung auf die wesentlichen Objekteigenschaften, die wohldefinierten Regeln folgen. Zur Objekterkennung werden jedoch selten alle in Tabelle 3.3 aufgelisteten Eigenschaftskategorien gleichzeitig herangezogen, vielmehr wird objektspezifisch eine ökonomische Auswahl aus diesen getroffen.

Eigenschaft	Beispiel
Farbdifferenzen	Unterschied zwischen Ball- und Bodenfarbe
Form	Kreisform des Balls
Größe	Feldboden ist das Objekt mit der größten Fläche
Position	Ball ist nur innerhalb des Spielfeldes zu suchen
Rotation	Torpfosten haben eine vertikale Ausrichtung

Tab. 3.3: Verwendete Eigenschaften zur Erkennung von Objekten

3.3 Festlegung der Verarbeitungsreihenfolge

Vor der Etablierung des in dieser Arbeit vorgestellten Objekterkennungsverfahrens, in dem die Bildverarbeitung in mehreren Erkennungs- und Filterungsphasen erfolgt, ist eine Festlegung der Reihenfolge dieser einzelnen Verarbeitungsschritte von entscheidender Bedeutung, denn nach jedem Objekterkennungsschritt stehen neue Informationen über den aktuellen Bildinhalt zur Verfügung, deren Nutzung für anschließende Erkennungsschritte einen Vorteil bringen kann. Jene Vorteile ergeben sich beispielsweise aus charakteristischen Farbwerten für bereits erkannte Objekte als auch aus Informationen über deren Form, Position, Größe und Rotation. Die im Folgenden definierte und in Abbildung 3.2 dargestellte Reihenfolge der Verarbeitungsschritte ist Teil eines Gesamtkonzepts dieser Arbeit, welches in den Abschnitten 3.4 bis 3.6 detailliert beschrieben und begründet wird.

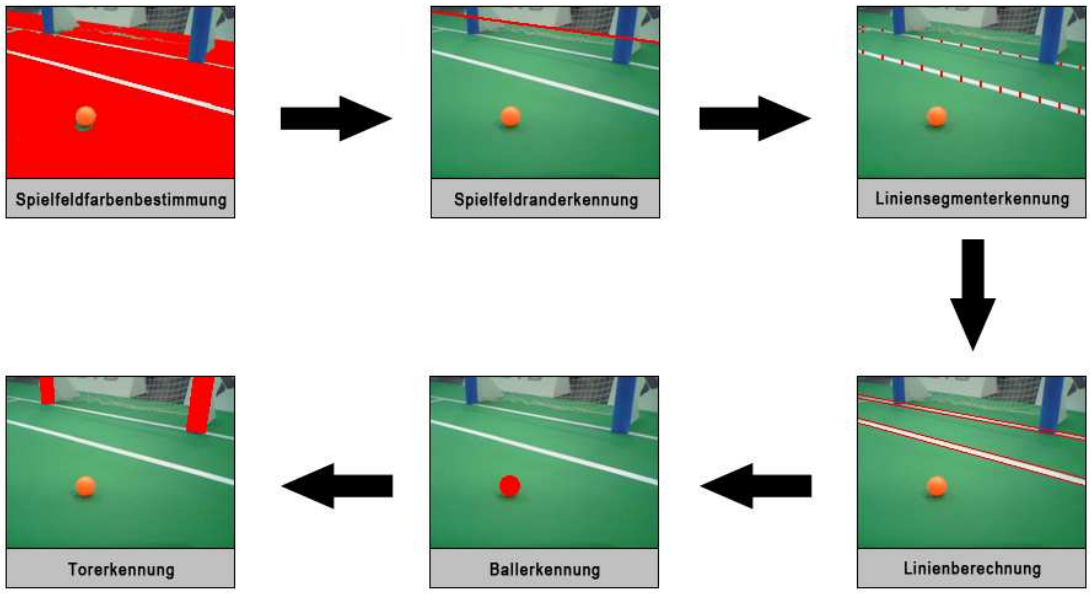


Abb. 3.2: Schematischer Programmablauf bei der Objekterkennung

Als erster Verarbeitungsschritt wird in Abschnitt 3.4 der mittlere Farbton des Spielfeldes bestimmt, denn jener kann ohne Vorwissen durch Ermittlung der dominierenden Farbe des Bildes errechnet werden. Anhand dieser Information lassen sich in weiteren Schritten alle andersfarbigen Objekte von dem Feldboden unterscheiden. Da sich die gesuchten Objekte ganz oder teilweise innerhalb des eigenen Spielfeldes befinden müssen, wird im zweiten Verarbeitungsschritt das Bild anhand des Helligkeitskanals in mehrere zusammengehörige Bereiche bzw. Pixelketten zerlegt und werden hiermit sowohl Spielfeldgrenzen (Abschnitt

3.5.4) als auch potenzielle Liniensegmente (Abschnitt 3.5) ermittelt. In Abschnitt 3.5.5 wird anschließend ein Verfahren zur Gruppierung jener Liniensegmente beschrieben, mit dem sowohl eine Generierung relevanter Linienobjekte ermöglicht als auch eine Identifizierung falsch erkannter Liniensegmente realisiert wird. Für die weitere Erkennung des Spielballs (Abschnitt 3.6) stehen durch die bisherigen Verarbeitungsschritte nun nützliche Informationen über die Feldbodenfarbe, die Linienfarbe und die Spielfeldbegrenzung zur Verfügung. Durch eine Kreisformerkennung (Abschnitt 3.6.5) kann die Position des Balls ermittelt und dessen Farbton abgespeichert werden. Als letzter und anspruchstvollster Schritt erfolgt, unter Verwendung aller bisher bestimmten Farbwerte, die Analyse des Bildes nach potentiellen Torpfosten (Abschnitt 3.7).

3.4 Feldfarbenermittlung

Die Extraktion der relevanten Informationen aus dem Roboterfußball-Bildmaterial wird durch eine Spielfelderkennung wesentlich vereinfacht, da sich alle relevanten Objekte ganz oder teilweise innerhalb des eigenen Spielfeldes befinden und außerhalb dementsprechend für das Spielgeschehen unwichtige Informationen vorkommen. Die Erkennung des Spielfeldes ermöglicht somit sowohl eine Fokussierung wesentlicher Objekte als auch die Ausblendung irrelevanter Informationen. Viele Ansätze wurden dazu, insbesondere im Bereich von Fußball-Analyse-Systemen im TV-Bereich, veröffentlicht [NYC10a, TMS04, ET03, NC08] und basieren im Allgemeinen auf der initialen Berechnung der dominanten Farbe des Spielfeldes und der anschließenden Identifikation der Spielfeldgrenzen. In dieser Arbeit wurde ein zeiteffizientes Verfahren zur robusten Extraktion des dominanten Spielfeldfarbtons entwickelt, in dem die im folgenden Abschnitt genannten Eigenschaften und Heuristiken zur Erkennung Anwendung finden.

3.4.1 Eigenschaften

Im Regelwerk der SPL-Liga [Rob10e] wird folgende Information über den Felduntergrund vorgegeben: „The field (carpet) itself is green (color is not specified, but it should not be too dark).”

Es sollte sich also um einen grünen Farbton handeln, wobei der exakte Farbton nicht definiert wird. So zeigt Abbildung 3.3 beispielhaft den direkten Vergleich zwischen den Feldfarben bei der WM 2009 in Graz und der WM 2011 in Istanbul, die einen deutlichen Unterschied aufweisen.

(a) (b) (c)

Abb. 3.3: Vergleich der Spielfeldfarben zur WM 2009 in Graz (a) und der WM 2011 in Istanbul (b). Bild (c) zeigt die direkte Gegenüberstellung der beiden verschiedenen Feldfarben.

Farbdifferenzen

Aufgrund der oben genannten Vorgabe durch das Regelwerk ist zur Erkennung des Feldbodens vermutlich der Cr-Kanal im YCbCr-Modell der aussagekräftigste. Aus diesem Grunde wurde in der vorliegenden Arbeit unter Verwendung der Testbilddatenbank zur Überprüfung jener Annahme die Eigenschaft der Farbdifferenzen zwischen dem Feldboden und der Umgebung untersucht:

Für jedes Einzelbild der Datenbank wurden, wie schon in Abschnitt 3.1 beschrieben, die Information zur Feldgrenze durch zwei Geraden abgespeichert. Dadurch lassen sich für jedes Bild alle Pixel außerhalb des Spielfeldbodens finden und werden im Weiteren als Hintergrund bezeichnet. Untersucht wird nun die Eignung eines jeden Farbkanals zur Abgrenzung der Feldfarbe vom Hintergrund. Dazu wurden die Werte des in der Datenbank zu jedem Bild abgelegten $(\bar{Y}, \bar{Cb}, \bar{Cr})$-Tripels der tatsächlichen Spielfeldfarbe mit den drei ermittelten Farbwerten jedes gefilterten Pixels des Hintergrundes verglichen und je die Anzahl der Pixel bestimmt, für welche diese Differenz entweder positiv oder negativ ist

$$Y_{neg} := \sum_{i=1}^{n} V(Y_i - \bar{Y}_i) \qquad Y_{pos} := \sum_{i=1}^{n} V(\bar{Y}_i - Y_i)$$

$$Cb_{neg} := \sum_{i=1}^{n} V(Cb_i - \bar{Cb}_i) \qquad Cb_{pos} := \sum_{i=1}^{n} V(\bar{Cb}_i - Cb_i)$$

$$Cr_{neg} := \sum_{i=1}^{n} V(Cr_i - \bar{Cr}_i) \qquad Cr_{pos} := \sum_{i=1}^{n} V(\bar{Cr}_i - Cr_i)$$

wobei

$$V(d) := \begin{cases} 1, & falls\ d < 0 \\ 0 & sonst \end{cases}$$

und n Anzahl jener Hintergrund-Pixel sei. Nach Analyse des Datenbestandes ergeben sich für die oben definierten Summen folgende Messwerte:

Farbkanal	negativ		positiv	
Y	28,7 Mio. Pixel	51,7%	26,5 Mio. Pixel	47,7%
Cb	9,4 Mio. Pixel	16,9%	44,3 Mio. Pixel	79,7%
Cr	0,15 Mio. Pixel	0,26%	55,5 Mio. Pixel	99,7%

Tab. 3.4: Ergebnisübersicht zur Farbdifferenzmessung bezüglich der realen Spielfeldfarbe

Aus diesen Ergebnissen lässt sich schließen, dass außerhalb der Spielfeldgrenzen statistisch bei nur 0,26% aller Pixel ein Cr-Wert kleiner als $\bar{Cr}$ zu erwarten ist, so dass die anfängliche Annahme bestätigt wird, dass die $Cr-$Komponente ein guter Ansatzpunkt ist, um die Feldfarbe vom Hintergrund abzugrenzen. Eine Kontrollmessung der Pixel innerhalb der Feldgrenzen zeigt für $Cr < \bar{Cr}$ einen deutlichen Anstieg auf 20,8% der Bildpunkte, welche hauptsächlich Teil der grünen Spielfeldfläche sind.

Größe der Fläche

Um eine Aussage über die zu erwartende Größe der Fläche tätigen zu können, ist es ebenso möglich, das Testbildmaterial statistisch zu analysieren und den prozentualen Anteil der Pixel zu bestimmen, die der Feldfarbe zugeordnet werden können. Durch ein einfaches Schwellwertverfahren kann mit den vorgegebenen $(\bar{Y}, \bar{Cb}, \bar{Cr})$-Tripeln errechnet werden, dass von den 600 Testbildern ca. 70% der Pixel, also 128,7 Mio. von insgesamt 184,3 Mio. Pixel, der Feldfarbe zugeordnet werden können. Dieser recht hohe Anteil lässt sich auch durch die Überlagerung aller 600 Testbilder visualisieren, wie in Abbildung 3.4a gezeigt. Bei der späteren Algorithmenentwicklung ist jedoch zu beachten, dass es sich hierbei lediglich um einen Durchschnittswert diverser Bilder handelt. Die Größe der Spielfeldfläche in einem Bild schwankt jedoch je nach Blickwinkel des Roboters zwischen 0 und 100% (Vergleich Abbildung 3.4c und 3.4b).

(a) (b) (c)

Abb. 3.4: Bild (a) zeigt ein Durchschnittsbild, welches aus Bildung des arithmetischen Mittels der Farbwerte der jeweiligen Pixel aller Testbilder generiert wurde, wobei die Sättigung zur besseren Visualisierung verstärkt wurde. Im Vergleich dazu sind für den Anteil der sichtbare Spielfeldfläche die beiden Extrema (b) und (c) dargestellt.

3.4.2 Inhomogenität der Feldfarbe

Neben dem natürlichen Rauschen des Bildes gibt es weitere Störeinflüsse, die die Anwendbarkeit von Schwellwertverfahren limitieren. Diese Störgrößen können nach dem Grad ihrer Vorhersehbarkeit eingeteilt werden. So führen Schatten und Lichtreflexionen zu unvorhersehbaren Inhomogenitäten auf Objektflächen, wohingegen das Kamerabild zusätzlich einen vorhersehbaren, natürlichen Randlichtabfall aufweist. Diese Helligkeitsabnahme am Rand des Bildes ist hauptsächlich Folge des unterschiedlichen Einfallswinkels des Lichtes auf die einzelnen planar angeordneten CMOS-Sensorelemente. Diese statische Bildverfälschung kann gegebenenfalls durch eine Inhomogenitätskorrektur kompensiert werden. Hierzu wurde zunächst untersucht, inwiefern eine Veränderung der Helligkeit und Farbe in Abhängigkeit der Pixelkoordinate im Kamerabild unter gleichbleibenden Lichtbedingungen auftritt.

Messung des sensorbedingten Helligkeitsabfalls

Eine einfache Methode zur Messung des Randlichtabfalls im Kamerabild besteht darin, eine gleichmäßig ausgeleuchtete, farblose Oberfläche parallel zur Bildebene vor dieser Kamera zu positionieren und aufzunehmen. Durch Analyse der Helligkeiten der so aufgenommenen Pixel könnte nun auf die Stärke des Randlichtabfalls geschlossen werden. Da aber in Realität keine optimalen Bedingungen für die gleichmäßige Ausleuchtung der genannten Oberfläche garantiert werden können, wird folgendes Vorgehen zur Messung des Randlichtabfalls angewendet:

Statt eine einzige Aufnahme des gesamten Kamerabildes zu analysieren, werden mehrere Einzelmessungen mit unterschiedlichen Neigungs- und Rotationswinkeln der Kamera durchgeführt. Jeder dieser Einzelmessungen schließt sich eine programmgesteuerte definierte Drehbewegung des Roboterkopfes an, so dass sich der Neigungswinkel der auf dem Kopf montierten Kamera entsprechend verändert. Es lässt sich anhand dieser Winkel bestimmen, an welchen Bildkoordinaten ein- und dieselbe Spielfeldkoordinate projiziert wird. Folglich können an diesen Bildkoordinaten Helligkeits- und Farbwerte gemessen werden, die ausschließlich durch sensorbedingte Inhomogenitäten verfälscht wurden, da an dieser Stelle immer das gleiche Objekt abgebildet wird. Voraussetzung hierbei war die Konstanz der Lichtbedingungen des Raumes während sämtlicher Messungen.

Betrachtung einer Inhomogenitätskorrektur

Zur Korrektur der hier beschriebenen statischen Inhomogenitäten in Form eines Randlichtabfalls soll zu deren Beschreibung zunächst ein geeignetes mathematisches Modell untersucht werden. In der Literatur wird der natürliche Helligkeitsverlust am Rand eines Kamerabildes durch das cos^4-Gesetz [KP07] beschrieben. Dieses definiert für die Abbildung eines gleichmäßig hellen Motivs durch ein Objektiv die Abnahme der Bildhelligkeit um den Faktor $C(\alpha) := C(0) * \cos^4 \alpha$ gegenüber der Helligkeit $C(0)$ in der Bildmitte. Der Winkel α repräsentiert den Einfallswinkel eines Lichtstrahls auf die Kameraoptik, wobei für einen parallel der optischen Achse der Kamera verlaufenden Strahl der Einfallswinkel $\alpha = 0$ sei. Eine Korrektur des Bildes anhand dieses Modells erfolgt nun mittels Division der Helligkeitswerte Y aller Pixel durch den Kehrwert des Korrekturfaktors $C(\alpha)$ gemäß der Formel:

$$Y' := \frac{Y}{\cos^4 \alpha} \tag{3.1}$$

Von Interesse ist nun, inwiefern die gemessenen Inhomogenitäten diesem mathematischen Modell entsprechen und folglich hiermit adäquat korrigiert werden können. Ferner setzt jene Formel voraus, dass das verwendete Objektiv - im Gegensatz zur Kamera des Roboters - weder Verzeichnung noch Vignettierung oder Pupillenaberration aufweist. Allein diese Bedingung widerspricht schließlich der Realisierung einer optimalen Korrektur des durch den Roboter aufgenommenen Kamerabildes. Zur weiteren Verdeutlichung des Problems sind in Abbildung 3.5 sowohl die Werte der relativen Helligkeiten $Y_{rel} := \frac{Y}{Y_{center}}$ als auch des cos^4-Gesetzes dargestellt, wobei $Y_{center} := C(0)$. Idealerweise liegen einander entsprechende Funktionswerte beider Graphen möglichst dicht beieinander, während sie

im untersuchten Fall jedoch deutlich voneinander abweichen. Aus diesem Grund kann die Inhomogenitätskorrektur nicht auf Basis des cos^4-Gesetzes erfolgen.

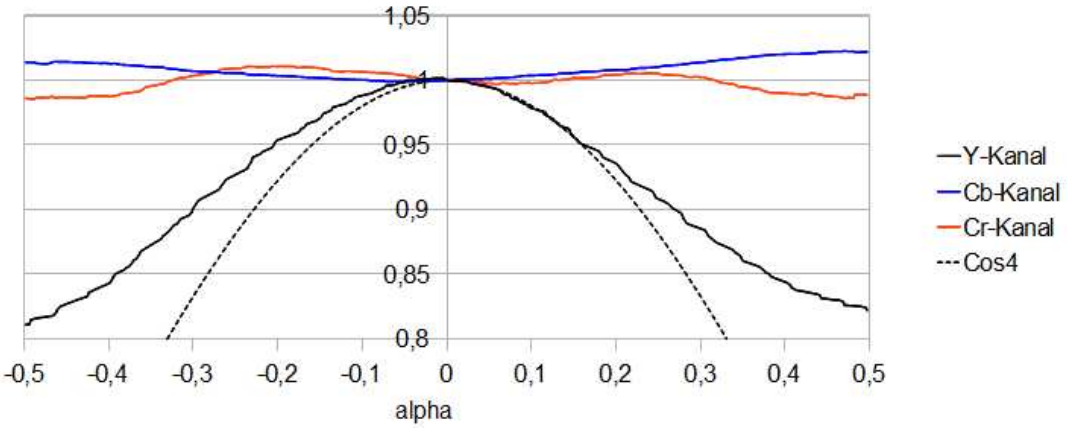


Abb. 3.5: Randlichtabfall: Die gemessene Helligkeit am Rand entspricht nur etwa 80% der Helligkeit in der Bildmitte. Nur geringer Einfluss auf den Cb- und Cr-Kanal.

Eine leicht zu implementierende Alternative ist dahingegen die Erstellung einer eindimensionalen Lookup-Tabelle für verschiedene Werte von α und den entsprechend gemessenen Korrekturwerten. Hiermit kann bei gegebenem α durch lineare Interpolation zweier benachbarter Werte der Lookup-Tabelle ein Korrekturwert zeiteffizient errechnet werden. Es sei erwähnt, dass bei der Erstellung einer solchen Lookup-Tabelle eine Mittlung verschiedener Messungen in unterschiedlichen Lichtsituationen und deren anschließende Glättung empfehlenswert ist, um Sensorrauschen zu mindern. Abbildung 3.6 zeigt beispielhaft ein mit dieser Methode korrigiertes Bild.

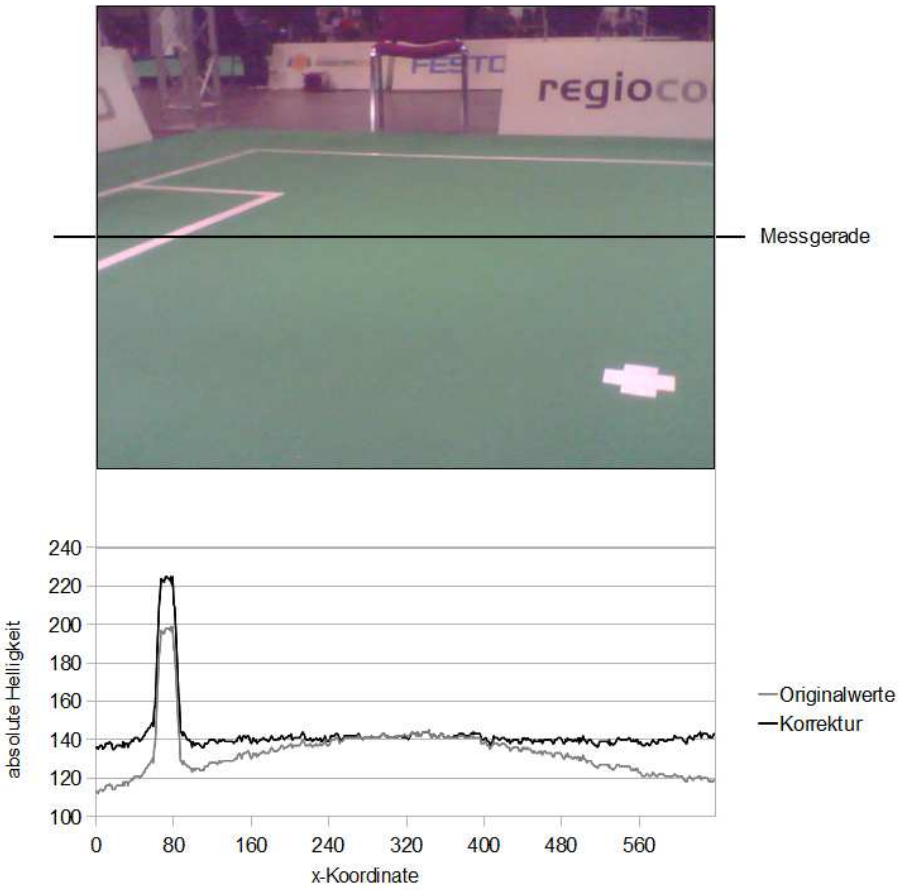


Abb. 3.6: Inhomogenitätskorrektur als Höhenprofil dargestellt

Es sei jedoch darauf hingewiesen, dass externe Helligkeitsschwankungen aufgrund der ungleichmäßigen Beleuchtung des Spielfelds, wie zum Beispiel Schatten, oft einen zum Randlichtabfall vergleichsweise stärkeren Helligkeitsunterschied verursachen und durch das beschriebene Inhomogenitätskorrekturverfahren nicht berücksichtigt werden.

Da zudem die Korrektur eines jeden Helligkeitswertes des Bildes zu einem hohen Berechnungsaufwand führen würde, kann die Korrektur zwecks Aufwandsverminderung alternativ nur für bestimmte Pixel durchgeführt werden, für welche eine spätere Weiterverarbeitung zur Objekterkennung erfolgt.

3.4.3 Algorithmische Umsetzung

Um für ein Kamerabild den Farbton des Spielfeldbodens zu berechnen, wird aufgrund der guten Differenzierbarkeit der Spielfeldfarbe vom Hintergrund im Cr-Kanal zunächst das zugehörige Histogramm berechnet. Die Idee hierbei besteht darin, dass aufgrund der hohen

Anzahl an vorkommenden gleichfarbigen Feldpixeln im Kamerabild möglicherweise der dominante Wert Cr_{max} dem angestrebten Wert $\bar{Cr}$ nahe kommt. Für eine zeiteffiziente Histogrammberechnung wird die Anzahl der Abtastpunkte durch Verwendung eines starren Rasters reduziert. Es sei s_{Raster} der horizontale bzw. vertikale Abstand zwischen zwei benachbarten Punkten auf diesem Raster. Hierzu verdeutlicht die nachfolgende Abbildung, welche Bildpunkte bei einem Pixelabstand von $s_{Raster} = 8$ ausgewählt würden. Der Koordinatenursprung befindet sich hier - wie in allen im Rahmen dieser Arbeit gezeigten Bildern - in der linken oberen Ecke und die Abmessungen des Bildes betragen $w := 640$ sowie $h := 480$.

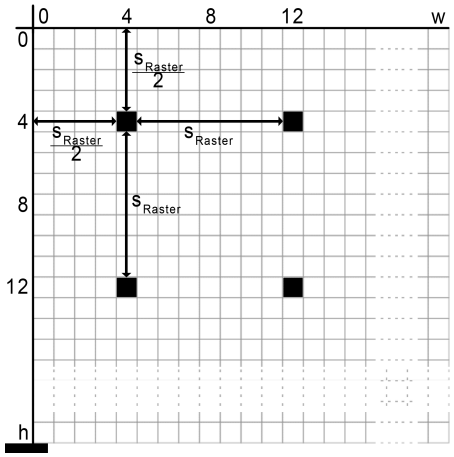
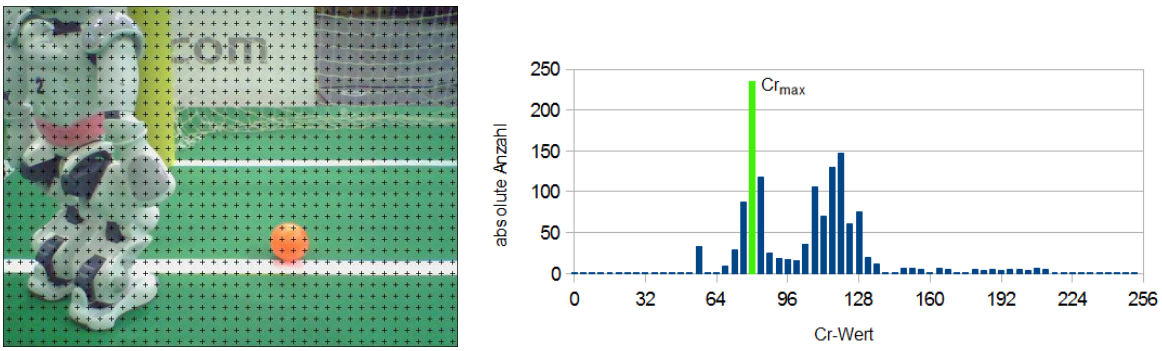


Abb. 3.7: Visualisierung der durch ein Suchpunktraster ausgewählten Pixel am Beispiel von $s_{Raster} = 8$

Die Reduktion der Pixel durch das Raster führt zu entsprechend geringerem Aufwand für die Histogrammberechnung, wohingegen jedoch ebenfalls die Repräsentativität seiner Werte schwindet. Um dieser Qualitätsabnahme entgegenzuwirken, kann die Histogrammklassenbreite q_{Bin} erhöht werden, so dass es bei 256 Farbklassen insgesamt nur noch $n_{Klassen} := \frac{256}{q_{Bin}}$ Histogrammklassen gibt, deren Balkenlängen sich jedoch mit einer Steigerung von q_{Bin} erhöhen.

Abbildung 3.8 zeigt ein Beispiel für $s_{Raster} := 16$ und $q_{Bin} := 4$.



(a) (b)

Abb. 3.8: Für ausgewählte Pixel (a) wurden die entsprechenden Cr-Werte in einem Histogramm (b) eingetragen. Der dominante (maximale) Wert entspricht dem Cr-Wert der Feldfarbe in Abbildung (a).

Zur Histogrammberechnung wurde zunächst der nachfolgende Algorithmus 3.1 verwendet:

Algorithm 3.1 getHistogram
pixels:=Pixeldaten eines Kanals des Bildes.

```
getHistogram(pixels, s_Raster, q_Bin)
{
    for(x := s_Raster div 2; x < w; x := x + s_Raster)
    {
        for(y := s_Raster div 2; y < h; y := y + s_Raster)
        {
            hist[pixels_{x,y} div q_Bin]++
        }
    }
    return hist
}
```

wobei die Maximumbestimmung innerhalb des errechneten Histogramms erfolgt durch:

Algorithm 3.2 getMax
data := Histogrammwerte

```
getMax(data, q_Bin)
{
    max := 0
    for(i := 0; i < sizeof(data); i := i + 1)
    {
        if(data_i > max)
        {
            max := data_i
            result := i * q_Bin + q_Bin div 2
        }
    }
    return result
}
```

Ein erster, sehr einfacher und schneller Algorithmus zur Klassifikation der Pixel bezüglich ihrer Spielfeldzugehörigkeit verwendet ausschließlich

$$Cr_{max} := getMax(getHistogram(data_{Cr}, s_{Raster}, q_{Bin})). \tag{3.2}$$

Dabei werden alle Pixel, deren Abstand ihres Cr-Wertes zu Cr_{max} kleiner als ein vorgegebener Schwellenwert T_{Cr} ist, dem Spielfeld gemäß des Algorithmus 3.3 zugeordnet.

Algorithm 3.3 isFieldcolor

```
isFieldcolor(Cr, Cr_max, T_Cr)
{
    if(|Cr - Cr_max| < T_Cr)
        return true
    else
        return false
}
```

Die Anwendung des Algorithmus 3.3 auf das in Abbildung 3.8a dargestellte Bild ist nachfolgend für den Schwellenwert $T_{Cr} = 15$ durch Hervorhebung der zugeordneten Feldpixel visualisiert.

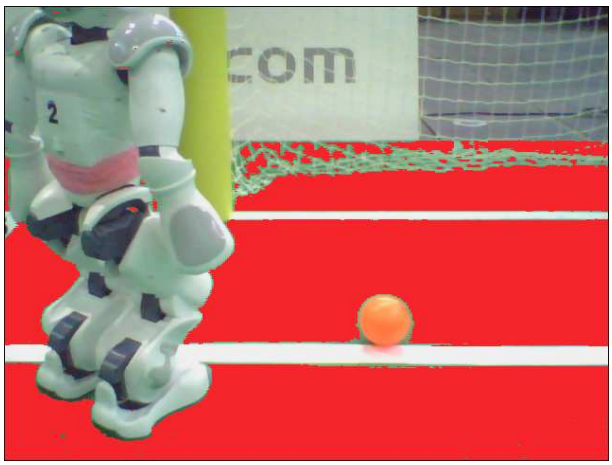


Abb. 3.9: Alle Pixel, für die isFieldcolor(Cr,Cr_{max},15)=true ist, wurden farbig markiert.

Hierdurch wird offensichtlich, dass allein die Bestimmung des dominierenden Farbwertes im Cr-Histogramm für dieses Beispiel zu einem brauchbaren Ergebnis führt. Um jedoch eine konkrete Aussage über die Erkennungsleistung dieses Verfahrens treffen zu können, sind weitere Untersuchungen unter Zuhilfenahme der Testbilddatenbank nötig.

Dazu werden zunächst die Parameter s_{Raster} und q_{Bin} auf ihre Auswirkung auf die Erkennungsleistung des Algorithmus 3.3 untersucht. Die Erkennungsleistung e_f sei definiert als die relative Anzahl der Testbilder, bei denen der geschätzte Wert der Feldfarbe Cr_{max} und der tatsächliche Wert $\bar{Cr}$ dicht beieinander liegen:

$$e_f := \frac{1}{n_B} \sum_{i=1}^{n_B} W(|\bar{Cr}_i - Cr_{Max,i}|) \tag{3.3}$$

wobei

$$W(d) := \begin{cases} 1, & falls\ d < 8 \\ 0 & sonst \end{cases}$$

ist. Sei der Abstand $d := |Cr - Cr_{Max}|$ also kleiner als ein für dieses Experiment heuristisch festgelegter Wert, so gilt die Feldfarbe als erkannt. Das somit resultierende Diagramm 3.10 für verschiedene q_{Bin}- und s_{Raster}-Werte zeigt einen deutlichen Zusammenhang zwischen der Erhöhung der Histogrammklassenbreite q_{Bin} und der Steigerung der Erkennungsleistung. Auf eine weitere Erhöhung von q_{Bin} wurde aufgrund der entstehenden Ungenauigkeiten durch die Histogrammquantisierung verzichtet.

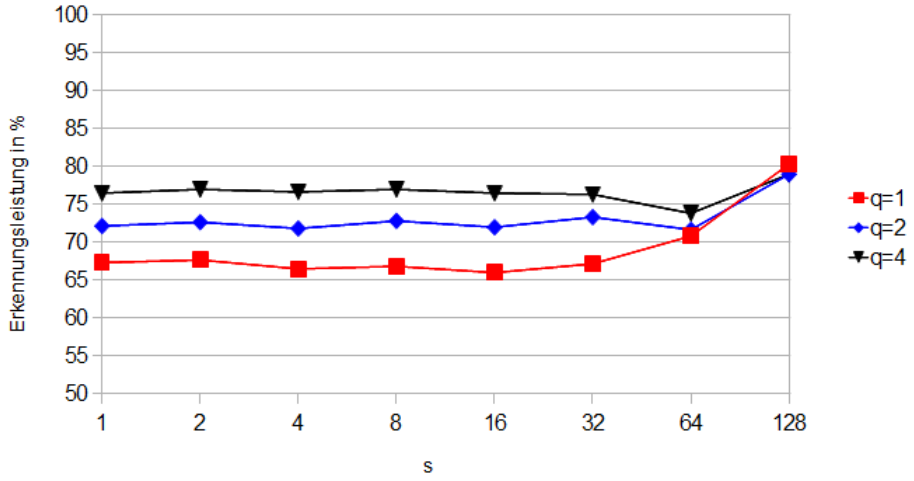


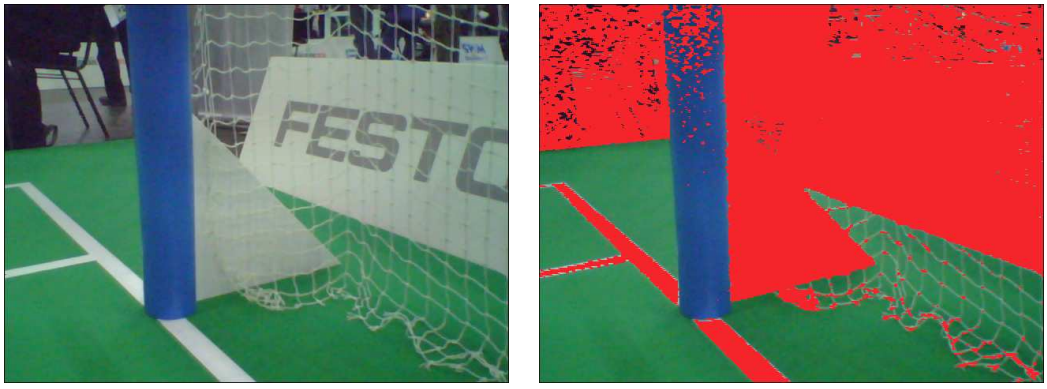
Abb. 3.10: Darstellung der Erkennungsleistung für die Berechnung nach 3.2 in Abhängigkeit von der Rastergröße s_{Raster}.

Bei der Erhöhung der Rastergröße s_{Raster} tritt des Weiteren folgendes Phänomen auf: Die aufgrund der hiermit verbundenen fallenden Repräsentativität des Histogramms zu erwartende schrittweise Verschlechterung der Erkennungsleistung bleibt aus. Vielmehr verhält sie sich konstant und verbessert sich darüber hinaus für $s_{Raster} := 128$. Für letzteren Wert wurden zur Berechnung der Histogramme nur $n_{Raster} := 20$ Pixel pro Bild analysiert. Im ungünstigsten Fall kann es also 20 verschiedene Histogrammklassen geben, die alle den gleichen absoluten Wert von Eins haben. Der Algorithmus 3.2 zur Berechnung des Histogrammmaximums würde aber von diesen gleichwertigen Histogrammklassen genau diejenige auswählen, für die Cr_{max} minimal ist. Im diesem Fall wird also von den 20 Pixeln gleichsam derjenige ausgewählt, der am stärksten grün erscheint. Bedenkt man, dass - wie in Abschnitt 3.4.1 gezeigt - für nur etwa 0,15% der Hintergrundpixel der zugehörige Cr-Wert kleiner als $\bar{Cr}$ ist, wird offensichtlich, dass „der grünste Pixel" oft nahe der Spielfeldfarbe liegt. Allein nach dem intensivsten Grünton im Bild zu suchen ist jedoch keinesfalls robust, bleibt doch die Eigenschaft der Größe des Feldes hierbei unberücksichtigt. Für eine robuste Erkennung sollten demnach deutlich mehr als 20 Pixel einbezogen und entsprechend s_{Raster} verkleinert werden. Wie genau dieser Wert eingestellt wird, ergibt sich anhand einiger weiterer im Folgenden beschriebenen Untersuchungen.

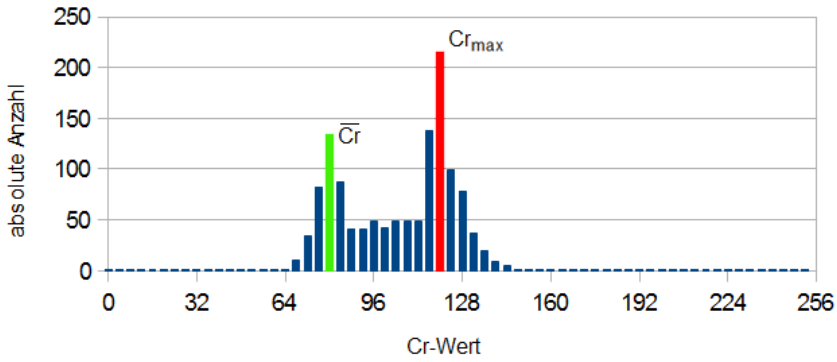
Für das weitere Vorgehen sind genau die Bilder von Interesse, bei denen eine Fehlerkennung der Feldfarbe bei kleineren s_{Raster}-Werten aufgetreten ist, da hier gegebenenfalls Verbesserungspotenzial vorliegt. Abbildung 3.11 zeigt ein solches Beispiel. Das entsprechende Cr-Histogramm zu diesem Bild gibt Aufschluss über den Grund dieser Fehlerkennung:

Die meisten weißen, grauen und schwarzen Objektflächen in der oberen Hälfte des Bildes

haben unabhängig von ihrer Helligkeit einen ähnlichen Cr-Wert von etwa 120, was insgesamt zu einem Maximum innerhalb des Histogrammes an genau dieser Stelle führt. Ein lokales Maximum liegt zwar korrekt bei $\bar{Cr}$, wird jedoch durch die höhere Anzahl an gleichartigen Hintergrundpixeln vom entsprechenden globalen Maximum übertroffen, woraus letztendlich die Fehlerkennung resultiert. Der Algorithmus ist also anfällig für farblose oder einfarbige Flächen, die größer sind als die Fläche des sichtbaren Spielfeldes.



(a) (b)



(c)

Abb. 3.11: Für alle Pixel in (a) für die gilt $|Cr_{max} - Cr_{xy}| < 15$ wurde eine rote Markierung (b) gesetzt. Das zum Ursprungsbild zugehörige Histogramm ist in Abbildung (c) dargestellt.

Bei der Berechnung des dominierenden Farbwertes durch ein Histogramm wird im Regelfall nur die Größe der Fläche durch die Anzahl gleichartiger Messwerte berücksichtigt, nicht jedoch die Eigenschaft der Farbintensität. Zur Lösung dieses Problems wird folgende gewichtete Histogrammberechnung eingeführt, die die in Abschnitt 3.4 beschriebene Eigenschaft des Cr-Wertes nutzt:

Algorithm 3.4 getWeightedHistogram

```
getWeightedHistogram(hist, q_Bin)
{
    for(i := 0; i < sizeof(hist); i := i + 1)
    {
        hist[i] := hist[i] * G(i * q_Bin);
    }
    return hist;
}
```

wobei die Gewichtsfunktion

$$G(i) := \max(0, 128 - i) \tag{3.4}$$

ist. Für alle Cr-Werte größer gleich 128 ist die Gewichtsfunktion Null, da angenommen werden kann, dass ein als „grün" definierter Farbton im Kamerabild weder grau (Cr=128) noch rot (Cr>128) sein wird. Je kleiner der Cr-Wert ist, desto höher wird der entsprechende Histogrammbalken gewichtet. Die resultierenden Auswirkungen auf die Erkennungsleistung sind für verschiedene s_{Raster}- und q_{Bin}-Werte im folgenden Diagramm dargestellt. Im Vergleich zur ungewichteten Histogrammberechnung ist hier ein deutlicher Anstieg der Erkennungsleistung auf bis zu 98% zu verzeichnen:

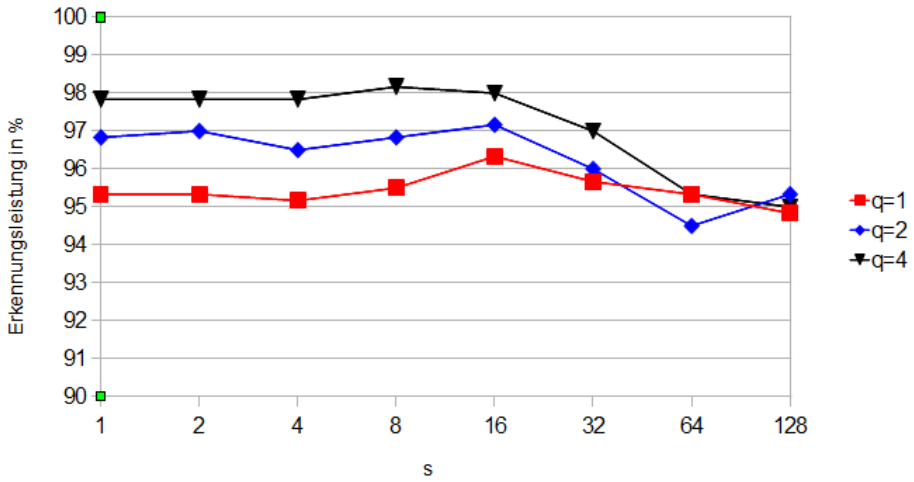


Abb. 3.12: Darstellung der Erkennungsleistung für die Histogrammgewichtung nach 3.4 in Abhängigkeit von der Rastergröße s_{Raster}.

Eine weitere Verbesserung wurde erzielt, indem die Funktion G durch eine quadratische Gewichtung G' mit

$$G'(i) := \max{}^2(0, 128 - i) \tag{3.5}$$

ersetzt wurde. Die Ergebnisse sind im Diagramm 3.13 dargestellt. Für $q_{Bin} := 4$ und $s_{Raster} := 16$ konnte für 598 von 600 Bildern der Cr-Wert der Feldfarbe korrekt berechnet werden, so dass diese beiden Parameter bei der weiteren Nutzung der Feldfarbenerkennung als Standardwerte festgelegt wurden. In den zwei falsch erkannten Bildern kam das eigene Spielfeld nicht vor, weshalb es unmöglich war, die Feldfarbe korrekt zu berechnen.

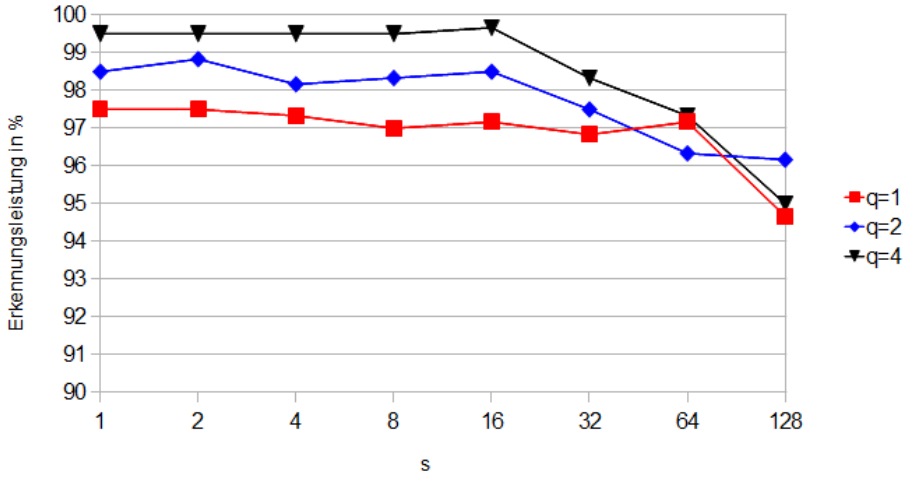
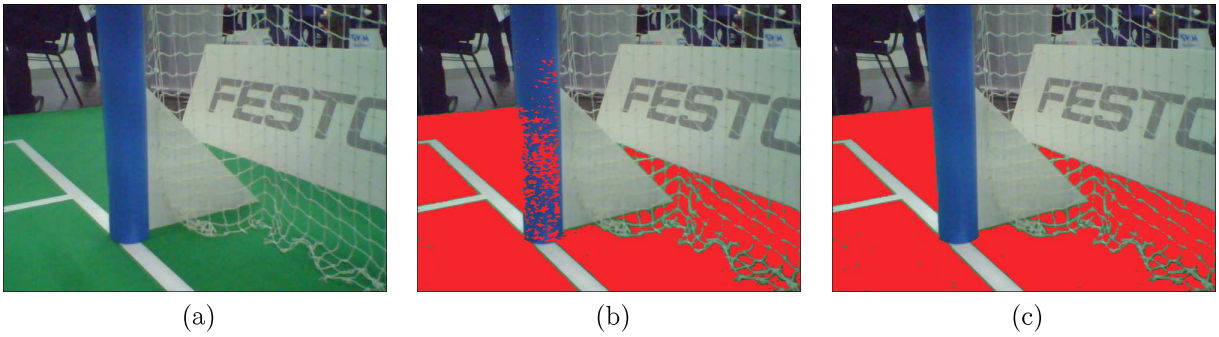


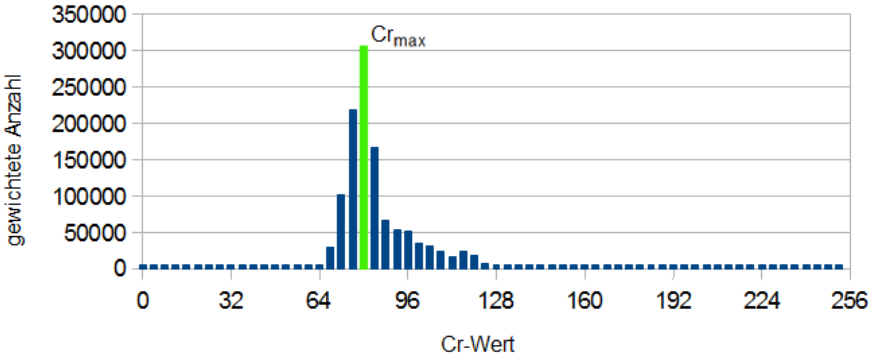
Abb. 3.13: Darstellung der Erkennungsleistung für die quadratische Histogrammgewichtung nach 3.5 in Abhängigkeit von der Rastergröße s_{Raster}.

Diese quadratische Gewichtung des Histogramms wurde erneut auf das Testbild in Abbildung 3.14a angewandt. Im dazugehörigen Histogramm zeigt sich nunmehr ein einziges relevantes Maximum. Die größtenteils korrekte Markierung des Spielfeldes in Abbildung 3.14b verdeutlicht, dass Cr_{max} hinreichend genau dem gewünschten $\bar{Cr}$-Wert entspricht.

In diesem Beispiel wird ebenfalls gezeigt, dass eine Zuordnung der Pixel zur Feldfarbe nicht allein durch die Cr-Komponente getroffen werden sollte, da durchaus andersfarbige Objekte mit ähnlichem Cr-Wert vorkommen können, die nun möglicherweise falsch klassifiziert würden. Als Beispiel hierzu sei der in Abbildung 3.14b vorkommende blaue Torpfosten erwähnt, der irrtümlich teilweise als Spielfeld erkannt wurde.



(a) (b) (c)



(d)

Abb. 3.14: Korrektes Ergebnis (b) der Feldfarbenbestimmung für Cr durch quadratische Gewichtung des Histogramms (d) und weitere Verbesserung (c) durch Hinzunahme der übrigen Komponenten Cb und Y. Alle als Feldfarbe klassifizierten Pixel wurden rot markiert.

Abhilfe schafft hierbei die Hinzunahme weiterer Farbinformationen aus dem Cb- und Y-Kanal des Bildes. Die dominierenden Farbwerte Cb_{max} und Y_{max} dieser Kanäle können ebenso durch Maximumbestimmung ihrer Histogramme errechnet werden. Zu deren Erzeugung werden jedoch nur die Werte von Pixeln (x, y) verwendet, für die bereits $|Cr_{xy} - Cr_{max}| < T_{Cr}$ gilt. Durch diese gezielte Selektion wird im Voraus bereits ein Großteil der Hintergrundpixel ausgeschlossen, so dass eine negative Beeinflussung der Werte Cb_{max} und Y_{max} vermieden werden kann. Deren Berechnung kann aus diesem Grunde im Gegensatz zu Cr_{max} ungewichtet erfolgen. Zur Klassifikation eines Pixels werden zwei weitere Schwellenwerte T_{Cb} und T_Y benötigt, die den maximalen Abstand zu Cb_{max} und Y_{max} definieren. Ein $YCbCr$-Tripel wird schließlich als feldzugehörig klassifiziert, wenn

$$|Cr_{xy} - Cr_{max}| < T_{Cr} \wedge |Cb_{xy} - Cb_{max}| < T_{Cb} \wedge |Y_{xy} - Y_{max}| < T_Y$$

gilt. Da die ermittelte Werte der Spielfeldfarbe $(Y_{max}, Cb_{max}, Cr_{max})$ auch in anderen Ab-

schnitten dieser Arbeit Verwendung finden, werden sie im Weiteren mit $(Y_{field}, Cb_{field}, Cr_{field})$ bezeichnet. Die Abbildung 3.14c zeigt die detektierten feldfarbenen Bildpunkte für die Schwellenwerte $T_{Cr} := 19$, $T_{Cb} := 12$ und $T_Y := 65$. Diese Werte wurden anhand von empirischen Untersuchungen festgelegt. Für den Schwellenwert T_{Cr} wird im nachfolgenden Abschnitt ein Überblick über das Vorgehen bei einer solchen Untersuchung gegeben.

3.4.4 Schwellenwertbestimmung

Gesucht ist ein im Mittel optimaler Schwellenwert T_{Cr}, durch welchen zum einen möglichst viele Bildpunkte korrekt als Spielfeld klassifiziert werden, zum anderen aber möglichst wenig Hintergrundpunkte falsch klassifiziert werden. Die Auswirkungen auf die Spielfelderkennung bei einem ungünstig gewählten Schwellenwert T_{Cr} sind in nachfolgenden Abbildungen dargestellt:

(a) (b)

Abb. 3.15: Bei $T_{Cr} := 4$ werden zu wenige Punkte korrekt klassifiziert (a). Ein zu hoher Wert von $T_{Cr} := 40$ führt zu Fehlklassifikationen vieler Hintergrundpixel (b) .

Hilfreich ist es nun, eine Häufigkeitsverteilung für alle möglichen Differenzen $d := (Cr_{xy} - Cr_{field})$ zu berechnen und in Form des Histogramms 3.16 darzustellen.

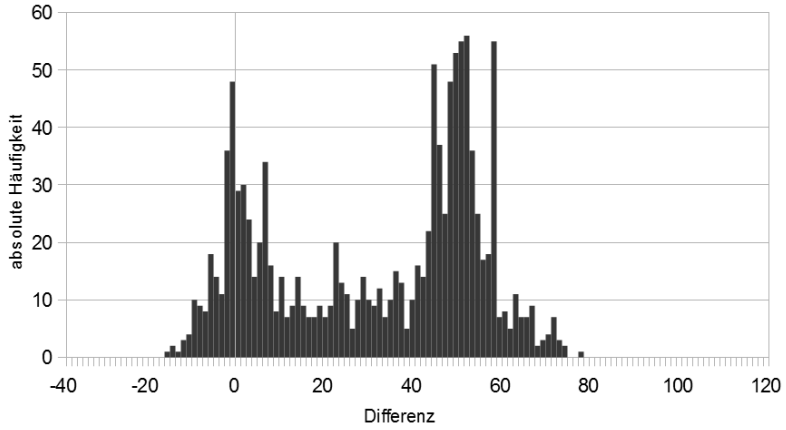


Abb. 3.16: Häufigkeitsverteilung für die Differenzen $Cr_{xy} - Cr_{field}$.

Hierbei fällt auf, dass diese Differenzen offensichtlich in zwei Klassen eingeteilt werden können, deren Verteilungsfunktionen einander überlappen. So lassen sich die zugehörigen feldfarbenen Pixelwerte hauptsächlich der linken Klasse und die Farben des Hintergrundes größtenteils der rechten Klasse zuordnen. Da es sich bei dieser Beobachtung möglicherweise nur um einen Ausnahmefall für speziell dieses Bild handelt, wurde für 598 Bilder aus der Testdatenbank - jedes, auf dem das Spielfeld zu sehen ist - das folgende gemittelte Histogramm[1] berechnet. Dieses bekräftigt die Allgemeingültigkeit obiger Aussage.

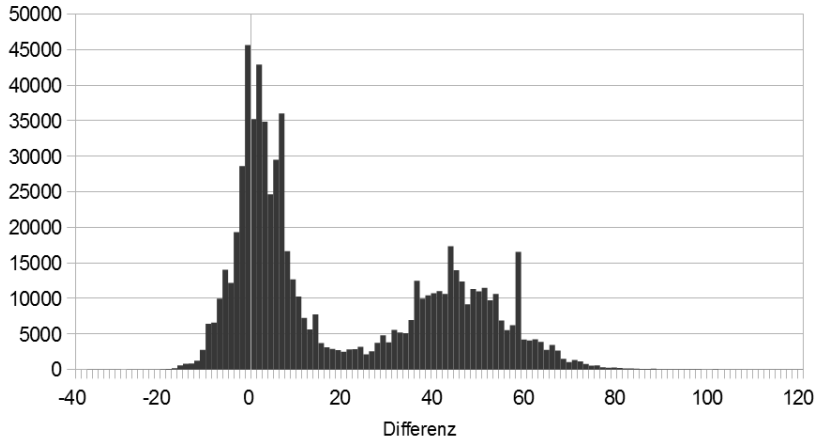


Abb. 3.17: Gemittelte Häufigkeitsverteilung für die Differenzen $Cr_{xy} - Cr_{field}$ über 598 Bilder der Datenbank.

[1]Es sei erwähnt, dass die einzelnen Spitzen innerhalb des gemittelten Histogramms durch häuftig in den Bildern wiederkehrende großflächige Objekte zustande kommt, die zur mittleren Feldfarbe oft den gleichen Farbabstand haben, so dass sie in ihrer Gesamtheit für jene lokalen Maxima verantwortlich sind.

Zur Ermittlung eines Schwellenwertes T_{Cr} zur bestmöglichen Trennung dieser beiden Klassen wurde das Otsu-Schwellwertverfahren angewendet. Dieses 1979 entwickelte Verfahren [Ots79] verwendet zur Suche nach dem günstigsten Schwellenwert die Varianz als statistisches Hilfsmittel: Errechnet wird ein Wert, bei dem die Streuung innerhalb von zwei durch den Schwellenwert separierten Klassen möglichst klein, zwischen diesen Klassen aber gleichzeitig möglichst groß ist. Die Anwendung dieses Verfahrens auf das gemittelte Histogramm ergab für T_{Cr} einen Wert von 19. Dieser wird als Standardwert für alle weiteren Experimente und Anwendungen des Algorithmus genutzt.

Anstelle der einmaligen Berechnung des Otsu-Schwellenwertes über das gemittelte Histogramm wäre auch eine für jedes Einzelbild erfolgende Neuberechnung denkbar. Dagegen spricht jedoch sowohl der CPU-Zeit-Aufwand dieser Berechnung als auch das Vorkommen von Einzelbildern, bei denen das Verfahren nach Otsu keinen brauchbaren Wert für T_{Cr} liefert. Würde beispielsweise das Bild ausschließlich Pixel der Spielfeldfarbe beinhalten, existierte im entsprechenden Histogramm die für die Berechnung nötige „Hintergrundklasse" nicht.

3.5 Linien- und Feldranderkennung

Die Erkennung der Spielfeldlinien und der Spielfeldbegrenzungen ist Voraussetzung für eine im Spielverlauf kontinuierlich erfolgende Schätzung der Roboterposition auf der Feldfläche - auch Selbstlokalisierung genannt - die jedoch nicht mehr Teil dieser Arbeit ist. Im Folgenden werden die für jene visuelle Erkennungsleistung erforderlichen Kriterien vorgestellt.

3.5.1 Erkennungsmerkmale der Spielfeldlinien

Aus dem Regelwerk der Standard-Plattform-Liga [Rob10e] lassen sich folgende Vorgaben zur Farbe der Spiellinien als auch zur Linienbreite entnehmen: „The lines on the field are white.", „All robot-visible lines on the soccer field [...] are 50 mm in width."

Somit ist die Linienbreite $d_{line} = 50mm$. Zusätzlich werden die Maße und Positionen der Linien durch folgende Grafik festgelegt:

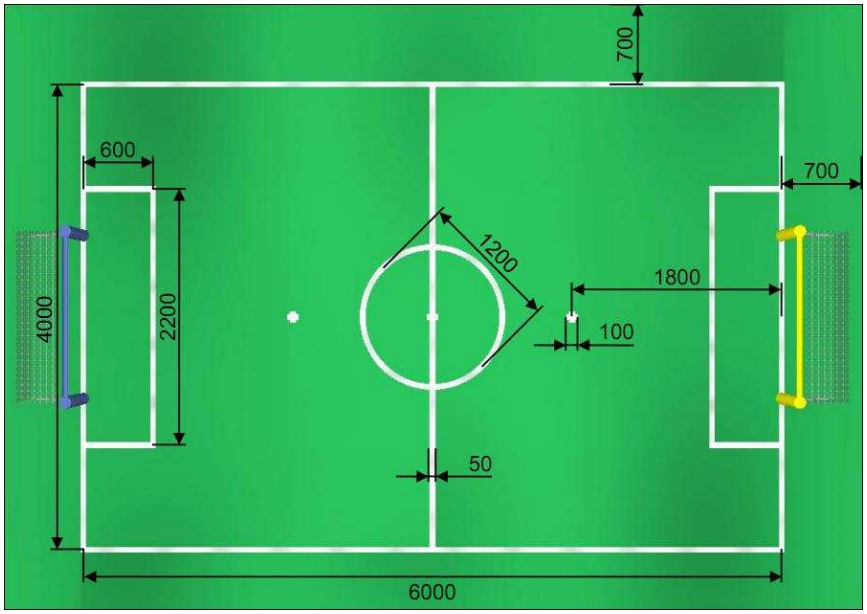
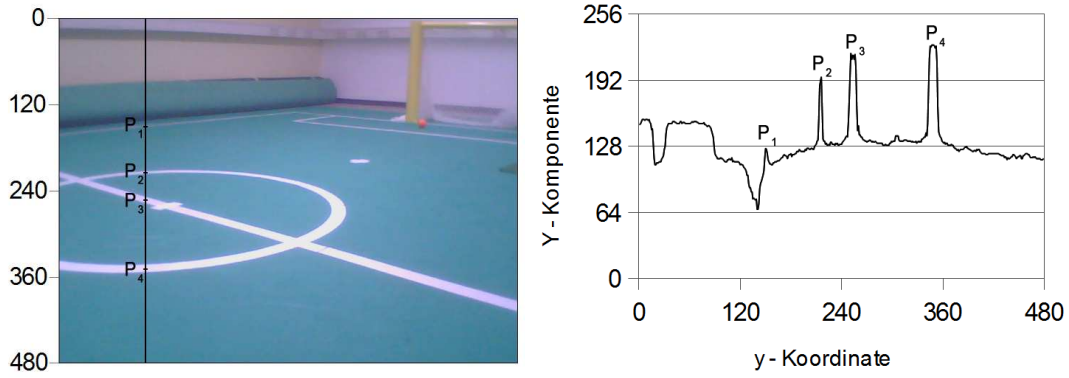


Abb. 3.18: Maße der Spielfeldlinien

Anhand dieser Informationen und durch die Analyse von realen Situationen mittels der Testbilddatenbank können nun entsprechende Eigenschaften der Spielfeldlinien im Kamerabild festgestellt werden.

Farbdifferenzen

Eines der wichtigsten Erkennungsmerkmale ist der Helligkeitsunterschied der weißen Linien zu ihrer unmittelbaren Umgebung, der im Y-Kanal des Kamerabildes ermittelt wird. Anhand der Feldgeometrie ist davon auszugehen, dass es sich bei der angrenzenden Umgebung größtenteils um die grüne Spielfeldfläche handelt, deren Erkennung im Abschnitt 3.4 beschrieben ist. Abbildung 3.19b zeigt ein Helligkeitsprofil eines Querschnitts durch das Bild 3.19a. Für die vier an den Stellen P_1 bis P_4 geschnittenen Linien zeigt das zugehörige Intensitätsdiagramm einen Anstieg der entsprechenden Helligkeitswerte. Es fällt jedoch auf, dass die hintere Linie an der Stelle P_1 wesentlich dunkler als die drei verbleibenden Linien und zusätzlich auch dunkler als die Feldbereiche zwischen P_2, P_3 und P_4 ist. Hauptursache ist die Unschärfe des Bildes, die aufgrund der geringen Linienbreite zu einer Vermischung der Grauwerte von Linien- und Spielfeldpixeln führt. Von bedeutender Relevanz ist, dass dadurch die Anwendung eines globalen Schwellenwertes zur Unterscheidung der Linienhelligkeit von der Spielfeldhelligkeit ausgeschlossen ist. Infolge jener Bildunschärfe ergibt sich des Weiteren eine Verfälschung der Cb- und Cr-Werte, so dass jene dünnen Linien häufig durch die bereits beschriebene Feldfarbenerkennung fälschlicherweise dem Spielfeld zugeordnet werden. Ein demgegenüber robustes Erkennungsmerkmal sind daher die Kanten im Y-Kanal.



(a) (b)

Abb. 3.19: Für das Bild (a) wurde das Helligkeitsprofil (b) erzeugt

Größe

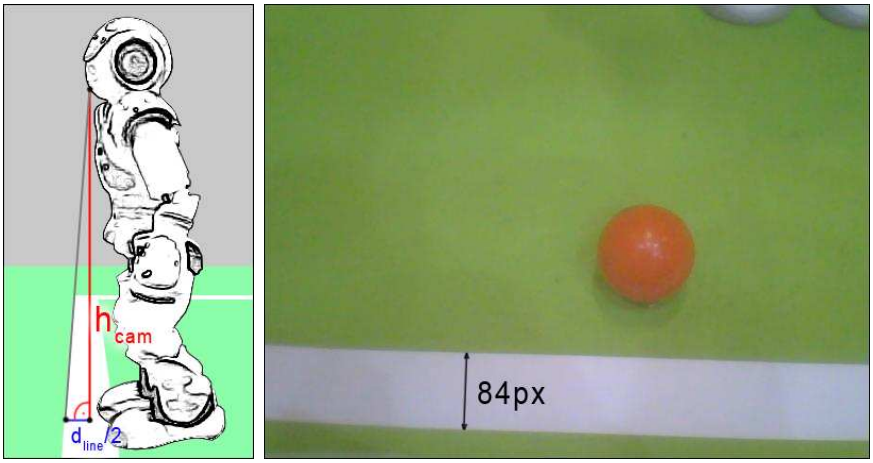
Mittels der Informationen aus der Abbildung 3.18 lassen sich minimale und maximale Breite einer Linie im Kamerabild schätzen, die insbesondere für die in Abschnitt 3.5.3

beschriebenen Algorithmen zur Liniensegmenterkennung als wichtiges Ausschlusskriterium relevant sind.

Für diese Schätzung wird angenommen, dass der Roboter aufrecht steht und somit die Kamerahöhe h_{cam} etwa $450mm$ beträgt sowie dass jene Kamera einen horizontalen Öffnungswinkel von $\alpha_{fov} = 46,4°$ [AR11] hat. Da sich eine möglichst nah an der Kamera des Roboters befindliche Linie im optimalen Fall direkt unter ihm befinden müsste, lässt sich über ein rechtwinkliges Dreieck (siehe Abbildung 3.20a) eine grobe Abschätzung der maximal möglichen Breite l_{max} einer im Kamerabild abgebildeten Linie durch

$$l_{max} := \frac{w}{\alpha_{fov}} * 2 * \tan^{-1}(\frac{d_{line}}{2 * h_{cam}}) = \frac{640px}{46,4°} * 2 * \tan^{-1}(\frac{50mm}{2 * 450mm}) \approx 88px$$

berechnen. Zur Untermauerung dieser Schätzung wurde im Testbildmaterial das Bild mit der breitesten Linie gesucht. Sie befand sich wenige cm vor den Füßen des Roboters und war $84px$ breit (siehe Abbildung 3.20b), was obige Schätzung bestärkt.



(a) (b)

Abb. 3.20: Schätzung der maximalen Linienbreite: rechnerisch (a) und am Beispielbild gemessen (b)

Für die Abschätzung der minimalen Breite einer Linie ist die Analyse des Testbildmaterials sinnvoller als eine analytische Berechnung, da hier hauptsächlich durch Unschärfe der Kameraoptik relevante Verfälschungen der Linienbreite zustande kommen. 44462 in der Testbilddatenbank gespeicherte Liniensegmente wurden automatisiert untersucht und die Anzahl der Vorkommen für gerundete Linienbreiten im Kamerabild bestimmt. Für Werte von 1px bis 5px ist das Ergebnis in folgender Tabelle dargestellt:

gerundete Liniensegmentbreite in Pixel	Anzahl Segmente
1	30
2	3731
3	12439
4	2552
5	1976

Tab. 3.5: Häufigkeiten für Liniensegmente mit unterschiedlicher Länge

Daraus lässt sich ableiten, dass die zu erwartende minimale Breite einer Linie etwa zwei Pixel, in seltenen Fällen jedoch auch ein Pixel betragen kann.

Es ist zudem für die robuste Linienerkennung die Vorgabe einer Linien-Mindestlänge zur Minimierung der Falsch-Positiv-Rate empfehlenswert.

Form

Spielfeldlinien lassen sich bezüglich ihrer Form grundsätzlich in zwei Kategorien einteilen: Es gibt gerade Linienabschnitte und jene, die zum Mittelkreis gehören und einen gekrümmten Kurvenverlauf aufweisen.

Rotation

Prinzipiell ist die Verlaufsrichtung einer Linie im Kamerabild beliebig. Es fällt jedoch auf, dass das Testbildmaterial anscheinend deutlich mehr horizontal als vertikal gelegene Spielfeldlinien beinhaltet. Zur Analyse dieser Vermutung wurde für 3645 im Datenbestand markierte Linien die Häufigkeitsverteilung ihrer jeweiligen Winkel untersucht. Das Histogramm 3.21 zeigt eine deutlich höhere Anzahl an nahezu parallel zur x-Achse (nahezu 0°) des Bildes verlaufenden Linien gegenüber einer geringeren Anzahl an senkrechten Linienverläufen (nahezu ±90°). Konkret liegen etwa 93% aller Linienwinkel zwischen $-45°$ und 45°. Der Grund hierfür ist die perspektivische Verzerrung des Bildes. Je geringer die Höhe der Kamera des Roboters über dem Spielfeld ist, desto deutlicher stellt sich dieser Effekt dar.

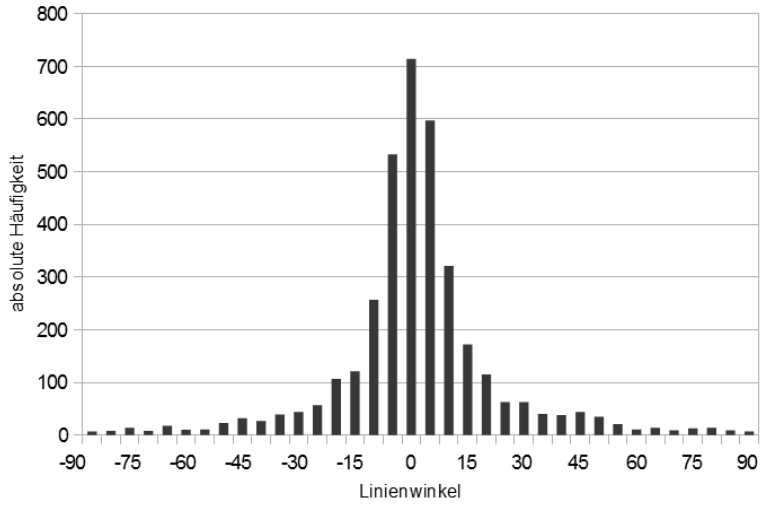


Abb. 3.21: Häufigkeitsverteilung der Linienausrichtungen

Besonderheit

Die beschriebenen Erkennungsmerkmale der Spielfeldlinien treffen jedoch auch auf bestimmte Bereiche der Tornetze zu (siehe Abbildung 3.22). Ein mögliches Unterscheidungskriterium ist in solchen Fällen die Analyse der Anzahl vieler benachbarter dünner „Linienabschnitte". Häufungen von vielen potentiellen Liniensegmenten zeugen aufgrund der begrenzten Anzahl der Spielfeldlinien (siehe Linienmodell in Abbildung 3.18) eher für das Vorhandensein eines Tor-Netzes im Kamerabild.

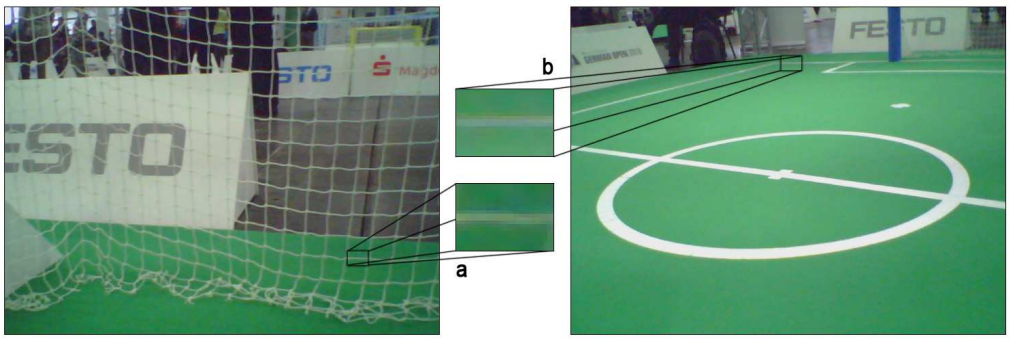


Abb. 3.22: Vergleich zwischen einem Netz (a) eines Tores und einer Spielfeldlinie (b)

3.5.2 Erkennungsmerkmale der Spielfeldbegrenzung

Form

Die Ursprungsform des Spielfeldes ist ein 6×4 m^2 großes Rechteck. Die unverzerrte Spielfeldbegrenzung kann deshalb durch ein konvexes Polygon, bestehend aus vier senkrecht bzw. parallel zueinander stehenden Kanten, dargestellt werden. Da durch die projektive Abbildung einer solchen Form deren Konvexität nicht beeinflusst wird, ist die in einem Kamerabild gegebene Spielfeldform die Schnittmenge zweier konvexer Polygone, wobei die rechteckige Form des Kamerabildes dabei das zweite Polygon darstellt (siehe Abbildung 3.23). Dies impliziert, dass jene Schnittmenge ebenso konvex sein muss [Tou85]. Abbildung 3.24a zeigt ein solches durch die Punkte P_1 bis P_5 definiertes Polygon zur Modellierung des sichtbaren Spielfeldes.

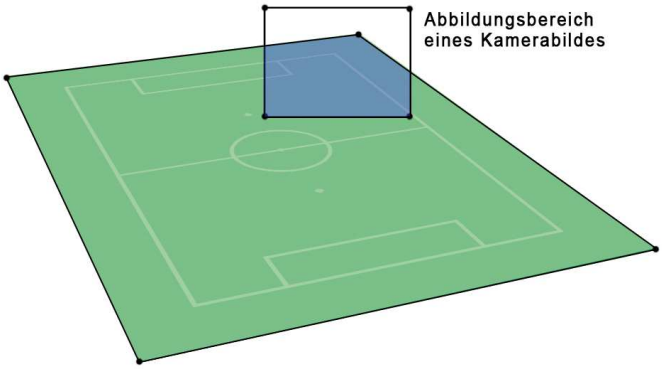
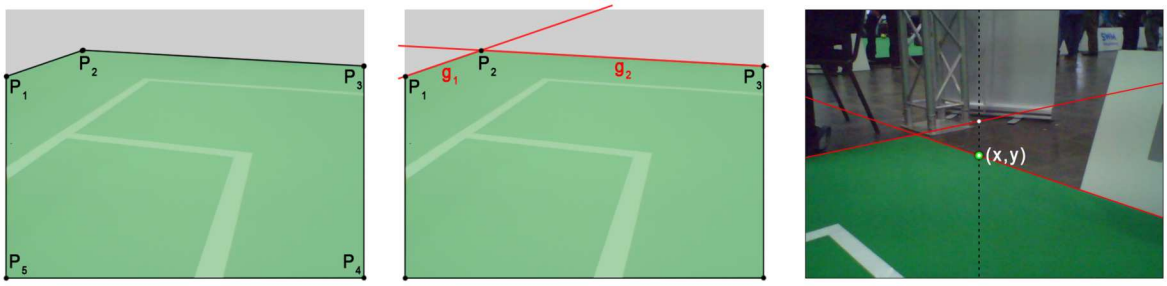


Abb. 3.23: Darstellung der Schnittmenge (blau) zwischen Spielfeld (grün) und dem Abbildungsbereich des Kamerabildes

In [LHB+09a] wird für den Anwendungsfall Roboterfußball ausschließlich die obere Begrenzung der konvexe Hülle des Spielfeldes bestimmt, da entsprechend häufig der untere Teil der Spielfeldform durch den Bildrand abgegrenzt wird. So wären in diesem Fall ausschließlich die Punkte P_1 bis P_3 zur Modellierung der Spielfeldgrenzen von Interesse. In [RHH10] werden Spielfeldgrenzen ausschließlich durch ein bis zwei Geraden approximiert (Abbildung 3.24b), was aufgrund des geringen Öffnungswinkels der Roboterkamera eine praxistaugliche Vereinfachung darstellt. Durch die Darstellung des Spielfeldrandes mittels zweier Geraden ergeben sich an einer Stelle x zwei Möglichkeiten für die y-Koordinate. In diesem Fall wird der weiter unten im Bild liegende y-Wert ausgewählt, wie folgende Abbildung 3.24c zeigt.



(a) (b) (c)

Abb. 3.24: Definition der Feldgrenze als konvexer Polygonzug (a) oder durch zwei Geraden (b) und (c)

Rotation

Ebenso wie die Feldlinien verlaufen auch die Spielfeldbegrenzungslinien eher horizontal als vertikal. Zudem lässt sich für die Geometrie jener Begrenzungslinien - anders als bei den Spielfeldlinien - nahezu ausschließen, dass sie exakt vertikal im Kamerabild liegen, sofern der Roboter selbst nicht auf einer solchen Linie steht. Im Testbildmaterial liegen die Winkel aller abgespeicherten Spielfeldbegrenzungslinien beispielsweise zwischen -70 und 70 Grad.

Farbdifferenzen

In Abschnitt 3.5.1 wurde bereits gezeigt, dass sich Helligkeitsveränderungen im Y-Kanal als robustes Erkennungsmerkmal für Spielfeldlinien erweisen. Es wird davon ausgegangen, dass auch in der unmittelbaren Umgebung der Spielfeldgrenze mindestens kleinere Helligkeitsänderungen vorkommen.

Aus der Geometrie des Spielfeldes lässt sich zudem erschließen, dass sich unmittelbar unterhalb der Spielfeldkante im Kamerabild mehrere feldfarbene Pixel befinden müssen. Da sich die Spielfeldgrenze aus den Kanten zwischen der grünfarbenen Feldfläche und dem Hintergrund ergibt, liegt es nahe, zusätzlich auch den Cr-Kanal zur Erkennung zu berücksichtigen, denn - wie in Abschnitt 3.4.1 bereits gezeigt - weisen 99,7% der Pixel außerhalb des Spielfeldes im Vergleich zur Spielfeldfarbe einen höheren Cr-Farbwert auf.

3.5.3 Kantenerkennung und Bereichsklassifizierung

Sowohl bei den Spielfeldlinien als auch bei der Feldgrenze sind lokale Helligkeitsveränderungen an ihren Objektkanten zu erwarten, die im Folgenden durch Anwendung einer Kantendetektion im Y-Kanal erkannt werden sollen. Zur zeiteffizienten Berechnung kommt hierbei die in den Grundlagen dieser Arbeit bereits beschriebene Scanlinetechnik zur Anwendung (siehe Abschnitt 2.4.2), bei welcher nur einzelne wenige - beispielsweise jede achte oder sechzehnte - Zeilen und/oder Spalten des Bildes analysiert werden.

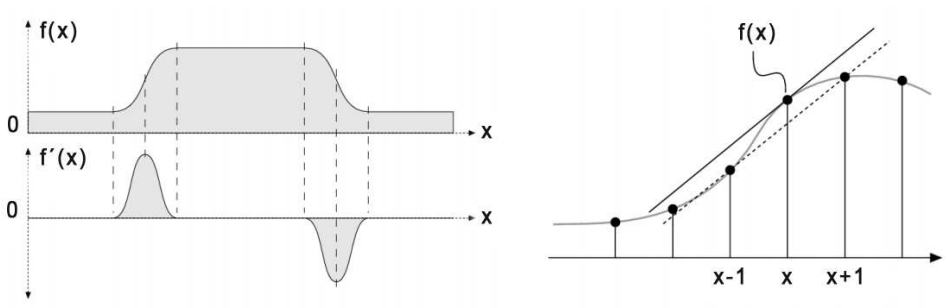
Die Kanten auf einer Scanline sind Stellen, an denen sich die Helligkeiten benachbarter Pixel stark verändern. Entlang einer Bildzeile oder -spalte wird die Intensitätsänderung $df(x)$, bezogen auf die Bilddistanz dx, durch die Ableitung der Bildintensität

$$f'(x) := \frac{df(x)}{dx}$$

dargestellt (siehe auch Abbildung 3.25a). Für den zu behandelnden diskreten Fall wird eine Approximation dieser Ableitung benötigt. Zur Näherung kann der symmetrischer Gradient $g(x)$ an der Stelle x genutzt werden, wobei

$$g(x) := \frac{1}{2}(f(x+1) - f(x-1)) \tag{3.6}$$

ist (siehe Abbildung 3.25b).



(a) (b)

Abb. 3.25: Ableitung der Funktion f (a), symmetrischer Gradient (b) [BB05]

Angewendet auf eine ausgewählte Scanline eines Beispielbildes entstehen für die genäherten Ableitungen der Pixelhelligkeiten die in Abbildung 3.26 dargestellten, durch Streckenzüge miteinander verbundenen, Werte.

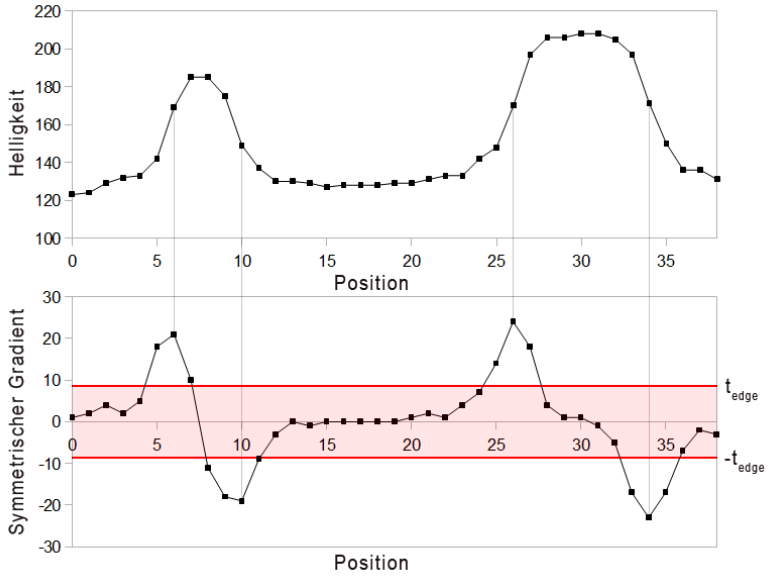


Abb. 3.26: Bestimmung von Kanten einer Funktion (a) durch Berechnung von Extremwerten der zugehörigen Ableitung (b).

Von Interesse für die weitere Verarbeitung sind lokale Minima und Maxima jenes symmetrischen Gradienten, mittels derer sich die Positionen relevanter Kanten auf der Scanline ermitteln lassen. Zu diesem Zweck wurde der Algorithmus 3.5 entwickelt, der effizient die Positionen x von lokalen Extrema des zur Scanline gehörigen symmetrischen Gradienten bestimmt. Durch einen Schwellenwert t_{edge} kann dabei eine Mindesthöhe des Betrags dieses Gradienten mit

$$\left|(f(x+1) - f(x-1))\right| > 2 * t_{edge}$$

vorgegeben werden, so dass innerhalb von annäherend homogenen Bereichen der Scanline keine Kanten detektiert werden. Im Folgenden wird die Funktionsweise jenes Algorithmus näher erläutert:

Die Berechnung des Gradienten g für eine Position x auf der Scanline erfolgt in Zeile 12 durch Differenzbildung des aktuellen Intensitätswertes f_x und des im letzten Interationsdurchgang verwendeten Wertes f_{last}. Für den Fall, dass g außerhalb des Intervalls $[-t_{edge}, t_{edge}]$ liegt, wird als Wert der Variablen g_{min} bzw. g_{max} der jeweils niedrigste bzw. höchste Funktionswert g abgespeichert. Dabei wird immer dann, wenn ein neues Minimum oder Maximum detektiert wird, die dazugehörige Position x als Wert der Variable x_{peak} abgelegt. Da sich die korrekte Position des Gradienten g zwischen f_x und f_{last} befindet, wird beim Setzen

von x_{peak} in Zeile 19 und 27 die Position x um 1 verringert. Sobald der Gradient g wieder innerhalb des Intervalls $[-t_{edge}, t_{edge}]$ liegt oder dieses überspringt, wird zur Resultatmenge *edges* die Position x_{peak} des letzten gefundenen Extrempunktes hinzugefügt.

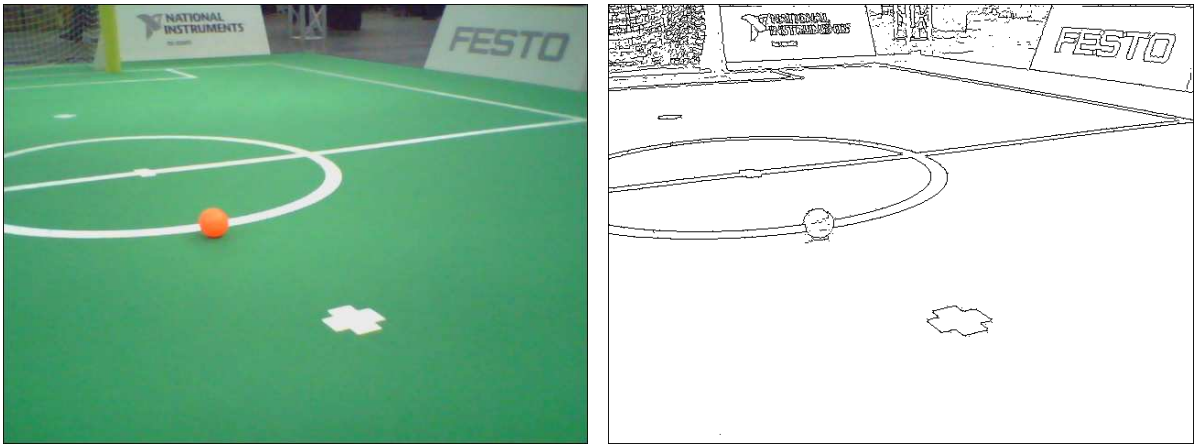
Algorithm 3.5 findEdges
scanline := Array der Intensitätswerte entlang einer Bildzeile oder -spalte

```
1  findEdges(scanline, t_edge)
2  {
3      edges := {}
4      t_edge := t_edge * 2
5      g_max := -t_edge
6      g_min := t_edge
7      x_peak := 0
8      f_last := scanline_0
9      for(x := 2; x < sizeof(scanline); x := x + 2)
10     {
11         f_x := scanline_x
12         g := f_x - f_last
13         if(g > g_max)
14         {
15             if(g_min < -t_edge)
16                 edges := edges ∪ x_peak
17             g_max := g
18             g_min := t_edge
19             x_peak := x - 1
20         }
21         if(g < g_min)
22         {
23             if(g_max > t_edge)
24                 edges := edges ∪ x_peak
25             g_min := g
26             g_max := -t_edge
27             x_peak := x - 1
28         }
29         f_last := f_x
30     }
31     return edges
32 }
```

Aus Effizienzgründen wird nur jeder zweite Pixelwert der Scanline verwendet, da in Abschnitt 3.5.1 gezeigt wurde, dass nur selten eine Linie weniger als zwei Pixel breit ist. Dieses Vorgehen führt zu einer Verdopplung der Verarbeitungsgeschwindigkeit, jedoch auch zu einer Halbierung der Genauigkeit der Kantenpositionen. Es kann jedoch für alle Elemente in der Kantenmenge *edges* eine lokale Suche mit dem Radius 1 zur pixelgenauen Bestimmung ihrer Position auf der Scanline durchgeführt werden, wodurch jener Nachteil der Positionsungenauigkeit aufgehoben wird. Der Geschwindigkeitsvorteil bleibt jedoch

größtenteils erhalten, da die zu erwartende Anzahl der Kanten auf einer Scanline wesentlich geringer als die Anzahl der Pixel auf der Scanline ist.

Die Abbildung 3.27b zeigt die Anwendung dieses Algorithmus auf jede Zeile und Spalte des Beispielbildes 3.27a. Die Elemente der Menge *edges* wurden durch schwarze Punkte visualisiert.



(a) (b)

Abb. 3.27: Anwendung des Algorithmus 3.5 auf das Bild (a) führt zum Kantenbild (b), $t_{edge} := 4$

Bevor diese Menge der Kantenpositionen zur Erkennung des Spielfeldrandes und zur Erkennung der Spielfeldlinien genutzt wird, ist noch ein weiterer Verarbeitungsschritt nötig, nämlich die Klassifizierung der farblich homogenen Bereiche auf den Scanlines zwischen den ermittelten Kanten hinsichtlich ihrer drei YCbCr-Farbkomponenten.

Zuerst werden für jeden dieser eindimensionalen Bereiche repräsentative Farbwerte (Y_M, Cb_M, Cr_M) errechnet. Hier wäre die Berechnung des arithmetischen Mittels über alle Pixel für jeden Farbkanal möglich. Dieses Vorgehen erfordert jedoch - zu Ungunsten der Zeiteffizienz - einen Zugriff auf alle Pixel innerhalb der Scanline. Aus diesem Grunde werden maximal fünf Pixelpositionen pro Bereich ausgewählt, für die von den dazugehörigen (Y, Cb, Cr)-Tripeln für jede Komponente separat der Medianwert[2] bestimmt wird. Das so entstehende Tripel (Y_M, Cb_M, Cr_M) repräsentiert nun die Farbe des jeweiligen Bereiches. Da bei der im Voraus erfolgten Feldfarbenerkennung (Abschnitt 3.4) bereits die dominierende Spielfeldfarbe in Form des Tripels $(Y_{field}, Cb_{field}, Cr_{field})$ bestimmt wurde, kann nun eine Klassifizierung zur Feldfarbenzugehörigkeit des jeweiligen Bereiches mit der Bedingung

[2] Zur effizienten Bestimmung des Medians aus fünf Werten wird der im Anhang B.3 abgebildete Algorithmus verwendet, der maximal lediglich sechs Vergleiche zur Lösung des Problems benötigt.

$$|Cr_M - Cr_{field}| < T_{Cr} \ \wedge \ |Cb_M - Cb_{field}| < T_{Cb} \ \wedge \ |Y_M - Y_{field}| < T_Y$$

unter Verwendung der drei aus Abschnitt 3.4.4 bereits bekannten Schwellenwerte T_{Cr}, T_{Cb} und T_Y vollzogen werden. Es sei erwähnt, dass die Reihenfolge der drei Vergleiche durchaus den Berechnungsaufwand beeinflusst. Denn schon wenn eine der Bedingungen nicht erfüllt ist, müssen die übrigen Bedingungen nicht weiter untersucht werden, da durch die Und-Verknüpfungen die Gesamtaussage nicht mehr wahr werden kann. Die Überprüfung des Farbwertes im Cr-Kanal geschieht deshalb als erstes, weil sie die schärfste Bedingung darstellt, da sich hiermit schon ein Großteil der Bereiche als nicht feldfarben aussortieren lässt. Demgegenüber wird der Farbwertvergleich im Y-Kanal zuletzt ausgeführt, da es sich hierbei um eine eher schwache Bedingung handelt. Mit dieser Optimierung lässt sich beispielsweise bei der Klassifizierung der Scanlinebereiche im Testbildmaterial etwa 10-15% Geschwindigkeitszuwachs erreichen. Ähnliche Optimierungen wurden auch an anderen Stellen in dieser Arbeit durchgeführt, jedoch ohne immer im Detail darauf einzugehen.

Anschließend werden entlang der Scanlines potentielle Linienkandidaten gesucht. Als Linienkandidat wird ein zwischen zwei Kantenpositionen liegender Bereich mit folgenden Eigenschaften bezeichnet: Er muss zwischen zwei direkt angrenzenden und bereits als spielfeldfarben klassifizierten Bereichen liegen und seine ermittelte Helligkeit Y_M muss höher als die jeweiligen Helligkeiten der direkt angrenzenden Bereiche sein. Alle Bereiche der Scanline, für die jene Bedingungen zutrifft, werden als „Linienkandidat" klassifiziert sowie anschließend alle bisher weder als „Spielfeld" noch als „Linienkandidat" erkannten Bereiche als „Unbekannt" klassifiziert.

Zusammenfassend zeigt die folgende Abbildung beispielhaft an einer einzigen horizontal verlaufenden Scanline nochmals die Zusammenhänge zwischen Kantenpositionen, Bereichen und Klassifikation dieser Bereiche.

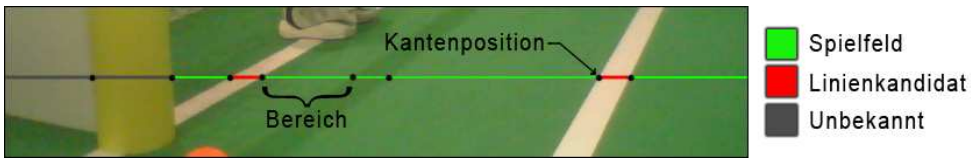
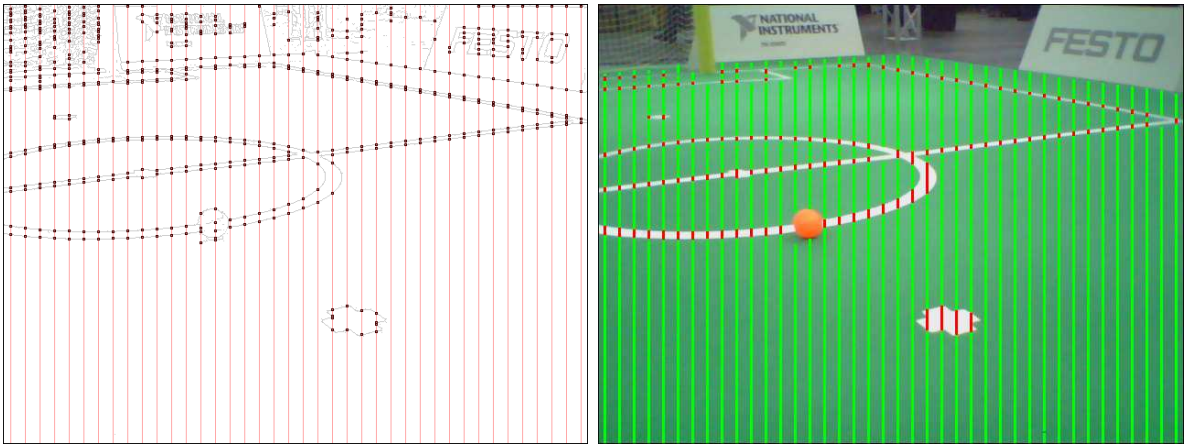


Abb. 3.28: Bereiche, Kantenpositionen und Klassifizierungen einer Scanline

Um eine echtzeitfähige Scanlineanalyse zu realisieren, werden lediglich jede sechzehnte Zeile und Spalte berücksichtigt, was zu einer Verminderung auf $\frac{1}{8}$ der nötigen Speicherzugriffe im Kamerabild führt. Abbildung 3.29a zeigt die dabei genutzten Scanlines sowie die darauf

ermittelten Kantenpunkte. In der nebenstehenden Abbildung 3.29b sind die entsprechend zugehörigen Bereiche, die entweder als Spielfeldboden oder als Linienkandidat klassifiziert wurden, dargestellt.

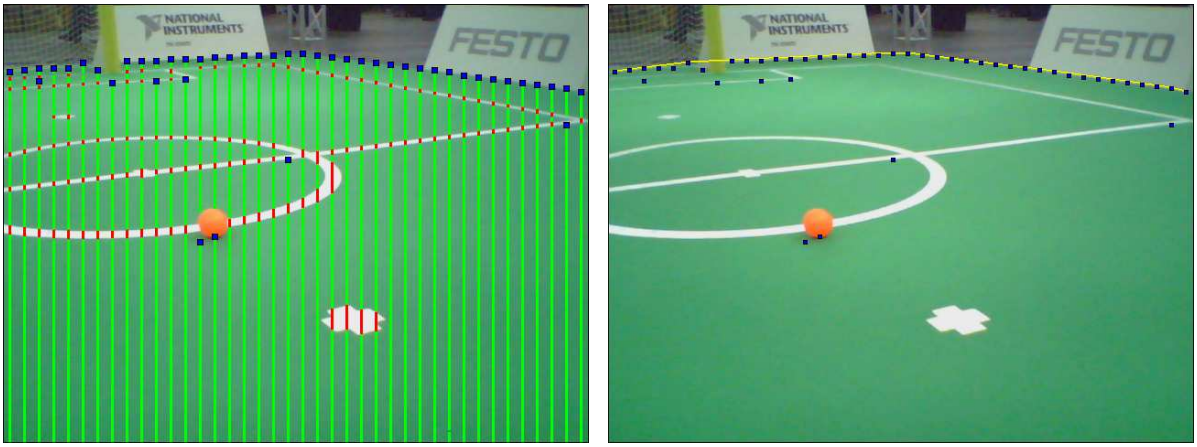


(a) (b)

Abb. 3.29: Der Übersichtlichkeit halber sind nur vertikale Scanlinebereiche visualisiert

3.5.4 Spielfeldranderkennung

Zur Spielfeldranderkennung werden die im vorherigen Abschnitt erläuterten Ergebnisse der Bereichsklassifizierung nun weiterverarbeitet. Allein die entlang der vertikalen Scanlines ermittelten Bereiche und Kantenpositionen liefern hierbei hinreichend Informationen zur Bestimmung der Spielfeldgrenze. Jeder Kantenposition lassen sich nun zwei direkt angrenzende Bereiche zuordnen, die durch diese Kante voneinander abgegrenzt werden. Da es für einen Bereich drei mögliche Klassen gibt (Spielfeld, Linienkandidat oder Unbekannt), ergeben sich für zwei auf der Scanline unmittelbar übereinander liegende Bereiche neun verschiedene Klassifizierungskombinationen. Genau eine dieser Möglichkeiten wird für die Erkennung der Spielfeldgrenze als Modell verwendet: Ausschließlich wenn der untere dieser beiden Bereiche als spielfeldfarben klassifiziert und der darüber liegende Bereich als „Unbekannt" eingestuft wird, kann nach diesem Modell die dazwischenliegende Kantenposition auf dem Spielfeldrand liegen. Diese Menge an Kantenpositionen wurde in nachfolgender Abbildung 3.30a für das bereits aus dem letzten Abschnitt bekannte Beispielbild visualisiert. Durch Berechnung der oberen Begrenzung der konvexen Hülle dieser Punktemenge lässt sich eine erste Approximation der Feldgrenze, wie in nebenstehender Abbildung 3.30b zu sehen, ermitteln.



(a) (b)

Abb. 3.30: Potentielle Randpunkte des Spielfeldes (blau) und die obere konvexe Hülle dieser Punkte (gelb)

Ein effizientes Verfahren zur Berechnung der konvexen Hülle einer endlichen, planaren Punktmenge $S := \{P_0, ..., P_{n-1}\}$ mit n Elementen ist der Graham Scan [Gra72]. Dieser Algorithmus benötigt zunächst die folgenden zwei Vorverarbeitungsschritte:

1. Startpunktsuche

Zu Beginn wird ein Punkt Q gesucht, der zur konvexen Hülle gehört. Jener wird gewöhnlich mit linearem Rechenaufwand dadurch ermittelt, dass der Punkt aus der Menge S mit der kleinsten Ordinate und - sollte es mehrere davon geben - der mit der kleinsten Abszisse ausgewählt wird. In der Originalveröffentlichung von Graham werden hingegen aus der Menge S drei nicht kollineare Punkte bestimmt und aus diesen ein Dreieck gebildet, dessen Schwerpunkt als Startpunkt Q für das weitere Vorgehen dient.

2. Sortierung

Alle Punkte P_i der Menge S werden nun nach ihrem Winkel zwischen $Q \rightarrow P_i$ und der positiven x-Achse aufsteigend sortiert und in die Liste L_{sorted} abgelegt. Dazu kann die Funktion

$$ccw((x_0, y_0), (x_1, y_1), (x_2, y_2)) := (x_1 - x_0) * (y_2 - y_0) - (x_2 - x_0) * (y_1 - y_0) \qquad (3.7)$$

herangezogen werden, wobei

$$ccw(A,B,C) \quad \begin{cases} < 0 & wenn\ C\ rechts\ von\ \overrightarrow{AB} \\ = 0 & wenn\ C\ auf\ \overrightarrow{AB} \\ > 0 & wenn\ C\ links\ von\ \overrightarrow{AB} \end{cases}$$

gilt. Das Sortieren der Punkte nach Winkeln garantiert, dass bei sukzessiver Verbindung aller benachbarten sowie des ersten und des letzten Punktes ein Polygonzug entsteht, in dem sich keine zwei Kanten kreuzen.

Aufbau der konvexen Hülle

Nach diesen Vorverarbeitungsschritten erfolgt nun der Aufbau der konvexen Hülle. Hierzu wird nacheinander für alle Punkte P_i in der sortierten Liste L_{sorted} geprüft, ob sie die Kriterien für einen Eckpunkt der konvexen Hülle erfüllen und bei Zutreffen jener Bedingung konsekutiv der Liste $H := (P_0, ..., P_m)$ hinzugefügt werden können, wobei sich hierbei m jeweils um 1 erhöhen würde. Diese Überprüfung erfolgt durch die Berechnung des Vorzeichens des Winkels zwischen den letzten beiden bisher verarbeiteten Hüllenpunkten P_{m-1} und P_m sowie dem zu überprüfenden Punkt P_i mittels der Funktion ccw.

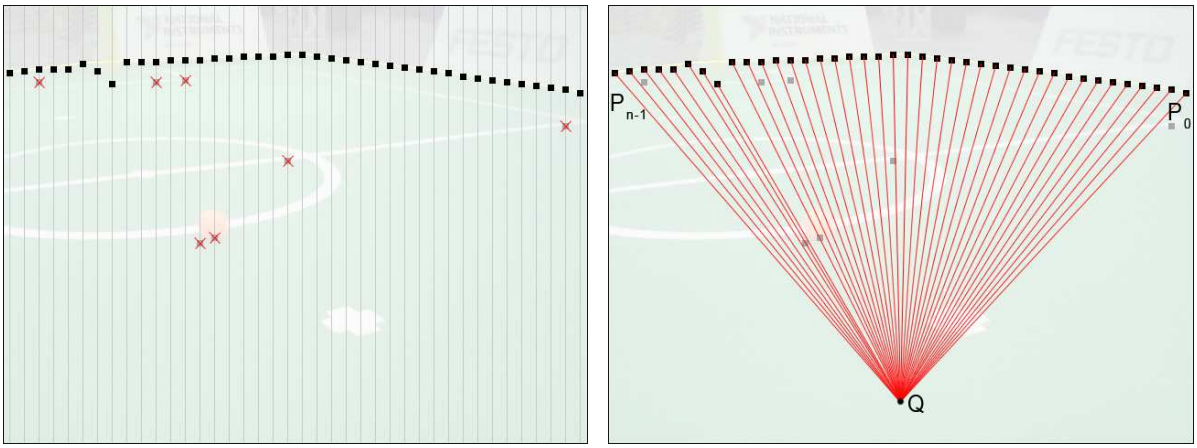
Bei sukzessiver Abarbeitung dieser Punkte nach dem beschriebenen Prinzip wird ein „Abbiegen" entgegen dem Uhrzeigersinn erwartet. Wenn dagegen für einen Punkt P_i die Bedingung $ccw(P_{m-1}, P_m, P_i) \leq 0$ erfüllt ist, so muß der zuletzt bestimmte Punkt P_m aus der Liste H wieder entfernt werden, da bereits ein konvexes Polygon existiert, welches P_m enthält.

Umsetzung zur Bestimmung der oberen konvexen Hülle

Da zur Bestimmung der Spielfeldgrenze nur die obere Begrenzung der konvexen Hülle von S benötigt wird, werden folgende Veränderungen der Vorverarbeitungsschritte vorgenommen:

- Liegen auf einer vertikalen Scanline mehrere potentielle Rand-Kantenpunkte übereinander, wird nur der oberste dieser Punkte in die Menge S aufgenommen, da alle darunterliegenden Punkte nicht mehr zur oberen konvexen Hülle gehören können. Für jede Scanline existiert also nur noch maximal ein Kantenpunkt. (Siehe Abbildung 3.31a)

- Die Bestimmung des Startpunktes (Schritt 1) und die Sortierung der Punktmenge S (Schritt 2) kann entfallen, da die Scanlines - bereits nach ihrer x-Koordinate geordnet - im Speicher liegen und dem entsprechend ebenso alle Kantenpunkte P nach ihrer x-Koordinate geordnet sind und in dieser Reihenfolge in L_{sorted} abgelegt werden können. Dies impliziert die geforderte Sortierung jener Punkte nach ihrem Winkel zu einem unterhalb der konvexen Hülle liegenden Startpunkt Q, sofern dieser weit genug auf der y-Achse entfernt liegt (siehe Abbildung 3.31b). Da lediglich die obere Begrenzung der konvexen Hülle von Interesse ist, kann sich Q beliebig weit unterhalb der Hülle befinden. Eine konkrete Position eines solchen Punktes muss jedoch nicht bestimmt werden, da das Wissen über seine Existenz für die Anwendung eben beschriebener Vereinfachung ausreichend ist. Verbindet man alle benachbarten Punkte miteinander, entsteht durch die Sortierung anhand der x-Koordinate ein sich nicht überkreuzender Kantenzug.



(a) (b)

Abb. 3.31: Reduktion auf einen Kantenpunkt pro Scanline (a) und Visualisierung eines möglichen Punktes Q (b)

Der folgende Pseudocode zeigt eine mögliche Implementierung des Verfahrens, wobei die Liste der übergebenen Punkte so konstruiert ist, dass m nie kleiner 0 werden kann.

Algorithm 3.6 Graham Scan

```
1  grahamScan((P_0, ..., P_{n-1}))
2  {
3      m := 1
4      for(i := 2; i < n; i := i + 1)
5      {
6          while(ccw(P_{m-1}, P_m, P_i) ≤ 0)
7              m := m − 1
8          m := m + 1
9          swap(P_m, P_i)
10     }
11     return (P_0, ..., P_m)
12 }
```

In jedem Iterationsschritt bilden die Punkte P_0 bis P_m die Ecken der aktuelle konvexe Hülle, wobei P_{m-1} und P_m die beiden zuletzt hinzugefügten Punkte darstellen. Bei der Verarbeitung des Punktes P_i wird geprüft, ob der letzte Punkt P_m nun innerhalb der neuen konvexen Hülle liegt (siehe Zeile 6). Trifft dies zu, kann P_m nicht mehr Element dieser aktuellen konvexen Hülle sein und wird „entfernt", indem der Index m des letzten auf der konvexen Hülle liegenden Punktes um eins verringert wird. Erfüllt P_m jedoch weiterhin das Konvexitätskriterium, so wird auch P_i der konvexen Hülle hinzugefügt. Dies erfolgt effizient durch das Austauschen von P_i und einem bereits gelöschten Punkt P_{m+1} mittels der Funktion *swap* (siehe Zeile 9). Die Vertauschung wirkt sich jedoch ausschließlich auf die Koordinaten der Punkte, nicht jedoch auf ihre Indizes aus. Die Ausführung von $H = grahemScan(L_{sorted})$ liefert schließlich eine Liste H sämtlicher $m + 1$ Eckpunkte der konvexen Hülle der Liste L_{sorted}, wobei letztere hierbei durch Verwendung einer Kopie nicht verändert werden soll.

Evaluierung der Feldranderkennung

Zur Untersuchung der Genauigkeit der Spielfeldranderkennung dieses Algorithmus werden dessen Resultate mit den in der Testbilddatenbank hinterlegten Ground-Truth-Daten abgeglichen. Für jedes Bild sind in dieser Datenbank ein oder zwei Geraden abgespeichert, die die Spielfeldgrenze repräsentieren. Für jede x-Koordinate in einem Bild lässt sich damit eine zugehörige y-Koordinate - im Weiteren als $\hat{y_x}$ bezeichnet - bestimmen, so dass der Punkt $(x,\hat{y_x})$ auf dem Spielfeldrand liegt.

Die Evaluierung der Werte wird nun folgendermaßen durchgeführt: Nach der Bestimmung der oberen konvexen Hülle mittels Graham Scan kann zu jeder x-Koordinate im Bild durch

lineare Interpolation zwischen zwei Eckpunkten der Hülle auch eine y-Koordinate, die im Idealfall genau auf dem Spielfeldrand liegt, berechnet werden. Sie wird im Weiteren als $\tilde{y_x}$ bezeichnet. Von Interesse ist nun der Abstand $d_{x,j} := |\hat{y_{x,j}} - \tilde{y_{x,j}}|$ an der Stelle x in einem Bild j, der möglichst gering ausfallen sollte, so dass der geschätzte Wert $\tilde{y_x}$ möglichst nahe an dem realen Wert $\hat{y_x}$ liegt. Für alle Spalten sämtlicher Bilder aus der Testbilddatenbank kann nun der mittlere Abstand $\bar{d}$ durch Bildung des arithmetischen Mittels

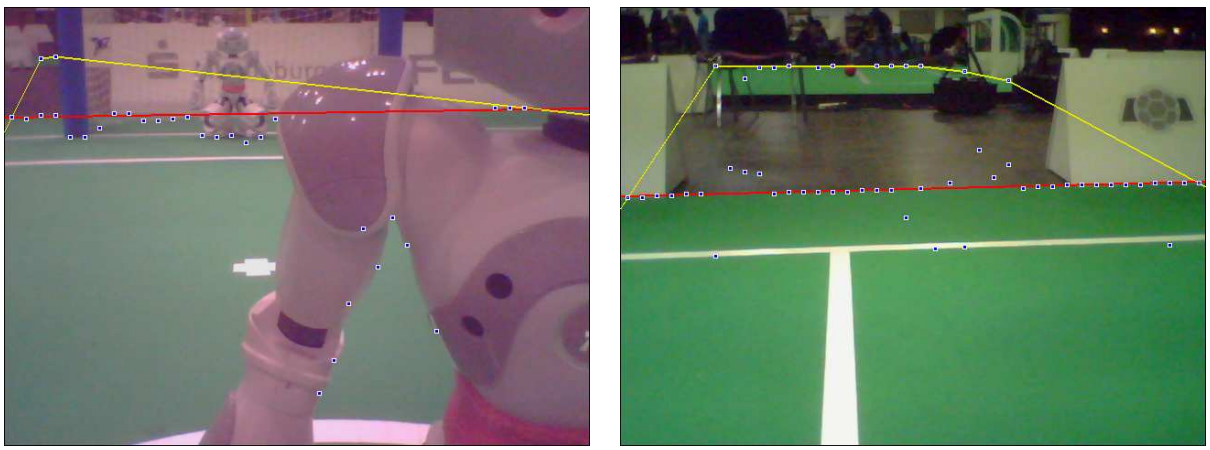
$$\bar{d} := \frac{1}{n_B} \sum_{j=1}^{n_B} \bar{d_j}$$

errechnet werden, wobei

$$\bar{d_j} := \frac{1}{w} \sum_{x=0}^{w-1} d_{x,j}$$

als das arithmetische Mittel aller Spalten in einem einzigen Bild j definiert ist und w die konstante Breite (die Spaltenanzahl) eines Bildes ist. Angewendet auf den vorgestellten Algorithmus 3.6 beträgt der Fehlerwert $\bar{d} = 24,25$. Durchschnittlich liegt bei diesem Verfahren also die geschätzte Feldgrenze etwa 24 Pixel von der realen Feldgrenze entfernt.

Zur Optimierung der Feldranderkennung empfiehlt es sich zunächst, das Resultat des Algorithmus 3.6 für diejenigen Bilder aus der Testbilddatenbank auszuwerten, für die die Einzelbewertungen $\bar{d_j}$ besonders hohe Fehlerwerte aufweisen. Zwei dieser Bilder, für welche eine besonders ungünstige Bewertung von $\bar{d_1} = 31,2$ bzw. $\bar{d_2} = 95,6$ errechnet wurde, sind in nachfolgenden Abbildungen 3.32a bzw. 3.32b dargestellt.



(a) (b)

Abb. 3.32: Fehler bei der Berechnung der Feldgrenze durch die obere konvexe Hülle

Sie verdeutlichen beispielhaft die Problematik in der Anwendbarkeit jenes Algorithmus, nämlich die Anfälligkeit gegenüber Ausreißern, insbesondere gegenüber inkorrekten Kantenpositionen, die oberhalb der realen Spielfeldgrenze liegen. Daher haben vor allem benachbarte Spielfelder aufgrund ihrer Farbähnlichkeit zum eigenen Feld oft einen störenden Einfluss auf die korrekte Bestimmung der eigenen Spielfeldgrenze.

Verbesserung der Feldranderkennung

Bisher wurden lediglich die obereren Kantenpunkte der Scanlines zur Berechnung der oberen Begrenzung der konvexen Hülle H verwendet. Jedoch wird durch die vorangegangenen Beispiele in Abbildung 3.32 deutlich, dass auch die übrigen Kantenpositionen wichtige Informationen zum Verlauf der eigenen Spielfeldgrenze liefern.

Bestandteil vorliegender Arbeit ist nun, jene Ausreißerpunkte Schritt für Schritt zu eliminieren um folglich aus den übrigen Kantenpunkten eine neue und möglicherweise „bessere" obere Begrenzung der konvexen Hülle zu berechnen. Dazu wurde das folgende iterative Verfahren entwickelt:

In jedem Iterationsschritt $k \in \mathbb{N}$ wird genau der Eckpunkt $P_i \in H_k$ der aktuellen konvexen Hülle H_k bestimmt, welcher den größten Einfluss auf deren Verlauf hat. Hierzu wird derjenige Punkt ausgewählt, für welchen der horizontale Abstand $e(i) := P_{i+1,x} - P_{i-1,x}$ der beiden Nachbarpunkte P_{i-1} und P_{i+1} maximal wird, denn je größer dieser Abstand ist, desto mehr Einfluss hat P_i auf den Verlauf der konvexen Hülle. Gibt es mehrere Punkte, für die die Bewertung e identisch ist, wird der am weitesten oben im Bild liegende ausgewählt. Dieser Punkt P_i wird als potentieller Ausreißer angenommen und aus der Liste L_{sorted} gelöscht, sowie anschließend eine neue obere konvexe Hülle H_{k+1} aus L_{sorted} berechnet. Vor dem ersten Iterationsschritt sei $H_1 := H$.

Dieser iterative Löschvorgang ist in Abbildung 3.33 für $1 \leq k \leq 20$ visualisiert. Alle berechneten konvexen Hüllen H_k wurden hierzu in unterschiedlichen Farben von gelb ($k = 1$) bis blau ($k = 20$) in das Beispielbild eingezeichnet.

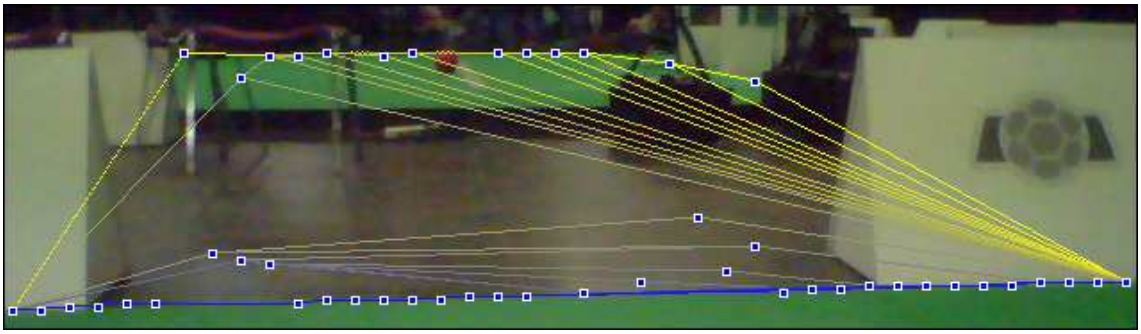


Abb. 3.33: Visualisierung des iterativen Löschvorgangs.

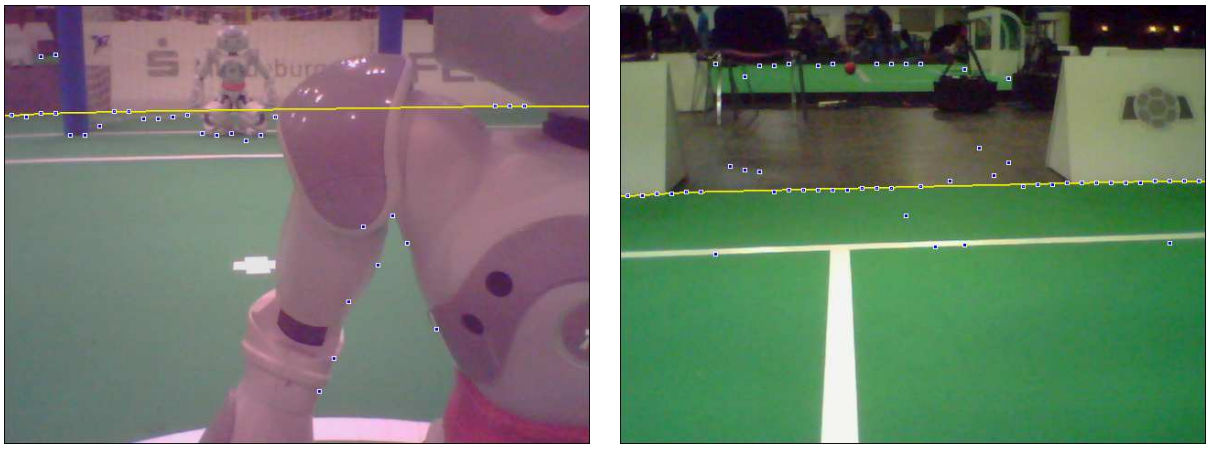
Um aus dieser Vielzahl an Möglichkeiten nun diejenige auszuwählen, die am ehesten für den Verlauf der eigenen Feldgrenze infrage kommt, wird in jeder Iteration k eine Bewertung b_k der aktuellen konvexen Hülle H_k ermittelt. Diese Bewertung ergibt sich aus der Anzahl der Kantenpunkte, die auf oder in der Nähe von H_k liegen. Die Berechnung von b_k erfolgt mit folgender modifizierten Variante des Graham Scans:

Algorithm 3.7 Bewertung mittels modifiziertem Graham Scan

```
1  getRating((P_0, ..., P_{n-1}), t_y)
2  {
3      m := 1
4      for(i := 2; i < n; i := i + 1)
5      {
6          while(ccw(shift(P_{m-1}, t_y), P_m, shift(P_i, t_y)) ≤ 0)
7              m := m - 1
8          m := m + 1
9          swap(P_m, P_i)
10     }
11     b_k := m + 1
12     return b_k
13 }
14
15 shift(P, t_y)
16 {
17     P'_x := P_x
18     P'_y := P_y + t_y
19     return P'
20 }
```

Durch Addition eines Schwellenwertes t_y zur y-Koordinate der Eckpunkte P_{m-1} und P_i wird die konvexe Hülle temporär um einige Pixel abgesenkt, so dass auch Kantenpunkte, die sich maximal t_y Pixel unterhalb der oberen Begrenzung der konvexen Hülle befinden, in die Bewertung eingehen.

Im Beispielbild 3.33 konnten so für H_1 (gelb dargestellt) 14 zugehörige Punkte ermittelt werden, woraus sich eine Bewertung von $b_1 = 14$ ergibt. Die beste Bewertung wird jedoch für $b_{20} = 29$ erreicht, da hier besonders viele Kantenpunkte der durch H_{20} beschriebenen Feldgrenze (blau) zugeordnet werden können. Das Resultat des entwickelten Algorithmus ist die durch b_k am höchsten bewertete konvexe Hülle H_k. Die Ergebnisse jener optimierten Feldgrenzenbestimmung sind für die eingehend gezeigten Beispielbilder in den folgenden Abbildungen dargestellt.



(a) (b)

Abb. 3.34: Resultate der optimierten Feldgrenzenerkennung für $t_y := 4$. (Die Werte der Feldgrenze an der linken und rechten Seite des Bildes, welche nicht Teil der oberen konvexen Hülle sind, wurden anhand der am nächsten liegenden Datenpunkte extrapoliert.)

Ergänzend sei erwähnt, dass bei der Implementierung dieser optimierten Feldgrenzenerkennung die alternative Bewertungsfunktion $b'_k := b_k - \frac{k}{2}$ verwendet wurde und zu besseren Ergebnissen führte: Konkret ergab die abschließende Evaluierung über allen 600 Testbildern einen Fehlerwert von $\bar{d} = 9,90$, eine deutliche Verbesserung gegenüber dem urpsrünglichen Wert von $\bar{d} = 24,25$.

3.5.5 Linienerkennung

Als eine Linienerkennung wird eine Gruppierung von den durch Kantenerkennung und Bereichsklassifikation (Abschnitt 3.5.3) ermittelten Endpunkten der potentiellen Linienbereiche bezeichnet. Jeder im Kamerabild sichtbaren Spielfeldlinie sollen also, wie in Abbildung 3.35 dargestellt, im Idealfall jeweils zwei Gruppen von Kantenpunkten zugeord-

net werden. Bei diesen Kantenpunkten handelt es sich genau um diejenigen Punkte, die einen zuvor als linienfarben klassifizierten Bereich einschließen.

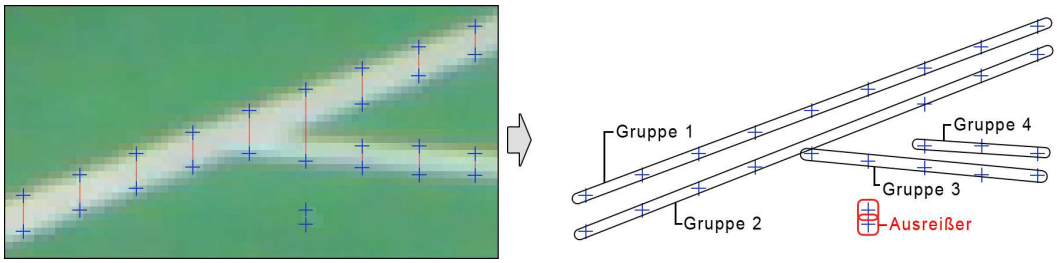


Abb. 3.35: Zusammenfassung einzelner Kantenpunkte entlang einer Linienkante

Durch diesen Gruppierungsvorgang können zum einen falsch klassifizierte Kantenpunkte gefiltert werden, zum anderen sind derartige Gruppen von Punkten wesentlich einfacher in ihrer Weiterverarbeitung zu handhaben, als zahlreiche einzelne und zusammenhanglose Kantenpositionen.

Diese Thematik ist Gegenstand zahlreicher Veröffentlichungen, insbesondere auf den Gebieten der Fußball-Analyse-Systeme im TV-Bereich [SL09, GJMR10, HPV04, NYC10b], der Automobilbranche zur automatisierten Fahrbahnstreifenerkennung [HKP10, Aly08] und der automatisierten Luft- und Satellitenbildauswertung [GFBG05].

Als etabliertes Standardverfahren zur robusten Erkennung parametrisierbarer geometrischer Objekte, wie Geraden oder Kreisen, aus einer Menge von Kantenpunkten sei die Hough-Transformation [Hou62] erwähnt. Jene ermöglicht die Transformation aller im Ursprungsbild auf einer Geraden liegenden Kantenpositionen auf eine einzige, in einem Parameterraum liegende Position, die den Verlauf jener Geraden im Originalbild repräsentiert und so eine Gruppierung von Linienelementen gestattet. Angesichts diverser Nachteile kommt dieses Verfahren für die hier benötigten Echtzeitanforderungen jedoch nicht infrage. Aufgrund der schlechten Laufzeiteigenschaften und der Tatsache, dass durch die Hough-Transformation lediglich Geraden, jedoch keine Anfangs- und Endpunkte von Strecken berücksichtigt werden, wurde als Alternative in dieser Arbeit der im Folgenden beschriebene Algorithmus entwickelt, der vergleichsweise laufzeiteffizient arbeitet, Anfangs- und Endpunkte von Strecken erfasst und zusätzlich Gradienteninformationen der Linienkanten einbezieht, was eine robuste Linienerkennung ermöglicht.

Gradienten

Durch die Bestimmung der Richtungen von Linienkanten mittels Berechnung der zugehörigen zweidimensionalen Gradienten wird deren Aussagekraft bezüglich der im Bild gegebenen Linienstruktur wesentlich gesteigert. Der Gradient einer zweidimensionalen Grauwertfunktion f an dem Punkt (x, y) ist definiert als der Vektor

$$\nabla f(x,y) := \begin{bmatrix} G_X(x,y) \\ G_Y(x,y) \end{bmatrix} \tag{3.8}$$

der partiellen Ableitung des Bildes in x-Richtung

$$G_X(x,y) := \frac{\partial f(x,y)}{\partial x}$$

sowie der in y-Richtung

$$G_Y(x,y) := \frac{\partial f(x,y)}{\partial y}\,.$$

Dieser Vektor hat die Eigenschaft, an der Stelle (x, y) in die Richtung der größten Grauwertänderung von f zu zeigen. Im Fall einer Spielfeldlinie würde ein auf deren Kante berechneter Gradient orthogonal zu dieser verlaufen und zum Linieninneren zeigen.

Analog zum eindimensionalen Fall (Abschnitt 3.5.3) werden für die zugrundeliegenden diskreten Intensitätsdaten Approximationen der Ableitungen G_X und G_Y benötigt. Hierfür eignen sich verschiedene zweidimensionale Filter, die das gegebene Grauwertbild an der Stelle (x, y) falten und dadurch die partiellen Ableitungen G_X und G_Y näherungsweise durch Grauwertdifferenzen in einer lokalen Region bestimmen.

Eine nützliche Eigenschaft dieser Filter ist deren Separierbarkeit: Das bedeutet, dass sich die Auswirkung eines zweidimensionalen Filters auch durch die Hintereinanderausführung zweier eindimensionaler Filter erreichen lässt, so dass folglich die zweidimensionale Faltung auf zwei eindimensionale reduziert werden kann. Es lassen sich die in dieser Arbeit untersuchten Filtermatrizen zur approximativen Bestimmung von G_X und G_Y in einen Ableitungs- und einen Glättungsanteil separieren. Der Ableitungsteil entspricht dem im Abschnitt 3.5.3 bereits vorgestellten symmetrischen Gradienten in einer Richtung, also der Grauwertdifferenz zweier Nachbarbildpunkte. Da jedoch diese Differenzbildung aufgrund ihrer Hochpasseigenschaft zu einer Verstärkung des Rauschens führt, nutzt man zusätzlich

einen Glättungsfilter, der dieser Verstärkung entgegenwirkt. Es sind viele Filter mit beliebigen Filtergrößen zur Bestimmung des Gradienten möglich. Im Rahmen dieser Arbeit werden exemplarisch drei der bekanntesten Filtermatrizen aufgezeigt, die mit Glättungs- und Ableitungsanteilen in nachfolgender Tabelle aufgelistet sind.

Operator	Matrix		Glättung	Ableitung
$Prewitt_x$	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$=$	$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$	$* \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$
$Sobel_x$	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$=$	$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$	$* \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$
$Scharr_x$	$\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$	$=$	$\begin{bmatrix} 3 \\ 10 \\ 3 \end{bmatrix}$	$* \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$

Tab. 3.6: Abgebildet sind drei Faltungsmatrizen zur Detektion der Gradientenrichtung entlang der x-Achse, wobei sich die Matrizen für die Approximation der Ableitung entlang der y-Achse durch Transponierung der dargestellten Matrizen ergeben.

Der Vergleich dieser Operatoren und die Wahl des für das vorliegende Bildmaterial am ehesten geeigneten erfolgt anhand der zu erwartenden Güte ihrer Richtungsgenauigkeit. Die Richtungsgenauigkeit gibt an, wie nahe die vom jeweiligen Operator ermittelte Richtung - der größte Anstieg des Gradienten - der Realität kommt. Die präzise Ermittlung dieser Richtungen spielt, wie später noch gezeigt, eine wesentliche Rolle für die Gruppierung gleichausgerichteter Kanten zu zusammengehörigen Linienobjekten.

In [Sch00] wurden hinsichtlich der Richtungsgenauigkeit bereits umfassende Untersuchungen durchgeführt, wobei der in jener Quelle vorgestellte Scharr-Operator die besten Ergebnisse im Vergleich zu dem Sobel- und Prewitt-Operator lieferte. Da jedoch möglicherweise die Ergebnisse der Operatoren bei der Anwendung auf das in dieser Arbeit verwendete Bildmaterial anders ausfallen - zumal aufgrund der perspektivischen Verzerrung bestimmte Gradientenrichtungen wahrscheinlicher als andere sind (siehe Abschnitt 3.5.1) - erfolgte vom Autor unter Verwendung der Testbilddatenbank abermals ein Vergleich für die drei gegebenen Operatoren:

Dazu werden für relevante Kantenpunkte die durch die zu vergleichenden Operatoren berechneten Gradientenrichtungen mit gegebenen Ground-Truth-Daten verglichen. Als

„relevant" werden solche Kanten bezeichnet, die sich einer in der Datenbank abgespeicherten Spielfeldlinie zuordnen lassen, so dass aus diesen vorgegebenen Linien die idealen Richtungen der Gradienten (die Ground-Truth-Daten) abgeleitet werden können.

Als Vergleichswert dient die Winkelabweichung $\Delta\beta$ zwischen dem zur Kante gehörenden Gradienten g und den zugehörigen Vektor $\hat{g}$ der Ground-Truth-Daten, wobei unter Verwendung des Skalarproduktes $g \cdot \hat{g}$ die gesuchte Winkelabweichung

$$\Delta\beta := \arccos \frac{g \cdot \hat{g}}{|g| * |\hat{g}|}$$

ist. Als Gütemaß wird daraus über alle n_B Bilder der Datenbank der Fehler

$$e := \sqrt{\frac{1}{n_B} * \sum_{i=1}^{n_B} \left(\frac{1}{n_{G,i}} \sum_{j=1}^{n_{G,i}} ((\Delta\beta_{ij})^2) \right)} \tag{3.9}$$

bestimmt, wobei $n_{G,i}$ die Anzahl der in der Testbilddatenbank abgespeicherten Gradienten mit dem Index j für ein bestimmes Bild mit dem Index i angibt.

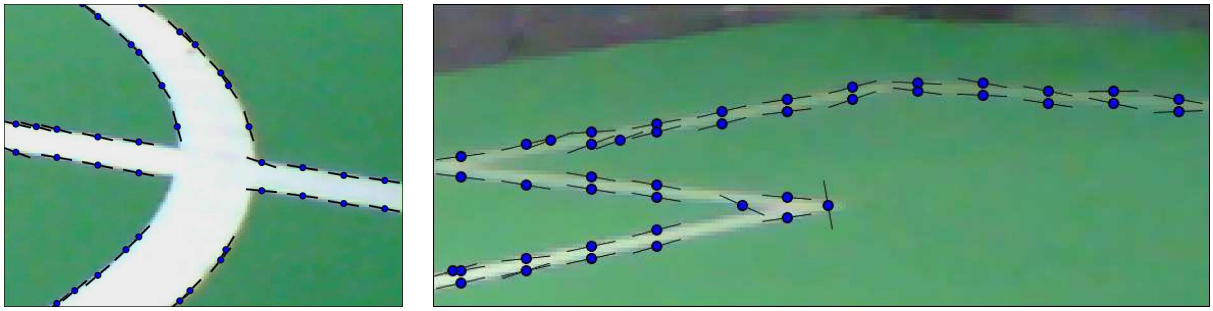
Wie Tabelle 3.7 zeigt, eignet sich - anders als die Untersuchungen in [Sch00] vermuten lassen - der Sobeloperator für das bestehende Datenmaterial am besten zur Bestimmung der Gradientenrichtung. Zwei Gründe dafür könnten die ungleichmäßige Häufigkeitsverteilung der Gradientenwinkel oder Aliasing-Effekte bei sehr dünnen Linien sein.

Operator	Fehler e
Prewitt	0.05256
Sobel	0.03684
Scharr	0.04645

Tab. 3.7: Bewertung der Richtungsgenauigkeit verschiedener Kantendetektionsfilter

Zur Weiterverarbeitung wird nun ausschließlich der normierte[3] und mittels Sobeloperator geschätzte Gradientenvektor $g_n := \frac{1}{|g|} * g$ zu jedem Kantenpunkt abgespeichert. In den Abbildungen 3.36a und 3.36b sind für jeden Kantenpunkt auf Grundlage dieser Vektoren g_n die orthogonal verlaufenden Linienrichtungen dargestellt.

[3]Zur Normierung von Vektoren ist der im Anhang B.1 dargestellte Algorithmus hilfreich, der zu einer gegebenen Zahl (Beispielsweise dem quadratischen Betrag eines Vektors) zeiteffizient das Reziproke ihrer Wurzel berechnen kann.



(a) (b)

Abb. 3.36: Für jeden Kantenpunkt (blau) wird der dazugehörige Gradientenvektor als eine zu ihm orthogonal liegende schwarze Linie eingezeichnet.

Insbesondere anhand der Abbildung 3.36b erkennt man, dass die Ermittlung der Gradientenrichtung durch den Sobel-Operator bei sehr dünnen Linien mit wenig Kontrast teils ungenau wird, da hier zunehmend das Bildrauschen einen störenden Einfluss auf die Vektorberechnung hat. Um dennoch die Richtungsgenauigkeit weiter zu steigern, könnte man größere Filtermatrizen nutzen. In [Sch00] wurde gezeigt, dass mit 5x5 Matrizen Verbesserungen möglich sind, da durch den entsprechend größeren Glättungsanteil das Bildrauschen effektiver gemindert werden kann. Je größer jedoch die Filtermatrix, desto höher das Risiko, dass Grauwerte benachbarter Kanten mit einem ganz anderen Winkel in die Filterumgebung hineinfallen und einen Störeinfluss auf die Gradientenberechnung ausüben. Deshalb wurde in dieser Arbeit eine einfache Methode zur Steigerung der Richtungsgenauigkeit entwickelt, die keine weiteren Pixelzugriffe erfordert, sondern ausschließlich bereits ermittelte Koordinaten und Gradientenrichtungen analysiert, filtert und schließlich optimiert.

Steigerung der Richtungsgenauigkeit durch Liniensegmentbildung

Anhand der Eigenschaften von Spielfeldlinien kann erschlossen werden, dass zwei nahe beieinander liegende Kantenpunkte, die ähnliche Gradientenrichtungen aufweisen, wahrscheinlich zur selben Linienkante gehören. Nach einer Suche solcher Kantenpunktpaare lassen sich die Strecken zwischen den jeweiligen Punkten eines Paares, die im Idealfall annähernd parallel zur Richtung der Linienkante verlaufen, ermitteln (Abbildung 3.37a).

Die Idee besteht nun darin, dass der Richtungsvektor dieser Strecken zur Optimierung der Richtungsgenauigkeit der orthogonal verlaufenden Gradienten g beider Kantenpunkte - in Abbildung 3.37a als gestrichelte Linien dargestellt - verwendet werden könnte.

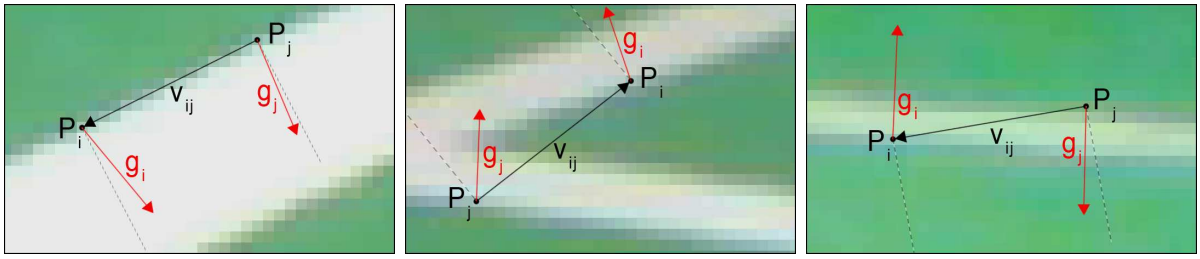


Abb. 3.37: Gradientenrichtungen ausgewählter Kantenpunktpaare

Für eine Suche nach solchen Kantenpunktpaaren wird ein Ähnlichkeitsmaß definiert, das angibt, inwiefern zwei Kantenpunkte auf ein und dieselbe Gerade passen. Sind P_i und P_j zwei beliebige Kantenpositionen sowie g_i und g_j die dazugehörigen, mittels Sobel-Operator bestimmten Gradienten, so verlaufen im Idealfall g_i und g_j orthogonal zum Vektor $v_{ij} := P_i - P_j$. Je weiter die Richtungen beide Gradienten von diesem Ideal abweichen, desto weniger passen die entsprechenden Kantenpunkte auf ein und dieselbe Linie (siehe Abbildung 3.37). Zur Ermittlung der Ähnlichkeit zweier Kantenpunkten P_i und P_j wird ein Ähnlichkeitskoeffizient $s(i,j)$ definiert durch

$$s(i,j) := \begin{cases} 1 - (\frac{|v_{ij} \cdot g_i|}{|v_{ij}| * |g_i|} + \frac{|v_{ij} \cdot g_j|}{|v_{ij}| * |g_j|})/2 & wenn \;\; g_i \cdot g_j > 0 \\ 0 & sonst \end{cases}$$

unter der Bedingung

$$d_{min} \leq |v_{ij}| \leq d_{max} \;\; ,$$

so dass der Abstand der Kantenpunkte P_i und P_j immer größer als ein vorgegebener Schwellenwert d_{min} und kleiner als d_{max} ist. Der Wert des Koeffizient $s(i,j)$ liegt im Intervall $[0,1]$. Verlaufen die beiden Gradienten g_i und g_j orthogonal zu v_{ij}, erreicht der Koeffizient den höchsten Wert von 1, da dann jeweils $|v \cdot g| = 0$ ist. Die Bedingung $g_i \cdot g_j > 0$ garantiert, dass ausschließlich Gradienten, die einen Winkel kleiner 90 Grad einschließen, einen Ähnlichkeitskoeffizienten größer 0 haben können. Verliefen sie jedoch senkrecht bzw. entgegengesetzt, wäre deren Zugehörigkeit zu ein und der selben Linienkante höchst unwahrscheinlich. Abbildung 3.37a zeigt den Normalfall: Hier verlaufen die Gradienten g_i und g_j etwa orthogonal zum Vektor v_{ij}. Warum es nicht ausreicht, zur Bestimmung der Ähnlichkeit lediglich die Parallelität zwischen g_i und g_j zu prüfen, zeigt Abbildung 3.37b:

Offensichtlich ist der Winkel zwischen g_i und g_j zwar relativ klein, dennoch verlaufen beide Gradienten nicht orthogonal zu v_{ij}, woraus eine Minderung des Ähnlichkeitskoeffizienten resultiert. Eben jene Verringerung des Koeffizientenwertes ist hier gewollt, da es sich in diesem Fall um zwei Kanten unterschiedlicher Linien handelt. Ein weiterer wichtiger Fall ist in Abbildung 3.37c dargestellt, in dem beide Gradienten parallel verlaufen und zur gleichen Spielfeldlinie, nicht jedoch zur gleichen, sondern zur entgegengesetzten Linienkante gehören. Die Bedingung $g_i \cdot g_j > 0$ verhindert hier, dass der Ähnlichkeitskoeffizient $s(i,j)$ für dieses Punktpaar $(P_i,\ P_j)$ größer 0 sein kann.

Anhand des vorgestellten Ähnlichkeitsmaßes wird nun für alle Kantenpunkte P_i ein zweiter Kantenpunkt P_j gesucht, für den $s(i,j)$ maximal wird. Durch diesen einfachen Such- bzw. Zuordnungsalgorithmus lassen sich hauptsächlich genau solche Kantenpunktpaare (P_i, P_j) finden, die sich auf ein und derselben Linienkante befinden, wie nachfolgende Abbildung zeigt.

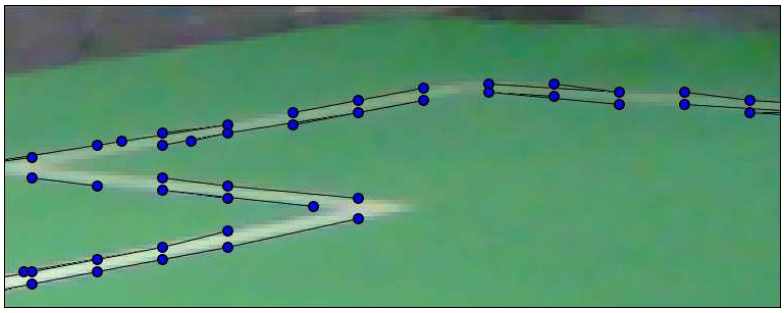


Abb. 3.38: Dargestellt sind alle Kantenpunktpaare für welche $s(i,j)$ maximal ist unter der Bedingung $s(i,j) > 0$.

Weiterführend sei auf die Relevanz des Schwellenwertes d_{min} für einen hinreichenden Abstand zwischen P_i und P_j hingewiesen. Denn anderenfalls käme womöglich bei der Berechnung von $s(i,j)$ eine Division durch 0 zustande (falls $P_i = P_j$) bzw. führten kleine Positionsungenauigkeiten von P_i und P_j zur unpräzisen Schätzung von v_{ij}. Die Maximaldistanz d_{max} wiederum wurde in erster Linie zur Laufzeitbegrenzung der beschriebenen Kantenpaarsuche eingeführt. Ohne diese hätte der Zuordnungsalgorithmus ein Laufzeitverhalten von mindestens $O(n_G^2)$, da für alle möglichen Punktpaare der Koeffizient $s(i,j)$ berechnet würde. Die Einschränkung auf eine feste Maximaldistanz d_{max} erlaubt für einen beliebigen Kantenpunkt P_i nur eine konstante Anzahl k an benachbarten Kantenpunkten P_j innerhalb dieser Distanz, so dass insgesamt lediglich $n_G * k$ verschiedene Berechnungen des Ähnlichkeitskoeffizienten durchgeführt werden müssen. In der Praxis erwieß sich der Wert $d_{max} := 2.8 * s_{Raster}$ als geeignet und liegt die Anzahl k von benachbarten Kantenpunkten

üblicherweise zwischen 0 bis 20.

Inwiefern eine Steigerung der Richtungsgenauigkeit durch die beschriebene Paarbildung erreicht wurde, kann nun mittels des bereits vorgestellten Bewertungskriteriums e (Formel 3.9) überprüft werden. Dazu werden die zu testenden Gradienten g eines jeden Kantenpunktes so korrigiert, dass sie nun orthogonal zur Strecke $\bar{P_iP_j}$ verlaufen. Hierdurch wird schließlich ein Fehlerwert von $e \approx 0.016232$ erreicht, der weniger als halb so hoch ist, wie der des Sobel-Operators und damit eine deutliche Verbesserung der Richtungsgenauigkeit widerspiegelt.

Gruppierung von Kantenpunktpaaren

Zum Zweck der Ermittlung zusammenhängender Liniengruppen wurde in dieser Arbeit eine Distanzfunktion z entwickelt, anhand derer ein Vergleich der aus jeweils zwei Kantenpunkten bestehenden Kantenpunktpaare v und w erfolgt. Diese Funktion überprüft nun die Plausibilität der Zugehörigkeit dieser vier gegebenen Punkte zur selben Linienkante. Hierzu wird jeweils ein Punkt A aus dem Kantenpaar v und ein Punkt B aus dem Kantenpaar w so ausgewählt, dass der euklidische Abstand $d_{AB} = \left|A - B\right|$ maximal wird. Anschließend wird durch A und B ein Gerade g_{AB} gelegt und die euklidischen Abstände ϵ_1 und ϵ_2 der übrigen beiden Punkte C und D zu dieser Geraden bestimmt (siehe Abbildung 3.39). Der Wert der Distanzfunktion z sei nun definiert als das arithmetische Mittel von ϵ_1 und ϵ_2, wobei

$$\epsilon_1 := \frac{(C_x - A_x) * (A_y - B_y) + (C_y - A_y) * (B_x - A_x)}{d_{AB}}$$

und

$$\epsilon_2 := \frac{(D_x - A_x) * (A_y - B_y) + (D_y - A_y) * (B_x - A_x)}{d_{AB}}$$

ist.

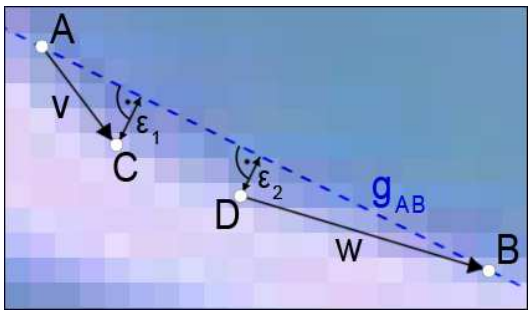


Abb. 3.39: Euklidische Distanzen als Bewertungskriterium für Kantenpunktpaare

Im Idealfall sind alle vier Punkte kollinear und die Distanzfunktion ergibt $z(v,w) = \frac{\epsilon_1+\epsilon_2}{2} = 0$. Im abgebildeten Beispielbild trifft dies jedoch nicht zu, sondern ist $z(v,w) \approx 2,1$. Je größer $z(v,w)$ wird, desto unwahrscheinlicher ist die Zugehörigkeit der Kantenpaare v und w zur selben Linienkante. Zur Entscheidung, ob diese Kantenpaare zu einer Liniengruppe zusammenzufassen sind, wird unter Verwendung eines Schwellenwertes ϵ_{max} die Relation

$$R(v,w) := \begin{cases} true & falls \quad z(v,w) < \epsilon_{max} \\ false & sonst \end{cases}$$

definiert. Sie gibt an, ob v und w zur gleichen Linienkante zugeordnet werden sollen. Anhand von R können nun die Mengen aller zusammenhängenden Kantenpaare ermittelt werden, wobei jede dieser Mengen eine Spielfeldlinie repräsentiert. Zur Erkennung von Ausreißern wird zusätzlich eine Mindestanzahl der Elemente einer solchen Menge definiert, wobei empirisch eine Anzahl von mindestens drei Kantenpunkten festgelegt wurde. Die folgenden zwei Abbildungen zeigen alle Mengen mit mindestens drei Kantenpunkten. Zur übersichtlichen Darstellung wurden hierbei immer nur benachbarte Kantenpunkte einer Menge miteinander durch Linien verbunden.

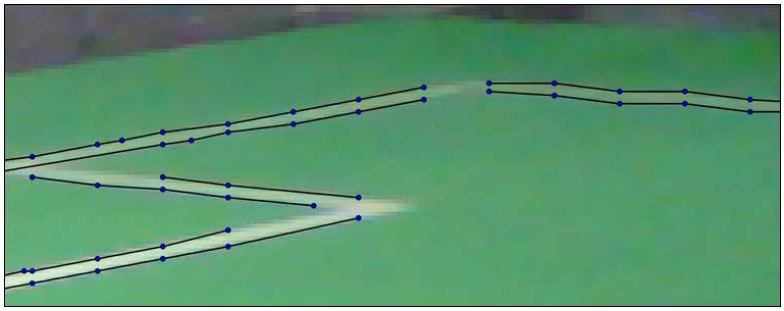


Abb. 3.40: Darstellung von zusammenhängenden Kantenpunkten für $e_{max} := 1$

Weiterführend wird aus diesen Mengen jeweils mit der Methode der kleinsten Quadrate

eine Ausgleichsgerade berechnet. Außerdem werden zur Detektion der Linienelemente des Mittelkreises zusätzlich eine Krümmungsanalyse und ein Ellipsenmatching vollzogen. Diese Verfahren werden jedoch aus Umfangsgründen nicht im Detail in dieser Arbeit beschrieben.

Das beschriebene Verfahren zur Linienerkennung erreicht nach einer Analyse durch die Testbilddatenbank eine Trefferquote von 90,0%, wobei zur Berechnung aller Einzelbewertungen die jeweilige Linienlänge als lineare Gewichtung einbezogen wird und eine Linie genau dann als detektiert gilt, wenn mindestens 50% der vollständigen Linienlänge richtig erkannt wurde. Abbildung 3.41a zeigt den Fall, dass zwei Linien (an der linken und rechten Seite des Bildes) nicht registriert wurden. Da diese Linienabschnitte im Kamerabild jedoch besonders kurz ausfallen, gehen sie mit einer - entsprechend ihrer Länge - etwas geringen Gewichtung in die Gesamtbewertung dieses Bildes ein.

Auffällig ist die sich ergebende Falsch-Positiv-Rate von 4,6% aufgrund falsch detektierter Linie auf Teilen des Tornetzes (siehe Abbildung 3.41b). Die Erkennung und Eliminierung dieser Ausreißer erfolgt in der Praxis jedoch im Zusammenhang mit der Selbstlokalisierung des Roboters, bei der versucht wird, möglichst viele ermittelte Linien mit einem Modell des Spielfeldes zu vereinen. Dieser Schritt ist jedoch nicht Bestandteil dieser Arbeit.

(a) (b)

Abb. 3.41: kurze Linienstücke werden teilweise nicht erkannt (a), Linien-Fehlerkennungen im Netz eines Tores sind möglich (b)

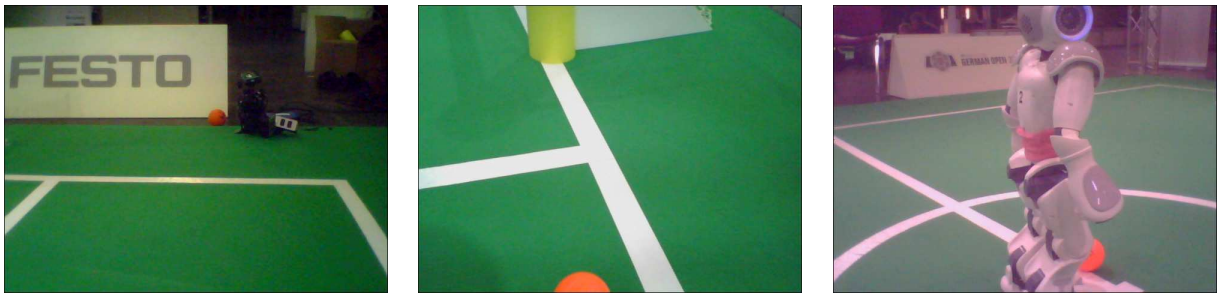
Für die nachfolgenden Abschnitte (Ball- und Torerkennung) ist die Farbe der Linien in Form des Tripels $(Y_{line}, Cb_{line}, Cr_{line})$ relevant. Dieses Tripel wird durch Bildung des arithmetischen Mittels der Farbwerte entlang der ermittelten Linienverläufe bestimmt. Sollte in einem Bild keine einzige Linie erkannt werden, wird auf ältere Daten zuvor analysierter Kamerabilder zurückgegriffen.

3.6 Ballerkennung

3.6.1 Erkennungsmerkmale

Position

Der Spielball sollte lediglich innerhalb der eigenen Spielfeldgrenzen als solcher erkannt werden, denn auch das Vorkommen weiterer Bälle außerhalb der Spielfeldbegrenzung (siehe Abbildung 3.42a) und eine hiermit verbundene Verwechslung ist denkbar, so dass die Roboter womöglich den falschen Ball ansteuern und das Spielfeld hierbei verlassen.



(a) (b) (c)

Abb. 3.42: mögliche Positionen des Balls

Des Weiteren ist zu berücksichtigen, dass der Mittelpunkt des Spielballs außerhalb des Kamerabildes liegen kann, der Ball aber dennoch erkannt werden muss. Insbesondere beim Dribbeln und Schießen tritt dieser Fall des Öfteren auf (siehe Abbildung 3.42b).

Form

Das wohl aussagekräftigste Merkmal des Spielballs ist dessen kreisförmige Kontur im Kamerabild. Allein durch diese Eigenschaft lässt er sich von allen anderen möglichen Objekten auf dem Spielfeld unterscheiden. Wird der Ball lediglich zum Teil im aktuellen Kamerabild abgebildet oder durch Objekte überdeckt, steht dem Roboter jene kreisförmige Kontur jedoch nicht immer vollständig zur Ballerkennung zur Verfügung (siehe Abbildung 3.42c).

Größe

Die maximale Größe des Balls im Bild ist abhängig von seiner Entfernung zur Kamera. Je kleiner diese, desto größer der Balldurchmesser. Nimmt man an, dass die Höhe h_{cam} der Roboterkamera über dem Spielfeldboden etwa $450mm$ beträgt, die Kamera selbst einen horizontalen Öffnungswinkel von $\alpha_{fov} = 46,4°$ und der Spielball einen Durchmesser d_{ball} von etwa $65mm$ hat, kann man den maximalen Kreisdurchmesser d_{max} des Balls im Kamerabild analog zur Bestimmung der Linienbreite in Abschnitt 3.5.1 durch die Formel

$$d_{max} \approx \left\lceil \frac{w}{\alpha_{fov}} * 2 * \tan^{-1}(\frac{d_{ball}}{2 * h_{cam}}) \right\rceil = \left\lceil \frac{640px}{46,4°} * 2 * \tan^{-1}(\frac{65mm}{2 * 450mm}) \right\rceil = 114px$$

grob approximieren. Zur Untermauerung jenes Näherungswertes wurde zusätzlich das Bild 3.43a aufgenommen, in welchem der Spielball direkt vor den Füßen des Roboters liegt. Der gemessene Durchmesser beträgt hier 115 Pixel, was dem zuvor errechneten Schätzwert nahe kommt. Für die spätere Filterung von Ausreißern wurde jedoch ein großzügiger Wert, nämlich $d_{max} := 130px$, festgelegt, da durch Bewegungsunschärfe oder eine geringere Höhe des Roboters der berechnete Durchmesser des Balls im späteren Spielverlauf etwas größer als $115px$ ist.

(a) (b)

Abb. 3.43: maximale (a) und minimale (b) Größe des Balls

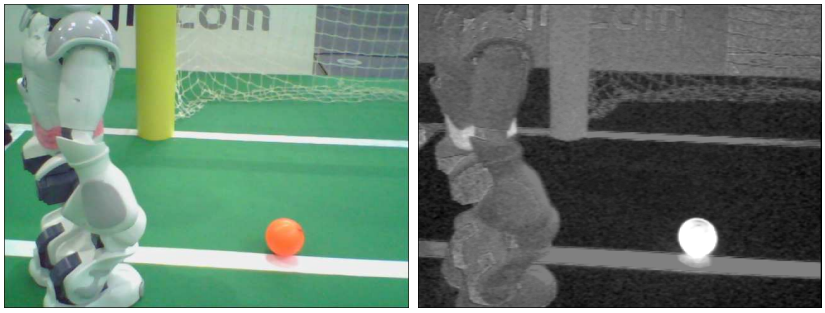
Die minimale Ballgröße d_{min} wird analog zur minimalen Linienbreite ausschließlich anhand von Testbildern festgelegt. Denn je weiter der Ball entfernt ist, desto schlechter bleibt dessen Kreisform aufgrund der abnehmenden Auflösung erhalten. Im ungünstigsten Fall steht der Roboter an einer Ecke des Spielfeldes und der Ball liegt an der diagonal gegenüberliegenden

Spielfeldecke (siehe Abbildung 3.43b). Hier beträgt der Durchmesser lediglich 8 Pixel, so dass ein minimaler Balldurchmesser von $d_{min} := 8px$ definiert wird. Kleiner sollte jener Wert nicht ausfallen, da dies aufgrund der geringen Anzahl an zugehörigen Pixeln mit einer zunehmenden Gefahr von Fehlerkennungen verbunden wäre.

In [BJ02] wird zudem vorgeschlagen, die Größe des Balls für eine bestimmte Pixelkoordinate anhand des Neigungswinkels der Kamera zum Spielfeldboden analytisch zu errechnen. Da man hierbei jedoch von zusätzlichen Sensoren - den Neigungssensoren des Roboters - abhängig wäre und im Worstcase durch eine schlechte Kalibrierung jener Sensoren eine möglicherweise falsche Ballgröße errechnet werden könnte, wurde in dieser Arbeit von weiteren Einschränkungen der Ballgröße abgesehen.

Farbdifferenzen

Ein weiteres wichtiges Erkennungsmerkmal ist die rötliche Farbe des Spielballs, die im Cr-Kanal des Kamerabildes besonders hohe Werte aufweist. Die nachfolgende Abbildung (b) zeigt die Intensitätswerte des Cr-Kanals als Graustufenbild (a), wobei hellere Pixel für höhere Cr-Werte stehen.



(a) (b)

Abb. 3.44: Intensitätswerte im Cr-Kanal als Grauwertbild

Des Weiteren ist die unmittelbare Umgebung des Balls zum größten Teil Spielfeldfarben oder Linienfarben. Dieser rot-grüne bzw. rot-weiße Farbunterschied spielt bei der Erkennung der Objektgrenze des Spielballs im Abschnitt 3.6.4 eine wesentliche Rolle.

3.6.2 Grundsätzliches Vorgehen

Das Prinzip der Ballerkennung umfasst folgende drei Schritte und ist in Abbildung 3.45 unterstützend dargestellt:

1. Grobe Bestimmung einer Position im Bild, die am ehesten innerhalb der Fläche des Spielballs liegt
2. Detektion der Objektgrenzen durch sternförmig angeordnete Scanlines
3. Durchführung einer Formerkennung zur Überprüfung, ob es sich bei dem analysierten Objekt tatsächlich um den Spielball handelt

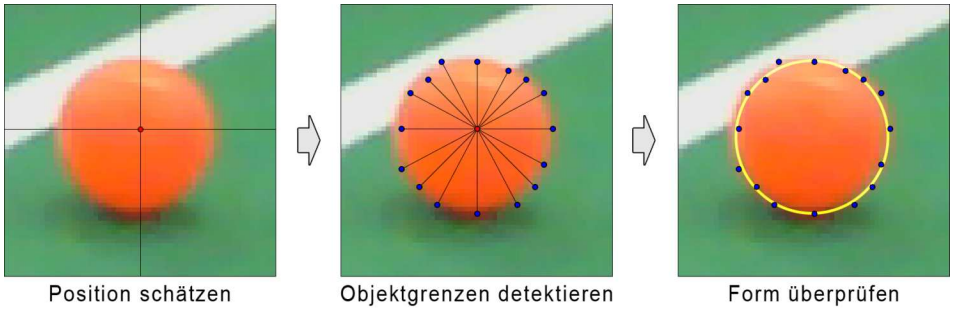


Abb. 3.45: Ablauf der Ballerkennung

3.6.3 Positionsschätzung

Eine möglichst schnelle Berechnung der Ballposition ist für die Echtzeitanforderung im Roboterfußball essentiell. Hierbei sollte einerseits ein Mindestmaß an Bildpunkten analysiert werden, so dass auch weit entfernte Bälle mit minimalem Durchmesser von etwa 8 Pixeln in jedem Fall detektiert werden, andererseits ist eine möglichst geringe Verarbeitungszeit pro Bildpunkt anzustreben. Wie eingangs erwähnt, gibt es bereits einige Ansätze der Verwendung dynamischer Suchpunktraster oder Scanlinegitter zur Ballsuche, welche sich je nach Kameraneigung der perspektivischen Verzerrung der Spielfeldebene anpassen und für weit entfernte Objekte somit höher auflösen als für nahe Objekte. Dies erfordert jedoch gut kalibrierte Neigungssensoren. Um möglichst unabhängig von weiteren Sensoren zu bleiben, kommen derartige dynamische Suchpunktraster/-gitter in dieser Arbeit nicht zur Anwendung, zumal die hierfür benötigten Neigungssensoren der NAO-Plattform aufgrund ihrer Temperaturabhängigkeit [LHB+09b] kontinuierlich kalibriert werden müssten. Zudem kann sich auch der Winkel der Kamera bezüglich des Roboterkopfes nach einem Sturz um einige Grad verstellen, was einer weiteren Rekalibrierung bedürfte.

Um bei der Nutzung eines Suchpunktrasters mit einem festen Pixelabstand dennoch möglichst zeiteffizient das Bild zu analysieren, wird untersucht, inwiefern nur ein einziger der drei Farbkanäle zur Schätzung der Ballposition ausreicht. Denn - wie im Abschnitt 3.6.1 erwähnt - beinhaltet allein der Cr-Kanal, welcher rotfarbene von türkisfarbenen Werten unterscheidet, aufschlussreiche Pixeldaten zur Erkennung der Ballfarbe.

Hierzu wird mittels der Testdatenbank folgende Untersuchung durchgeführt:

Auf insgesamt 241 von 600 Bildern in der Datenbank sind der Ball sichtbar sowie die Position seines Mittelpunktes und seines Radius als Ground-Truth-Daten hinterlegt. Für diese ausgewählten Bilder erfolgt nun auf einem starren Raster die Analyse aller Pixel unterhalb der gegebenen Spielfeldgrenze hinsichtlich ihres Cr-Wertes sowie die Lokalisierung desjenigen Pixels mit dem größten Cr-Wert. Vermutlich liegt dieser Bildpunkt auf der Fläche des Spielballs, da es theoretisch keine anderen Objekte auf dem Spielfeld geben kann, die ein farbintensiveres Rot - d.h. einen höheren Cr-Wert - aufweisen. Es interessiert nun die Anzahl der Bilder, für die jene Vermutung zutreffend ist:

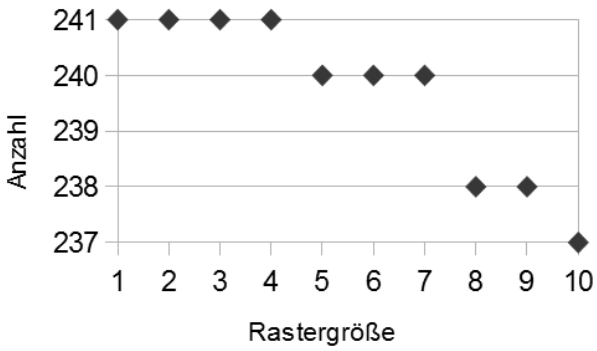


Abb. 3.46: Bestimmung der idealen Rastergröße

Das Ergebnis zeigt, dass für alle der 241 Bilder bei einem Raster mit dem Pixelabstand 4 die ermittelte Pixelposition jeweils auf der Ballfläche lag, so dass geschlussfolgert werden kann, dass allein die Analyse des Cr-Kanals hinreicht, um eine grobe Positionsbestimmung des Spielballs vorzunehmen. „Grobe Positionsbestimmung" impliziert in diesem Fall, dass die Stelle des höchsten Cr-Wertes zwar auf der Ballfläche liegt, sich jedoch nicht zwingend exakt in dessen Mittelpunkt befindet.

Da dieses Verfahren zu jedem Bild - unabhängig vom Vorhandensein eines Balls - eine Objektposition berechnet, ist zur Vermeidung von Fehlerkennungen eine anschließende Überprüfung der Objektform erforderlich.

3.6.4 Ermittlung der Objektgrenzen

Zur Untersuchung der Form eines Objektes muss diese zunächst durch Erkennung der Objektgrenzen bestimmt werden. In [MC04, LEC07] wird eine einfache und zeiteffiziente Methode beschrieben, in der - ausgehend von einem beliebigen Startpunkt innerhalb des zu untersuchenden Objektes - durch n_K Strahlen n_K Kantenpositionen ermittelt werden. Je größer hierbei n_K gewählt wird, desto besser approximieren diese Kantenpositionen die Objektform. Die Koordinatenberechnung derartiger Kantenpunkte erfolgt nach [LEC07] anhand der Suche eines entsprechend großen Helligkeits- bzw. Farbabstandes benachbarter Pixel auf einem Strahl unter Verwendung des Sobel-Operators. Das nachfolgende Beispielbild zeigt die durch diese Methode ermittelten Objektgrenzen für $n_K = 8$.

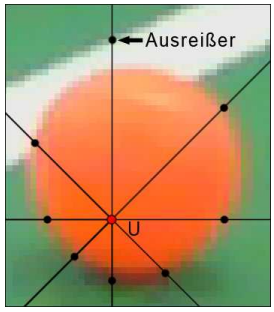


Abb. 3.47: Ermittlung von Punkten auf der Objektgrenze

Die Problematik dieses Verfahrens besteht jedoch darin, dass nicht definiert ist, welchen Abstand die zu bestimmenden Kantenpunkte vom Ursprungspunkt U haben dürfen. Beispielbild 3.47 verdeutlich, dass der stärkste Helligkeits- bzw. Farbunterschied nicht auf der Kontur des Balls liegt, sondern auf benachbarten Objekten, wie beispielsweise Linienkanten.

Da jedoch in der vorliegenden Arbeit durch die bereits vorgestellten Algorithmen die Linienfarbe - bezeichnet durch das Tripel $(Y_{line}, Cb_{line}, Cr_{line})$ - als auch die Spielfeldfarbe - im Weiteren mit $(Y_{field}, Cb_{field}, Cr_{field})$ bezeichnet - ermittelt wurden, erscheint es zur Lösung zuvor erwähnter Problematik sinnvoll, diese Farbinformationen in die Detektion der Objektgrenzpunkte einfließen zu lassen. Denn ein Randpunkt des Balls liegt gewöhnlich genau an jener Pixelposition, an der ein Wechsel von der Ballfarbe zur Linien- oder Spielfeldfarbe stattfindet. Zur Ermittlung dieser Pixelposition ist folglich eine initiale Schätzung der Ballfarbe erforderlich. Da der im vorherigen Abschnitt ermittelte Ursprungspunkt U bereits auf dem zu untersuchenden Objekt liegt, wird die Farbe des Pixels an dieser Position U im Weiteren als Ball-Referenzfarbe $(Y_{ball}, Cb_{ball}, Cr_{ball})$ verwendet.

Der in dieser Arbeit umgesetzte Algorithmus zur Bestimmung der Objektgrenze analysiert

nun der Reihe nach, ausgehend vom Ursprungspunkt U, die auf einem Strahl liegenden Pixel bezüglich ihrer Farbwerte, die im Folgenden mit $(Y_{ray}, Cb_{ray}, Cr_{ray})$ bezeichnet sind. Dieser sukzessive Vorgang wird bis zum Eintreten der Abbruchbedingung

$$D(Cb_{ray}, Cr_{ray}, Cb_{ball}, Cr_{ball}) \geq D(Cb_{ray}, Cr_{ray}, Cb_{field}, Cr_{field}) \ \wedge$$

$$D(Cb_{ray}, Cr_{ray}, Cb_{ball}, Cr_{ball}) \geq D(Cb_{ray}, Cr_{ray}, Cb_{line}, Cr_{line})$$

durchgeführt, wobei

$$D(Cb_1, Cr_1, Cb_2, Cr_2) := (Cb_1 - Cb_2)^2 + (Cr_1 - Cr_2)^2$$

ist. Die Funktion D ermittelt also anhand der Cb- und Cr-Werte den quadrierten euklidischen Farbabstand zwischen zwei Bildpunkten. Es sei erwähnt, dass zur Berechnung des Farbabstandes der Helligkeitskanal Y nicht verwendet wird, da dieser durch Schattenbildung auf den Objekten größere Schwankungen aufweist, die einen negativen Einfluss auf die Erkennung der Objektgrenzen ausüben. Solange ein zu analysierender Pixel auf dem Strahl zum Referenzfarbtupel (Cb_{ball}, Cr_{ball}) einen kleineren Farbabstand als zur Linien- oder Spielfeldfarbe hat, ist die Randbegrenzung jenes Objektes noch nicht erreicht, so dass der nächste Pixel auf dem Strahl hinsichtlich dieser Bedingung analysiert wird.

Liegt während der Bestimmung der Objektgrenzen entlang der Strahlen ein zu untersuchender Pixel auf dem Rand des Kamerabildes, so liegt es nahe, dass das analysierte Objekt am Bildrand abgeschnitten wurde. In diesem Fall kann entweder n_k um 1 verringert und der betrachtete Randpunkt verworfen werden oder die Richtung des Strahls - beispielsweise durch Spiegelung am Bildrand - an dieser Stelle so verändert werden, dass er zurück ins Bild läuft und hierdurch ein Grenzpunkt des betrachteten Objektes gefunden werden kann. Insbesondere für teilweise außerhalb des Kamerabildes liegende Bälle können hiermit deutlich mehr Kantenpunkte auf der Objektgrenze ermittelt werden, was einen Vorteil bei der Analyse der Objektform bringen kann.

3.6.5 Analyse der Objektform

Zur Klärung, ob es sich bei dem wie oben beschrieben ermittelten Objekt tatsächlich um den Spielball handelt oder lediglich um ein beliebiges anderes Objekt, dessen Farbwert

im Cr-Kanal der höchste innerhalb des Bildes ist, dient die Analyse und Erkennung der Objektform.

Für die in Abschnitt 3.6.4 ermittelten, auf der Objektgrenze liegenden n_K Punkte soll nun untersucht werden, inwieweit sie einem mathematischem Modell des Spielballs entsprechen. Ein in diesem Zusammenhang nahe liegendes Modell ist die Beschreibung der Ballkontur als Kreis, so dass im Idealfall all diese Punkte $P_i := (x_i, y_i)$ eine Gleichung

$$r^2 = (x_c - x_i)^2 + (y_c - y_i)^2 \tag{3.10}$$

erfüllen und damit auf einem Kreis mit dem Mittelpunkt (x_c, y_c) und dem Radius r liegen. Die Parameter x_c, y_c und r werden im Folgenden als Modellparameter bezeichnet. Eine konkrete Belegung für das Tripel (x_c, y_c, r) wird als ein Modell $m := (x_c, y_c, r),\ m \in \mathbb{R}^3, r > 0$ bezeichnet.

Für die Bestimmung der drei Modellparameter sind mindestens drei Datenpunkte nötig, so dass $n_K \geq 3$ gilt. Denn für genau drei nicht kollineare Punkte ist ein Kreis eindeutig definiert: Das Dreieck, das durch jene drei Punkte gegeben ist, hat einen eindeutigen Umkreis. Der Mittelpunkt dieses Umkreises ist der Schnittpunkt der Mittelsenkrechten der drei Seiten und der Radius ergibt sich aus dem euklidischen Abstand dieses Mittelpunktes zu einem beliebigen der drei Datenpunkte (siehe Abbildung 3.48).

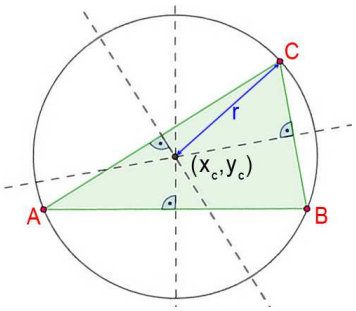


Abb. 3.48: Bestimmung des Kreismittelpunktes aus drei Punkten

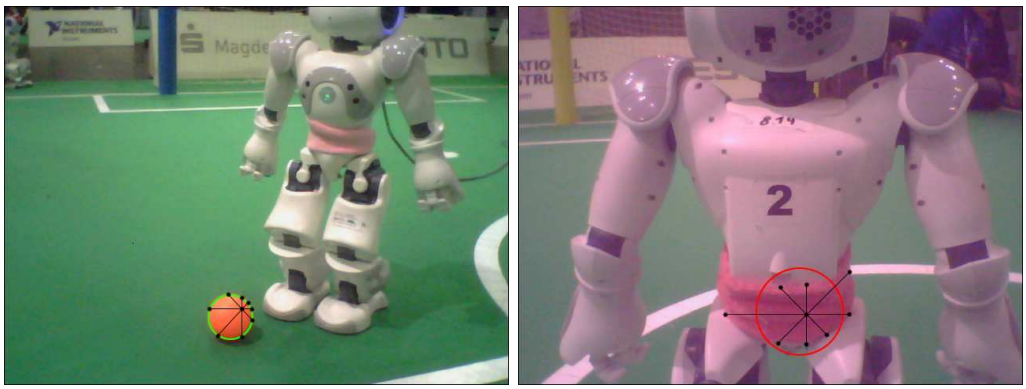
Da eine Auswahl lediglich dreier Punkte zum Zweck der Modellberechnung jedoch aufgrund von Ungenauigkeiten in der Positionsbestimmung der Datenpunkte erhebliche Modell-Ungenauigkeiten hervorrufen kann, wird eine höhere Anzahl n_K an Punkten herangezogen und hiermit eine robustere Berechnung garantiert: Sind mehr als drei Datenpunkte gegeben, so lassen sich womöglich keine Modellparameter finden, so dass alle Datenpunkte die Gleichung 3.10 erfüllen. Aus diesem Grunde werden Modellparameter gesucht, für die der Abstand der Datenpunkte zum berechneten Kreis insgesamt möglichst gering ist. Eine

hierzu häufig verwendete Methode ist die Bestimmung der Summe der quadratischen Abstände

$$dist(x_x, y_c, r) := \sum_{i=1}^{n_K} (\sqrt{(x_c - x_i)^2 + (y_c - y_i)^2} - r)^2 \tag{3.11}$$

und die Optimierung der drei Modellparameter, so dass *dist* minimal wird. Zur Lösung dieses Optimierungsproblems wird in [Bir08, RLM+09, Coa03] der Levenberg-Marquardt-Algorithmus [Mar63] vorgeschlagen, welcher mit Hilfe der Methode der kleinsten Quadrate nichtlineare Ausgleichsprobleme numerisch löst. Nach Bestimmung der Modellparameter repräsentiert der minimale Wert von *dist* die Plausibilität des ermittelten Kreises in Bezug auf die n_K Datenpunkte.

Ist *dist* nun kleiner als ein Schwellenwert, so wird das untersuchte Objekt als „kreisförmig" klassifiziert (siehe Abbildung 3.49a) oder andernfalls als potentieller Spielball ausgeschlossen (siehe Abbildung 3.49b).



(a) (b)

Abb. 3.49: korrekte Klassifizierung der Objektformen (grün=erkannter Ball, rot=kein Ball)

Die Grenzen der Anwendbarkeit jenes Klassifikationsverfahrens offenbaren sich jedoch - wie in Abbildung 3.50 gezeigt - im Falle der teilweisen Überdeckung des Spielballs, beispielsweise durch einen Roboter. Denn hier handelt es sich zwar um den gesuchten Ball, jedoch kann kein Modell gefunden werden, für welches alle n_K Kantenpunkte nah genug am entsprechenden Kreis liegen, also *dist* klein genug ist. Die Werte der hier rot dargestellten Ausreißer bedingen einen besonders hohen quadratischen Fehler und verhindern hiermit die korrekte Berechnung der Kreisform.

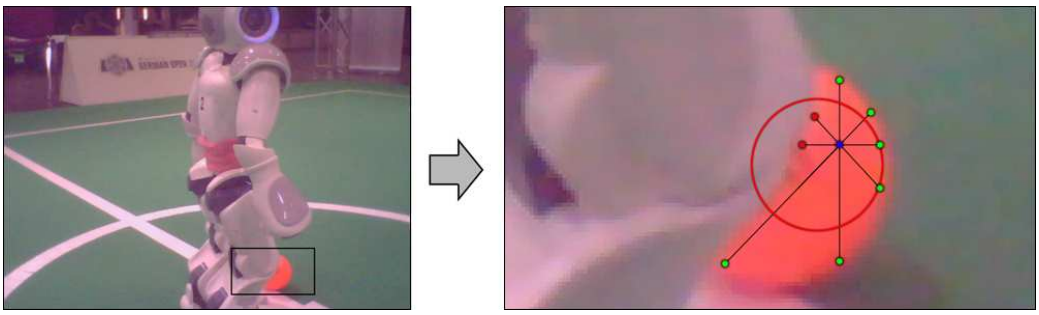


Abb. 3.50: vollständige Ballform aufgrund der Überdeckung nicht sichtbar

Zur Lösung dieses Problems müssten jene Ausreißerpunkte, die nicht auf dem gesuchten Kreisumfang liegen, explizit gefunden und aussortiert werden, so dass die Bestimmung und Bewertung der Modellparameter mittels der Formel 3.11 ausschließlich anhand der „richtigen" Datenpunkte (grün) erfolgen kann.

Zur Klärung, ob ein Datenpunkt P_i mit einem Modell m vereinbar oder ein Ausreißer ist, wird die binäre Funktion

$$w((x_i, y_i), (x_c, y_c, r)) := \begin{cases} 1 & falls \quad (r - t_{dist})^2 \leq (x_c - x_i)^2 + (y_c - y_i)^2 \leq (r + t_{dist})^2 \\ 0 & sonst \end{cases} \tag{3.12}$$

unter Verwendung des Schwellenwertes t_{dist} , unter der Bedingung $t_{dist} \leq r$, definiert, die genau dann 1 ist, wenn ein Punkt P_i einen minimalen euklidischen Abstand zu dem durch m definierten Kreis von maximal t_{dist} hat (siehe Abbildung 3.51). Im Weiteren wird daher ein Punkt P_i, für den $w(P_i, m) = 1$ ist, als Modellpunkt bezeichnet.

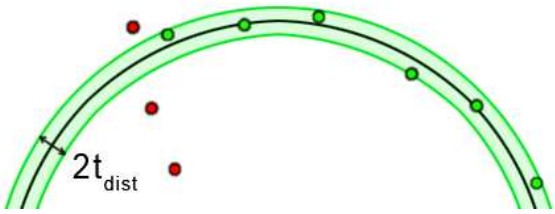


Abb. 3.51: Erkennung von Ausreißern (rot) und Markierung der Modellpunkte (grün)

Gesucht ist nun ein Modell m, für das

$$\sum_{i=1}^{n_K} w(P_i, m)$$

maximal wird, so dass es dem entsprechend die höchste Anzahl an Modellpunkten umfasst. Dieses wird unter allen möglichen Varianten als wahrscheinlichstes Modell angesehen. Da für m unendlich viele Modelle infrage kommen, wird zur Vereinfachung des Problems nur eine endliche Teilmenge $E \subset \mathbb{R}^3$ betrachtet, die all diejenigen Kreise beinhaltet, die mindestens durch drei gegebene Datenpunkte verlaufen. Die Elemente von E ergeben sich also durch die Berechnung aller Kreise, die durch beliebige Kombinationen von drei nicht kollinearen Datenpunkten bestimmt werden können.

Ein naiver Ansatz zur Ermittlung des „besten" Modells m besteht in der Prüfung sämtlicher Modelle aus E auf die Anzahl ihrer Modellpunkte und die Selektion desjenigen, für welches $\sum_{i=1}^{n_K} w(P_i, m)$ schließlich maximal ist. Dieser Ansatz ist jedoch rechenaufwendig, da die Anzahl der zu überprüfenden Elemente $\left|E\right| = \begin{pmatrix} n_K \\ 3 \end{pmatrix} = \frac{n_K!}{3!*(n_K-3)!}$ beträgt. Eine dagegen zeiteffiziente Alternative stellt der randomisierte Algorithmus 3.8 [FB81] dar, der eine geringe Anzahl $s \ll \left|E\right|$ der Modelle zufällig aus E auswählt und nur diese auf die Anzahl ihrer Modellpunkte hin überprüft.

Algorithm 3.8 getCircleRANSAC

```
getCircleRANSAC(E, P, s)
{
    max := 0
    for(j := 0; j < s; j := j + 1)
    {
        wähle ein zufälliges Modell m ∈ E
        c := Σ_{i=1}^{n_K} w(P_i, m)
        if(c > max)
        {
            max := c
            m' := m
        }
    }
    return m'
}
```

Hierbei stellt sich die Frage, wie s gewählt werden muss, dass mit gewisser Sicherheit ein korrektes Modell tatsächlich gefunden wird. Geschätzt werden soll die Wahrscheinlichkeit p_r, dass bei s Stichproben mindestens ein Kreis durch drei Punkte definiert wird, der der tatsächlichen Form des Spielballs entspricht.

Sei p_o die Wahrscheinlichkeit dafür, dass ein Datenpunkt nicht korrekt auf dem Rand des Spielballs detektiert wurde, dann hat die Wahrscheinlichkeit p_c, dass ein zufällig ausgewähltes Modell m aus drei korrekten Punkten berechnet wurde, den Wert $(1 - p_o)^3$.

Folglich ergibt sich die Wahrscheinlichkeit $p_r = 1 - (1 - p_c)^s$, dass nach s Iterationen mindestens ein den Anforderungen entsprechendes Modell m ausgewählt wurde.

Ist beispielsweise eine Ausreißerwahrscheinlichkeit von $p_o := 30\%$ gegeben, dann ist nach $s := 10$ Iterationen die Wahrscheinlichkeit, dass mindestens ein korrektes Modell unter den getesteten ermittelt wurde $p_r := 1 - (1 - (1 - 0.30)^3)^{10} = 0.985$.

Im Gegensatz zum vorherigen Verfahren, bei dem eine Erkennung der Ballform durch den Vergleich des Fehlerwertes *dist* mit einem Schwellenwert erfolgt, wird im vorgestellten RANSAC-Verfahren zur Entscheidungsfindung, ob es sich bei dem untersuchten Objekt um den Spielball handelt, ausschließlich die Anzahl der Modellpunkte betrachtet. Weiterhin wird definiert, dass nur dann, wenn diese Anzahl größer oder gleich $\frac{n_K}{2}$ ist, die untersuchte Objektform als Spielball klassifiziert wird.

Durch diesen Ansatz wurde schließlich mittels der Testbilddatenbank unter Verwendung eines vom Ballradius abhängigen Schwellenwertes $t_{dist} := 0.05 * r$ eine Erkennungsrate von 99,2% (239 von 241 Bilder) erzielt, wobei die Falsch-Positiv-Rate bei 1,5% (9 von 600 Bildern) liegt.

Bei den fälschlicherweise als Spielball erkannten Objekten liegt typischerweise der Ursprungspunkt U am Objektrand. Dies hat zur Folge, dass über die Hälfte der n_k Randpunkte dicht beieinander liegen und aufgrund ihres geringen Abstandes zueinander unabhängig von der eigentlichen Objektform als Modellzugehörig registriert werden. Die Falsch-Positiv-Rate kann jedoch deutlich reduziert werden, wenn nach der Bestimmung der n_k Randpunkte eine neue Positionsschätzung des Ursprungspunktes U anhand jener Randpunkte vorgenommen wird. Beispielsweise kann hierzu die neue Position von U auf den Mittelpunkt des umschließenden Rechtecks aller bisherigen Kantenpunkte gelegt werden. Dieses Vorgehen reduziert die Falsch-Positiv-Rate von 1,5% auf 0,17% (1 von 600 Bildern).

Weiterführend wird, falls ein Ball in einem Bild erkannt wurde, für die anschließende Torerkennung der Farbton des Balls in Form des Tripels $(Y_{ball}, Cb_{ball}, Cr_{ball})$ abgespeichert.

3.7 Torerkennung

3.7.1 Eigenschaften

Form

Das Tor besteht aus zwei zylindrischen Pfosten und einem verbindenden Querbalken. Jener befindet sich jedoch häufig außerhalb des Kamerabildes, beispielsweise wenn der Roboter nah am Tor steht oder der Neigungswinkel der Kamera zu niedrig ist, so dass hier nur die einzelnen Torpfosten abgebildet sind. Des Weiteren tritt häufig der Fall ein, dass einer der Pfosten außerhalb des Kamerabildes liegt oder durch ein Hindernis - wie beispielsweise einem Roboter - verdeckt wird (siehe Abbildung 3.52a). Deshalb wird im Folgenden hauptsächlich die Erkennung einzelner Torpfosten behandelt.

Die Form eines Torpfostens im Kamerabild entspricht in vielen Fällen annähernd einem Rechteck, wobei dessen Höhen-Breiten-Verhältnis anhand der realen Größen des Pfostens abgeleitet werden kann. Dieser hat mit einer Höhe von 80 cm und einem Durchmesser von 10 cm ein Höhen-Breiten-Verhältnis von 8:1. Wie Abbildung 3.52b zeigt, handelt es sich bei der Form eines Torpfostens jedoch nicht immer um ein Rechteck, da aufgrund der perspektivischen Verzerrung zum einen die beiden langen Seiten nicht zwingend parallel verlaufen müssen, zum anderen die beiden kurzen Seiten (oben oder unten) abgerundet sein können. Durch diese Gegebenheiten und die Möglichkeit, dass ein Torpfosten nur teilweise innerhalb des Kamerabildes liegen kann, fällt zudem das erkennbare Höhen-Breiten-Verhältnis häufig geringer aus.

(a) (b)

Abb. 3.52: mögliche Verdeckung (a) und perspektivische Verzerrung (b) eines Torpfostens

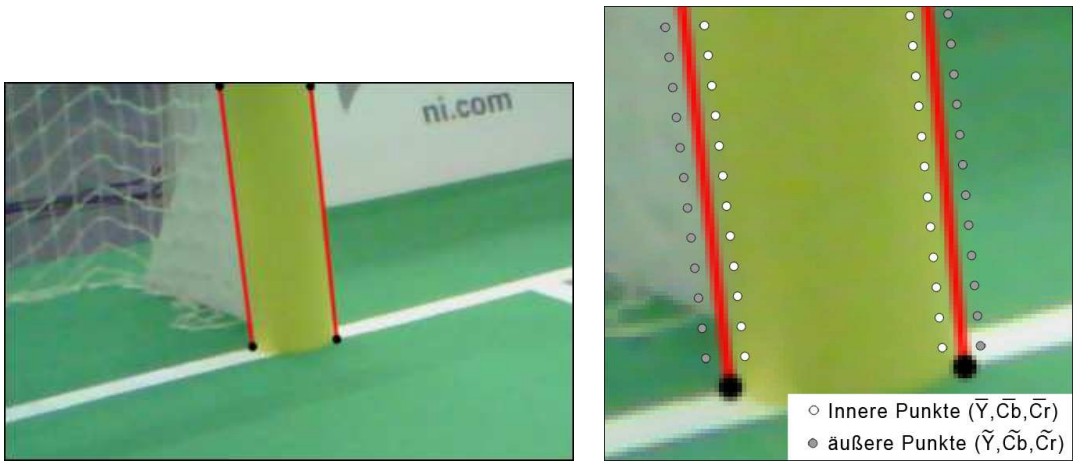
Farbdifferenzen

Eine der größten Herausforderungen der Torpfostenerkennung ist der undefinierte Farbton des Hintergrundes, der es erschwert, die Grenzen der Torpfosten genau zu erkennen. Als Hintergrund werden hier all diejenigen Bildpunkte bezeichnet, die weder zum Tor noch zum Spielfeld gehören.

Zur Vereinfachung dieser Problematik werden bestimmte Erkennungskriterien der Farbdifferenzen zwischen Hintergrund- und Torfarbe durch Untersuchungen mittels der Testbilddatenbank festgelegt. In Abschnitt 3.4.1 wurde bereits beschrieben, wie sich die Farbe des Hintergrundes im Vergleich zur Spielfeldfarbe verhält. Hier wurde gezeigt, dass 99,7% der Hintergrundpixel einen höheren als den durchschnittlichen Cr-Wert des Spielfeldes aufweisen.

Um nun anhand der Testbilddatenbank den Farbkanal zu ermitteln, welcher den Unterschied zwischen Tor- und Hintergrundfarben am besten differenziert, werden zwei ganz ähnliche Untersuchungen in diesem Abschnitt für gelbe bzw. blaue Torpfosten durchgeführt:

Zu jedem Bild in der Datenbank, das mindestens einen Torpfosten enthält, sind zu jedem Pfosten jeweils zwei Strecken abgespeichert, die den realen Verlauf der rechten und linken Pfostenkante beschreiben (siehe Abbildung 3.53a). Die Pixel eines Pfostens befinden sich somit zwischen diesen Strecken, die Pixel des Hintergrundes und des angrenzenden Spielfeldes entsprechend auf deren jeweils gegenüberliegenden Seiten.



(a) (b)

Abb. 3.53: Ground-Truth-Daten der Torpfosten durch zwei gegebene Strecken (a) und Analyse Farbdifferenzen von inneren und äußeren Pixel (b)

Da für jede einzelne dieser Strecken durch den zugehörigen Datenbankeintrag bekannt

ist, auf welcher Seite sich die torfarbenen Pixel und die Hintergrundpixel befinden, kann nun für drei Farbkanäle das Vorzeichen der Farbdifferenz von Pixelwerten innerhalb und außerhalb der Torpfosten bestimmt werden.

Entlang der beiden gegebenen Stecken werden nun die Farben einer Reihe von Pixeln innerhalb des Torpfostens - im Weiteren mit $(\bar{Y}, \bar{Cb}, \bar{Cr})$ bezeichnet - mit den jeweiligen Farben der Pixel außerhalb des Torpfostens - im Weiteren mit $(\tilde{Y}, \tilde{Cb}, \tilde{Cr})$ bezeichnet - paarweise verglichen. In Abbildung 3.53b sind für das gegebene Beispielbild die „inneren" und „äußeren" Pixelpositionen markiert. Für alle verarbeiteten Bilder werden insgesamt $n_{yellow} = 25396$ solcher Pixelpaare für gelbe Torpfosten und $n_{blue} = 18785$ Pixelpaare für blaue Torpfosten automatisiert untersucht. Hieraus erfolgt die Berechnung der folgenden Summen:

$$Y_{neg} := \sum_{i=1}^{n} V(\bar{Y}_i - \tilde{Y}_i) \qquad Y_{pos} := \sum_{i=1}^{n} V(\tilde{Y}_i - \bar{Y}_i)$$

$$Cb_{neg} := \sum_{i=1}^{n} V(\bar{Cb}_i - \tilde{Cb}_i) \qquad Cb_{pos} := \sum_{i=1}^{n} V(\tilde{Cb}_i - \bar{Cb}_i)$$

$$Cr_{neg} := \sum_{i=1}^{n} V(\bar{Cr}_i - \tilde{Cr}_i) \qquad Cr_{pos} := \sum_{i=1}^{n} V(\tilde{Cr}_i - \bar{Cr}_i)$$

Dabei ist

$$V(d) := \begin{cases} 1, & falls\ d < 0 \\ 0 & sonst \end{cases}$$

und es seien die Anzahl an Pixelpaaren je nach betrachteter Torpfostenfarbe entweder n_{yellow} oder n_{blue}. Für die oben definierten Summen ergeben sich folgende Messwerte:

Farbkanal	negativ	positiv
Y	54,8%	44,2%
Cb	99,4%	0,5%
Cr	23,0%	71,4%

(a) gelbe Pfosten

Farbkanal	negativ	positiv
Y	94,3%	5,5%
Cb	0,6%	99,2%
Cr	90,3%	8,6%

(b) blaue Pfosten

Tab. 3.8: Analyse der Farbkanäle für die Torpfostenerkennung

In 99,4% der untersuchten Fälle haben die innerhalb des gelben Torpfostens liegenden Pixel einen geringeren Cb-Wert als ein außerhalb des Pfostens liegender Pixel. Umgekehrt

gilt bei blauen Torpfosten für 99,2% der untersuchten Pixelpaare, dass $\bar{Cb}_i > \tilde{Cb}_i$ ist. Aus diesen Ergebnissen lässt sich schließen, dass sowohl gelbe als auch blaue Torpfosten zu charakteristischen Änderungen der Pixelwerte, insbesondere innerhalb des Cb-Kanals, führen. Zwar bestehen auch im Y- sowie Cr-Kanal statistische Zusammenhänge jener Farbdifferenzen, diese sind jedoch weniger stark ausgeprägt. Für eine grobe Positionsbestimmung der Torpfosten wird folglich ausschließlich der Cr-Kanal verwendet.

Die anderen beiden Farbkanäle werden anschließend bei der Suche der Fußpunkte - der Punkte, an denen die Torpfosten den Spielfeldboden berühren - von Interesse sein: Hier muss einer der folgenden drei Farbübergänge vorliegen, da sich unmittelbar unterhalb des Torpfostens nur

1. das Spielfeld mit der bereits ermittelten Farbe $(Y_{field}, Cb_{field}, Cr_{field})$,
2. eine Linie mit der bereits ermittelten Farbe $(Y_{line}, Cb_{line}, Cr_{line})$ oder
3. ein Ball mit der bereits ermittelten Farbe $(Y_{ball}, Cb_{ball}, Cr_{ball})$

befinden können.

Größe

Die minimale Höhe und Breite eines Torpfosten ist beispielsweise in folgendem Bild bestimmbar, in dem der Roboter die am weitesten vom blauen Tor entfernte Position am Rand des Spielfeldes einnimmt. Hier beträgt die Torhöhe $l_{min} = 92px$. Analog lässt sich die minimale Breite von $b_{min} = 12px$ ermitteln.

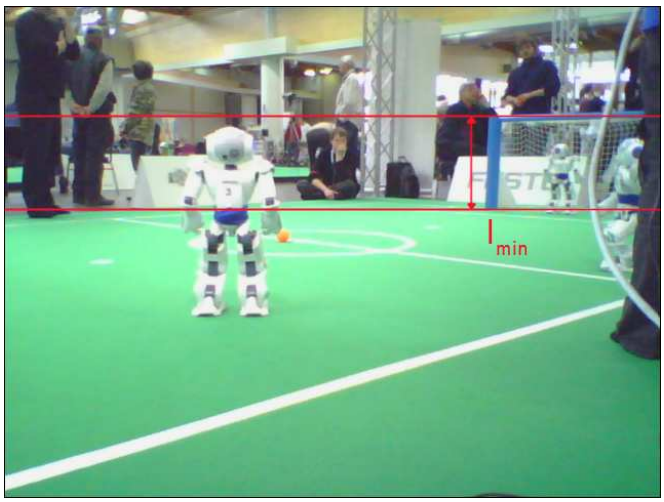


Abb. 3.54: minimale Höhe eines Torpfostens

Position

Die vertikale Position von Torpfosten im Kamerabild lässt sich aufgrund der Feldgeometrie wie folgt einschränken: Da ein Fußpunkt eines Torpfostens immer innerhalb des Spielfeldes liegt, muss auch die Position dieses Fußpunktes im Kamerabild unterhalb der vom Roboter erkannten Spielfeldgrenze liegen. Ebenso lässt sich mittels der durch das Regelwerk [Rob10e] gegebenen Geometrien darauf schließen, dass der höchste Punkt eines Torpfostens im Bild nie unterhalb der Spielfeldgrenze liegen kann, da die Pfosten mit 80 cm höher als die Roboter mit etwa 57 cm sind. Dies wurde durch einen Test mittels der Testbilddatenbank für alle Bilder, in denen mindestens ein Torpfosten sichtbar ist, bestätigt. Sollte der Torpfosten durch die obere Bildgrenze abgeschnitten werden, ist es möglich, dass der am weitesten oben liegende sichtbare Pixel eines solchen Pfostens zwar auf der Spielfeldgrenze liegt, jedoch nach wie vor nicht unterhalb dieser Grenze.

Rotation

Da der Roboter üblicherweise aufrecht steht, sollten die Torpfosten im Bild etwa vertikal ausgerichtet sein. Diesbezüglich wurde das folgende Histogramm erstellt, welches die Verteilung der Winkel aller Torpfostenkanten innerhalb des Beispielbildmaterials darstellt und damit verdeutlicht, dass jene Winkel in einem engen Intervall von lediglich 85-97 Grad liegen.

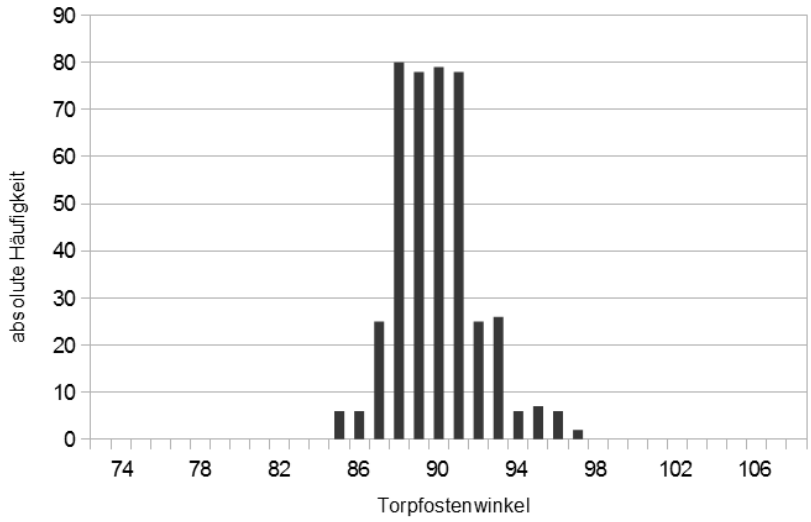


Abb. 3.55: Histogramm aller Torpfostenwinkel in Grad

3.7.2 Grundsätzliches Vorgehen

Das Prinzip der Torerkennung basiert auf folgenden vier Schritten und ist in Abbildung 3.56 unterstützend dargestellt:

1. Schätzung der vertikalen Position der Torpfosten im Kamerabild anhand der Spielfeldgrenze
2. Schätzung der horizontalen Position potentieller Torpfostenkanten durch Analyse der Farbdifferenzen im Cb-Kanal
3. Fußpunktbestimmung
4. Plausibilitätsüberprüfungen

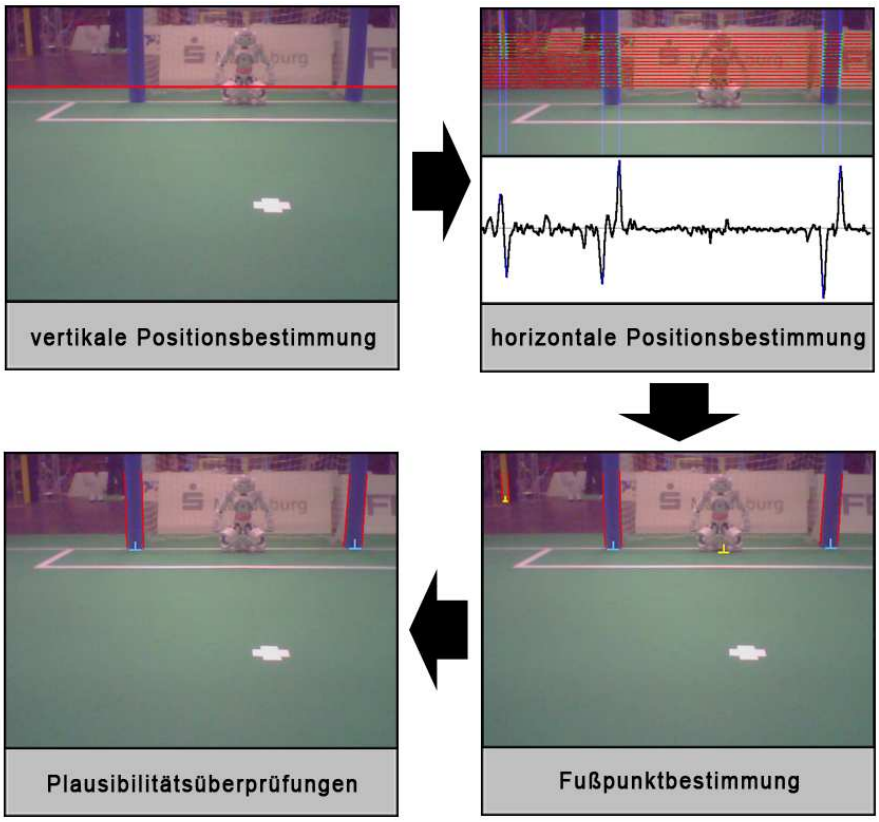


Abb. 3.56: Unterteilung der Torerkennung in vier Einzelschritten

3.7.3 vertikale Positionsbestimmung

Es wurde bereits gezeigt, dass sich die Position der Torpfosten anhand der Spielfeldgrenze beschränken lässt. Durch eine derartige Begrenzung der potentiellen Aufenthaltsbereiche kann die Suche nach Torpfostenkandidaten im Kamerabild zeiteffizienter durchgeführt werden, so dass lediglich jene Bildausschnitte analysiert werden müssen, in denen sich diese gesuchten Objekte mit großer Wahrscheinlichkeit befinden.

Hierfür soll nun eine geeignete Methode etabliert werden, die für ein gegebenes Bild anhand der für dieses Bild ermittelten Spielfeldgrenzen eine waagerechte Gerade mit der Höhe $\bar{y}$ berechnet, die möglichst durch solche Bildpunkte verläuft, die dem gesuchten Torpfosten zugeordnet werden können. Die Koordinate $\bar{y}$ dieser Geraden soll anschließend als Grundlage der Auswahl bestimmter Bildzeilen dienen, anhand derer eine Analyse der Werte im Cb-Kanal zur Bestimmung der x-Koordinaten der Torpfosten erfolgt.

Zunächst wird für jedes Bild der Testbilddatenbank, in dem mindestens ein Torpfosten vorkommt, untersucht, inwiefern die Bildung des arithmetischen Mittels aller y-Koordinaten der Spielfeldgrenzen-Punkte brauchbare Resultate für $\bar{y}$ liefert. Dazu wird zu jedem Bild i aller n_B Bilder der Testbilddatenbank jeweils die Koordinaten $\bar{y}_i$ berechnet und basierend auf diesen Werten die Trefferquote

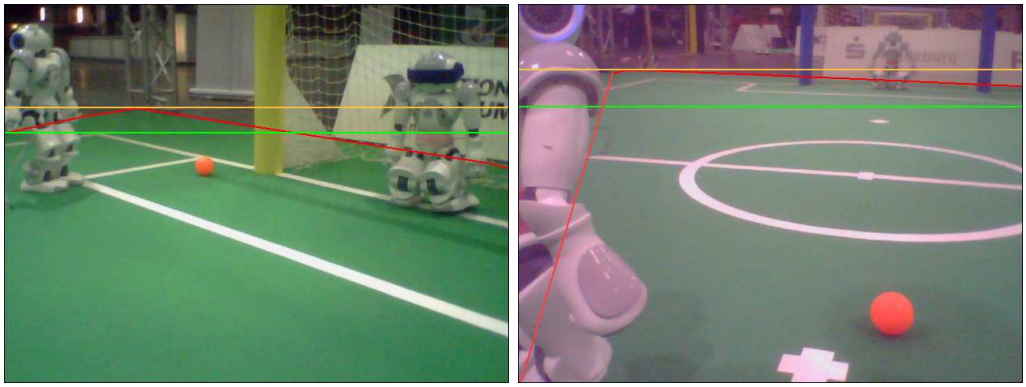
$$q := \frac{1}{n_B * \sum_{i=1}^{n_B} n_{T,i}} * \sum_{i=1}^{n_B} \left(\sum_{j=1}^{n_{T,i}} Z(\check{y}_j, \hat{y}_j, \bar{y}_i) \right) \tag{3.13}$$

ermittelt. Die Funktion Z sei

$$Z(a,b,c) := \begin{cases} 1 & falls\ a < c < b \\ 0 & sonst \end{cases}$$

und ist genau dann 1, wenn $\bar{y}_i$ zwischen der niedrigsten ($\check{y}_j$) und der höchsten ($\hat{y}_j$) Koordinate des jeweils getesteten Torpfostens j liegt. Für ein konkretes Bild i wird die Anzahl der Torpfosten mit $n_{T,i}$ bezeichnet, so dass $n_{T,i}$ entweder 0, 1 oder 2 ist. Ist $n_{T,i} = 0$, sei die Summe $\sum_{j=1}^{0} Z(\check{y}_j, \hat{y}_j, \bar{y}_i) := 0$.

Schließlich ergibt sich eine Trefferquote von $q = 93,4\%$. Offensichtlich weisen also für $6,6\%$ der Torpfosten die ermittelten y-Koordinaten keine akzeptablen Werte auf. Abbildung 3.57 zeigt anhand zweier Beispielbilder, wann die Nutzung des arithmetischen Mittels der y-Koordinaten der Punkte der Feldgrenze zu brauchbaren (a) bzw. zu falschen Ergebnissen (b) führt.



(a) (b)

Abb. 3.57: Bestimmung der vertikalen Position der Torpfosten im Kamerabild

Abbildung 3.57b verdeutlicht, dass durch Hindernisse - wie in diesem Fall den Roboter - die ermittelte Feldgrenze (roter Polygonzug) verfälscht wird und für den daraus errechneten Mittelwert $\bar{y}$ (grüne Linie) folglich eine Position unterhalb der Torpfosten ermittelt wird.

Eine einfache Lösung des Problems besteht darin, statt des arithmetischen Mittels, zur Bestimmung von $\bar{y}$ die oberste y-Koordinate (orangefarbene Linie) der Spielfeldgrenze zu verwenden. Denn hiermit lässt sich für das Testbildmaterial eine ideale Trefferquote von $q = 100\%$ erreichen.

3.7.4 horizontale Positionsbestimmung

Ausgehend von der vertikalen Positionsschätzung $\bar{y}$ der Torpfosten sollen nun durch eine Analyse im Cb-Kanal die entsprechenden x-Koordinaten möglicher Torpfostenkanten ermittelt werden. Der Grundgedanke hierbei ist die Lokalisierung der beiden langen Kanten eines jeden Torpfostens durch Differenzenbildung benachbarter Pixelwerte im Cb-Kanal. Analog zur Detektion von Linienkanten werden zur zeiteffizienten Analyse ebenso Scanlines verwendet. Entscheidend hierbei ist deren Anordnung:

Aufgrund der vertikalen Ausrichtung der Torpfosten eignen sich insbesondere horizontal gelegene Scanlines, denn sie schneiden die Seiten der Torpfosten annähernd orthogonal und vereinfachen dadurch eine Kantendetektion. Um möglichst nur die für eine Torpfostensuche relevanten Bereiche im Bild zu analysieren, werden jene Scanlines lediglich auf bestimmte y-Koordinaten im Intervall $[y_{top}, y_{base}]$ gelegt, wobei

$$y_{base} := max(\bar{y}, n_{scan} * s_{min}) \qquad y_{top} := max(y_{base} - l_{min}, 0)$$

und s_{min} der minimale vertikale Pixelabstand zwischen zwei Scanlines sowie n_{scan} die Gesamtanzahl der zu verwendenden Scanlines seien.

Durch den Algorithmus 3.9 wird nun zu jeder Scanline mit dem Index i eine passende y-Koordinate berechnet und in das Array Y gespeichert.

Algorithm 3.9 getYCoords

```
1  getYCoords(ȳ, n_scan, y_base, y_top)
2  {
3      Array Y mit n_scan Elementen bereitstellen
4      for(i := 0; i < n_scan; i := i + 1)
5      {
6          Y_i := y_top + ((y_base − y_top) * i) div (n_scan − 1)
7      }
8      return Y
9  }
```

Für jede horizontale Scanline i mit der Koordinate Y_i werden nun durch Differenzenbildung der Cb-Werte nebeneinanderliegender Pixel die eindimensionalen symmetrischen Gradienten $g(x) := \frac{1}{2}(f(x+1) - f(x-1))$ in x-Richtung bestimmt, wobei infolge des Chroma-Subsamplings des verwendeten Bildformates dabei nur jeder zweite Pixel betrachtet wird. Wie Abbildung 3.58 a und c zeigen, werden an genau den Stellen auf den Scanlines, wo ein Farbübergang von „Hintergrund"- und „Torpfosten"-farbenen Pixeln stattfindet, besonders hohe Beträge der Farbdifferenzen im Cb-Kanal registriert.

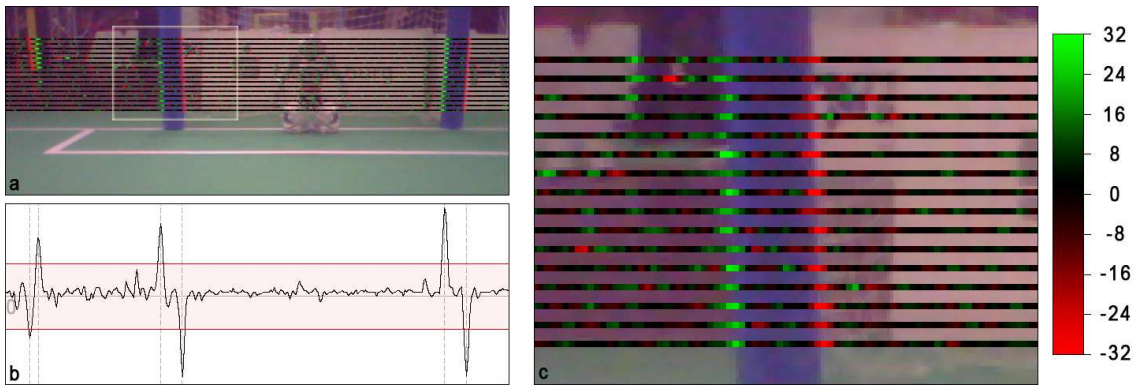


Abb. 3.58: Analyse der Pixelwerte durch $n_{scan} := 16$ Scanlines (a), Detailansicht der farblich visualisierten Gradientenwerte (b) und summierte Gradientenwerte (c)

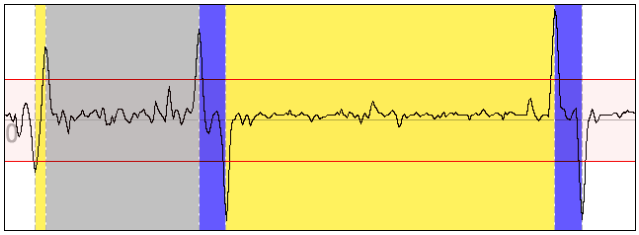
Durch die anschließende Berechnung der Spaltensummen der Gradientenwerte aller Scanlines (siehe Abbildung 3.58b) ist nun die Bestimmung der am ehesten plausiblen x-Koordinaten für Torpfostenkanten möglich. Hierfür wird der bereits im Abschnitt 3.5.3

definierte Kantenerkennungsalgorithmus 3.5 verwendet, durch den die gesuchten Stellen der Extremwerte der Spaltensummen zeiteffizient ermittelt werden können. Dieser nutzt, wie im Beispiel 3.58b durch den rotfarbenen Bereich dargestellt, einen Schwellenwert um die Gesamtheit der ermittelten Extrema auf die wesentlichen zu reduzieren.

Der nächste Schritt besteht darin, die Bereiche zwischen den ermittelten potentiellen Torpfostenkanten nach ihrer Farbe zu klassifizieren. Um weiterhin unabhängig von absoluten Farbwerten zu bleiben geschieht dies ausschließlich anhand der Vorzeichen der jeweiligen Extrema. Da es für einen zu klassifizierenden Bereich jeweils zwei „angrenzende" Extremwerte gibt, die entweder ein positives oder ein negatives Vorzeichen haben, kommen insgesamt vier verschiedene Kombinationsmöglichkeiten in Frage. Tabelle 3.59a stellt für jede dieser vier Möglichkeiten die zuzuordnende Klassifizierung dar. Abbildung 3.59b zeigt schließlich die entsprechende Anwendung für das gegebene Beispiel, in dem die beiden blaufarbenen Torpfostenbereiche korrekt klassifiziert, jedoch auch zwei potentielle gelbfarbene Torpfosten erkannt werden. Es wurde schließlich für alle Bilder der Testdatenbank eine Trefferquote von 98,2% ermittelt, jedoch auch eine sehr hohe Falsch-Positiv-Rate von 82,9%. Zum Ausschluss von Fehlklassifizierungen sind deshalb weitere Daten über die jeweiligen Torpfostenkandidaten notwendig, weshalb im folgenden Abschnitt zusätzlich die Fußpunkte der entsprechenden Bereiche detektiert werden.

linker Extrempunkt	rechter Extrempunkt	Klassifizierung
positiv	negativ	blaues Tor
negativ	positiv	gelbes Tor
positiv	positiv	kein Tor
negativ	negativ	kein Tor

(a)



(b)

Abb. 3.59: Regeln zur Klassifizierung eines Bereiches (a) und Anwendung dieser Regeln auf das gegebene Beispiel (b)

3.7.5 Berechnung der Fußpunkte

Als Fußpunkte der Torpfosten werden genau diejenigen Punkte bezeichnet, an denen ein Wechsel von torpfostenfarbenen (Farbe $(Y_{goal}, Cb_{goal}, Cr_{goal})$) Pixeln zu feldfarbenen (Farbe $(Y_{field}, Cb_{field}, Cr_{field})$), linienfarbenen (Farbe $(Y_{line}, Cb_{line}, Cr_{line})$), oder ballfarbenen (Farbe $(Y_{ball}, Cb_{ball}, Cr_{ball})$) Pixeln stattfindet. Für letztere drei Farbtripel wurden im bisherigen Verlauf der vorliegenden Arbeit bereits Methoden zu ihrer Ermittlung vorgestellt.

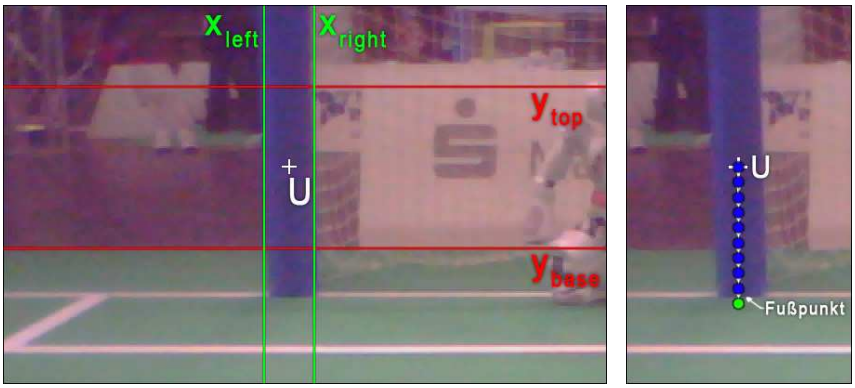
Für die konkrete Bestimmung des Fußpunktes eines gegebenen Bereiches werden zunächst die folgenden vier Variablen benötigt:

- die Koordinaten x_{left} und x_{right} des linken und rechten Extremwertes des zu untersuchenden Bereiches
- das vertikale Intervall $[y_{top}, y_{base}]$ der Scanlines

Aus diesen Werten wird ein für die Fußpunktsuche nötiger Startpunkt U mit den Koordinaten

$$U_x := \frac{1}{2}(x_{left} + x_{right}) \qquad U_y := \frac{1}{2}(y_{top} + y_{base})$$

ermittelt. Aufgrund der geometrischen Annahmen für Torpfosten (siehe Abschnitt 3.7.1) kann davon ausgegangen werden, dass U auf dessen Fläche liegt (siehe Abbildung 3.60a), sofern es sich bei dem zu untersuchenden Bereich tatsächlich um einen regelkonformen Torpfosten handelt .



(a) (b)

Abb. 3.60: Darstellung des Startpunktes U in Abhängigkeit der vier Torpfostenbegrenzungen (a) und Suche des Fußpunktes (b)

An dem Punkt U werden daher die im Weiteren benötigten Farbwerte $(Y_{goal}, Cb_{goal}, Cr_{goal})$ ausgelesen. Anschließend werden - ähnlich wie bei der Ermittlung der Kanten des Spielballs - entlang eines vertikal nach unten verlaufenden Strahls (siehe Abbildung 3.60b) die Farbwerte (Cb_{ray}, Cr_{ray}) der darauf liegenden Pixel sukzessiv bis zum Eintreten der Abbruchbedingung

$$D(Cb_{ray}Cr_{ray}, Cb_{goal}, Cr_{goal}) \geq D(Cb_{ray}Cr_{ray}, Cb_{field}, Cr_{field}) \ \wedge$$
$$D(Cb_{ray}Cr_{ray}, Cb_{goal}, Cr_{goal}) \geq D(Cb_{ray}Cr_{ray}, Cb_{line}, Cr_{line}) \ \wedge$$
$$D(Cb_{ray}Cr_{ray}, Cb_{goal}, Cr_{goal}) \geq D(Cb_{ray}Cr_{ray}, Cb_{ball}, Cr_{ball})$$

abgearbeitet, wobei nach wie vor durch die Funktion D der quadratische Farbabstand zweier (Cb, Cr)-Tupel berechnet wird. Es wird hierdurch der erste Pixel gesucht, dessen Farbwert näher an der Spielfeld-, Linien- oder Ballfarbe als an der Torfarbe liegt. Dieser Punkt wird als Fußpunkt F bezeichnet.

3.7.6 Plausibilitätsüberprüfungen

Zur Verringerung der recht hohen Falsch-Positiv-Rate bei der Torpfostenerkennung von 82,9% werden im Folgenden mehrere Regeln zur Detektion von Fehlerkennungen beschrieben.

Einschränkung der Fußpunktposition

Aufgrund der Feldgeometrie ist es naheliegend, dass sich die jeweiligen Fußpunkte der Torpfosten einige Pixel unterhalb der Spielfeldgrenze befinden müssen. Durch die Bedingung $F_y > \bar{y} + t_s$ kann die Falsch-Positiv-Rate für $t_s = 10$ auf 21,0% reduziert werden (siehe Abbildung 3.62a). Im behandelten Beispielbild 3.61 wird hierdurch der falsch erkannte, außerhalb des Spielfeldes liegende gelbe Torpfosten gefiltert.

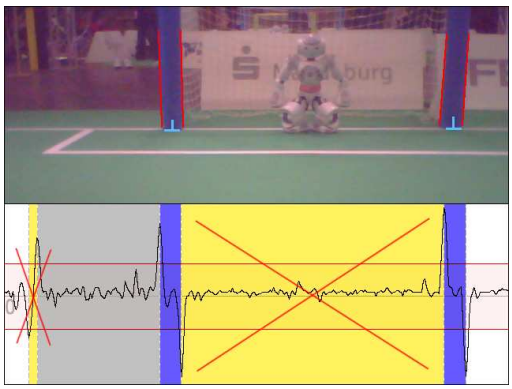


Abb. 3.61: Detektion und Ausschluss von falsch klassifizierten Bereichen

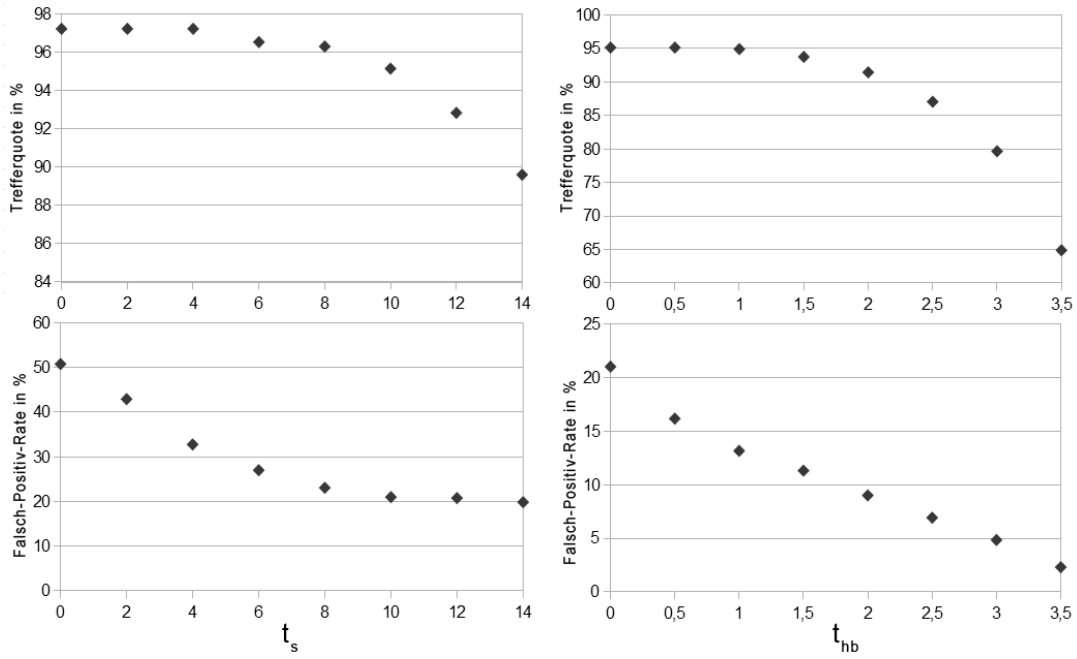
Begrenzung des möglichen Seitenverhältnis

Eine weitere Einschränkung der jeweiligen Bereiche wird durch Betrachtung einer groben Abschätzung ihrer jeweiligen minimalen Höhen-Breiten-Verhältnisse v_{hb} ermöglicht, wobei

$$v_{hb} := \frac{F_y - y_{top}}{x_{right} - x_{left}}$$

und ein Bereich nur dann als Torpfosten klassifiziert wird, wenn $v_{hb} > t_{hb}$ ist. Hierdurch werden - wie im Beispielbild 3.61 dargestellt - insbesondere Bereiche verworfen, die sich zwischen zwei korrekt erkannten Torpfosten befinden, da diese in der Regel hörere Breiten- als Höhen-Abmaße aufweisen. Bei der Wahl der Parameter v_s und v_{hb} geht man einen Kompromiss zwischen Trefferquote und Falsch-Positiv-Rate ein (siehe Abbildung 3.62): Für $v_{hb} = 2,0$ konnte beispielsweise eine Minderung der Falsch-Positiv-Rate auf 9,0% erreicht werden, jedoch sank hierbei auch die Trefferquote auf 91,5%.

Es sei erwähnt, dass in der Praxis die ermittelten Torpfostenpositionen - im Rahmen der Selbstlokalisierung des Roboters - mit der Anordnung der Spielfeldlinien abgeglichen und hierdurch üblicherweise alle restlichen fehlklassifizierten Bereiche gefiltert werden.



(a) (b)

Abb. 3.62: Trefferquote und Falsch-Positiv-Rate in Abhängigkeit der Parameter t_s (a) und t_{hb} (b)

4 Zusammenfassung der Ergebnisse

Im Hauptteil dieser Arbeit wurden Algorithmen zur Erkennung spezifischer Objekte im Roboterfußball entwickelt. Die folgenden Abschnitte zeigen zusammenfassend eine Übersicht zu den benötigten Programmlaufzeiten eines jeden Erkennungsschritts und in diesem Zusammenhang die Ergebnisse einer Analyse zur Anzahl der benötigten Pixelzugriffe entsprechender Methoden. Des Weiteren werden die jeweils erreichten Erkennungsraten für das verwendete Testbildmaterial aufgezeigt sowie Beispiele für erfolgreiche sowie fehlgeschlagene Objekterkennungen gegeben.

4.1 Programmlaufzeiten

Funktion	Laufzeit in ms			Anzahl Pixelzugriffe in %		
	min	avg	max	min	avg	max
Feldfarbenberechnung	0,72	0,85	0,99	0,59	0,59	0,59
Kantenerkennung	5,8	6,3	7,6	3,1	3,4	4,1
Spielfeldranderkennung	0,50	0,82	1,7	0	0	0
Linienerkennung	0,31	4,8	7,3	0	0,23	0,85
Spielballerkennung	0,66	3,2	4,3	3,1	3,1	3,2
Torerkennung	2,4	2,7	2,9	0,83	0,83	0,84
Gesamt	10,4	18,7	24,8	7,62	8,15	9,58

Tab. 4.1: Laufzeiten und relative Anzahl der benötigten Pixelzugriffe

Die Tabelle 4.1 zeigt zum einen Messungen der minimalen, der durchschnittlichen sowie der maximalen Verarbeitungszeiten für die sechs Hauptprogrammteile, die auf der NAO-Plattform während eines fünfminütigen Testspiels ermittelt wurden, zum anderen die jeweiligen minimalen, durchschnittlichen und maximalen prozentualen Anteile der Speicherzugriffe zum Auslesen der Pixelwerte in Bezug zur gesamten Bildgröße, die anhand der Testbilddatenbank bestimmt bzw. errechnet wurden. Die minimalen und maximalen Messwerte der Verarbeitungszeiten wurden infolge größerer Messfehler aufgrund parallel

laufender Prozesse aus dem Median der jeweils 1% kleinsten bzw. 1% größten Werte berechnet.

Die Berechnung der Feldfarbe benötigt unabhängig vom Bild immer die gleiche Anzahl an Pixelzugriffen, so dass auch nur eine geringe Veränderung zwischen minimaler und maximaler Laufzeit festgestellt werden kann. Da für alle drei Farbkanäle jeder sechzehnte (s_{Raster}) Pixel jeder Zeile und Spalte betrachtet wird, ergibt sich eine feste Anzahl von $3 * \frac{640}{16} * \frac{480}{16} = 3.600$ Speicherzugriffen in dem $640 * 480 * 2$ Byte großen Bild.

Dagegen bestehen bei der Kantenerkennung und Bereichsklassifizierung deutlich größere Schwankungen der Verarbeitungszeiten als auch eine variable Anzahl benötigter Speicherzugriffe - abhängig von der Anzahl gefundener Kantenpositionen.

Die Spielfeldranderkennung basiert ausschließlich auf der Auswertung der Ergebnisse der Kantenerkennung und Bereichsklassifizierung und bedarf keinen weiteren Pixelzugriffen.

Die Linienerkennung hingegen wertet zu jedem potentiellen Liniensegment mittels Sobel-Operator eine kleine Gruppe von Pixeln zur Richtungsbestimmung aus und weist daher - abhängig von der Anzahl jener Segmente - unterschiedliche Verarbeitungszeiten auf. Die Kantenerkennung und Linienerkennung erweisen sich als die rechenintensivsten Verarbeitungsschritte.

Das Verfahren zur Erkennung des Spielballs analysiert zunächst konstant viele Bildpunkte ($\frac{640}{4} * \frac{480}{4} = 19.200$) zur Positionsschätzung des Balls, benötigt jedoch anschließend eine variable Anzahl an Pixelzugriffen zur Auswertung der Objektform.

Ähnlich verhält es sich mit der Bestimmung der Torpfostenbereiche. Hierbei werden durch 16 Scanlines genau $16 * \frac{640}{2} = 5.120$ Pixelwerte zur horizontalen Positionsbestimmung der Pfosten analysiert und zusätzlich wird eine variable Anzahl an Pixelzugriffen bei der Suche der Fußpunkte verwendet.

4.2 Erkennungsraten und Beispielbilder

Erkennung von	Trefferquote	Falsch-Positiv-Rate
Feldfarbe	100,0%	0,3%
Feldgrenze	98,8%	1,5%
Linien	90,0%	4,6%
Spielball	99,2%	0,17%
Tor	91,5%	9,0%

Tab. 4.2: Trefferquote und Falsch-Positiv-Raten

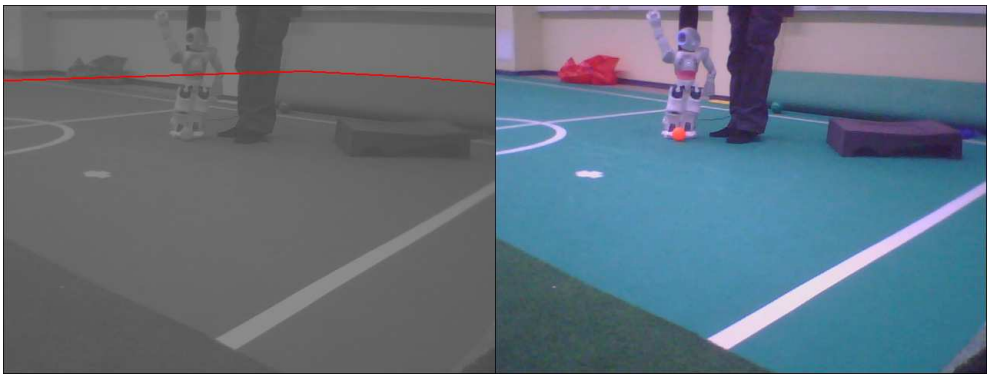
Tabelle 4.2 zeigt die jeweiligen Trefferquoten und Falsch-Positiv-Raten der Objekterkennungsalgorithmen. Sowohl bei der Feldfarbenbestimmung als auch bei der Ermittlung der Spielfeldgrenzen wird für jedes analysierte Bild angenommen, dass mindestens ein Teil des Spielfeldes abgebildet ist. Beide Verfahren treffen deshalb keine Entscheidung darüber, ob der Roboter gerade ein Spielfeld „sieht". Für alle Bilder, in denen kein Spielfeld abgebildet ist, würde deshalb eine Fehlerkennung die Folge sein. In der Testbilddatenbank ist dies in 2 von 600 Bildern der Fall, was die Falsch-Positiv-Rate der Feldfarbenerkennung von 0.3% erklärt.

Die nachfolgenden Beispielbilder demonstrieren zum einen die Erkennungsleistung der entwickelten Algorithmen, zum anderen sollen sie aber auch die Grenzen dieser aufzeigen. Auf der rechten Seite dieser Abbildungen befindet sich jeweils das Ausgangsbild, auf der linken Seite dementsprechend die durch den jeweiligen Algorithmus identifizierten Objekte (rot Markierungen).

(a) korrekte Erkennung

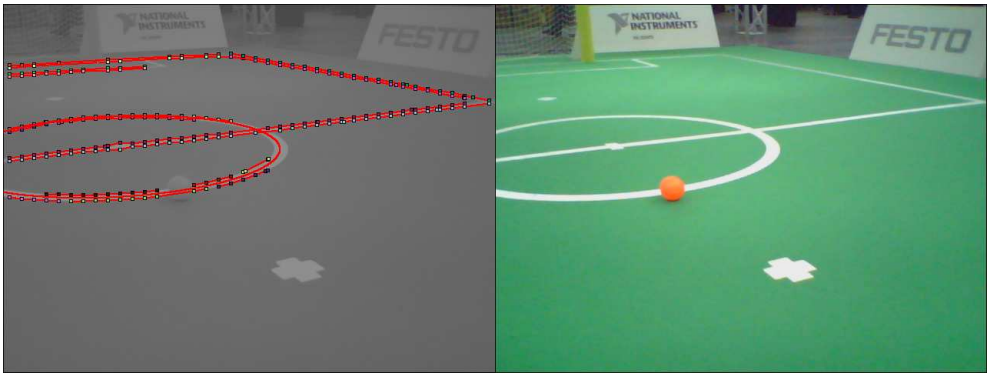
(b) korrekte Erkennung trotz benachbartem Spielfeld

(c) korrekte Erkennung trotz geringer sichtbarer Fläche

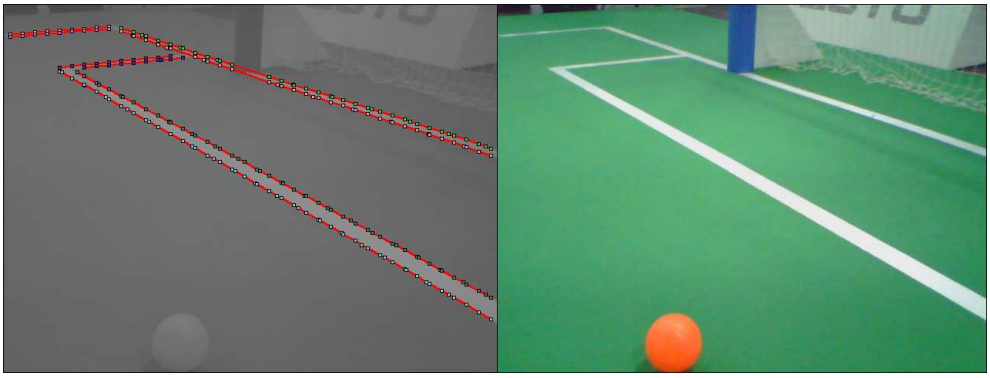


(d) Fehlerkennung aufgrund eines ähnlichfarbigen Teppichs am Spielfeldrand

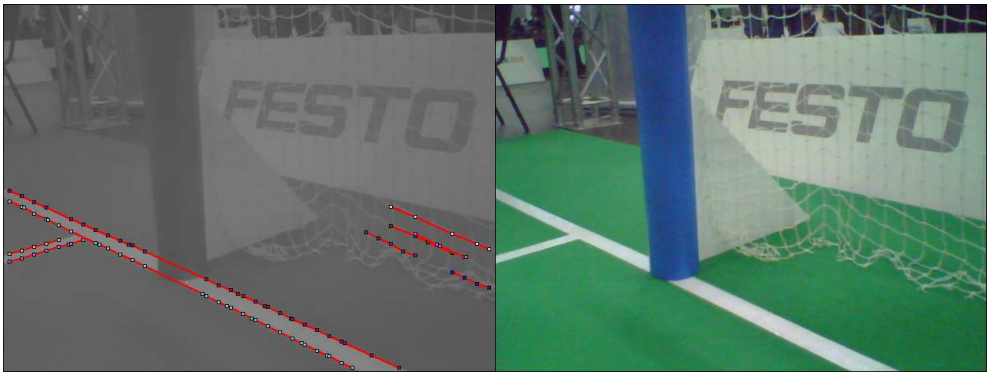
Abb. 4.1: Erkennung der Spielfeldgrenze



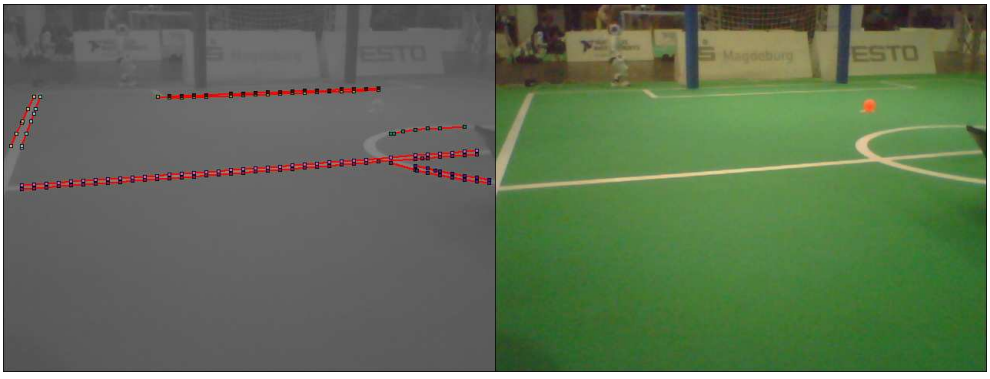
(a) korrekte Erkennung



(b) korrekte Erkennung



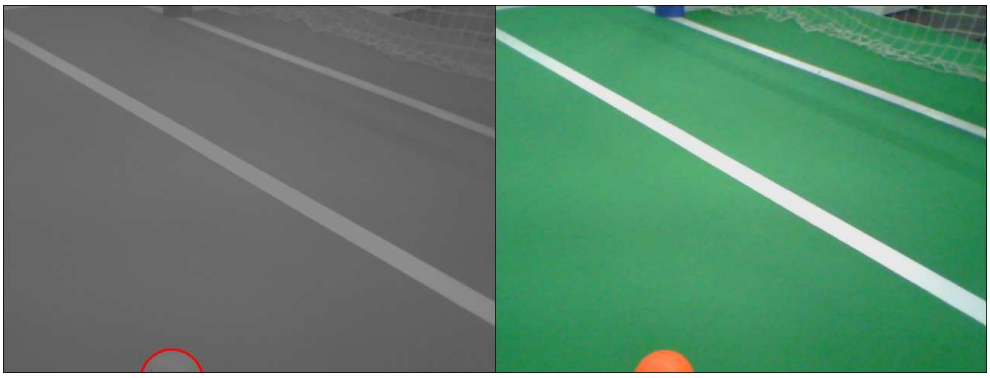
(c) Fehlerkennungen am Tornetz



(d) einige Linien wurden nicht erkannt

Abb. 4.2: Erkennung der Spielfeldlinien

(a) korrekte Erkennung trotz großer Entfernung des Balls



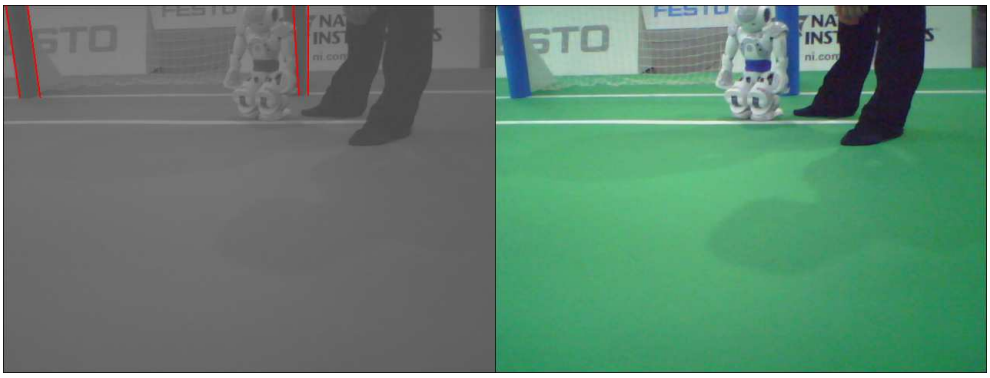
(b) korrekte Erkennung

(c) korrekte Erkennung trotz Überdeckung durch einen anderen Roboter

(d) Fehlerkennung aufgrund der ball-ähnlichen Form des Hüftbandes

Abb. 4.3: Erkennung des Spielballs

(a) korrekte Erkennung



(b) korrekte Erkennung trotz partieller Verdeckung des rechten Pfostens

(c) nur der rechte Pfosten wurde korrekt erkannt, da der linke Pfosten nicht vollständig im Bild ist

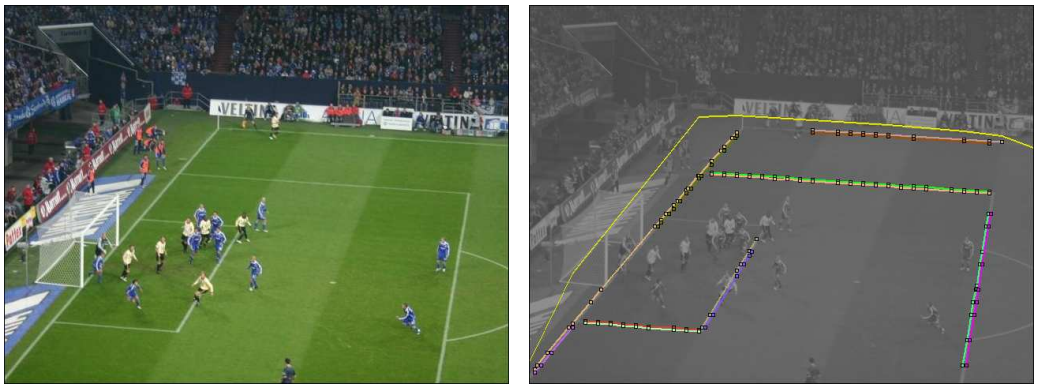
(d) Fehlerkennung aufgrund der Ähnlichkeit des Hosenbeins zu einem Torpfosten

Abb. 4.4: Erkennung der Torpfosten

5 Abschließende Bemerkungen

5.1 Ausblick

Die in dieser Arbeit entwickelten Algorithmen nutzen die im aktuellen Regelwerk des Robocup definierten Objekteigenschaften, um eine objektspezifische, zeiteffiziente Bildanalyse zu gewährleisten. Es ist zu erwarten, dass jene Regeln in den kommenden Jahren zunehmend anspruchsvoller gestaltet werden, um dem gemeinsamen Ziel im Jahr 2050 näher zu kommen. So werden sowohl variable und unvorhersehbare Lichtsituationen als auch die Aufhebung farbiger Objektmarkierungen in Zukunft eine wichtige Rolle spielen. Zur Bewältigung dieser Herausforderungen liegt deshalb die Betrachtung von Objektformen und weiteren farbunabhängiger Objekteigenschaften nahe. Die in dieser Arbeit beschriebenen Algorithmen wurden zwar speziell anhand der aktuell durch die Regeln des Robocup definierten Objekteigenschaften entwickelt, weisen aber dennoch einen gewissen allgemeinen Charakter auf. So können, wie nachfolgende Abbildung zeigt, die Spielfeld- und Linienerkennung ohne Anpassungen von Parametern bereits für ganz unterschiedliches Bildmaterial verwendet werden.



(a) (b)

Abb. 5.1: Spielfeldgrenzen- und Linienerkennung am Beispiel eines Fußballspiels

5.2 Danksagung

Während der Entwicklung der beschriebenen Algorithmen im Rahmen des Robocup hatte ich die Gelegenheit mit vielen herausragenden Menschen zu arbeiten, ohne deren Unterstützung die vorliegende Arbeit nicht hätte entstehen können.

Allen voran gilt mein Dank für die jederzeit engagierte und fachkompetente Betreuung Herrn Professor Karl-Udo Jahn, der mir nicht nur im Rahmen meiner Masterarbeit, sondern seit Beginn meines Informatik-Studiums mit zahlreichen fachlichen Diskussionen und Ratschlägen zur Seite stand.

Ebenso danke ich allen Mitgliedern des Nao-Team HTWK sowohl für die interessante und sehr inspirierende Zusammenarbeit als auch für die einzigartigen Tage und Nächte des Programmierens, Diskutierens und Organisierens um die RoboCup-Weltmeisterschaften.

Ein besonderer Dank gilt meiner Freundin Anne, die mir stets Kraft und Motivation während der Konzipierung meiner Arbeit gegeben hat und mir bei der Formulierung und der Korrektur der Texte zur Seite stand. Ohne Sie wäre diese Arbeit nicht möglich gewesen.

Abschließend möchte ich meinen lieben Eltern, Christine Reinhardt und Ludwig Reinhardt, dafür danken, dass sie mir das Studium und diese Masterarbeit nicht nur finanziell ermöglicht, sondern mich stets umfassend unterstützt haben.

Danke an alle für die moralische und technische Unterstützung, für die anregenden Gespräche und Inspirationen und für die spannende und interessante Zeit während meines Masterstudiums an der HTWK.

Literaturverzeichnis

[Ald11] Aldebaran-Robotics: *NAO, offical player of the RoboCup since 2008.* http://www.aldebaran-robotics.com/en/node/1168. Version: 08 2011

[Aly08] Aly, Mohamed: Real Time Detection of Lane Markers in Urban Streets. In: *IEEE Intelligent Vehicles Symposium*, 2008

[ÁMO05] Álvarez, Raziel ; Millán, Erik ; Oropeza, Ricardo S.: Multilevel Seed Region Growth Segmentation. In: *MICAI*, 2005, S. 359–368

[AR11] Aldebaran-Robotics: *Nao's Full Documentation - Camera.* http://users.aldebaran-robotics.com/docs/site_en/reddoc/hardware/video_camera.html. Version: 10 2011

[Bau08] Baumgartner, Josef: *Automatic color classification for application in RoboCup*, Technische Universität Darmstadt, Diplomarbeit, 2008

[BB05] Burger, Wilhelm ; Burge, Mark J.: *Digitale Bildverarbeitung : Eine Einführung mit Java und ImageJ.* 2., überarbeitete Auflage. Berlin, Heidelberg : Springer-Verlag, 2005

[BBV00] Bruce, J. ; Balch, Tucker ; Veloso, Maria M.: Fast and inexpensive color image segmentation for interactive robots. In: *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)* Bd. 3, 2000, S. 2061 – 2066

[Bir08] Birbach, Oliver: *Accuracy Analysis of Camera-Inertial Sensor-Based Ball Trajectory Prediction*, Mathematics and Computer Science, University of Bremen, Diplomarbeit, 2008

[BJ02] Bach, J. ; Jüngel, M.: Using pattern matching on a flexible, horizon-aligned grid for robotic vision. Concurrency, Specification and Programming. In: *CSP'2002* 1 (2002)

[CHH02] Campbell, Murray ; Hoane, A. J. Jr. ; Hsu, Feng-hsiung: Deep Blue. In: *Artif. Intell.* 134 (2002), January, 57–83. http://dx.doi.org/10.1016/

S0004-3702(01)00129-1. – DOI 10.1016/S0004–3702(01)00129–1. – ISSN 0004–3702

[Coa03] Coath, Genevieve: Adaptive arc fitting for ball detection in robocup. In: *In APRS Workshop on Digital Image Analysing*, 2003, S. 63–68

[ET03] Ekin, Ahmet ; Tekalp, A. M.: Robust dominant color region detection and color-based applications for sports video. In: *ICIP (1)*, 2003, S. 21–24

[Fas09] Fassbender, Torsten: *Computervision für humanoide Roboter*, Freien Universität Berlin, Diplomarbeit, 2009

[FB81] Fischler, Martin A. ; Bolles, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Commun. ACM* 24 (1981), Juni, Nr. 6, 381–395. http://dx.doi.org/10.1145/358669.358692. – DOI 10.1145/358669.358692. – ISSN 0001–0782

[FFM10] Furtig, Andreas ; Friedrich, Holger ; Mester, Rudolf: *Robust Pixel Classification for RoboCup - Farbworkshop 2010 Ilmenau*. http://217.92.194.243/fws2010/presentation/Fuertig_farbworkshop.pdf. Version: 2010

[GFBG05] Goeman, Werner ; Fonte, Leyden M. ; Bellens, Rik ; Gautama, Sidharta: Automated verification of road network data by VHR satellite images using road statistics. In: *ISPRS Hannover Workshop 2005 High Resolution Earth Imaging for Geospatial Information*, 2005

[GJMR10] Gioi, R. Grompone v. ; Jakubowicz, J. ; Morel, J. M. ; Randall, G.: LSD: A Fast Line Segment Detector with a False Detection Control. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (2010), April, Nr. 4, 722–732. http://dx.doi.org/10.1109/TPAMI.2008.300. – DOI 10.1109/TPAMI.2008.300. – ISSN 0162–8828

[Gra72] Graham, Ronald L.: An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. In: *Inf. Process. Lett.* 1 (1972), Nr. 4, S. 132–133

[HKP10] Hui Kong, Jean-Yves A. ; Ponce, Jean: General Road Detection from A Single Image. In: *IEEE Transactions on Image Processing*, 2010

[Hou62] Hough, Paul: *Method and Means for Recognizing Complex Patterns*. U.S. Patent 3.069.654, Dezember 1962

[HPV04] Hayet, J.B. ; Piater, J. ; Verly, J.: Incremental rectification of sports

fields in video streams with application to soccer. In: *Proc. of IEEE Conf. on Advanced Concepts for Intelligent Vision Systems (ACIVS'04)*. Brussels, 2004

[JHL04] Jüngel, Matthias ; Hoffmann, Jan ; Lötzsch, Martin: A Real-Time Auto-Adjusting Vision System for Robotic Soccer. In: *In 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences), Lecture Notes in Artificial Intelligence*, Springer, 2004, S. 214–225

[Jün04] Jüngel, Matthias: Using Layered Color Precision for a Self-Calibrating Vision System. In: *in The Eighth International RoboCup Symposium*, Springer, 2004, S. 209–220

[JSM+02] Jamzad, Mansour ; Sadjad, B. S. ; Mirrokni, V. S. ; Kazemi, Moslem ; Chitsaz, Hamid R. ; Heydarnoori, A. ; Hajiaghayi, Mohammad T. ; Chiniforooshan, Ehsan: A Fast Vision System for Middle Size Robots in RoboCup. In: *RoboCup 2001: Robot Soccer World Cup V*. London, UK : Springer-Verlag, 2002. – ISBN 3–540–43912–9, 71–80

[Ken84] Kendall, D. G.: Shape Manifolds, Procrustean Metrics, and Complex Projective Spaces. In: *Bulletin of The London Mathematical Society* 16 (1984), S. 81–121. http://dx.doi.org/10.1112/blms/16.2.81. – DOI 10.1112/blms/16.2.81

[KP07] Kerr, Douglas A. ; P.E.: *Derivation of the Cosine-Fourth-Law for Falloff of Illuminance Across a Camera Image*. http://www.dougkerr.net/pumpkin/articles/Cosine_Fourth_Falloff.pdf. Version: 2007

[LEC07] Lovell, Nathan H. ; Estivill-Castro, Vladimir ; Lima, Pedro (Hrsg.): *Color Classification and Object Recognition for Robot Soccer Under Variable Illumination*. Vienna, Austria: I-Tech Education and Publishing, 2007. – 71–94 S.

[LHB+09a] Laue, Tim ; Haas, Thijs J. ; Burchardt, Armin ; Graf, Colin ; Röfer, Thomas ; Härtl, Alexander ; Rieskamp, Andrik: Efficient and Reliable Sensor Models for Humanoid Soccer Robot Self-Localization. In: Zhou, Changjiu (Hrsg.) ; Pagello, Enrico (Hrsg.) ; Menegatti, Emanuele (Hrsg.) ; Behnke, Sven (Hrsg.) ; Röfer, Thomas (Hrsg.): *Proceedings of the Fourth Workshop on Humanoid Soccer Robots*, 978-88-95872-03-2, 2009

[LHB+09b] Laue, Tim ; Haas, Thijs J. ; Burchardt, Armin ; Graf, Colin ; Röfer, Thomas ; Härtl, Alexander ; Rieskamp, Andrik: Efficient and Reliable Sensor

Models for Humanoid Soccer Robot Self-Localization. In: ZHOU, Changjiu (Hrsg.) ; PAGELLO, Enrico (Hrsg.) ; MENEGATTI, Emanuele (Hrsg.) ; BEHNKE, Sven (Hrsg.) ; RÖFER, Thomas (Hrsg.): *Proceedings of the Fourth Workshop on Humanoid Soccer Robots*, 978-88-95872-03-2, 2009

[lig11a] *RoboCup 2D Soccer Simulation League*. http://en.wikipedia.org/wiki/RoboCup_2D_Soccer_Simulation_League. Version: 08 2011

[lig11b] *Simple Soccer Agent*. http://www.naoteamhumboldt.de/projects/simple-soccer-agent. Version: 08 2011

[Mar63] MARQUARDT, Donald W.: An Algorithm for Least-Squares Estimation of Nonlinear Parameters. In: *SIAM Journal on Applied Mathematics* 11 (1963), Nr. 2, 431–441. http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal&id=SMJMAP000011000002000431000001&idtype=cvips&gifs=yes

[MC04] MURCH, Craig L. ; CHALUP, Stephan K.: Combining edge detection and colour segmentation in the Four-Legged League. In: *Australian Robotics & Automation Association*, 2004

[McE07] MCENIRY, Charles: The Mathematics Behind the Fast Inverse Square Root Function Code. (2007)

[Med11] *MedianOfFive*. http://stackoverflow.com/questions/480960/code-to-calculate-median-of-five-in-c-sharp. Version: 10 2011

[nao11] *Nao-Team HTWK*. http://naoteam.imn.htwk-leipzig.de/. Version: 10 2011

[NC08] NGO, VietAnh ; CAI, Jianfei: Converting 2D soccer video to 3D cartoon. In: *ICARCV*, 2008, S. 103–107

[Nor05] NORTH, Alex: *Object recognition from sub-sampled image processing*, Diplomarbeit, 2005

[NYC10a] NGO, VietAnh ; YANG, Wenxian ; CAI, Jianfei: Accurate playfield detection using Area-of-Coverage. In: *ISCAS*, 2010, S. 3441–3444

[NYC10b] NGO, VietAnh ; YANG, Wenxian ; CAI, Jianfei: Accurate playfield detection using Area-of-Coverage. In: *ISCAS*, 2010, S. 3441–3444

[Ots79] OTSU, N.: A threshold selection method from gray-level histograms. In: *IEEE Transactions on Systems, Man and Cybernetics* 9 (1979), Januar, Nr. 1, S. 62–66

[Poy03a] Poynton, Charles: *Digital Video and HDTV Algorithms and Interfaces*. 1. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2003. – ISBN 1558607927

[Poy03b] Poynton, Charles: YUV and luminance considered harmful. In: *Digital video and HDTV algorithms and interfaces*, 2003

[Rei11] Reinhardt, Thomas: *Datenbank zum Testen kalibrierungsfreier Bildverarbeitungsalgorithmen zur echtzeitfähigen Objekterkennung im Roboterfußball*. http://www.thomas-reinhardt.de/Vision. Version: 10 2011

[RHH10] Ratter, Adrian ; Hengst, Bernhard ; Hall, Brad: rUNSWift Team Report 2010 - Robocup Standard Platform League. 2010. – Forschungsbericht

[RLM+09] 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/file_download/26/bhuman09_coderelease.pdf

[rob05] *Definition Robustheit*. http://de.wikipedia.org/wiki/Robustheit. Version: 2011 05

[Rob10a] RoboCup Technical Committee: *Laws of the RoboCup Small Size League 2010*. http://small-size.informatik.uni-bremen.de/_media/rules:ssl-rules-2010.pdf. Version: 2010

[Rob10b] RoboCup Technical Committee: *Middle Size Robot League - Rules and Regulations for 2010*. http://robocupmsl.googlegroups.com/web/msl-rules-2010-05-12.pdf. Version: 2010

[Rob10c] RoboCup Technical Committee: *RoboCup Soccer - Humanoid League - Rules and Setup for the 2010 competition in Singapore*. http://www.tzi.de/humanoid/pub/Website/Downloads/HumanoidLeagueRules2010_as_of_2010-03-02.pdf. Version: 2010

[Rob10d] RoboCup Technical Committee: *RoboCup Soccer - Simulation League - 3D Competition - Rules and Setup for the 2010 competition in Singapore*. http://homepages.feis.herts.ac.uk/~sv08aav/RCSoccerSim3DRules2010.2.pdf. Version: 2010

[Rob10e] RoboCup Technical Committee: *RoboCup Standard Platform League (Nao) Rule Book*. http://www.tzi.de/spl/pub/Website/Downloads/Rules2010.pdf. Version: Mai 2010

[Rob11] RoboCup Technical Committee: *Middle Size Robot League - Rules and Regulations for 2011*. http://robocupmsl.googlegroups.com/web/msl-rules-2010-12-31.pdf. Version: 2011

[Sch00] Scharr, Hanno: *Optimale Operatoren in der Digitalen Bildverarbeitung*. 2000

[SL09] Sun, Li ; Liu, Guizhong: Field lines and players detection and recognition in soccer video. In: *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. Washington, DC, USA : IEEE Computer Society, 2009 (ICASSP '09). – ISBN 978–1–4244–2353–8, 1237–1240

[Sán09] Sánchez, Tomás G.: *Artificial Vision in the Nao Humanoid Robot*, Rovira i Virgili University, Diplomarbeit, 2009

[Str05] Strutz, Tilo: *Bilddatenkompression. Grundlagen, Codierung, Wavelets, JPEG, MPEG, H.264*. Vieweg+Teubner, 2005

[TMS04] Troter, Arnaud L. ; Mavromatis, Sébastien ; Sequeira, Jean: Soccer Field Detection in Video Images Using Color and Spatial Coherence. In: *ICIAR (2)*, 2004, S. 265–272

[Tou85] Toussaint, Godfried T.: A Simple Linear Algorithm for Intersecting Convex Polygons. In: *The Visual Computer* 1 (1985), S. 118–123

[WWW99] Wesolkowski, Slawomir B. ; Wesolkowski, Slawomir B. ; Wesolkowski, Slawomir B.: *Color Image Edge Detection and Segmentation: A Comparison of the Vector Angle and the Euclidean Distance Color Similarity Measures*. 1999

A Inhalt der CD

Ordner	Inhalt
/Masterarbeit	enthält dieses Dokument als PDF-Datei
/Literatur	Literatur, die im PDF-Format zugänglich war
/Testbilder	600 Bilder aus typischen Szenen des Roboterfußballs
/Quelltext Java	Java-Umsetzung der in dieser Arbeit beschriebenen Algorithmen
/Quelltext Robocup	die komplette Roboter-Software in C++
/Multimedia	ausgewählte Bilder und Video vom Robocup

B Nützliche Funktionen

Im Folgenden ist eine Auswahl einiger Funktionen, durch die die Verarbeitungsgeschwindigkeit der Programme deutlich gesteigert werden konnte, dargestellt.

B.1 Inverse Wurzel

Bei der Normierung von Vektoren wurde folgende zeiteffiziente Berechnung der Inversen Wurzel ($y = x^{-\frac{1}{2}}$) verwendet [McE07]. Es sei jedoch darauf hingewiesen, dass dieses Verfahren gewisse numerische Ungenauigkeiten mit sich bringt, die jedoch keine messbaren Auswirkungen auf die Ergebnisse dieser Arbeit hatten.

Algorithm B.1

```
float invSqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;
    x2 = number * 0.5F;
    y = number;
    i = * ( long * ) &y;
    i = 0x5f3759df - ( i >> 1 );
    y = * ( float * ) &i;
    y = y * ( threehalfs - ( x2 * y * y ) );
    return y;
}
```

B.2 Median aus drei Elementen

Algorithm B.2 medianOfThree

```
float medianOfThree(float a, float b, float c)
{
    return a > b ? a < c ? a : b < c ? c : b : b < c ? b : a < c ? c : a;
}
```

B.3 Median aus fünf Elementen

Eine bezüglich der Rechenzeit annäherend optimale Bestimmung des Medians aus fünf Werten ist mit nachfolgendem C-Quelltext möglich. Hier werden zur Ermittlung des Ergebnis ausschließlich sechs Vergleiche benötigt.

Algorithm B.3 medianOfFive [Med11]

```
float medianOfFive(float a, float b, float c, float d, float e)
{
    return b < a ? d < c ? b < d ? a < e ? a < d ? e < d ? e : d
                                                 : c < a ? c : a
                                       : e < d ? a < d ? a : d
                                                 : c < e ? c : e
                                  : c < e ? b < c ? a < c ? a : c
                                                 : e < b ? e : b
                                       : b < e ? a < e ? a : e
                                                 : c < b ? c : b
                             : b < c ? a < e ? a < c ? e < c ? e : c
                                                 : d < a ? d : a
                                       : e < c ? a < c ? a : c
                                                 : d < e ? d : e
                                  : d < e ? b < d ? a < d ? a : d
                                                 : e < b ? e : b
                                       : b < e ? a < e ? a : e
                                                 : d < b ? d : b
                        : d < c ? a < d ? b < e ? b < d ? e < d ? e : d
                                                 : c < b ? c : b
                                       : e < d ? b < d ? b : d
                                                 : c < e ? c : e
                                  : c < e ? a < c ? b < c ? b : c
                                                 : e < a ? e : a
                                       : a < e ? b < e ? b : e
                                                 : c < a ? c : a
                             : a < c ? b < e ? b < c ? e < c ? e : c
                                                 : d < b ? d : b
                                       : e < c ? b < c ? b : c
                                                 : d < e ? d : e
                                  : d < e ? a < d ? b < d ? b : d
                                                 : e < a ? e : a
                                       : a < e ? b < e ? b : e
                                                 : d < a ? d : a;
}
```