Bachelorarbeit

Anwendung von maschinellem Lernen zur echtzeitfähigen und kalibrierungsfreien Erkennung von humanoiden Fußballrobotern

Martin Engel
Medieninformatik
51257
8. August 2012

Selbstständigkeitserklärung

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

Leipzig, den 8. August 2012

Danksagung

Ich möchte mich zuerst bei Prof. Dr. Siegfried Schönherr bedanken, dessen Anmerkungen diese Arbeit bereichert haben. Seine Ideen veranlassten mich zum Nachdenken und ermöglichten das vorliegende Resultat.

Weiterhin bedanke ich mich bei M. Sc. Thomas Reinhardt, welcher mit seiner Bildverarbeitung den Grundstein legte und dessen Ideen die Erkennung voran brachten. Ohne ihn wäre diese Arbeit nicht möglich gewesen.

Nicht zu vergessen Juliane Adler und meine Mutter Hannelore Engel, welche beide mit Adleraugen die Arbeit Korrektur gelesen haben. Ohne sie wären viele Fehler unerkannt geblieben.

Auch bedanke ich mich bei meiner langjährigen Freundin Juliane Scholze, welche mir zur Seite gestanden und meine Laune gehoben hat, wenn mal etwas nicht so funktioniert hat wie es gedacht war.

Ich möchte mich bei allen meinen Freunden bedanken, die diese Arbeit gelesen, kommentiert oder mir mit ihren Worten Mut zugesprochen haben.

Vielen Dank!

Vielen Dank an alle, die diese Arbeit ermöglichten!

Kurzfassung

Bei einem Fußballspiel ist es unentbehrlich, alle Spieler auf dem Feld identifizieren und deren Position bestimmen zu können. Dies gilt sowohl für ein Spiel zwischen Menschen als auch zwischen Robotern. Das Thema dieser Arbeit ist die visuelle Erkennung humanoider Fußballroboter mittels der im Kopf der Roboter integrierten Kameras. Es werden Merkmale ermittelt, die Regionen in diesen Kamerabildern beschreiben und eine Unterscheidung zwischen Robotern und anderen Hindernissen ermöglichen. Anschließend lernt ein Programm diese Merkmale und kann eine Aussage darüber zu treffen, ob es sich bei einem gegebenen Bildbereich um einen Roboter handelt oder nicht.

Inhaltsverzeichnis

1. Vorbemerkungen

Im Rahmen dieser Arbeit soll ein Algorithmus entwickelt werden, der auf einem Kamerabild Roboter erkennen und von anderen Objekten unterscheiden kann. Einsatzgebiet wird der Robocup[1] sein, bei dem es sich um eine weltweite Veranstaltung handelt, welche sich vorrangig dem Fußballspiel zwischen Robotern widmet.

1.1. Motivation

Die Grundlage für ein Fußballspiel ist das Erkennen der Mitspieler. Ein Spieler muss in der Lage sein, eigene Teammitglieder als auch Gegner ausfindig zu machen und entsprechend deren Position, aktueller Bewegungsrichtung und Handlung sowie der Spielsituation Entscheidungen zu treffen. Dies ist eine Voraussetzung für das Spiel sowohl zwischen Menschen, als auch für humanoide Roboter. Ohne akkurate Informationen über das aktuelle Spielgeschehen ist eine Beurteilung der weiteren Verfahrensweise nur eingeschränkt möglich.

In der aktuellen Softwarelösung des Nao Teams der HTWK Leipzig findet sich noch keine adäquate Umsetzung dieses brisanten Themas. Derzeit lässt sich erkennen, dass sich Objekte im Kamerabildbereich befinden, allerdings kann weder deren exakte Anzahl, noch Art und Größe der Objekte genau bestimmt werden. Dies hat zur Folge, dass Hindernisse, für welche bisher keine spezialisierte Erkennung existiert[2], nicht in die Entscheidungsfindung einbezogen werden können und somit auch keinen Einfluss auf das Spielgeschehen nehmen.

Wie zuvor angedeutet, ist dieser Ansatz unzureichend und birgt auch Gefahren. Das Regelwerk der Robocup Standard Platform League[3] sieht vor, dass Roboter, welche andere Spieler bedrängen und behindern, für 30 Sekunden aus dem Spiel genommen werden. In

[1] Hierzu ausführlich siehe Kapitel 2.1.

[2] Bisher werden Tore, der Spielfeldrand und der Spielball mit Hilfe von angepassten Algorithmen im Bild erkannt. Weitere Hinderniserkennungen sind bisher nicht implementiert, oder werden nicht im produktiven Einsatz verwendet.

[3] Dies ist jene Liga, an welcher die in dieser Arbeit besprochenen Fußballroboter teilnehmen. Weitere Informationen zu dieser Liga sind in Kapitel 2.1 zu finden.

diesem Zeitraum ist das betreffende Team in Unterzahl und muss möglicherweise spielentscheidende Wendungen hinnehmen. Zudem wird nicht verhindert, dass Spieler des gleichen Teams sich gegenseitig behindern oder den Weg versperren.

In den vergangenen Jahren hat sich diese Lücke in der Entscheidungsfindung nicht auffallend negativ auf die Spielerfolge ausgewirkt[4]. Die Notwendigkeit zur Unterstützung bei der Lokalisierung der eigenen Position bestand nicht, da verschiedenfarbige Tore die eigene Spielhälfte bereits genau definierten. Das Pushing, also das Behindern gegnerischer Roboter, konnte durch den Einsatz einer speziellen Armposition, wie in Abbildung 1.1 dargestellt, minimiert werden und auch der eigentliche Spielfluss, wie das Auffinden und Bewegen des Balles, wurde nicht außerordentlich beeinflusst. Ein Grund hierfür ist, dass bisher nur wenige Teams der RoboCup Standard Platform League eine effiziente Robotererkennung im produktiven Einsatz haben.[5] Lediglich die Strategiefindung und das eingeschränkte Teamplay[6] zeigte die Notwendigkeit einer Objekterkennung.

Mit Beginn des Jahres 2012 ist die Priorität für eine spezialisierte Objekterkennung radikal gestiegen. Grund hierfür ist eine Regeländerung der RoboCup Standard Platform League[7], welche gleichfarbige, gelbe Tore vorsieht. Dies erschwert die Bestimmung der eigenen Spielhälfte und Position, was sowohl die Entwicklung neuer, als auch die Anpassung bestehender Lokalisierungsalgorithmen erfordert.[8] Um beide Felder zu erkennen und voneinander zu unterscheiden, ist es daher notwendig, diese mit einzigartigen Merkmalen zu versehen. Dazu kann man beispielsweise das Vorhandensein des eigenen oder gegnerischen Torwartes, die Positionen der Mitspieler oder Gegenstände am Rand des Spielfeldes verwenden.[9]

[4] Alle Titel im Überblick:
RoboCup German Open 2009 in Hannover: 2. Platz.
RoboCup 2009 in Graz: 1. Platz, sowie 1. Platz in der Technical Challenge.
RoboCup German Open 2010 in Magdeburg: 5. Platz.
RoboCup 2010 in Singapur: Viertelfinale, sowie 1. Platz in der Open Challenge.
RoboCup German Open 2011 in Magdeburg: 3. Platz.
RoboCup 2011 in Istanbul: 4. Platz.
RoboCup German Open 2012 in Magdeburg: 4. Platz.
RoboCup 2012 in Mexico: 4. Platz.

[5] Es gibt Erkennungen, welche auf Farbtabellen basieren, allerdings benötigen diese eine Kalibrierung vor jedem Spiel und sind nicht robust gegenüber Änderungen der Lichtverhältnisse.

[6] So könnte man beispielsweise die Lokalisierung unterstützen, indem Hypothesen über die eigene Position visuell durch Mitspieler bestätigt werden. Auch sind komplexere Strategien möglich, wenn bekannt ist, wo sich aktuell Gegner befinden.

[7] Nähere Informationen zu dieser Liga sind in Kapitel 2.1 zu finden.

[8] Beide Spielhälften sind visuell nicht mehr voneinander unterscheidbar, wodurch das Wissen, in welcher Hälfte man sich befindet, von anderen Quellen bezogen werden muss.

[9] Diese Gegenstände dürfen nicht durch das jeweilige Team beabsichtigt sein, da dies eine unerlaubte, direkte Beeinflussung des Spielgeschehens bedeuten würde. Es ist allerdings erlaubt, den Außenrand des Spielfeldes in Bereiche mit individuellen Merkmalen einzuteilen, um einen Roboter bei der ei-

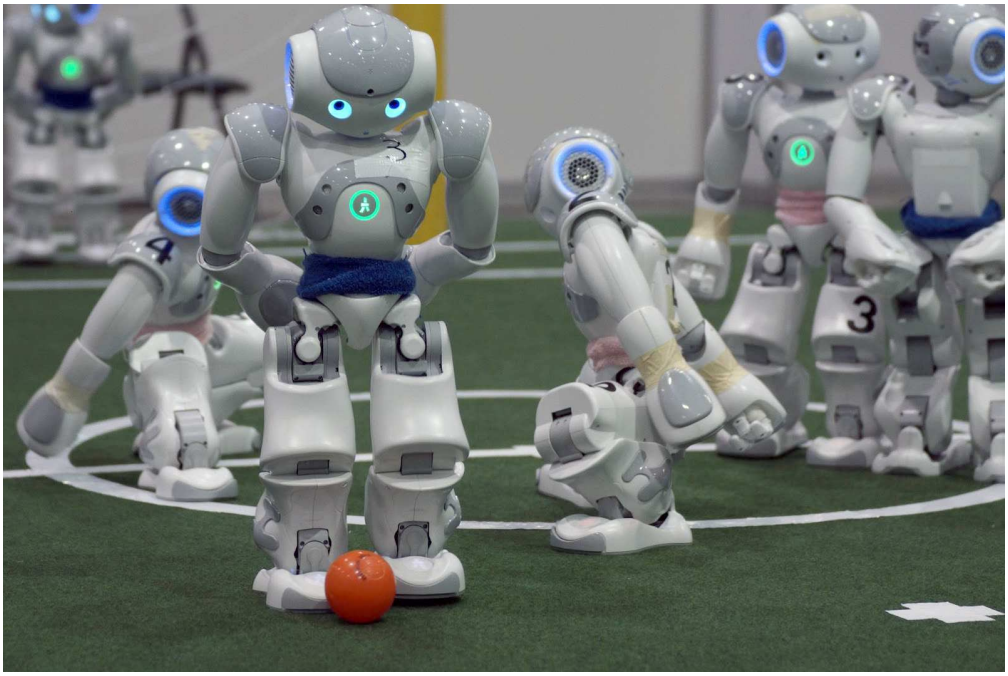


Abbildung 1.1. – Die Darstellung zeigt die spezielle Armposition der Roboter. Sie verkleinert die Fläche, so dass die Arme nicht stören, verringert die Wahrscheinlichkeit, sich selbst und andere zu behindern und sie schützt bei Stürzen.

1.2. Zielsetzung und Abgrenzung

Das primäre Ziel dieser Arbeit besteht in der Entwicklung einer spezialisierten Objekterkennung, welche Roboter von anderen Hindernissen unterscheiden kann.

Diese Informationen bilden anschließend sowohl die Grundlage für intelligente Entscheidungen auf Basis der aktuellen Spielsituation, als auch für ausgefeiltere Lokalisierungs- und Teamplaystrategien, welche allerdings nicht Inhalt dieser Arbeit sind.

Dazu soll ein Klassifizierungsalgorithmus entwickelt werden, welcher in der Lage ist, über ein gegebenes Bild eine Aussage zu treffen, ob und falls vorhanden wie viele Roboter abgebildet sind. Weitere Ausbaustufen bilden die Bestimmung der Positionen dieser Objekte auf dem Spielfeld, die Unterscheidung zwischen Gegnern und Mitspielern, deren aktuelle Bewegungsrichtung unter Berücksichtigung vergangener Bilddaten sowie der Erkennung der momentanen Tätigkeit eines jeden erkannten Spielers, sofern es sich nicht um Mitspieler handelt.

genen Lokalisierung mit Hilfe des Feldrandes zu unterstützen. Eine Implementierung einer solchen Lokalisierung ist bei dem Nao-Team der HTWK Leipzig in Arbeit.

Eine zusätzliche Bedingung für den Algorithmus sowie die Extraktion notwendiger Bildinformationen ist eine performante Ausführung. Da die Nao-Roboter von Aldebaran nur über begrenzte Hardwareressourcen verfügen[10] und die Erkennung von Objekten zwischen zwei Bildintervallen erfolgen muss[11], darf die Laufzeit wenige Millisekunden nicht übersteigen. Dadurch ergibt sich die Prämisse, dass sowohl die Berechnung von Merkmalen in Bildregionen, als auch deren Klassifizierung auf einem Roboter in einer Geschwindigkeit möglich sein muss, welche als *echtzeitfähig* zu bezeichnen ist.

Eine weitere Prämisse ist, dass die Erkennung keinerlei Kalibrierung voraussetzt. Ein Teil der am RoboCup teilnehmenden Teams arbeitet mit sogenannten Farbtabellen. Dabei werden diversen Tonwertintervallen eine distinkte Anzahl an Farben zugeordnet. Beispielsweise erfolgt eine pauschale Zuordnung eines für das Spielfeld repräsentativen Farbtons auf alle Grüntöne. Dadurch sinkt der Informationsgehalt eines Bildes, aber im Gegenzug wird eine höhere Verarbeitungsgeschwindigkeit erreicht. Die Kalibrierung mittels Farbtabellen wird vor jedem Spiel unter zeitlichem Aufwand durchgeführt und bildet die Basis für die weitere Bildverarbeitung. Der fundamentale Nachteil dieser Methode ist neben der notwendigen Zeit die Abhängigkeit von den aktuellen Lichtverhältnissen. Sollten diese sich während eines Spiels ändern, kann es vorkommen, dass die bildverarbeitenden Prozesse nicht mehr akkurat arbeiten oder gänzlich falsche Ergebnisse liefern. Daher verzichtet die bisherige Bildverarbeitung des Nao Teams der HTWK auf diese Form der Verarbeitung und arbeitet stattdessen auf der Grundlage der Erkennung visueller Merkmale, wie beispielsweise Kanten im Bild, welche eine lichtunabhängige Bildauswertung ermöglichen. Dieser Ansatz soll auch weiterhin beibehalten werden, weshalb eine kalibrierungsfreie Implementierung der Erkennung von Robotern zu ermitteln ist.

Für die Umsetzung werden Methoden des maschinellen Lernens, der Bildverarbeitung, sowie der Mustererkennung zum Einsatz kommen. Diese sollen nur so viele Berechnungen wie unbedingt notwendig nach sich ziehen, um im Hinblick auf die begrenzten Ressourcen der Nao-Plattform[12] eine effektive Implementierung zu gewährleisten. Daher liegt der Fokus nicht auf der optimalen Erkennung, sondern auf der effizienten Ressourcennutzung.

Die Entwicklung und Implementierung der Ausbaustufen bilden keinen Bestandteil dieser Arbeit. Das Resultat der Klassifizierung besteht in der Aussage, wie viele Roboter im Bild erkannt worden sind und an welcher Position im Bild sich diese befinden. Die Entwicklung weiterer Funktionalitäten ist direkt abhängig vom verbleibenden Zeitrahmen.

[10] Siehe Kapitel 2.1.
[11] Zusammen mit weiteren Berechnungen, welche für das Spielgeschehen wichtig sind.
[12] Vgl. Kapitel 2.3.

Falls mehrere Roboter hintereinander stehen und somit nur unter zusätzlichem Rechenaufwand auftrennbar sind, müssen diese nicht einzeln erkannt werden. Die Aussage, dass sich in dieser Region mindestens ein Roboter befindet, ist ausreichend. Weitere Details ergeben sich sowohl durch Informationen von Mitspielern aus anderen Perspektiven, als auch in späteren Frames. Somit ist eine zusätzliche Ressourcennutzung zur genaueren Differenzierung von Robotern innerhalb eines Bildes nicht notwendig und damit nicht erwünscht. Des Weiteren soll es wichtiger sein, dass keine Objekte fälschlicherweise als Roboter erkannt werden, da Falschinformationen diesbezüglich größere negative Auswirkungen auf das Spielgeschehen haben als ein nicht erkannter Spieler.

1.3. Aufbau der Arbeit

Das nachfolgende Kapitel 2 gibt eine Einführung in das Thema des Roboterfußballs. Es befasst sich mit dem RoboCup als zentralen Wettbewerb, seinen einzelnen Ligen und Zielen. Des Weiteren werden alle notwendigen Grundlagen besprochen, welche zum Verständnis dieser Arbeit notwendig sind und es ermöglichen, die Ergebnisse in den passenden Kontext zu setzen. Darunter sind beispielsweise die Regeln des Spielfeldes, die Hardwareeigenschaften der Roboterplattform *Nao* sowie der *YCbCr*-Farbraum, mit welchem die Kameras arbeiten.

Der Hauptteil in Kapitel 3 enthält Informationen zur Testbilddatenbank, deren Aufbau, Zusammensetzung und Nutzung. Im Unterkapitel 3.2 werden alle Merkmale beschrieben, welche für jede Region in einem Bild bestimmt werden, die potenziell einen Roboter darstellen könnte. Dazu zählt sowohl deren Ermittlung, Normalisierung, als auch Evaluierung. Ab Kapitel 3.3 wird aufgezeigt, wie diese Merkmale anhand von Beispielen gelernt, abstrahiert und zur Klassifizierung von unbekannten Bildbereichen genutzt werden.

Abschließend stellt Kapitel 4 eine Zusammenfassung der Ergebnisse dar, welche im Rahmen dieser Arbeit entstanden sind. Auch werden die endgültigen Erkennungsraten genannt und ein Ausblick gegeben, was mit diesen möglich ist. Zukünftige Auswirkungen auf das Zusammenspiel zwischen den einzelnen Robotern, die Lokalisierung innerhalb des Feldes, aber auch Problematiken sind Inhalt dieses letzten Abschnittes.

2. Grundlagen

2.1. Der RoboCup

Der RoboCup wurde geschaffen, um die Robotik und Forschung an künstlicher Intelligenz der Öffentlichkeit näher zu bringen und eine Herausforderung zu bieten, an der sich internationale Teams messen können.[13]

Um dem RoboCup eine langfristige Teilnahmemotivation zu geben, sowie eine konkrete Richtung aufzuzeigen, wurde ein „ultimatives Ziel" definiert. Im Jahr 2050 soll ein Team, bestehend aus vollkommen autonomen Fußballrobotern, ein Fußballspiel nach den Regeln der FIFA gegen den dann aktuellen Weltmeister spielen und gewinnen. Dieses Ziel soll ein Ansporn sein, über die aktuelle Technologie hinaus Ideen zu verwirklichen und die Forschung der Robotik und künstlichen Intelligenz auf eine neue Ebene zu bringen.

Der RoboCup bietet die Möglichkeit, unter realen Wettkampfbedingungen Theorien, Algorithmen und Architekturen zu evaluieren. Er wurde geschaffen, um komplexe, reale Probleme in einer distinkten Umgebung zu bearbeiten, ohne dabei die sonst notwendigen hohen Forschungs- und Entwicklungskosten zu besitzen. Es werden Forschungsbereiche unter realen Bedingungen abgedeckt, welche unter anderen Umständen kostenintensiv werden können, darunter beispielsweise Multi-Agenten Systeme, Kontexterkennung, Computer Vision und reaktives Verhalten.

Der RoboCup besitzt neben dem Fußball weitere Bereiche, welche im Laufe der Jahre hinzugekommen sind, darunter zählen der *RoboCup Rescue* zur Entwicklung von Robotern für den Notfalleinsatz, *RoboCup@Home* für Roboter, die Alltagsaufgaben übernehmen können und für Schüler der *RoboCup Junior*, bei welchem es beispielsweise Wettbewerbe mit der Lego Mindstorms Plattform[14] gibt.

Der Bereich für den Roboterfußball, genannt *RoboCup Soccer*, ist selbst nochmals unterteilt in verschiedene Ligen. Die Zugehörigkeit zu einer dieser Ligen ist abhängig von der Größe der Roboter, deren Bauart und ob diese selbst erstellt oder standardisiert sind.

[13] Vgl. [Fed12].

[14] Lego Mindstorms ist eine Marke von Lego, welches programmierbare Bauteile beinhaltet. Diese sind unter anderem mittels einer speziellen Abwandlung der Programmiersprache C ansprechbar.

Die **Small Size** Liga ist eine der ältesten Ligen. Sie ist auch bekannt als *F180* Liga und konzentriert sich vornehmlich auf die Problematik von Multi-Agenten Systemen in einer dynamischen Umgebung. Das Spielfeld ist lediglich 6×4 Quadratmeter groß und wird von zwei Teams mit jeweils fünf Robotern bespielt. Diese besitzen standardmäßig zur Fortbewegung Räder und werden von einem zentralen Computer gesteuert. Zur Orientierung ist über dem Spielfeld eine Kamera angebracht, welche die aufgenommenen Bilder an einen Server sendet. Die Erkennung der Roboter gestaltet sich in dieser Liga sehr einfach – deren Positionen sind anhand von farbigen Markierungen, welche auf den Spielern angebracht sind, ermittelbar.[15]

Die nächstgrößere Liga ist die **Middle Size**. Hier spielen bis zu sechs Roboter pro Team auf einem 18×12 Quadratmeter großen Spielfeld mit einem Ball, der den Standardmaßen der FIFA entspricht. Eingesetzte Roboter folgen keinen Standardisierungen, sondern müssen lediglich Vorgaben erfüllen, wie beispielsweise eine maximale Höhe von 80cm und ein Gewicht von weniger als 40 kg. Sie werden von den Teilnehmern selbst entwickelt und sind demzufolge in Bezug auf Sensorik und Leistungsfähigkeit unterschiedlich ausgestattet. Die Kommunikation mit Mitspielern und Schiedsrichtercomputer ist technologisch auf WLAN beschränkt. Der Forschungsfokus dieser Liga liegt auf Autonomie und Verhalten, daher darf nach Spielbeginn kein externer Einfluss mehr ausgeübt werden.

Im Gegensatz zu den bisher vorgestellten Ligen kommen bei der **Humanoid** Liga Roboter zum Einsatz, welche einen dem Menschen nachgeahmten Aufbau haben. Dementsprechend müssen Problemstellungen gemeistert werden, welche auch Kleinkinder vor eine Herausforderung stellen. Dazu zählt der aufrechte, zweibeinige Gang, das Sicherstellen der Balance, die Orientierung innerhalb der Umgebung, Teamplay sowie die visuelle Wahrnehmung der Spieler, des Balles und des Spielfeldes. Gespielt wird auf einem 6×4 Quadratmeter großen Feld zu je drei Robotern pro Team. Diese müssen ebenfalls autonom agieren und die Kommunikation erfolgt ebenfalls ausschließlich über WLAN. Die Humanoid Liga wird unterteilt in drei Unterligen, wobei die Höhe der selbstgebauten Roboter das entscheidende Kriterium darstellt. Es gibt die *Kid Size* mit maximal 60 cm, die *Teen Size* bis 120 cm und die *Adult Size* ab 120 cm.

Eine Ausnahme beim RoboCup ist die **Simulationsliga**. Bei dieser kommen keine realen Roboter zum Einsatz, sondern lediglich ihre digitalen Gegenstücke. Sie ist eine der ältesten am RoboCup vertretenen Ligen und legt den Fokus der Forschung auf die künstliche Intelligenz, sowie Teamstrategien. Elf unabhängige Spieler je Team, Agenten genannt, spielen Fußball auf einem virtuellen Spielfeld, wobei zwischen zweidimensionalen und dreidimensionalen Simulationen unterschieden wird.

[15] Vgl. [Fed12] und [Rei11] ab Seite 10.

Die für diese Arbeit relevante Liga ist die **Standard Plattform Liga**, auch kurz *SPL* genannt. Hier kommen im Gegensatz zu den bisher vorgestellten Ligen keine selbstgebauten Roboter zum Einsatz, sondern eine von der französischen Firma Aldebaran[16] gebaute Standardplattform namens *Nao*. Die genauen Spezifikationen dieses Roboters sind in Kapitel 2.3 zu finden.

Die SPL fokussiert sich gänzlich auf die Entwicklung der Software, ohne dabei den Vorteil von zeitgemäßer Robotertechnologie zu missen und ersetzt die vorherige *Four-Legged* Liga, welche auf Sonys *AIBO* basierte. Der Vorteil der SPL ist, dass entwickelte Lösungen nicht nur simuliert, sondern unter realen Bedingungen getestet und verglichen werden, ohne dass auf die Entwicklung von Robotern Wert gelegt werden muss. Ohne diese RoboCup-Kategorie wäre ein objektiver Test und Vergleich von Algorithmen nicht möglich, da entweder unterschiedlich entwickelte und leistungsfähige Roboter das Ergebnis verfälschen oder innerhalb einer Simulation die Umwelteinflüsse und Ungenauigkeiten der Motorik nicht akkurat abgebildet werden können.

Dies birgt zahlreiche Vorteile gegenüber den zuvor vorgestellten Ligen. Entwicklungen können unter den verschiedenen Teams ausgetauscht und verbessert werden. Über die Teilnehmergrenzen hinaus ist eine Zusammenarbeit möglich, da alle von der gleichen Plattform ausgehen. Lösungen sind wiederverwendbar und ermöglichen einen schnellen Einstieg von neuen Teams in die RoboCup-Domain. Der wichtigste Vorteil dieser Liga ist sowohl die Zeit- als auch Kostenersparnis zur Entwicklung von funktionsfähigen Programmen.

Die SPL besitzt ein sehr breites Spektrum und deckt zahlreiche Problemstellungen ab, darunter die dynamische, zweibeinige Fortbewegung, die Analyse der Umgebung, aktuellen Spielsituation und Ballbewegung, sowie die Selbstlokalisierung und Teamstrategie.[17]

2.2. Das Spielfeld

Das Spielfeld in der SPL folgt fest definierten Regeln. So wird sichergestellt, dass alle Teams mit den gleichen Voraussetzungen antreten und, da der RoboCup ein internationaler Wettbewerb ist, es keine regionalen Unterschiede gibt.

Der Teppich, auf dem gespielt wird, hat im Ganzen eine Länge von 7,4 Metern und eine Breite von 5,4 Metern, wobei das eigentliche Spielfeld mit 6×4 Quadratmetern kleiner ausfällt. Die Abbildung 2.1 zeigt, wie ausgearbeitet die Spezifikationen des Spielfeldes sind. Die Spielfeldfarbe ist grün, wobei der eigentliche Farbton nicht näher spezifiziert

[16] Siehe hierzu `http://www.aldebaran-robotics.com/en/`.

[17] Das NAO-Team der HTWK Leipzig setzte von 2009 bis 2012 den Fokus auf die Fortbewegung, Umgebungsanalyse und Lokalisierung. Seit Mitte 2012 liegt das Hauptaugenmerk auf der Teamstrategie.

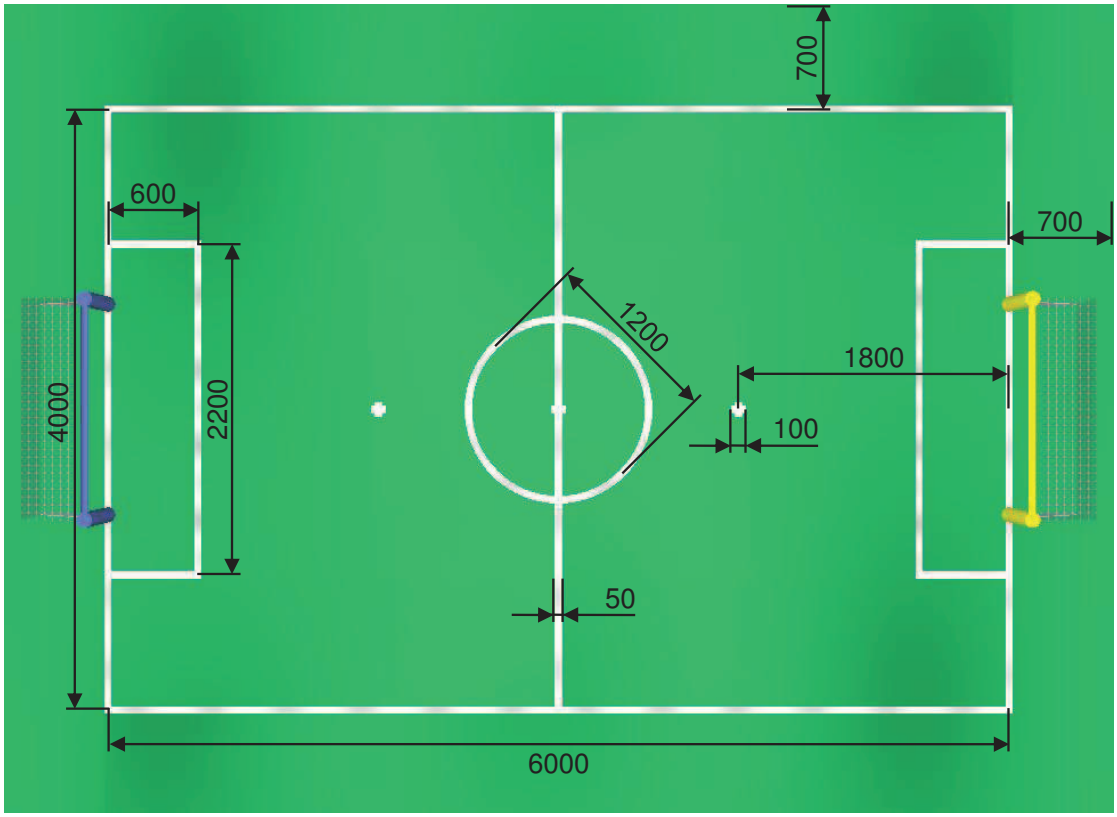


Abbildung 2.1. – Darstellung des Spielfeldes mit allen Abmessungen entnommen den Regeln von 2011. Alle Angaben in Millimeter.

ist. Dies führt mitunter zu starken Farbschwankungen. So kann die tatsächliche Farbe von leicht bläulich bis hin zu gelblich reichen. Beispielsweise wurde auf dem RoboCup 2011 in Istanbul auf einem gelbstichigen Teppich gespielt, was eine Anpassung der Spielfeldschwellenwerte innerhalb der Software nach sich zog. Je nachdem wie die jeweilige Software arbeitet, kann dies leichte oder schwerwiegende Folgen für die Bildverarbeitung eines Teams nach sich ziehen.[18]

Alle auf dem Feld dargestellten Linien besitzen eine Breite von genau 50 Millimetern. Diese Standardisierung ist bei der Lokalisierung und Bildverarbeitung von Bedeutung, da dadurch der Abstand zu erkannten Linien und der Winkel, wie der Roboter zu diesen Streifen positioniert ist, ermittelt werden kann. Ebenfalls hilfreich ist hierbei die Vorgabe, dass laut Regelwerk alle Außengrenzen exakt 70 Zentimeter nach der Spielfeldgrenze liegen.

[18] Vgl. [Rob11] ab Seite 1.

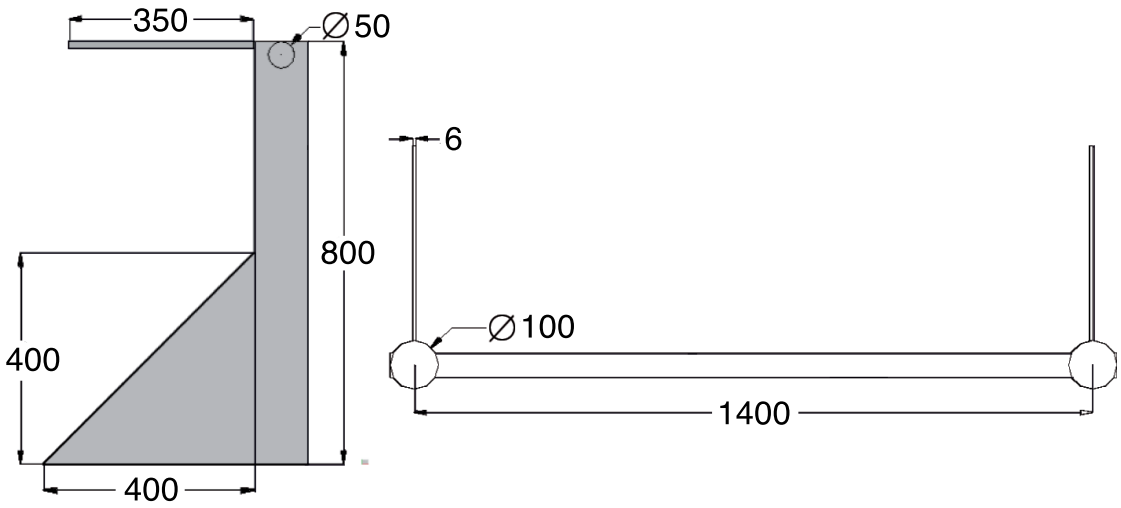


Abbildung 2.2. – Darstellung der Tore laut den Regeln von 2011. Alle Angaben in Millimeter.

Seit 2012 existieren neue Regeln, welche vorrangig die Tore betreffen. Zuvor war festgelegt, dass beide in verschiedenen Farben aufgestellt werden müssen, eines blau, das andere gelb. Mit der Regeländerung wurden beim RoboCup 2012 in Mexico beide Tore gelb gestrichen, was die einzelnen Teams vor neue Herausforderungen stellte. Bisher konnte man leicht die eigene Spielfeldhälfte anhand der Torfarbe ausmachen. Da dieser Vorteil wegfällt, gilt es neue Methoden der Selbstlokalisierung zu finden. Eine Darstellung der Abmessungen der Tore zeigt Abbildung 2.2.[19]

Das Spielfeld ist farbcodiert, um die visuelle Erkennung zu vereinfachen. Der Teppich ist grün, die Linien weiß, die Tore gelb und der Ball mit einem Durchmesser von 6,5 Zentimetern rot. Dieses Prinzip findet sich auch in der Klassifizierung der Roboter als Mitspieler und Gegner wieder. Die Naos eines Teams tragen blaue Hüftbänder, wohingegen das konkurrierende Team magentafarbene Bänder erhält.

2.3. Standardplattform „Nao"

Damit jedes Team, welches am RoboCup teilnimmt, die gleichen Voraussetzungen hat, wird die Roboterplattform *Nao* genutzt. Dabei handelt es sich um eine humanoide Standardplattform, an welcher keinerlei Modifikationen durch die Teilnehmer erlaubt sind. So ist sichergestellt, dass einzig die darauf laufende Software den Unterschied zwischen Gewinnen und Verlieren ausmacht. Die Abbildung 2.3 zeigt einen solchen Nao-Roboter.

[19] Vgl. [Rob11], Seite 2.

Abbildung 2.3. – Nao Standardplattform, hier in orange. Bei Verwendung während des RoboCups sind standardmäßig alle farbige Bauteile einheitlich grau (Quelle: Aldebaran Pressematerial, `www.aldebaran-robotics.com/`).

Der Nao wird im akademischen Umfeld eingesetzt, kann allerdings seit 2012 auch privat, mit höherem finanziellen Aufwand, erworben werden. Je nach Ausführung verfügt er über verschiedene Freiheitsgrade. In der „RoboCup Edition" sind es 21 und in der „Academics Edition" 25.[20]

Die Ausführung, welche beim RoboCup zum Einsatz kommt, unterscheidet sich in Details von der von Aldebaran angebotenen Standardvariante. So sind u.a. die Finger nicht mit Motoren versehen, da diese nur bedingt beim Fußballspielen von Nutzen wären, die Kosten erhöhen und, da die Roboter des Öfteren umfallen, auch schnell zu Bruch gehen würden. Die Spezifikationen der verbauten Hardware sind in Tabelle 2.1 zu finden.[21]

[20] Vgl. [Rei11], Seite 16.
[21] Vgl. [Rei11], Seite 16.

CPU	x86 AMD Geode mit 500 MHz
RAM	512 MB
USB Speicher	32 MB bis 4 GB (je nach Edition, eigene verwendbar)
Digitalkamera	2 mit je 640 x 480 Pixel Auflösung und 30fps
Kommunikation	WLAN
Betriebssystem	Linux
Gewicht	ca. 4.3kg
Höhe	ca. 58cm
Sensoren	Gyrometer, Beschleunigungssensoren, Fußsensoren, Ultraschall

Tabelle 2.1. – Hardwarespezifikationen des Nao.

Das im Kopf des Naos befindliche USB-Speichermedium dient sowohl als Bootmedium aber auch als Speicher im eigentlichen Sinne. Darauf enthalten ist ein Linux-Betriebssystem, welches einen speziellen gepatchten Kernel enthält, der für Echtzeitberechnungen ausgelegt ist.[22] Für die Kommunikation mit anderen Robotern und Geräten, z.B. dem Computer des Schiedsrichters, steht ausschließlich WLAN zur Verfügung. Um die Umgebung wahrnehmen zu können, besitzt der Nao eine Reihe von unterschiedlich nützlichen Sensoren, darunter Drucksensoren in den Füßen zur Schwerpunktbestimmung, Ultraschall zur Abstandsbestimmung[23] sowie Mikrofone und verschiedene Buttons.[24] Um die aktuelle Position des Oberkörpers zu messen, besitzt der Nao Beschleunigungs- und Winkelsensoren. Diese sind notwendig für alle Bewegungen, vom Stand zum Ausbalancieren, bis hin zum Laufen und Schießen.

Auch sind in der Mitte des Kopfteils des Naos zwei Kameras vertikal zueinander verbaut. Die obere Kamera dient der Lokalisierung auf dem Spielfeld und gibt einen guten Überblick über alle Geschehnisse vor dem Roboter. Die untere hingegen ist nützlich um Bälle direkt vor den eigenen Füßen zu erkennen sowie eigene und fremde Fußbewegungen auszumachen. Beide Kameras arbeiten mit einer Auflösung von 640×480 Pixel und einer Framerate von maximal 30 Bildern pro Sekunde.[25] Die obere Kamera bildet das Ausgangsmaterial für diese Arbeit, da sie das ganze Spielfeld überblicken kann.[26]

[22] Vgl. [Rei11], Seite 16.

[23] Dieser arbeitet sehr unzuverlässig und ist anfällig für Stürze, daher werden diese Sensoren nicht vom Nao-Team der HTWK Leipzig eingesetzt.

[24] Der zentral gelegene Hauptbutton in der Mitte des Brustkorbs dient sowohl als Start- und Resetbutton, als auch um via Audio Informationen über den aktuellen Zustand des Nao zu erhalten. Darunter der Batteriestand, die aktuelle IP-Adresse und der vergebene Name.

[25] Vgl. [Rei11], Seite 16.

[26] Zum Zeitpunkt dieser Arbeit brachte Aldebaran eine neue Generation des Nao heraus, welche mit hochaufgelösten Kameras arbeitet und einen schnelleren Prozessor besitzt. Allerdings wird diese Edition nicht Inhalt der Betrachtung sein, da zum einen kein vergleichbar vielschichtiges und auswertbares Bildmaterial in der Bearbeitungszeit dieser Arbeit erstellbar gewesen wäre und zum anderen der Pro-

2.4. Farbraum

Im Gegensatz zu Digitalkameras oder Monitoren arbeiten die beiden Kameras des Naos nicht mit dem weit verbreiteten *RGB*-Farbraum, sondern mit dem für digitale Fernseh- und Videoübertragungen üblichen **YCbCr**. Außerdem kommt es bei DVDs, Bild- und Videoaufzeichnungen, sowie bei Speicherformaten wie *JPEG* und *MPEG* zum Einsatz.

Beide Farbräume verwenden drei Kanäle. Im Falle von RGB bestehen diese aus den Grundfarben Rot, Grün und Blau. YCbCr dagegen wird in einen Helligkeits- und zwei Farbkanäle eingeteilt. Die Grundhelligkeit wird vom Y-Kanal (*Luma*) bestimmt, welcher an die Helligkeitsachse des CIE-Normvalenzsystems[27] angelehnt ist. Die beiden Farbkomponenten, *Chrominance* oder auch *Buntheit* genannt, sind Farbdifferenzsignale. Farbanteile im Spektrum von Blau bis Gelb werden von dem *Cb*-Kanal und im Bereich von Rot bis Grün von *Cr* erzeugt. Eine Umwandlung in den jeweils anderen Farbraum erfolgt durch Linearkombination der drei Kanäle, wie in Formel 2.1[28] ersichtlich wird.

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ -0{,}168736 & -0{,}331264 & 0{,}5 \\ 0{,}5 & -0{,}418688 & -0{,}081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{2.1}$$

In Anbetracht der Tatsache, dass das menschliche Auge empfindlicher auf Helligkeitsunterschiede reagiert als auf Farbunterschiede[29] bietet dieser Farbraum für die Digitalisierung von Bild- und Videomaterial einen entscheidenden Vorteil.[30] So können die Farbanteile mit einer geringeren Frequenz abgetastet werden, wodurch sich die Datenmenge um

zess der Anpassung der vorhandenen Software noch nicht vollständig abgeschlossen ist. Ein Mischen alter und neuer Kamerabilder ist ausgeschlossen, da die neueren über eine bessere Bildqualität und höheren Kontrast verfügen und somit das Ergebnis der Klassifizierung verfälschen würden.

[27] Das CIE-Normvalenzsystem, oder auch CIE-Normfarbsystem, ist ein Standard der Commission internationale de l'éclairage (CIE), der Internationalen Beleuchtungskommission. Es basiert auf der Definition von Farbe als Gesichtssinn und enthält alle durch das menschliche Auge wahrnehmbaren Farben. Subjektive Farbempfindungen wurden durch Versuchsreihen mit unterschiedlichen Testpersonen auf Farbvalenzen zurückgeführt. Die *Farbvalenz* ist dabei die Bewertung eines Farbreizes anhand der Empfindlichkeiten des menschlichen Auges. Vgl. hierzu [JB06], Seite 269.

[28] Vgl. [Sze08], Seite 79.

[29] Die *Stäbchen* im Auge sind für das Erkennen von Helligkeitsunterschieden verantwortlich, während die *Zäpfchen*, die Farbempfindung hervorrufen. Allerdings ist die Anzahl der ersteren um ein Vielfaches höher, wodurch Menschen bei Nacht die Umgebung durch Helligkeitsdifferenzen wahrnehmen, aber nicht mehr in verschiedene Farben unterscheiden können.

[30] Vgl. hierzu [JB06], Seite 5 und [But06] ab Seite 73.

mindestens ein Drittel[31] und bis zu 50%[32] reduziert, je nachdem welches Verfahren bei der Digitalisierung und späteren Speicherung eingesetzt wird.

Auch im Rahmen des RoboCup bietet dieser Farbraum zahlreiche Vorteile. Das gesamte Spielumfeld ist, wie in Kapitel 2.2 dargestellt, farbcodiert. Dabei sind die Farben im *YCbCr*-Farbraum so positioniert, dass eine Aufteilung in Bereiche sehr gut möglich ist. Allein durch Verwendung der *Cb* und *Cr* Anteile lassen sich, abgesehen von den Robotern selbst, fast alle wichtigen Objekte auf dem Spielfeld voneinander trennen. Einzig durch die Segmentierung des *Cb-Kanals* ist es möglich, den roten Ball vom Grün des Feldes abzutrennen und mit geringem Rechenaufwand über mehrere Frames hinweg zu verfolgen. Ähnlich verhielt es sich bis 2011 mit den blauen und gelben Toren, welche mit Hilfe des *Cb*-Anteils voneinander unterschieden werden konnten. Seit der Regeländerung von 2012 ist dieser Vorteil allerdings hinfällig.

[31] Dies beschreibt die Digitalisierung nach dem *4:2:2-Verfahren*. Die beiden Farbanteile *Cb* und *Cr* werden mit der halben Abtastfrequenz, verglichen mit der Helligkeitsinformation, abgetastet. Dieser Prozess wird auch auch *Color Subsampling* bezeichnet. Für jeweils vier Pixel mit Helligkeitswerten werden zwei Farbwerte gespeichert, wodurch sich die Datenmenge um 33% reduziert.

[32] Wie bereits in Fußnote 31 beschrieben werden auch hier die Farbanteile mit einer geringeren Frequenz abgetastet, welche in diesem Fall einem Viertel der Helligkeit entspricht. Dabei kommen auf vier Pixel an Helligkeitswerten genau ein Pixel mit Farbinformationen. Diese Methode ist das *4:2:0-Verfahren* und wird vorrangig beim DV-Format eingesetzt. Im Vergleich zum *4:4:4-Verfahren*, welches ohne Color Subsampling arbeitet, reduziert sich die Datenmenge um 50%.

3. Umsetzung der Robotererkennung

Um eine Klassifizierung von möglichen Robotern im Bild bewerkstelligen zu können, gilt es zuerst im Bild Regionen zu finden, deren Inhalt noch nicht zugeordnet werden konnte. Das bedeutet Bereiche im Bild, welche weder einem Tor, dem Ball, dem Spielfeld oder den Feldlinien angehören.

Von diesen Regionen im Bild werden anschließend diverse Merkmale, sogenannte „Features" errechnet, welche in Kombination in der Lage sind, eine Aussage darüber zu treffen, ob es sich bei dem Inhalt der Bereiche um einen Roboter handelt oder nicht. Um eine solche Entscheidung treffen zu können, wird ein maschinelles Lernverfahren angewandt, die Logistische Regression, welches nach einer Lernphase mit Testdaten in der Lage ist, neue und unbekannte Regionen zu klassifizieren.[33]

3.1. Datengrundlage

Damit Lernalgorithmen verwendet werden können, muss ein Grundbestand an Bildmaterial erstellt werden, von welchem diverse Informationen bekannt sind. Wichtig dabei ist, auch in Hinblick auf spätere Ausbaustufen, die Position der Roboter im Bild, deren Anzahl sowie Farbe und Position des Hüftbandes. Diese Informationen können anschließend automatisiert verarbeitet werden und eine Aussage treffen, ob eine erkannte Region einen Roboter beinhaltet oder nicht, sodass qualifizierte Trainings-, Validierungs- und Testsets mit dem vorhandenen Bildmaterial erstellt werden können.

3.1.1. Ermittlung der interessanten Regionen

Die bisher implementierte Bildverarbeitung[34] klassifiziert mit Hilfe der bereits zuvor ermittelten Spielfeldfarbe und Feldgrenzen Bereiche, die einen Roboter darstellen könnten. Unter Ausnutzung der bei der Felderkennung entstandenen Daten werden Regionen ermittelt, welche später die Robotererkennung nutzen kann.

[33] Siehe hierzu Kapitel 3.3.1.
[34] Im Folgenden auch *Vision* genannt.

Zur Bestimmung der noch nicht erkannten Regionen im Bild sind die folgenden Schritte notwendig:

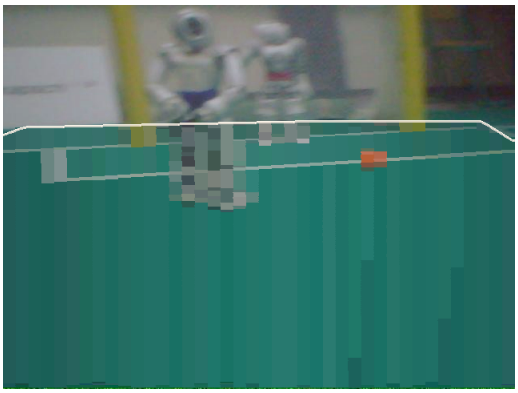
1. Vertikaler Scan des Bildes in einem festgelegten Abstand.
2. Speicherung der dabei ermittelten Kanten sowie weiterer grundlegender Informationen des direkten Umfeldes.
3. Suche nach unbekannten Regionen anhand dieser Scans.
4. Ermittlung und Bewertung von Rechtecken anhand der gefundenen Bereiche.
5. Anpassung der Rechtecke in Breite und Höhe.

Um die potenziellen Roboterregionen im Spielfeld zu erkennen, wird das Bild von oben nach unten gescannt, mit einem Abstand von 16 Spalten.[35] Dabei ist zu beachten, dass ausschließlich vertikale Scanlines verwendet werden, da horizontale Scanlines weniger Informationsgehalt im Bild besitzen, wenn man die distinkte Anzahl möglicher Objekte im Bild betrachtet.[36] Werden auf den Linien Kanten erkannt, bilden diese neue Elemente in einer Scanline-Klasse.[37]

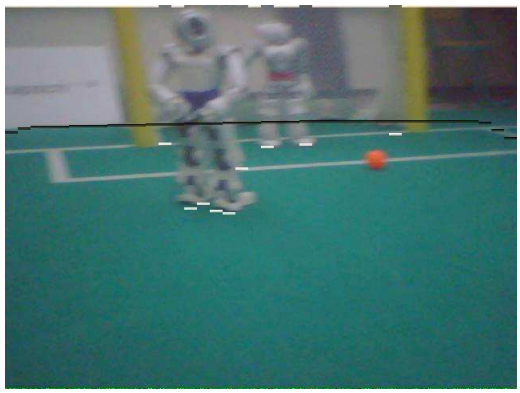
Diese Klasse speichert alle relevanten Informationen, welche pro Durchlauf gefunden wurden, darunter die Anzahl der Kanten einer kompletten Linie sowie deren x und y-Koordinaten. Zusätzlich, und für die Regionenerkennung wichtig, wird der Mittelwert der Farben der Region unterhalb diesen Koordinaten sowie zwei Wahrheitswerte, ob es sich um eine Spielfeld- oder Linienfarbe handelt, gespeichert. Diese Wahrheitswerte sind für diese Arbeit besonders relevant, da diese ausschlaggebend dafür sind, ob eine Region von bekannter oder unbekannter Art ist. Nur Bereiche, welche als unbekannt klassifiziert wurden, werden im Folgenden weiter betrachtet.

Dabei wird nach allen gespeicherten, unbekannten Regionen im Bild gesucht, das heißt alle Bereiche, die weder die Spielfeldfarbe grün noch das als Linienfarbe erkannte weiß beinhalten. Nur wenn mindestens drei dieser Felder übereinander auftreten, wird der Bereich als potenzieller Roboter in Betracht gezogen. Eine Visualisierung dessen ist in Abbildung 3.1 (**a**) zu sehen. Man beachte, wie sich Roboterregionenmittelwerte farbig von der Umgebung abheben.

[35] Dieser Abstand der zu scannenden Pixel kann beliebig gewählt werden. Die Zahl 16 hat sich in der Vergangenheit als das ideale Maß zwischen Genauigkeit der Scans und Geschwindigkeit des Algorithmus erwiesen, siehe dazu [Rei11], Seite 42.
[36] Vgl. [Rei11], Seite 24, 25 sowie ab Kapitel 3.5.3.
[37] Diese Klasse speichert alle Informationen, die auf einer Scanline erfasst wurden.



(a) Ausgabe der Farbmittelwerte des Regionen-Scans. Die weiße Linie stellt den Feldrand dar. Alle Bereiche darunter entsprechen den mittleren Farbwerten der entsprechenden Region.



(b) Ermittelte potenziell interessante Kanten. Die schwarze Linie stellt den Feldrand dar, die weißen Linien verdeutlichen die unteren Grenzen potenziell interessanter, unbekannter Regionen.

Abbildung 3.1. – Diese beiden Bilder zeigen, wie aus den unbekannten Farbmittelwerten die unteren Grenzen der gesuchten Regionen ermittelt werden. Man beachte dabei die weiße Linie am oberen Rand von Bild **(b)**, welche an Stellen unterbrochen ist, wo unterhalb der Feldgrenze Mittelwerte in **(a)** auftauchen, die noch nicht klassifiziert sind.

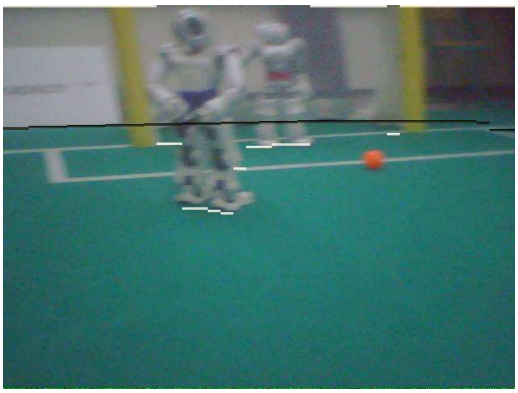
Von der Feldgrenze abwärts wird die unterste Y-Koordinate pro Scanline gespeichert, falls eine minimale Anzahl unbekannter Bereiche erreicht ist, wie in Abbildung 3.1 (**b**) visualisiert. Anschließend werden diese Koordinaten zu Gruppen zusammengefasst und Rechtecke daraus gebildet (siehe Abb. 3.2).

Das Bilden der Rechtecke folgt ähnlichen Prinzipien wie die Erkennung interessanter Bereiche in den unbekannten Regionen. Nur wenn der Bereich von Feldgrenze bis zur gespeicherten Y-Position größer als ein festgelegtes Mindestmaß ist, wird das daraus resultierende Rechteck in eine Bewertungsliste aufgenommen.

Um dies zu bewerkstelligen, existiert ein Bewertungskriterium, welches von allen sich überschneidenden Rechtecken die Fläche bildet und versucht das Maximum dieser zu finden. Dabei wird die Höhe stärker gewichtet als die Breite, da ein Roboter in der Regel deutlich weniger breit als hoch ist.[38] Das Ergebnis dieser Suche ist das Rechteck mit der größten Fläche.

Da ein Scan nach Kanten lediglich im Abstand von 16 Pixeln erfolgt, werden Roboterregionen nur grob aufgelöst erkannt. Eine genauere Bestimmung der möglichen Robotergrenzen ist erstrebenswert. Zudem kann sich das linke oder rechte Ende eines Roboters genau inmitten zweier Scans befinden.

[38] Etwa im Verhältnis Höhe zu Breite von 2,4:1



(a) Gruppierte Koordinaten potenzieller Robotergrenzen.

(b) Farbige Kennzeichnung der interessanten Regionen.

Abbildung 3.2. – Abbildung (**a**) zeigt die gruppierten Untergrenzen interessanter Regionen auf Basis der ermittelten Y-Koordinaten. In (**b**) sind die daraus resultierenden und im Weiteren zu betrachtenden Bereiche farbig markiert.

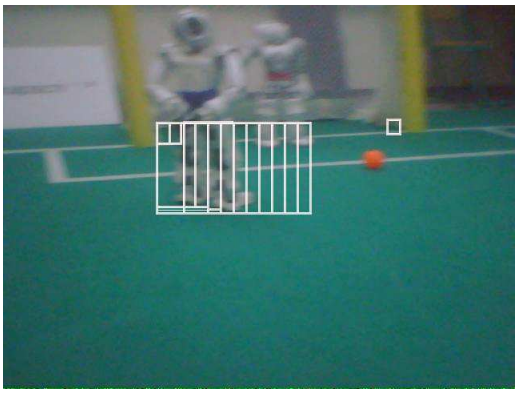
Zur genaueren Bestimmung der Grenzen einer unbekannten Region werden gefundene Rechtecke erneut gefiltert. Dabei erfolgt zuerst eine Breitenerkennung. Scanlines[39] werden aus dem Inneren des zu bestimmenden Bereiches nach links und rechts gesendet. Um die Dauer dieser Suche nach den Bereichsgrenzen nicht unnötig zu erhöhen, wird nicht jeder Pixel betrachtet, sondern alle in einem festgelegten Abstand.[40] Jede Linie wird so lang verfolgt, bis entweder die Spielfeldfarbe oder die Ballfarbe erkennbar ist. Anschließend berechnet der Algorithmus den Median[41] aus den gespeicherten x-Koordinaten, welcher die neue Kantenposition des Rechteckes wird.

Da über die zuvor bestimmte Feldgrenze hinaus kein Scan erfolgt, aber Roboter im Bild in der Regel oberhalb dieser Grenze enden, wird das Rechteck in der Höhe nochmals angepasst. Dabei wird eine Methode angewandt, welche auf der begründeten Annahme des Verhältnisses des Roboters von Höhe zu Breite beruht und durch zahlreiche visuelle Tests empirisch bestätigt wurde. Die zuvor ermittelte Breite der Region wird durch Addition der 1,4-fachen Breite vergrößert, sofern die Bildgrenzen dies zulassen. Das Resultat ist in Abbildung 3.3 visualisiert. Wie sich erkennen lässt, wurde die Höhe in einem ausreichenden Maß genau getroffen. Eine höhere Genauigkeit würde weitere Berechnungszeit nach sich ziehen, welche nicht notwendig und somit auch nicht erwünscht ist.

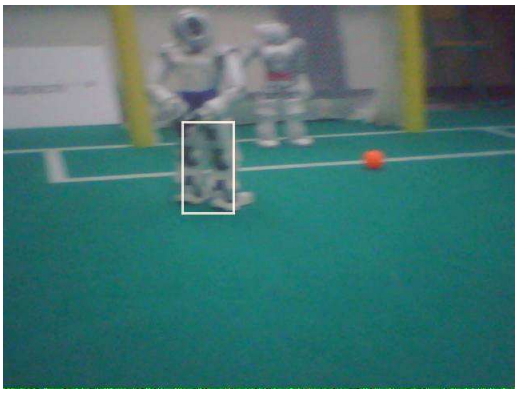
[39] Sieben Scanlines haben sich in manuellen Tests als ein guter Kompromiss zwischen Geschwindigkeit und Genauigkeit erwiesen.

[40] Der Abstand von 2 Pixeln erwies sich als ausreichend genau.

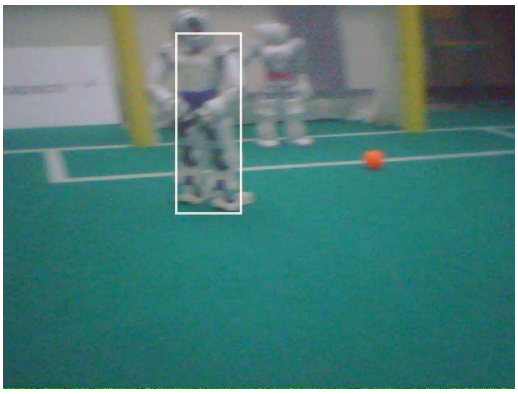
[41] Hierbei wird bewusst der Median und nicht ein Mittelwert verwendet, da sehr häufig einzelne Ausreißer vorkommen, welche die Regionen unnötig breit machen würden. Insbesondere ist das der Fall, wenn horizontale Feldlinien an die Region grenzen.



(a) Darstellung aller Rechtecke, die sich aus den ermittelten Koordinaten ergeben.



(b) Dasjenige Rechteck mit dem höchsten Wert des Bewertungskriteriums.



(c) Anpassung des ermittelten Rechtecks in der Höhe und Breite.

Abbildung 3.3. – Die einzelnen Bilder zeigen die Phasen der Rechteckermittlung, von der Liste aller Kandidaten (**a**), über das finale Rechteck (**b**), bis hin zur Anpassung der Größe (**c**).

Die genaue Berechnung ist im nachfolgenden Quelltext dargestellt. Dabei wird das Maximum genommen, welches 0 oder größer ist. Das verhindert, dass Regionen über Bildgrenzen hinausgehen und Fehler verursachen.

Quelltext 3.1 – Bestimmung der Roboterhöhe anhand der Breite.

```
1    region.yTop=(int)(Math.max(0, region.yTop-1.4*(robotRight-robotLeft)));
```

In seltenen Fällen kommt es vor, dass einzelne Regionen sich überlappen. Diese Bereiche werden gefiltert und zu einem Rechteck verbunden. Anschließend werden alle Regionen übernommen und den Klassifizierungsalgorithmen übergeben, sofern sie eine minimale Breite für Roboter besitzen und sich nicht an der Position des Tores befinden.

3.1.2. Erstellung der Testdaten

Im Rahmen dieser Arbeit kommen zwei verschiedene Testdatensätze zum Einsatz. Der erste und rudimentärere besteht aus den mit in Kapitel 3.1.1 beschriebenem Algorithmus erkannten Regionen jedes Bildes, welche in Einzelbilder ausgeleitet und gespeichert wurden, um anschließend durch Kopieren in verschiedene Unterordner manuell klassifiziert zu werden in "Roboter" und "Nicht-Roboter". Dieses erste Testset ermöglicht eine schnelle und effiziente Testdatenerstellung, um bereits in einem frühen Stadium dieser Arbeit Aussagen über den Nutzen verschiedener ermittelter Features treffen zu können.

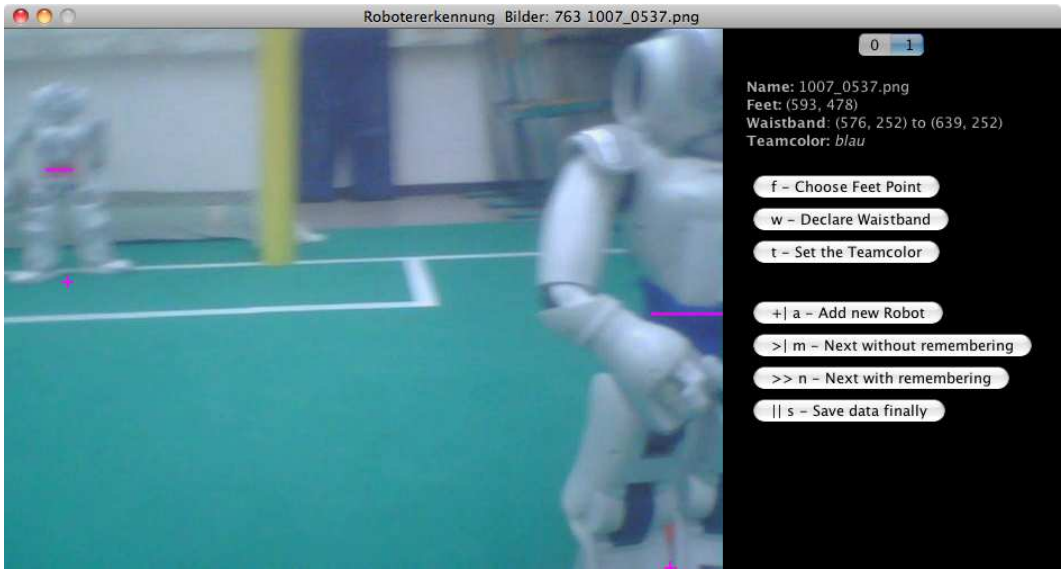
Die zweite, weitaus detailliertere Methode verwendet eine eigens für diesen Zweck erstellte Softwarelösung, welche es schnell und effizient ermöglicht, im Bild vorhandene Roboter zu taggen[42], sprich mit Informationen anzureichern. Dazu werden manuell pro Roboter der Fußpunkt, d.h. ein Punkt mittig von den Robotergrenzen, sowie die Position des Hüftbandes gespeichert. Die Farbe dieses Bandes ermittelt ein speziell hierfür entwickelter Algorithmus, kann andererseits aber auch manuell gesetzt werden, sollte das Bildmaterial nicht ausreichend qualitativ hochwertig sein.

Diese oben genannte Software wurde in Java geschrieben und stellt verschiedene Möglichkeiten bereit, das Tagging der Informationen im Bild zu bewerkstelligen. Je nach Geschmack und vorhandener Hardware kann zwischen einem auf Buttons, auf Tastatur und rein mausbasiertem Tagging gewählt werden, wobei die letztere Lösung eine mittlere Maustaste voraussetzt[43], aber dafür die mit Abstand schnellste Variante darstellt. Die Software ist in Abbildung 3.4 dargestellt. Sie zeigt sowohl die beiden wichtigsten Varianten des Programms als auch wie ein einmal getaggter Roboter im Bild aussieht.

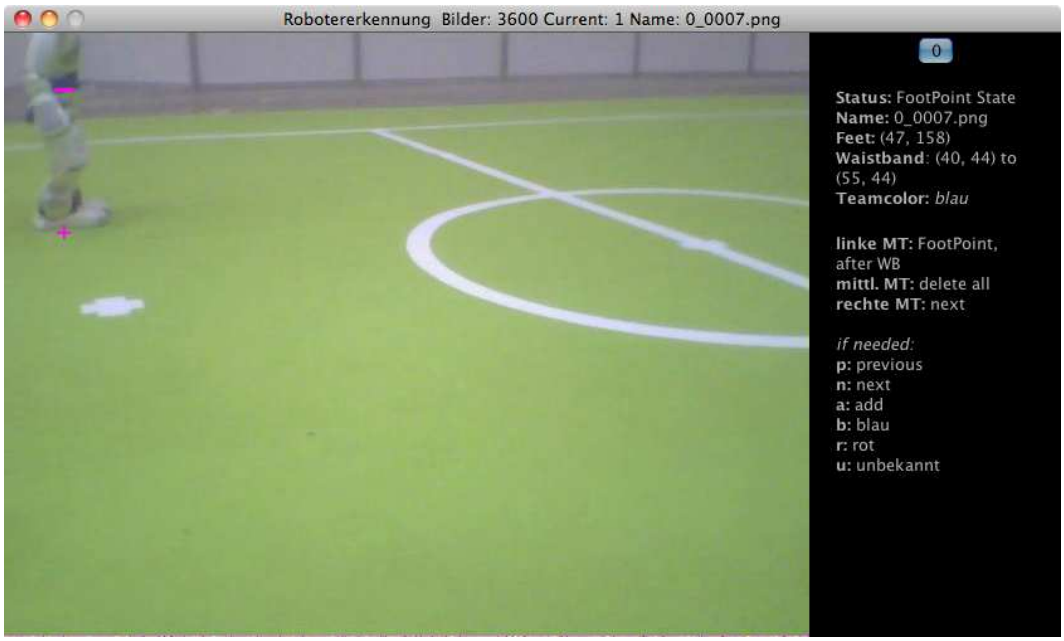
Die Speicherung der ermittelten Informationen erfolgt in XML. Da dieses Format durch den Menschen gelesen werden kann, ist sichergestellt, dass alle Personen mit Interesse an diesen Daten diese, ohne die Verwendung einer speziellen Software, öffnen, lesen und bearbeiten können. Zudem kann auf diese Art und Weise eine Schnittstelle mit anderen Programmen ermöglicht werden, da XML von nahezu jeder Programmiersprache verarbeitet werden kann. Der Quelltext A.1 im Anhang auf Seite 90 zeigt beispielhaft, wie eine solche XML aufgebaut sein kann.

[42] Tagging bezeichnet einen Vorgang, bei dem, in diesem Fall, ein Bild mit festgelegten Informationen belegt wird, beispielsweise die Anzahl an Robotern im Bild oder die Position eines Fußpunktes.

[43] Um eine Bedienung ausschließlich auf Basis der Maus zu realisieren, ist es unabdingbar, die mittlere Maustaste mit einzubeziehen, da ansonsten nicht alle Funktionalitäten bereitgestellt werden können.



(a) Die tastaturgesteuerte Variante zeigt zwei bereits vollständig getaggte Roboter.



(b) Hier sieht man eine auf Maussteuerung konzipierte Ansicht. Daher wurde hier auf zusätzliche Buttons verzichtet. Die rechte Leiste zeigt alle Möglichkeiten, welche dem Nutzer zur Verfügung stehen. Diese wechseln abhängig vom Modus, in welchem sich die Software befindet, beispielsweise Tagging des Hüftbandes oder Fußpunktes.

Abbildung 3.4. – Die beiden Teilbilder zeigen jeweils die einzelnen Ansichten der möglichen Bedienkonzepte. In **(a)** kann man vorrangig mit Tastatur und via Buttons agieren, wohingegen **(b)** eine Variante zeigt, welche einzig für die Maus konzipiert ist. Je nach Geschmack lässt sich diese auch mit der Tastatur steuern.

	Breite	Höhe
Median	91	249
Mittelwert	107	260
Minimum	21	27
Maximum	638	478

Tabelle 3.1. – Auswertung der Größen von 3.474 Rechtecken.

3.1.3. Normalisierung der Testdaten

Die erkannten Rechtecke, welche potenzielle Roboter darstellen, können sehr unterschiedliche Größen aufweisen. Einen Eindruck wie stark diese Schwankung der Formate ist, bietet Tabelle 3.1. Darin werden mehr als 3000 in Bildern ermittelte, nicht klassifizierte Bereiche miteinander verglichen.

In ihr lässt sich sehr gut die Verteilung der Größen ablesen. Die Höhe schwankt zwischen 27 und 478 Pixeln, hat allerdings im Mittel 260 Pixel. Dies entspricht in etwa der Hälfte des Maximums, was auf eine Glockenkurve hindeuten könnte. Aufschluss über diese Vermutung bietet Abbildung 3.5.

Anders verhält es sich mit der Breite von unbekannten Regionen. Die Schwankung fällt stärker aus, als dies in der Höhe der Fall ist. Zudem befindet sich der Mittelwert näher am Minimum als an der mathematischen Hälfte des maximalen Wertes, im Gegensatz zur Höhe.

Da die ermittelten Werte von Tabelle 3.1 nur ein Indiz über die Größenverteilung geben, sollen die folgenden Graphen einen detaillierteren Überblick bieten.

Abbildung 3.5 zeigt die Häufigkeit von unterschiedlichen Höhen innerhalb der erkannten Regionen in Pixeln. Deutlich erkennbar ist eine sehr starke Schwankung in den Verteilungen. Hohe wie auch niedrige Vorkommen sind sehr nahe beieinander und geben den Eindruck eines breiten Feldes von Werten. Allerdings ist ebenfalls eine flache, sehr langgestreckte Glockenkurve zu erkennen.

Auffällig ist der große Ausschlag bei 480 Pixeln, welcher der gesamten Bildhöhe entspricht. Grund hierfür sind Bilder mit Robotern aus nächster Nähe, die den gesamten Bildbereich von oben bis unten einnehmen. Da derartige Bilder keine Seltenheit darstellen und auch sehr häufig bei den Testbildern auftreten, verwundert dieser Ausreißer nicht. Ohne diese Häufung bei 480 Pixeln ist die flache Glockenkurve der Verteilungen deutlich besser ersichtlich, allerdings wird an dieser Stelle auf eine doppelte Darstellung des gleichen Sachverhaltes verzichtet.

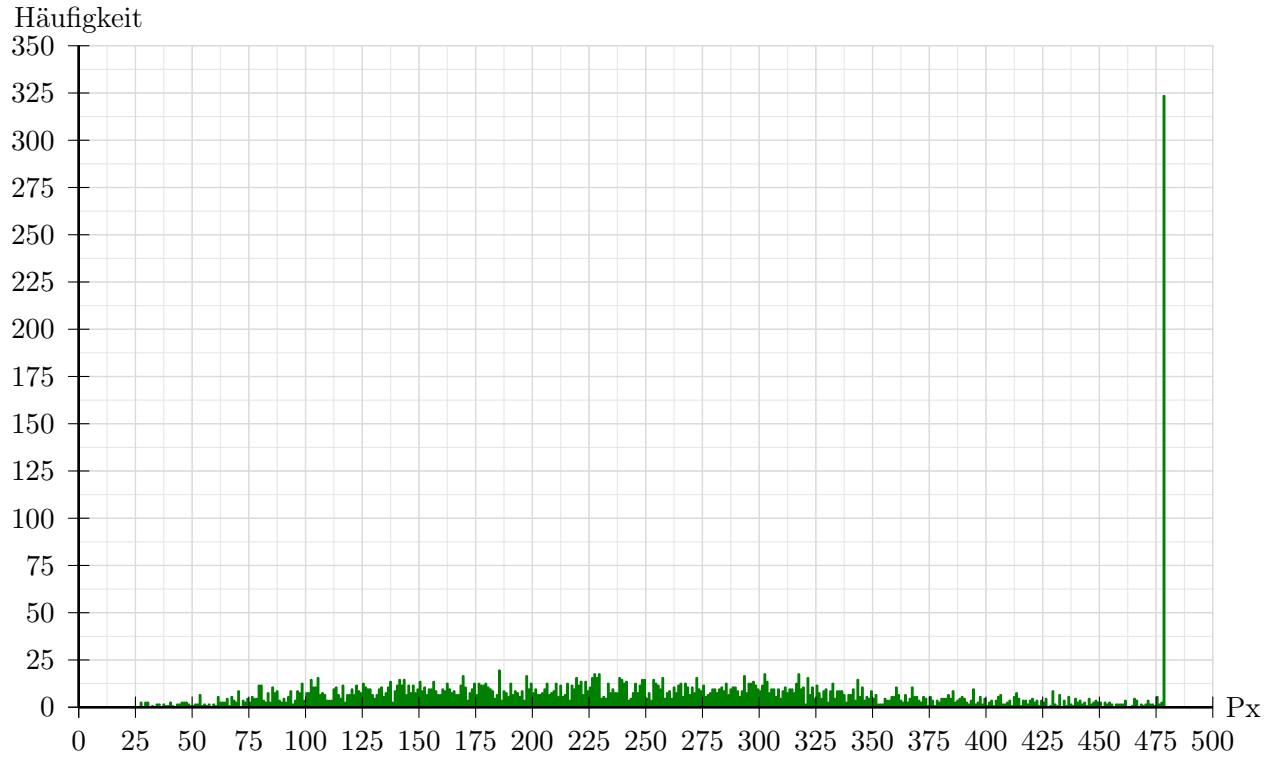


Abbildung 3.5. – Dieser Graph zeigt die Verteilung der Höhen erkannter Regionen. Die x-Achse zeigt die Größen in Pixel, wohingegen die y-Achse eine Aussage darüber trifft, wie oft diese Höhe in den Testdaten vorkam.

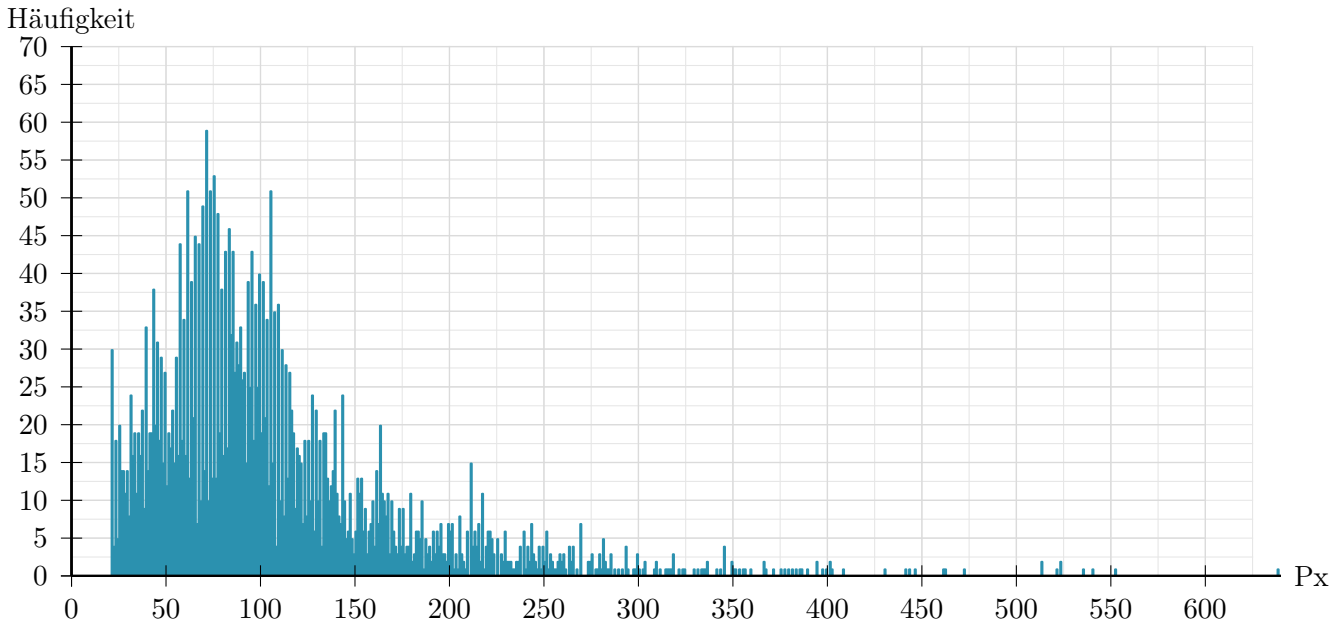


Abbildung 3.6. – Dieser Graph zeigt die Verteilung der Breiten erkannter Regionen. Die x-Achse zeigt die Größen in Pixel, wohingegen die y-Achse eine Aussage darüber trifft, wie oft die jeweilige Breite in den Testdaten vorkam.

Die Breite zeigt dagegen ein sehr deutliches Bild bei der Verteilung der Pixelwerte, wie in Abbildung 3.6 sichtbar. Das Vorkommen beschränkt sich bis auf wenige Ausnahmen fast ausschließlich auf Werte zwischen 30 und 230 Pixeln Breite. Verantwortlich hierfür ist die Methodik der Erkennung von unbekannten Regionen, welche lediglich den größten und damit wahrscheinlichsten Bereich eines Bildes verwendet. Allerdings können hier auch Werte bis 640 Pixel vorkommen, da in der Breite keinerlei Restriktionen gesetzt werden, wie dies beispielsweise bei der Höhe durch die Spielfeldgrenze gegeben ist. Einen direkten Vergleich der Verteilung von Höhen und Breiten der Regionen bietet die nachfolgende Abbildung 3.7.

Das Ziel der Ermittlung der Mittelwerte und Schwankungen in Höhe und Breite der Regionen ist die Bestimmung einer optimalen Größe zur Normierung. Alle Bilder sollten für die Berechnung der einzelnen Features das gleiche Format besitzen, unabhängig von ihrer ursprünglichen Größe. So lassen sich die einzelnen Regionen besser vergleichen und keine Abmessungen eines Bereiches dominieren das Ergebnis der Algorithmen.

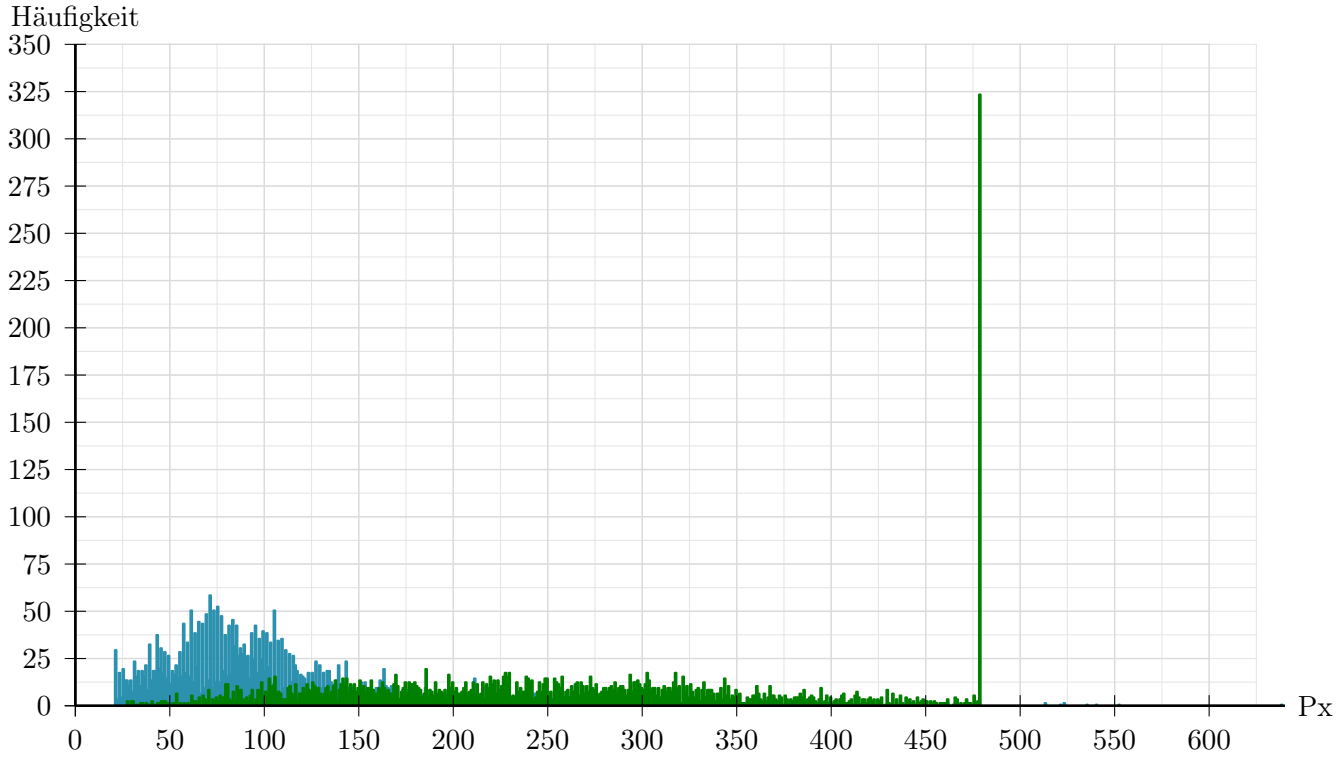


Abbildung 3.7. – Dieser Graph zeigt die Verteilung der Breiten und Höhen erkannter Regionen im direkten Vergleich. Die x-Achse zeigt die Größen in Pixel, wohingegen die y-Achse eine Aussage darüber trifft, wie oft die jeweilige Größe in den Testdaten vorkam.

	Wert in %	x	y	
Maximale Korrektheit	93,81%	11	20	6,16% falsch positive
Minimale falsch positive	4,087%	17	24	92,27% absolute Korrektheit

Tabelle 3.2. – Die besten Ergebnisse der statistischen Auswertung von verschiedenen Normgrößen.

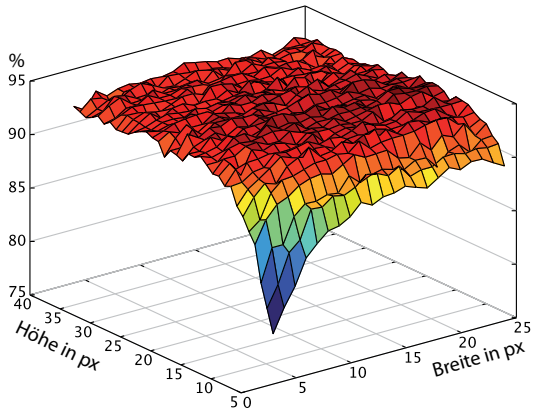
Hinzu kommt, dass je weniger Pixel verarbeitet werden müssen, desto weniger Berechnungen zur Ermittlung der einzelnen Features sind notwendig und die Geschwindigkeit der Erkennung steigt. Da die Ressourcen auf den Nao-Robotern begrenzt sind und nur wenige Millisekunden für die Klassifizierung zur Verfügung stehen, ist dies ein wichtiger Faktor, welcher von vornherein bedacht werden muss.

Um die optimale Normgröße zu bestimmen, wurde eine Testreihe über verschiedene Breiten und Höhen durchgeführt und jeweils die Erkennungsrate gespeichert. Die Abbildung 3.8 zeigt die Ergebnisse dieser Auswertung. Auffällig ist, dass jeder Wert, sobald eine minimale Größe der normierten Bilder erreicht ist, ein Plateau bildet, auf welchem die Erkennungsrate nur minimal schwankt. Lediglich die Falsch-Positiv-Rate, sprich Bilder, die keinen Roboter enthalten, aber in denen diese dennoch klassifiziert wurden, zeigt eine deutlich höhere, scheinbar zufällige Schwankung. Eigene Nachforschungen haben ergeben, dass je nachdem wie der randomisierte Algorithmus zur Unterteilung der Beispieldaten in Trainings-, Validierungs- und Testset diese Aufspaltung vornimmt, die Anzahl der falsch positiv klassifizierten Bilder steigt oder fällt, das heißt je nach Zusammenstellung der einzelnen Beispiele.[44]

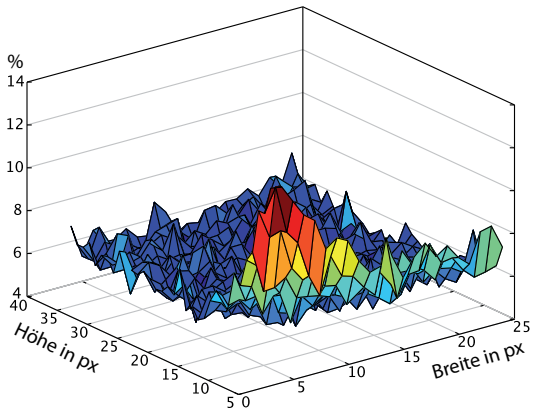
Diese Beobachtung lässt den Schluss zu, dass größere normierte Bilder auf die Erkennungsrate keinen weiteren positiven Effekt haben. Aus diesem Grund kann eine Normgröße gewählt werden, welche unterhalb des Minimums der potenziellen Breite und Höhe der erkannten Regionen liegt und somit die Geschwindigkeit der Verarbeitung dieser Bilder gesteigert werden.[45]

Für das Finden der optimalen Größe sind nur zwei der fünf Erkennungsraten relevant – die absolute Korrektheit aller Werte, sprich wie viele Bilder insgesamt korrekt klassifiziert worden sind, sowie die Falsch-Positiv-Rate. Aus diesem Grund wurde das Maximum des ersten Wertes sowie das Minimum des zweiten ermittelt, wie in Tabelle 3.2 ersichtlich ist. Sie zeigt zusätzlich die jeweils andere Rate, um beide miteinander vergleichen zu können. Eine Aussage zur mittleren quadratischen Schwankung der Erkennungsraten wird in Kapitel 4.1 in Verbindung mit den endgültigen Ergebnissen getroffen.

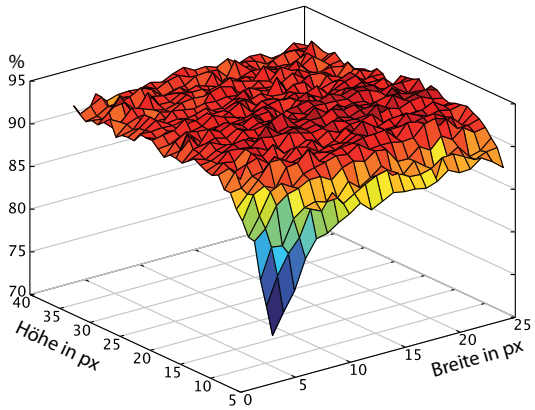
[44] Zur Erstellung der Testreihe und Graphen wurde der Median der Erkennungswerte von 10 zufällig erstellten Test- und Trainingssets je Normgröße gewählt.
[45] Siehe dazu Tabelle 3.1 auf Seite 28.



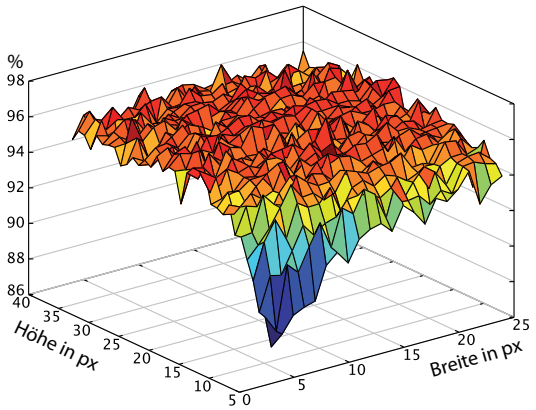
(a) Die absolute Korrektheit aller Werte bezogen auf die unterschiedlichen Normgrößen.



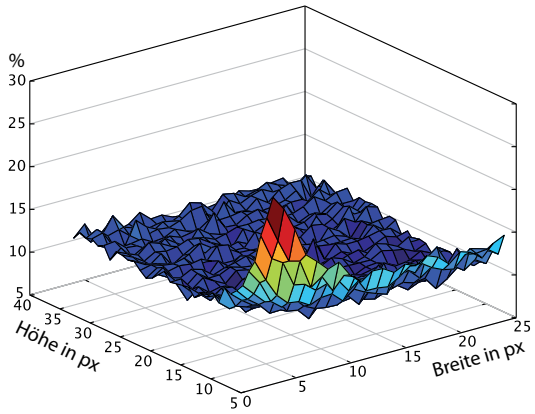
(b) Die Falsch-Positiv-Rate der verschiedenen Normgrößen – Bilder, die keinen Roboter zeigen, aber als solcher markiert wurden.



(c) Die korrekt erkannten Roboter.



(d) Korrekt erkannte Bilder ohne Roboter.



(e) Falsch erkannte Bilder, welche einen Roboter zeigen, aber als negativ bewertet worden sind.

Abbildung 3.8. – Dargestellt sind die Erkennungsraten abhängig von der gewählten Normgröße. Die Normgröße-Achse zeigt die Breite und die y-Achse die Höhe in Pixeln der genormten Bilder. z hingegen verdeutlicht die jeweilige Erkennungsrate in Prozent. In Reihenfolge: **(a)** zeigt die tatsächliche Korrektheit über alle Werte, **(b)** alle falsch positiven, **(c)** die wirkliche Positiv-Rate, **(d)** wirklich negative und **(e)** falsch negativ zugeordnete Bilder.

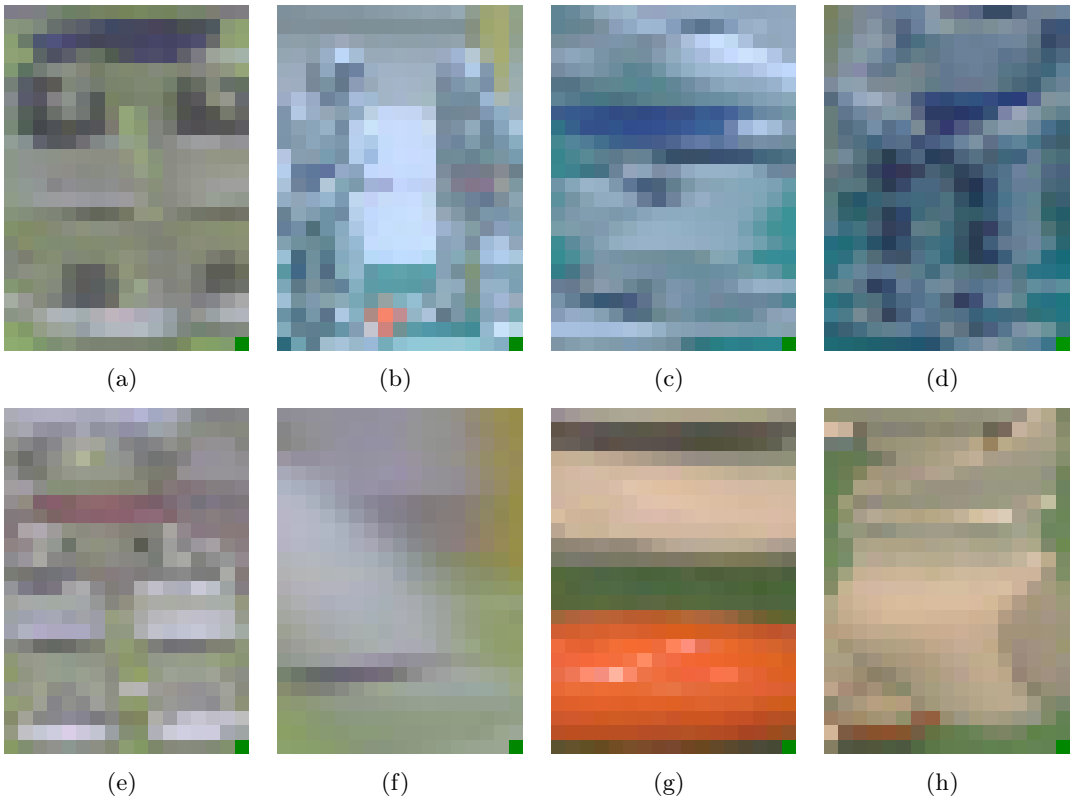


Abbildung 3.9. – Hier aufgelistet sind eine Reihe von Bildern, welche zuvor normiert wurden. Derartige Abbildungen sind die Grundlage, auf welcher diverse Merkmale bestimmt werden.

Auffällig ist, dass bei der besten Erkennungsrate, also bei einer Normgröße von 11×20 Pixeln, die Falsch-Positiv-Rate sehr hoch ist. Allerdings ist dieser zweite Wert für den Roboterfußball von höherer Bedeutung, da ein nicht erkannter Roboter im Vergleich zu einem falsch erkannten Roboter in einem einzelnen Frame während eines Spiels weniger starke Auswirkungen hat. Daher wurde zusätzlich das Minimum der falsch positiv klassifizierten Bilder ermittelt, welches sich bei 17×24 Pixeln befindet. Diese Normgröße wird zukünftig standardmäßig bei allen Klassifizierungen zum Einsatz kommen.[46] Abbildung 3.9 zeigt Beispiele, wie ein solches normiertes Bild aussehen kann. Diese sind repräsentativ für das Datenmaterial, für welches im folgenden Kapitel diverse Merkmalsgruppen ermittelt werden.

[46] Anmerkung: Durch Verwendung der Normierung von Bildgrößen konnte nicht nur die Verarbeitungsgeschwindigkeit gesteigert, sondern auch die Erkennungsrate von durchschnittlich 90% auf 92% gesteigert werden.

3.2. Merkmale

Um eine Region in einem Bild klassifizieren, beziehungsweise eine Aussage darüber treffen zu können, ob es sich um einen Roboter handelt oder nicht, werden für jeden dieser Bildbereiche Merkmale bestimmt. Anhand dieser Features muss ein Klassifikator[47] in der Lage sein, eine solche Aussage treffen zu können.

Der nachfolgende Abschnitt wird jedes ermittelte Merkmal sowie dessen Berechnung beschreiben. Um einer Dominanz durch einzelne Features vorzubeugen, werden alle einzeln normiert. Der dazu verwendete Algorithmus ist in Kapitel 3.2.3 beschrieben. Abschließend erfolgt in Kapitel 3.2.4 eine Darstellung, welche Merkmale die größte Aussagekraft besitzen und wie diese ermittelt wurden. Es sollen auf dem Roboter aus Performancegründen nur Features berechnet werden, welche für die Erkennung absolut notwendig sind, daher ist eine Begrenzung auf die aussagekräftigsten Merkmale unabwendbar.[48]

3.2.1. Ermittelte Merkmale

Die ab diesem Kapitel verwendeten Symbole sind nachfolgend gelistet. Jedes neue Symbol wird am Anfang jedes Unterkapitels in ähnlicher Form bekannt gemacht. Ein vollständiges Symbolverzeichnis ist im Anhang zu finden.

$\mathbf{K}$ Alle Kanäle eines Bildes

$\mathbf{s}(\mathbf{x}, \mathbf{y})$ ein Pixel in einem Bild mit den zweidimensionalen Koordinaten $\mathbf{x}$ und $\mathbf{y}$

$\mathbf{S} = \{\mathbf{s}(\mathbf{x}, \mathbf{y})\}$ ein Kanal in einem 3-kanaligen Bild mit $\mathbf{L}$ Zeilen und $\mathbf{R}$ Spalten[49]

$\mathbf{L}, \mathbf{R}$ Zeilen und Spalten in einem Kanal eines Bildes

$\mathbf{M} = \mathbf{L} * \mathbf{R}$ Anzahl der Bildpunkte von $\mathbf{S}$

Alle ermittelten Merkmale einer Region können grob in zwei Klassen eingeteilt werden. Die erste Kategorie bilden farbkanalbasierte Features. Das bedeutet, dass für jeden der drei Kanäle des *YCbCr*-Farbraumes die Berechnungen getrennt erfolgen und gespeichert werden. Die zweite Kategorie sind gradientenbasierte Merkmale, welche auch in zwei Unterklassen, abhängig von ihren jeweiligen partiellen Ableitungen, eingeteilt werden können. Je nachdem, von welchen Pixeln der Gradient bestimmt wird, gehört das Merkmal in die Gruppe *X-Richtung* oder *Y-Richtung*. Dabei beschränkt sich die Berechnung

[47] Vgl. dazu Kapitel 3.3.
[48] Wie die Merkmale mit der höchsten Aussagekraft ermittelt wurden, ist in Kapitel 3.2.4 erklärt.
[49] Entspricht einer Menge von Pixeln. Eine genaue Auflistung aller Symbole ist im Anhang zu finden.

ausschließlich auf Basis des Y-Kanals.[50] Da die Resultate bereits Erkennungsraten von 90% überschreiten, wird auf die Berechnung der Gradienten der anderen beiden Kanäle verzichtet, um Rechenkapazität zu sparen. Die Wahl, welcher Kanal die Grundlage zur Berechnung sein soll, fiel bewusst auf den Helligkeitskanal, da dieser den verlässlichsten Informationsgehalt besitzt.[51]

Vorraussetzung für ein aussagekräftiges Merkmal ist die einfache Trennbarkeit durch einen Schwellenwert, der besagt, ab welchem Wertebereich eine Region aller Wahrscheinlichkeit nach einen Roboter zeigt.[52]

3.2.1.1. Farbkanalbasierte Merkmale

In dieser Merkmalsgruppe werden Symbole verwendet, die zuvor noch nicht spezifiziert wurden. Einen Überblick verschafft die folgende Auflistung:

$\mathbf{p_S(g)}$ Histogramm der relativen Häufigkeiten eines Grauwertes **g** des Kanals **S**

$\mathbf{a(g)}$ Häufigkeit des Auftretens des Tonwertes **g**

G Grauwertmenge von 0 bis 255

g Grauwert aus Grauwertmenge **G**

$\mathbf{m_S}$ Mittelwert von Kanal **S** (mittlerer Grauwert)

$\mathbf{q_S}$ Mittlere quadratische Abweichung von Kanal **S**

$\mathbf{H_S}$ Entropie eines Kanals

$\alpha_\mathbf{S}$ Der Anisotropiekoeffizient eines Kanals

Ein Großteil der gradienten- und farbbasierten Merkmale beruht auf den Histogrammen im Bild vorhandener Tonwerte. Im letzteren Fall wird für jeden Kanal ein eigenes Histogramm erstellt, das einen Überblick darüber gewährt, wie oft ein Tonwert im jeweiligen Kanal auftritt. Daher wird dieses Fundament auch *Histogramm der relativen Häufigkeiten* genannt.[53]

[50] Siehe dazu Kapitel 3.2.4. Hier wurde getestet, welche Features ausschlaggebend sind. Dabei wurden alle Merkmale, welche auf den Kanälen *Cb* und *Cr* basieren, entfernt.

[51] Die Farbkanäle sind, wie bereits zuvor erwähnt, sehr stark lichtabhängig. Einzelne Farbinformationen bieten daher keine generell verlässliche Informationsquelle, im Gegensatz zum Helligkeitskanal. Dieser zeigt die Helligkeitsunterschiede einzelner Bildregionen und ist von Farbschwankungen nicht betroffen. Je nach Art der Merkmale ist die Grundhelligkeit eines Bildes nicht von Belang.

[52] Eine Übersicht aller Merkmalskategorien ist im Anhang unter A.1 zu finden.

[53] Vgl. [AN04], Seite 388.

Allerdings werden nicht alle 256 Tonwerte eines jeden 8-Bit basierten Kanals gespeichert, sondern es erfolgt eine Reduktion der Anzahl an Farbwerten auf 64. Diese Farbtiefe ist ausreichend, um eine Aussage zur generellen Verteilung der Grauwerte zu geben. Der Informationsgehalt je Histogrammbalken wird erhöht und zeitgleich der notwendige Rechenaufwand verringert. In der Umsetzung wird diese Reduktion durch die Bitoperation *shift* >> erreicht, beispielsweise wird aus 256 >> 2 die Zahl 64. Dies entspricht einer Verkürzung um zwei binäre Stellen des Zahlenformats.

Der Algorithmus 3.1 zeigt auf, wie die Berechnung des Histogramms für alle drei Kanäle des Farbraums erfolgt.

Algorithmus 3.1 Erstellung der Histogramme aller drei Farbkanäle

Input: K

1. **for all** $S \in K$ **do**
2. **for all** $g \in G$ **do**
3. $p_S(g) = \frac{a(g)}{M}$
4. **end for**
5. **end for**
6. **return** $p_S(g)$

Anschließend wird das Histogramm über die Anzahl von Bildpunkten M je Kanal normiert, so dass gilt

$$\sum_{g=0}^{255} p_S(g) = 1 \tag{3.1}$$

und $0 \leq p_S(g) \leq 1$. Wie bereits einleitend erwähnt, kommen bei der Umsetzung auf dem Roboter nur Merkmale zum Einsatz, welche auf dem Y-Kanal basieren. Daher wird im späteren Verlauf dieser Arbeit auf die Berechnung der Histogramme für beide Farbkanäle verzichtet.

Die Mittelwerte der einzelnen Farbkanäle wurden erstellt, um zu testen, ob die mittlere Farbe der drei Kanäle eine Aussage zum Inhalt der Bildregion treffen kann. Insbesondere durch die farbigen Hüftbänder lag diese Vermutung nahe. Allerdings ist dieses Merkmal zu stark von den umgebenden Lichtverhältnissen abhängig und wurde daher hier nicht weiter verfolgt.[54] Die Berechnung gestaltet sich folgendermaßen:

[54] Beispielsweise wären Features denkbar, die Aussagen zu den relativen Abständen der Mittelwerte treffen. Allerdings sind die absoluten Werte zu instabil in Bezug auf das Umgebungslicht, als dass dieses Merkmal eine zutreffende, kalibrierungsfreie Klassifizierung ermöglichen könnte.

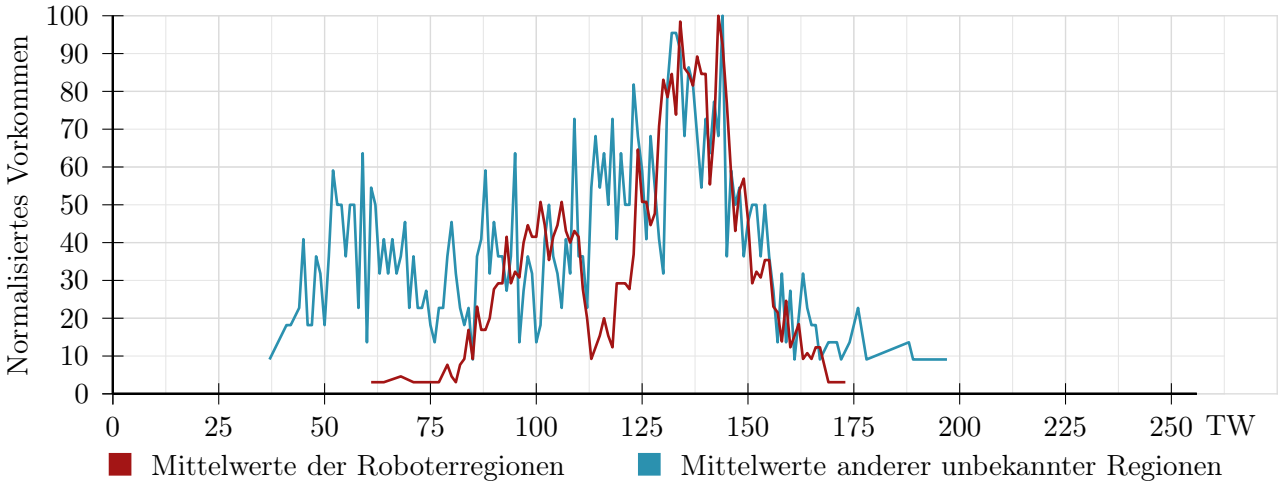


Abbildung 3.10. – Mittelwerte des Y-Kanals von Roboterregionen und Regionen mit anderem unbekannten Inhalten. Der Graph zeigt die einzelnen Tonwerte (TW) in Relation zur Häufigkeit ihres Vorkommens. Weitere Graphen siehe Anhang unter A.3.2.

Algorithmus 3.2 Mittelwerte jedes Kanals

Input: K

1. **for all** $S \in K$ **do**
2. $m_S = \frac{1}{M} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} s(x,y)$
3. **end for**
4. **return** m_S

Wie in Abbildung 3.10 sichtbar[55], unterscheidet sich der Wertebereich von Regionen, die Roboter zeigen, nicht grundsätzlich von jenen, welche anderen, unbekannten Inhalt besitzen. Daher ist dieser Wert nicht aussagekräftig.

Die mittlere quadratische Abweichung trifft eine Aussage über den Kontrast eines Kanals des jeweiligen Bildes. So können Bildregionen ausgeschlossen werden, die größtenteils kontrastarme, einfarbige Flächen bilden. In diese Kategorie fallen beispielsweise Schiedsrichter, deren Hosen eine unbekannte Region im Bild darstellen können. In der Umsetzung wird die mittlere quadratische Abweichung des Helligkeitskanals verwendet. Auch hier boten die Farbkanäle keine stabile Aussagekraft.

[55] Im Folgenden werden alle Graphen mit Merkmalen der gleichen normalisierten y-Skala entsprechen. Das höchste Vorkommen je Klasse entspricht der Zahl 100, alle anderen Werte dem prozentualen Anteil von diesem maximalen Wert. Der Grund für diese Anpassung liegt in der besseren optischen Vergleichbarkeit von Features in Bezug auf Bildregionen mit Robotern und solchen ohne, da die Testbilddatenbank deutlich mehr Regionen mit Robotern enthält als Bereiche mit anderem Inhalt.

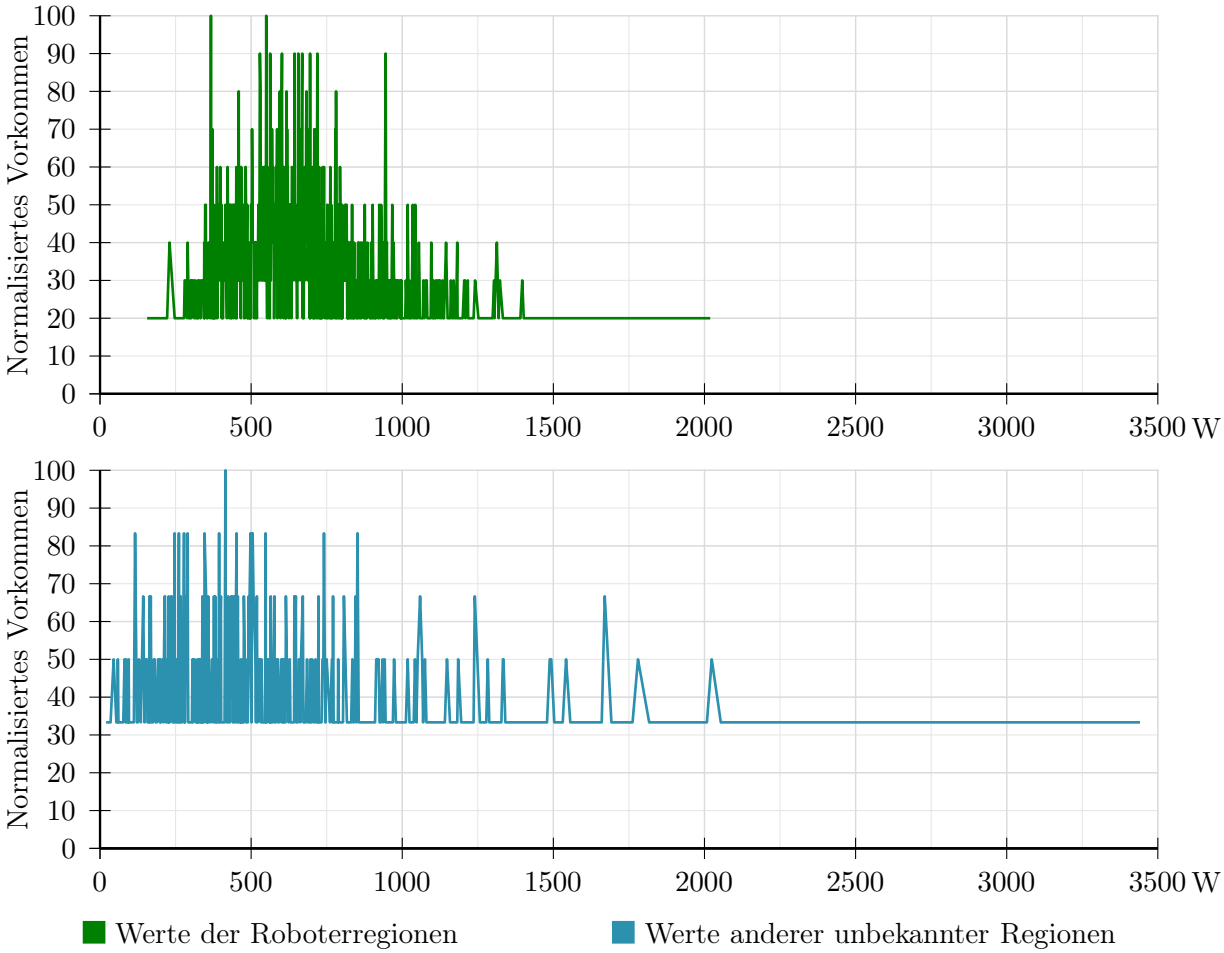


Abbildung 3.11. – Mittlere quadratische Abweichung des Y-Kanals von Roboterregionen und Regionen mit anderen unbekannten Inhalten. Der Graph zeigt die einzelnen Werte (W) in Relation zur Häufigkeit ihres Vorkommens.

Algorithmus 3.3 Mittlere quadratische Abweichung jedes Kanals

Input: K

1. **for all** $S \in K$ **do**
2. $q_S = \frac{1}{M} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} (s(x,y) - m_S)^2$
3. **end for**
4. **return** q_S

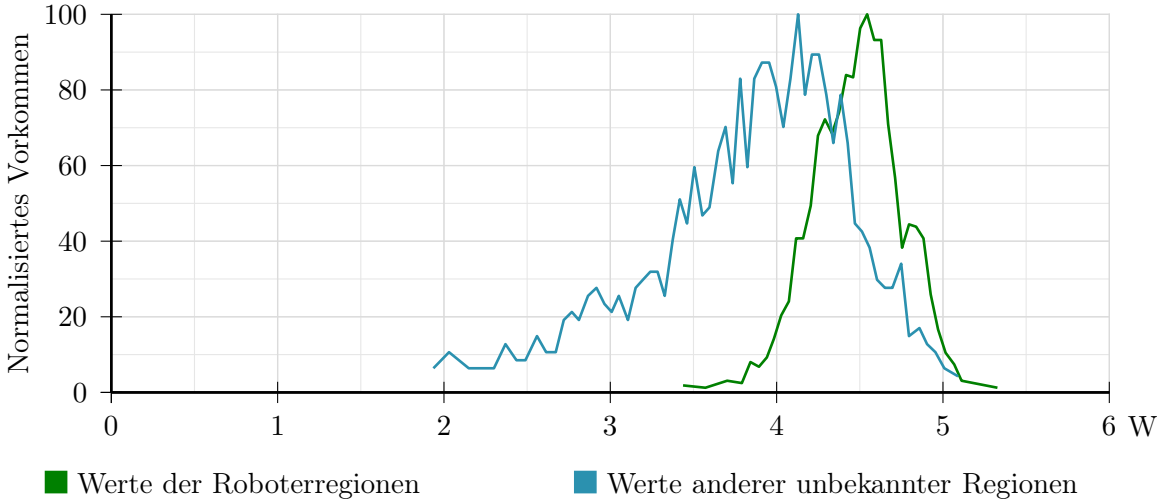


Abbildung 3.12. – Entropie des Y-Kanals von Roboterregionen und Regionen mit anderen unbekannten Inhalten. Der Graph zeigt die einzelnen Werte (W) in Relation zur Häufigkeit ihres Vorkommens.

Entropie der Farbkanäle Die Entropie eines Bildes ist ein Maß, das vorrangig bei der Datenkompression eingesetzt wird. Ziel ist es, bei der Reduktion des benötigten Speicherbedarfs möglichst geringe Informationsverluste zu haben. Im weiteren Sinne kann man sie als ein Merkmal für den mittleren Informationsgehalt betrachten. Sie wird in der Regel verwendet, um die notwendige Anzahl Bits je Pixel zu ermitteln. In der Theorie kann ein Bild nicht ohne Informationsverlust mit weniger Bit gespeichert werden als es der Wert der Entropie H aussagt. Beispielsweise ist bei einer einfarbigen Fläche $H = 0$, wohingegen ein Bildkanal, in welchem alle Tonwerte gleich verteilt sind, das heißt jeder Grauwert eine Wahrscheinlichkeit von $p_s(g) = \frac{1}{256}$ besitzt, eine Entropie von $H = 8$ hat.[56]

Wie in Abbildung 3.12 sichtbar, ist dieses Merkmal sehr aussagekräftig und kommt daher verstärkt zum Einsatz, insbesondere bei den gradientenbasierten Features.

Algorithmus 3.4 Entropie jedes Kanals

Input: K

1. **for all** $S \in K$ **do**
2. $H_S = -\sum_{g=0}^{255}(p_S(g) \cdot log_2(p_S(g)))$
3. **end for**
4. **return** H_S

[56] Vgl. [AN04], Seite 390 sowie zum nachfolgenden Anisotropiekoeffizienten [AN04], Seite 392.

Ein von der Entropie abgeleitetes Merkmal ist der **Anisotropiekoeffizient**. Er trifft eine Aussage über die Symmetrie eines Histogramms.

Algorithmus 3.5 Der Anisotropiekoeffizient jedes Kanals

Input: K

1. **for all** $S \in K$ **do**
2. $\alpha_S = \frac{-\sum_{g=0}^{k}(p_S(g) \cdot log_2 p_S(g))}{H_S}$
3. **end for**
4. **return** α_S

Die Grenze k ist derjenige Grauwert, für den gilt:

$$\sum_{g=0}^{k} p_S(g) \geq 0,5. \tag{3.2}$$

Je weiter α_S von $0,5$ entfernt ist, desto unsymmetrischer ist die Region im Bild. Leider hat sich dieses Merkmal als zu instabil herausgestellt, daher wird es nicht im produktiven Einsatz angewandt. Je nach Zusammenstellung der Test- und Trainingsbilder verbessert oder verschlechtert sich das Ergebnis der Erkennung. Diese Unsicherheit der Verbesserung und mögliche Verschlechterung wiegt den erneuten Rechenaufwand nicht auf.

Wie in Abbildung 3.13 ersichtlich, bietet dieses Merkmal von sich aus keine sehr gute Differenzierung von Regionen mit und ohne Robotern im Bild. Es kann lediglich als Indiz verwendet werden, um zu hohe und zu niedrige Werte auszuschließen. Dazu ist allerdings eine andere Form der Klassifizierung notwendig, da die Logistische Regression dafür nicht geeignet ist.

3.2.1.2. Gradientenbasierte Merkmale

Auch die gradientenbasierten Merkmale einer Region bauen auf Histogrammen auf, allerdings stellen diese nicht die relative Häufigkeit eines Tonwertes dar, sondern die der partiellen Ableitungen. Genauer gesagt werden zwei Histogramme berechnet, welche die Ableitungen nach den Koordinaten x und y beinhalten. Die Algorithmen 3.6 und 3.7 zeigen auf, wie dies vonstatten geht.

$\mathbf{d_{min_x}}$, $\mathbf{d_{max_x}}$ Minimale und maximale partielle Ableitung nach $\mathbf{x}$ einer Bildregion

$\mathbf{d_{min_y}}$, $\mathbf{d_{max_y}}$ Minimale und maximale partielle Ableitung nach $\mathbf{y}$ einer Bildregion

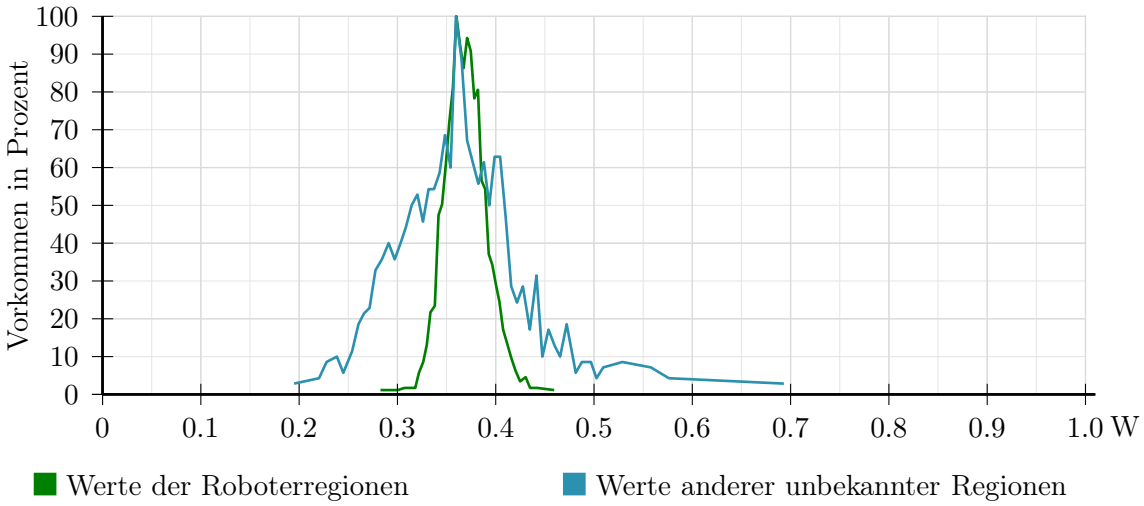


Abbildung 3.13. – Anisotropiekoeffizient des Y-Kanals von Roboterregionen und Regionen mit anderen unbekannten Inhalten. Der Graph zeigt die einzelnen Werte (W) in Relation zur Häufigkeit ihres Vorkommens.

$\mathbf{p_{Y_x}(i)}$ Histogramm der relativen Häufigkeit der partiellen Ableitungen nach **x** auf Basis des **Y**-Kanals (der Helligkeitskanal im *YCbCr*-Farbraum)

$\mathbf{p_{Y_y}(i)}$ Histogramm der relativen Häufigkeit der partiellen Ableitungen nach **y** auf Basis des **Y**-Kanals (der Helligkeitskanal im *YCbCr*-Farbraum)

$\mathbf{a(i)}$ Häufigkeit des Auftretens der partiellen Ableitung **i** in einer Bildregion

Algorithmus 3.6 Erstellung des gradientenbasierten Histogramms nach x

Input: Y-Kanal $\in K$, d_{min_x}, d_{max_x}

1. **for all** $i \in [d_{min_x}, d_{max_x}]$ **do**
2. $\quad p_{Y_x}(i) = \frac{a(i)}{M}$
3. **end for**
4. **return** $p_{Y_x}(i)$

Algorithmus 3.7 Erstellung dea gradientenbasierten Histogramms nach y

Input: Y-Kanal $\in K$, d_{min_y}, d_{max_y}

1. **for all** $i \in \left[d_{min_y}, d_{max_y}\right]$ **do**
2. $\quad p_{Y_y}(i) = \frac{a(i)}{M}$
3. **end for**
4. **return** $p_{Y_y}(i)$

Der Vorteil der partiellen Ableitungen ist die schnelle und ressourcenschonende Umsetzung, da es sich prinzipiell lediglich um eine Subtraktion handelt, wie in Quelltext 3.2 beispielhaft dargestellt.

Quelltext 3.2 – Gradientenbasiertes Histogramm des Y-Kanals in X- und Y-Richtung.

```
1  /**
2   * Berechnet die Histogramme nach x und nach y
3   *
4   * @param Robot region
5   *  Region in einem Bild mit festen Merkmalen
6   * @return double[][][]
7   *  double[0] Histogramm nach x
8   *  double[1] Histogramm nach y
9   */
10 private double[][][] getGradientHistogramm_in_Direction(Robot region) {
11     // liefert das normierte Bild einer Bildregion mit 17px * 21px
12     ImageFast img = region.getNormImage();
13
14     // hist[0] := x-Richtung
15     // hist[1] := y-Richtung
16     // Wert der Bitoperation shift
17     // HISTOGRAM_SHIFT_GRADIENT := 2
18     int[][][] hist = new int[2][(256*2)>>HISTOGRAM_SHIFT_GRADIENT];
19
20     // Initialisierung des Zaehlers
21     int cnt = 0;
22
23     // Initialisiert alle Werte in den Arrays mit 0
24     Arrays.fill(hist[0], 0);
25     Arrays.fill(hist[1], 0);
26
27     // Iteriert ueber die normierte Breite von 17px
28     for (int x = 0; x < img.getWidth(); x++) {
29         // Iteriert ueber die normierte Hoehe von 21px
30         for (int y = 0; y < img.getHeight(); y++) {
31             cnt++;
32
33             // zaehlt fuer jeden Wert den Zaehler im Histogramm hoch
34             hist[0][((img.getY(x, y)-img.getY(x+1, y))>>HISTOGRAM_SHIFT_GRADIENT)
35             // Addition um negative Indizes zu verhindern
36                 + (256>>HISTOGRAM_SHIFT_GRADIENT)]++;
37
38             hist[1][((img.getY(x, y)-img.getY(x, y+1))>>HISTOGRAM_SHIFT_GRADIENT)
39                 + (256>>HISTOGRAM_SHIFT_GRADIENT)]
40         }
41     }
42
43     // normiert das Histogramm, so dass die Summe aller Werte 1 ist
44     return UtilityFunctions.normHistogram(hist, cnt);
45 }
```

Wie in den Zeilen 34 und 38 sichtbar, werden die Tonwerte voneinander abgezogen, so dass ein theoretischer Wertebereich von [-255, 255] möglich ist. Praktisch erfolgt auch hier wieder eine Reduktion der Grauwerte auf 2×64 Tonwerte. Zur Abspeicherung der Werte wird der Wert 64 addiert, da negative Indizes nicht möglich sind.

Daraus resultieren zwei Merkmalsgruppen. Eine enthält die Werte, wie sie hier abgespeichert wurden, bei der anderen wird das Vorzeichen wiederhergestellt. Diese einfache Methode zur doppelten Nutzung berechneter Werte spart nicht nur Ressourcen auf dem Roboter, sondern verdoppelt ohne Mehraufwand die Features. Der Test zur Aussagekraft aller Merkmale in Kapitel 3.2.4 ergab, dass beide Gruppen gleichermaßen notwendige Merkmale bereitstellen.

In Zeile 44 ist erkennbar, dass auch die gradientenbasierten Histogramme am Ende normiert werden, so dass sowohl für partielle Ableitungen nach x als auch nach y gilt:

$$\sum_{i=d_{min}}^{d_{max}} p_Y(i) = 1 \tag{3.3}$$

und $0 \leq p_Y(i) \leq 1$, $i \in [d_{min}, d_{max}]$. So ist gewährleistet, dass gradientenbasierte Merkmale keine Dominanz gegenüber den farbbasierten Features besitzen.

Entropie der Gradienten Genau wie bereits bei den farbbasierten Merkmalen werden auch bei den Gradienten die einzelnen Entropien je Histogramm berechnet. Sie bilden die Grundlage für weitere Features.

Algorithmus 3.8 Entropie der Gradienten

Input: $p_{Y_x}(i)$, $p_{Y_y}(i)$

1. $H_{D_x} = -\sum_{i=d_{min_x}}^{d_{max_x}} (p_{Y_x}(i) \cdot log_2(p_{Y_x}(i)))$
2. $H_{D_y} = -\sum_{i=d_{min_y}}^{d_{max_y}} (p_{Y_y}(i) \cdot log_2(p_{Y_y}(i)))$
3. **return** H_{D_x}, H_{D_y}

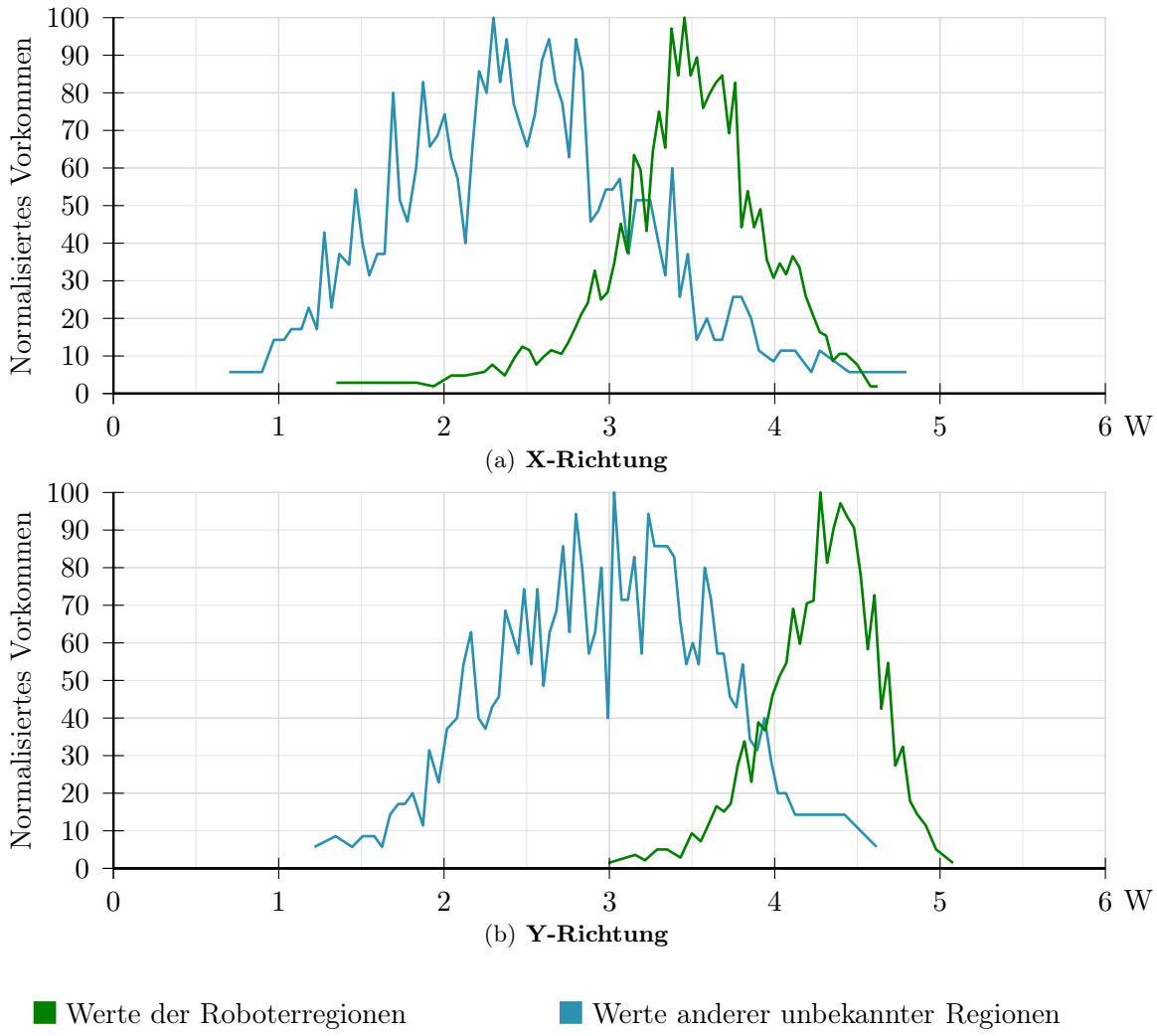
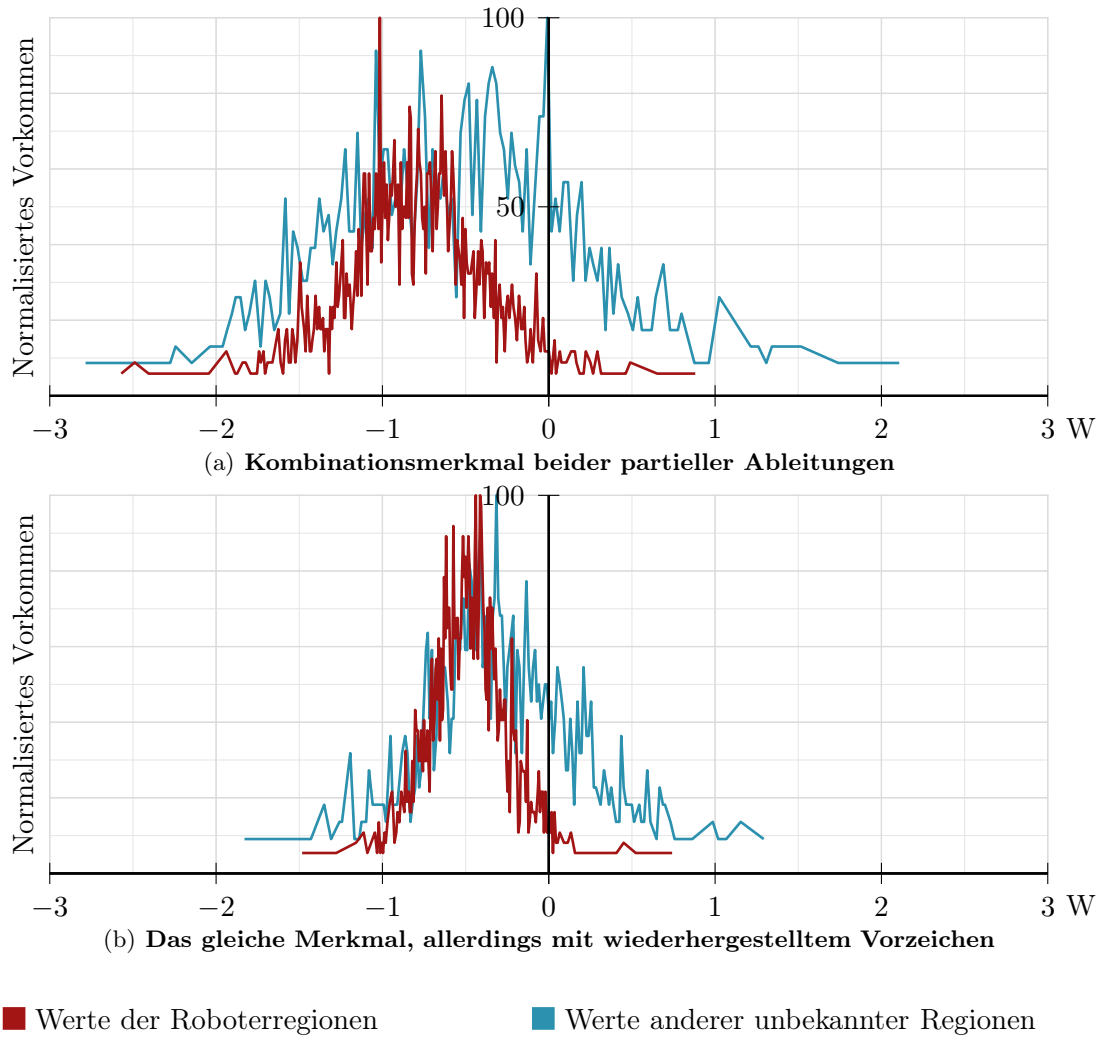


Abbildung 3.14. – Entropie der Gradienten in beide Richtungen. Der Graph zeigt die einzelnen Werte (W) in Relation zur Häufigkeit ihres Vorkommens.

Wie in Abbildung 3.14 ersichtlich, bilden die partiellen Ableitungen nach Y ein deutlich aussagekräftigeres Merkmal zur Bestimmung von Robotern in Bildregionen, daher kommt diese Gruppe an Features vorrangig zum Einsatz.

Ein weiteres, drittes Merkmal wird durch die Kombination der Entropien beider partieller Ableitungen gebildet. Es entsteht durch Subtraktion der zwei Werte, wie in Formel 3.4 dargestellt ist.

$$\text{Feature}_{neu} := H_{D_y} - H_{D_x} \tag{3.4}$$



(a) **Kombinationsmerkmal beider partieller Ableitungen**

(b) **Das gleiche Merkmal, allerdings mit wiederhergestelltem Vorzeichen**

Abbildung 3.15. – Das subtraktiv erstellte Kombinationsmerkmal beider partieller Ableitungen mit Berechnungsdaten, deren Vorzeichen erhalten wurde und ohne die Wiederherstellung. Der Graph zeigt die einzelnen Werte (W) in Relation zur Häufigkeit ihres Vorkommens.

Eine Darstellung, wie sich dieses kombinierte Merkmal auswirkt, ist in Abbildung 3.15 zu finden. Es sind die Varianten mit und ohne Vorzeichen dargestellt. Die Basis dieser Berechnungen bilden die rein positiv gespeicherten Werte. Wie bereits im vorherigen Abschnitt erläutert, kommt zu jedem Feature dieser positiven Merkmalsgruppe das entsprechende Äquivalent hinzu, welches sich durch die Wiederherstellung des Vorzeichens definiert. Wie die Umsetzung aussieht, ist in Algorithmus 3.9 auf Seite 47 ersichtlich.

Die Graphen der Abbildung 3.15 zeichnen ein eindeutiges Bild, dass diese Merkmalsgruppe nicht im gewünschten Umfang dazu fähig ist, zwischen Regionen in Kamerabildern

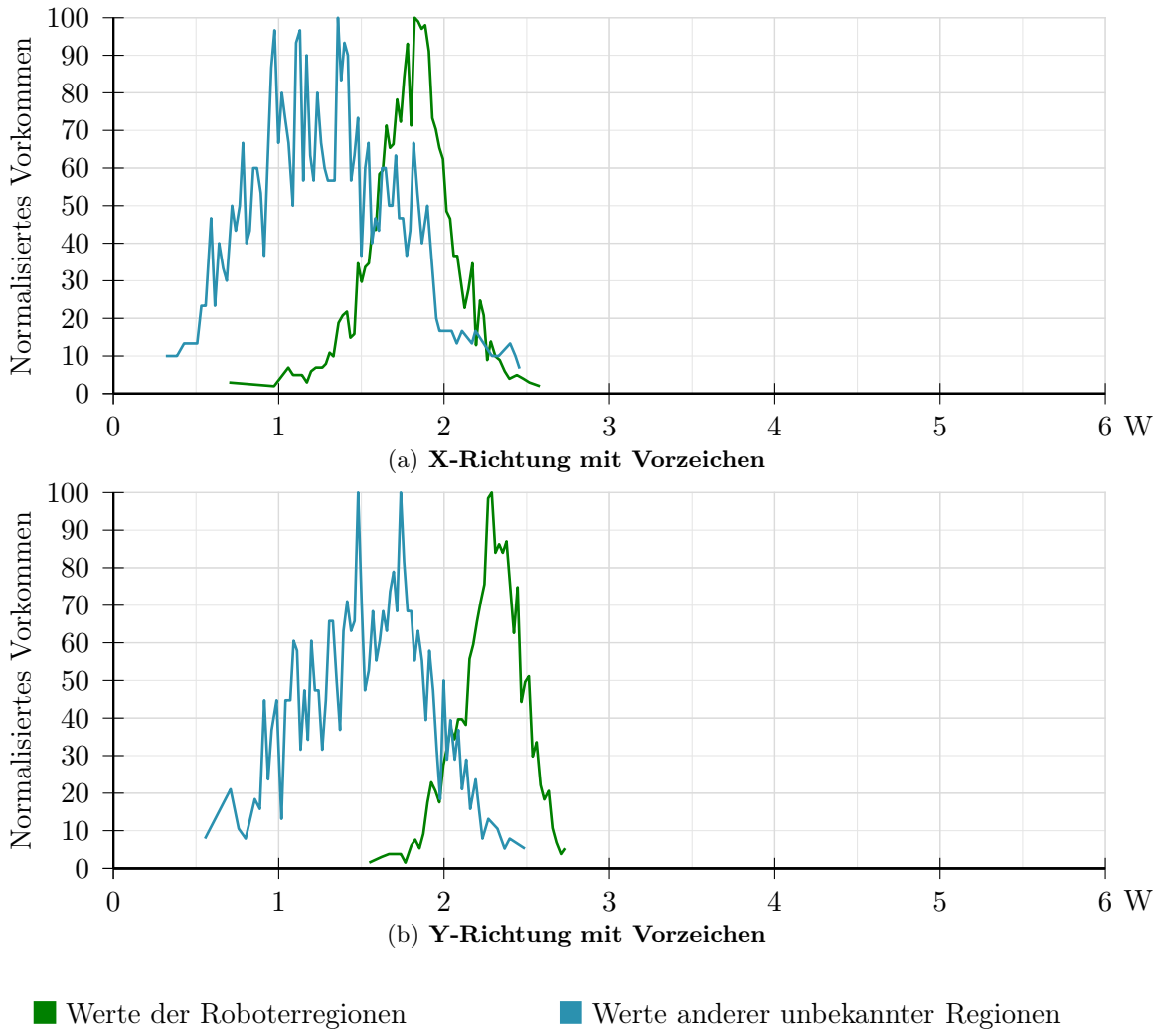


Abbildung 3.16. – Entropie der Gradienten mit Berechnungsdaten, deren Vorzeichen erhalten wurde, in beide Richtungen. Der Graph zeigt die einzelnen Werte (W) in Relation zur Häufigkeit ihres Vorkommens.

zu unterscheiden, welche Roboter zeigen und solchen, die anderen Inhalts sind. Allerdings hat sich bei den automatisierten Tests von Kapitel 3.2.4 bezüglich des Einflusses von Merkmalen auf die Erkennungsrate herausgestellt, dass diese Gruppe ihre Berechtigung zur Umsetzung besitzt. Die Variante, welche zur Berechnung Daten mit Vorzeichen verwendet, hat positive Auswirkungen auf die Klassifizierung der Bildbereiche.[57]

Die Abbildung 3.16 zeigt ebenfalls die Entropien beider partieller Ableitungen, wobei die Berechnung mit Basisdaten erfolgte, bei denen das Vorzeichen wiederhergestellt wur-

[57] Im Mittel liegt die Verbesserung bei ca. 0.5%.

Algorithmus 3.9 Wiederherstellung des Vorzeichens

Input: p_Y, d_{min}, d_{max}

1. **for all** $i \in [d_{min}, d_{max}]$ **do**
2. $\quad p_{Y_i - d_{min}} = p_{Y_i}$
3. **end for**
4. **return** p_Y

de. Anhand der beiden Graphen 3.16 **(a)** und **(b)** wird deutlich, dass die Y-Richtung mehr Aussagekraft besitzt. Die Werte von Roboterregionen heben sich deutlich ab.

Entropie der Liniensummen in X und Y Richtung Eine weitere Merkmalsreihe beruht nicht wie zuvor auf den Pixelwerten einzelner Felder, sondern auf den Summen der Zeilen und Spalten. Die Berechnungen selbst ändern sich nicht, lediglich der zugrunde liegende Datenbestand. Die partiellen Ableitungen erfolgen auf die einzelnen Liniensummen, sowohl horizontal als auch vertikal.

Die Berechnung der Entropie erfolgt analog zur vorherigen und wird nicht nochmals explizit aufgelistet. Unterschiede gibt es hingegen bei der Datengrundlage, da zur Ermittlung zwei weitere Histogramme erstellt werden müssen, eines für die Zeilen- und eines für die Spaltensummen.

Wie genau die Berechnung erfolgt, ist in Algorithmus 3.10 gezeigt. Da auch hier die Farbkanäle keine starke Verbesserung der Erkennung versprechen, erfolgt die Berechnung ausschließlich auf Basis des Helligkeitskanals. Dies geschieht sowohl für die partielle Ableitung nach x, als auch nach y.

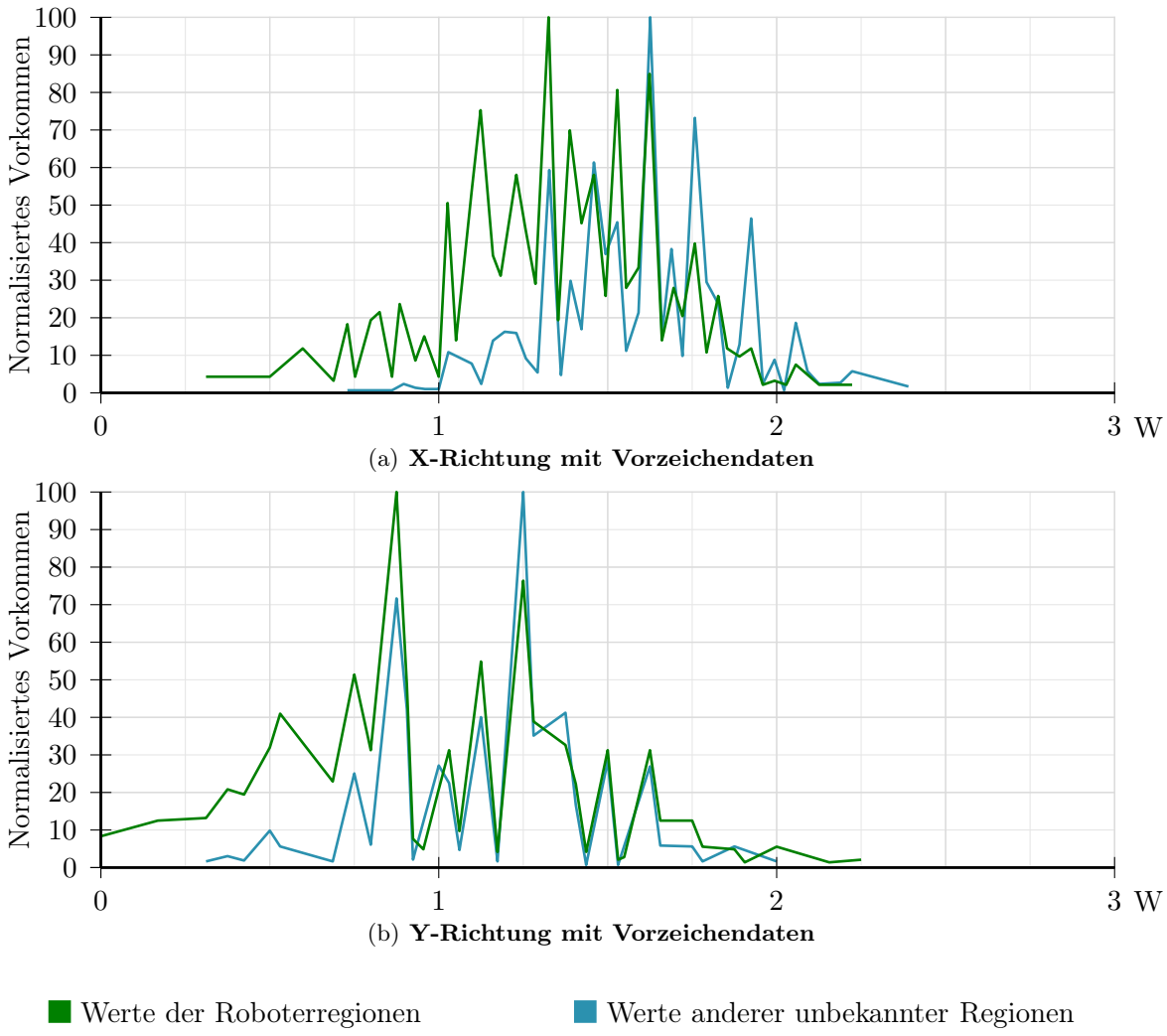
Algorithmus 3.10 Histogramm der Liniensummen von Spalten und Zeilen

Input: L, R

1. **for all** $y \in L$ **do**
2. $\quad d_y = \frac{\partial \sum_{x \in R} s(x,y)}{\partial y}$
3. $\quad p_{Y_y}(d_y) = \frac{a(d_y)}{M}$
4. **end for**
5. **for all** $x \in R$ **do**
6. $\quad d_x = \frac{\partial \sum_{y \in L} s(x,y)}{\partial x}$
7. $\quad p_{Y_x}(d_x) = \frac{a(d_x)}{M}$
8. **end for**
9. **return** p_{Y_y}, p_{Y_x}

In Abbildung 3.17 ist die Entropie der Liniensummen dargestellt. Betrachtet man die beiden Graphen, so wird ersichtlich, dass dieses Merkmal in dieser Form nicht aussagekräftig ist. Tests haben dies bestätigt, wodurch es so nicht in die Umsetzung auf dem Roboter gelangt ist. Allerdings erwies es sich bei der Verwendung von Merkmalen höherer Ordnung als essenzielles Feature. Die Varianten zweiter Ordnung dieser Merkmalsreihe erhöhen die Erkennung um mehrere Prozentpunkte.

Auch in dieser Merkmalsgruppe existiert das dritte Basismerkmal, welches sich aus der Subtraktion der beiden vorherigen Features ergibt. Dabei wird die Entropie der Spalten



(a) **X-Richtung mit Vorzeichendaten**

(b) **Y-Richtung mit Vorzeichendaten**

Abbildung 3.17. – Entropie der Liniensummen beider partieller Ableitungen, wobei die Vorzeichen der Basisdaten erhalten wurden. Der Graph zeigt die einzelnen Werte (W) in Relation zur Häufigkeit ihres Vorkommens.

von der Entropie der Zeilen abgezogen. Alle drei Merkmalsarten werden gleichermaßen sowohl mit als auch ohne Vorzeichen auf Grundlage des Histogramms erstellt, so dass die doppelte Anzahl an Merkmalen zustande kommt. Diese Merkmalsermittlung erfolgt in der Umsetzung allerdings nicht getrennt, sondern innerhalb eines gemeinsamen Schrittes.

Vektorlängen der Entropien Eine letzte Merkmalsreihe bilden die Vektorlängen. Sie werden aus Kombinationen von allen Merkmalswerten gleichermaßen erstellt. Die Berechnung erfolgt bei allen gleich, wie beispielhaft in Formel 3.5 dargestellt.

$$C = \sqrt{\sum_{c_i \in C} c_i^2} \tag{3.5}$$

Tests haben ergeben, dass die Vektorlängen als Merkmalsgruppe im Ganzen nicht aussagekräftig genug sind, wie in Kapitel 3.2.4 ersichtlich wird. Daher kommt diese Featuregruppe bei der Umsetzung auf dem Roboter nicht zum Einsatz.

3.2.2. Merkmale höherer Ordnung

Die im vorherigen Abschnitt beschriebenen Merkmale bilden den Grundstock zur Beschreibung einer Region. Belässt man es bei diesen, so werden 90% aller unbekannten Bildbereiche korrekt klassifiziert. In Kapitel 3.3 ist dieser Vorgang erläutert. Allerdings werden Wechselwirkungen zwischen verschiedenen Features bisher nicht mit in die Erkennung einbezogen. So kann es beispielsweise vorkommen, dass Gradienten in X- und in Y-Richtung multipliziert ebenfalls ein aussagekräftiges Merkmal bilden.

Um diese höhere Ebene an Merkmalen nicht außer Acht zu lassen, wurde ein Algorithmus geschrieben, der zu allen aktiven Features der ersten Ordnung neue bildet und dem Merkmalsvektor hinzufügt. Dazu wird jedes Feature mit allen anderen, einschließlich sich selbst, multipliziert und als neues Merkmal betrachtet, beispielsweise $x_1 x_3$.

Auf Grund dieser neuen Features konnte die korrekte Erkennungsrate im Schnitt um 2% auf ca. 92% gesteigert werden. In diese Berechnung sind allerdings nur Merkmale bis zur zweiten Ordnung integriert. Eine zusätzliche Multiplikation der 3. Ebene, beispielsweise $x_1 x_3 x_5$, erbrachte keine signifikante Verbesserung und wird daher, auch zur Vermeidung unnötiger Berechnungen, nicht verwendet.

3.2.3. Normalisierung der Merkmale

Je nach Art der einzelnen Features können diese sehr unterschiedliche Wertintervalle aufweisen. Beispielsweise sind die Mittelwerte der Kanäle im Intervall $[0, 255]$ zu fin-

den, während die Entropie eines 8 Bit Kanals [0, 8] als Grenzen aufweist. Diese starke Schwankung der möglichen Werte einzelner Merkmale kann dazu führen, dass bestimmte Argumente andere dominieren und entsprechend geringer gewichtet werden müssten.

Um derartigen Effekten vorzubeugen, werden alle Features normalisiert, beziehungsweise so angeglichen, dass alle im gleichen Intervall angesiedelt sind, welches in diesem Fall zwischen $-2 \leq x \leq 2$ liegt.

Die Normalisierung der Merkmale erfolgt mit Hilfe der Standardabweichung. Wie die Berechnung erfolgt, ist in Algorithmus 3.11 ersichtlich. Der Algorithmus erhält eine Liste mit allen vorhandenen Features[58]. Für jedes wird zuerst der Mittelwert über die Anzahl an Beispielen berechnet, wie in Zeile 3 sichtbar. Dies ist notwendig, um in Zeile 4 die Varianz eines jeden Merkmals individuell zu bestimmen, aus welcher in Zeile 5 die Standardabweichung ermittelt wird.

Diese Standardabweichung ist das eigentliche Ziel des Algorithmus, da durch Division mit ihr in den Zeilen 6 bis 8 jedes x_i, sprich jedes Trainingsbeispiel eines jeden Merkmals, normalisiert wird.

$\mathbf{V_{n,m}}$ Matrix aller Merkmale der einzelnen Beispiele mit **n** Spalten und **m** Zeilen

b Vektor mit **m** Werten einer Merkmalskategorie

$\mathbf{b_{normalized}}$ normalisierter Vektor **b**

Algorithmus 3.11 Normalisierung der einzelnen Merkmale

Input: $V_{n,m}$

1. **for all** $\mathbf{b} \in V_{n,m}$ **do**
2. $\quad \bar{b} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{b_i}$
3. $\quad s = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{b_i} - \bar{b})^2$
4. $\quad s = \sqrt{s}$
5. $\quad$ **for all** $\mathbf{b_i} \in \mathbf{b}$ **do**
6. $\quad\quad b_{normalized,i} = \frac{\mathbf{b_i}}{s}$
7. $\quad$ **end for**
8. **end for**
9. **return** $\mathbf{b_{normalized}}$

[58] Genauer gesagt eine Matrix, deren Spalten die Merkmale darstellen und Zeilen die einzelnen Trainingsbeispiele.

3.2.4. Verwendete Merkmale

Unter Beachtung der Quadrate kommen bisher über 40 Features zum Einsatz, die allerdings unterschiedlich aussagekräftig sind. Um Rechenzeit und -aufwand zu sparen, empfiehlt es sich, diese Anzahl zu reduzieren. Dies kann manuell geschehen, anhand der Wichtungen der einzelnen Merkmale,[59] indem gering gewichtete Features eliminiert werden, oder automatisiert. Im Rahmen dieser Arbeit kam die zweite Variante zum Einsatz.

Es wurde eine Methode geschrieben, welche unter Verwendung eines konstanten Testbilddatensatzes jeweils ein Merkmal entfernt und mit den verbliebenen erneut eine Trainingsphase durchläuft. Anschließend werden die Ergebnisse der Erkennungsraten mit den vorherigen verglichen und bewertet. Das Bewertungskriterium ist dabei, dass die einzelnen Werte der Erkennungsraten sich nicht stärker ändern als um $0,01$. Ist diese Bedingung erfüllt, wird das jeweilige Feature entfernt und der Prozess beginnt erneut mit den verbliebenen Features. Sollte dies nicht der Fall sein, wird dieser Parameter als *notwendig* markiert und aus dem Eliminierungsprozess genommen. Die Vergleichswerte liefern bei der Initialisierung des Algorithmus die Erkennungsraten der Ausgangsanzahl an Features, danach jeweils die Erkennungsraten nach der letzten Featureeliminierung.

Die Auswertung des Algorithmus hat ergeben, dass lediglich sieben der ursprünglich über 40 Features für die Erkennung von Robotern auf einem Kamerabild notwendig sind, darunter zwei quadratische:

- die gradientenbasierte Entropie in Y-Richtung
- die gradientenbasierte Entropie in Y-Richtung mit Vorzeichen
- die gradientenbasierte Entropie in X-Richtung mit Vorzeichen minus des gleichen Features in Y-Richtung
- die Entropie des Y-Farbkanals / Helligkeitskanal
- die mittlere quadratische Abweichung des Y-Kanals
- die Entropie der Liniensummen der Gradienten in X-Richtung multipliziert mit dem gleichen Feature unter Vorzeichenerhaltung
- die Entropie der Liniensummen der Gradienten in Y-Richtung unter Vorzeichenerhaltung multipliziert mit sich selbst

[59] Siehe hierzu die Beschreibung zur Implementation der Klassifikationen ab Kapitel 3.3.1.

Hinzu kommt das initiale Feature x_0, welches immer den Wert 1 besitzt. Da dies einer Konstante entspricht, wird sie im Rahmen dieser Arbeit nicht zur Anzahl der Merkmale hinzugezählt. Bemerkenswert an den Ergebnissen dieses Tests ist, dass nur zwei Features direkt auf den Farbkanälen basieren und auch hier ausschließlich auf dem Y-Kanal. Da dieser Kanal auch bei der Ermittlung der Gradienten zum Einsatz kommt, werden die Farbkanäle *Cb* und *Cr* an keiner Stelle innerhalb dieser Erkennung verwendet. Folgt man dieser Beobachtung, so kann die Aussage getroffen werden, dass der Helligkeitskanal allein bereits ausreichend Informationen liefert, um einen Roboter auf dem Feld zu identifizieren.

3.3. Klassifizierung von Regionen

Was man lernen muß, um es zu tun, das lernt man, indem man es tut.

Aristoteles (384 – 322 v. Chr.)

Um einzelne Bereiche in den von den Kameras der Roboter aufgenommenen Bildern klassifizieren zu können, muss ein Programm zuerst lernen, was ein Roboter ist und was nicht. Dazu erhält es die ab Kapitel 3.2 vorgestellten Merkmale einer jeden Region, über die es eine Aussage treffen muss.

Das Grundproblem dieser Arbeit ist das Erkennen von Robotern. Die Entscheidung, ob es sich um einen Nao handelt oder nicht, kann von einem Algorithmus nicht selbstständig ermittelt werden. Ein Programm muss wissen, welche Merkmale ein Objekt ausmachen. Eine Klassifikation ohne dieses Wissen, wie es beispielsweise bei einem Clustering-Problem[60] der Fall ist, kommt demzufolge nicht in Frage. Daher wird eine Form des unterstützten Lernens eingesetzt, bei der ein künstliches System, der Algorithmus, anhand von Beispielen *lernt*, dieses erworbene Wissen nach Beendigung der Lernphase verallgemeinert und auf neue, bisher unbekannte Beispiele anwendet. Daraus folgt, dass es nicht die vorgelegten Informationen auswendig lernt, sondern in den Daten verschiedene Gesetzmäßigkeiten „erkennt", um auch diese unbekannten Beispiele zu beurteilen.

Nachdem die Testbilder erstellt, Regionen in den Bildern erkannt und von jedem dieser Bereiche die einzelnen Merkmale berechnet und normalisiert wurden, gilt es, diese Daten zu verarbeiten. Die errechneten Features werden einem Algorithmus übergeben, welcher diese *lernt* und am Ende in der Lage ist, ein gegebenes, vorher unbekanntes, Teilbild zu klassifizieren in *Roboter* oder *kein Roboter*.

[60] Das Finden von Klassen in einem gegebenen Datensatz ohne die Vorgabe einer Lösung, d.h. ohne den y-Vektor.

Dabei stehen eine Reihe von Algorithmen zur Auswahl, welche diese Klassifizierung durchführen können, wobei jeder seine eigenen Vor- und Nachteile besitzt. Im Rahmen dieser Arbeit kommt die *Logistische Regression* zum Einsatz, eine Spezialform des unterstützten Lernens. Das bedeutet, dass ein Algorithmus trainiert wird, indem die „richtigen" Antworten gegeben werden. Die *Logistische Regression* lernt eine Funktion aus gegebenen Paaren von Ein- und Ausgaben. Während der Lernphase wird dem Algorithmus der korrekte Funktionswert zu einer Eingabe bereit gestellt. Sie stellt die effizienteste Realisierung im Hinblick auf die begrenzten Ressourcen der Nao-Plattform dar, da nach Ermittlung der Features lediglich eine Addition der gewichteten Parameter notwendig ist, um eine Klassifikationshypothese auf dem Roboter zu erstellen.[61]

3.3.1. Logistische Regression

Die Logistische Regression ist ein klassifizierender Algorithmus, wobei die zu vorherzusagende Variable y, sprich das Ergebnis der Klassifikation, diskrete Werte annimmt. Das Wort *Regression* beschreibt sinngemäß eine Vorhersage mit kontinuierlichen Werten. Diese Bezeichnung ist historisch bedingt und nicht mit der eigentlichen Wortbedeutung belegt. Beispiele für die Anwendung eines solchen Algorithmus kann sowohl die Kategorisierung von Emails in Spam oder kein Spam sein, als auch das Erkennen eines bestimmten Buchstabens, eines Autos oder eines Tieres. Es handelt sich um eine Form des unterstützten Lernens, das heißt für jedes Trainingsbeispiel wird die „richtige Antwort" gegeben.[62]

Im Falle dieser Arbeit sollen gegebene Regionen in Bildern, die ein Roboter mit seinen Kameras aufnimmt, klassifiziert werden. Dabei beschränkt sich die Anzahl an möglichen Klassen auf zwei – Roboter oder kein Roboter – und der Wertebereich des Outputs des Algorithmus demnach auf $y = \{0, 1\}$, wobei 0 die negative Klasse, also kein Roboter, und 1 die positive Antwort beschreibt. Vereinfacht dargestellt ist dies ein Wahrheitswert, das heißt es wird eine Anfrage an den Klassifikator gestellt, ob eine Bildregion einen Roboter beinhaltet und dieser antwortet mit ja oder nein. Allerdings entspricht die Antwort eher einer gewichteten Wahrscheinlichkeit, einem Wert zwischen Null und Eins, welcher aussagt, wie wahrscheinlich ein Ja für diese Region ist.[63] Ob dies ausreicht, um einen Bereich positiv zu klassifizieren, bestimmt ein Schwellenwert, welcher beliebig gewählt werden kann. Wird dieser Wert mit der Vorhersage erreicht oder überschritten, so gibt der Klassifikator für die Region eine 1 aus und betrachtet sie als einen Roboter.

[61] Hinzu kommt die Sigmoidfunktion, welche in Formel 3.9 dargestellt ist. Vgl. [Ng11].

[62] Vgl. [Ng11] – die Videos 6.1 bis 6.7 zur Logistischen Regression. Eine aktuelle Version dieses Kurses wird demnächst unter `https://www.coursera.org/course/ml` verfügbar sein.

[63] Ein konkretes Beispiel ist die Rückgabe von 0.7, was einer 70% Wahrscheinlichkeit entspricht, dass es sich um einen Roboter handelt.

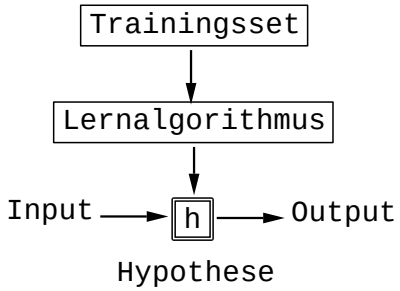


Abbildung 3.18. – Das Mapping von x zu y durch die Hypothese **h**. Vgl. [Ng11], Video 2.1.

3.3.2. Arbeitsweise des Algorithmus

Die folgende Auflistung erklärt die grundlegende Notation der Variablen, sowie deren Verwendung und Bedeutung.[64]

Notation:

m Anzahl an Trainingsbeispielen

n Anzahl an Merkmalen

x Inputvektor oder auch Merkmalsvektor mit n Merkmalswerten

y Output- oder auch Zielvariable, die versucht wird vorherzusagen

(x,y) ein Trainingsbeispiel

$(x^{(i)},y^{(i)})$ i_{tes} Trainingsbeispiel

Die Abbildung 3.18 veranschaulicht die Arbeitsweise der *logistischen Regression* als Beispiel für *Unterstütztes Lernen*. Der lernfähige Algorithmus erhält ein Set an Beispielen zum Trainieren. Anschließend gibt dieser eine Funktion h aus, auch *Hypothese* genannt, welche als Input die Merkmale x erhält und als Rückgabewert den geschätzten Wert von y. Im Falle dieser Arbeit ist $y = 1$, falls es sich bei den gegebenen Merkmalen um einen Roboter handelt, und $y = 0$, falls nicht.

Bei der Berechnung der Hypothese wird jeder Wert des Merkmalsvektors x individuell mit Hilfe eines Parametervektors Θ gewichtet. Standardmäßig besitzt x_0 immer den Wert **1**, so dass bei der Berechnung der Hypothese stets $\Theta_0 * 1 = \Theta_0$ ergibt, wie anhand der Beispielhypothese einer linearen Regression in Formel 3.7 ersichtlich.

[64] Basierend auf den Ausführungen des Stanford Machine Learning Course von Andrew Ng [Ng11].

$$\Theta = \begin{bmatrix} \Theta_0 \\ \Theta_1 \\ \Theta_2 \\ \dots \end{bmatrix} \; x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \end{bmatrix} \tag{3.6}$$

$$h_{Beispiel}(x) = \sum_{i=0}^{n} \Theta_i x_i = \Theta^T x = \begin{bmatrix} \Theta_0 \Theta_1 \Theta_2 \dots \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \end{bmatrix} \tag{3.7}$$

Da es sich um ein Klassifikationsproblem mit zwei Klassen handelt, soll die Hypothese zwei diskrete Werte $\{0, 1\}$ ausgeben. Dies ist mit Formel 3.7 und der Eigenschaft $0 \leq h_\Theta(x) \leq 1$ bisher nicht sichergestellt. Daher wird die Hypothese wie folgt abgewandelt:

$$h_\Theta(x) = g(\Theta^T x) \tag{3.8}$$

Die Funktion g stellt sicher, dass bei der Vorhersage diskrete Werte auftreten. Sie ist definiert als:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{3.9}$$

Diese Funktion g heißt *Sigmoidfunktion.*[65] Sie gewährleistet, dass die Bedingung $0 \leq h_\Theta(x) \leq 1$ erfüllt ist, da sie zwei Asymptoten besitzt, bei 0 und bei 1. Wie in Abbildung 3.19 erkennbar, ist $\lim_{x \to -\infty} g(z) = 0$ und $\lim_{x \to \infty} g(z) = 1$.

3.3.2.1. Wahl des Schwellwertes

Wie bereits zuvor angedeutet, kann $h_\Theta(x)$ als die Wahrscheinlichkeit P interpretiert werden, dass $y = 1$ ist für einen gegebenen Input x unter zu Hilfenahme der Parameter Theta Θ.

$$h_\Theta(x) = P(y = 1 | x; \Theta) \tag{3.10}$$

Daher können die folgenden Aussagen getroffen werden:

$$P(y = 0 | x; \Theta) + P(y = 1 | x; \Theta) = 1 \tag{3.11}$$

$$P(y = 0 | x; \Theta) = 1 - P(y = 1 | x; \Theta) \tag{3.12}$$

[65] Auch Daher auch der Name des Algorithmus: *Logistische Regression.*

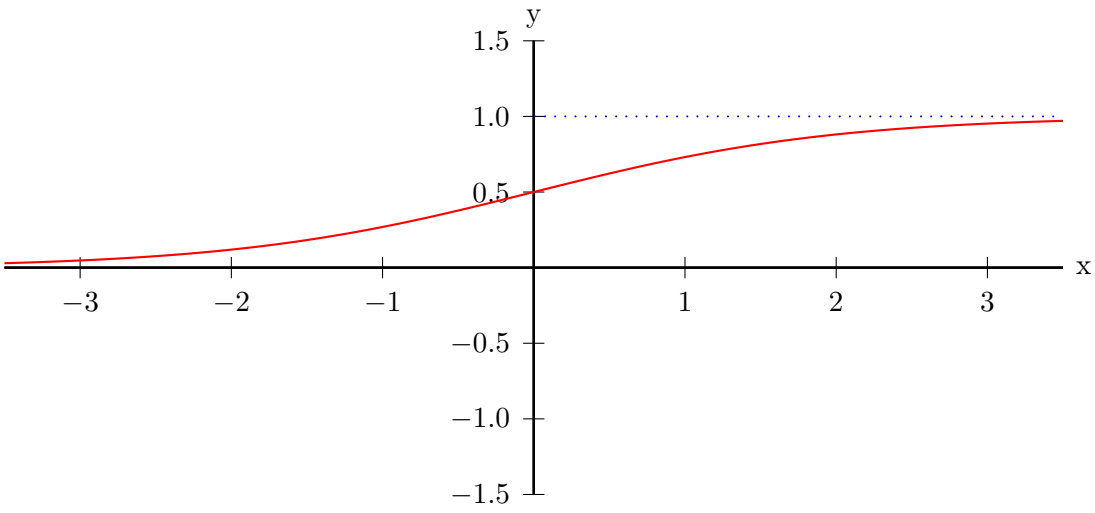


Abbildung 3.19. – Sigmoidfunktion.

Die Wahrscheinlichkeit für $y = 0$ und $y = 1$ muss zusammen 1 ergeben. Daher kann bei berechneter $h_\Theta(x)$ ebenfalls die Wahrscheinlichkeit von $P(y = 0|x; \Theta)$ ermittelt werden.

Das Ergebnis y des Algorithmus soll nach Ablauf aller Berechnungen nur die Werte 0 oder 1 beinhalten. Daher wird ein Schwellwert benötigt, welcher ab einer gegebenen Wahrscheinlichkeit für $y = 1$ die Entscheidung trifft, dass es sich tatsächlich um ein positives Beispiel, sprich in diesem Fall einen Roboter, handelt.

Angenommen der gewählte Schwellenwert betrüge 0,5. Die Logistische Regression würde $y = 1$ vorhersagen, falls $h_\Theta(x) \geq 0{,}5$ und $y = 0$ bei $h_\Theta(x) < 0{,}5$. Das hier bei exakt 0,5 y auf 1 gesetzt wird ist keine einflussreiche Detailentscheidung und kann auch umgekehrt behandelt werden. Die Bedeutung dessen wird in Abbildung 3.20 deutlich. Sobald $\Theta^T x \geq 0{,}5$ ist auch $h_\Theta(x) = g(\Theta^T x) \geq 0{,}5$, sichtbar anhand der blauen und grauen Markierung innerhalb der Abbildung.

Daraus folgt, dass bei dem zufällig gewählten Schwellenwert 0,5 y immer dann 1 wird, wenn $\Theta^T x \geq 0{,}5$. Dies garantiert allerdings nicht die optimale Vorhersage, da 0,5 ohne entsprechende Datengrundlage verwendet wurde. Um den Schwellenwert anzupassen und die maximale Erkennungsrate zu erreichen, wurde eine Reihe von Tests durchgeführt.

Von den Schwellenwerten 0 bis 1 bei einer Schrittweite von 0,01 wurden je Iteration 15 Testdatensätze zufällig erstellt und klassifiziert. Der Mittelwert dieser 15 Durchläufe pro Schwellenwert wurde gespeichert und in Abbildung 3.21 dargestellt. Auffällig ist, dass sich im Bereich von $0{,}72 \leq x \leq 0{,}75$ die Linien kreuzen. Es ist anzunehmen, dass in diesem Bereich der optimale Schwellenwert zu finden ist.

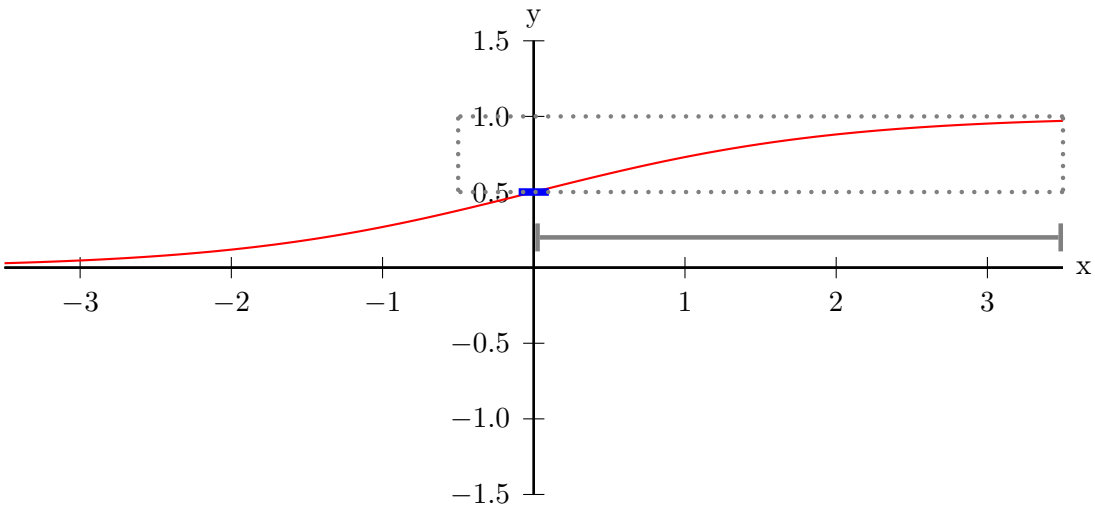


Abbildung 3.20. – Sigmoidfunktion mit Schwellenwert bei 0,5.

Allerdings gibt es für die Wahl des Schwellenwertes zusätzliche Bedingungen, welche die Ermittlung des optimalen Wertes anhand des Graphen nur bedingt möglich macht. Zum einen soll die Rate der falsch positiv als Roboter klassifizierten Regionen nicht größer als 4% werden, zum anderen die gesamte Erkennungsrate nicht unterhalb von 90% sinken. Es muss ein Gleichgewicht zwischen beiden gefunden werden, da, sobald die *false positives*[66] sinken, auch die Korrektheit der Erkennung im Ganzen sinkt, wie in Abbildung 3.21 ersichtlich.

Dazu werden in Tabelle 3.3 alle Werte aus der potenziell interessanten Schwellenwertregion dargestellt. Mit rot sind alle Zeilen gekennzeichnet, die herausfallen, da sie nicht den eben angesprochenen Vorgaben entsprechen. Interessante Schwellenwerte sind bei 0,88 bis 0,91 zu finden. Hier sinkt die Rate der fälschlicherweise als positiv klassifizierten Regionen um nahezu einen vollen Prozentpunkt unterhalb von 2%, wohingegen die komplette Erkennungsrate nur geringfügig abfällt. Basierend auf den Ergebnissen dieser Auswertung ist der optimale Schwellwert, entsprechend den zuvor definierten Bedingungen, bei **0,89** zu finden. Dieses Ergebnis wird in Abbildung 3.22 dargestellt.

[66] Englisch für falsch positiv deklarierte Beispiele. Wie bereits zuvor definiert, handelt es sich um Regionen, die keinen Roboter zeigen, aber die als solche erkannt wurden.

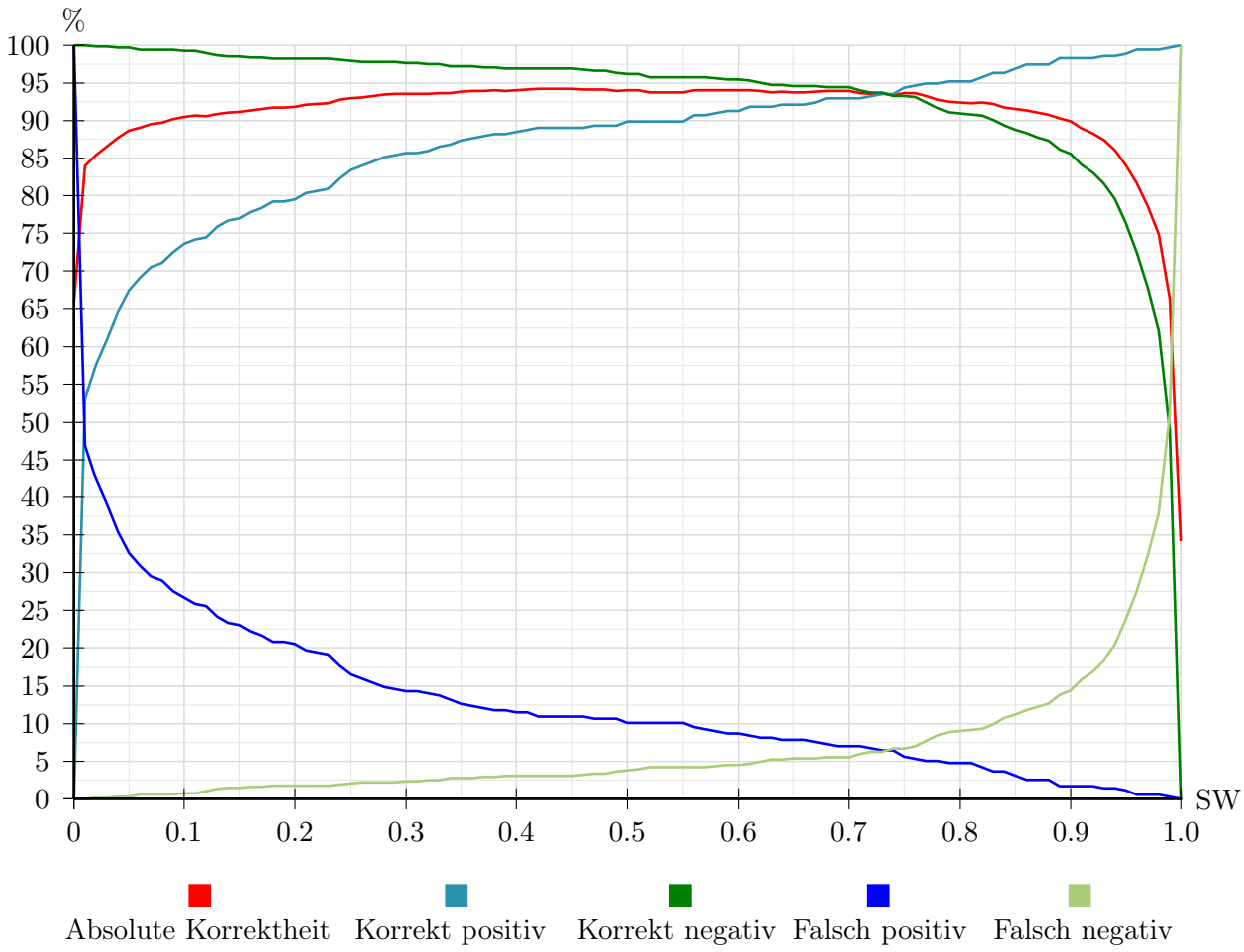


Abbildung 3.21. – Die Abbildung zeigt die Erkennungsraten in Abhängigkeit vom jeweiligen Schwellenwert.

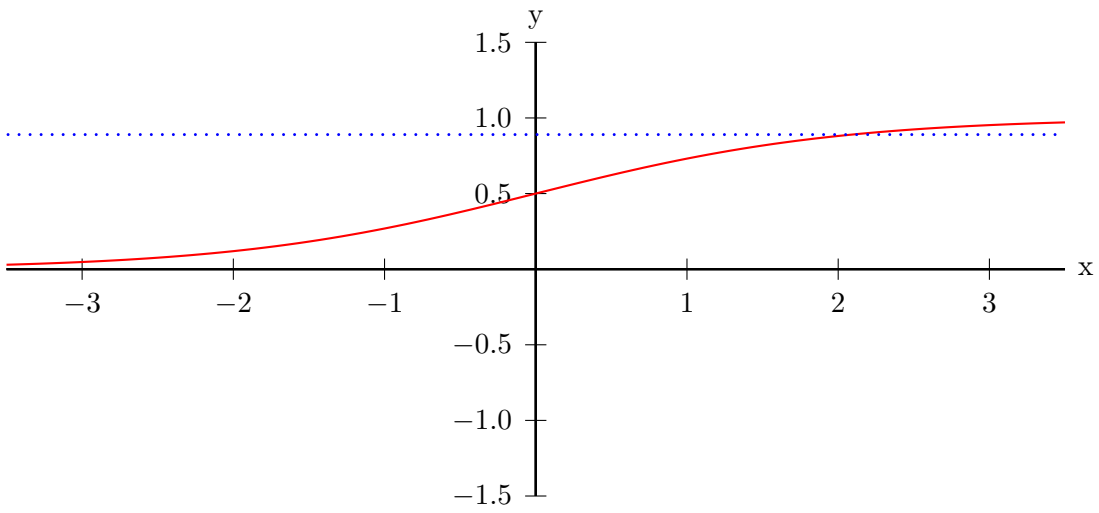


Abbildung 3.22. – Schwellenwert von 0,89 bei Anwendung der Sigmoidfunktion.

Schwellenwert	Korrekte Erkennung	falsch positive Regionen
0,77	93,28%	5.056%
0,78	92,80%	5,056%
0,79	92,51%	4,775%
0,80	92,41%	4,775%
0,81	92,32%	4,775%
0,82	92,41%	4,213%
0,83	92,22%	3,651%
0,84	91,74%	3,651%
0,85	91,55%	3,089%
0,86	91,36%	2,528%
0,87	91,07%	2,528%
0,88	90,78%	2,528%
0,89	90,30%	1,685%
0,90	89,92%	1,685%
0,91	88,96%	1,685%
0,92	88,29%	1,685%
0,93	87,42%	1,404%

Tabelle 3.3. – Auswertung der Erkennungsraten in Abhängigkeit zum Schwellenwert.

3.3.2.2. Wahl einer Konstanten zur Sigmoidfunktion

Die Sigmoidfunktion kann im Exponenten um eine Konstante t erweitert werden, die den Verlauf der Funktion beeinflusst, siehe Gleichung 3.13.

$$g(z,t) = \frac{1}{1+e^{-(t\cdot z)}} \tag{3.13}$$

Bisher wurde t immer gleich 1 gesetzt, was dem regulären Verlauf der Sigmoidfunktion entspricht. Allerdings ist der Anstieg von $g(z)$ verglichen mit $g(z,t)$, bei $t > 1$, flach, was die Vermutung nahelegt, dass durch die Wahl einer von 1 verschiedenen Konstante das Ergebnis der Erkennung verbessert werden kann. In Abbildung 3.23 ist die Sigmoidfunktion mit verschiedenen Werten für t dargestellt, welche aufzeigen, wie sich der Verlauf ändert. Die rote Kurve zeigt den Fall von $t = 1$, in türkis $t = 2{,}5$ und in grün ist $t = 6$ dargestellt.

Um die Auswirkungen des t-Faktors auf die Erkennung herauszufinden, wurde eine Testreihe durchgeführt, die für verschiedene Werte von t die Erkennungsraten speichert. Begonnen bei t = 0,1 bis 10, mit einer Steigerung um 0,1, wurden jeweils 15 Durchläufe getätigt und der Median ermittelt. Die Ergebnisse zeigt die Abbildung 3.24.

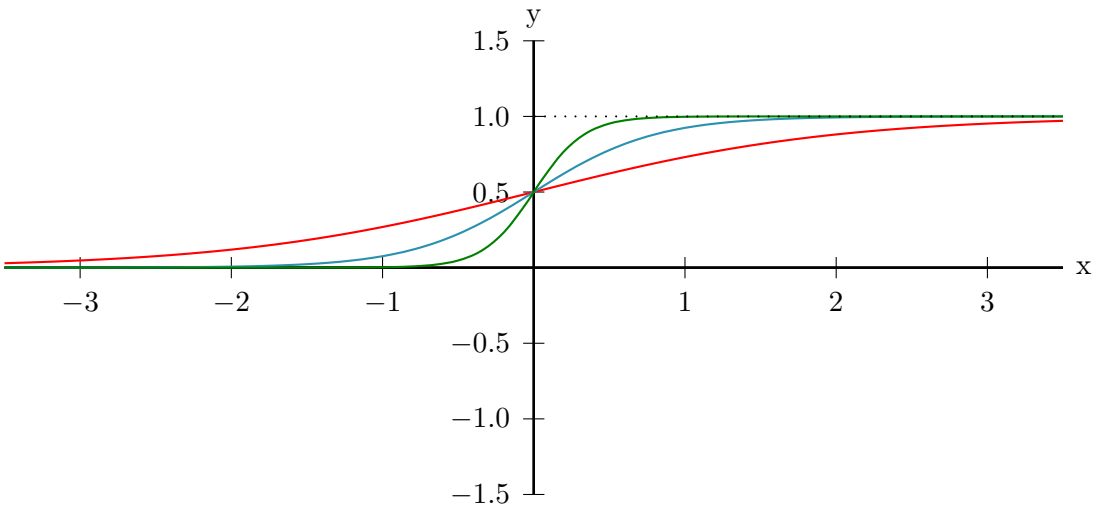


Abbildung 3.23. – Sigmoidfunktion mit verschiedenen Werten für t.

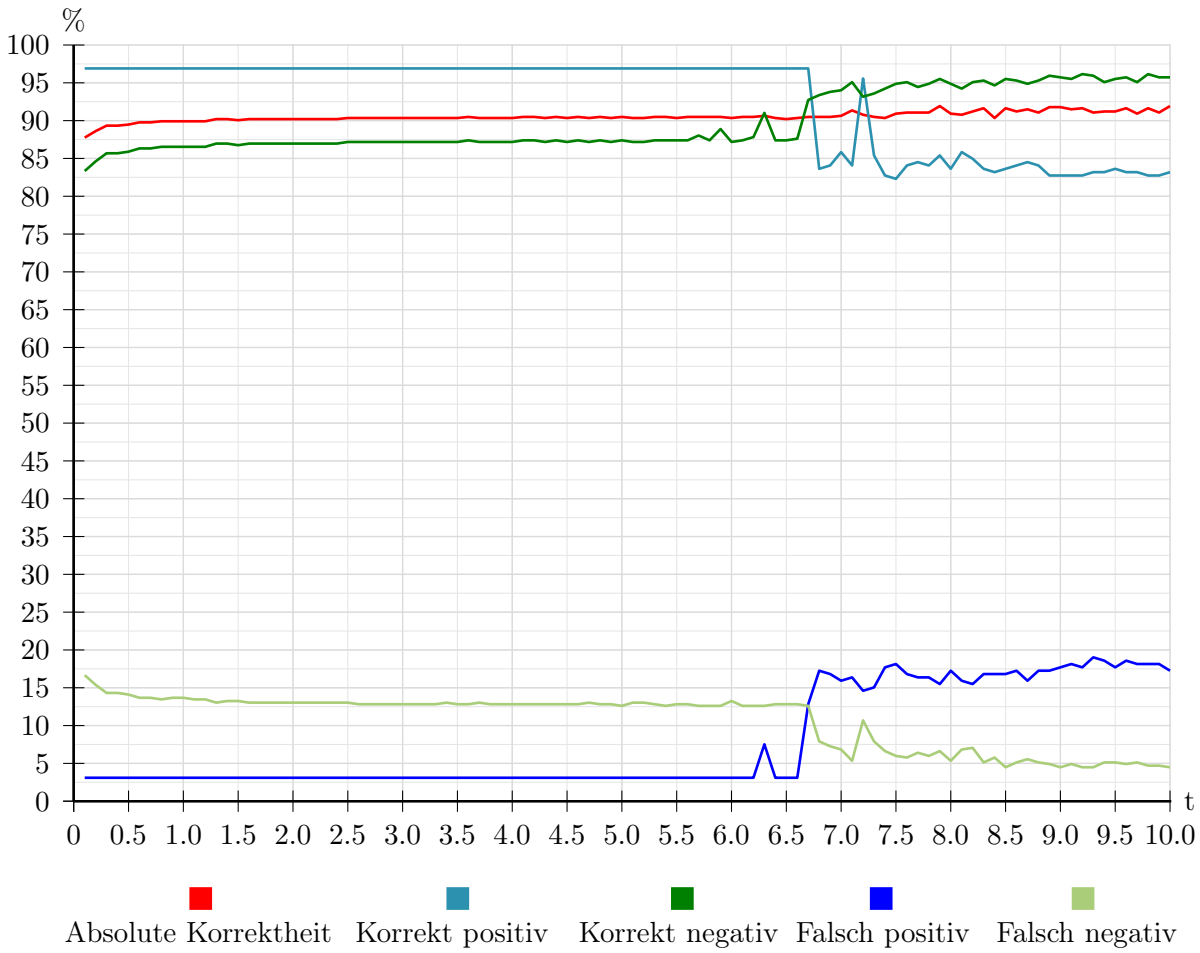


Abbildung 3.24. – Die Abbildung zeigt die Erkennungsraten in Abhängigkeit zur jeweiligen Konstante der Sigmoidfunktion.

Bemerkenswert ist der stete Verlauf bis zu einem Faktor von 6, wohingegen ab diesem Wert keine Linearität mehr vorhanden ist. Dies ist dem Umstand geschuldet, dass die Lernrate ab diesem Wert wiederholt zu groß wurde und dynamisch angepasst werden musste.

Der Graph zeigt eine eindeutige Tendenz hin zur besseren Erkennung von negativen Beispielen, wohingegen positive zunehmend falsch klassifiziert wurden. Auffällig an dieser Stelle ist der Umstand, dass die absolute Korrektheit aller Raten zusammen nur geringfügig beeinflusst worden ist. Optisch scheinen beide Tendenzen das Gleichgewicht zu halten.

Da im Rahmen dieser Arbeit die Erkennungsrate der fälschlicherweise als Roboter klassifizierten Regionen wichtiger ist als die korrekte Erkennung der Negativbeispiele, ist der Bereich $t > 5{,}5$ nicht von Interesse. Eine genauere Betrachtung der leichten Tendenz zur Verbesserung der Negativraten im Bereich von $1 \leq t \leq 3{,}5$ hingegen verspricht gewinnbringendere Ergebnisse. Dazu ist der vielversprechendste Bereich in Tabelle 3.4 dargestellt.[67] [68]

Betrachtet man die Ergebnisse genauer, fällt auf, dass sich die Erkennungsrate im positiven Bereich durch die Konstante nicht ändert. Sie ist nicht beeinflusst durch diese Modifikation der Sigmoidfunktion. Allerdings lässt sich bei der Erkennung der negativen Beispiele eine Verbesserung vermerken, welche die absolute Erkennungsrate um ca. 0,4% steigen lässt. Dies wird ersichtlich anhand der grau unterlegten Spalten.

Die falsch als keine Roboter erkannten Bildregionen sinken prozentual stetig bis im Bereich von $2{,}6 \leq t \leq 3{,}3$ ein Plateau erreicht wird, nach welchem die Erkennungsraten schwanken und keiner eindeutigen Tendenz mehr folgen. Daraus resultiert, dass die optimale Konstante für die Anpassung des Verlaufs der Sigmoidfunktion bei $\lceil \frac{2{,}6+3{,}3}{2} \rceil$ liegt, was **3,0** entspricht.

Dieses Ergebnis verbessert nicht die eigentlich erwünschte Falsch-Positiv-Rate, allerdings ist die Senkung der Falsch-Negativen ein Resultat, das eine zusätzliche Multiplikation rechtfertigt. Hinzu kommt, dass ab einem t von 1,3 die gesamte Erkennungsrate auf über 90% steigt, wodurch eine Anpassung der Konstante ebenfalls sinnvoll ist.[69]

[67] Die Tabelle zeigt gerundete Medianwerte der einzelnen Erkennungsraten, daher kann es sein, dass einzelne Werte sich unterscheiden in der Form, dass sie rechnerisch nicht durch die verbleibenden ermittelt werden können.

[68] Je nachdem, wie das Trainings- und Testset zusammengestellt wurde, unterscheiden sich die Erkennungsraten. Daher sind die hier dargestellten Werte nur als Tendenzen vergleichbar mit denen von Tabelle 3.3, da beide anhand von verschiedenen Setzusammenstellungen entstanden sind.

[69] Zuvor wurde die absolute Erkennungsrate zu Gunsten der Falsch-Positiv-Rate mit Hilfe des Schwellenwertes gesenkt.

t	abs. Korr.	korr. negativ	korr. positiv	falsch positiv	falsch negativ
0,5	89,481%	85,897%	96,902%	3,097%	14,102
0,6	89,769%	86,324%	96,902%	3,097%	13,675
0,7	89,769%	86,324%	96,902%	3,097%	13,675
0,8	89,913%	86,538%	96,902%	3,097%	13,461
0,9	89,913%	86,538%	96,902%	3,097%	13,675
1,0	89,913%	86,538%	96,902%	3,097%	13,675
1,1	89,913%	86,538%	96,902%	3,097%	13,461
1,2	89,913%	86,538%	96,902%	3,097%	13,461
1,3	90,201%	86,965%	96,902%	3,097%	13,034
1,4	90,201%	86,965%	96,902%	3,097%	13,247
1,5	90,057%	86,752%	96,902%	3,097%	13,247
1,6	90,201%	86,965%	96,902%	3,097%	13,034
			keine Änderung		
2,5	90,345%	87,179%	96,902%	3,097%	13,034
2,6	90,345%	87,179%	96,902%	3,097%	12,820
			keine Änderung		
3,3	90,345%	87,179%	96,902%	3,097%	12,820
3,4	90,345%	87,179%	96,902%	3,097%	13,034
3,5	90,345%	87,179%	96,902%	3,097%	12,820
3,6	90,489%	87,393%	96,902%	3,097%	12,820
3,7	90,345%	87,179%	96,902%	3,097%	13,034
3,8	90,345%	87,179%	96,902%	3,097%	12,820
			keine Änderung		
4,6	90,489%	87,393%	96,902%	3,097%	12,820
4,7	90,345%	87,179%	96,902%	3,097%	13,034
4,8	90,489%	87,393%	96,902%	3,097%	12,820
4,9	90,345%	87,179%	96,902%	3,097%	12,820
5,0	90,489%	87,393%	96,902%	3,097%	12,606
5,1	90,345%	87,179%	96,902%	3,097%	13,034
5,2	90,345%	87,179%	96,902%	3,097%	13,034
5,3	90,489%	87,393%	96,902%	3,097%	12,820
5,4	90,489%	87,393%	96,902%	3,097%	12,606
5,5	90,345%	87,393%	96,902%	3,097%	12,820

Tabelle 3.4. – Die Erkennungsraten in Abhängigkeit von der Sigmoid-Konstante.

3.3.2.3. Kostenfunktion

Im vorherigen Kapitel wurde der Vektor Θ eingeführt, der jedes Feature in x individuell wichtet. Die zentrale Fragestellung ist nun, wie die einzelnen Parameter Θ_i gewählt werden.

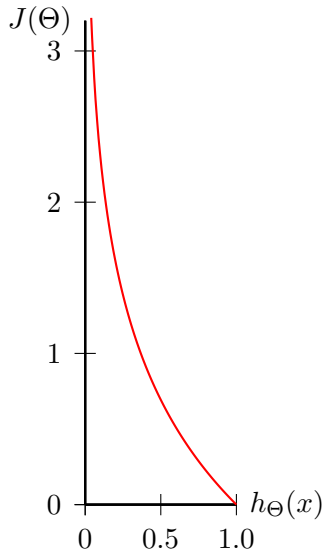


Abbildung 3.25. – Verlauf der Kostenfunktion für $y = 1$.

Die grundlegende Idee besteht darin, alle Werte von Θ so zu wählen, dass $h_\Theta(x)$ möglichst nahe dem y der Trainingsbeispiele *(x,y)* kommt. Zu diesem Zweck wird an dieser Stelle die Fehlerfunktion $J(\Theta)$ eingeführt. Sie gibt an, welche *Kosten* unter Verwendung des aktuellen Vektors Θ entstehen. Dabei werden die individuellen Kosten je Trainingsbeispiel aufsummiert und gemittelt, wie in Formel 3.14 ersichtlich.[70]

$$J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\Theta(x^{(i)}), y^{(i)}) \tag{3.14}$$

Die eigentliche Kostenfunktion ist wie folgt definiert:

$$Cost(h_\Theta(x), y) = \begin{cases} -log(h_\Theta(x)), & \text{if } y = 1 \\ -log(1 - h_\Theta(x)), & \text{if } y = 0 \end{cases} \tag{3.15}$$

Wie anhand von Abbildung 3.25 ersichtlich ist, hat die *Cost*-Funktion einige wünschenswerte Eigenschaften. Wenn $y = 1$ und auch exakt $h_\Theta(x) = 1$ ist, wird $Cost(h_\Theta(x), y) = 0$. Das heißt, dieses Beispiel würde keinerlei Kosten erzeugen. Im umgekehrten Schluss, falls $h_\Theta(x) = 0$ durch den Lernalgorithmus vorhergesagt wird, aber $y = 1$ ist, wird dieser mit sehr hohen Kosten bestraft, da für $h_\Theta(x) \to 0$ die $Cost \to \infty$ geht. Im Falle von $y = 0$ arbeitet die Kostenfunktion genau umgekehrt. Die Kosten betragen 0, falls sowohl y als auch $h_\Theta(x)$ diesen Wert besitzen, und gehen gegen unendlich für $h_\Theta(x) \to 1$.

[70] Vgl. [Ng11], Video 6.4 – „Cost function".

In Ergänzung zu den vorangegangenen Erläuterungen ist zu beachten, dass die Kostenfunktion $Cost(h_\Theta(x), y)$ auch kompakter dargestellt werden kann, da der Wertebereich von y nur 0 und 1 beinhaltet. Die folgende Formel 3.16 verdeutlicht diesen Schritt:

$$Cost(h_\Theta(x), y) = -y \log h_\Theta(x) + (1 - y) \log(1 - h_\Theta(x)) \tag{3.16}$$

Je nachdem ob $y = 1$ oder $y = 0$ ist, wird einer der Blöcke Null. Übrig bleibt der gleiche Term wie in Formel 3.15 gezeigt. Dadurch ändert sich die Fehlerfunktion in folgende Variante:

$$J(\Theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\Theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\Theta(x^{(i)}))] \tag{3.17}$$

3.3.2.4. Gradientenabstiegsverfahren

Da jetzt sowohl die Hypothese $h_\Theta(x)$, als auch die Fehlerfunktion $J(\Theta)$ definiert ist, bleibt die Frage, wie genau der Algorithmus der *Logistischen Regression* lernt. Das Ziel bei diesem Lernvorgang ist eingängig. Es gilt diejenigen Parameter Θ zu finden, welche die Funktion $J(\Theta)$ minimieren, denn je kleiner der Fehler, desto näher ist die Vorhersage am jeweiligen y. Zu diesem Zweck kommt der Algorithmus des *Gradientenabstiegsverfahrens*[71] zum Einsatz.

Die grundlegende Idee des Algorithmus sind die folgenden zwei Phasen, wobei die zweite stetig wiederholt wird:

1. Gestartet wird mit beliebigen Werten für Θ, beispielsweise

$$\Theta = \vec{0} = \begin{pmatrix} 0 \\ 0 \\ ... \end{pmatrix}$$

2. Anschließend werden ständig die Werte des Vektors Θ verändert, um $J(\Theta)$ zu reduzieren, bis man im besten Fall ein Minimum erreicht hat

Dabei arbeitet das *Gradientenabstiegsverfahren* folgendermaßen: Den Ausgangspunkt bildet ein n-dimensionaler Graph beziehungsweise eine Funktion, wobei n die Anzahl der Merkmale darstellt. Beispielsweise ein dreidimensionaler Graph mit zwei Parametern, wobei $\mathbf{x} = \Theta_0$, $\mathbf{y} = \Theta_1$ und $\mathbf{z} = J(\Theta)$ ist.

[71] Auch *Gradient Descent* genannt.

1. Das Verfahren startet an einem beliebigen Punkt innerhalb des Graphen.
2. Anschließend wird die komplette Umgebung in 360° angesehen, die gewünschte Richtung ermittelt und ein kleiner, vordefinierter Schritt in Richtung Minimum durchgeführt.
3. Dies wiederholt sich bis ein *lokales* Minimum gefunden wurde.

Eine zentrale Eigenschaft des *Gradientenabstiegsverfahrens* ist dabei, dass, abhängig vom Startpunkt, der Endpunkt eine andere Position besitzt. Es wird stets ein anderes lokales Optimum gefunden. Daher gibt es keine Garantie das globale Minimum zu finden. Je nach Topologie ist es wahrscheinlicher, dass der Algorithmus in einem lokalen Minimum terminiert.

Der eigentliche Algorithmus für das *Gradientenabstiegsverfahren* lautet wie in 3.12 definiert. Dabei ist α die Lernrate, die beliebig gewählt werden kann. Allerdings muss diese einen Wert aufweisen, welche die Fehlerfunktion verkleinert. Ist α zu groß, so entfernen sich die Parameter Θ vom Minimum und terminieren gegebenenfalls auch nicht, sofern keine zusätzlichen Mechanismen diesen Fehler abfangen. Es empfiehlt sich, die Lernrate dynamisch anzupassen, so dass diese stets einen Wert aufweist, der den Algorithmus effizient dem Minimum zuführt, ohne dabei zu groß oder zu klein zu werden. Ist α dauerhaft zu klein, erfolgt eine Minimierung, aber die Berechnungszeit steigt.

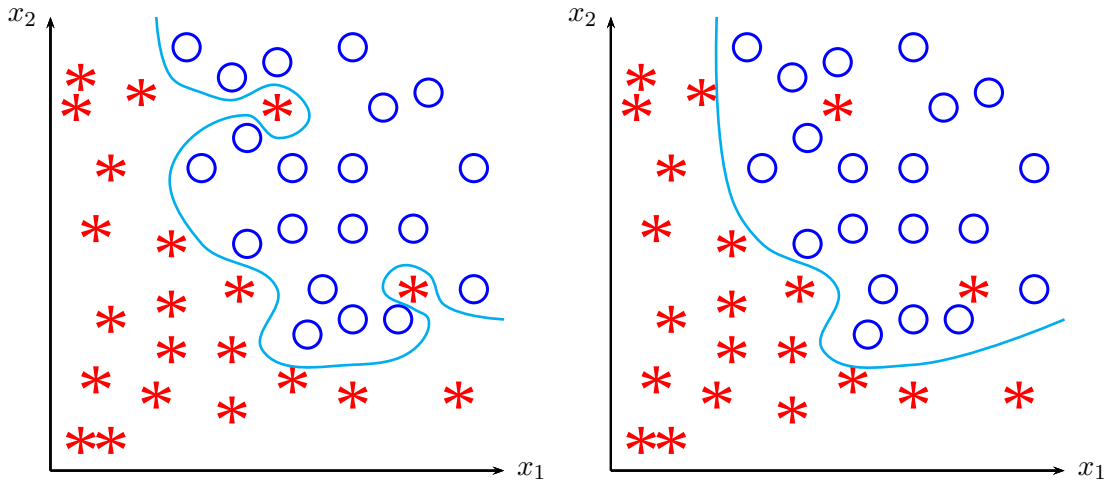
Algorithmus 3.12 Algorithmus des Gradientenabstiegsverfahrens mit Ableitung

repeat
$\Theta_j := \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta)$;
until Konvergenz

Die Aktualisierung aller Werte des Vektors Θ muss dabei simultan erfolgen. In der Umsetzung sieht der Algorithmus wie in Algorithmus 3.13 dargestellt aus, wenn die Ableitung von $J(\Theta)$ ermittelt wurde. Eine Optimierung kann durch eine zufällige Initiierung des Vektors Θ erfolgen, anstatt den Nullvektor $\vec{0}$ zu verwenden. Das führt bei erneutem Durchlauf zu veränderten Ergebnissen und kann den Lernerfolg positiv beeinflussen.

Algorithmus 3.13 Algorithmus des Gradientenabstiegsverfahrens in der Umsetzung

repeat
$\Theta_j := \Theta_j - \alpha \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$;
{simultane Aktualisierung aller Θ_j}
until Konvergenz



(a) Überbestimmte Parameter (b) Ausreichende Bestimmung der Parameter

Abbildung 3.26. – Anpassung der Parameter an das Trainingsbeispielset.

3.3.2.5. Regularisierte Logistische Regression

Werden viele Merkmale verwendet, die auch zusätzlich noch durch Polynome höherer Ordnung kombiniert werden, ist die *Logistische Regression* anfällig zur Überanpassung[72]. Dabei handelt es sich um eine Überbestimmung des gesuchten Parametervektors. Die ermittelten Theta sind zu stark an die Trainingsbeispiele angepasst, übermäßig komplex und können daher bei neuen Eingaben keine akkuraten Vorhersagen treffen. Ein Beispiel dafür ist in Abbildung 3.26 zu finden. Teilbild 3.26 (**a**) zeigt eine klare Überbestimmung der Parameter, die sich durch eine hohe Komplexität im Verlauf auszeichnet, wohingegen (**b**) eine ausreichende Bestimmung ist, welche auch neue Merkmalsvektoren klassifizieren kann.[73]

Man kann dem *Overfitting*-Effekt entgegen wirken, indem die Anzahl der Merkmale, manuell oder automatisiert, reduziert wird. Falls dies nicht gewollt[74] oder auch nicht möglich ist, sollte verhindert werden, dass einzelne Thetas zu groß werden und andere dominieren. Dazu wird die Fehlerfunktion um einen neuen Term erweitert, der diese Aufgabe übernimmt.

[72] Auch als *Overfitting* bezeichnet.
[73] Vgl. [Ng11], Videos 7.1 bis 7.3 zu „Regularization".
[74] Da im Rahmen dieser Arbeit bereits die Anzahl der Features auf die notwendigsten reduziert wurde, ist eine weitere Reduktion nicht möglich.

$$J(\Theta) = -[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\Theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\Theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n} \Theta_j^2 \quad (3.18)$$

Diese Methode ermöglicht es, viele Merkmale zu verwenden, ohne dass die Überbestimmung der Parameter ein ernsthaftes Problem darstellt. Alle Werte Θ_j, bzw. das Ausmaß, welches jedes einzelne am Gesamtergebnis hat, wird reduziert. Je kleiner die Parameter Θ, desto geringer die Neigung zum *Overfitting*. Nachfolgend beschrieben ist der endgültige Algorithmus des Gradientenabstiegsverfahrens mit implementierter Regularisierung.[75]

Algorithmus 3.14 Finaler Algorithmus mit Regularisierung

repeat

$\Theta_0 := \Theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^{(i)}) - y^{(i)})x_0^{(i)}$

$\Theta_j := \Theta_j - \alpha[\frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\Theta_j$

{j = 1, 2, 3, ...,n (nicht Θ_0)}

until Konvergenz

Um den Parameter λ zu bestimmen, wurde neben dem Trainings- und Testset ein drittes Set mit Bildmaterial erstellt, welches der Validierung des Parameters λ dient. Anschließend trainierte der Algorithmus 100-fach mit unterschiedlichen Werten für Lambda. Die Analyse der Ergebnisse ergab, dass eine Regularisierung nicht notwendig ist, da auf Grund der geringen Anzahl an Features keine Tendenz zur Überbestimmung feststellbar und auch keine Verbesserung in den Erkennungsraten ablesbar ist. Daher wird λ mit 0 initialisiert, was keiner Regularisierung entspricht.

3.4. Umsetzung der Klassifizierung

Jeder Roboter soll in der Lage sein, unbekannte Regionen auf einem von ihm selbst erstellten Kamerabild zu klassifizieren. Wie bereits zuvor angedeutet, können die Roboter nicht von sich aus entscheiden, ob auf einem ihrer Kamerabilder ein Mitspieler zu sehen ist oder nicht. Daher ist eine vollständige Umsetzung der Logistischen Regression auf einem Nao nicht sinnvoll.

[75] Für den Algorithmus gibt es zwei Konvergenzkritierien. Er terminiert entweder, wenn eine festdefinierte, maximale Anzahl an Iterationen erreicht worden ist oder wenn der Unterschied zwischen den Fehlerfunktionswerten zweier aufeinanderfolgender Iterationen einen Mindestwert unterschreitet.

Es gilt zwischen dem *Lernvorgang* und der *Klassifizierung* zu unterscheiden. Das Lernen erfolgt anhand eines festen Beispielsets, welches sich nicht auf dem Roboter befinden muss. Es läuft einmalig ab. Sind die erforderlichen Wichtungen der einzelnen Merkmale ermittelt, so können sie zur Klassifikation verwendet werden, ohne dass ein wiederholter Lernvorgang notwendig ist. In der Umsetzung gestaltet sich dies folgendermaßen: An einem Rechner wird ein in Java geschriebenes Programm gestartet, das alle vorhandenen Beispiele einliest, mitgeteilt bekommt, um welche Klasse es sich bei den einzelnen Bildern handelt und anschließend mit einer vorher festgelegten maximalen Anzahl an Iterationen lernt, welche Merkmale wichtig sind für einen Roboter und was einen solchen ausmacht. Das Ergebnis dieser Berechnung ist der Parametervektor Θ. Auf dem Roboter wird im Anschluss daran lediglich der Vektor Θ benötigt. Erhält er ein neues Kamerabild, so werden die einzelnen Regionen ermittelt, alle Merkmale berechnet und anschließend klassifiziert. Dabei sind nur diejenigen Features auf dem Nao umgesetzt, welche sich in Kapitel 3.2.4 als aussagekräftig herausgestellt haben. Die Klassifikation setzt lediglich die Implementierung der Hypothese mit den gelernten Wichtungen voraus, nicht der vollständigen Logistischen Regression. Die Merkmalsvektoren der einzelnen Bildbereiche werden erstellt, der Hypothese übergeben und das Ergebnis in Form von 0 oder 1 ausgegeben.

Es ist festzuhalten, dass der rechenintensivste Anteil unabhängig vom Roboter einmalig auf einem beliebigen Computer stattfindet. Auf dem Roboter finden hingegen nur die notwendigen Berechnungen statt, wie Merkmalsermittlung und Klassifikation. Damit ist das Ziel, eine performante Erkennung von Naos auf einem Roboter zu implementieren, erreicht. Der ressourcenintensivste Anteil auf dem Nao ist dabei die Sigmoidfunktion. Eine Approximation wird diese ablösen, damit weniger Rechenoperationen benötigt werden.

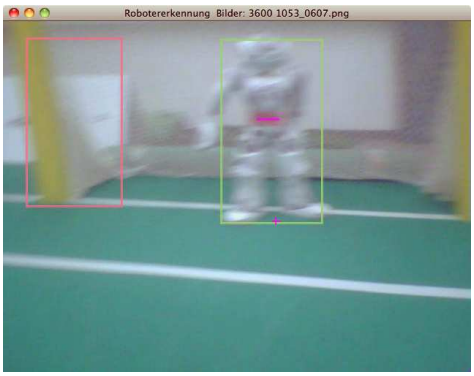


Abbildung 3.27. – Die Darstellung zeigt zwei unbekannte Regionen im Bild, die durch den Algorithmus klassifiziert wurden. Die mit rot gekennzeichnete Region ist korrekt als kein Roboter erkannt und grün ebenfalls korrekt als Roboter bestimmt. Weiterhin sind die aus Kapitel 3.1.2 bekannten Tag-Informationen visualisiert, um die Genauigkeit der Erkennung zu zeigen.

4. Ergebnisse und Ausblick

Ziel dieser Arbeit war es, einen Roboter zu befähigen, während eines Spiels auf den selbst erstellten Bildern unbekannte Regionen zu klassifizieren. Diese Erkennung soll echtzeitfähig und kalibrierungsfrei ablaufen, das heißt, es darf zu keinen Verzögerungen durch die Berechnung der Merkmale oder Erstellung einer Hypothese kommen, sowie keine Ermittlung von Werten vor jedem Spiel notwendig sein. Dieses Ziel wurde erreicht.

4.1. Ergebnisse

Neben den in Kapitel 3.2.4 aufgelisteten Merkmalen kommen für den endgültigen Algorithmus drei weitere zum Einsatz – die Breite und Höhe einer Region sowie Breite × Höhe als quadratisches Merkmal. Diese einfachen Merkmale, deren Erstellung keinerlei weitere Berechnungen erfordert, haben die korrekte Erkennung von Regionen im Mittel um 1% gesteigert.[76]

4.1.1. Endgültige Wichtungen

Wie in Kapitel 3.3.2 zur Arbeitsweise der Logistischen Regression beschrieben ist, wird jedes verwendete Merkmal individuell gewichtet. Wie diese Wichtung ausfällt, ist abhängig von der jeweiligen Aussagekraft. Je besser ein Merkmal die beiden Klassen voneinander trennt, desto stärker wird es gewichtet. Dabei können auch negative Werte ermittelt werden, wobei das Vorzeichen kein Indiz für ein starkes oder schwaches Merkmal ist. Einzig die absolute Höhe der Wichtung entscheidet darüber. Negative Werte balancieren das Ergebnis aus.

In Abbildung 4.1 sind die einzelnen Wichtungen in Form eines Balkendiagramms dargestellt. Es zeigt, wie unterschiedlich wichtig die einzelnen Merkmale für die Erstellung der Hypothese sind. Während beispielsweise **(f)** nur wenig Einfluss hat, ist eine gute Erkennung ohne das Merkmal **(b)** nicht wahrscheinlich. Welches Merkmal der jeweilige Buchstabe repräsentiert, ist der nachfolgenden Auflistung zu entnehmen.

[76] Basis des Vergleichs: 2×30 Testseterstellungen bei einer mittlere quadratische Schwankung 1,04 (mit den neuen Merkmalen) und 0,97 (ohne die drei Merkmale).

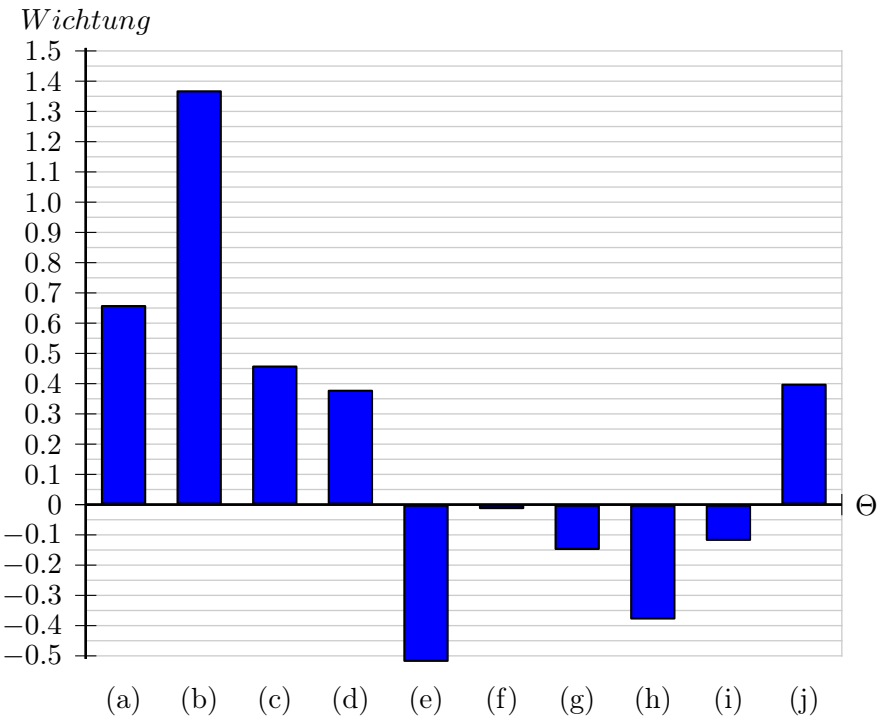


Abbildung 4.1. – Wichtungen der finalen Merkmale.

(a) *0.66* – 1. Ordnung – die Entropie der partiellen Ableitung nach Y

(b) *1.37* – 1. Ordnung – die Entropie der partiellen Ableitung nach Y mit Vorzeichen

(c) *0.46* – 1. Ordnung – die Entropie der partiellen Ableitung nach X mit Vorzeichen minus des gleichen Features nach Y

(d) *0.38* – 1. Ordnung – die Entropie des Helligkeitskanals

(e) *-0.52* – 1. Ordnung – die mittlere quadratische Abweichung des Helligkeitskanals

(f) *-0.015* – 2. Ordnung – die Entropie der Liniensummen der Gradienten nach X multipliziert mit dem gleichen Feature unter Vorzeichenerhaltung

(g) *-0.15* – 2. Ordnung – die Entropie der Liniensummen der Gradienten nach Y unter Vorzeichenerhaltung multipliziert mit sich selbst

(h) *-0.38* – 1. Ordnung – die Breite einer Bildregion

(i) *-0.12* – 1. Ordnung – die Höhe einer Bildregion

(j) *0.4* – 2. Ordnung – Kombination von Breite und Höhe

	Median	Mittelwert	Min.	Max.	Schwankung
Korrekt insgesamt	90,4899	90,4205	87,8962	92,7953	1,2075
Nao korrekt erkannt	86,9369	86,8975	83,4394	90,2597	2,3355
korrekt als keiner erkannt	97,3568	97,2825	95,1807	99,1304	0,7532
falsch als Nao erkannt	2,6548	2,7174	0,8695	4,8387	0,7539
falsch als keiner erkannt	3,0718	13,1024	9,7505	16,5957	2,3361

Tabelle 4.1. – Die finalen Erkennungsraten für 1.500 Testdurchläufe.

4.1.2. Finale Erkennungsraten

Um die endgültigen Erkennungsraten zu ermitteln, wurde das Test- und Trainingsset 1500 Mal neu erstellt und die Ergebnisse ausgewertet. Dabei wurden Ausreißer nicht betrachtet. In Tabelle 4.1 sind die gerundeten Resultate aufgelistet. Sie zeigt neben dem Median und Mittelwert auch die Minima und Maxima, sowie die mittlere quadratische Schwankung der einzelnen Erkennungsraten.

Wie anhand der einzelnen Werte sichtbar ist, sind die Ergebnisse der Erkennung für den praktischen Einsatz geeignet und übertreffen die Erwartungen, die vor Beginn der Implementierung an sie gestellt wurden. Mit einem Mittelwert von 2,71 % befindet sich die für diese Arbeit wichtigste Rate der falsch als Roboter erkannten Regionen unterhalb der anfänglichen > 4 %. Lediglich die maximalen Ausreißer spiegeln diese Anfangswerte wider. Daran zeigt sich, dass die Anpassungen des Algorithmus erfolgreich arbeiten. Eine mittlere quadratische Schwankung von nur 0,75 % macht deutlich, dass das Resultat des Klassifikators stabile Ergebnisse gewährleistet. Auch das Maximum von 4,8 % Fehlerkennungen bleibt unter der Negativmarke von 5 %.

Somit werden im ungünstigsten Fall 48 von 1000 ermittelten Bildregionen falsch als Roboter erkannt, im besten Fall kommen 8 auf 1000 – Werte, die innerhalb eines Spieles tolerierbar sind. Zudem lassen sich diese Ausreißer bei der Betrachtung von mehreren Frames in Folge sowie durch Informationen der Mitspieler ausschließen und wirken sich somit nur minimal negativ auf das Spiel aus.

Die vergleichsweise hohe mittlere quadratische Schwankung von 2,3 % bei korrekt erkannten Naos sowie bei Bildregionen, wo fälschlicherweise kein Roboter erkannt wurde, obwohl ein solcher vorhanden war, lässt sich teilweise durch nicht optimal ausgewähltes Bildmaterial erklären. Diese Ursache und eine mögliche Lösung wird in Kapitel 4.1.3 beschrieben.

Das zweite Bewertungskriterium neben der Falsch-Positiv-Rate ist die Korrektheit aller Klassifizierungen. Sowohl Mittelwert als auch Median sind ca. 0,5 % über der selbstgesetzten Grenze von 90 %, wobei das Minimum aller Tests nur ca. 2 % unterhalb dieser

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

Abbildung 4.2. – Diese Abbildung zeigt Regionen, die fälschlicherweise als Roboter erkannt werden.

liegt. Eine mittlere quadratische Schwankung von 1,2 % zeigt, dass die Robotererkennung verlässliche Resultate liefert und für den produktiven Einsatz geeignet ist.

4.1.3. Problematiken

Ausgehend von den finalen Erkennungsraten arbeitet der Algorithmus sehr gut, aber nicht perfekt. Es gibt Ausreißer, die falsch klassifiziert werden und so für ungewollte Fehlinformationen sorgen.

In Abbildung 4.2 sind Bilder gezeigt, die fälschlicherweise als Nao erkannt worden sind. Von diesen gibt es nicht viele[77], und wie anhand der Beispielbilder ersichtlich, auch nur unter den folgenden Voraussetzungen. Eine Ursache der Fehlerkennung ist die Kombination aus Roboterarm und Ball, wie in Teilbild 4.2 **(b)** dargestellt. Da der Arm ähnliche Merkmale besitzt wie der Rest des Körpers und der Ball je nach Lichtsituation mit einem rötlichen Hüftband vergleichbar ist, kommt dieser Irrtum in seltenen Fällen vor.[78]

Des Weiteren ist die generell schlechte Qualität der Kamera eine Ursache. Teilbild **(a)** zeigt einen Fall, in dem ein Fuß für einen ganzen Roboter gehalten wird. Allerdings

[77] In dem für diese Auflistung verwendeten Testset genau fünf von ca. 700 Testbildern – die 5, welche in Abbildung 4.2 dargestellt werden.
[78] Das verwendete Testset enthielt diese Fehlerkennung nur einmal, siehe dazu Fußnote 77.

stellt sich hier die Frage, ob es sich dabei zwangsläufig um einen Fehler handelt, da ein Fuß die Anwesenheit eines Roboters impliziert. Während eines Spiels wirkt sich diese Fehlerkennung nicht negativ aus, daher handelt es sich um eine Definitionsfrage, ab wann ein Roboter ein solcher ist. Im Rahmen dieser Arbeit gilt die Festlegung, dass ein Roboter mindestens über ein erkennbares Hüftband verfügen muss, um als Nao erkannt zu werden.

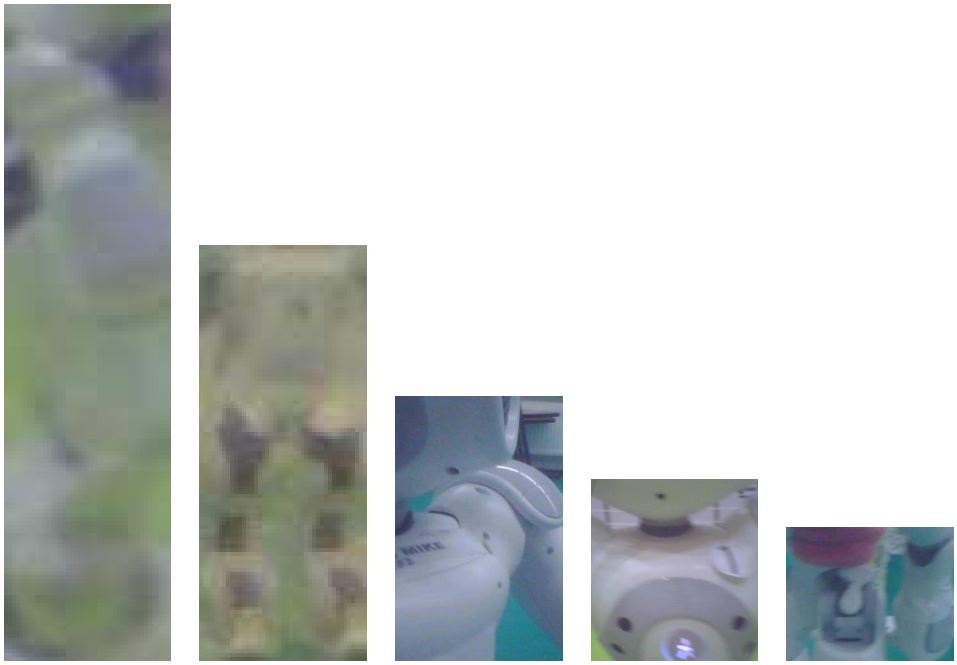
Die dritte und häufigste Ursache ist das Tornetz. Abhängig von dem jeweiligen Hintergrund besitzt es ebenfalls viele Kanten und Helligkeitsschwankungen, so dass der Unterschied zu einem Roboter minimal ist. Abhilfe können hier komplexere Merkmale schaffen, welche die Topologie einer Region mit einbeziehen. Allerdings gilt es hier abzuwägen, ob der höhere Rechenaufwand die bessere Erkennung rechtfertigt. Effizienter ist es, Informationen aus Tornähe weniger Glaubwürdigkeit zu geben und mehrfach zu überprüfen. Da derartige Ausreißer in der Regel nur einzelne Frames betreffen, lassen sie sich interframebasiert leicht ausschließen, indem Roboter in Tornähe erst als erkannt gelten, wenn sie in einer Mindestanzahl an Bildern gleichermaßen erkannt wurden.[79]

Abbildung 4.3 zeigt eine Auswahl an Bildern, welche als Roboter erkannt wurden, obwohl sie laut Vorgabe keinen darstellen. Allerdings ist diese Aussage nur bedingt korrekt, da in allen Fällen Teile von Robotern gezeigt werden. Bereiche, die keine Informationen über die Position des Roboters liefern, sollen nicht als ein solcher erkannt werden. Dazu zählen beispielsweise die Arme, der Torso und der Kopf.

Die hohe Rate an Fehlerkennungen hat eine zentrale Ursache. Da bei der Testseterstellung nicht jedes einzelne Bild gesichtet und bewertet wurde, kommt es vereinzelt vor, dass Bilder enthalten sind, die eigentlich korrekt klassifiziert wurden. Die Teilabbildungen 4.3 **(b)**, **(c)** und **(d)** zeigen diesen Fall. Bei **(b)** ist kein Hüftband vorhanden. Da dieser Fall in einem Spiel nicht vorkommt, kann die Fehlerkennung ignoriert werden. Mit den anderen beiden Teilbildern verhält es sich ähnlich, da sie lediglich den Torso eines Nao zeigen und somit auch nicht als Roboter gelten Die Abbildungen 4.3 **(a)** und **(e)** sind korrekte Fehlklassifizierungen, deren Ursache noch nicht bekannt ist und weitere Tests erfordert. Da das verwendete Bildmaterial aufgrund des ausgeprägteren Situationenspektrums und auch der größeren Anzahl an Material von der Vorgängergeneration der aktuellen Naos stammt,[80] wird die Ursache der nicht optimal ausgewählten Testbilder in Zukunft nicht weiter bestehen. Für die moderneren Kameras muss eine neue Testdatenbank erstellt werden, wobei von Anfang an auf diesen Umstand acht gegeben wird und das Bildmaterial als Ursache minimiert werden kann.

[79] Unter anderem besteht das Testset aus einer Bildreihe, bei der ein Lauf auf das Tor durch den Roboter aufgezeichnet wurde. Da der Fehler hierbei nicht vermehrt auftrat, soll das Tornetzproblem als ein Einzelfallproblem gelten.

[80] Siehe hierzu Fußnote 26 auf Seite 18.



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

Abbildung 4.3. – Diese Abbildung zeigt Regionen, die falsch als keine Roboter erkannt wurden.

4.2. Ausblick

Das grundlegende Ziel dieser Arbeit – das Erkennen von Robotern auf Kamerabildern – ist erreicht worden. Allerdings ist an dieser Stelle das Projekt selbst noch nicht beendet. Es sind Verbesserungen und Ausbaustufen möglich, die allerdings den Rahmen dieser Ausarbeitung überschreiten würden.

4.2.1. Verbesserung der Erkennung

Ein Schritt, an dem angesetzt werden könnte, ist die Verbesserung der Erkennungsraten. Allerdings sollte mit in Betracht gezogen werden, dass die Klassifikation bereits sehr gut funktioniert und eine Änderung je nach Art der Umsetzung auch mehr Rechenoperationen voraussetzt. Ob dieser Mehraufwand gerechtfertigt ist, muss im Einzelfall entschieden werden.

Der erste Ansatz ist das Ersetzen des Klassifikationsalgorithmus. Wie bereits in Kapitel 3.3.1 erwähnt, ist die Logistische Regression ein sehr einfacher Ansatz und kann durch andere Verfahren abgelöst werden. Hierbei sind insbesondere Neuronale Netze interessant. In einem an das menschliche Gehirn angelehnten Modell werden schichtenbasiert Neu-

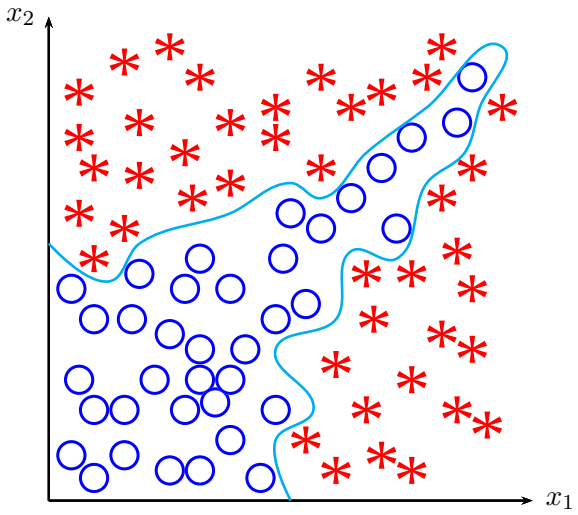


Abbildung 4.4. – Nichtlineare Klassifikation.

ronen aktiviert, die lernen, ab welcher Reizintensität eine Reaktion erfolgen muss. Der Vorteil besteht unter anderem darin, dass im Gegensatz zur Logistischen Regression eine nichtlineare Klassifikation möglich ist. Ein Beispiel hierzu ist in Abbildung 4.4 dargestellt. Ob diese Herangehensweise die Erkennung maßgeblich verbessert, müssen konkrete Messungen zeigen. Insbesondere ist hierbei ein Vergleich in der Laufzeit interessant, da der bisherige Algorithmus bereits sehr gute Ergebnisse liefert und die Dauer der Rechenoperationen dementsprechend von größerer Bedeutung ist.

Eine weitere Möglichkeit der Verbesserung bietet sich bei der Sigmoidfunktion an. Die Berechnung der natürlichen Exponentialfunktion erfordert bei der Erstellung der Hypothese viel Rechenzeit, welche nur begrenzt verfügbar ist. Um der Anforderung der Echtzeitfähigkeit gerecht zu werden, bietet sich an dieser Stelle eine Approximation der Funktion an. Sofern die Genauigkeit dieser ausreicht und die Erkennungsraten nicht negativ beeinflusst werden, rechtfertig der zu erwartende Geschwindigkeitsvorteil eine Implementierung. Auch hier können erst genaue Messungen zeigen, wie erfolgreich dieser Ansatz ist.

Zudem lässt sich die Erkennung durch weitere, aussagekräftige Merkmale erhöhen. Sie bieten eine Möglichkeit, die Klassifikationsergebnisse stetig zu verbessern, da mit jeder neuen und implementierten Idee die einzelnen Raten beeinflusst werden können. Denkbar sind hier komplexere Merkmale, welche auch über die Topologie einer Bildregion eine Aussage treffen können.

4.2.2. Anwendung der Robotererkennung

Die Umsetzung der Robotererkennung war im Hinblick auf die spätere Nutzung dieser Informationen wünschenswert. Der Roboterfußball ist ein sich sehr schnell weiterentwickelndes Themengebiet. Während bisher die Effizienz der Tor-, Ball- und Linienerkennung vorrangig behandelt werden musste, ist das Nao-Team der HTWK Leipzig 2012 an einem Punkt, wo die Teamstrategie den größten Fokus einnimmt, um auch in Zukunft konkurrenzfähig zu sein.

Hinzu kommt, wie in Kapitel 2.2 bereits erläutert, die Regeländerung von 2012, dass beide Tore gleichfarbig gelb sein sollen. Aufgrund dessen ist eine optische Unterscheidung der Spielhälften nicht wie zuvor möglich, so dass neue Wege zur Orientierung auf dem Spielfeld notwendig sind. Auch hier bietet die Erkennung von Robotern einen Lösungsansatz.

4.2.2.1. Gegnererkennung

Die wichtigste Weiterentwicklung des im Rahmen dieser Arbeit entwickelten Algorithmus ist die Unterscheidung von erkannten Robotern in Mitspieler und Gegner. Derart wertvolle Informationen können anschließend in der Teamstrategie und der Lokalisierung von Vorteil sein.

Eine konkrete Idee, wie diese Ausbaustufe implementiert werden kann, ist bereits in Arbeit. Ein zweiter Klassifikationsalgorithmus soll lernen, bereits erkannte Roboterregionen bezüglich der Teamfarbe zu unterscheiden. Er wird in der Aufrufreihenfolge nach der Robotererkennung gestartet und erhält lediglich Bildbereiche als Eingabe, die zuvor als Nao erkannt worden sind.

Um den zusätzlichen Rechenaufwand gering zu halten, werden vorrangig Merkmale auf ihre Aussagekraft getestet, die bereits in der Erkennung der Roboter zum Einsatz kamen. Da im Rahmen dieser Arbeit die auf Farbkanälen basierenden Features größtenteils ausgeschlossen wurden, eine Teamerkennung aber Farbmerkmale benötigt, ist eine Wiederverwendung berechneter Zwischenergebnisse unwahrscheinlich.

Eine Verbesserungsmöglichkeit ist, im Zuge der Bildsegmentierung mögliche Hüftbänder in den als Roboter erkannten Regionen zu ermitteln und die Klassifikation auf derartige Bereiche zu beschränken. Dabei bietet sich die Verwendung eines weiteren Klassifikators an – einer zur Erkennung von Hüftbändern und ein zweiter für die Teamfarbe. Ob dieser Ansatz sinnvoll ist, kann erst mit der Implementierung festgestellt werden.

4.2.2.2. Lokalisierung

Falls ein Roboter die Orientierung verliert und nicht feststellen kann, in welcher Spielhälfte er sich befindet, können Informationen der Robotererkennung die Lokalisierung unterstützen. Da das Spielfeld symmetrisch aufgebaut ist, gibt es nur eine begrenzte Anzahl an Standorten, wo sich der Nao befinden kann.[81]

Da Informationen über die eigene Position den Mitspielern bereits jetzt und Positionen erkannter Roboter in Zukunft innerhalb des Teams ausgetauscht werden sollen, können diese zur Lokalisierung genutzt werden. Ein Nao ohne Orientierung kann diese Informationen auswerten und mit dem vergleichen, was er über seine Kamera wahrnimmt. So wird es ihm ermöglicht, Hypothesen über die eigene Position zu validieren und Positionen auszuschließen, die den aktuell wahrgenommenen Bildern widersprechen.

Dies erhöht die Genauigkeit der Lokalisierung und stellt sicher, dass ein Roboter nicht in die falsche Spielhälfte spielt. Sobald eine Teamerkennung implementiert ist, wird es ebenso möglich sein, den eigenen Torwart zur Orientierung zu nutzen. Ein Nao könnte seine Position validieren, indem er in diejenige Richtung blickt, in der er den eigenen Torwart erwartet.

4.2.2.3. Teamstrategie

Die meisten Anwendungsmöglichkeiten der Robotererkennung sind in der Teamstrategie zu finden. Beispielsweise kann eine Verfolgung der Bewegungsrichtung von Gegnern einen Hinweis auf die aktuelle Ballposition geben, Spieler bewusst umgangen oder auch von größerer Distanz auf das Tor geschossen werden, falls ersichtlich ist, dass keine Roboter im Weg sind oder der Torwart sich nicht in der Nähe befindet.

Mit der steten Weiterentwicklung der Software wird es zunehmend schwerer, direkt auf das Tor zuzulaufen oder den immer schneller werdenden Robotern zu entgehen. Daher steht auf der mittelfristigen Agenda des Nao-Teams der HTWK das Passspiel. So können Hindernisse gezielt umspielt oder auch geeignetere Winkel zum Tor gefunden werden. Hierfür ist die Robotererkennung eine wichtige Vorraussetzung, um einen gezielten Pass an einen Mitspieler ausführen zu können.

Auch für komplexere Teamstrategien ist die Klassifizierung unersetzlich. Entscheidungen können auf Grund von Gegnerbewegungen getätigt, Informationen über das Spielgeschehen validiert und Rollen innerhalb des Teams effektiver verteilt werden. Rollen wie die eines Beobachters sind möglich, welcher neben dem Torwart das Feld überwacht und

[81] Die Erfahrung der vergangenen Jahre hat gezeigt, dass ein Roboter in der Regel mindestens eine Linie und die Spielfeldgrenze sieht. Durch Drehung des Kopfes werden weitere Linien sichtbar, welche bei der Orientierung helfen und die Anzahl der Möglichkeiten stark begrenzen.

das aktuelle Geschehen den Mitspielern zeitnah mitteilen kann. Beispielsweise könnten so zeitraubende Aktionen wie die Suche nach dem Ball entfallen, da dieser stets von einem der Beobachter zusammen mit allen sichtbaren Robotern im Blick gehalten wird. Der nächste Mitspieler ist anschließend in der Lage, ohne die Position des Balles validieren zu müssen, einen Pfad zum Ball zu berechnen, auch wenn dieser von mehreren Gegnern verdeckt wird. Zudem sind derartige Informationen hilfreich, um Bewegungspfade in Echtzeit an auftauchende Hindernisse anzupassen, da Roboterpositionen und Bewegungsrichtungen bekannt sind.

In Zukunft wird die Teamstrategie der Fokus des Nao-Teams der HTWK sein – unter Verwendung der dafür essenziellen Robotererkennung sowie deren Ausbaustufen.[82]

[82] Festlegung im Zuge des Teamtreffens vom 20.07.2012.

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Quelltexte

Literaturverzeichnis

[Aha11] Ahad, Md. Atiqur Rahman: *Computer Vision and Action Recognition.* Atlantis Ambient and Pervasive Intelligence. Atlantis Press, 2011.

[AN04] Alfred Nischwitz, Peter Haberäcker: *Masterkurs Computergrafik und Bildverarbeitung.* Vieweg & Sohn Verlag, 2004.

[Bri] Brinkmann, Rudolf: *Formelsammlung zur beschreibenden Statistik.* `http://www.brinkmann-du.de/mathe/gost/bstat_formel_01.htm` (Letzter Zugriff: 03.2012).

[But06] Butzhammer, Sabine: *ABC der Mediengestaltung.* Verlag Beruf und Schule, 5 Auflage, 2006.

[Fed12] Federation, The Robocup: *RoboCup.* `http://www.robocup.org/about-robocup/` (Letzter Zugriff: 03.2012), 04 2012.

[GO07] Goro Obinata, Ashish Dutta: *Vision Systems - Segmentation and Pattern Recognition.* I-Tech Education and Publishing, 2007.

[HM09] Heinrich Mellmann, Prof. Dr. Hans-Dieter Burkhard: *Navigation humanoider Roboter.* `http://www2.informatik.hu-berlin.de/cv/isprs/WG/cfp/2009%20WS%20OptecBB/cont/2009-11-13-Navigation-humanoider-Roboter.pdf` (Letzter Zugriff: 04.2012), 11 2009.

[Jäh93] Jähne, Bernd: *Digitale Bildverarbeitung.* Springer-Verlag Berlin Heidelberg, 3. Auflage, 1993.

[JB06] Joachim Böhringer, Peter Bühler, Patrick Schlaich: *Kompendium der Mediengestaltung für Digital- und Printmedien.* Springer-Verlag Berlin Heidelberg, 2 Auflage, 2006.

[Kri] Kriesel, David: *Ein kleiner Überblick über Neuronale Netze.* `http://www.dkriesel.com/science/neural_networks` (Letzter Zugriff: 04.2012).

[Kur11] Kurniawan, Jimmy: *Multi-modal Machine-learned Robot Detection for RoboCup SPL*. School of Computer Science & Engineering, University of New South Wales, August 2011.

[Ng11] Ng, Andrew: *Stanford Machine Learning Online Course*. Online Course, 2011.

[RC10] Roberto Cipolla, Sebastiano Battiato, Giovanni Maria Farinella: *Computer Vision - Detection, Recognition and Reconstruction*, Band 285 der Reihe *Studies in Computational Intelligence*. Springer-Verlag Berlin Heidelberg, 2010.

[Rei11] Reinhardt, B. Sc. Thomas: *Kalibrierungsfreie Bildverarbeitungsalgorithmen zur echtzeitfähigen Objekterkennung im Roboterfußball*. Masterarbeit. Oktober 2011.

[Rob11] RoboCup, Technical Committee: *RoboCup Standard Platform League (Nao) Rule Book*. `www.tzi.de/spl/pub/Website/Downloads/Rules2011.pdf` (Letzter Zugriff: 05.2012), Mai 2011.

[Sze08] Szeliski, Richard: *Computer Vision: Algorithms and Applications*. Springer, 2008.

[Sze11] Szeliski, Richard: *Computer Vision - Algorithms and Applications*. Springer-Verlag London Limited, 2011.

[Ude10] Ude, Aleš: *Robot Vision*. In-Teh, 2010.

A. Anhang

A.1. Symbolverzeichnis

$\mathbf{a}(\mathbf{g})$ Häufigkeit des Auftretens des Tonwertes g

$\mathbf{a}(\mathbf{i})$ Häufigkeit des Auftretens der partiellen Ableitung $\mathbf{i}$ in einer Bildregion

$\alpha_{\mathbf{S}}$ Der Anisotropiekoeffizient eines Kanals

$\mathbf{b}$ Vektor mit $\mathbf{m}$ Werten einer Merkmalskategorie

$\mathbf{b_{normalized}}$ normalisierter Vektor $\mathbf{b}$

$\mathbf{C}$ Wert einer Vektorlängenberechnung

$\mathbf{d_x}$, $\mathbf{d_y}$ Partielle Ableitungen nach x und nach y

$\mathbf{d_{min}}$, $\mathbf{d_{max}}$ Minimale und maximale partielle Ableitung einer Bildregion

$\mathbf{d_{min_x}}$, $\mathbf{d_{max_x}}$ Minimale und maximale partielle Ableitung nach $\mathbf{x}$ einer Bildregion

$\mathbf{d_{min_y}}$, $\mathbf{d_{max_y}}$ Minimale und maximale partielle Ableitung nach $\mathbf{y}$ einer Bildregion

$\mathbf{D_x}$, $\mathbf{D_y}$ Menge aller partiellen Ableitung nach x und nach y

$\mathbf{G}$ Grauwertmenge von 0 bis 255

$\mathbf{g}$ Grauwert aus Grauwertmenge $\mathbf{G}$

$\mathbf{g}(z)$ Sigmoidfunktion von z

$\mathbf{h_\Theta}(\mathbf{x})$ Hypothese des Merkmalsvektors $\mathbf{x}$

$\mathbf{H_S}$ Entropie eines Kanals

$\mathbf{H_{D_x}}$, $\mathbf{H_{D_y}}$ Entropie der partiellen Ableitungen nach $\mathbf{x}$ und nach $\mathbf{y}$

$\mathbf{J}(\mathbf{\Theta})$ Fehlerfunktion der Logistischen Regression

$\mathbf{K}$ Alle Kanäle eines Bildes

$\mathbf{L}$ Zeilen in einem Kanal eines Bildes

$\mathbf{M} = \mathbf{L} * \mathbf{R}$ Anzahl der Bildpunkte von $\mathbf{S}$

$\mathbf{m_S}$ Mittelwert von Kanal $\mathbf{S}$ (mittlerer Grauwert)

m Anzahl an Traingsbeispielen

n Anzahl an Merkmalen

P Wahrscheinlichkeit

$\mathbf{p_S}(\mathbf{g})$ Histogramm der relativen Häufigkeiten eines Grauwertes $\mathbf{g}$ des Kanals bzw. der Pixelmenge $\mathbf{S}$

$\mathbf{p_Y}(\mathbf{i})$ Histogramm der relativen Häufigkeit der partiellen Ableitungen

$\mathbf{p_{Y_x}}(\mathbf{i})$, $\mathbf{p_{Y_y}}(\mathbf{i})$ Histogramm der relativen Häufigkeit der partiellen Ableitungen nach $\mathbf{x}$ und nach $\mathbf{y}$ auf Basis des $\mathbf{Y}$-Kanals (der Helligkeitskanal im *YCbCr*-Farbraum)

$\mathbf{q_S}$ Mittlere quadratische Abweichung von Kanal $\mathbf{S}$

$\mathbf{R}$ Spalten in einem Kanal eines Bildes

$\mathbf{s}(\mathbf{x}, \mathbf{y})$ ein Pixel in einem Bild mit den zweidimensionalen Koordinaten $\mathbf{x}$ und $\mathbf{y}$

$\mathbf{S} = \{\mathbf{s}(\mathbf{x}, \mathbf{y})\}$ ein Kanal in einem 3-kanaligen Bild mit $\mathbf{L}$ Zeilen und $\mathbf{R}$ Spalten, entspricht einer Menge von Pixeln

$\mathbf{s}$ Standardabweichung

$\mathbf{t}$ Konstante zur Verlaufsbeeinflussung der Sigmoidfunktion

$\mathbf{V_{n,m}}$ Matrix aller Merkmale der einzelnen Beispiele mit $\mathbf{n}$ Spalten und $\mathbf{m}$ Zeilen

x Inputvektor oder auch Merkmalsvektor mit n Merkmalswerten

$\mathsf{(x,y)}$ ein Trainingsbeispiel bestehend aus Merkmalsvektor und Zielvariable

$\mathsf{(x^{(i)},y^{(i)})}$ $\mathrm{i_{tes}}$ Trainingsbeispiel

y Output- oder auch Zielvariable, welche versucht werden vorherzusagen

$\mathbf{\Theta}$ Parametervektor oder auch Wichtungsvektor

A.2. Algorithmen

A.2.1. Definition der Funktion *size*()

Algorithmus A.1 Die *size*()-Funktion

Input: {Liste mit Elementen}

1. size := 0
2. **for all** $e \in$ {Liste mit Elementen} **do**
3. size += 1
4. **end for**
5. **return** size

A.3. Abbildungen

A.3.1. Merkmalskategorien

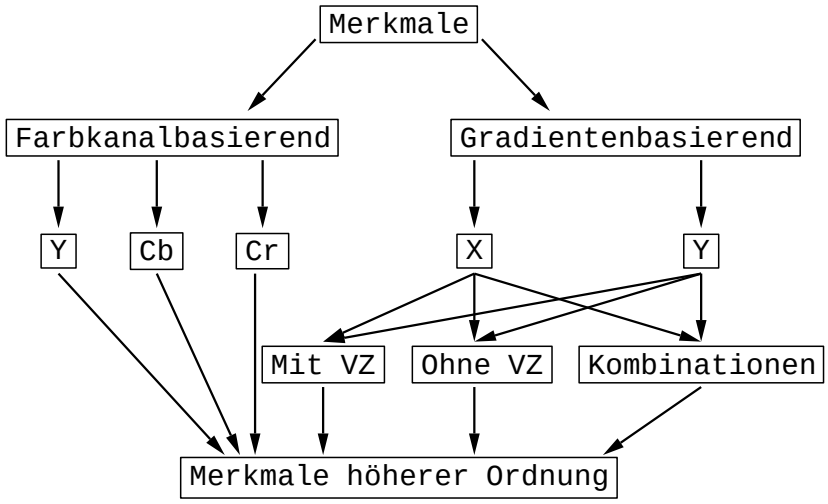


Abbildung A.1. – Diese Abbildung stellt die einzelnen Kategorien dar, in welche sich alle getesteten Merkmale unterteilen. Die farbbasierten basieren auf den einzelnen Kanälen des Farbraums, wohingegen die gradientenbasierten Merkmale die partiellen Ableitungen nach den X- und Y-Koordinaten verwenden. Mit VZ sind Features gemeint, die über ein Vorzeichen verfügen.

A.3.2. Mittelwerte der einzelnen Kanäle

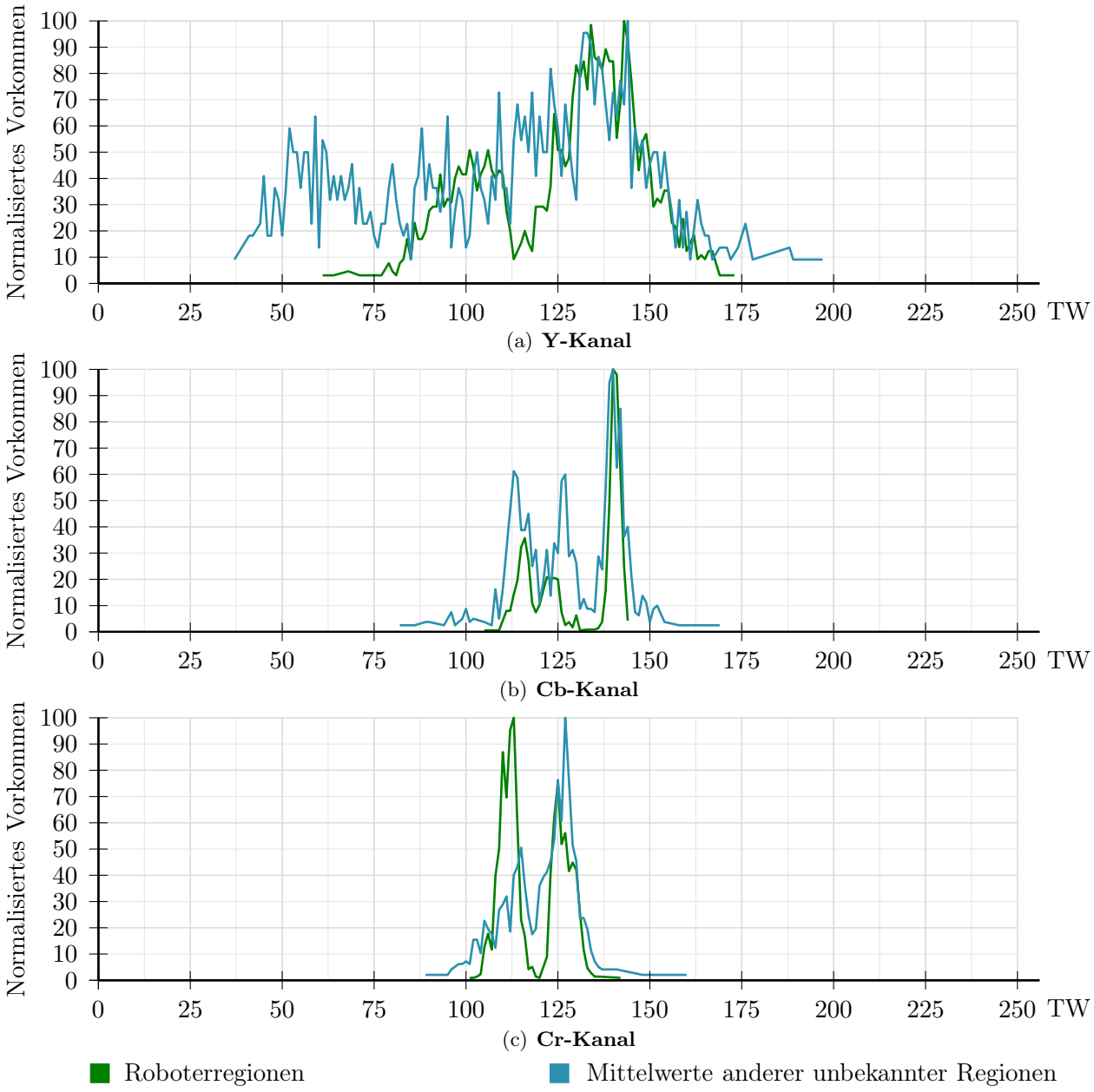


Abbildung A.2. – Mittelwerte des Kanäle von Roboterregionen und Regionen mit anderem unbekannten Inhalten. Der Graph zeigt die einzelnen Tonwerte (TW) in Relation zur Häufigkeit ihres Vorkommens.

A.4. Quelltext

Quelltext A.1 – Beispieldatensatz eines getaggten Bildes.

```
<taglist>
        <tag>
            <feetX>47</feetX>
            <feetY>158</feetY>
            <found>true</found>
            <hashCode>1205299100</hashCode>
            <path>/Testbilder/testdatenset/9999_0007.png</path>
            <robotCount>0</robotCount>
            <teamcolor>1</teamcolor>
            <waistbandX_end>55</waistbandX_end>
            <waistbandX_start>40</waistbandX_start>
            <waistbandY>44</waistbandY>
        </tag>
        <tag>
            <feetX>298</feetX>
            <feetY>221</feetY>
            <found>true</found>
            <hashCode>-896801236</hashCode>
            <path>/Testbilder/testdatenset/1000_0529.png</path>
            <robotCount>0</robotCount>
            <teamcolor>2</teamcolor>
            <waistbandX_end>303</waistbandX_end>
            <waistbandX_start>283</waistbandX_start>
            <waistbandY>136</waistbandY>
        </tag>
        <tag>
            <feetX>345</feetX>
            <feetY>226</feetY>
            <found>true</found>
            <hashCode>-1167849398</hashCode>
            <path>/Testbilder/testdatenset/1001_0530.png</path>
            <robotCount>0</robotCount>
            <teamcolor>2</teamcolor>
            <waistbandX_end>354</waistbandX_end>
            <waistbandX_start>334</waistbandX_start>
            <waistbandY>144</waistbandY>
        </tag>
</taglist>
```