HTWK
Hochschule für Technik,
Wirtschaft und Kultur Leipzig

FAKULTÄT INFORMATIK & MEDIEN
STUDIENGANG INFORMATIK

Bachelorarbeit

Bildverarbeitungsmethoden zur zuverlässigen Mustererkennung auf dem Roboterjersey im humanoiden Roboterfußball

LEA KUNZ

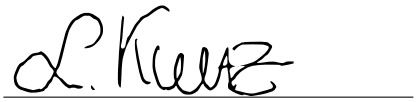
SEPTEMBER 2023 – LEIPZIG

BETREUER:
PROF. DR.-ING. JEAN-ALEXANDER MÜLLER
M. SC. TOBIAS JAGLA

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, September 2023



Lea Kunz

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

QUELLTEXTVERZEICHNIS

ACRONYMS

SPL Standard Platform League

HCC2D High Capacity Colored 2 Dimensional

Teil I

BILDVERARBEITUNGSMETHODEN ZUR ZUVERLÄSSIGEN MUSTERERKENNUNG AUF DEM ROBOTERJERSEY IM HUMANOIDEN ROBOTERFUSSBALL

1 RELEVANZ FÜR DEN ERFOLG IM HUMANOIDEN ROBOTERFUSSBALL

Ohne moderne Bildverarbeitung wären viele Dinge, die als Standard in unserer heutigen Gesellschaft hingenommen werden, undenkbar. Die Einsatzgebiete sind vielfältig und reichen von der Früherkennung verschiedener Tumore [20], Personenkontrolle am Flughafen [26] bis hin zur einfachen Einparkhilfe im Auto [7]. So ist ein Ziel des wissenschaftlichen Wettkampfs *„RoboCup“*, die Forschung in der Bildverarbeitung und anderen Bereichen der Informatik dynamisch in Bewegung zu halten. Seit 2009 treten die *„HTWK Robots“* [8] (ehemals *„Nao-Team HTWK“*) von der HTWK Leipzig regelmäßig in der SPL des Wettkampfes erfolgreich an.

1.1 MOTIVATION

Die SPL ist eine spezielle Liga des *RoboCups*, welche mit dem Forschungsroboter *„Nao“* ausgetragen wird. In der SPL ist es zwingend notwendig, dass die Roboter ihre eigene Jerseynummer zu Beginn des Spieles kennen, da so den Robotern bei einem Regelverstoß eine Strafe zugeordnet werden kann. Für jedes Spiel wird der GameController [1] so konfiguriert, dass er alle Spieler mit ihrer Jerseynummer listet. Über diesen wählt ein menschlicher Schiedsrichter bei einem Regelverstoß den zu bestrafenden Roboter aus. Üblicherweise sind die Strafen in Form einer kurzen Auszeit, auf die die Roboter sofort reagieren müssen, indem sie in eine Art Standby-Modus wechseln. Anhand ihrer Jerseynummer wissen die Spieler, wenn sie eine Strafe erhalten haben.

Es gibt bis jetzt keine technische Möglichkeit, die Jerseynummer des Roboters automatisiert, zu bestimmen, da die Jerseynummer den Robotern nicht eindeutig zugeordnet sind. Je nach technischen Mängeln werden die Roboter als Spieler ausgewählt und vor dem Spiel die Jerseynummern mit dem Anziehen der Jerseys durch das (menschliche) Team determiniert. Erst dann kann eine automatisierte Zuordnung stattfinden.

Bislang erfolgt die Konfiguration der Roboter mit ihren Jerseynummern manuell vor jedem Spiel über die Kommandozeile, nachdem den Robotern ihre Jerseys angezogen worden. Jedoch ist dieser Vorgang zeitaufwändig, stressbehaftet und anfällig für menschliche Fehler. Deshalb soll die Erkennung der Jerseynummern durch die Robo-

1 GameController ist die offizielle Bezeichnung für den Schiedsrichtercomputer in der SPL, vgl. [18]

ter erfolgen, wenn ein Spiel beginnt oder ein neuer Roboter während des Spiels eingewechselt wird. Dafür soll ein Muster im Sichtfeld der unteren Kamera des Roboters auf dem Jersey platziert werden, anhand dessen der Roboter seine Nummer selbstständig erkennt.

Die Analyse des Musters muss zuverlässig, robust und eindeutig erfolgen. Eine automatische Erkennung der Jerseynummer wurde in der SPL bislang noch nicht umgesetzt, daher gibt es keine Erkenntnisse, welche Muster sich eignen und welche äußeren Einflüsse die Identifikation erschweren. Zu letzteren zählen verschiedene Lichtverhältnisse und verrutschte Jerseys. Ebenfalls ist unklar, welche Methoden der Bildverarbeitung für eine erfolgreiche Identifikation des Musters auf dem Jersey ideal sind.

1.2 ZIELSETZUNG UND ABGRENZUNG

Das Ziel der Arbeit ist es, ein Muster zu entwerfen, welches die jeweilige Jerseynummer codiert. Weiterhin soll erforscht werden, welche Methoden der Bildverarbeitung eine zuverlässige Identifizierung des Musters auch unter schwierigen Verhältnissen gewährleisten. Für das Design des Musters ist es essenziell, vorher notwendige Anforderungen an dieses zusammenzutragen. Zu relevanten Kriterien gehören beispielsweise die Größe, mögliche Menge an codierbaren Informationen und Farben. Die finale Gestaltung soll anhand der Anforderungen bewertet werden.

Außerdem ist eine Analyse möglicher Szenarien erforderlich, in denen Einflussfaktoren die Sichtbarkeit des Codes im Kamerabild behindern. Anhand dieser soll eine breite Reihe an Testbildern aufgestellt werden, an denen zum Schluss eine erfolgreiche Erkennung des Musters gemessen wird. Dafür ist ein Bewertungsmaß zu definieren sowie Schwellwerte für akzeptable und nicht akzeptable Ergebnisse zu finden.

Mithilfe eines entwickelten Algorithmus sollen auf den aufgenommenen Bildern des Musters verschiedene Methoden der Bildverarbeitung angewandt werden, welche die Qualität der Bilder für die spätere Erkennung verbessern sollen. Gängige Techniken hierfür zielen auf die Veränderung der Schärfe und des Kontrastes ab. Weiterhin soll ein Algorithmus entwickelt werden, welcher auf verbreiteten Methoden der Mustererkennung, wie der Farbwerterkennung, beruht. Mit dessen Hilfe soll das Muster in den Bildern detektiert werden.

Auf die Decodierung des Musters in die entsprechende Jerseynummer und der damit verbundenen Fehlererkennung und -korrektur wird nur am Rande der Arbeit eingegangen. Kein Bestandteil der Abhandlung bilden Verfahren aus dem Bereich des maschinellen Lernens. Diese werden zwar häufig in der Mustererkennung eingesetzt, sind jedoch aufgrund ihrer Rechenintensität ungeeigneter als klassi-

sche Methoden der Bildverarbeitung. Auch der direkte Softwaretest auf dem *Nao* ist nicht Teil dieser Abhandlung.

1.3 AUFBAU DER ARBEIT

Im nachfolgenden Kapitel 2 werden essenzielle Hintergrundinformationen zum *RoboCup*, seinen Zielen und Ligen, insbesondere der hier relevanten SPL, aufgeführt. Außerdem werden wichtige Hardwareeigenschaften zum Forschungsroboter *Nao* erläutert, der in dieser Arbeit verwendet wird. Ebenfalls wird der Farbraum *YCbCr* beschrieben, mit dem die Kameras des *Nao* arbeiten.

Kapitel 3 dieser Abhandlung beschäftigt sich mit dem Design des Jerseymusters. Es wird geklärt, welchen Ansprüchen das Muster gerecht werden muss, ob sich gängige Codearten eignen und wie die finale Gestaltung den Kriterien gerecht wird. Dabei spielt auch die Platzierung des Musters auf dem Jersey eine Rolle.

Eine Beschreibung zum Aufbau, Inhalt und Zweck der Testdatenbank leitet das Kapitel 4 ein. In diesem Kapitel wird ebenfalls darauf eingegangen, welche Methoden der Bildverarbeitung sich eignen, um die Qualität der Ausgangsbilder zu verbessern. Diese Bilder bilden die Grundlage für die Entwicklung eines Algorithmus, der auf Techniken aus der Bildanalyse zurückgreift und das Muster erkennen soll.

Inwieweit der Algorithmus seiner Aufgabe gerecht wird, ist Thema des 5. Kapitels. Dafür wird dieser einem praktischen Test an den Versuchsbildern unterzogen und der Erfolg der Mustererkennung bewertet.

Zum Schluss der Arbeit wird in Kapitel 6 ein Ausblick gegeben, welche weiteren Möglichkeiten der Optimierung für den entwickelten Algorithmus existieren. Außerdem soll angeschnitten werden, wie die physischen Möglichkeiten des *Naos* für eine zuverlässigere Erkennung des Musters beitragen können.

2 GRUNDLAGEN

2.1 STANDARD PLATTFORM LEAGUE ALS TEIL DES ROBOCUPS

Die zentrale Intention des *RoboCups* besteht darin, mit Fußball die Öffentlichkeit für die Forschungsgebiete Robotik und Künstliche Intelligenz zu begeistern und so die Forschung auf beiden Gebieten zu fördern. Der erste Wettkampf fand 1997 im Beisein von 40 Teams und 5000 Zuschauern. Zu den Gründern zählen Universitätsprofessoren, unter anderem aus Japan und den USA. [1]

Um die Motivation für den Wettkampf anzuspornen, setzt sich der *RoboCup* das ehrgeizige Ziel, dass

> „BY THE MIDDLE OF THE 21ST CENTURY,
> A TEAM OF FULLY AUTONOMOUS
> HUMANOID ROBOT SOCCER PLAYERS
> SHALL WIN A SOCCER GAME,
> COMPLYING WITH THE OFFICIAL RULES OF FIFA,
> AGAINST THE WINNER OF THE
> MOST RECENT WORLD CUP.“ [17]

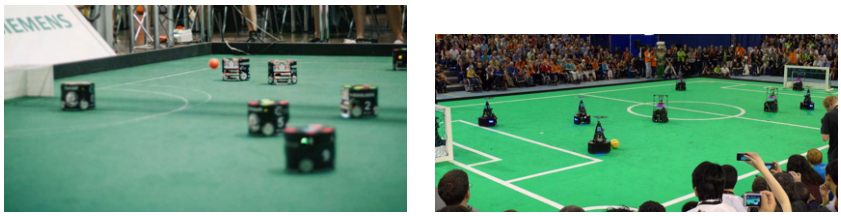
Ein charakteristisches Merkmal des *RoboCups* ist die dynamische Weiterentwicklung seiner Regeln. Infolge werden die Herausforderungen der Liga realitätsnäher gestaltet, wodurch die entwickelten Lösungen einen höheren Grad an praktischer Relevanz besitzen. Weiterhin bietet der Wettkampf die Möglichkeit, an realen und komplexen Problemen in einem abgegrenztem Umgebungsfeld zu forschen, ohne dabei hohe Kosten in Kauf zu nehmen. [17]

Neben dem Roboterfußball in der *RoboCupSoccer League* fördert das Turnier auch andere Bereiche der Robotik. So wird unter anderem in der *RoboCupRescue League* an leistungsstarken Robotern für schwierige Rettungsmissionen geforscht und bei *RoboCup@Home* an Haushaltsrobotern für alltägliche Aufgaben. Auch die Nachwuchsforschung wird mit der *RoboCupJunior League* unterstützt. Die *RoboCupSoccer League* ist ihrerseits nochmal in einzelne Ligen unterteilt.

Zu einer der ältesten *RoboCupSoccer* Ligen gehört dabei die SMALL SIZE LEAGUE, welche sich mit der Zusammenarbeit von intelligenten Multi-Agenten-Systemen in einer äußerst dynamischen Umgebung befasst. Ein Spiel wird zwischen zwei Teams mit je sechs Robotern auf einem grünen Feld der Größe 9 x 6 m² ausgetragen. Als Ball dient ein orangefarbener Golfball. Die in Abbildung 1a dargestellten, auf Rädern fahrenden Roboter werden von einem zentralen Rechner gesteuert, wobei die Kommunikation über Funk erfolgt. Alle Objekte

auf dem Spielfeld werden von einem standardisierten Visionssystem erfasst, das die Daten von Kameras verarbeitet, die um das Spielfeld platziert sind.[2]

In der nächstgrößeren Liga, der MIDDLE SIZE LEAGUE, spielen pro Team jeweils fünf vollständig autonome Roboter auf einem 18x12 m² Spielfeld gegeneinander, wie in Abbildung 1b zu sehen. Ein regulärer FIFA-Spielball kommt bei den Spielen zum Einsatz. Die Teams entwerfen ihre eigene Hardware, müssen jedoch sicherstellen, dass alle Robotersensoren installiert sind und die Roboter eine Größe von 80 cm sowie ein Gewicht von 40 kg nicht überschreiten. Während eines Fußballspiels dient aufgrund der Gegebenheiten WLAN als Kommunikationskanal. Der Schwerpunkt der Forschung liegt hier auf dem Design der Mechatronik sowie auf dem autonomen Verhalten der Roboter. [10, 14]



(a) *Small Size* League [2] (b) *Middle Size* League [10]

Abbildung 1: Roboter bei Spielen in der *Small Size* und *Middle Size* League.

Die HUMANOID LEAGUE stellt einen Kontrast zu den bisher vorgestellten Ligen dar, da Spiele in dieser Liga von Robotern mit einem menschlichen ähnlichen Körper und Sinnen bestritten werden. Zu den größten Herausforderungen der Teams zählen dabei das dynamische Gehen, Gleichgewichtsverhalten, Spielen des Balls und die Erkennung anderer Spieler und des Spielfelds. Zudem werden Themen wie die Selbstlokalisierung und das Zusammenspiel im Team intensiv erforscht. Bei einem Spiel treten jeweils drei autonome Roboter auf einem 6 x 4 m² großen Spielfeld an, welche mittels WLAN kommunizieren. Die Liga ist ihrerseits in drei Unterligen aufgeteilt[14, 16]:

1. die *KidSize* League mit 40-90 cm großen Robotern, beispielhaft in Abbildung 2
2. die *TeenSize* League mit Robotern der Größe 80-140 cm
3. die *AdultSize* League mit Höhenvorschriften 130-180 cm

In der **Simulation League** wird hingegen ganz auf Hardware verzichtet und die Fußballspiele werden stattdessen virtuell ausgetragen. Dabei treten elf autonome Spieler, die Agenten, auf einem computergenerierten Spielfeld gegeneinander an. Als eine der ältesten Ligen des *RoboCups* liegt der primäre Fokus allein auf Strategiespiel und

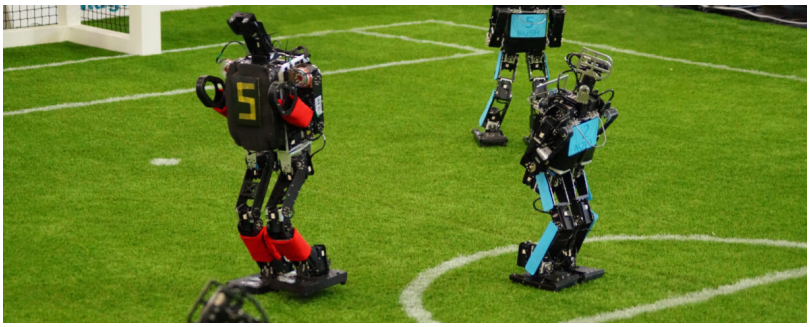


Abbildung 2: Die *KidSize* League ist eine Unterliga der Humanoid League. [16]

künstlicher Intelligenz. Auch hier existiert eine Aufteilung in mehrere Unterligen: die 2D-Simulation und 3D-Simulation, wie in Abbildung 3 dargestellt. [19]

(a) 2D Simulation (b) 3D Simulation

Abbildung 3: Spiele in der 2D und 3D Simulation League. [19]

Die **SPL** unterscheidet sich von den anderen Ligen in der Hinsicht, dass allen Teams die gleiche Hardware zur Verfügung steht und somit die individuelle Software spielentscheidend ist. Aus diesem Grund besitzen finanzielle Mittel weniger Einfluss auf das Spielgeschehen, da alle Teams mit identischen Voraussetzungen arbeiten. So wird ein Spiel beispielsweise nicht von der unterschiedlichen Leistung eingebauter Motoren und Sensoren beeinflusst, die sich auf die Schnelligkeit und Balance der Roboter auswirken. Außerdem wird so der Fokus gezielt auf softwareseitige Innovationen und Entwicklungen gerichtet und ein realistischer Vergleich der verschiedenen Programmierlösungen ermöglicht.

Zu den Forschungsschwerpunkten in der SPL zählen die folgenden fünf Schwerpunkte:

1. der dynamische Lauf auf zwei Beinen
2. die visuelle Umgebungsanalyse
3. Ballmanipulation
4. Teamstrategie

5. Selbstlokalisation

Von gleichen Ausgangsbedingungen profitieren ebenfalls die Teams, da zum einen der gegenseitige Austausch erleichtert wird, wenn alle Teams mit der gleichen Hardware arbeiten. Ebenfalls finden neue Teams durch wiederverwendbare Programme leichter einen Einstieg in die Liga und können in kürzerer Zeit den Stand von etablierten Teams erreichen. Ein weiterer Vorteil sind die verhältnismäßig niedrigen Entwicklungskosten im Vergleich zu selbstgebauten Robotern sind.

Derzeit dient als Standardplattform der Forschungsroboter *Nao* der französischen Firma *Aldebaran*, welche 2008 den vorher eingesetzten Roboter *„AIBO"* der Firma *Sony* ablöste. [24] In einem Spiel treten pro Team jeweils 7 *Naos* an, welche völlig autonom auf einem grünen 10,4 x 7,4 m² großen Kunstrasenfeld agieren. Die Feldlinien sind denen des richtigen Fußballs identisch und auch der Spielball besitzt den fußballtypischen schwarz-weißen Aufdruck. Während des Spiels erfolgt die Kommunikation über WLAN. [18] Eine Momentaufnahme eines Spiels ist in Abbildung 4 zu sehen.

Abbildung 4: Ein Spiel der *HTWK Robots* in der SPL.

Die in dieser Arbeit entwickelten Algorithmen sind für den Einsatz auf dem *Nao* optimiert, weswegen im nachfolgenden Kapitel einige technische Details des Roboters erläutert werden.

2.2 STANDARDPLATTFORM NAO

Aldebarans *Nao*, illustriert in Abbildung 5a ist ein weltweit bekannter Forschungsroboter, der in vielen akademischen Einrichtungen für praktische Ziele eingesetzt wird. Bei einer Größe von ca. 58 cm, einem

Gewicht von ca. 4,3 kg und 25 verschiedene Freiheitsgraden verfügt er über die notwendige Agilität, die seine Verwendung als Standardplattform in der SPL ermöglichen. [12]

Zur Datenverarbeitung dient dem *Nao* der im Kopfteil verbaute *Atom E3845* Prozessor mit einer Taktfrequenz von 1.91 GHz sowie insgesamt 4 GB DDR3 RAM. Ein USB-Stick stellt den permanenten Speicher bereit. Für den netzwerkseitigen Anschluss besitzt der *Nao* eine LAN und WLAN Schnittstelle, wobei über WLAN die essenzielle Kommunikation der Roboter mit dem GameController während der Spiele erfolgt. Weiterhin ist Linux als Betriebssystem installiert. Sein humanoides Design ist mit sieben Sensoren ausgestattet, darunter Sensoren zur Abstandsmessung und Gelenkwinkelsensoren zur präzisen Bewegungssteuerung. Zusätzlich sind vier Mikrofone, diverse Buttons und Lautsprecher im *Nao* verbaut, mit denen der *Nao* ebenfalls mit seiner Umwelt agiert. [22]

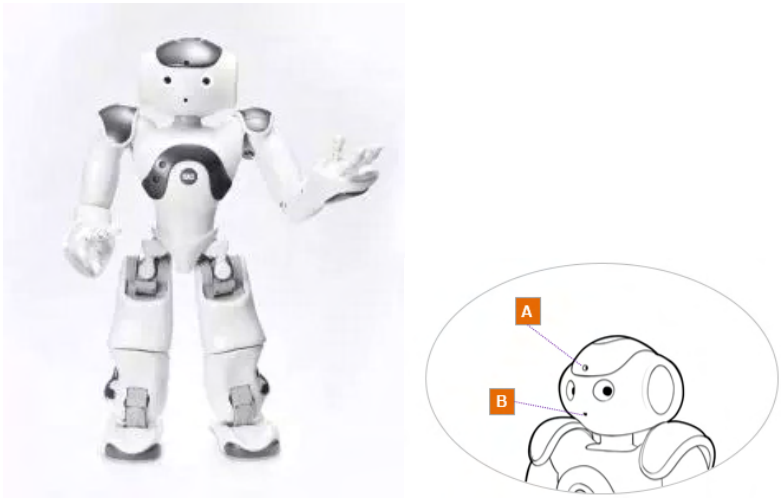
Zentral für die Leistungsfähigkeit des *Nao* Roboters auf dem Feld ist sein anspruchsvolles Kamerasystem, das eine umfassende Sicht auf die Umgebung bieten. Die Positionen der zwei im Kopf verbauten 2D Video Kameras sind in Abbildung 5b zu sehen. Sie verfügen über eine Auflösung von 1560x1920 bei 1 fps oder 640x480 bei 30 fps, wobei letztere derzeit bei den *HTWK Robots* verwendet wird. Die Kameras sind dem CSI Kamera Model 5MP OV5640 zuzuordnen. Softwareseitig können mehrere Einstellungen der Kameras festgelegt werden, zu denen Helligkeit, Kontrast, Sättigung, Schärfe und Fokus zählen. [22] Mittels der unteren der beiden Kameras soll der Roboter später das Muster auf seinem Jersey detektieren, welches dessen Jerseynummer codiert. Für den reibungslosen Ablauf von Spielen in der SPL ist es eine notwendige Voraussetzung, dass die Roboter ihre eigene Jerseynummer kennen. [1]

Zur Abbildung der Farbinformationen greifen die Kameras des *Naos* auf das *YCbCr*-Farbmodell zurück. Die in der Arbeit vorgestellten Algorithmen basieren ebenfalls auf diesem Farbmodell, da die aktuelle Bildverarbeitung der *HTWK Robots* auf das *YCbCr*-Farbmodell zurückgreift, um keine Rechenzeit für die Konvertierung in ein anderes Modell verloren geht. Nachfolgend wird das Farbmodell erläutert.

2.3 FARBMODELL YCBCR

Das *YCbCr*–Farbmodell gehört zusammen mit *YUV* und *YIQ* zu einem der weit verbreiteten Farbräume für das digitale Fernsehen und spielt dort eine essenzielle Rolle in der Übertragung, Speicherung und Darstellung von TV-Signalen. Er findet ebenfalls in vielen weiteren Bereichen der digitalen Bild- und Videoaufzeichnung Anwendung, wie beispielsweise der Kompression von JPEG Bildern. Dabei

1 Vgl. Abschnitt 1.1



(a) NAO Roboter [12] (b) Kamerapositionen [23]

Abbildung 5: Der humanoide NAO Roboter besitzt zwei Kameras zur Wahrnehmung seiner Umgebung.

ist *YCbCr* eine international standardisierte Weiterentwicklung von *YUV*. [3]

Im *YCbCr*–Farbmodell wird ein Pixel durch seine Grundhelligkeit (Luminanz) *Y* und die zwei Farbkanäle *Cb* (Blue-Yellow Chrominanz) und *Cr* (Red-Green Chrominanz) charakterisiert. Im Gegensatz zum *RGB*-Farbmodell werden bei *YCbCr* nicht die direkten Farben codiert, sondern stattdessen die Farbunterschiede. [3] So wird mit dem *Cb* Wert und dem *Cr* Wert die Abweichung von einem neutralen Grauton in ihre respektiven Farbkanäle bestimmt. [14]

Im *YCbCr* Modell können im Vergleich zu *RGB* die Bilddaten stark kompromittiert werden, ohne in einen Qualitätsverlust zu resultieren. Das ist infolge der Trennung von Luminanz und Chrominanz möglich, da zur Übertragung dieser Informationen unterschiedliche Bandbreiten verwendet werden.

Hierfür wurde sich an dem menschlichen Auge orientiert, welches detaillierte Veränderungen im Farbton weit weniger wahrnimmt als in der Helligkeit. So liegen Schattierung und Reflexion, die vom Menschen besonders intensiv erkannt werden, hauptsächlich einer Änderung der Helligkeit zugrunde und nur geringfügig einer des Farbtons. Das ist beispielhaft in Abbildung 6 verdeutlicht. So ist es ohne sichtbaren Qualitätsverlust möglich, die Informationsmenge der Chrominanzkanäle soweit zu reduzieren, dass sie nur etwa 25 % der Menge an übertragenen Information der Luminanz entspricht. [3, 14]

In der Software der *HTWK Robots* wird auf den *YCbCr* Farbraum zurückgegriffen, um beispielsweise die grüne Feldfarbe von den weißen Feldlinien zu unterscheiden und die Jerseyfarbe der gegnerischen Mannschaft zu erkennen.

Abbildung 6: Die Darstellung eines Bildes in seinem *Y* Kanal, *Cb* Kanal und *Cr* Kanal zeigt, dass der Mensch den *Y* Kanal am besten erkennt. [6]

3 CODIERUNG IN FORM EINES MUSTERS

3.1 VORAUSSETZUNGEN DURCH DEN NAO

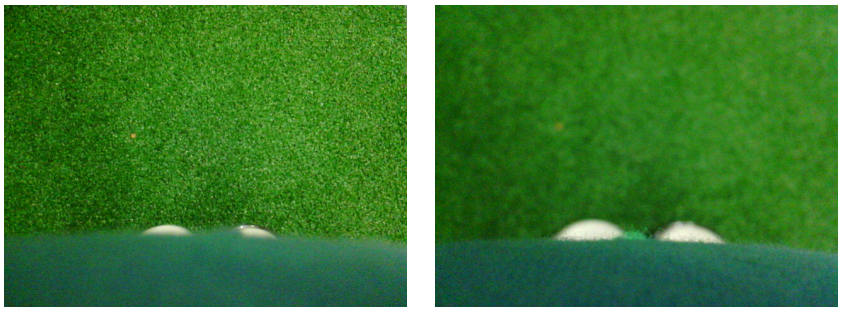
Das geeignete Design des Musters, welches die Jerseynummern codiert, ist ein essenzieller Bestandteil für die erfolgreiche Mustererkennung. Daher sollten passende Kriterien für den Entwurf definiert werden. Eine wichtige Voraussetzung für diese ist die Bestimmung des verfügbaren Platzes und der Position des Musters auf dem Jersey.

Eine geeignete Bewegung für ein breites Bild auf das Jersey des *Naos* ist die geradlinige Absenkung seines Kopfes auf die Roboterbrust. Wie in Abbildung 7a dargestellt, ist das Jersey so zu ca. 25 % im unteren Kamerabild zu sehen. Die andere Kamera des *Naos* befindet sich zu weit oben am Kopf, als dass sie das Jersey einfängt. Eine mittige Platzierung des Jerseys in den durch die gerade Kopfbewegung erkennbaren Bereich bietet so den Vorteil, dass der Musterentwurf möglichst breit ausfallen kann. Gleichzeitig befindet sich das Muster so auch bei verrutschtem Jersey häufiger im Kamerabild, als wenn es im linken oder rechten Bildbereich platziert wird.

Eine sinnvolle Position für das Muster bildet somit der obere, mittige Jerseybereich. Im Gegensatz dazu befindet sich die richtige Jerseynummer unter der linken Schulter des *Naos* und ist deshalb nicht vollständig mit der Kamera zu sehen.

Um die Qualität der Bilder zu steigern, sind weiterhin diverse Kameraeinstellungen zu treffen. Dazu zählt insbesondere der Fokus, der während der Spiele immer auf automatisch gestellt ist, um die wichtigsten Objekte in der Kamera scharf zu erkennen. Um einen klaren Blick auf das Jersey zu erhalten wird der Fokus für die Mustererkennung jedoch auf den unteren Bildbereich gesetzt, da sich dort das Jersey befindet. Bei den meisten Bildern wird mit einer höheren Helligkeit ebenfalls die Qualität verbessert, da die Aufnahmen oft unter dunkleren Lichtverhältnissen entstehen. Weitere Kameraparameter sind im Anhang in Tabelle 3 gelistet. Anhand dieser Einstellungen wird das Rauschen in den Bildern, wie in Abbildung 7b zu sehen, abgeschwächt.

Aus den Gegebenheiten lässt sich ableiten, dass ein passendes Muster auf den wenigen Platz und der vorhandenen Auflösung zugeschnitten werden muss. Die Mustererkennung soll gleich nach dem Einschalten des Roboters zu Beginn der Spiele ausgeführt werden. Wird das Jersey während eines Spieles getauscht, muss der *Nao* dafür ausgeschaltet werden. Im eingeschalteten Zustand gestaltet sich ein Wechsel sehr schwierig, da der Roboter seine Gelenke versteift



(a) Vorher (b) Nachher

Abbildung 7: Mit dem Anpassen der Kameraeinstellungen verbessert sich der Blick auf das Jersey deutlich. Durch das gerade Absenken des Kopfes wird ein möglichst breites Bild auf die Roboterbrust erzielt.

und sich bewegt. Der *Nao* führt somit stets beim Start der Software die Mustererkennung aus, um sich über einen Wechsel seines Jerseys bewusst zu werden.

3.2 ANFORDERUNGEN AN DAS DESIGN

Eines der Hauptkriterien an das Muster ist die Menge an Informationen, die es speichern kann, da diese die Anzahl der verfügbaren Jerseynummern abdecken muss. Seit der neuesten Regeländerung im Jahr 2023 dürfen die Roboterjerseys mit den Nummern 1-20 ausgestattet werden, welches in eine Mindestmenge von 20 codierbaren Informationen resultiert.

Außerdem spielt die Größe des Designs eine Rolle, da dieses in einem einzigen Kamerabild zu sehen sein sollte. Der verfügbare Platz in der unteren Kamera ist sehr klein, da sich das Jersey sehr nah an der Kamera befindet. Das Muster ist auf eine Breite von ca. 1,7 cm und eine Höhe von ca. 0,21 cm beschränkt, wenn der gesamte Platz komplett verwendet wird. Es ist jedoch auszugehen, dass bei einem Verrutschen des Jerseys das Muster nicht mehr vollständig zu sehen ist. Bereits das kleinste Verrutschen hat direkten Einfluss auf das untere Kamerabild, da sich das Jersey sehr nah an dieser befindet.

Ebenfalls sollte das Muster bei Knicken im Jersey erkennbar sein, die beim Verrutschen des Jerseys wie in Abbildung 8a auftreten können. Dabei wird der Blick auf das Muster verzerrt, welches in eine unterschiedliche Informationsdichte resultiert. Dem wird algorithmisch entgegengewirkt, da die Erkennung Zeile für Zeile stattfindet und Farbübergänge sowie Farbbereiche analysiert.

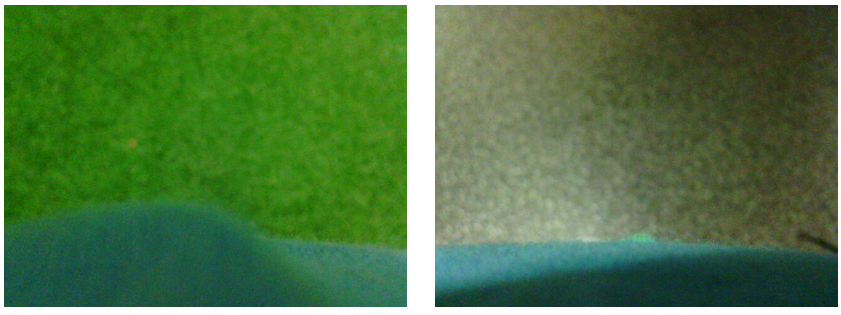
Entscheidend für mögliche Muster ist außerdem deren farbliche Gestaltung, da diese einen ausreichenden Kontrast zur Jerseyfarbe darstellen muss. Standardmäßig verwenden die *HTWK Robots* hellblaue Heimjerseys mit weißen Nummern und gelbe mit dunkelblau-

en Zahlen für Auswärtsspiele, wie im Anhang in Abbildung 23 dargestellt. Seit diesem Jahr existiert aufgrund einer Regeländerung ein grünes Jersey mit weißem Aufdruck für den Torwart. Das Muster sollte auf mindestens den drei Farben deutlich auszumachen sein.

Weiterhin können nicht nur die Farben der jetzigen Jerseynummern verwendet werden, da das Muster zur Besserung Codierung mindestens zweifarbig sein sollte und beispielsweise dunkel- auf hellblau schwierig zu erkennen ist. Ebenfalls ist ein Wechsel der Farben in den nächsten Jahre nicht auszuschließen, weswegen es vorbeugend sinnvoller ist, dass die Musterfarben einen starken Kontrast zu vielen Farben bildet. Von Relevanz ist besonders, dass im Graustufenbild ein Kontrast zu erkennen ist.

Zusätzlich sollte das Muster auch den verschiedenen Lichtbedingungen standhalten, denen das Jersey ausgesetzt ist. Dafür sollte das Muster auch unter schwierigeren Umwelteinflüssen erkennbar sein. Viele Jahre waren die Spielfelder in der SPL nur mit Deckenlampen beleuchtet, bei denen allein sich die Helligkeit von Turnier zu Turnier stark unterschied. Doch seit mehreren Jahren existiert mindestens ein Tageslicht Feld auf dem *RoboCup*, bei dem üblicherweise eine viel größere Helligkeit vorherrscht als bei den Hallenfeldern. Somit gibt es keine einheitlichen Lichtbedingungen oder -regeln für die Felder, da es nur eine Rolle spielt, ob die Felder und ihre zugehörigen Elemente ausreichend zu sehen sind. Das Muster sollte also auch bei verschiedenen Lichtverhältnissen ausreichend in der Kamera zu erkennen sein.

Jedoch sollte auch nicht von der gleichen Helligkeit in einem Kamerabild ausgegangen werden, da durch die absinkende Kopfbewegung des *Naos* ein weiteres Problem auftritt. Bei ungünstiger Beleuchtung entsteht durch den Roboterkopf Teilschatten auf dem Jersey, der eine Hälfte des sichtbaren Bereichs wie in Abbildung 8b deutlich verdunkelt. Das beeinträchtigt die Erkennung stark und muss bei der Wahl der Jerseyfarben beachtet werden.



(a) Knick (b) Halbschatten

Abbildung 8: Die Mustererkennung sollte robust gegenüber schwierigen Umweltbedingungen, wie ein Knick im Jersey oder Halbschatten, sein.

3.3 BETRACHTUNG GÄNGIGER CODEARTEN

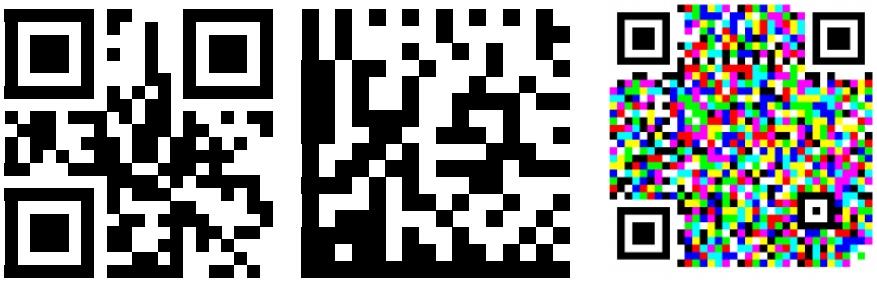
Bei der Gestaltung eines geeigneten Musters liegt es nahe, auf bereits existierende Codearten zurückzugreifen. Dazu zählt insbesondere der QR Code als einer der bekanntesten 2D Codes, welcher in Abbildung 9a illustriert ist. Der QR Code besteht aus einer Kombination von minimal 21 x 21 und maximal 177 x 177 schwarzen und weißen Modulen. Er wird häufig eingesetzt, da er die Codierung großer Datenmengen ermöglicht und zudem in der Lage ist, Fehler in der Datenübertragung zu korrigieren. So kann ein QR Code bis zu 7089 Nummern codieren und auch bei einem Verlust von 7 % bis 30 % noch erfolgreich übertragen. [28]

Als Muster zur Codierung der Jerseynummern sind die Eigenschaften des QR Codes nur bedingt von Vorteil. So überschreitet die mögliche Menge an codierbaren Informationen weit die Mindestmenge von 20, die für die Jerseynummern benötigt werden. Das umfasst eine große Menge an überflüssigen Daten. Ebenfalls setzt die Fehlerkorrektur des QR Codes voraus, dass ein Großteil des Codes sichtbar ist. Das ist bei einem nach unten verrutschtem Jersey nicht der Fall, da die Wölbung der Roboterbrust mehr als die Hälfte des Codes verdecken würde. In dieser Situation hilft es auch nicht, wenn der Roboter mit einer seitlichen Kopfbewegung einen besseren Blickwinkel anstrebt. Aufgrund seiner zweidimensionalen Struktur befinden sich weiterhin wichtige Bestandteile des QR Codes außerhalb der Reichweite der unteren Kamera. Außerdem ist zu beachten, dass das Risiko für einen unpräzisen Textildruck aufgrund der kleineren Module des Codes steigt.

Ähnliche Eigenschaften gelten ebenfalls für weitere 2D Codes wie beispielsweise der Datenmatrix, zu sehen in Abbildung 9b. Der Code ist in der Lage, bis 3116 Nummern zu speichern. Das ist eine kleinere Menge, als der QR Code ermöglicht, dafür nimmt die Datenmatrix mit 10 x 10 bis 144 x 144 Modulen weniger Platz in Anspruch. Weiterhin korrigiert der Code auch Fehler bei der Informationsübertragung [27]. Jedoch treffen die Nachteile des QR Codes auch bei der Datenmatrix zu.

Neben schwarz-weißen Designs existieren auch farbige 2D Codes wie der High Capacity Colored 2 Dimensional (HCC2D) Code in Abbildung 9c. Dieser baut auf dem QR Code auf, besitzt allerdings im Vergleich eine höhere Datendichte. [13]

Neben Matrix 2D Codes kommen auch Strichcodes in verschiedenen Anwendungszwecken zum Einsatz. Zu gängigen Arten zählen u. A. der EAN und ITF. [25] Als Jerseymuster bieten Barcodes gegenüber den Matrixcodes einige Vorteile. So bestehen diese aus breiten Streifen und benötigen daher einen weniger präzisen Druck als der QR Code oder die Datenmatrix. Außerdem gehen keine Informationen verloren, wenn das Jersey nach unten rutscht, da Strichcodes einen



(a) QR Code (b) Datenmatrix (c) HCC2D [13]

Abbildung 9: Die Matrix 2D Codes sind ähnlich, unterscheiden sich allerdings vor allem in der Menge von codierbaren Informationen und ihrer Größe.

eindimensionalen Aufbau besitzen. Ebenfalls sind die codierten Informationen intuitiver vom Menschen zu lesen als bei einem Matrixcode, da sich eine einfache Streifenabfolge besser merken lässt. So würde sich das spätere Debugging der Software vereinfachen, wenn auch ein Mensch leichter das Kamerabild in eine Jerseynummer decodieren kann.

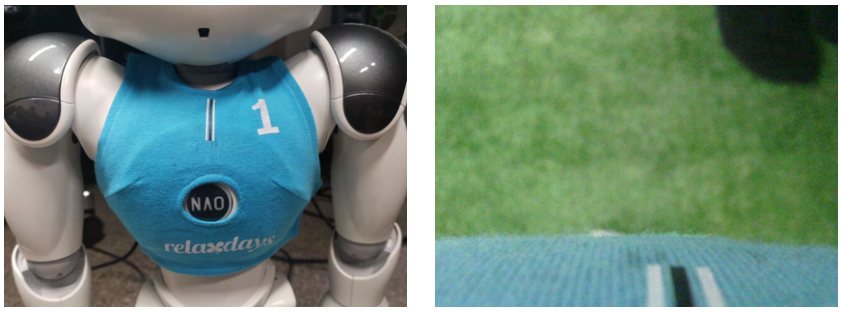
Somit sollen bekannte Strichcodes als Grundlage für ein mögliches Jerseymusters dienen. Das Design wird dabei auf die in Abschnitt 3.2 gestellten Anforderungen angepasst.

3.4 MÖGLICHE GESTALTUNG

Basierend auf den definierten Kriterien wurde ein Strichcode als Jerseymuster entworfen, welches diesen gerecht werden soll. Dieses wird im oberen mittigen Bereiches des Jerseys, wie in Abbildung 10a dargestellt, platziert.

Das Design besitzt bei fünf Streifen mit einer Breite von ca. 10 mm und einem Abstand zwischen den Streifen von ca. je 7 mm insgesamt eine Breite von ca. 78 mm. So überschreitet das Muster die Maximalbreite von ca. 1,7 cm nicht und bietet sogar noch einen kleinen Spielraum, wie in Abbildung 10b zu sehen. Das ist von Vorteil, wenn das Jersey leicht verrutscht.

Wie in Abschnitt 3.2 erwähnt, soll die spätere Erkennung des Musters Zeile für Zeile unter anderem anhand von Farbübergängen erfolgen. Dafür ist ein klarer Farbunterschied zwischen Muster- und Jerseyfarbe wichtig, um einen Wechsel von einzelnen Musterstreifen zu Jerseyfarbe bzw. von Jerseyfarbe zu einzelnen Streifen zu detektieren. Eine Erkennung des Musters allein über Kontraste ist nicht möglich, da sich zwischen den Streifen ein Abschnitt von Jerseyfarbe befindet. Jedoch verbessert ein starker Kontrast die algorithmische Identifizierung der Farbübergänge, weswegen die Farbe der Streifen von besonderer Relevanz ist.



(a) Muster von vorne

(b) Muster aus Sicht der unteren Kamera

Abbildung 10: Das entworfene Muster wird mittig in der oberen Hälfte des Jerseys platziert. So wird es den definierten Anforderungen gerecht und lässt leichten Spielraum, wenn das Jersey leicht verrutscht.

Im designten Strichcode fiel die Farbwahl auf Schwarz und Weiß, da die Töne besser geeignet sind als bunte Farben. Ein Muster in Graustufen ist auf jeder Jerseyfarbe gut zu erkennen und lässt zudem die Möglichkeit offen, in Zukunft andere Farben als blau, gelb und grün zu verwenden. Um einen starken Kontrast zur Jerseyfarbe bei einem farbigen Muster zu erzielen, würde die Variante bestehen, die Streifen auf einen neutralen Untergrund wie Grau zu drucken. Allerdings ist die Verwendung eines zusätzlichen Untergrundes bei einem schwarz-weißen Design unnötig, weswegen dieses auch aus optischen Gründen einem farbigen Muster vorzuziehen ist.

Weiterhin bieten Schwarz und Weiß einen zusätzlichen Vorteil, denn diese sind spezielle Töne im verwendeten Farbraum *YCbCr*. Im farblichen Koordinatensystems, welches die Chrominanzkanäle *Cb* und *Cr* wie in Abbildung 11 aufspannen, befinden sich Schwarz und Weiß im Koordinatenursprung befinden. Das liegt der Tatsache zugrunde, dass für beide Farbtöne der Gehalt von Blau, Gelb, Grün und Rot gleich 0 ist. Nur die Helligkeit Y ist ausschlaggebend, ob es sich um Schwarz oder Weiß handelt: Bei einem Y Wert von 1 handelt es sich um Weiß, bei einem von -1 um Schwarz. Ist der Helligkeitskanal ebenfalls 0, so liegt ein neutrales Grau vor.

Das bedeutet, dass für Schwarz und Weiß die Chrominanzkanäle den gleichen Wert haben und so die spätere Mustererkennung vereinfacht wird, da anhand dieser Bedingung alle Teile des Musters zu finden sind. Ob es sich um einen schwarzen oder weißen Teil des Codes handelt, entscheidet der Luminanzkanal. Somit sind mehrfarbige Codes für den hier vorgestellten Anwendungsfall ungeeigneter als schwarz-weiße.

Die codierten Informationen werden über die Strichfarbe und deren Anordnung übertragen. Das steht im Gegensatz zu bekannten Strichcodes, bei denen die Daten über die Breite und den Abstand

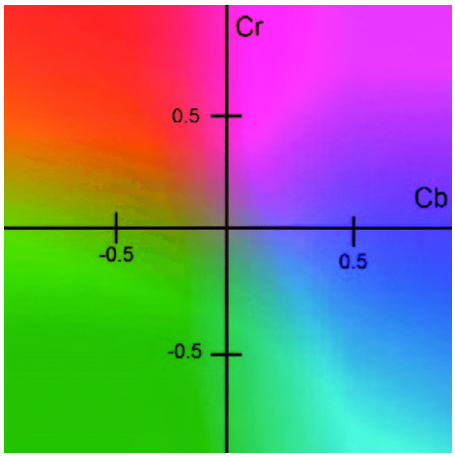


Abbildung 11: Der *Cb* Kanal und der *Cr* Kanal des *YCbCr* Farbsystems spannen ein Koordinatensystem auf, bei dem der *Y* Kanal eine eigene Achse durch den Mittelpunkt bilden würde. [15]

der Streifen codiert werden. [13]. Aufgrund der Gefahr eines ungenauen Textildrucks wurde darauf jedoch verzichtet.

Weiterhin kann das entworfene Muster insgesamt 32 Informationen speichern und bietet so neben der Codierung aller 20 Jerseynummern gleichzeitig einen kleinen Puffer, falls die Anzahl in Zukunft erweitert wird. Auf zusätzliche Streifen zur Fehlerkorrektur wurde verzichtet, da bereits die fünf Streifen den verfügbaren Platz nahezu ausreizen. Bei dem Hinzufügen weiterer Streifen würde sich das Muster vollständig über den verfügbaren Platz erstrecken und bei dem kleinsten Verrutschen des Jerseys nicht mehr komplett in der Kamera zu sehen sein.

Dafür bietet das Muster jedoch die Möglichkeit der Fehlererkennung. So gilt ein Muster nur als erfolgreich erkannt, wenn das gefundene Muster tatsächlich eine Jerseynummer codiert. Zur Codierung der Jerseynummern können dafür geeignete Musterfolgen gewählt werden, die gemeinsame Merkmale aufweisen. Die Tabelle 4 im Anhang berücksichtigt beispielsweise nur Codes, welche entweder drei schwarze oder drei weiße Streifen besitzen. Sollte ein Muster mit vier weißen, vier schwarzen, fünf weißen oder fünf schwarzen Streifen erkannt werden, ist offensichtlich, dass ein falsches Muster detektiert wurde.

Zusammenfassend zählen zu den wichtigsten Eigenschaften des entwickelten Strichcodes:

- starker Kontrast zur unterliegenden Jerseyfarbe
- geringe Breite und beliebige Höhe
- Möglichkeit zur Fehlererkennung
- Codierung der notwendigen Zahlen 1-20

- leichte Lesbarkeit und einfaches Debugging
- geringe Ungenauigkeit im Druck

Damit gestaltet sich der Strichcode als ein einfaches Muster, welches trotzdem die Kriterien erfüllt.

4

BILDVERARBEITUNGSMETHODEN FÜR EINE ROBUSTE ERKENNUNG

4.1 DATENGRUNDLAGE

4.1.1 *Erstellung der Testreihen*

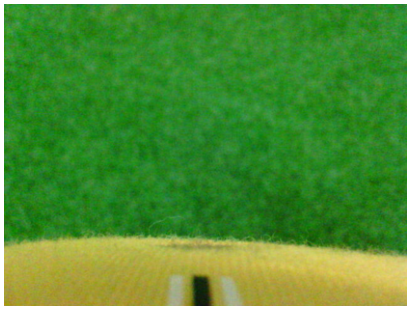
Um den Erfolg der im Rahmen dieser Arbeit entwickelten Bildverarbeitungsalgorithmen zu testen, wurden zwei verschiedene Testreihen erstellt. In den Bildern soll der entwickelte Algorithmus das aufgedruckte Jerseymuster erkennen. Zur Bildaufnahme diente das in Kapitel 2.2 vorgestellte Kamerasystem des *Nao* Roboters. Unterschiedliche Codierungen des entworfenen Musters [1] wurden mittig im oberen Bereich der Jerseys aufgedruckt, sodass diese idealerweise in der unteren Kamera des *Naos* zu sehen sind. Für die Bilder wurden die in 3.1 beschriebenen Kameraeinstellungen gesetzt und der Kopf des *Naos* gerade nach unten gesenkt. Die Daten wurden während der *GORE 2023*, dem *RoboCup* 2023 und im eigenen Roboter Labor an der HTWK aufgenommen.

Die erste Testreihe umfasst 715 Bilder. Diese wurden unter günstigen Umweltfaktoren aufgenommen, welche beispielhaft in Abbildung 12 dargestellt sind. Ein gemeinsames Merkmal der Bilder ist, dass sich das Muster relativ mittig befindet und die Streifen somit gerade im Bild erscheinen. Die Helligkeit variiert unter den Aufnahmen, jedoch stammt die Beleuchtung immer von Deckenlampen im Rauminneren, wie es bei Abbildung 12a und Abbildung 12b der Fall ist. Weiterhin schwankt die Helligkeit nicht innerhalb der Bilder, da kein Teilschatten vorkommt. Die Reihe enthält Aufnahmen von unterschiedlichen Codierungen des Musters auf verschieden farbigen Roboterjerseys, wie in Abbildung 12c und Abbildung 12d zu sehen. Auf manchen Bildern weist der mittlere Streifen einen Knick auf, da dieser auf eine Kante im Textil gedruckt ist. Dieser Fall ist in Abbildung 12d illustriert. Obwohl die Kante nur bei älteren Jerseymodellen auftritt, wird der entwickelte Algorithmus auf solche Fälle getestet, um dessen Robustheit einzuschätzen.

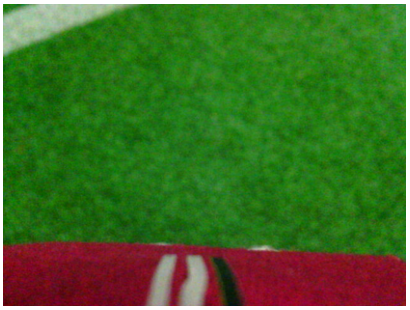
1 Das in Kapitel 3.4 präsentierte Muster umfasst fünf Streifen und codiert bis zu 32 verschiedene Informationen. In der Testreihe sind Muster mit nur drei Streifen zu sehen, da diese vor der Regeländerung 2023 aufgenommen wurden, die die Anzahl der Jerseynummern von sechs auf 20 erhöhte. Bei sechs Jerseynummern sind drei Streifen ausreichend, da diese acht Nummern kodieren können. Für die Bewertung der Bildverarbeitungsalgorithmen ist es irrelevant, ob das Muster drei oder fünf Streifen enthält. Von Relevanz ist, ob alle Streifen erkannt werden, die sich im Bild befinden.

(a) Blaues Jersey bei dunkleren Lichtverhältnissen

(b) Blaues Jersey bei helleren Lichtverhältnissen



(c) Gelbes Jersey bei helleren Lichtverhältnissen



(d) Pinkes Jersey mit Knick

Abbildung 12: Die Bilder aus Testreihe 1 entstanden unter günstigen Umweltfaktoren. Für die Auswertung zählt nur die Erkennung der Streifenfolgen, da das aktuelle Muster aus 5 Streifen besteht.

Der Umfang der zweiten Testreihe beträgt nur 50 Aufnahmen und ist in Auszügen in Abbildung 13 dargestellt. Die Bilder sind unter kritischeren Umwelteinflüssen aufgenommen, wie beispielsweise in Abbildung 13a, Abbildung 13b und Abbildung 13c im Schatten. Weiterhin fällt auf einige Aufnahmen ein ungünstiger Halbschatten, wie in Abbildung 13d zu sehen. Ebenfalls wurden Fälle betrachtet, in denen das Jersey verrutscht ist. In den entsprechenden Bildern ist das Muster nur am Rand wie in Abbildung 13b oder gar nicht zu sehen. Letztes ist in Abbildung 13a dargestellt. Parallel zu der ersten Testreihe wurden auch in der zweiten verschiedenfarbige Jerseys mit unterschiedlichen Codierungen verwendet.

4.1.2 *Kriterien für Testerfolge*

Für die Mustererkennung ist es essenziell, dass diese robust gegenüber ungünstigen Umweltfaktoren ist. Robustheit bedeutet im Rahmen der vorliegenden Arbeit, dass die Detektion auch bei unterschiedlichen Lichtverhältnissen oder einem verrutschtem Jersey noch zuverlässig erfolgen soll. Für eine hohe Zuverlässigkeit wird angestrebt,

(a) Blaues Jersey im Schatten ohne Muster

(b) Blaues Jersey im Schatten mit Muster am Rand

(c) Pinkes Jersey im Schatten

(d) Pinkes Jersey im Halbschatten

Abbildung 13: Die Bilder aus Testreihe 2 entstanden unter ungünstigen Umweltfaktoren. Für die Auswertung zählt nur die Erkennung der Streifenfolgen, da das aktuelle Muster aus 5 Streifen besteht.

dass der Algorithmus nicht nur das richtige Muster erkennt, sondern sich dieser Entscheidung ebenfalls sehr sicher ist.

Falls der Algorithmus kein eindeutiges Muster identifizieren kann, dann sollte angezeigt werden, dass kein Muster gefunden werden konnte. Anhand daran, wie häufig das korrekte Muster pro analysierter Bildzeile erkannt wird, lässt sich die Eindeutigkeit der Entscheidungen bewerten.

Es darf in keinem Fall passieren, dass ein falsches Muster erkannt wird. Dies würde fatale Folgen mit sich ziehen, da so während eines SPL Spiels beispielsweise zwei Roboter auf die gleiche Jerseynummer reagieren und ein Roboter auf seine richtige nicht. [2]

In beiden Testreihen sollte der Anteil der richtig detektierten Muster 100 % betragen, da die Zuverlässigkeit der Erkennung eine kritische Rolle einnimmt. Obwohl die Bilder der zweiten Testreihe unter schlechteren Bedingungen aufgenommen wurden, muss der Algorithmus ebenfalls bei diesen ihre Aufgabe erfüllen. Allerdings wäre es noch akzeptabel, wenn in der zweiten Testreihe nicht bei allen Bildern ein eindeutiges Muster gefunden wird. Für die erste Testreihe ist dies kritischer einzuschätzen, da der implementierte Algorithmus

2 Vgl. Abschnitt 1.1

unter optimalen Bedingungen fehlerfrei laufen sollte. Inakzeptabel für beide Testreihen wäre, wenn ein falsches Muster erkannt wird.

4.2 METHODEN DER BILDVERARBEITUNG ZUR VERBESSERUNG DER QUALITÄT

4.2.1 *Glättung der Kanten*

Um die Qualität der aufgenommenen Bilder zu verbessern, eignet sich der Einsatz geeigneter Methoden der Bildverarbeitung. Diese sollen das Rauschen im Bild verringern, damit sich die Streifen des Musters besser von dem Jersey unterscheiden lassen und ein größeres Signal-To-Noise Ratio erzielt wird. Bei dem Signal-To-Noise Ratio handelt es sich um ein Maß, das die Stärke des zu übertragenden Signals zum Hintergrundrauschen in Relation setzt. [21]

Eine geläufige Methode, um Rauschen zu verringern, ist das Weichzeichnen eines Bildes. Mit dieser Technik wird der Kontrast in Bildern verringert, in dem vorhandene Kanten geglättet werden und so ein fließender Übergang von einer Farbe in die andere erzielt wird. Als Kanten werden die Umrisse von Bildobjekten bezeichnet. Infolge der entstehenden Unschärfe bleiben grobe Strukturen erhalten, während kleine Details verschwinden. Das Glätten der Kanten erfolgt über einen passenden Filterkernel, der die Werte eines Pixels mit den Werten anliegender Pixel mittelt. [5]

Häufig wird hierfür der Gaußsche Weichzeichner verwendet, dessen Kernel eine 5x5 Matrix bildet:

$$1/256 \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

Der Kernel beruht auf der Normalverteilung. Das bedeutet, dass die größten Werte im Mittleren des Kernels liegen und sich nach außen verringern. Für jedes Pixel, das weichgezeichnet werden soll, müssen neue Werte kalkuliert werden. Dabei wird für jedes Pixel anhand des Kernel und den umliegenden Pixeln, die sich in einem Bereich gleich der Größe des Kernels befinden, ein gewichteter Durchschnitt gebildet. Das Ergebnis dient als neuer Wert für die entsprechenden Pixel. [4]

Bei dem Gaußschen Weichzeichner handelt es sich somit um einen Tiefpassfilter, da die direkt anliegenden Pixel eines Pixels mehr Einfluss auf die neuen Pixelwerte nehmen als weiter entfernte liegende Pixel. [5]

Diese Methode der Kantenglättung wurde auf die aufgenommenen Bilder angewandt, um einen fließenderen Übergang zwischen den Streifen des Musters und der Jerseyfarbe zu erzielen. Dabei wird die beschriebene Berechnung pro Bildpunkt für den Y Kanal, *Cb* Kanal und *Cr* Kanal angewandt, wie in Quelltext 1 im Anhang zu sehen ist.

Insgesamt wird der Gaußsche Weichzeichner sechsmal pro Bild in dem Bereich angewendet, in dem sich das Jersey befindet. So verbessert sich die Qualität deutlich im Vergleich zur ursprünglichen Aufnahme und das Muster ist noch zu erkennen, wie Abbildung 14 illustriert.

(a) Vorher (b) Nachher

Abbildung 14: Mithilfe des Gaußschen Weichzeichners werden die Kanten geglättet, um die Qualität der Bilder zu verbessern.

4.2.2 *Anpassung der Helligkeit zur Kontrasterhöhung*

Die unterschiedlichen Lichtbedingungen nehmen starken Einfluss auf die Sichtbarkeit des Musters, weswegen sich eine dynamische Anpassung der Helligkeit im Bild anbietet. Um eine Abbildungsfunktion für die Luminanzwerte zu bestimmen, wurde auf datenbasierte Untersuchungen zurückgegriffen. So werden Unterschiede besser berücksichtigt, die infolge stark schwankender Lichtverhältnissen eintreten.

Um Kenntnisse über die Helligkeit der Jerseyfarbe unter optimalen Spielbedingungen zu gelangen, wurde der Y Kanal der entsprechenden Pixel aus den 715 Bildern der Testreihe 1 analysiert. Diese sollen als Referenz für weitere Berechnungen dienen. Dafür wurde ein Histogramm der Y Werte der unteren 60 Zeilen jedes Bildes erstellt, welches in Abbildung 24 im Anhang zu sehen ist. Mit der in Abschnitt 3.1 beschriebenen Kopfeinstellung des *Naos* befindet sich das Jersey immer in den unteren 60 Bildzeilen, weswegen nur dieser Bereich betrachtet wird.

Für die Berechnung des Histogramms wurde versucht, die Pixel des Musters nicht zu erfassen. Diese Methode wird als Winsorizing bezeichnet und zielt auf das Herausfiltern von Extremwerten in den vorhandenen Daten ab, damit diese eine statistische Analyse nicht verfälschen. [9] Jedes Muster besteht aus entweder zwei weißen oder

zwei schwarzen Streifen, wobei die einzelnen Streifen eine Breite von ca. 35 Pixeln aufweisen. Das bedeutet, dass sich entweder Schwarz aus 70 und Weiß aus 35 Pixeln zusammensetzt oder umgekehrt. Um potenzielle Pixel des Musters herauszufiltern, wurden somit in jeder Zeile nur die Bildpunkte in die Datenbank aufgenommen, die nicht zu den 70 hellsten oder den 70 dunkelsten Pixeln gehören.

Aus dem Histogramm lässt sich ableiten, dass die meisten Pixel eine Helligkeit zwischen 128 und 148 besitzen. Weiterhin berechnete sich aus den Werten ein Median von 132 und ein Mittelwert von ca. 128. Als Referenzwert für weitere Berechnungen wird der Mittelwert dienen, da beide Werte eng beieinander liegen und sich so kaum voneinander unterscheiden. Der Durchschnitt besitzt jedoch den Vorteil, dass grundsätzlich präziser als ist der Median.

Zur Berechnung einer Funktion, die die Helligkeitswerte der Pixel auf neue Werte abbilden soll, wurde bei einer Reihe an Bildern der Kontrast des Musters durch manuelle Helligkeitsveränderungen erhöht. Als Grundlage für die Anpassung diente eine lineare Funktion, deren Anfangs- und Endwert jeweils den Wert des dunkelsten bzw. hellsten Pixels abbildete. Weiterhin wurde die lineare Funktion an einem Punkt verschoben, um den Kontrast im Bild zu erhöhen. Bei der Illustration der Verschiebungspunkte als Cluster in Abbildung 25 im Anhang ist deutlich zu erkennen, dass dunkle Bilder aufgehellt und helle abgedunkelt wurden.

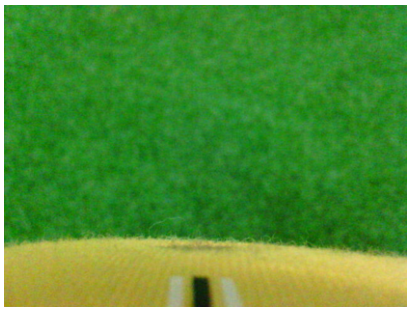
Aus dem Cluster lässt sich eine neue lineare Funktion ableiten. Gemeinsam mit der durchschnittlichen Helligkeit jeder Zeile ergibt diese eine Abbildungsfunktion, die auf die Y Werte aller Pixel der jeweiligen Zeile angewandt wird. So erhöht sich der Kontrast des Bildes, in dem für jede Zeile speziell eine Funktion zum Abbilden des Y Kanals auf neue Helligkeitswerte kalkuliert wird. Der Unterschied ist beispielhaft in Abbildung 15 zu beobachten. Dieser Schritt erfolgt nach der Verarbeitung mittels des Gaußschen Weichzeichners. Untersuchungen an Testbildern ergaben, dass das Muster so besser zu erkennen ist, als wenn erst die Helligkeit angepasst wird und dann die Glättung der Kanten erfolgt.

Wie in Abbildung 16 zu sehen, resultieren die angewandten Bildverarbeitungsmethoden zu einer deutlichen Qualitätsverbesserung.

4.3 ERKENNUNG ANHAND VON FARBWERTEN

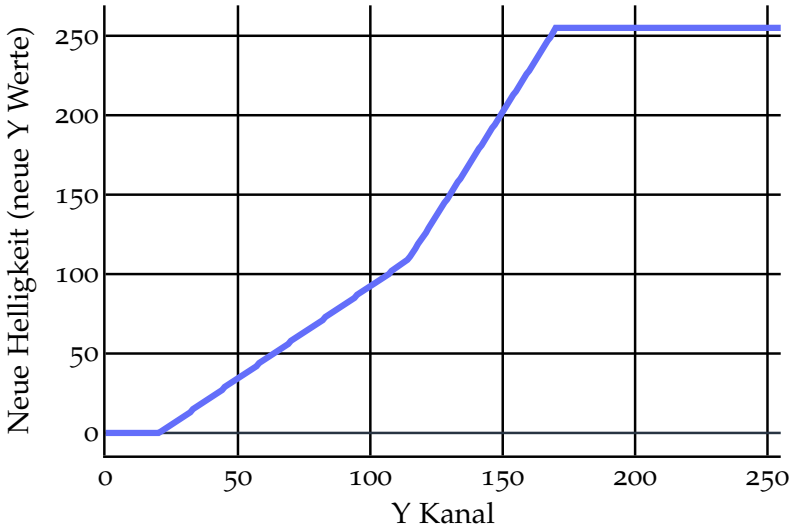
4.3.1 *Chrominanzkanäle Cb und Cr*

Infolge der Steigerung der Bildqualität erfolgt die algorithmische Erkennung des Musters unter besseren Voraussetzungen. Diese soll ähnlich dem Scanline-Algorithmus im unteren Bereich des Bildes Zeile für Zeile auf ein mögliches Muster analysieren.[11] Wie bereits in Unterabschnitt 4.2.2 erwähnt, werden nur die unteren 60 Bildzeilen



(a) Vorher

(b) Nachher



(c) Abbildungsfunktion der letzten Bildzeile

Abbildung 15: Die Veränderung der Helligkeit in jeder Bildzeile erfolgt mittels linearer Abbildungsfunktion. So wird der Kontrast im Gesamtbild erhöht.

(a) Vorher

(b) Nachher

Abbildung 16: Die Qualität der Ausgangsbilder wird durch die angewandten Bildverarbeitungsmethoden deutlich gesteigert.

betrachtet. Anhand der Häufigkeit erkannter Muster soll schließlich ermittelt werden, ob bzw. welches Muster im Bild zu sehen ist.

Anhand wechselnder Farbwerte in den Kanälen *Y*, *Cb* und *Cr* soll das Muster vom Jersey unterschieden werden. Diese Methode eignet sich gut, da Schwarz und Weiß die gleichen Werte im *Cb* Kanal und *Cr* Kanal annehmen. [3]. Somit kann anhand gleicher Parameter nach allen Streifen des Musters gesucht werden. In der existierenden Software der HTWK Robots befindet sich der Ursprung des durch die Chrominanzkanäle aufgespannten Koordinatensystems bei $(128, 128)$, da alle Kanäle auf die Werte von 0 bis 256 skaliert sind. Somit liegen in der Theorie der *Cb* Wert und der *Cr* Wert der Musterpixel bei 128. Praktisch gesehen nehmen die Kanäle nur selten diesen Wert an, da aufgrund unterschiedlicher Lichtverhältnisse die Streifen nicht als perfektes Schwarz oder Weiß erscheinen. Dennoch ergaben Untersuchungen verschiedener Bildzeilen, dass sich entweder der *Cb* Wert oder der *Cr* Wert oder beide Kanäle dem Koordinatenursprung deutlich annähern, wenn es sich um Pixel des Musters handelt. Zu beobachten ist das Verhalten in Abbildung 17. Das bedeutet, dass die Streifen des Musters dennoch die schwärzesten oder weißesten Pixel in einer Bildzeile sind.

Um die richtigen Bildpunkte zu filtern, muss ein Bereich festgelegt werden, in dem die *Cb* Werte und die *Cr* Werte von Pixeln liegen sollen. Dafür gilt es, den Kanal zu wählen, der am ehesten die Streifen repräsentiert. Mithilfe des entsprechenden Kanals werden geeignete Grenzen für valide Pixel berechnet.

Eine Methode für die Wahl des zu betrachteten Chrominanzkanals ist über den Abstand zwischen größtem und kleinsten *Cb* Wert bzw. *Cr* Wert pro Zeile. Ein weiter Abstand indiziert eine deutliche Schwankung der Farbwerte, wie in Abbildung Abbildung 17b zu sehen ist.

Wenn somit

$$\mathrm{Cb}_{\mathrm{dis}} = \mathrm{Cb}_{\mathrm{max}} - \mathrm{Cb}_{\mathrm{min}} > \mathrm{Cr}_{\mathrm{dis}} = \mathrm{Cr}_{\mathrm{max}} - \mathrm{Cr}_{\mathrm{min}},$$

dann ist *Cb* der zu betrachtende Kanal. Bei

$$\mathrm{Cb}_{\mathrm{dis}} = \mathrm{Cb}_{\mathrm{max}} - \mathrm{Cb}_{\mathrm{min}} < \mathrm{Cr}_{\mathrm{dis}} = \mathrm{Cr}_{\mathrm{max}} - \mathrm{Cr}_{\mathrm{min}},$$

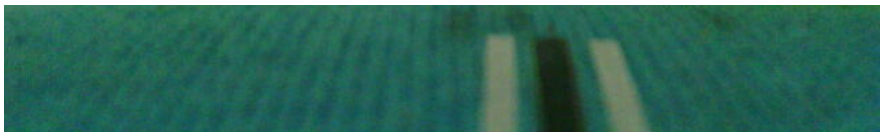
wird der *Cr* Kanal untersucht. [4]

Um nun einen Bereich für valide Musterpixel zu definieren, muss entweder eine Obergrenze oder eine Untergrenze ermittelt werden, wie in Abbildung Abbildung 18 illustriert.

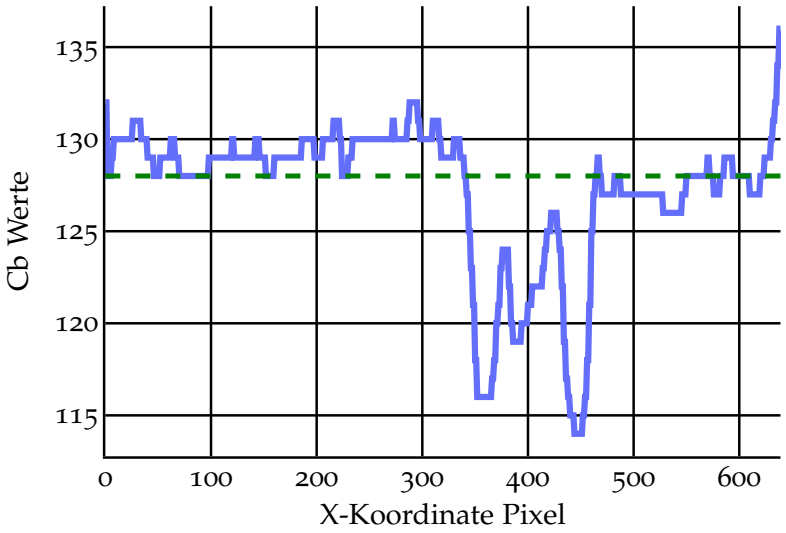
Eine Obergrenze ist erforderlich, wenn die Streifen des Musters im entsprechenden Chrominanzkanal drei Tiefen in der Kurve darstellen. Bei drei Höhen muss eine untere Grenze kalkuliert werden.

3 Vgl. Abschnitt 3.4

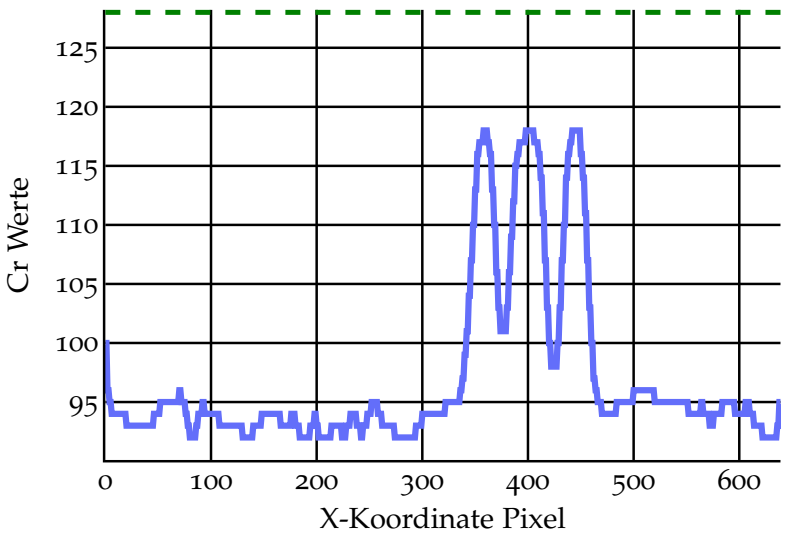
4 Sind die Werte identisch wählt der entwickelte Algorithmus aus Einfachkeit den *Cb* Kanal, da es in dem Fall keinen Unterschied macht.



(a) Muster auf blauem Jersey bei günstigen Lichtverhältnissen.

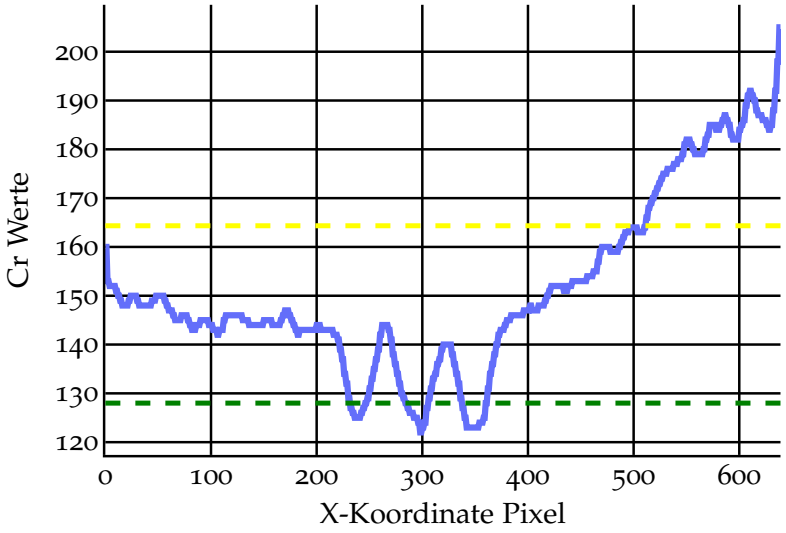


(b) *Cb* Werte von Zeile 450.

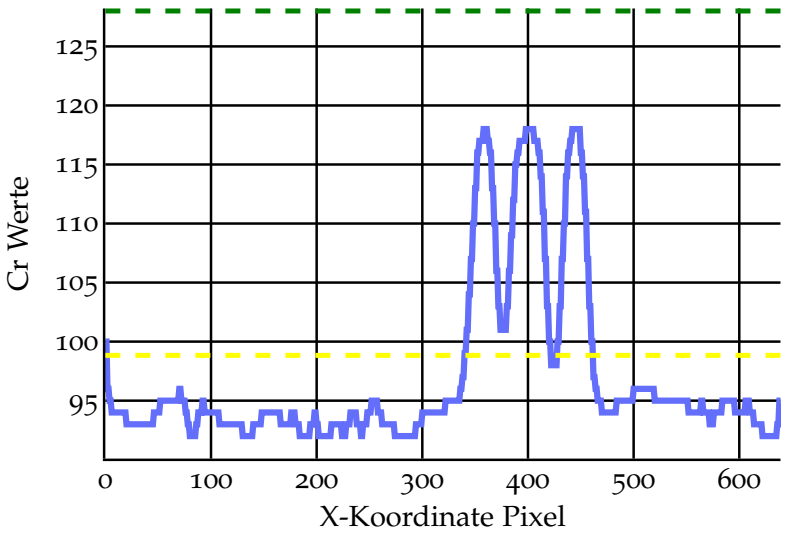


(c) Cr Werte von Zeile 450.

Abbildung 17: Am Verlauf beider Kanäle lässt sich die Position des Musters bestimmen, wobei es hier im *Cr* Kanal deutlich zu erkennen ist. Die Musterpixel sind in beiden Kanälen nah an der 128 (grüne Linie).



(a) Obergrenze für *Cr* Werte (Aufnahme im Halbschatten)



(b) Untergrenze für *Cr* Werte (Aufnahme bei guten Lichtverhältnissen)

Abbildung 18: Kriterium für Pixel des Musters ist es, dass sich die Werte des relevanten Chrominanzkanals zwischen einer kalkulierten oberen oder unteren Grenze (gelber Linie) und 128 zu liegen (grüne Linie),

Wenn die Bedingung

$$|\mathrm{Pixel}_{\max} - 128| < |\mathrm{Pixel}_{\min} - 128|$$

erfüllt ist, dann wird eine untere Grenze mit

$$\mathrm{Grenze}_{\mathrm{unten}} = \mathrm{Pixel}_{\max} - \frac{\mathrm{Pixel}_{\max} - \mathrm{Pixel}_{\mathrm{avg}}}{2} - \frac{\mathrm{Pixel}_{\mathrm{dis}}}{2}$$

definiert. Sonst wird eine obere Grenze mit

$$\mathrm{Grenze}_{\mathrm{oben}} = \mathrm{Pixel}_{\min} + \frac{\mathrm{Pixel}_{\mathrm{avg}} - \mathrm{Pixel}_{\min}}{2} + \frac{\mathrm{Pixel}_{\mathrm{dis}}}{2},$$

kalkuliert. Für die Gleichungen gilt für den jeweils zu untersuchendem Kanal

$\mathrm{Pixel}_{\max}$	Pixel höchster *Cb* Wert oder *Cr* Wert
$\mathrm{Pixel}_{\min}$	Pixel niedrigster *Cb* Wert oder *Cr* Wert
$\mathrm{Pixel}_{\mathrm{avg}}$	Durchschnittlicher *Cb* Wert oder *Cr* Wert
$\mathrm{Pixel}_{\mathrm{dis}}$	Abstand zwischen $\mathrm{Pixel}_{\max}$ und $\mathrm{Pixel}_{\min}$

Anhand verschiedener Tests ergaben sich die Gleichungen für die jeweiligen Grenzen, welche sich als geeignete Kennlinien erweisen. Befinden sich die Pixel im zu untersuchenden Chrominanzkanal Minimum und der oberen Grenze bzw. Maximum und der oberen Grenzen, sind sie ein potenzieller Bestandteil des Musters.

4.3.2 *Rolle des Luminanzkanals Y*

In vielen Bildern ist der Verlauf der Chrominanzkanäle bereits ein guter Indikator für die Position des Musters. Jedoch ist dieser in bestimmten Situationen zu ungenau, wie in Abbildung 18a zu sehen, als dass der *Cb* Kanal und *Cr* Kanal allein eine zuverlässige Erkennung garantieren. Daher wird mithilfe des *Y* Kanals die Anzahl der validen Bildpunkte weiter eingeschränkt. Zudem lässt sich anhand des *Y* Wertes bestimmtem, ob ein valides Pixel Weiß oder Schwarz repräsentiert.

Dafür sollen die dunkelsten und hellsten Pixel pro Zeile herausgefiltert werden, welche die Kriterien an die Chrominanzkanäle erfüllen. [5] Die Anzahl der jeweils dunkelsten und hellsten Pixel wird dabei durch die Bedingung

$$\mathtt{Pixelmenge} < \mathtt{AnzahlStreifenFarbe}_{\max} * \mathtt{Breite}_{\mathrm{Streifen}}$$

mit

$\mathtt{Pixelmenge}$	Anzahl der jeweils dunklen oder hellen Pixel
$\mathtt{AnzahlStreifenFarbe}_{\max}$	Maximale Anzahl der Streifen einer Farbe im Muster
$\mathtt{Breite}_{\mathrm{Streifen}}$	Breite eines Streifens in Pixel

5 Vgl. Unterabschnitt 4.3.1

begrenzt.

Für die Testbilder entspricht somit die maximale Anzahl von Bildpunkten jeweils 70, da die Streifen eine Breite von ca. 35 Pixeln besitzen und das Muster aus maximal zwei weißen oder schwarzen Streifen besteht. Weiterhin werden für die hellen Pixel nur die 320 hellsten und für die dunklen Pixel die 320 dunkelsten Pixel betrachtet. Dabei entspricht die Hälfte einer Bildzeile 320. So wird verhindert, dass ein Pixel zu den schwarzen und weißen Pixeln zugeordnet wird.

Mit dem Einbezug aller drei Farbkanäle *Y*, *Cb* und *Cr* erfolgt die Auswahl valider Musterpixel in jeder Zeile sehr zuverlässig. Dabei ist die Erkennung unabhängig der Jerseyfarbe, da sich nur an den Farben der Streifen orientiert wird. Aus den erkannten Bildpunkten muss nun ein Muster zusammengesetzt werden.

4.3.3 *Berechnung der Jerseynummer aus erkanntem Muster*

Grundsätzlich ist die Anzahl der gefundenen Pixel höher als die tatsächliche Menge an Musterpixeln, da für die Bilder der Testreihe von zwei schwarzen und zwei weißen Streifen ausgegangen wird. In der Realität besteht das Muster jedoch nur aus drei Streifen. Um die richtigen Bildpunkte zu erkennen, besteht die Möglichkeit, sich an den X-Koordinaten der Pixel zu orientieren. Anhand dieser ist erkennbar, welche Pixel zu weit von der Mehrzahl der Pixel entfernt sind, um noch Teil des Musters zu sein.

Für eine erste Analyse werden die gefundenen Pixel in Bereiche zusammenhängender Bildpunkte unterteilt. Die Breite der einzelnen Streifen schwankt in den Bildern zwischen 15 und 35 Pixeln, da manche Streifen geknickt sind oder sich schräg im Bild befinden. Bereiche kleiner als 15 Pixel werden verworfen. Sind die Felder größer als die maximale Streifenbreite, werden diese auf 35 Pixel gekürzt. Dafür werden alle Pixel erfasst, welche einen Maximalabstand von 17 Pixeln zum Mittelpunkt des jeweiligen Bereiches besitzen.

Sollten nach diesem Schritt mehr als zwei Streifen von einer Farbe übrig bleiben, werden die beiden Bereiche weiter analysiert, bei denen der *Y* Kanal die stärksten Schwankungen besitzt. Infolge der Anwendung des Gaußschen Weichzeichners[6] gleicht sich der Streifenrand der Jerseyfarbe an und der *Y* Kanal bei Randpixeln eines weißen Streifens ist geringer als in der Streifenmitte. Bei schwarzen Streifen ist der Y Wert am Rand höher und fällt zur Mitte hin ab. So schwankt bei validen Streifen die Helligkeit vom Rand zur Mitte stärker als bei einzelnen Bereichen der Jerseyfarbe.

Neben der Streifenbreite dient auch der Abstand als ein Kriterium für die Analyse. Dieser ist ebenfalls nicht einheitlich und nimmt Breiten von 10 bis 25 Pixel an. Für jede Zeile müssen mindestens drei Bereiche in diesem Abstand zueinander liegen, damit das Muster als

6 Vgl. Unterabschnitt 4.2.1

gültig erkannt wird. Sind es weniger als drei, dann gilt die Zeile als invalide. Bei vier Streifen werden die drei Bereiche gewertet, welche am engsten beieinander liegen.

Nach diesem Verfahren liefert jede Zeile entweder drei Streifen zurück, aus denen die Farbe sowie deren Anordnung heraus gelesen wird, oder keine. Wenn ein gefundenes Muster in der Codierungstabelle gelistet ist, wird es in die entsprechende Jerseynummer decodiert. Nachdem alle Zeilen analysiert wurden, wird ausgewertet, wie viele Zeilen eine Nummer erkannt haben, welche am häufigsten vorkommt und wie oft. Damit eine Nummer als eindeutig erkannt gilt, müssen mindestens 50 % der Zeilen eine Nummer erkennen und davon muss die am häufigsten vorkommende Nummer mindestens 50 % ausmachen. Einzig dann gibt der Algorithmus ein eindeutiges Ergebnis für das analysierte Bild zurück. Der vollständige Ablauf zur Bildverarbeitung und Auswertung der Kamerabilder ist in Quelltext 2 im Anhang zu sehen.[7]

7 Zur besseren Vorstellung, wie das fünf streifige Muster später decodiert wird, wird das dreistreifige hier beispielhaft mittels Binärcode in Jerseynummern umgewandelt. Der Algorithmus für fünf Streifen nutzt zur Codierung Tabelle 4. Bedeutsam ist nur, dass die Muster richtig erkannt werden.

5 BEWERTUNG DES ENTWICKELTEN ALGORITHMUS

Im Hauptteil der Abhandlung wurde ein Algorithmus zur Erkennung eines Musters auf dem Roboterjersey entwickelt, der auf gängige Bildverarbeitungsmethoden zurückgreift. Hauptkriterien an die Mustererkennung sind dabei Zuverlässigkeit und Robustheit. [1]

Um den Erfolg des Algorithmus an den in Unterabschnitt 4.1.1 beschriebenen Testreihen zu messen, wurde eine Funktion geschrieben, die in Kopien der Testbilder das erkannte Muster farbig markiert. So lässt sich leicht debuggen, an welchen Positionen der Algorithmus Streifen im Bild detektiert und wo sie tatsächlich liegen. Zusätzlich gibt er jeweils die Anzahl der Zeilen an, in denen eine valide Musterfolge gefunden wurde und wie oft der am häufigsten erkannte Code vorkommt.

Die Tabelle 1 zeigt, wie oft das richtige, falsche oder kein Muster in beiden Testreihen detektiert wurde. Der Algorithmus erkannte in Testreihe 1 das richtige Muster zu 100 % und in Testreihe 2 zu 95 %, was 48 von 50 Bildern entspricht. In der zweiten Testreihe lieferte der Algorithmus in 5 % der Aufnahmen kein eindeutiges Ergebnis. Ein falsches Muster wurde in keinen Aufnahmen gefunden.

Testreihe 1 erfüllt mit diesen Ergebnissen das gesetzte Bewertungskriterium von 100 % für richtig erkannte Codes. Das zeigt, dass der Algorithmus bei optimalen Umweltbedingungen zuverlässig arbeitet und seine Funktion erfüllt. Die zweite Testreihe erreicht den angestrebten Wert knapp nicht. Das Ergebnis ist dennoch akzeptabel, da der Anteil der nicht eindeutigen erkannten Muster gering ist und die Testreihe Bilder bei schwierigen Lichtverhältnissen enthält. Hervorzuheben ist besonders, dass kein falscher Code gefunden wurde. Anhand der Ergebnisse lässt sich schließen, dass der entwickelte Algorithmus zum Großteil robust gegenüber schlechten Umweltfaktoren ist.

Mithilfe der Ergebnisse in Tabelle 2 soll eingeschätzt werden, wie eindeutig der Algorithmus das erkannte Muster gefunden hat. Dafür wurde bestimmt, wie viele Zeilen der Algorithmus pro Testreihe analysierte und wie hoch davon der Prozentsatz der gefundenen Muster sowie des richtigen Musters war. Pro Bild wurden jeweils die unteren 60 Zeilen betrachtet [2], was in 57.200 Zeilen für die erste und in 4000 Zeilen für die zweite Testreihe resultiert.

In Testreihe 1 liegt der Anteil der gefundenen validen Musterfolgen bei ca. 80 %, wobei in insgesamt ca. 76 % das richtige Muster gefun-

1 Vgl. Unterabschnitt 4.1.2
2 Vgl. Unterabschnitt 4.2.2

TESTREIHE	RICHTIGES MUSTER	FALSCHES MUSTER	KEIN MUSTER
1	100 %	0 %	0 %
2	95 %	0 %	5 %

Tabelle 1: Anhand daran, wie oft das richtige Muster in beiden Testreihen erkannt wird, ist die Zuverlässigkeit und Robustheit des Algorithmus einzuschätzen.

TESTREIHE	ANALYSIERTE ZEILEN	GÜLTIGES MUSTER	RICHTIGES MUSTER
1	57.200	80 %	76 %
2	4000	68 %	55 %

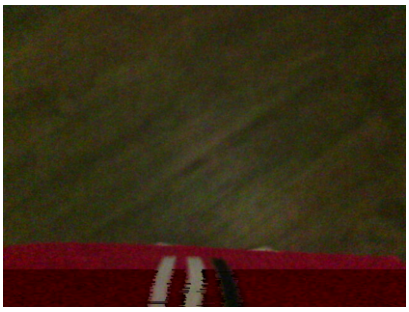
Tabelle 2: Die Eindeutigkeit des Algorithmus ist daran zu messen, wie häufig valide Musterfolgen und das richtige Muster detektiert werden.

den wurde. Das Ergebnis zeigt, dass es sich meistens um die richtige Streifenfolge handelt, wenn der implementierte Algorithmus ein Muster detektiert. Bei der zweiten Testreihe entdeckte der Algorithmus in ca. 68 % der Zeilen eine gültige Musterfolge und zu ca. 55 % der 4000 Zeilen handelte es sich dabei und das richtige Muster. Das Verhältnis ist auch hier noch akzeptabel. Es ist wünschenswert, wenn der Anteil der detektieren Muster höher liegt. Jedoch sind die Streifen in manchen Testzeilen selbst für einen Menschen kaum zu erkennen, sodass die Ergebnisrate im Erwartungshorizont liegt. Bedeutsam ist, dass auch in der zweiten Testreihe der Anteil richtig erkannter Muster nahe dem Anteil der gefundenen Muster ist. Demzufolge liefert der Algorithmus im Allgemeinen ein sicheres Ergebnis.

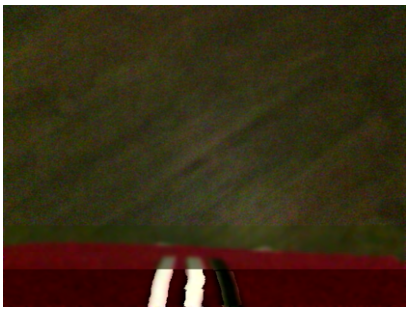
Weiterhin lässt sich mithilfe der eingefärbten Testbilder die Effektivität des Gaußschen Weichzeichner sowie der Helligkeitsanpassung überprüfen. Wie in Abbildung 19 dargestellt, wird das Muster in den verarbeitenden Bildern deutlich besser erkannt als in den Originalaufnahmen. So tragen beiden Verfahren erheblich zur Qualitätsverbesserung der Aufnahmen bei.

Bei einer Laufzeit von 600 ms und einer geringen Speicherauslastung arbeitet der Algorithmus schnell und ressourcensparend. Beide Faktoren sind für den Einsatz des Programmes von geringerer Bedeutung, dennoch sind die Ergebnisse als sehr komfortabel zu betrachten. So werden die Anwendungsmöglichkeiten nicht eingeschränkt.

Anhand der erfolgreichen Tests fällt die Bewertung des Algorithmus positiv aus. Zum einen eignen sich die angewandten Bildverarbeitungsmethoden zur Mustererkennung auf dem Jersey. Anderer-



(a) Originalaufnahme pinkes Jersey bei dunkleren Lichtverhältnissen



(b) Korrekt erkanntes Muster

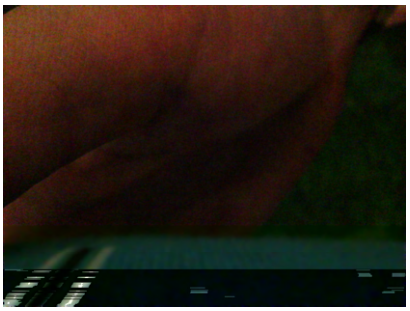
(c) Originalaufnahme blaues Jersey bei dunkleren Lichtverhältnissen

(d) Korrekt erkanntes Muster

Abbildung 19: Die Abbildungen zeigen die Muster, welche von dem Algorithmus erkannt werden. Dabei wird dies nach Anwendung des Gaußschen Weichzeichners und der Helligkeitsanpassung viel besser detektiert.

seits läuft die algorithmische Analyse ebenfalls zuverlässig, robust und eindeutig ab, welches den Anforderungen gerecht wird. Die Beispielbilder in Abbildung 20, Abbildung 21 und Abbildung 22 demonstrieren die aktuelle Erkennungsleistung des Algorithmus und zeigt deren Stärken und Schwächen auf. Die Bilder auf der linken Seite sind jeweils Originalaufnahmen und auf der rechten Seite befinden sich die dazugehörigen bearbeitenden Aufnahmen, in denen das erkannte Muster eingefärbt ist.

(a) Originalaufnahme bei dunkleren Lichtverhältnissen mit Muster am Rand

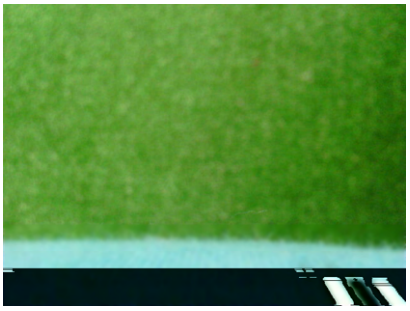


(b) Korrekt erkanntes Muster in der verarbeitenden Aufnahme.

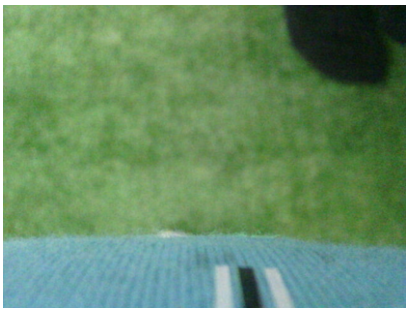
(c) Originalaufnahme bei dunkleren Lichtverhältnissen ohne Muster.

(d) Korrekt erkanntes Muster in der verarbeitenden Aufnahme.

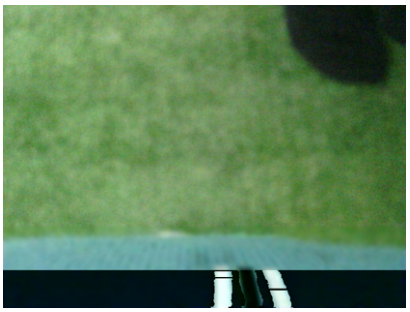
(e) Originalaufnahme bei günstigen Lichtverhätnissen mit Muster am Rand.



(f) Korrekt erkanntes Muster in der verarbeitenden Aufnahme.

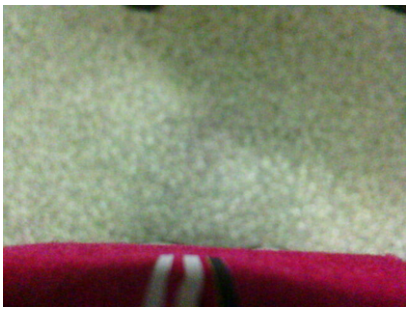


(g) Originalaufnahme bei günstigen Lichtverhätnissen.

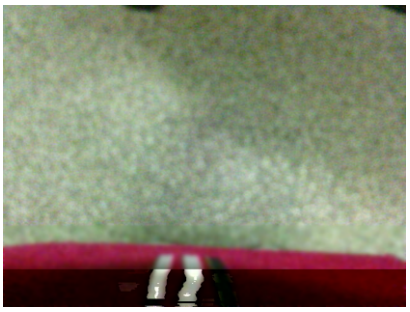


(h) Korrekt erkanntes Muster in der verarbeitenden Aufnahme.

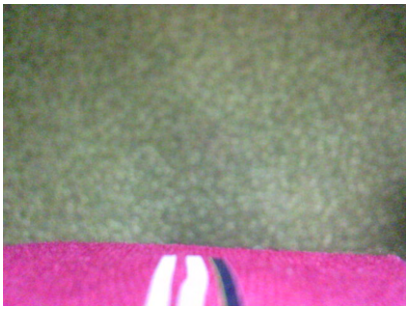
Abbildung 20: Anwendung des Algorithmus auf Originalbilder mit blauem Jersey.



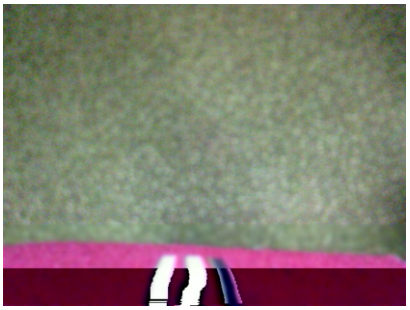
(a) Originalaufnahme bei Halbschatten.



(b) Korrekt erkanntes Muster in der verarbeitenden Aufnahme.

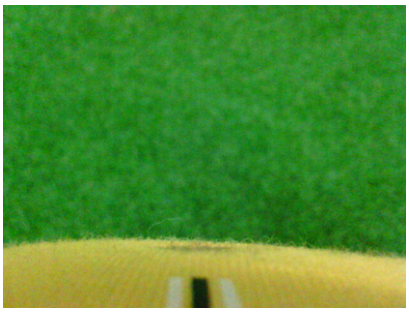


(c) Originalaufnahme bei Taschenlampenlicht.



(d) Korrekt erkanntes Muster in der verarbeitenden Aufnahme.

Abbildung 21: Anwendung des Algorithmus auf Originalbilder mit pinkem Jersey.



(a) Originalaufnahme von gelbem Jersey bei helleren Lichtverhältnissen.

(b) Korrekt erkanntes Muster in der verarbeitenden Aufnahme.

Abbildung 22: Anwendung des Algorithmus auf Originalbilder mit gelbem Jersey.

6

ZUKÜNFTIGE MÖGLICHKEITEN DER PRAKTISCHEN ANWENDUNG AUF DEM NAO

Der in dieser Arbeit entwickelte Algorithmus erkennt das Jerseymuster anhand der Analyse von Farbbereichen und -übergängen zuverlässig und robust. Grundlage für den Erfolg bildet zum einen die Verarbeitung der Bilder mittels Gaußschen Weichzeichners und der Verbesserung des Kontrastes über funktionelle Helligkeitsanpassung. Andererseits trägt auch das Design des Musters dazu bei, dass auch bei kritischeren Umweltbedingungen in der Kamera des *Naos* ausreichend zu sehen ist.

Es ist zu erwarten, dass die Regeln des *RoboCups* bezüglich der Belichtung in Zukunft verschärft werden, um das gemeinsame Ziel zu erreichen, 2050 die Menschen im Fußball zu schlagen. So ist es denkbar, dass in den nächsten Jahren nur auf Tageslichtfeldern gespielt wird. Demzufolge liegt eine Optimierung des Algorithmus bezüglich seiner Zuverlässigkeit nahe, da diese gerade unter schwierigeren Lichtverhältnissen nicht 100 % beträgt.

Eine Möglichkeit hierfür ist die Betrachtung der X-Koordinaten der gefundenen Streifen über die Zeilen hinweg. So wäre erkennbar, ob die identifizierten Streifen im Gesamtbild ein einheitliches Muster ergeben. Weiterhin könnte an der Verbesserung des Kontrastes gefeilt werden, indem zum Beispiel auf Bezierfunktionen statt lineare Funktionen zurückgegriffen wird. Auch der *Nao* bietet technische Optionen, die Zuverlässigkeit zu verbessern. Beispielsweise kann er den Kopf drehen, wenn er kein Jersey sieht oder sich drehen, wenn ungünstiger Halbschatten ins Kamerabild fällt.

Nach erfolgreicher Integration des Algorithmus in die bestehende Software der *HTWK Robots* wird die automatische Erkennung der Jerseynummer deutlich den Stress vor den Spielen verringern. Die Kenntnisse über die eigene Jerseyfarbe, die die *Naos* bei der Mustererkennung erlangen, kann als Grundlage für weitere Analysen dienen. So besteht die Möglichkeit, dass diese zur Unterscheidung von Mitspielern und gegnerischen Robotern beiträgt. Auch könnte der Algorithmus als Grundlage für die Erkennung der Nummern anderer Spieler dienen. Jedoch ist dieses Wissen irrelevant, da die Jerseynummer nur für die eigene Kommunikation mit dem GameController von Bedeutung ist.

Solange der *Nao* als Standardplattform in der SPL eingesetzt wird, ist die Verwendung der automatischen Identifizierung der Jerseynummer möglich. Der Algorithmus ist bis auf wenige gängige Einstellungen unabhängig von der Kamera, sodass diese auch auf potenziell

andere Kamerasysteme in zukünftigen Versionen des *Naos* anwendbar wäre.

Die Automatisierung der Erkennung der Jerseynummer ist bislang einmalig in der SPL und könnte andere Teams als Inspiration dienen, ebenfalls ein entsprechendes Verfahren zu entwickeln.

Teil II

ANHANG

A

ABBILDUNGEN

A.1 AKTUELLE JERSEYS DER HTWK ROBOTS

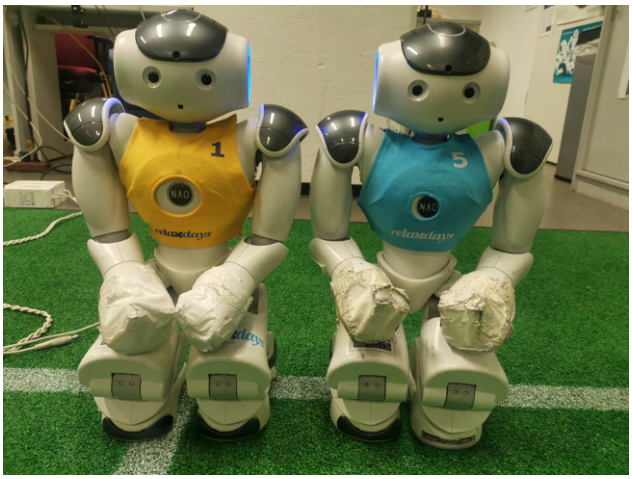


Abbildung 23: Seit einigen Jahren tragen die *HTWK Robots* standardmäßig blaue Heimjerseys und gelbe Auswärtsjerseys. Die Farben repräsentieren die Stadtfarben von Leipzig.

A.2 HISTOGRAMM DES Y KANALS DER TESTREIHE 1

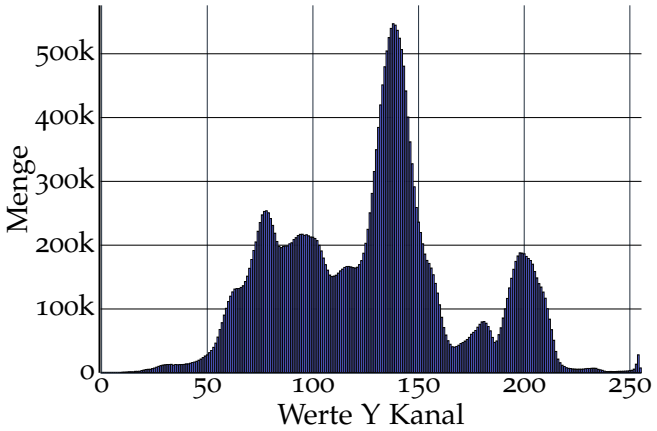


Abbildung 24: Das Histogramm stellt die Y Werte aller 715 Bilder aus Testreihe 1 dar. Am häufigsten existiert ein Helligkeitswert von 139 bei ca. 544000 Pixeln.

A.3 CLUSTER VERSCHIEBUNGSPUNKTE BEI HELLIGKEITSANPASSUNG

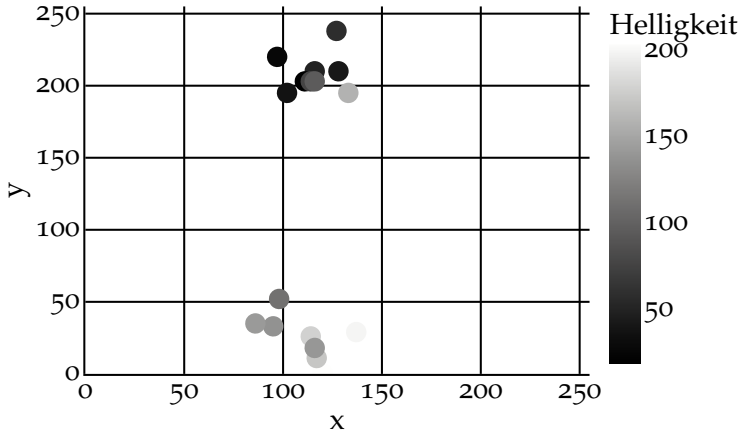


Abbildung 25: Mit den Verschiebungspunkten der linearen Funktion wurde die Helligkeit in einzelnen Bildern angepasst. Die Farbe der Punkte repräsentiert die durchschnittliche Helligkeit der Ausgangsbilder.

B

TABELLEN

B.1 HARDWAREEINSTELLUNGEN DER KAMERA DES NAOS

Die gelisteten Kameraeinstellungen in Tabelle 3 können modifiziert werden, um die Wahrnehmung des *Naos* anzupassen. Für die Erkennung des Musters wurde der automatische Fokus deaktiviert und manuell auf einen Wert von 200 gesetzt. Außerdem wurde die Helligkeit auf 255 erhöht. Die übrigen Parameter verbleiben bei ihren Standardwerten.

B.2 MÖGLICHE MUSTERFOLGEN ZUR CODIERUNG DER JERSEY-NUMMERN

Die Tabelle 4 zeigt mögliche Musterfolgen, in denen zur besseren Fehlererkennung nur Codes gelistet sind, die entweder dreimal Schwarz und zweimal Weiß oder umgekehrt enthalten.

PARAMETER	MIN VALUE	MAX VALUE	DEFAULT VALUE
Brightness	-255	255	0
Contrast	0	255	32
Saturation	0	255	64
Hue	-180	180	0
Gain (Read only if auto exposure enabled)	0	1023	16
Horizontal Flip	0	1	0
Vertical Flip	0	1	0
Auto Exposition (0: disabled 1: enabled)	0	1	1
Auto White Balance (0: disabled 1: enabled)	0	1	1
Exposure (Read only if auto exposure enabled)	0	65535	32
Sharpness	0	9	4
Backlight Compensation (0: disabled 1: enabled)	0	1	0
Average Luminance (Read Only)	0	255	NA
Auto Focus (0: disabled 1: enabled)	0	1	0
Manual Focus (only when Auto Focus disabled)	0	250	0

Tabelle 3: Hardwareeinstellungen der Kameras. [23]

MUSTERFOLGE	JERSEYNUMMER
WSSSW	1
WSSWS	2
WSWSS	3
WWSSS	4
SWSSW	5
SWSWS	6
SWWSS	7
SSWSW	8
SSWWS	9
SSSWW	10
SWWWS	11
SWWSW	12
SWSWW	13
SSWWW	14
WSWWS	15
WSWSW	16
WSSWW	17
WWSWS	18
WWSSW	19
WWWSS	20

Tabelle 4: Mögliche Musterfolgen zur Codierung der Jerseynummern mit S = Schwarz und W = Weiß.

C

QUELLTEXTE

C.1 GAUSSSCHE WEICHZEICHNER

Der Gaußsche Weichzeichner verbessert die Qualität der Kamerabilder, da er die Kanten glättet und einen fließenderen Übergang zwischen Muster- und Jerseyfarben kreiert. Der Algorithmus verändert dabei die Werte der Kanäle *Y*, *Cb* und *Cr* gleichermaßen.

```
/*
 * Gaussian Algorithm to blur lower part of image
 * @param img   image to be processed
 * @return      processed image
 */
uint8_t *JerseyNumberDetector::blurImg(uint8_t *img) {
    uint8_t *imgNew = (uint8_t *)malloc(sizeof(uint8_t) * 640 *
        480 * 2);
    memcpy(imgNew, img, sizeof(uint8_t) * 640 * 480 * 2);
    double gaussianBlur[5][5] = {{1.0, 4.0, 6.0, 4.0, 1.0},
                                 {4.0, 16.0, 24.0, 16.0, 4.0},
                                 {6.0, 24.0, 36.0, 24.0, 6.0},
                                 {4.0, 16.0, 24.0, 16.0, 4.0},
                                 {1.0, 4.0, 6.0, 4.0, 1.0}};

    for (int32_t y = 350; y < 480 - 2; y++) {
        for (int32_t x = 2; x < 640 - 2; x++) {
            double cy = 0;
            double cr = 0;
            double cb = 0;

            for (int i = -2; i < 3; i++) {
                for (int j = -2; j < 3; j++) {
                    double gauss = gaussianBlur[i + 2][j + 2] /
                        256.0;
                    cy += getY(img, x + j, y + i) * gauss;
                    cb += getCb(img, x + j, y + i) * gauss;
                    cr += getCr(img, x + j, y + i) * gauss;
                }
            }
            setYCbCr(imgNew, x, y, (uint8_t)cy, (uint8_t)cb, (
                uint8_t)cr);
        }
    }
    return imgNew;
}
```

Quelltext 1: Implementierung des Gaußschen Weichzeichners in C++.

C.2 VOLLSTÄNDIGER ALGORITHMUS ZUR ERKENNUNG DER JERSEYNUMMER

Der Algorithmus setzt sich aus folgenden Punkten zusammen:

- Anwendung des Gaußschen Weichzeichners
- Helligkeitsanpassung mittels linearer Funktion
- Herausfiltern der Streifenpixel anhand geeigneter Kriterien an deren *Y* Wert, *Cb* Wert und *Cr* Wert und deren Position in der Zeile
- Umwandlung des gefundenen Musters, wenn möglich, in die Jerseynummer
- Analyse der Häufigkeit gefundener Jerseynummern

```
/**
 * @brief detects the jerseynumber by a pattern of 3 stripes on
     the jersey
 * @param img   image to be processed
 * @return      jerseynumber or error why no number was detected
 */
int JerseyNumberDetector::proceed(uint8_t *img) {
    std::array<std::array<int, 2>, 6> foundNumbers = {{{1, 0},
        {2, 0}, {3, 0}, {4, 0}, {5, 0}, {6, 0}}};
    int finalNumber = 0;
    std::array<std::string, 6> codes = {"bbw", "bwb", "bww", "wbb
        ", "wbw", "wwb"};
    std::string finalCode = "";
    int validNumbersFound = 0;

    uint8_t *imgProcessed = blurImg(img);
    for (int i = 0; i < 6; i++) {
        uint8_t *imgBlurredTemp = blurImg(imgProcessed);
        imgProcessed = imgBlurredTemp;
    }

    imgProcessed = contrastImg(imgProcessed);
    for (int y = begin_jersey; y < height; y++) {
        // finds stripes and sorts them by their x coordinate
        std::vector<std::array<int, 3>> black_white_stripes =
            detect_stripes(imgProcessed, y);
        int foundNumber = 0;
        if (black_white_stripes.size() == 3) {
            foundNumber = 4 * black_white_stripes[0][2] + 2 *
                black_white_stripes[1][2] + black_white_stripes
                [2][2];

            if (foundNumber > 0 && foundNumber < 7) {
                foundNumbers[foundNumber - 1][1]++;
                validNumbersFound++;
```

```
            }
        }
    }

    // selects code occured the most
    int countNumber = 0;
    for (auto number : foundNumbers) {
        if (number[1] > countNumber) {
            countNumber = number[1];
            finalNumber = number[0];
        }
    }

    // No number detected in any line
    if(finalNumber > 0 && finalNumber < 7) {
        finalCode = codes[finalNumber - 1];
    } else {
        std::cout << "No code detected.\n\n";
        return -1;
    }

    // Not enough codes found
    if (validNumbersFound <= (height - begin_jersey) / 2) {
        std::cout << "Only " << validNumbersFound << " out of "
            << height - begin_jersey << " numbers found. Most
            detected code \"" << finalCode << "\" only appeared "
             << countNumber << " times.\n\n";
        return -2;
    // enough codes but to many different one
    } else if (countNumber <= validNumbersFound / 2) {
        std::cout << validNumbersFound << " numbers were detected
             in " << height - begin_jersey << " lines.
            Unfortunately, the best code \"" << finalCode << "\"
            appeared only " << countNumber << " times.\n\n";
        return -3;
    }

    std::cout << "The Finalcode is " << finalCode << " which
        appeared " << countNumber << " times in " << height -
        begin_jersey << " lines. Altogether " <<
        validNumbersFound << " valid codes were found.\n\n";
    return finalNumber;
}
```

Quelltext 2: Vollständiger Algorithmus zur Erkennung der Jerseynummer bei einem dreistreifigen Muster in C++.

[1] *A Brief History of RoboCup*. URL: https://www.robocup.org/a_brief_history_of_robocup (besucht am 7. Aug. 2023).

[2] *About RoboCup Small Size League*. URL: https://ssl.robocup.org/about/ (besucht am 7. Aug. 2023).

[3] W. Burger und M. J. Burger. *Digital Image Processing. An Algorithmic Introduction*. Springer, 2008, S. 400–402.

[4] L. Day. *What exactly is a gaussian blur?* URL: https://hackaday.com/2021/07/21/what-exactly-is-a-gaussian-blur/ (besucht am 20. Aug. 2023).

[5] S. Eswar. "Noise reduction and image smoothing using gaussian blur". California State University Northridge, 2015.

[6] *Getting to Grips with Chroma Subsampling*. URL: https://www.cined.com/tag/ycbcr/ (besucht am 5. Sep. 2023).

[7] R. Grbić und B. Koch. "Automatic vision-based parking slot detection and occupancy classification". In: *Expert Systems with Applications* 225 (2023), S. 120147.

[8] *HTWK Robots*. URL: https://robots.htwk-leipzig.de (besucht am 7. Aug. 2023).

[9] W. A. Hemmerich. *StatistikGuru. Winsorizing-Rechner*. URL: https://statistikguru.de/rechner/winsorizing-rechner.html (besucht am 24. Aug. 2023).

[10] *Home of the MSL*. URL: https://msl.robocup.org/ (besucht am 8. Aug. 2023).

[11] A. Iwainsky und W. Wilhelmi. "Lexikon: Der Computergrafik und Bildverarbeitung". In: Wiesbaden: Vieweg+Teubner Verlag, 1994, S. 251.

[12] *NAO*. URL: https://www.aldebaran.com/en/nao/ (besucht am 12. Aug. 2023).

[13] M. Querini. "Reliability and Data Density in High Capacity Color Barcodes". In: *Computer Science and Information System* 11 (2014), 1595–1615.

[14] M. S. T. Reinhardt. "Kalibrierungsfreie Bildverarbeitungsalgorithmen zur echtzeitfähigen Objekterkennung im Roboterfußball". HTWK Leipzig, 2011.

[15] *Relation Between Cb And Cr Channels In The YCbCr Color Space*. URL: https://www.researchgate.net/profile/German-Sanchez-16/publication/262777844/figure/fig2/AS:668944457269254@1536500195537/relation-between-cb-and-cr-channels-in-the-Ycbcr-color-space.png (besucht am 5. Sep. 2023).

[16] *RoboCup Humanoid League*. URL: https://humanoid.robocup.org/ (besucht am 8. Aug. 2023).

[17] *RoboCup Objective*. URL: https://www.robocup.org/objective (besucht am 7. Aug. 2023).

[18] *RoboCup Standard Platform League (NAO) Rule Book. 2023 rules, as of 2023-06-01*. URL: https://spl.robocup.org/wp-content/uploads/SPL-Rules-2023.pdf (besucht am 8. Aug. 2023).

[19] *RoboCupSoccer - Simulation*. URL: https://www.robocup.org/leagues/23 (besucht am 8. Aug. 2023).

[20] T. Saba. "Computer vision for microscopic skin cancer diagnosis usinghandcrafted and non-handcrafted features". In: *Micrroscopy Research and Technique* 84 (2021), S. 1272–1283.

[21] *Signal-Rausch-Verhältnis, Signal-Rausch-Abstand. Definition*. URL: https://www.computerweekly.com/de/definition/Signal-Rausch-Abstand-S-N-oder-SNR (besucht am 20. Aug. 2023).

[22] *Softbank Robotics Documantation - NAO. Technical Overview*. URL: http://doc.aldebaran.com/2-8/family/nao_technical/index_dev_naov6.html (besucht am 12. Aug. 2023).

[23] *Softbank Robotics Documantation - NAO. Video Cameras*. URL: http://doc.aldebaran.com/2-8/family/nao_technical/video_naov6.html (besucht am 12. Aug. 2023).

[24] *Standard Platform League History*. URL: https://spl.robocup.org/history/ (besucht am 8. Aug. 2023).

[25] *Strichcode-Mechanismen*. URL: https://www.keyence.de/ss/products/auto_id/codereader/basic/mechanism.jsp (besucht am 30. Aug. 2023).

[26] P. Thai, S. Alam, N. Lilith und B. T. Nguyen. "A computer vision framework using Convolutional Neural Networks for airport-airside surveillance". In: *Transportation Research Part C: Emerging Technologies* 137 (2022), S. 103590.

[27] *Was ist ein Data Matrix - Code?* URL: https://www.keyence.de/ss/products/auto_id/codereader/basic_2d/datamatrix.jsp (besucht am 17. Aug. 2023).

[28] *Was ist ein QR-Code?* URL: https://www.keyence.de/ss/products/auto_id/codereader/basic_2d/qr.jsp (besucht am 30. Aug. 2023).

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/