



Karlsruhe Institute of Technology



# Gegenstandserkennung und kategoriebasierter Transport anhand von kameraunterstützten NXT-Robotern

Studienarbeit

für die Prüfung zum  
Bachelor of Engineering

von

Sebastian Hüther & Lorenzo Toso

11. Mai 2015

**Bearbeitungszeitraum:** 2 Semester

**Matrikelnummer:** 8853105 & 1906813

**Kurs:** TINF12B3

**Studiengang:** Informationstechnik

**Ausbildungsfirma:** Karlsruher Institut für Technologie

**Betreuer:** Gertrud Nieder

## **Eidesstattliche Erklärung**

Gemäß § 5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011.  
Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen  
Quellen und Hilfsmittel verwendet.

---

Ort, Datum

Unterschrift

---

Ort, Datum

Unterschrift

# Abstract

## **Deutsch:**

In dieser Studienarbeit wurde ein Robotersystem entwickelt, dessen Ziel es ist kameragestützt Objekte zu erkennen. Der Roboter fährt die Gegenstände an, kategorisiert diese und transportiert sie abhängig von der Kategorisierung in eine Zielzone. Für die Realisierung wurde ein LEGO Mindstorms NXT Roboter und ein Google Nexus 5 Smartphone verwendet. Die Objekterkennung wurde mit bekannten Verfahren aus der Bildverarbeitung mit Hilfe der OpenCV-Bibliothek implementiert. Der Roboter orientiert sich im Raum mit Hilfe einer Positionsverfolgung über die gefahrene Strecke. Unter Verwendung dieser Methoden konnte ein System entworfen werden, welches einen Gegenstand in weniger als einer Minute verräumen kann.

## **English:**

In this student research project a robotic system was developed, which has the goal to recognize objects with an optical camera module. The robot approaches an object, categorizes it and transports it to a target location. Which target zone it is transported to depends on the categorization. The robot was built using a LEGO Mindstorms NXT kit and a Google Nexus 5 Smartphone. The object recognition algorithm was realized using methods known in image processing with the aid of the OpenCV library. The robot orients itself by tracking its own movement and the traveled distance. Using these techniques, a system could be developed which is able to rearrange an object in less than one minute.

# Inhaltsverzeichnis

<b>Eidesstattliche Erklärung</b>	<b>II</b>
<b>Abstract</b>	<b>III</b>
<b>1 Einleitung</b>	<b>3</b>
<b>2 Problemstellung</b>	<b>4</b>
2.1 Objekterkennung . . . . .	4
2.2 Mechanische Umsetzung des Objekttransports . . . . .	5
2.3 Kategorisierung und Wahl der Zielzone . . . . .	5
2.4 Transport in die Zielzone . . . . .	5
2.5 Bewertung des Ablaufs . . . . .	6
<b>3 Materialien und Methoden</b>	<b>7</b>
3.1 Wahl des Robotersystems . . . . .	7
3.2 Wahl des Kameramoduls . . . . .	12
<b>4 Hardwareumsetzung</b>	<b>14</b>
4.1 Anforderungen an den NXT-Roboter . . . . .	14
4.2 Entwurf des Roboters . . . . .	14
4.3 Steuerung des Roboters . . . . .	21
<b>5 Softwareumsetzung</b>	<b>24</b>
5.1 Wahl der Bildverarbeitungsbibliothek . . . . .	24

## Inhaltsverzeichnis

V

5.2	Algorithmen zur Objekterkennung . . . . .	25
5.3	Ortsbestimmung . . . . .	27
5.4	Struktur der Applikation . . . . .	29
<b>6</b>	<b>Arbeitsablauf und Problematiken</b>	<b>35</b>
6.1	Objektsuche . . . . .	35
6.2	Anfahren von Objekten . . . . .	42
6.3	Aufnehmen von Objekten . . . . .	47
6.4	Kategorisierung von Objekten . . . . .	47
6.5	Suche des Zielbereichs . . . . .	50
6.6	Ansteuerung des Zielbereichs . . . . .	50
6.7	Ablegen von Objekten . . . . .	51
6.8	Rückkehr zur Ausgangsposition . . . . .	51
<b>7</b>	<b>Tests des Robotersystems</b>	<b>53</b>
7.1	Realtests . . . . .	53
7.2	Manuelle Steuerung . . . . .	54
<b>8</b>	<b>Zusammenfassung und Ausblick</b>	<b>56</b>
8.1	Zusammenfassung . . . . .	56
8.2	Bewertung der Ergebnisse . . . . .	57
8.3	Ausblick . . . . .	58
8.4	Reflexion des Gelernten . . . . .	59
<b>Literaturverzeichnis</b>		<b>VI</b>

# Abbildungsverzeichnis

3.1	Hauptbestandteile des LEGO Mindstorms NXT-Systems . . . . .	8
3.2	Hauptbestandteile des LEGO Mindstorms EV3-Systems . . . . .	10
3.3	Beispiel-Aufbau des Fischertechnik . . . . .	11
3.4	Kameramodul des Nexus 5 . . . . .	13
4.1	Von Lego bereitgestellter Roboteraufbau . . . . .	15
4.2	Bild des fertig modifizierten Roboters . . . . .	16
4.3	NXT-Stein . . . . .	17
4.4	Tastsensor des NXT-Systems . . . . .	17
4.5	Farbsensor des NXT-Systems . . . . .	18
4.6	Ultraschallsensor des NXT-Systems . . . . .	19
4.7	Servo-Motor des NXT-Systems . . . . .	19
4.8	Greifarm des Robotersystems . . . . .	20
4.9	Bluetooth-Verbindung zwischen NXT und Nexus 5 . . . . .	21
5.1	Beispiel einer Aufspaltung in RGB-Kanäle (links) und HSV-Kanäle (rechts) . .	26
5.2	Beispiel der Binarisierung des Sättigungskanals . . . . .	26
5.3	Beispiel einer Segmentierung des binarisierten Sättigungskanals . . . . .	27
5.4	UML-Klassendiagramm der Grundstruktur der Applikation . . . . .	30
5.5	UML-Klassendiagramm der für die Auffindung eines Fokusobjekts verantwortlichen Klassen . . . . .	31
5.6	UML-Klassendiagramm der Kategorisierung von Fokus Objekten . . . . .	32
5.7	UML-Klassendiagramm der Bewegungssteuerung und Positionsverfolgung . .	33

6.1	Flussdiagramm des Arbeitsablaufs . . . . .	36
6.2	Auffindung eines Fokusobjekts . . . . .	37
6.3	Screenshot bei Objektfindung . . . . .	38
6.4	Objektverfolgung durch Kriterium der lokalen Nähe . . . . .	39
6.5	Objektverfolgung durch Größenkriterium . . . . .	39
6.6	Objektverfolgung durch farbliche Ähnlichkeit . . . . .	40
6.7	Verzerrte Aufnahme durch Bewegung des Roboters und Erkennungsverlust nach Helligkeitsänderung . . . . .	42
6.8	Zentrierung eines Objekts . . . . .	42
6.9	Stereokamerabasierte Entfernungsschätzung . . . . .	44
6.10	Aufbau der monokamerabasierten Entfernungsschätzung . . . . .	45
6.11	Exemplarische monokamerabasierte Entfernungsschätzung . . . . .	46
6.12	Aufnahme aus der monokamerabasierten Entfernungsschätzung . . . . .	47
6.13	Approximation von Objektkanten über Punktlisten . . . . .	49
6.14	Bestimmung von Objekten über Winkel zwischen Kantenpunkten . . . . .	49
6.15	Berechnung der X- und Y-Anteile und des Winkels bei Fahrt und Rotation des Roboters im Raum . . . . .	51
7.1	Manuelle Steuerung des Roboters . . . . .	55

# 1 Einleitung

Im Rahmen der Studienarbeit des fünften und sechsten Semesters der Prüfung zum Bachelor of Engineering, stellt diese Arbeit eine Dokumentation zur Entwicklung eines kameragestützten Roboters dar. Ziel der Arbeit ist es mit Hilfe eines Android-Smartphones und eines LEGO Mindstorm NXT-Kits einen Roboter zu entwerfen, der Gegenstände in einem Raum erkennt, anfährt und in eine vordefinierte Zielzone transportiert. Hierfür werden diverse Methoden der Bildverarbeitung eingesetzt, welche unter der Verwendung der OpenCV-Bibliothek [1] implementiert werden. Das Projekt wurde „CLEEN-R“ getauft. Der Name stellt hierbei ein Akronym für „Cleaning and Lifting Environment Evaluating NXT - Robot“ und damit eine Anspielung an den Disney-Film „WALL-E“ dar.

Das nachfolgende Kapitel beschreibt die Problemstellung und erklärt eine Grundproblematik der Zusammenarbeit der beiden Hardwaremodule. Kapitel drei beinhaltet genaue Daten zu den Hardwaremodulen und deren Zusammenspiel. Das Kapitel Hardwareumsetzung beinhaltet gesondert den Aufbau des Roboters, sowie die Ansteuerung einzelner Aktoren und Sensoren. Notwendige Grundlagen der Bildverarbeitung, sowie genaue Algorithmen zur Objekterkennung und genauer Aufbau der Applikation werden im darauf folgenden Kapitel Softwareumsetzung beschrieben. Den Kern der Arbeit stellt das Kapitel Arbeitsablauf und Problematiken dar. Es beinhaltet den genauen Arbeitszyklus des Roboters und beschreibt Problematiken, die während der Entwicklung aufraten, sowie Designentscheidungen zur Lösung dieser. Kapitel 7 beschreibt durchgeführte Tests sowohl unter speziell präparierten, als auch unter Realbedingungen. Das abschließende Kapitel ist ein letztes Fazit, welches einen Überblick über die gesamte Arbeit bildet und einen Ausblick über aufbauende Arbeiten liefert.

## 2 Problemstellung

Ziel dieser Studienarbeit ist die praktische Anwendung gelernter Kenntnisse in Hard- und Software. Durch die Benutzung zweier getrennter Module, sind sowohl Kenntnisse in der Programmierung, als auch der Prozessautomatisierung und in der Entwicklung verteilter Systeme erforderlich.

Konkrete Aufgabenstellung ist es einen kameragestützten Roboter zu entwickeln, der autonom Gegenstände in einem Raum erkennt und in definierte Zielzonen transportiert. Der Prozess kann hierbei in vier Teilprozesse unterteilt werden, welche in den folgenden Abschnitten näher beschrieben sind. Bewertungskriterien für den Arbeitsablauf des Roboters werden in Abschnitts 2.5 näher geschildert.

### 2.1 Objekterkennung

Der Roboter mit Hilfe eines Kameramoduls und Verfahren aus der Bildverarbeitung Objekte auf Grund ihrer physikalischen Beschaffenheit erkennen. Zu den Kriterien gehören beispielsweise Größe, Form und Farbe der Objekte. Unter den erkannten Objekten muss zielgerichtet ein geeignetes als „Fokusobjekt“ ausgewählt werden, damit dieses im folgenden Schritt angefahren werden kann.

## 2.2 Mechanische Umsetzung des Objekttransports

Der Roboter hat das Ziel das ausgewählte Fokusobjekt zu transportieren. Hierfür muss sich der Roboter gegenüber des Gegenstands optimal positionieren. Dies geschieht über einen beliebig umgesetzten motorisierten Antrieb. Hat der Roboter eine angemessene Entfernung zum Fokusobjekt erreicht, muss er dieses mit Hilfe eines mechanischen Greifarms aufnehmen und für den weiteren Transport sicher halten können.

## 2.3 Kategorisierung und Wahl der Zielzone

Das aufgenommene Objekt muss unterscheidbar von anderen Objekten kategorisiert werden. Eine solche Kategorisierung kann nach den in Abschnitt 2.1 beschriebenen Kriterien der Form, Farbe und Größe erfolgen. Je nach Kategorie des Objektes, soll eine passende Zielzone gewählt werden.

## 2.4 Transport in die Zielzone

Zuletzt muss der Roboter den aufgenommenen und kategorisierten Gegenstand in die gewählte Zielzone transportieren. Hierfür muss der Zielbereich ermittelt werden. Dies geschieht entweder, ähnlich der Objektsuche, über das Kameramodul, oder über eine vorhergehende Definition der Zielzonen. Der Transport kann hierbei, wie bereits in Abschnitt 2.2 beschrieben, funktionieren.

## 2.5 Bewertung des Ablaufs

Als Bewertungskriterien dienen hierbei beispielsweise ob der Roboter alle Gegenstände erfolgreich erkannt und kategorisiert hat. Auch das korrekte Zuordnen zu Zielzonen kann als Kriterium dienen. Weiterhin kann die mechanische Umsetzung des Transports und damit verbundene mögliche Verluste von Objekten bewertet werden. Räumt der Roboter alle Gegenstände korrekt auf sind die dafür benötigte Zeit, die Sorgfalt und die „Flüssigkeit“ der Bewegungen die zutreffendsten Kriterien.

# 3 Materialien und Methoden

## 3.1 Wahl des Robotersystems

Als Materialien stehen für das Projekt diverse Hardwaremodule zur Verfügung. Um die Problemstellung zu erfüllen ist zunächst ein mobiler Roboter notwendig. Für schnelle Prototypen-Erstellung bieten sich die Roboter-Baukästen von LEGO an.

### 3.1.1 LEGO Mindstorms

LEGO Mindstorms ist eine Produktreihe des Spielzeugherstellers LEGO, die bereits in drei Generationen erschienen ist. Sie zeichnen sich durch einen programmierbaren Hauptstein aus, der mit modularen Sensoren und Aktoren erweitert und in LEGO Technik-Aufbauten eingesetzt wird.

#### LEGO Mindstorms NXT

Der NXT-Bausatz besteht wie alle Mindstorms-Reihen aus einem intelligenten Baustein als Recheneinheit und Kernstück des Roboters. Er beinhaltet einen 32-Bit ARM7-Prozessor (48 MHz) und einen 8-Bit Atmega-Mikrocontroller als Koprozessor zur Ausführung der über die USB-Schnittstelle übertragenen Programme.

Programmieren kann man den NXT über die von LEGO bereitgestellte graphische Entwicklungsumgebung NXT-G, was für Einsteiger den einfachsten Weg darstellt. Jedoch existieren



Abbildung 3.1: Hauptbestandteile des LEGO Mindstorms NXT-Systems

auch für zahlreiche weitere Programmiersprachen wie C, Java, C#, Matlab, Lua oder Ruby APIs zum Ansteuern der verschiedenen Funktionen.

An den Hauptstein können bis zu drei Motoren und bis zu vier Sensoren angeschlossen werden. Kommuniziert wird hierbei im I<sup>2</sup>C-Protokoll. Die Servomotoren besitzen einen eingebauten Rotationssensor, die sich über den Rückkanal beim NXT-Stein melden.

Statusmeldungen werden über ein monochromes 100 mal 64 Pixel LCD-Display ausgegeben, auch bietet der Hauptstein einen eingebauten Lautsprecher und ein Bluetooth-Modul.

Als Sensoren sind eine große Auswahl von LEGO erhältlich:

- Tastsensor
- Ultraschallsensor
- Lichtsensor
- Schallsensor

- RFID-Sensor
- Infrarot-Sensor
- Beschleunigungssensor

Darüber hinaus können über Adapterkabel viele weitere Aktoren und Sensoren die das I<sup>2</sup>-Protokoll beherrschen (unter anderem die des älteren RCX-Systems) angeschlossen werden, was fast keine Bedürfnisse beim Roboterbau offen lässt.

Das Gerüst bilden wie bei allen LEGO-Systemen die tausende verschiedenen Bauteile von LEGO Technik, was den NXT für fast jeden erdenklichen Einsatzbereich qualifiziert.

Die Datenblätter aller Komponenten und Protokolle sind frei von LEGO erhältlich, was NXT auch für Elektronik-Bastler interessant macht.

### **LEGO Mindstorms EV3**

EV3 stellt die Nachfolge-Kollektion von NXT dar. Die Hauptänderung besteht in Ersetzen des ARM7-Prozessor durch einen ARM9-Prozessor (300 MHz), auf dem eine Linuxumgebung ausgeführt wird. Eine Schnittstelle für microSD-Karten wurde hinzugefügt, welche als Programm- und Ressourcenspeicher dient.

EV3 ist kompatibel zu allen NXT-Sensoren und -Motoren und besitzt einen weiteren Ausgang am EV3-Stein. Das LCD-Display ist auf 178 mal 128 Pixel gewachsen. Das Design der Module wurde leicht geändert, die Auswahl ist bis auf ein weiteres, kleineres Motormodul identisch zu NXT.

Das Linux-Betriebssystem ermöglicht es im Vergleich zum Mikrocontroller des NXT, weit größere und komplexere Programme auszuführen und so den Aufgabenbereich zu erweitern. EV3 unterstützt offiziell die Verbindung zu Apple-Geräten.

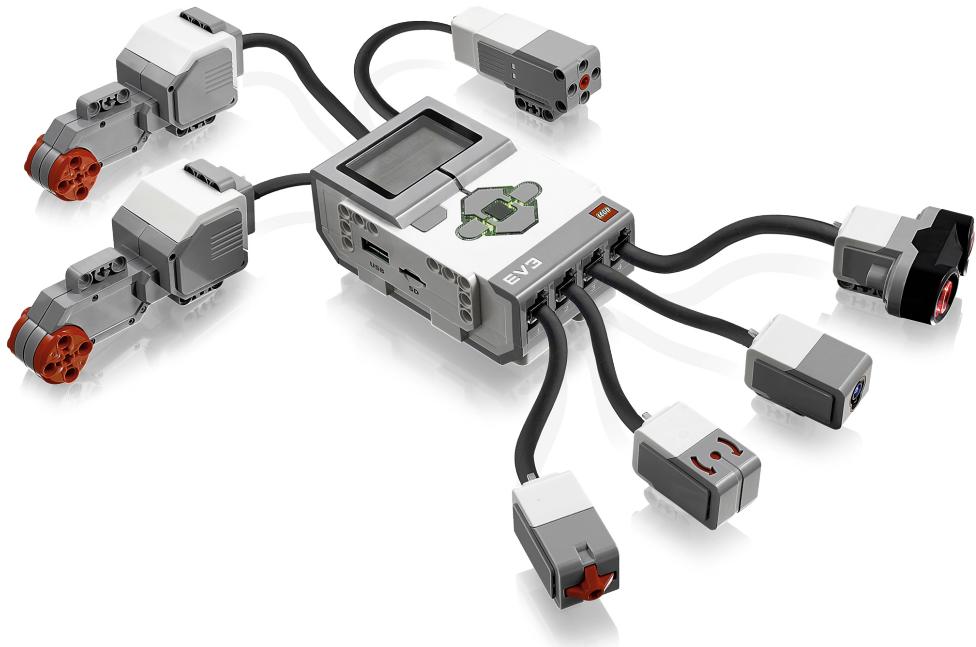


Abbildung 3.2: Hauptbestandteile des LEGO Mindstorms EV3-Systems

Über einen neuen USB-Host-Anschluss können nun USB-Geräte wie ein optionales WLAN-Dongle angeschlossen werden, was die Erweiterungsmöglichkeiten vervielfacht.

### Fischertechnik Robotics

Fischertechnik ist ein Konstruktions- und Baukastensystem, welches von den Fischerwerken produziert wird. Das Baukastensystem beruht auf einem Grundbaustein, der an allen 6 Seiten anbaubar ist. Für den Roboterbau stehen zusätzlich verschiedene Systemkomponenten wie Motoren, Zahnräder, Sensoren und Interfaces zur Verfügung. Diese sind sowohl untereinander als auch mit den Grundbausteinen beliebig kombinierbar.

Die Computing Bausätze als Grundaufbau eines Roboters sind in verschiedenen Varianten erhältlich und enthalten Bauanleitungen für drei bis acht Modelle. Sie sind mit unterschiedlichen Sensoren und Motoren ausgestattet, natürlich können auch hier selbst mit Bauteilen experimentiert und eigene Roboter entwickelt und gebaut werden.

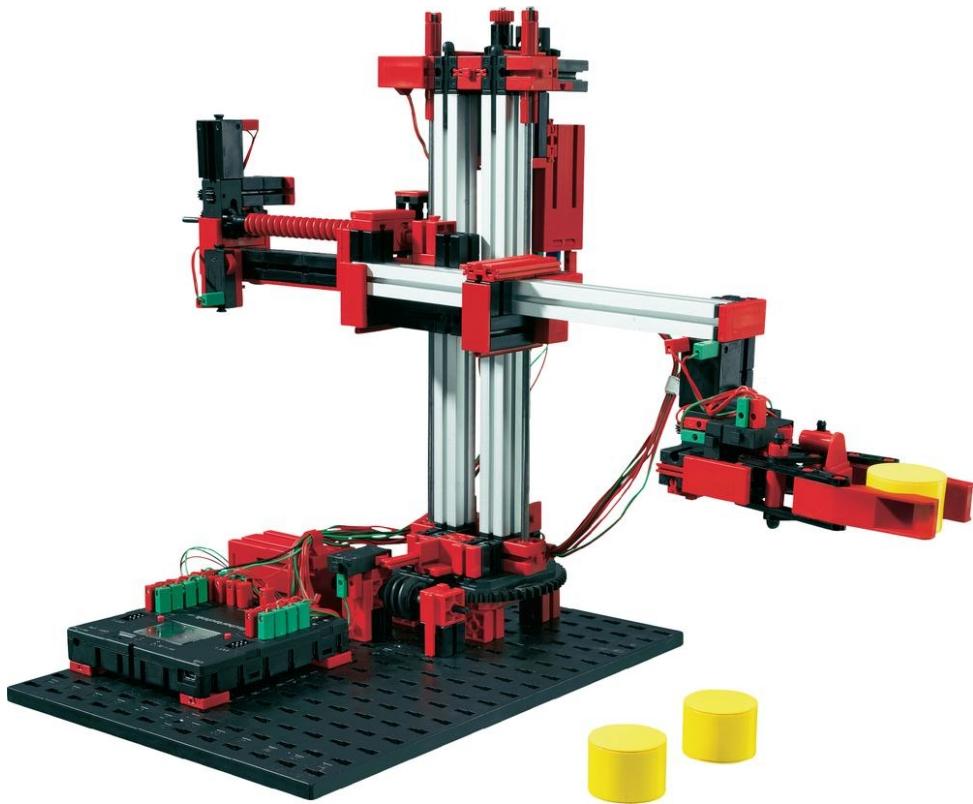


Abbildung 3.3: Beispiel-Aufbau des Fischertechnik

Die Programmausführung übernimmt das ROBO Interface, aufbauend auf einen 16 Bit Mikrocontroller, vergleichbar mit den intelligenten Mindstorms-Steinen. Dieser wertet abhängig der Programmierung die Signale der Sensoren an den Eingängen aus und speist die Motoren an den Ausgängen.

An den PC angeschlossen wird das ROBO Interface entweder über ein 9-poliges serielles Kabel oder über USB. Funkübertragung zwischen PC und den Robotern wird durch das Erweiterungs-pack ROBO RF Data Link ermöglicht.

Mit bis zu drei Erweiterungsmodulen ROBO I/O-Extension stehen zusätzlich jeweils 4 weitere Motorenausgänge, 8 digitale und ein analoger Eingang zum Anschluss der Module zur Verfügung. Das IR Control Set erweitert das ROBO Interface um Fernsteuerung via Infrarot.

Programmiert wird wahlweise mithilfe der grafischen Programmoberfläche ROBO Pro Software oder über freie APIs in den Sprachen C, C++, und Java.

Letztendlich wurde sich für den LEGO Mindstorms NXT-Bausatz entschieden. Er bietet für den gedachten Einsatzzweck alle nötigen Module und Schnittstellen; Der Nachfolger EV3 wäre nur dann nötig, wenn auf dem Roboter komplexere Algorithmen abgearbeitet werden müssten, was den ARM7-Prozessor des NXT überfordern und die Linux-Umgebung des EV3 voraussetzen würde. Da hier jedoch sämtliche Berechnungen durch die App erledigt werden, ist dies nicht nötig.

Die Dokumentation der LEGO Mindstorms-Systeme sind wesentlich ergiebiger, frei erhältliche Anleitungen zahlreicher und der Einstieg wesentlich leichter als die des Sets von Fischertechnik Robotics, was einen Prototypen-Entwurf in Fischertechnik im Vergleich zu NXT verlängert hätte.

## 3.2 Wahl des Kameramoduls

Die Hauptfrage bezüglich des Kameramoduls bestand in der Wahl zwischen einem Ein- oder einem Zweikamerasystem.

Der Vorteil eines Zweikamerasystems besteht in der Möglichkeit für wesentlich bessere Orientierung im 3D-Raum, da Entfernungsmittel der beiden Differenzbilder präziser berechnet werden können. Im Gegensatz dazu ist beim Einkamerasystem die Entfernungs berechnung auf ein 2D-Bild beschränkt und nicht annähernd so genau.

Jedoch ist der Berechnungsaufwand für das Auswerten zweier Differenzbilder ungleich höher, weshalb sich letztendlich aufgrund dieser Ungleichheit des Implementierungsaufwandes für ein Einkamerasystem entschieden werden.

Weitere Aspekte sind Auflösung und Öffnungswinkel des Kameramoduls. Höhere Auflösung bedeutet bessere Erkennung von Gegenständen auf weitere Entfernung; Ein größerer Öffnungswinkel heißt, dass mehr Raum in einem Bild erfasst werden kann, somit weniger Drehbewegung des Roboters in Richtung eines Objekts nötig ist, bis es erfasst und detektiert werden kann.



Abbildung 3.4: Kameramodul des Nexus 5

Ein Problem bei der Kamera des Nexus ist, das sich diese nicht zentral auf dem Rücken des Smartphones befindet, sondern nach links oben versetzt. Dies setzt voraus, dass entweder Software- oder Hardwareseitig dieser Versatz aus der Bildverarbeitung kompensiert wird.

Hier wurde der Einfachheit halber die Halterung am Roboter verschoben, wodurch aus der Sicht des Roboters die Kamera in der Mitte liegt.

# 4 Hardwareumsetzung

## 4.1 Anforderungen an den NXT-Roboter

Der Roboter hat einige hardwareseitige Anforderungen. Das System muss sich frei im Raum bewegen können, sowie eine Möglichkeit bieten mit Gegenständen zu interagieren. Diese müssen aufgehoben, transportiert und zielgerichtet abgelegt werden können. Es ist zusätzlich wichtig, dass die Gegenstände beim Ablegen sicher an ihrer Position verharren, damit sie sich nicht aus einer möglichen Zielzone heraus bewegen. Wichtiges Kriterium ist zusätzlich eine sichere Halterung für das Kameramodul, welches in Kapitel 3.2 näher beschrieben ist.

## 4.2 Entwurf des Roboters

Da das LEGO Mindstorm NXT Kit häufig zur Realisierung kleinerer Roboterprojekte verwendet wird, hat LEGO eine Datenbank an möglichen Bauplänen bereitgestellt [2]. Nach einiger Recherche und Durchsicht diverser Bauanleitungen für verschiedenste Anwendungsbereiche wurde sich für einen von LEGO bereitgestellten Aufbau den entschieden. Abbildung 4.1 zeigt den Roboter, wie LEGO ihn bereitstellt.

Da der Aufbau jedoch nicht allen Anforderungen genüge tut, wurde dieser abgeändert. Der von LEGO bereitgestellte Bauplan benutzt einen Schall- und einen Farbsensor. Diese wurden nicht benötigt und daher entfernt. An ihrer Stelle wurde stattdessen eine Halterung für das in Kapitel 3.2 beschriebene Kameramodul in Form eines Smartphones eingebaut. Um dem Gewicht des



Abbildung 4.1: Von Lego bereitgestellter Roboteraufbau

Smartphones entgegenzuwirken musste der in Abschnitt 4.2.1 beschriebene NXT-Stein etwas nach hinten verlagert werden. Abbildung 4.2 zeigt den fertig modifizierten Roboter.

#### 4.2.1 NXT-Stein

Der NXT-Stein bildet die Recheneinheit und damit die Hauptkomponente des Robotersystems [3]. Er besitzt drei Ausgänge für Motoren an der Oberseite, über die gleichzeitig die Rotations-sensoren in den Servo-Motoren ausgelesen werden.

Auf der Unterseite befinden sich vier Eingänge für verschiedene Sensoren, die je nach Anwendungszweck über Flachbandkabel bestückt werden können, etwa ein Licht-, Schall-, Tastsensor oder beliebige Kombinationen daraus. Dies macht das NXT-System zu einem sehr flexiblen da einfach konfigurierbaren Robotersystem.



Abbildung 4.2: Bild des fertig modifizierten Roboters

Über einen USB-Anschluss oben wird der NXT mit dem PC verbunden um ihn mit Programmen zu versorgen. Hierfür wird von LEGO eine graphische Entwicklungsumgebung bereitgestellt um auch Neulingen den Einstieg in die Roboter-Programmierung zu vereinfachen und ihnen die Möglichkeit zu bieten schon innerhalb weniger Minuten einen funktionierenden Prototypen zu konstruieren. Im Fall des CLEEN-R-Aufbaus wird jedoch, wie in Abschnitt 4.3 geschildert, lediglich die interne Bluetooth-Verbindung genutzt.

Auf der Vorderseite befindet sich ein  $100 \times 64$  Pixel auflösendes binäres LCD-Display über das Einstellungen getätigt oder Statusmeldungen ausgegeben werden können. Auch Sound-Ausgabe über einen integrierten 8-Bit-Lautsprecher ist möglich.



Abbildung 4.3: NXT-Stein

#### 4.2.2 Sensoren

LEGO stellt unterschiedliche Sensoren für den Betrieb an einem NXT-Robotersystem bereit. Die wichtigsten Sensorarten sind in den nachfolgenden Abschnitten beschrieben.

##### Tastsensor



Abbildung 4.4: Tastsensor des NXT-Systems

Der berührungsempfindliche Tastsensor dient in der ursprünglichen Vorlage zur Detektion von Gegenständen im Bereich des Greifarms. Wird er betätigt, so kann der Greifarm geschlossen werden. Im CLEEN-R-Projekt wurde dieser jedoch durch einen Ultraschallsensor ersetzt, der nicht nur die unmittelbare Berührung bemerkte sondern auch ein herannahendes Objekt während der Fahrt detektiert. Der Ultraschallsensor ist in einem nachfolgenden Abschnitt beschrieben.

### RotationsSENSOREN

Die Servomotoren des Roboters sind mit Rotationssensoren ausgestattet. Diese Sensoren erlauben es dem NXT-Roboter, die Geschwindigkeit der Motoren abhängig des Widerstands des Untergrunds zu regulieren. So werden unter anderem präzises Abbremsen und Fehlerminimierung bei der Positionsbestimmung ermöglicht.

### FARBSENSOR



Abbildung 4.5: Farbsensor des NXT-Systems

Dem ursprünglichen Entwurf des Roboters liegt ein RGB-Farbsensor bei. Seine Aufgabe war es die Farbe des Untergrunds festzustellen. Im CLEEN-R-Projekt wurde dieser Sensor zwar entfernt, könnte jedoch bei Erweiterung des Roboters dazu genutzt werden um die Farbe des Untergrunds aus Aufnahmen des Kameramoduls zu entfernen. Dies würde die Qualität der Objekterkennung auf farbigen Untergrund deutlich erhöhen. Zusätzlich ist es denkbar die Erkennung von Zielzonen über farbige Markierungen auf dem Untergrund zu realisieren.

### ULTRASCHALLSENSOR

Der Ultraschallsensor dient zur Messung der Distanz vom Roboter zum nächsten soliden Objekt. Der Sensor hat eine Reichweite von  $255\text{cm}$  und kann mit einer Präzision von  $\pm 3\text{cm}$  Entferungen angeben. Der Sensor wird in diesem Projekt benutzt um zu detektieren, wie weit ein



Abbildung 4.6: Ultraschallsensor des NXT-Systems

Objekt vom Greifarm entfernt ist. Hat es einen gewissen Abstand erreicht, so kann der Greifarm geschlossen werden um das Objekt aufzunehmen.

In Kapitel 6.2.1 werden alternative Möglichkeiten zur Entfernungsmessung geschildert. Der Ultraschallsensor stellt sich jedoch als geeignetstes Mittel heraus und wird daher im folgenden verwendet.

#### 4.2.3 Aktoren

Die Aktoren des Robotersystems beschränken sich beim CLEEN-R-Projekt auf Servomotoren. Im Folgenden werden die unterschiedlichen Anwendungen der Motoren beschrieben.

##### Antriebsmotoren



Abbildung 4.7: Servo-Motor des NXT-Systems

Der NXT-Roboter verwendet zwei Servomotoren an den Seiten als Differentialantrieb. Dieser ermöglicht eine freie Fortbewegung und das Rotieren des Roboters.

### Greifarm

Der Greifarm des Roboters wird über einen dritten Motor auf der Vorderseite bewegt. Der Motor erlaubt das freier öffnen und schließen des Greifarms und somit das mitführen von Gegenständen.

Bei der Ansteuerung des Greifarmmotors ist zu beachten, dass dieser nicht zu schnell geschlossen werden sollte, da dies zu tragende Objekte wegstoßen könnte. Abbildung 4.8 zeigt den Greifarm in geschlossenem Zustand mit einem transportierten Objekt.

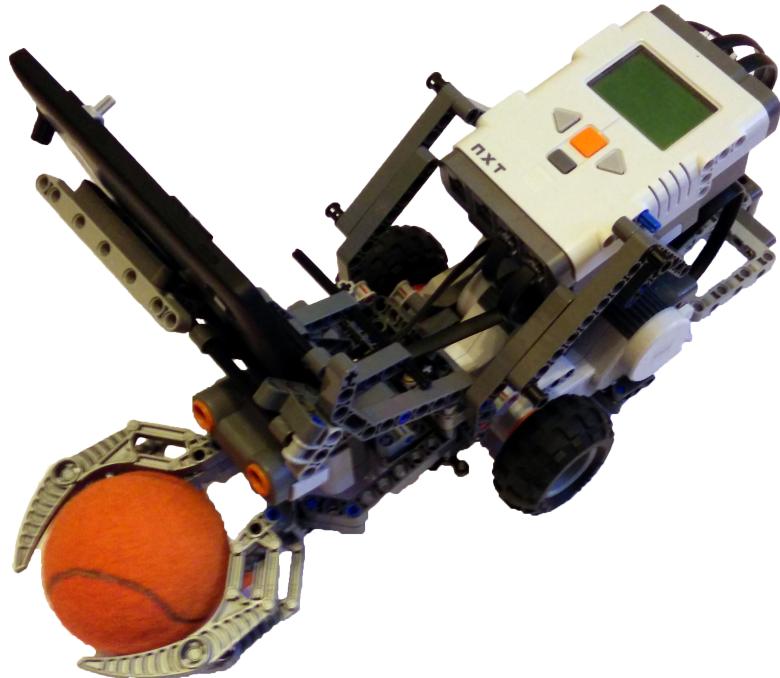


Abbildung 4.8: Greifarm des Robotersystems in geschlossenem Zustand mit Objekt

## 4.3 Steuerung des Roboters

Da die Kameraeinheit in Form des Smartphones und der NXT-Roboter zwei getrennte Module darstellen, ist ein nichttrivialer Kommunikationskanal notwendig. In den folgenden Abschnitten ist die intermodulare Kommunikation näher beschrieben.

### 4.3.1 Kommunikation zwischen den Modulen

Die Steuerung des Roboters durch das Smartphone erfolgt via Bluetooth. Der NXT-Roboter stellt hierzu als Bluetooth-Gerät das Serial Port Profile (SPP) bereit, welches das Standardinterface auf Bluetooth-Ebene darstellt und vom Großteil aller Bluetooth-Hostgeräte unterstützt wird. Es ermöglicht das simple serielle Senden und Empfangen von Bytes auf Low-Level-Ebene.

Das Kommunikationsprotokoll auf High-Level-Ebene und damit die nötigen Befehle zum Regeln der Aktoren und Auslesen der Sensoren wurde von LEGO dokumentiert und ist online erhältlich [4].



Abbildung 4.9: Bluetooth-Verbindung zwischen NXT und Nexus 5

Um die Logik zentral zu halten wird auf dem NXT selbst kein Programm ausgeführt. Das Smartphone führt zentral alle Berechnungen durch sendet dem Roboter lediglich Befehle zur Ansteue-

rung der Aktoren. Sensordaten können ebenfalls durch das Smartphone über die Bluetooth-Verbindung abgerufen werden.

### 4.3.2 Kommunikation über die Bluetooth-Schnittstelle

Zunächst muss eine Bluetooth-Verbindung erstellt werden. Hierfür müssen beide Geräte eine aktive Bluetooth-Schnittstelle aufweisen. Der Roboter muss diese Schnittstelle zusätzlich von außen sichtbar schalten.

Bei Erstverbindung muss der gesuchte NXT ausgewählt werden. Für Folgeverbindungen ist die Bluetooth-Adresse bekannt und die App kann ohne Benutzerinteraktion eine Verbindung mit dem NXT-Roboter aufnehmen.

Ist eine Verbindung geglückt, kann damit begonnen werden seriell byteweise Befehle an den NXT zu übertragen. Eventuelle Antworten, wie beispielsweise angeforderte Sensordaten werden über die selbe Schnittstelle empfangen.

### 4.3.3 Programmatische Umsetzung

Auf dem Smartphone übernimmt ein gesonderter Thread die Kommunikation mit dem NXT. Die Klasse „NxtTalker“ bietet Funktionen zum Versenden von Daten an den Roboter, sowie alle nötigen Daten um eine Verbindung aufzubauen und aufrecht zu erhalten.

Um die Motoren zu bewegen, muss zunächst ein Befehl an jeden Aktor geschickt werden, der eine Geschwindigkeit im Bereich von -100% bis 100% definiert. Hierdurch können die Motoren mit dem selben Befehl vorwärts und rückwärts fahren. Das stoppen der Motoren, und damit das Anhalten des Systems werden realisiert, indem allen Motoren eine Geschwindigkeit von null kommuniziert wird. Problematisch ist hierbei, dass die Motoren sequenziell gestartet werden. Dies hat zur Folge, dass ein Motor vor dem anderen startet und somit die Blickrichtung des

Roboters beeinflusst. Um dem vorzubeugen sieht die Schnittstelle einen Befehl vor, der die Motoren synchron starten lässt. Hierdurch wird verhindert, dass der Roboter in eine undefinierte Richtung abdriftet.

# 5 Softwareumsetzung

Mit Hilfe des in Kapitel 3.2 beschriebenen Kameramoduls müssen verschiedene Aufgaben aus dem Bereich der Bildverarbeitung bewältigt werden.

## 5.1 Wahl der Bildverarbeitungsbibliothek

Die Umsetzung der zu bewältigenden Aufgaben kann durch die Wahl einer geeigneten Bildverarbeitungsbibliothek deutlich vereinfacht werden. Wichtige Kriterien für die Wahl der Bibliothek sind unter anderem Funktionsumfang, Dokumentation und Aktivität der Community.

### 5.1.1 LibCCV

LibCCV [5] ist eine open-source Bildverarbeitungsbibliothek, die viele bekannte Algorithmen implementiert. LibCCV steht unter einer BSD-Clause-3-Lizenz und kann somit für eine Studienarbeit problemlos unbegrenzt verwendet werden. Die Bibliothek ist größtenteils in C++ verfasst und somit potenziell auf einem Android-Smartphone verwendet werden. Die Verwendung auf dem Smartphone wird jedoch nicht offiziell unterstützt und kann potenziell weitere Schwierigkeiten mit sich bringen.

### 5.1.2 Imagemagick

Bei Imagemagick [6] handelt es sich um eine Bildverarbeitungsbibliothek, welche sehr viele Algorithmen bereits implementiert hat. Algorithmen zur Objekterkennung müssten jedoch

vollständig selbst implementiert werden, was zu einem großen zusätzlichen Aufwand führen kann. Imagemagick wird unter der Apache 2.0 Lizenz vertrieben.

### 5.1.3 OpenCV

OpenCV [1, 7] stellt eine der größten Open-Source-Bibliotheken für Bildverarbeitung da. Die Bibliothek hat einen starken Fokus auf Echtzeitverarbeitung und wird daher auch in vielen Projekten im Bereich der Robotik verwendet. OpenCV hat eine große aktive Community, wodurch eventuelle Fragen und Probleme schnell beantwortet werden können. Zusätzlich bietet OpenCV eine offizielle Version für Android und eignet sich somit ideal für diese Studienarbeit.

## 5.2 Algorithmen zur Objekterkennung

Aufgabe des Roboters, ist es Gegenstände in einem Raum mit Hilfe von Kamerabildern zu erkennen. Folglich spielt die Objekterkennung eine große Rolle.

### 5.2.1 Farbbasierte Objekterkennung

Einen einfachen Ansatz der Objekterkennung unter Verwendung von Methoden der in 5.1.3 beschriebenen Bibliothek OpenCV stellt eine farbbasierte Objekterkennung dar. Hierfür wird das Kamerabild zunächst vom RGB-Format in das HSV-Format konvertiert. Dies wird durchgeführt, da das HSV-Format unempfindlicher gegen Veränderungen in der Beleuchtung ist als das RGB-Format [8]. Abbildung 5.1 zeigt die Aufteilung eines Bildes in die verschiedenen Kanäle.

Wie in Abbildung 5.1 zu sehen ist, eignet sich vor allem der Saturation-Kanal des Bildes um farbige Objekte zu erkennen, da dieser hohe Werte annimmt wenn die Farbintensität hoch ist. Zuletzt erfolgt eine Binarisierung mit einem empirisch ermittelten Schwellwert von 50. Dies ist

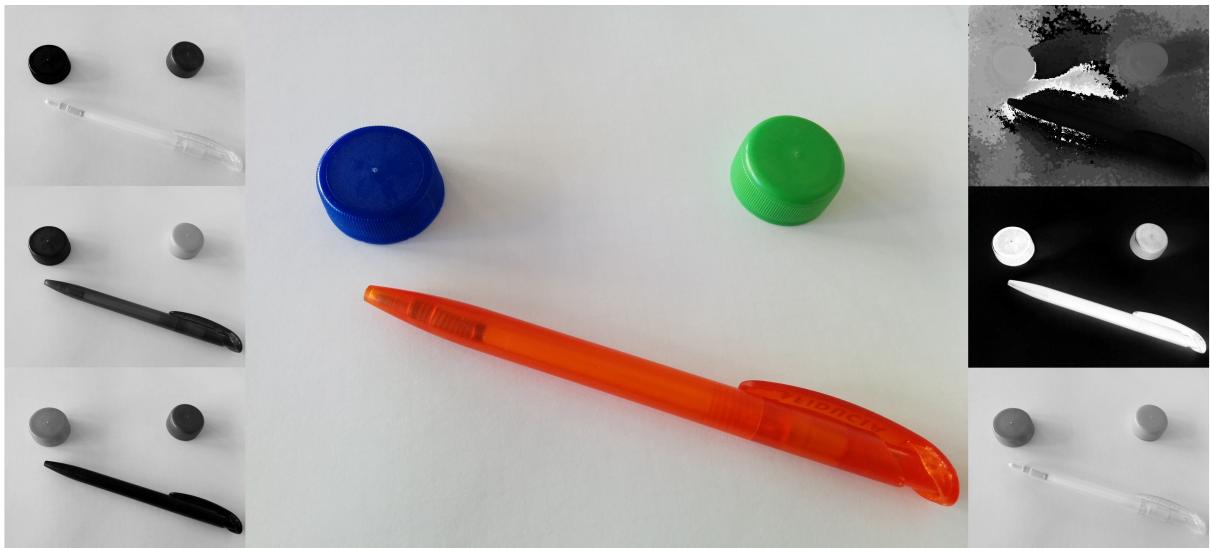


Abbildung 5.1: Beispiel einer Aufspaltung in RGB-Kanäle (links) und HSV-Kanäle (rechts)

notwendig, da die, in der OpenCV-Bibliothek implementierte, Methode von Suzuki und Abe [9] zur Segmentierung von Objekten ein Binärbild erwartet. Aus dem in Abbildung 5.1 dargestellten Beispiel entsteht nach der Binarisierung schließlich Abbildung 5.2. Der Algorithmus zur Segmentierung erkennt daraus drei Objekte und zeichnet ihren Konturen, wie in Abbildung 5.3 ersichtlich, ein.

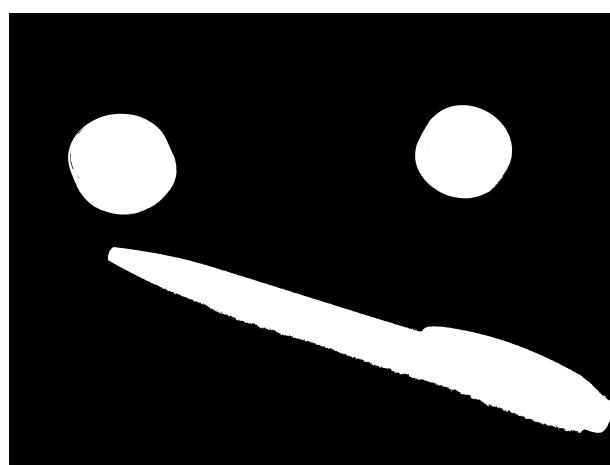


Abbildung 5.2: Beispiel der Binarisierung des Sättigungskanals

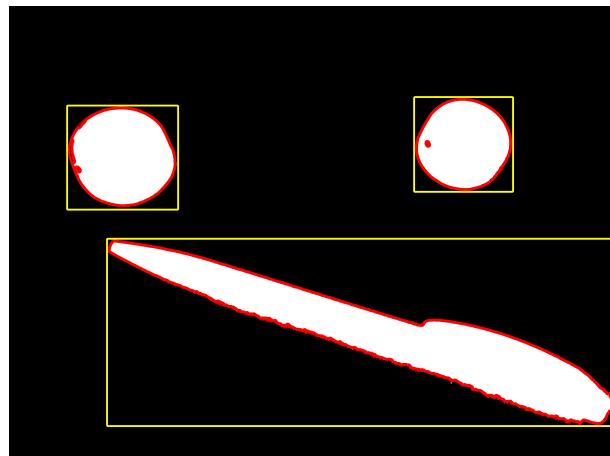


Abbildung 5.3: Beispiel einer Segmentierung des binarisierten Sättigungskanals

## 5.3 Ortsbestimmung

Die Aufgabe des Roboters ist es, Gegenstände in vordefinierte Bereiche zu befördern. Hierzu muss ihm jederzeit bekannt sein wo diese sich befinden, bzw. wo er sich relativ zu diesen aufhält, um sie anfahren zu können.

### 5.3.1 Mitteilung von Beobachterkomponente

Ist im Raum ein weiteres System vorhanden, dessen Position konstant und dem Roboter zu jedem Zeitpunkt bekannt ist, kann eine simple, und genaue Positionsbestimmung erfolgen. Eine solche Beobachterkomponente kann über Sensordaten, wie beispielsweise einer Kamera oder einem Ultraschallsensor, zu jedem Zeitpunkt Kontakt zu Roboter haben. Der Roboter muss hierbei keine Berechnungen durchführen und verlässt sich für die korrekte Positionsbestimmung auf den Beobachter, von welchem zu jedem Zeitpunkt Position und Orientierung des Roboters im Raum abrufbar ist.

Die Beobachterkomponente stellt eine sehr effiziente Art von Ortsbestimmung dar, hat jedoch den beträchtlichen Nachteil, dass zusätzliche Hardware erforderlich ist.

### 5.3.2 Raumanalyse

Bei der Raumanalyse werden Kamerabilder analysiert und aus ihnen Fixpunkte oder Geraden extrahiert, mit denen sich der Roboter bei Bewegung im Raum zu orientieren versucht. Dies ist ein sehr rechenaufwändiges Verfahren, welches zusätzlich mit einer großen Implementierungsarbeit verbunden ist.

Zusätzlich stellt dieses Verfahren Anforderungen an den Raum in dem es verwendet wird. Im Raum müssen genügend charakteristische Orientierungspunkte vorhanden sein. Verliert der Roboter die Orientierung, ist es nicht ohne weiteres möglich diese wiederzuerlangen.

Vorteil des Verfahrens der Raumanalyse ist, dass die benötigte Hardware in Form eines Kameramoduls und eines leistungsstarken Prozessors bereits vorhanden ist.

### 5.3.3 Positionsverfolgung

Die Positionsverfolgung als Mittel der Ortsbestimmung bietet sich bei Robotern an, welche einen sehr exakten Sensor zur Beobachtung der zurückgelegten Strecke besitzen. Über die zurückgelegte Strecke und die Orientierung des Roboters kann zu jedem Zeitpunkt ein Bewegungsvektor erzeugt werden, der die genaue Position des Roboters bestimmt.

Problematisch bei diesem Verfahren ist jedoch, dass eine sehr genaue Bestimmung der zurückgelegten Strecke benötigt wird. Fehler bei der Bestimmung wirken sich additiv aus. Die hat zur Folge, dass die Positionsverfolgung bei längeren Fahrten immer ungenauer wird. Eine Art der Streckenbestimmung stellen beispielsweise Rotationssensoren in Rädern dar. Hierbei wird die Genauigkeit des Verfahrens durch das Spiel der Räder und der Auflösung der Sensoren beeinträchtigt.

Da das CLEEN-R-Projekt nicht über die zusätzliche Hardware einer Beobachterkomponenten verfügt und der zusätzliche Aufwand der Auswertung der Kamerabilder zur Merkmalsextrakt-

tion zu aufwändig wäre, wurde sich für die Variante der Positionsverfolgung entschieden. Die Auflösung der Rotationssensoren ist mit  $1^\circ$  ausreichend genau. Die automatische Geschwindigkeitsanpassung der NXT-Motoren abhängig des Untergrunds fördern zusätzlich eine exakte Ortsbestimmung über die Positionsverfolgung.

## 5.4 Struktur der Applikation

Da es sich um eine große Applikation mit mehr als 3000 Zeilen Code und 33 Klassen handelt, sind in den folgenden Abschnitten lediglich Kernelemente herausgegriffen und im Detail erklärt.

### 5.4.1 Grundstruktur

Eine Android Applikation muss mindestens eine Klasse haben, die von der abstrakten Klasse *Activity* erbt. Diese Activity, oftmals *MainActivity* genannt, wird beim Starten der Applikation aufgerufen. Sie verwaltet grafische Oberflächen, sowie das Verhalten der Applikation beim Starten, Schließen oder Pausieren. Im CLEEN-R-Projekt stellt die *MainActivity* zusätzlich eine Schnittstelle zur OpenCV-Bibliothek dar. Die Activity wird über eingehende Kamerabilder informiert und vermittelt diese weiter an den Controller der Applikation; die Klasse *CleenrBrain*.

Zusätzlich hält die *MainActivity* eine Referenz zur Klasse der Hardwareansteuerung *NxtTalker*. Abbildung 5.4 zeigt die Grundstruktur der *MainActivity*

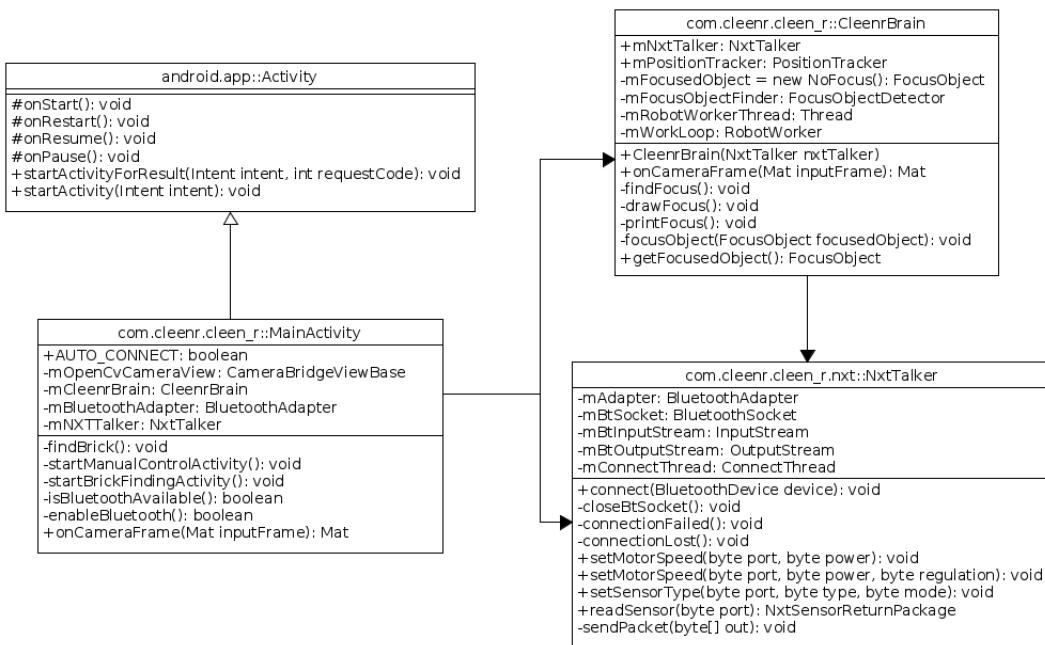


Abbildung 5.4: UML-Klassendiagramm der Grundstruktur der Applikation

## 5.4.2 Fokusobjektfindung

Die Controllerklasse `CleenrBrain` hat die Aufgabe den Roboter zu koordinieren. `CleenrBrain` hält eine Referenz zu einem `FocusObject`-Objekt, welches den aktuell fokussierten Gegenstand repräsentiert. Hierbei wird auch das Fehlen eines Fokusobjekts aus Gründen der Polymorphie als Objekt beschrieben. `CleenrBrain` wird beim vorliegen einer neuen Kameraaufnahme benachrichtigt und ermittelt daraufhin mit Hilfe einer `FocusObjectDetector`-Klasse ein neues `FocusObject`. Hierbei wird, wie in Kapitel 6.1.1 beschrieben versucht einen Gegenstand über mehrere Aufnahmen hinweg im Fokus zu behalten. Abbildung 5.5 beschreibt das Zusammenspiel der beschriebenen Klassen anschaulich.

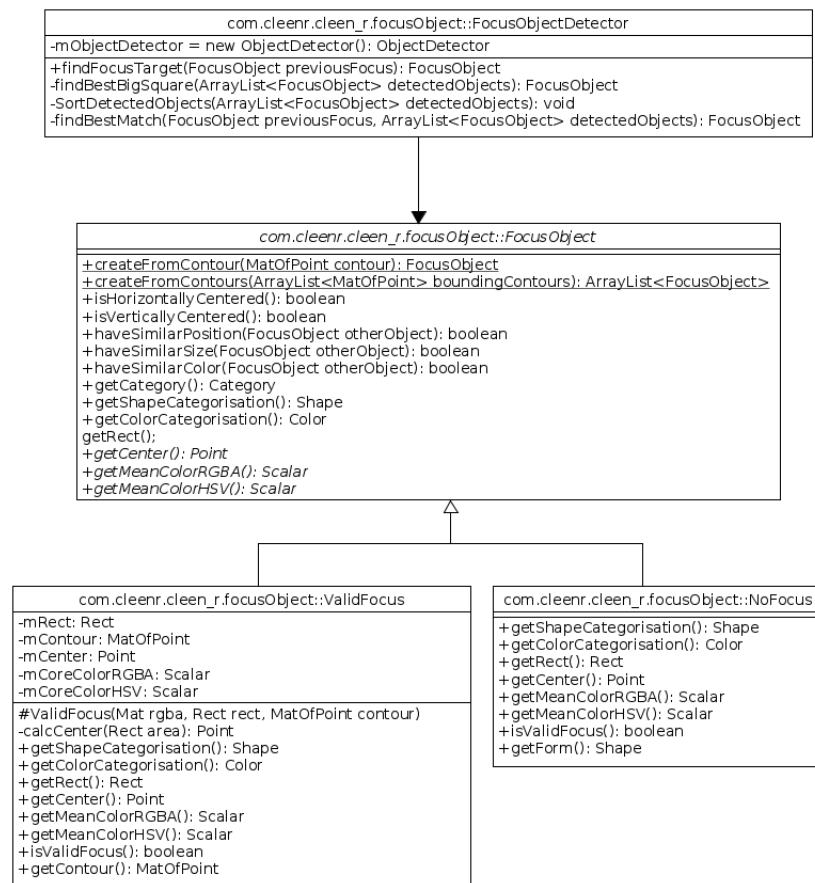


Abbildung 5.5: UML-Klassendiagramm der für die Auffindung eines Fokusobjekts verantwortlichen Klassen

### 5.4.3 Objektkategorisierung

Wie auch aus Abbildung 5.5 ersichtlich, beinhaltet die `FocusObject`-Klasse eine Methode zur Kategorisierung des Gegenstands. Hierbei wird eine Kategorie der Klasse `Category` erzeugt, welche sich aus den beiden Feldern `Color` und `Shape` zusammensetzt.

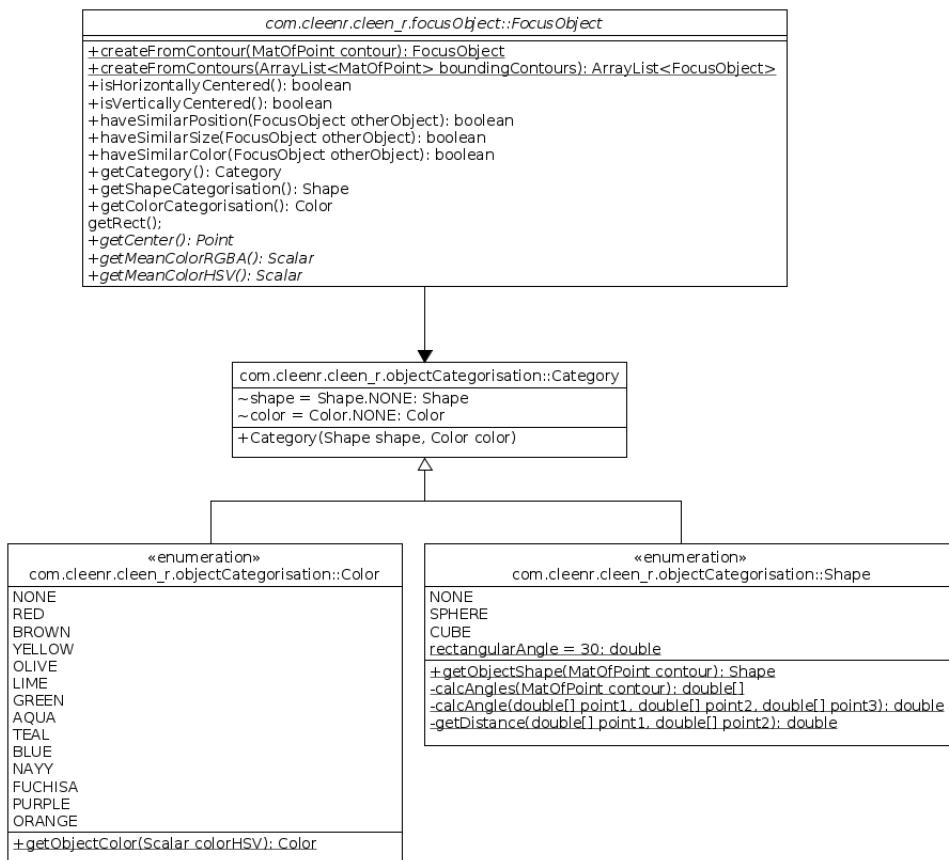


Abbildung 5.6: UML-Klassendiagramm der Kategorisierung von Fokus Objekten

#### 5.4.4 Bewegungssteuerung und Positionsverfolgung

Die Bluetooth-Verbindung wird mittels einer **NxtTalker**-Instanz durch einen Aufruf von *MainActivity* aus initiiert. Ein Thread überwacht den Stream auf eingehende Sensordaten und setzt Befehle an den NXT ab.

Bewegungsbefehle der Workloop gehen in die Klasse *NxtControlUnit* ein, von wo aus sie von **NxtTalker** in Bluetooth-Befehle übersetzt werden. Die Verfolgung der Roboterposition wird hauptsächlich in der Klasse *PositionTracker* abgewickelt. Jeder Bewegungsbefehl setzt eine

Mitteilung an **PositionTracker** ab, welcher dann anhand Dauer und Geschwindigkeit der Bewegung die neuen Koordinaten berechnet.

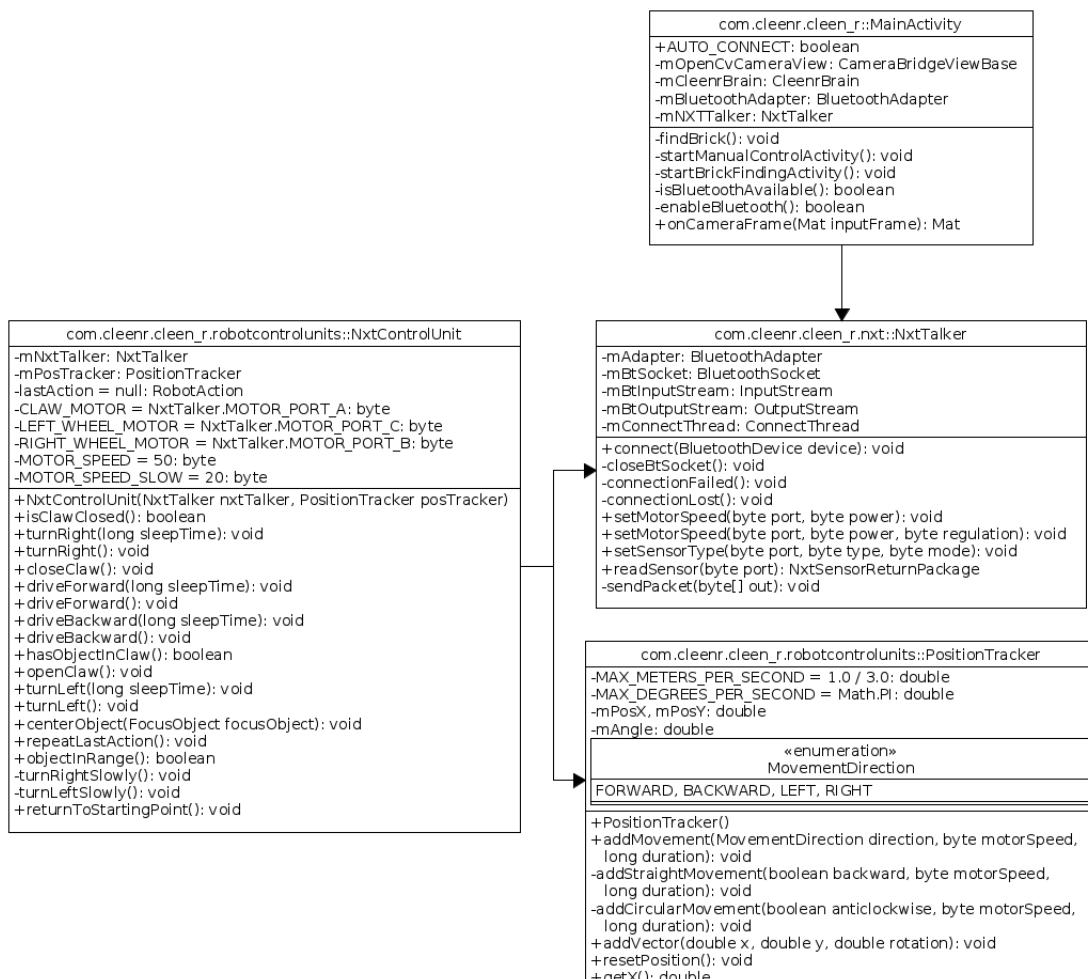


Abbildung 5.7: UML-Klassendiagramm der Bewegungssteuerung und Positionsverfolgung

#### 5.4.5 Arbeitsphasen

Die Controllerklasse *CleenrBrain* verwaltet mit Hilfe eines zusätzlichen Threads den Arbeitszyklus des Roboters. Durch die Verwendung eines zweiten Threads kann die Suche nach neuen

Fokus Objekten in folgenden Aufnahmen parallel zur Abarbeitung der in Kapitel 6 beschriebenen Arbeitsphasen geschehen.

Die Arbeitsphasen sind separat als eigene Klassen modelliert, die sich von einer abstrakten Klasse *WorkPhase* ableiten. Jede Arbeitsphase verfügt über eine virtuelle *executeWork*-Methode, welche Zugriff auf alle für die Arbeitsphase relevanten Operationen besitzt. Darunter zählen Steuerwerk des Roboters, eine Referenz auf das aktuelle Fokusobjekt und einer Möglichkeit die Arbeitsphase zu wechseln. Die Implementierung der *PickingUpObject*-Klasse ist in Listing 5.1 erkennbar. Es ist deutlich zu sehen, dass der Code durch die hinzugefügte Abstraktionsebene sehr einfach gehalten werden kann.

Listing 5.1: Implementierung der Arbeitsphase „Gegenstand aufnehmen“

```
public class PickingUpObject extends WorkPhase {
    public PickingUpObject(RobotWorker worker) {
        super(worker);
    }
    @Override
    public void executeWork(FocusObject focusObject, RobotControlUnit
controlUnit) {
        controlUnit.closeClaw();
        if (!controlUnit.hasObjectInClaw()) {
            controlUnit.openClaw();
            mRobotWorker.switchWorkphase(new
                SearchingObject(mRobotWorker));
            return;
        }
        mRobotWorker.switchWorkphase(new CategorisingObject(mRobotWorker));
    }
}
```

# 6 Arbeitsablauf und Problematiken

Der Arbeitsablauf des Roboters kann in unterschiedliche Phasen unterteilt werden. Diese Phasen sind in den nachfolgenden Abschnitten beschrieben und zusätzlich in Abbildung 6.1 als Flussdiagramm dargestellt.

## 6.1 Objektsuche

Sobald mehrere Objekte durch die in Kapitel 5.2 dargestellten Algorithmen erkannt wurden, muss eines als geeignetes Fokusziel gewählt werden. Wie diese Wahl erfolgt, ist im nachfolgenden Abschnitt beschrieben. Ist ein Fokusobjekt gefunden, so muss es in der nachfolgenden Aufnahme wiedererkannt werden um einen Wechsel zu verhindern.

Ein zufälliges Umspringen auf andere erkannte Objekte wäre für den Prozess sehr hinderlich. In den folgenden Abschnitten sind Kriterien und Probleme beschrieben, an Hand derer ein erkanntes Objekt in folgenden Aufnahmen erkannt werden kann.

Das Fokusobjekt wird von der Applikation durch ein grünes Rechteck, andere gefundene Objekte durch ein rotes markiert. Dieses Vorgehen ist in Abbildung 6.2 dargestellt.

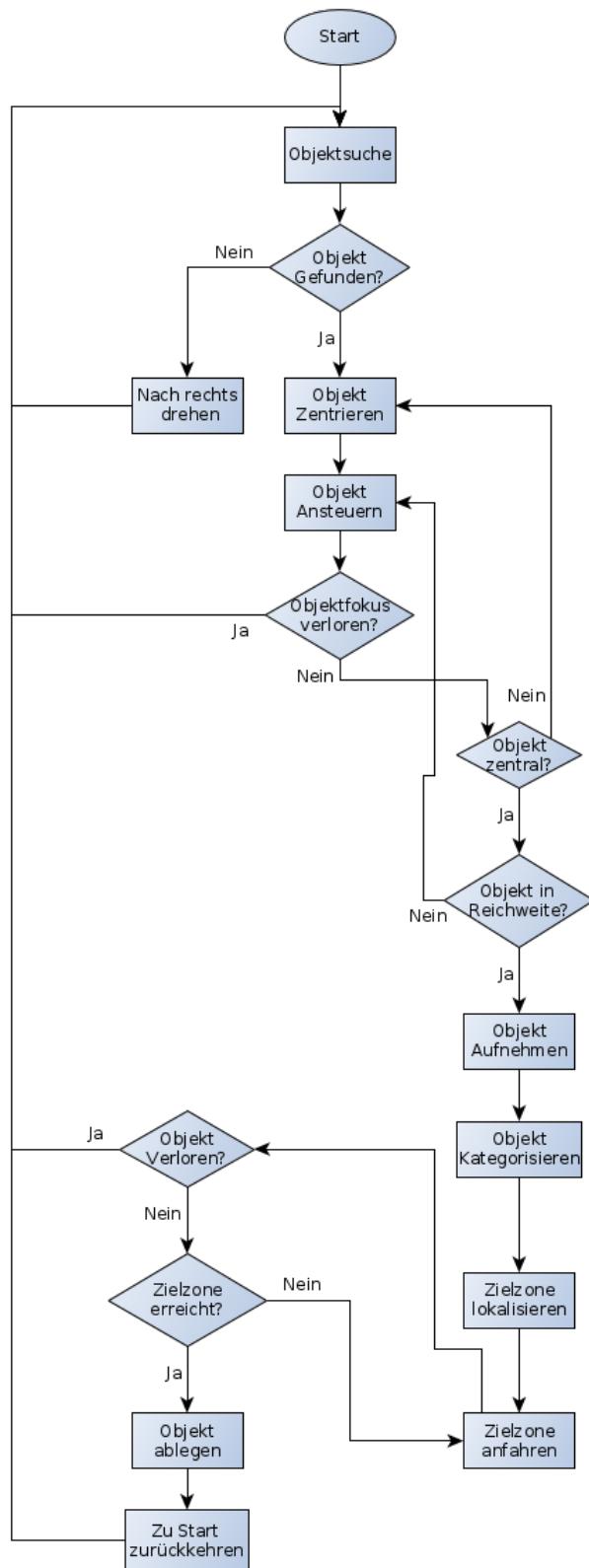


Abbildung 6.1: Flussdiagramm des Arbeitsablaufs

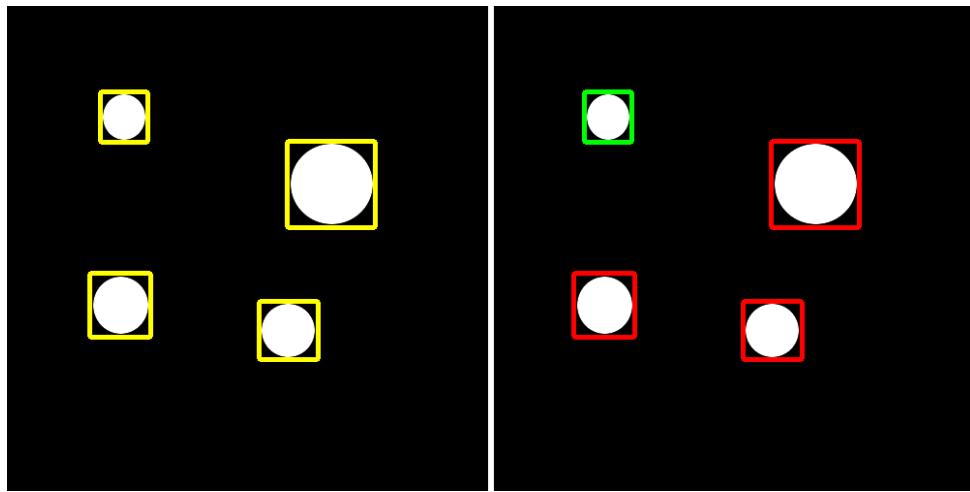


Abbildung 6.2: Auffindung eines Fokusobjekts

### 6.1.1 Auffinden eines Fokusobjekts

Die Wahl eines ersten Fokusobjekts erfolgt nach Positions- und Größenkriterien. Es wird versucht ein Objekt zu wählen, welches möglichst zentral in der Aufnahme liegt.

Eine weitere Bedingung ist, dass das begrenzende Rechteck des Fokusobjekts möglichst quadratisch sein soll. Dies ist der Annahme zu Grunde zu führen, dass die zu suchenden Objekte nur Würfel und Kugeln sein sollen. Beide Formen zeichnen sich dadurch aus, dass sie ein quadratisches Begrenzendes Rechteck haben.

### 6.1.2 Ähnlichkeitskriterien

Um den Fokus auf ein Objekt zu behalten lassen sich verschiedene Ähnlichkeitskriterien formulieren durch die das erkannte Objekt in der Menge der im nachfolgenden Bild erkannten Objekte wiedergefunden werden kann.

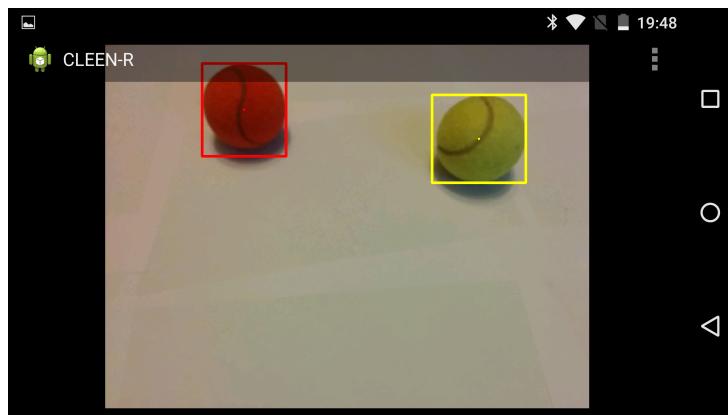


Abbildung 6.3: Screenshot bei Objektfindung

### Lokale Nähe

Das trivialste Kriterium stellt die lokale Nähe da. Geht man von einem Stillstand der Kamera und der aufgenommenen Szene aus, so befinden sich sämtliche Objekte in nachfolgenden Aufnahmen an der selben Stelle. Fokussierte Objekte können daher rein aus ihrer Position wiedergefunden werden. Ist das System in Bewegung, so kann nicht mehr von einer exakten Übereinstimmung der Koordinaten ausgegangen werden. Stattdessen wird eine gewisse Toleranz gegeben. Als Maß kann hierbei angenommen werden, dass sich die Position des Zentrums eines Objektes in fortlaufenden Aufnahmen um nicht mehr als beispielsweise 10% der Aufnahmegröße geändert hat. Abbildung 6.4 zeigt das beschriebene Verhalten anschaulich am Beispiel einer nachfolgenden Aufnahme aus Abbildung 6.2.

### Größenkriterium

Das Größenkriterium ähnelt zunächst dem Kriterium der lokalen Nähe, bezieht sich jedoch nicht auf den Mittelpunkt des Objekts, sondern auf die vom Objekt in der Aufnahme eingenommenen Flächen. Diese bleibt bei einer beliebigen Translation des Objektes in der Aufnahme konstant und eignet sich daher als Ähnlichkeitsmaß. Bewegt sich das Objekt jedoch auf die Kamera zu, oder von ihr weg, so verändert sich die eingenommene Fläche. Auch hier muss eine Toleranz gewählt

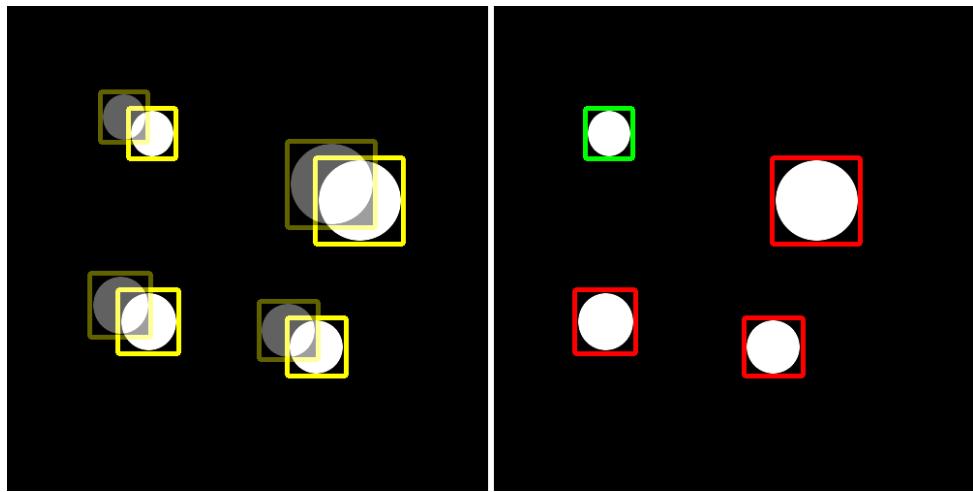


Abbildung 6.4: Objektverfolgung durch Kriterium der lokalen Nähe

werden. Diese kann entweder konstant gegeben sein (beispielsweise 10%) oder abhängig von der aktuellen Geschwindigkeit des Roboters gewählt werden. Abbildung 6.5 zeigt exemplarisch die Verfolgung eines Objektes dessen Größe sich im Verlauf der Aufnahmeserie verändert.

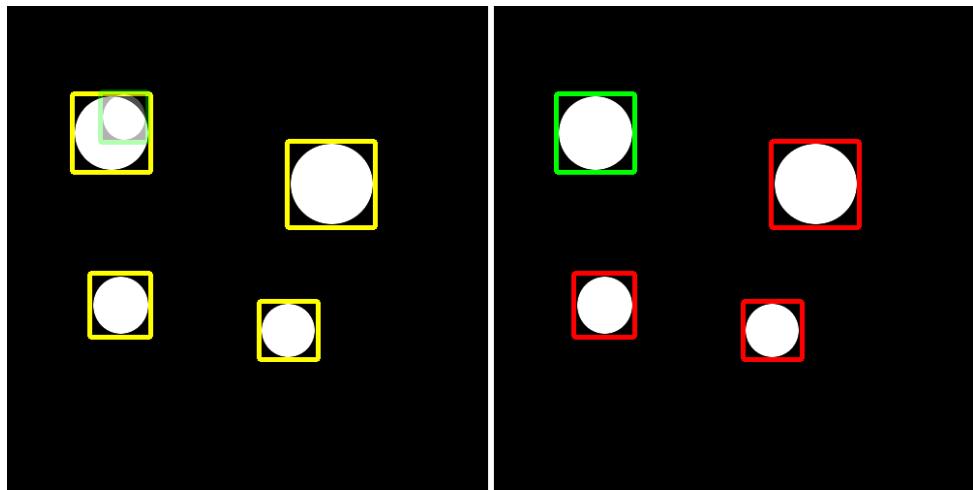


Abbildung 6.5: Objektverfolgung durch Größenkriterium

### Farbliche Ähnlichkeit

Ein drittes Ähnlichkeitsmaß stellt die farbliche Ähnlichkeit dar. Diese ist Abhängig vom Farbraum in dem die Aufnahme getätigt oder konvertiert wurde. Wie in Abschnitt 5.2 beschrieben, wird die Aufnahme bei der Objekterkennung in das HSV-Format konvertiert. Daher kann das Ähnlichkeitsmaß sich auf den Farbkanal, den Sättigungskanal oder den Intensitätskanal beziehen. Durch ausschließen des Intensitätskanals kann eine weitgehend helligkeitsunabhängige Bestimmung erreicht werden. Daher ist das Filtern nach Objekten ähnlicher Farbe und Sättigung durchaus sinnvoll. Auch hier lässt sich eine Toleranz formulieren, wie etwa 10% Abweichung von der durchschnittlichen Farbe des zuvor ermittelten Objektes. Abbildung 6.6 zeigt die farbliche Ähnlichkeit zweier Objekte und die dadurch resultierende Wiederfindung des Objektes in der Folgeaufnahme.

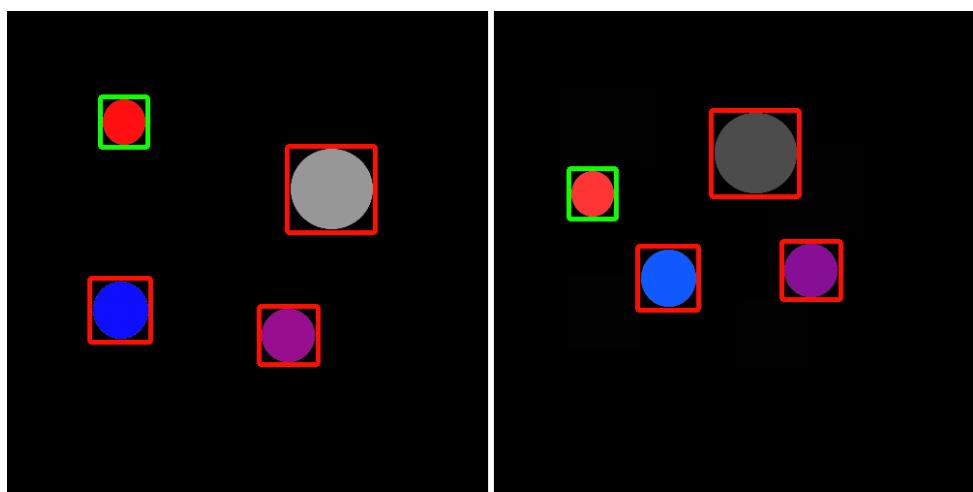


Abbildung 6.6: Objektverfolgung durch farbliche Ähnlichkeit

### 6.1.3 Kurzzeitiger Verlust des Verfolgungsziels

Tendenziell muss bei der Objektverfolgung immer von einem Verlust des verfolgten Objektes ausgegangen werden. Dies kann permanent sein, wie beispielsweise bei abrupter starker Veränderung der Lichtbedingungen, oder lediglich temporär durch Fehler bei der Bildaufnahme

geschehen. In diesem Abschnitt wird auf letzteres eingegangen. Bei längerfristigem Verlust des Objekts, begibt sich der Roboter zurück in die Phase der anfänglichen Objektsuche.

Es hat sich gezeigt, dass kleine Fehler oder Ungenauigkeiten bei der Bildaufnahme bereits dazu führen können, dass ein Objekt nicht in der darauf folgenden Aufnahme wiedergefunden werden kann. Ist dies nur von kurzer Dauer, muss dies gesondert behandelt werden um zu vermeiden, dass der Roboter einen plötzlichen Wechsel des Verfolgungsziels durchführt. Ähnlich der in Abschnitt 6.1.2 beschriebenen Ähnlichkeitskriterien muss auch hier eine Toleranz für eine maximale Anzahl Aufnahmen gelegt werden, in denen das verfolgte Ziel nicht gefunden werden muss ohne, dass das Objekt als „verloren“ angesehen wird. Die Wahl einer zu hohen Toleranz hat die Folge, dass der Roboter lange Zeit in einem undefinierten Zustand ein Objekt verfolgt, welches er nicht sieht. Ist die Schwelle zu niedrig gewählt, so kann dies ein häufiges Springen zwischen verschiedenen Objekten zur Folge haben, was sich sehr negativ auf die Laufzeit auswirken könnte. Empirisch hat sich gezeigt, dass eine solche Schwelle bereits bei etwa drei Aufnahmen ohne das verfolgte Objekt liegen kann. Wird ein Objekt verloren, so begibt sich der Roboter zurück in den Zustand der Objektsuche.

#### 6.1.4 Verzerrte Aufnahmen durch Bewegung des Roboters

Durch Bewegung des Roboters während der Bildaufnahme, kann es zu verzerrten Bildern kommen. Dies kann unterschiedliche Folgen haben. Abbildung 6.7 zeigt eine solche verzerrte Aufnahme.

Durch die Verzerrung können unterschiedliche Phänomene auftreten. Die Verfolgung von Objekten, die in Abschnitt 6.1.2 beschrieben wurde kann versagen. In diesem Fall muss die Aufnahme ignoriert und verworfen werden. Es können jedoch auch Objekte fälschlicherweise erkannt und kategorisiert. Durch die Verzerrung wird die Form und Position der Objekte verfälscht. Eine solche verfälschte Aufnahme kann einen plötzlichen Wechsel des fokussierten Objektes zur Folge haben.

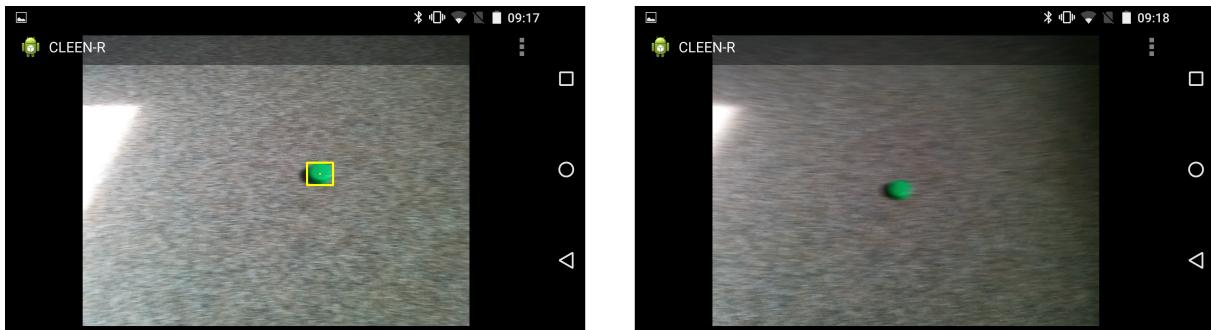


Abbildung 6.7: Verzerrte Aufnahme durch Bewegung des Roboters und Erkennungsverlust nach Helligkeitsänderung

## 6.2 Anfahren von Objekten

Sobald ein Objekt gefunden wurde, kann der Roboter beginnen dies anzufahren. Hierfür versucht er zunächst das Objekt horizontal zu zentrieren. Dafür dreht sich der Roboter auf der Stelle, bis das Objekt ganz zu sehen ist und das möglichst zentral in der Aufnahme liegt. Abbildung 6.8 zeigt, wie ein Objekt durch die Drehung des Roboters zentriert werden konnte.

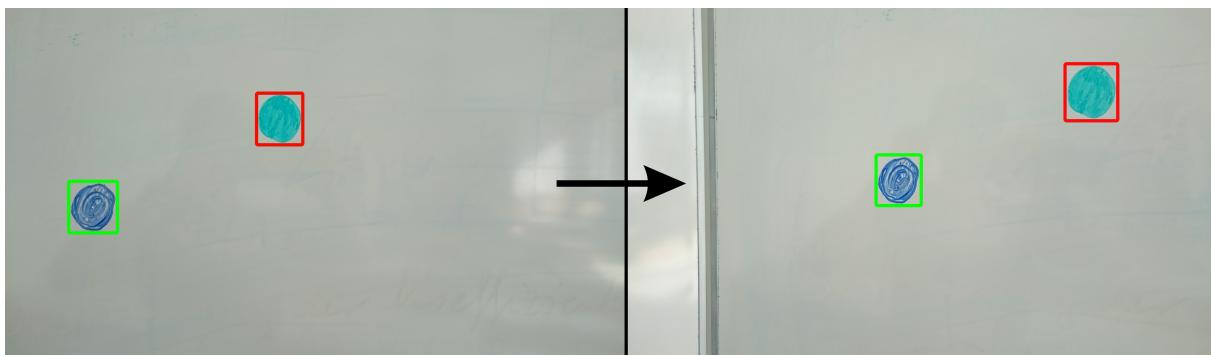


Abbildung 6.8: Zentrierung eines Objekts

Ist das Objekt zentral gelegen, so muss der Roboter lediglich geradeaus fahren, bis er es erreicht. Da eine perfekte Zentrierung jedoch nicht garantiert werden kann, ist es möglich, dass der Roboter abdriftet. Um dem vorzubeugen muss ständig die Position des Objekts in der Aufnahme überprüft und gegebenenfalls zentriert werden.

Weiterhin wichtig, ist es jedoch zu wissen, wann in den Zustand des Objekt Aufhebens übergegangen werden kann. Hierfür darf der Roboter nur eine geringe Entfernung zum Objekt aufweisen. Die folgenden Abschnitte gehen auf verschiedene Methoden der Entfernungsmessung ein.

### 6.2.1 Entfernungsschätzung

Die Entfernungsschätzung ist für mehrere Teile der Roboterroutine wichtig. Zunächst muss der Roboter, der ein Objekt per Kamera erkannt hat anfahren und aufheben. Hierfür muss er wissen ob der aufzuhebende Gegenstand in Reichweite des Greifarms ist. Weiterhin benötigt der Roboter die Entfernungsschätzung bei der Navigation. Er muss Hindernisse, wie beispielsweise Wände eines Raumes, erkennen bevor er kollidiert und unter Umständen seine Fracht verliert.

#### Stereokamerabasiert

Ein häufiger Ansatz für die Entfernungsschätzung stellt die Stereokamerabasierte Methode dar. Hierbei nimmt der Roboter seine Umgebung nicht mit nur einer Kamera, sondern mit mehreren, mindestens zwei, wahr. Durch einen bekannten Versatz zwischen den Aufnahmequellen der Bilder, lässt sich mittels Registrierung auch ein Versatz in den erzeugten Bildern ermitteln. Wie die beiden menschlichen Augen, erlaubt dies das Wahrnehmen der Umgebung in dritter Dimension. Abbildung 6.9 zeigt schematisch eine solche Entfernungsschätzung [10].

Der Stereokamerabasierte Ansatz ist von der Ausführung sehr elegant, da er sich an der Natur und dem menschlichen räumlichen Sehen orientiert.

Problematisch ist jedoch, dass zusätzliche Hardware benötigt wird. Eine weitere Kamera als bildgebendes Mittel birgt einen neuen Grad der Komplexität, da diese mit dem vorhandenen Smartphone synchronisiert werden muss. Zusätzlich müssen zeitintensive Berechnungen durchgeführt werden, wodurch die Echtzeitfähigkeit des Systems beeinträchtigt werden könnte.

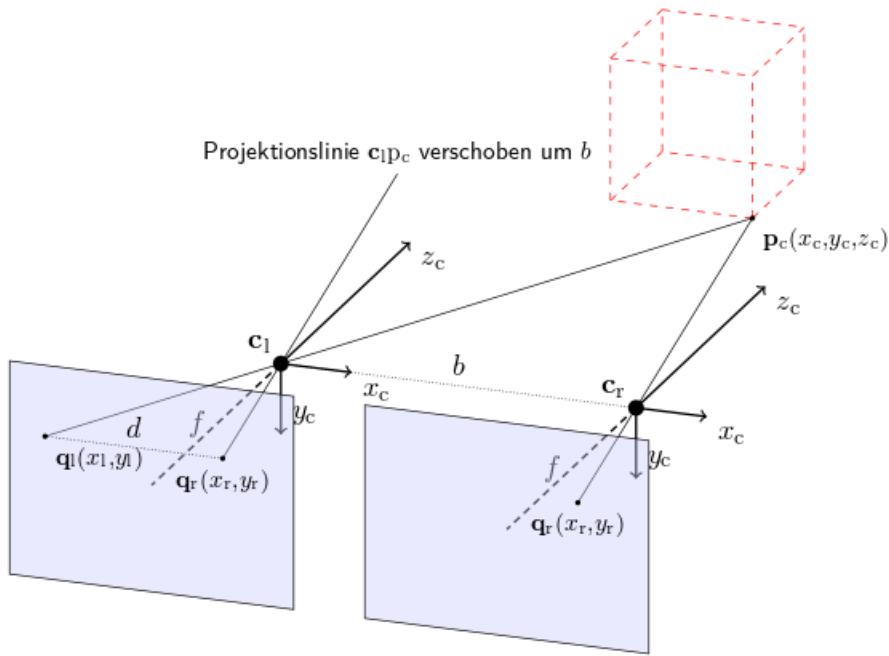


Abbildung 6.9: Stereokamerabasierte Entfernungsschätzung

### Monokamerabasiert

Der monokamerabasierte Ansatz stützt sich bei der Entfernungs berechnung auf alle vorhandenen Informationen. Hierbei wird mittels der bekannten Position und Inklination der Kamera, sowie ihres Öffnungswinkels ein virtuelles Blickfeld erstellt. Dieses Blickfeld breitet sich von der Kamera aus pyramidenförmig aus und trifft den als planar angenommenen Boden. Abbildung 6.10 zeigt den Aufbau eines solchen Systems.

Da der Boden als planar angenommen wird, ist davon auszugehen, dass die Position eines Objekts durch die niedrigste y-Koordinate in der Aufnahme approximiert werden kann. Dies ist aus Abbildung 6.11 näher ersichtlich.

Abbildung 6.12 zeigt hierbei die entstehende Aufnahme des Verfahrens, sowie eine Positions schätzung des Objekts. Die Parameter wurden realitätsgerecht angenommen und sind aus Tabelle 6.1 ersichtlich.

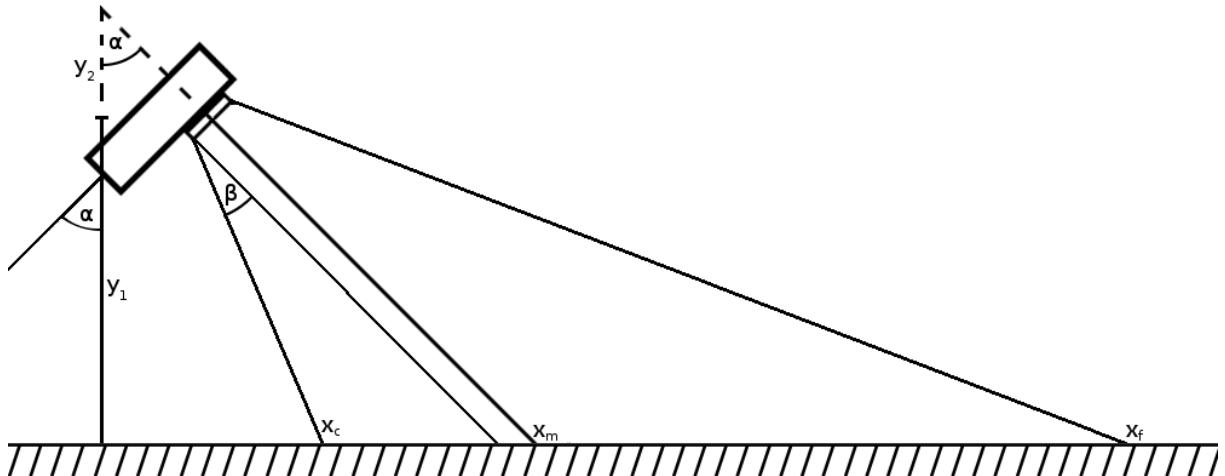


Abbildung 6.10: Aufbau der monokamerabasierten Entfernungsschätzung

Parameter	Wert	Erklärung
$y_1$	5cm	Kamera „schwebt“ 5cm in der Luft
$\alpha$	$45^\circ$	Kamera ist im $45^\circ$ Winkel zum Boden geneigt
$\beta$	$40^\circ$	Kamera hat $40^\circ$ Öffnungswinkel

Tabelle 6.1: Parameter der Positionsschätzung

Entfernungen lassen sich über einfache trigonometrische Verfahren berechnen.

Problematisch bei diesem Ansatz ist hierbei die Ungenauigkeit des Systems. Mit steigender Entfernung des Objekts steigt auch die Fehlerrate. Sowohl Fehler bei der Aufnahme, als auch Probleme durch eine maximale Größenbeschränkung des detektierten Objektes können in der Praxis zu Fehleinschätzungen und damit fehlerhafter Benutzung des Greifarms führen.

In den durchgeführten Tests, konnte lediglich eine Genauigkeit von  $\pm 5\text{cm}$  erreicht werden. Dies ist für die gegebene Anwendung unzureichend, da die fehlerhafte Benutzung des Greifarms gegebenenfalls Objekte verschieben und aus dem Blickfeld bewegen könnte.

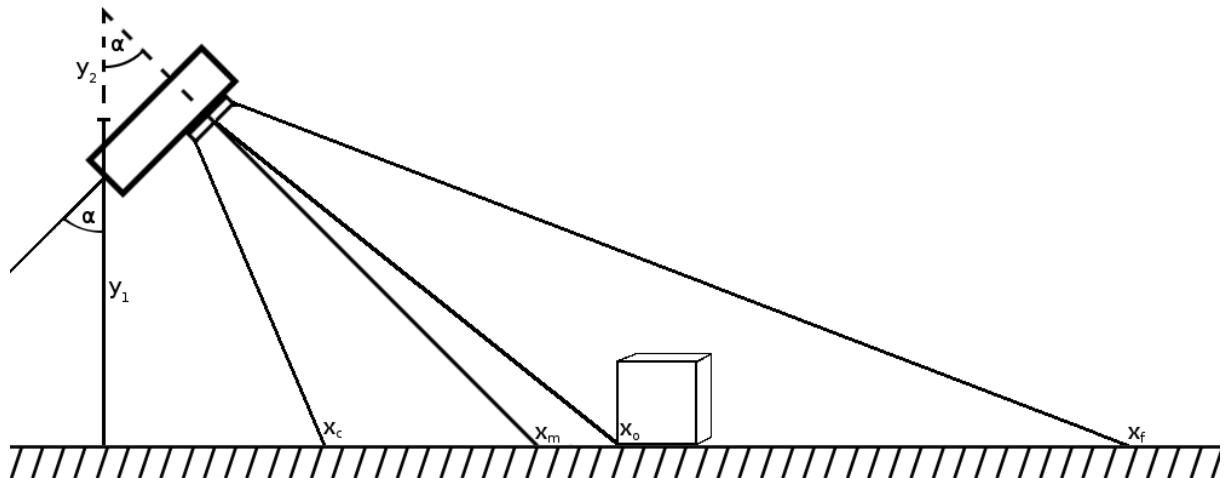


Abbildung 6.11: Exemplarische monokamerabasierte Entfernungsschätzung

### Ultraschallbasiert

Die ultraschallbasierte Entfernungsschätzung kann auch über große Distanzen von bis zu zwei Metern zentimetergenaue Entfernungen messen. Die Messung wird durch einen Echo-Mechanismus mit, für den Menschen unhörbaren, Schallwellen im Ultraschallbereich durchgeführt [11]. Zunächst wird jedoch, ähnlich dem in 6.2.1 beschriebenen Verfahren, zusätzliche Hardware in Form eines Ultraschallsensors benötigt. Ein solcher kann jedoch direkt von LEGO<sup>TM</sup> erworben und problemlos am Roboter angebracht werden. Der Sensor ist im Kapitel 6.2.1 näher beschrieben.

Das ultraschallbasierte Verfahren kann durch eine kleine Änderung des Roboters realisiert werden und liefert sehr genaue Ergebnisse. Die Koppelung mit dem Smartphone ist bereits gegeben und genügt Echtzeitanforderungen.

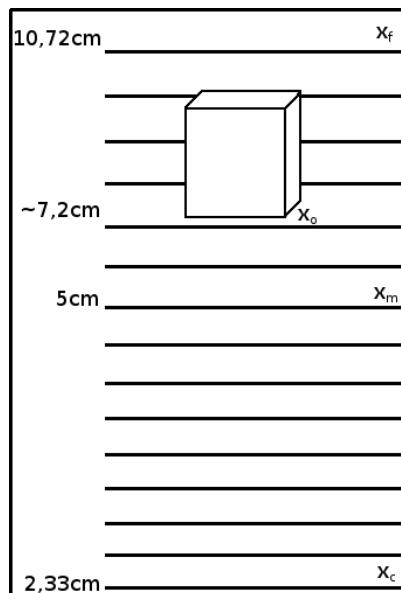


Abbildung 6.12: Aufnahme aus der monokamerabasierten Entfernungsschätzung

### 6.3 Aufnehmen von Objekten

Das Aufnehmen von Objekten kann erfolgen sobald der Roboter, wie in Abschnitt 6.2 beschrieben, nah genug an das Objekt herangefahren ist. Zum Aufnehmen wird der Greifarm durch das Betreiben des dritten Motors geschlossen. Näheres hierzu findet sich im Kapitel 4.2.3.

Mit Hilfe des in Abschnitt 6.2.1 beschriebenen Ultraschallsensors kann ebenfalls überprüft werden ob das Objekt während des Tragens verloren geht. In diesem Fall kehrt der Roboter zurück in den Zustand der Objektsuche.

### 6.4 Kategorisierung von Objekten

Objekte werden nach zwei Kriterien kategorisiert. Diese Kriterien sind in den folgenden Abschnitten näher beschrieben.

Farbe	HSV-Kriterium
Rot	$H < 15 \vee H \geq 160$
Orange	$15 \leq H < 20$
Gelb	$20 \leq H < 35$
Grün	$35 \leq H < 60$
Blau	$60 \leq H < 140$
Violett	$140 \leq H < 160$

Tabelle 6.2: Umrechnung von HSV-Werten in konkrete Farben.

#### 6.4.1 Kategorisierung nach Farbe

Die Farbe des Objekts stellt ein Kriterium dar. Diese wird über die gesamte Objektverfolgung ermittelt. So kann akkurat eine beleuchtungsunabhängige Kategorisierung in menschlich wahrnehmbare Farben stattfinden. Tabelle 6.2 beschreibt die Auswertung der im HSV Farbraum ermittelten Farbwerte.

In jeder Aufnahme wird die konkrete Farbe des Objekts bestimmt. Anschließend kann bei der Kategorisierung die am häufigsten bestimmte Farbe gewählt werden um Fehler durch Beleuchtung oder andere äußere Einflüsse zu korrigieren.

#### 6.4.2 Kategorisierung nach Form

Die Kategorisierung nach Form erfolgt über extrahierte Randpunkte des Objekts. Der in Kapitel 5.2 beschriebene Algorithmus liefert eine Liste von Punkten aus denen sich die Kante des segmentierten Gegenstands ergibt. Abbildung 6.13 zeigt wie ein Kreis und ein Quadrat über Kantenpunkte beschrieben werden.

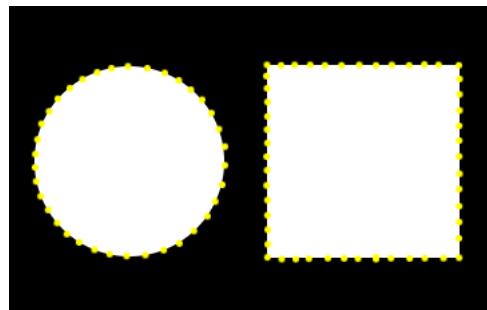


Abbildung 6.13: Approximation von Objektkanten über Punktlisten

Berechnet man aus der geordneten Liste von Kantenpunkten Winkel zwischen aufeinanderfolgenden Punkten, so lässt sie mit hoher Präzision die Form des Objektes schätzen. Abbildung 6.14 zeigt wie ein Kreis und ein Quadrat von einander unterschieden werden können. Der Kreis zeigt hierbei charakteristisch, dass der Winkel zwischen zwei Kantenpunkten immer konstant bleibt, während die Winkel bei Rechtecken sich nur an Ecken von null unterscheiden.

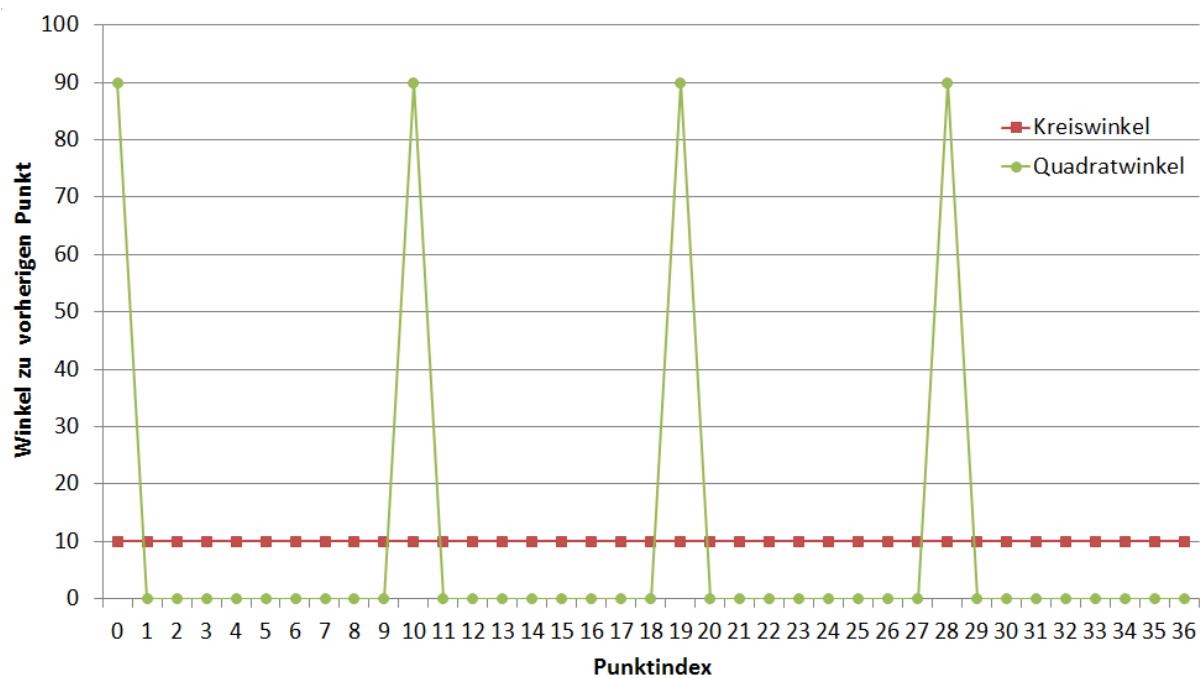


Abbildung 6.14: Bestimmung von Objekten über Winkel zwischen Kantenpunkten

Während ein Objekt angefahren wird, wird eine solche Kategorisierung wiederholt ausgeführt um eine höhere Genauigkeit zu erhalten.

## 6.5 Suche des Zielbereichs

Zur Orientierung im Raum wird die im Abschnitt 5.3 beschriebe Methode der streckenbasierten Positionsverfolgung genutzt. Hierbei wird zu Beginn der Startpunkt mit den Koordinaten (0|0) an der Position des Roboters definiert, die Drehung des Roboters mit  $0^\circ$ . Ab diesem Zeitpunkt wird jede Ansteuerung der Räder softwareseitig als Bewegung auf diesen Startpunkt aufaddiert.

Die Klasse *PositionTracker* behält Position und Winkel des Roboters im Blick. Da der Roboter nur gerade fahren oder sich auf der Stelle drehen kann, also die Geschwindigkeit der beiden Antriebsmotoren stets gleich ist, ist die Vektorberechnung recht einfach.

Ausgegangen wird von der Maximalgeschwindigkeit des Roboters, welche mit der angesteuerten prozentualen aktuellen Geschwindigkeit skaliert wird. Mit der Dauer der Bewegung erhält man die zurückgelegte Strecke. Abhängig vom Winkel des Roboters im Vergleich zum Startzeitpunkt werden nun trigonometrisch der X- und der Y-Anteil ( $X = \sin(\alpha)$ ,  $Y = \cos(\alpha)$ ) berechnet und auf die Roboterposition aufaddiert.

Bei Drehung um die eigene Achse wird die kalibrierte maximale Drehgeschwindigkeit skaliert, mit der Dauer multipliziert und der resultierende Winkel auf den des Roboters aufaddiert.

## 6.6 Ansteuerung des Zielbereichs

Durch das in Kapitel 6.5 beschriebene Verfahren besitzt der Roboter stets aktuelle Koordinaten des eigenen Standpunkts im Raum relativ zum Startpunkt. Somit kann er jederzeit die nötigen Strecken als Differenzvektor von sich zu den definierten Punkten der Zielbereiche berechnen.

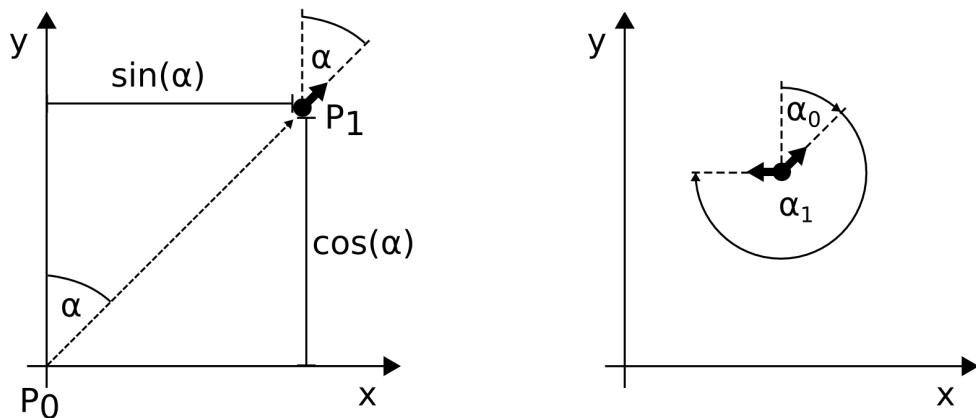


Abbildung 6.15: Berechnung der X- und Y-Anteile und des Winkels bei Fahrt und Rotation des Roboters im Raum

Ist dies erledigt, dreht sich der NXT zunächst um die eigene Achse, bis die Richtung des Streckenvektors mit dem Roboterwinkel übereinstimmt, er also dem Zielpunkt zugerichtet ist und fährt anschließend gerade darauf zu bis er die Zielzone erreicht hat.

## 6.7 Ablegen von Objekten

Das Ablegen von Objekten erfolgt analog zum in Abschnitt 6.3 beschriebenen Aufnehmen von Objekten. Sobald die Zielzone erreicht wird, kann das getragene Objekt abgelegt werden. Dies erfolgt über die Ansteuerung des dritten Motors, der den Greifarm steuert.

Sobald das Objekt an der Zielzone abgelegt wurde, kann der Roboter zurück an seine Ausgangsposition fahren.

## 6.8 Rückkehr zur Ausgangsposition

Analog zur in 6.6 beschriebenen Methode für die Ansteuerung von Zielbereichen, wird diese nun angewandt um den Roboter zum Startpunkt mit den Koordinaten (0|0) zu manövrieren.

Sobald der Roboter die Ausgangsposition erreicht hat, ist ein Zyklus seines Arbeitsablaufs abgeschlossen. Durch Wiederholen der in diesem Kapitel beschriebenen Schritte kann ein weiteres Objekt detektiert, kategorisiert und in eine Zielzone transportiert werden.

# 7 Tests des Robotersystems

## 7.1 Realtests

Das kontrollierte Testen eines Roboters ist nicht ganz leicht. Aufgrund der Abhängigkeiten zwischen Hard- und Software reichen reine Software-basierte Tests hier nicht aus. Bei der Übertragung der Befehle für Bewegung des Roboters beispielsweise muss auch tatsächlich der Roboter vorhanden und betriebsbereit sein, die Befehle müssen über eine bestehende Bluetooth-Verbindung an ihn übertragen und schließlich dessen Reaktion beobachtet werden.

Deshalb wurde häufig auf die Methode des Ausprobierens („Trial and Error“) zurückgegriffen. Sobald eine Komponente wie die Objekterkennung als ganzes geschrieben war und kompilierte, wurde die App auf das Smartphone geladen und an echten Objekten getestet.

Angefangen hat dies mit einfachen Versuchen in OpenCV, wie eine simple Erkennung der Farbe der Objekte vor der Linse. Hat dies nicht funktioniert, wurden die Parameter angepasst, also beispielsweise die Schwelle der Farbstärke des Erkennungsalgorithmus erhöht und danach erneut ausgeführt.

Auf diese Art wurde nach und nach um Objektverfolgung durch Kategorisierung und Erkennung der Form erweitert und die gut funktionierende Objekterkennung entwickelt wie sie nun in Betrieb ist.

Ebenso verhielt es sich mit der Ansteuerung der Motoren. Zunächst wurde der Grundaufbau des Roboters zum Testen mit einer im Play-Store erhältlichen allgemeinen Fernsteuerungs-App für NXT-Roboter „NXT Remote Control“ [12] verwendet. Sie bietet nach Verbindungsaufnahme Bedienelemente zum ansteuern jedes Motors an, was erste Fahrversuche des CLEEN-R

Roboters erlaubte. So wurde auch festgestellt, das der bestehende Aufbau die gestellten Anforderungen in Hinsicht auf Bewegungsfreiheit und Fähigkeit zur Objektaufnahme erfüllt und das Fahrgestell oder der Greifarm nicht weiter verändert werden muss.

Funktionierten mal Motoren oder die Bluetooth-Kommunikation nicht mehr, konnte mit NXT Remote Control auch ständig geprüft werden, ob der Fehler in der Hardware (Stromversorgung unzureichend, Bluetooth am NXT-Modul deaktiviert) oder in der Software (falsche Geschwindigkeit, abgebrochene Verbindung) lag und dementsprechend behoben werden.

Schließlich konnte so durch Ausprobieren, Optimieren und Erweitern unabhängig voneinander Steuerung und Objekterkennung nach und nach erarbeitet, letztendlich zusammengefügt und mit dem Roboter-Aufbau getestet werden.

## 7.2 Manuelle Steuerung

Zum Testen der Fahrfähigkeit, des Greifarms und hauptsächlich der Positionsverfolgung wurde unter dem Einstellungsmenü der CLEEN-R App der Menüpunkt „manual control“ hinzugefügt. Über ihn wird die in Bild 7.1 gezeigte Activity gestartet, die dem Benutzer oder Tester die Kontrolle über die Bewegungen des Roboters gibt.

Das Smartphone wird für die direkte Kontrolle vom Roboter in die Hand genommen, die Übertragung der Befehle erfolgt wie im Normalbetrieb via Bluetooth.

Über Buttons kann sowohl vor- und rückwärts gefahren, als auch im und gegen den Uhrzeigersinn rotiert werden. Des Weiteren lässt sich der Greifarm beliebig weit öffnen und schließen. Die aktuelle Position und der Winkel im Vergleich zum Start wird in Koordinatenform angezeigt und während der Bewegungen aktualisiert.

Im unteren Teil des Bildschirms befindet sich eine graphische Repräsentation des Roboters und des bereits zurückgelegten Weges durch das Zimmer, der Startpunkt wird ebenfalls markiert.

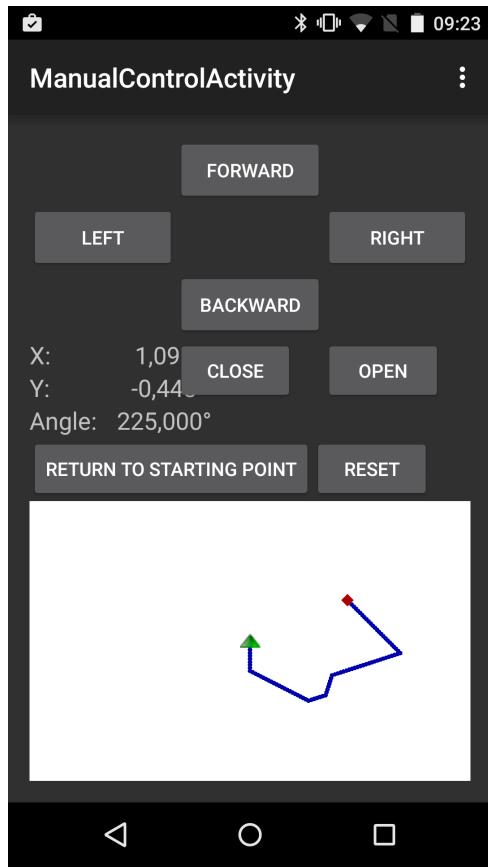


Abbildung 7.1: Manuelle Steuerung des Roboters

Nach dem manuellen Anfahren und Aufnehmen eines Objekts kann über den Knopf „Return to starting point“ der Roboter wie in Kapitel 6.8 vollautomatisch zum Startpunkt zurückgefahren werden. Über „Reset“ wird die aktuelle Position des NXT-Roboters zurückgesetzt und als neuer Startpunkt festgelegt.

# 8 Zusammenfassung und Ausblick

Im Rahmen dieser Studienarbeit wurde das Robotersystem CLEEN-R entwickelt. Ziel war es mit Hilfe eines LEGO® Mindstorm NXT-Robotersets und eines Google Nexus 5 Android Smartphone ein Robotersystem zu entwickeln, welches kameragestützt Objekte erkennen, kategorisieren und transportieren sollte.

In den vorhergehenden Kapiteln wurde zunächst die konkrete Anwendung der Hardware beschrieben. Daraufhin ist die Konstruktion des Roboters geschildert. In den darauf folgenden Kapiteln sind genaue Algorithmen zur Bildverarbeitung, sowie der allgemeine Arbeitszyklus des Systems beschrieben. Zuletzt folgen Tests in geschützten Raum, sowie Realtests.

## 8.1 Zusammenfassung

Wie aus Kapitel 7 ersichtlich, konnte ein Robotersystem konstruiert werden, welches erfolgreich Gegenstände „aufräumen“ kann. Das System besteht aus zwei getrennten Modulen: Ein LEGO Mindstorm NXT-Roboter und ein Google Nexus 5 Android-Smartphone kommunizieren über eine Bluetooth-Schnittstelle mit einander und steuern so die Aktoren des Roboters.

Es konnte ein Großteil der Komplexität des Systems dadurch reduziert werden, dass das Smartphone als zentrale Steuereinheit benutzt wird und das NXT-System lediglich als Vermittler zu Sensoren und Aktoren genutzt wird. Der Roboter basiert auf einem modifizierten Bauplan, der von LEGO zur Verfügung gestellt wurde. Dieser Bauplan wurde dahingehend angepasst, dass einige Sensoren entfernt und durch eine Halterung für das Smartphone ersetzt wurden.

Die Implementierung erfolgte mit der OpenCV-Bibliothek in einer Android-Applikation. Hierbei wurden verschiedene Verfahren der Bildverarbeitung eingesetzt um Objekte zu erkennen. Konkret wird in den Kameraaufnahmen nach Objekten mit hoher Farbsättigung gesucht. Bilder werden anschließend binarisiert und mit Hilfe von Algorithmen zur Kantenverfolgung segmentiert.

Ist ein Objekt erkannt, so fährt der Roboter es an und nimmt es mit einem Greifarm auf. Das Ansteuern des Objekts geschieht hierbei mit einer Kombination aus Sensordaten aus einem Ultraschallsensor und der Kamera des Smartphones. Wenn das Objekt erfolgreich aufgenommen wurde, wird es auf Grund seiner Farbe und Form kategorisiert. Je nach Kategorie, wird eine andere Zielzone ausgesucht und lokalisiert.

Ist eine Zielzone bestimmt worden, transportiert der Roboter den aufgenommenen Gegenstand dort hin und legt ihn ab. Anschließend begibt er sich zurück in die Startposition und beginnt die Suche von vorne. Die Orientierung im Raum wurde dabei über einen Positionsverfolgungsansatz gelöst, welcher die gefahrene Strecke berechnet um die aktuelle Position zu bestimmen.

## 8.2 Bewertung der Ergebnisse

Die Hauptaufgabe wurde erfolgreich gelöst. Es wurde ein Robotersystem entwickelt, welches alle in der Problemstellung beschriebenen Aufgaben erfolgreich Lösen kann.

Bei der Entwicklung traten an vielen Stellen, wie beispielsweise der in Kapitel 6.2.1 beschriebenen Entfernungsschätzung, schwierige Probleme auf, welche durch zum Teil sehr elegante Lösungswege gelöst werden konnten. Es wurden stets mehrere Möglichkeiten abgewägt, wobei sich an einigen Stellen gezeigt hat, dass zusätzliche Hardware eine elegantere Lösung darstellt. Es ist jedoch auch zu beachten, dass diese immer einen zusätzlichen Entwicklungsaufwand mit sich führt und zusätzliche Mittel in Anspruch nimmt, die bei einem kleineren Roboterprojekt oft nicht vorhanden sind.

Wie Kapitel 7 zeigt, funktioniert die Koppelung des Smartphones als Kameramodul mit dem NXT-Roboter. Die zentrale Designentscheidung das Smartphone als Steuereinheit zu benutzen und den Roboter lediglich ausführende Tätigkeiten verrichten zu lassen hat sich als sehr gewinnbringend herausgestellt. Durch die geringe Rechenleistung des NXT-Steins wäre eine effiziente Berechnung von Algorithmen der Bildverarbeitung nicht möglich gewesen.

Über das Verhalten des Roboters lässt sich sagen, dass es noch sehr abgehackt wirkt. Bewegungen sind nicht flüssig, Drehungen oft langsam und das Positionstracking manchmal ungenau. All diese Einflüsse erwecken den Eindruck der Roboter sei unfertig. In nachfolgenden Abschnitt wird ein Ausblick über weitere Vorgehensweisen bei der Bearbeitung dieser Themen gewährt.

## 8.3 Ausblick

Ein Hauptproblem des aktuellen Robotersystems stellt das „abgehackte“ Fahren dar. Der Roboter fährt nie eine Kurve, sondern hält an und dreht sich auf der Stelle. Dies erleichtert das Positionstracking und das Zentrieren von Objekten ohne das Risiko sie zu Überfahren, ist jedoch zeitlich ineffizient und wird von Beobachtern als „unschön“ empfunden. Ein flüssigeres Verhalten kann durch den ausgefeilteren Einsatz des Ultraschallsensors erreicht werden. Hierbei kann die Geschwindigkeit des Heranfahrens an ein Objekt an die gemessene Entfernung angepasst werden.

Eine weitere Möglichkeit der Erweiterung des Systems stellt die Verwendung einer zweiten Kamera dar. Diese zweite Kamera kann zwei mögliche Anwendungen haben. Sie könnte als zusätzliches Kameramodul auf dem Roboter Tiefeninformationen aus Bildern erzeugen und so eine Kartografierung der Umgebung beginnen. Der Roboter könnte so als lernendes System Räume einstudieren und zu transportierende Gegenstände als „Fremdkörper“ im Raum identifizieren. Alternativ könnte eine zweite Kamera als externer Beobachter des Systems verwendet

werden. Dies würde wie in Kapitel ?? beschrieben zu einer genaueren Positionserkennung beitragen und das System somit stabiler gegen Ungenauigkeiten der Rotationssensoren machen.

Insgesamt lässt sich sagen, dass noch einige offene Nahtstellen im Projekt vorhanden sind, an denen zukünftige Entwickler Erweiterungen und Verfeinerungen anbringen können. Durch die modulare Struktur der Applikation lässt sich nahezu der gesamte Code ohne große Änderungen für weitere Projekte benutzen.

## 8.4 Reflexion des Gelernten

# Literaturverzeichnis

- [1] G. Bradski. Open cv. *Dr. Dobb's Journal of Software Tools*, 2000.
- [2] Freie nxt bauanleitungen, 2015. <http://dienxteebene.blogspot.de/p/freie-bauanleitungen-im-netz.html>. Zuletzt besucht 08.05.2015.
- [3] Prakash Ranganathan, Richard Schultz, and Mojdeh Mardani. Use of lego nxt mindstorms brick in engineering education. In *Proceedings of the 2008 ASEE North Midwest Sectional Conference*, pages 17–19, 2008.
- [4] LEGO GROUP et al. Lego mindstorms nxt communication protocol. *LEGO MindStorms NXT Bluetooth Developer Kit*, 2006.
- [5] Libccv, 2015. <http://libccv.org/>. Zuletzt besucht 08.05.2015.
- [6] LLC ImageMagick Studio. Imagemagick, 2008. <http://www.imagemagick.org>. Zuletzt besucht 08.05.2015.
- [7] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. Ö'Reilly Media, Inc.", 2008.
- [8] Rita Cucchiara, Costantino Grana, Massimo Piccardi, Andrea Prati, and Stefano Sirotti. Improving shadow suppression in moving object detection with hsv color information. In *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 334–339. IEEE, 2001.
- [9] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

- [10] Florian Bähge. *3D-Objektverfolgung mit Stereokameras zur bildbasierten Navigation autonom fliegender Luftfahrzeuge*. PhD thesis, Bachelor's thesis, DLR/Universität Magdeburg, 2012.
- [11] Joachim Hertzberg, Kai Lingemann, and Andreas Nüchter. *Mobile Roboter*. Springer-Verlag, 2012.
- [12] Jacek Fedoryński. Nxt remote control, google play store, 2015. <https://play.google.com/store/apps/details?id=org.jfedor.nxtremotecontrol&hl=de>. Zuletzt besucht 08.05.2015.