



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

6. Mai 2015

**Bearbeitungszeitraum:** 2 Semester

**Matrikelnummer:** 8853105 & 1906813

**Kurs:** TINF12B3

**Studiengang:** Informationstechnik

**Ausbildungsfirma:** Karlsruher Institut für Technologie

**Betreuer:** Gertrud Nieder

# Inhaltsverzeichnis

<b>Eidesstattliche Erklärung</b>	<b>IV</b>
<b>Abkürzungsverzeichnis</b>	<b>1</b>
<b>1 Einleitung</b>	<b>3</b>
<b>2 Problemstellung</b>	<b>4</b>
<b>3 Materialien und Methoden</b>	<b>5</b>
<b>4 Hardwareumsetzung</b>	<b>6</b>
4.1 Entwurf des NXT-Roboters . . . . .	6
4.2 Steuerung des Roboters . . . . .	8
4.3 Wahl des Kameramoduls . . . . .	9
<b>5 Fundamentale Softwareumsetzung</b>	<b>10</b>
5.1 Wahl der Bildverarbeitungsbibliothek . . . . .	10
5.2 Algorithmen zur Objekterkennung . . . . .	11
5.3 Struktur der Android-App . . . . .	13
<b>6 Arbeitsablauf und Problematiken</b>	<b>14</b>
6.1 Objektsuche . . . . .	14
6.2 Anfahren von Objekten . . . . .	17
6.3 Aufnehmen von Objekten . . . . .	21
6.4 Kategorisierung von Objekten . . . . .	21

6.5	Suche des Zielbereichs . . . . .	22
6.6	Ansteuerung des Zielbereichs . . . . .	23
6.7	Ablegen von Objekten . . . . .	23
6.8	Rückkehr zur Ausgangsposition . . . . .	23
<b>7</b>	<b>Tests des Robotersystems</b>	<b>24</b>
7.1	Tests im gesicherten Rahmen . . . . .	24
7.2	Realtests . . . . .	24
<b>8</b>	<b>Zusammenfassung und Ausblick</b>	<b>25</b>
	<b>Literaturverzeichnis</b>	<b>V</b>

## **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

# Abkürzungsverzeichnis

# Abbildungsverzeichnis

5.1	Beispiel einer Aufspaltung in RGB-Kanäle (links) und HSV-Kanäle (rechts) . . . . .	12
5.2	Beispiel der Binarisierung des Sättigungskanals . . . . .	12
5.3	Beispiel einer Segmentierung des binarisierten Sättigungskanals . . . . .	13
6.1	Petrinetz des Arbeitsablaufs . . . . .	14
6.2	Objektverfolgung durch Kriterium der lokalen Nähe . . . . .	15
6.3	Objektverfolgung durch Größenkriterium . . . . .	15
6.4	Objektverfolgung durch farbliche Ähnlichkeit . . . . .	16
6.5	Verzerrte Aufnahme durch Bewegung des Roboters . . . . .	17
6.6	Zentrierung eines Objekts . . . . .	18
6.7	Stereokamerabasierte Entfernungsschätzung . . . . .	19
6.8	Monokamerabasierte Entfernungsschätzung . . . . .	20
6.9	Screenshot der Applikation bei Rückkehr zur Ausgangsposition . . . . .	23

# 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-Library [1] implementiert werden. Das Projekt wurde „CLEEN-R“ getauft. Der Name stellt hierbei ein Akronym für „Cleaning and Lifting Enviroment Evaluating NXT - Robot“ und stellt 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, sowie die notwendigen Grundlagen der Bildverarbeitung, die für spätere Methoden genutzt werden. Die beiden darauf folgenden Kapitel gehen gesondert auf den genauen Aufbau des Roboters samt Konstruktionsplänen, sowie Details zur Softwareimplementierung der Objekterkennung und der Roboteransteuerung ein. Kapitel 7 beschreibt durchgeführte Tests sowohl unter speziell präparierten Bedingungen, als auch Realbedingungen. Das abschließende Kapitel ist ein letztes Fazit, welches einen Überblick über die gesamte Arbeit bildet.

## 2 Problemstellung

Ziel dieser Studienarbeit ist die praktische Anwendung gelernter Kenntnisse in Hard- und Software. Durch die Benutzung zweier verschiedener 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 drei Teilprozesse unterteilt werden.

Erstens muss der Roboter mit Hilfe von bekannten Verfahren der Bildverarbeitung Objekte auf Grund ihrer physikalischen Beschaffenheit, beispielsweise ihrer Größe, ihrer Form und ihrer Farbe, erkennen und zielgerichtet anfahren.

Zweitens soll sich der Roboter gegenüber des Gegenstands optimal positionieren und diesen mit Hilfe eines mechanischen Greifarms aufnehmen.

Zuletzt muss der Roboter den aufgenommenen Gegenstand kategorisieren, nach der entsprechenden Zielzone suchen diese Anfahren und den Gegenstand ablegen. Die Erkennung der Zielzone kann dabei durch Markierungen an Wänden und Böden des Raumes erfolgen.

Als Bewertungskriterien dienen hierbei beispielsweise ob der Roboter alle Gegenstände erfolgreich erkennt, diese korrekt kategorisiert und in korrekte Zielzonen bewegt, sowie die Zeit in der dies geschieht.

# 3 Materialien und Methoden

Dieses Kapitel vielleicht löschen?

NXT-Roboter

Android-Smartphone

Bluetooth-Verbindung

Some more Stuff

# 4 Hardwareumsetzung

## 4.1 Entwurf des NXT-Roboters

Die hardwareseitigen Voraussetzungen an den Roboter bestanden im Hauptsächlichen aus der freien Bewegung im Raum und dem Aufnehmen, Mitführen und Ablegen von kleinen Gegenständen in einem vordefinierten Bereich.

Nach kurzer Recherche[2] und Durchsicht von Bauanleitungen für verschiedenste Anwendungsbereiche wurde sich für den Standardaufbau aus der zum Bauset zugehörigen LEGO NXT Bauanleitung entschieden.

Sie wurde lediglich um den Schall- und den Abstandssensor erleichtert; eine Halterung für das Smartphone wurde hinzugefügt.

**1.hier Bild des Roboters einfügen**

### 4.1.1 Sensoren

#### Tastsensor

Der berührungsempfindliche Sensor vorne dient zum Detektieren von Gegenständen im Bereich des Greifarms, woraufhin dieser geschlossen werden kann.

#### Rotationssensoren

Die Rotationssensoren in den Servomotoren 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.

**2.Farbsensor?**

### 4.1.2 Aktoren

#### Antriebsmotoren

Die beiden Servomotoren links und rechts des NXT-Roboters bilden den differentiellen Antrieb und ermöglichen freie Fortbewegung.

#### Greifarmmotor

Der dritte Motor im vorderen Teil des Roboters dient zum Öffnen und Schließen des Greifarms und so zur Mitführung von Gegenständen.

## 4.2 Steuerung des Roboters

Die Steuerung des Roboters durch das Smartphone erfolgt via Bluetooth. Das Kommunikationsprotokoll und damit die nötigen Befehle zum Regeln der Aktoren und Auslesen der Sensoren wurde von LEGO dokumentiert und ist online erhältlich[3].

Auf dem NXT selbst wird hierbei kein Programm ausgeführt, um alle Logik zentral in der NXT-App auf dem Smartphone zu halten.

Zunächst muss eine Bluetooth-Verbindung erstellt werden, wozu beide Geräte aktiviertes Bluetooth aufweisen, der Roboter zusätzlich sichtbar für das Smartphone sein müssen.

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

Kommt eine Verbindung zustande, können seriell Byte für Byte die Kommandos an den NXT übertragen, eventuelle Antworten empfangen werden.

App-seitig übernimmt ein gesonderter Thread in der Klasse NxtTalker nach Zustandekommen einer Verbindung das Management der Daten.

Zum Bewegen der Motoren muss zunächst per Befehl pro Aktor eine Geschwindigkeit (und Parameter wie Regulierung) übergeben, zum Stoppen können alle Motoren mit einem Befehl auf Geschwindigkeit '0' gesetzt werden.

Zwei Motoren können synchronisiert werden, sodass diese gleichzeitig starten. Ansonsten würde der Zeitversatz zwischen dem Absetzen der zwei 'setze Geschwindigkeit'-Befehle bewirken, dass der Roboter vor dem geradeaus fahren kurz nur das erste Rad ansteuert und in eine Richtung abdriftet.

### 4.3 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 Entfernungsrechnung 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.

# 5 Fundamentale Softwareumsetzung

Mit Hilfe des in Kapitel 4.3 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 [4] 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 [5] 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, 6] 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 [ZITAT EINFÜGEN] in das HSV-Format [ZITAT] konvertiert. Dies wird durchgeführt, da das HSV-Format unempfindlicher gegen Veränderungen in der Beleuchtung ist als das RGB-Format. 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.

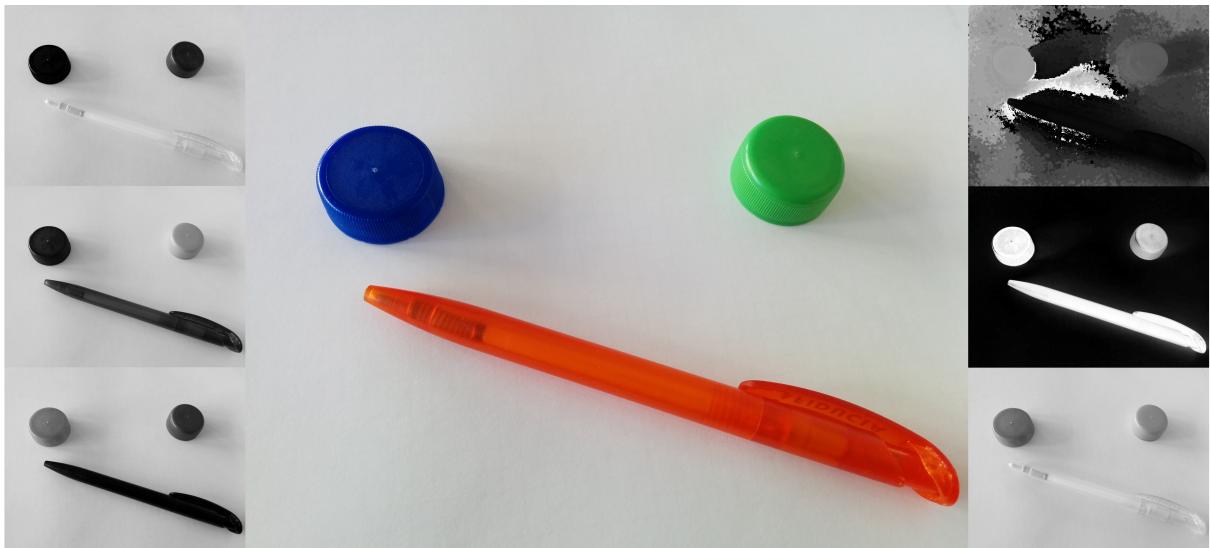


Abbildung 5.1: Beispiel einer Aufspaltung in RGB-Kanäle (links) und HSV-Kanäle (rechts)

Zuletzt erfolgt eine Binarisierung mit einem empirisch ermittelten Schwellwert von 50. Dies ist notwendig, da die, in der OpenCV-Bibliothek implementierte, Methode von Suzuki und Abe [7] 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.

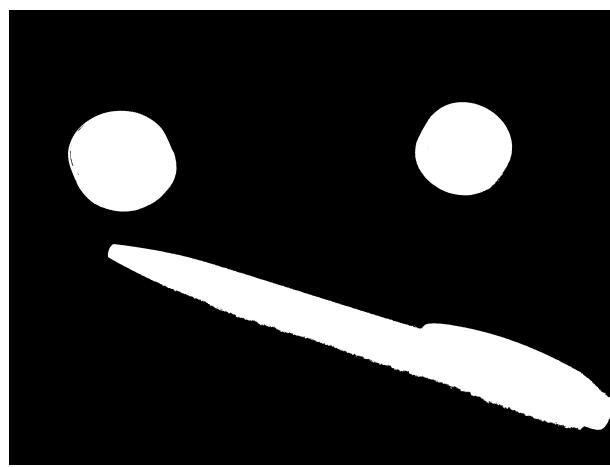


Abbildung 5.2: Beispiel der Binarisierung des Sättigungskanals

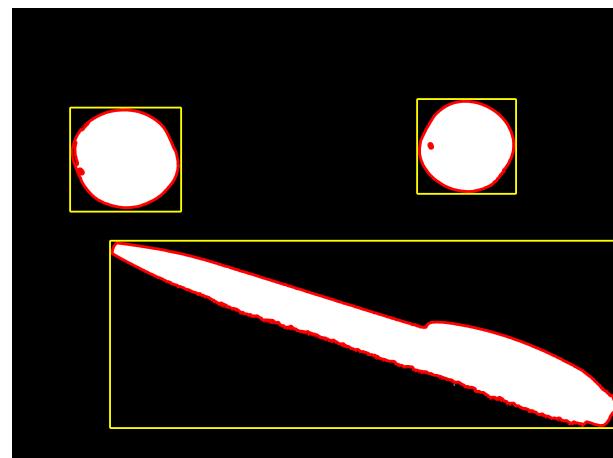


Abbildung 5.3: Beispiel einer Segmentierung des binarisierten Sättigungskanals

## 5.3 Struktur der Android-App

### 5.3.1 Activities

Aktivitätsdiagramm

# 6 Arbeitsablauf und Problematiken

Der Arbeitsablauf des Roboters kann in unterschiedliche Phasen unterteilt werden. Diese Phasen sind als in Abbildung als Petrinetz beschrieben.

Abbildung 6.1: Petrinetz des Arbeitsablaufs

Die Arbeitsweise sowie auftretende Probleme sind in den folgenden Abschnitten beschrieben.

## 6.1 Objektsuche

Sobald ein Objekt durch die in Kapitel 5.2 dargestellten Algorithmen erkannt wurde, muss sichergestellt werden, dass die Fokussierung auf dieses Objekt beibehalten werden kann. Ein zufälliges Umspringen auf zufällige 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 Bildern erkannt werden kann.

### 6.1.1 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 wieder gefunden werden kann.

### 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 also rein aus ihrer Position wiedergefunden werden. Geschieht Bewegung, so kann nicht mehr von einer exakten Übereinstimmung der Koordinaten ausgegangen werden. Stattdessen muss eine gewisse Toleranz gegeben werden. 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 6.2 zeigt das beschriebene Verhalten anschaulich.

Abbildung 6.2: Objektverfolgung durch Kriterium der lokalen Nähe

### 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 Fläche. 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 werden. Diese kann entweder konstant gegeben sein (beispielsweise 10%) oder abhängig von der aktuellen Geschwindigkeit des Roboters gewählt werden. Abbildung 6.3 zeigt exemplarisch die Verfolgung eines Objektes dessen Größe sich im Verlauf der Aufnahmeserie verändert.

Abbildung 6.3: 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 außschließen 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 6.4 zeigt die farbliche Ähnlichkeit zweier Objekte und die dadurch resultierende Wiederfindung des Objektes in der Folgeaufnahme.

Abbildung 6.4: Objektverfolgung durch farbliche Ähnlichkeit

### 6.1.2 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 apprupter starker Veränderung der Lichtbedingungen, oder lediglich temporär durch Fehler bei der Bildaufnahme geschehen. In diesem Abschnitt wird auf letzteres eingegangen, da das Verhalten bei längerfristigem Verlust wird in Abschnitt ?? beschrieben wird.

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 6.1.1 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 "verlorenängesehen 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.

### 6.1.3 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 6.5 zeigt eine solche verzerrte Aufnahme.

Abbildung 6.5: Verzerrte Aufnahme durch Bewegung des Roboters

Durch die Verzerrung können unterschiedliche Phänomene auftreten. Die Verfolgung von Objekten, die in Abschnitt 6.1.1 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.

## 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 6.6 zeigt, wie ein Objekt durch die Drehung des Roboters zentriert werden konnte.

Abbildung 6.6: 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 6.7 zeigt schematisch eine solche Entfernungsschätzung [8].

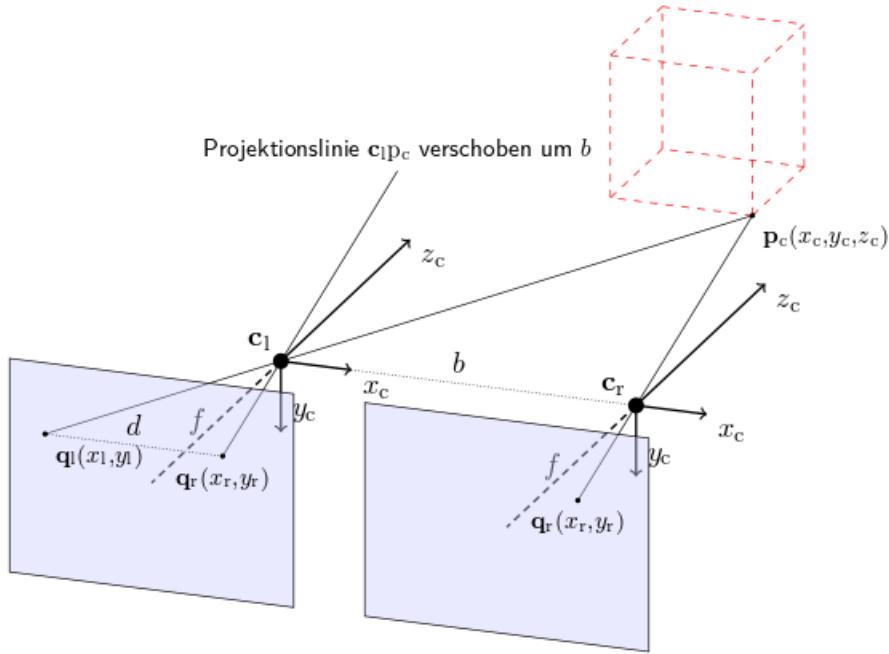


Abbildung 6.7: Stereokamerabasierte Entfernungsschätzung

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.

### 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 Streuwinkels ein virtuelles Blickfeld erstellt. Dieses Blickfeld breitet sich von der

Kamera aus pyramidenförmig aus und trifft den als planar angenommenen Boden. Abbildung 6.8 zeigt den Aufbau eines solchen Systems.

Abbildung 6.8: Monokamerabasierte Entfernungsschätzung

Anhand der Position des detektierten Objektes im Kamerabild lässt sich daraufhin eine ungefähre Entfernungsschätzung liefern. Problematisch ist hierbei die Ungenauigkeit des Systems. 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 von uns durchgeführten Tests, konnte lediglich eine Genauigkeit von  $\pm 5\text{cm}$  erreicht werden. Dies ist für die gegebene Anwendung unzureichend.

### 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 [9]. 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.

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.

## 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 ??.

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.

### 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 6.1 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.

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 6.1: Umrechnung von HSV-Werten in konkrete Farben.

#### 6.4.2 Kategorisierung nach Form

TODO

### 6.5 Suche des Zielbereichs

#### 6.5.1 Orientierung im Raum

Kameragestützt

Streckenbasiert

Über gefahrene Strecke

## 6.6 Ansteuerung des Zielbereichs

## 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

Mit Hilfe der in Kapitel 6.5.1 beschrieben Verfahren kann sich der Roboter im Raum orientieren und den Startpunkt ausfindig machen und ansteuern. Abbildung 6.9 zeigt einen Screenshot der Applikation sobald der Roboter in die Ausgangsposition zurückzukehren versucht.

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.

Abbildung 6.9: Screenshot der Applikation bei Rückkehr zur Ausgangsposition

# 7 Tests des Robotersystems

## 7.1 Tests im gesicherten Rahmen

## 7.2 Realtests

## 8 Zusammenfassung und Ausblick

# Literaturverzeichnis

- [1] G. Bradski. Open cv. *Dr. Dobb's Journal of Software Tools*, 2000.
- [2] Freie nxt bauanleitungen.
- [3] LEGO GROUP et al. Lego mindstorms nxt communication protocol. *LEGO MindStorms NXT Bluetooth Developer Kit*, 2006.
- [4] Libccv.
- [5] Imagemagick.
- [6] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. Ö'Reilly Media, Inc.", 2008.
- [7] 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.
- [8] Florian Bähge. *3D-Objektverfolgung mit Stereokameras zur bildbasierten Navigation autonom fliegender Luftfahrzeuge*. PhD thesis, Bachelor's thesis, DLR/Universität Magdeburg, 2012.
- [9] Joachim Hertzberg, Kai Lingemann, and Andreas Nüchter. *Mobile Roboter*. Springer-Verlag, 2012.