



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

10. Mai 2015

Bearbeitungszeitraum: 2 Semester

Matrikelnummer: 8853105 & 1906813

Kurs: TINF12B3

Studiengang: Informationstechnik

Ausbildungsfirma: Karlsruher Institut für Technologie

Betreuer: Gertrud Nieder

Inhaltsverzeichnis

Eidesstattliche Erklärung	IV
1 Einleitung	3
2 Problemstellung	4
2.1 Objekterkennung	4
2.2 Mechanische Umsetzung des Objekttransports	4
2.3 Kategorisierung und Wahl der Zielzone	5
2.4 Transport in die Zielzone	5
2.5 Bewertung des Ablaufs	5
3 Materialien und Methoden	6
3.1 Wahl des Robotersystems	6
3.2 Wahl des Kameramoduls	9
4 Hardwareumsetzung	11
4.1 Entwurf des NXT-Roboters	11
4.2 Steuerung des Roboters	15
5 Softwareumsetzung	17
5.1 Wahl der Bildverarbeitungsbibliothek	17
5.2 Algorithmen zur Objekterkennung	18
5.3 Ortsbestimmung	20
5.4 Struktur der Android-App	22

Inhaltsverzeichnis	III
6 Arbeitsablauf und Problematiken	23
6.1 Objektsuche	23
6.2 Anfahren von Objekten	28
6.3 Aufnehmen von Objekten	33
6.4 Kategorisierung von Objekten	34
6.5 Suche des Zielbereichs	35
6.6 Ansteuerung des Zielbereichs	36
6.7 Ablegen von Objekten	36
6.8 Rückkehr zur Ausgangsposition	36
7 Tests des Robotersystems	37
7.1 Tests im gesicherten Rahmen	37
7.2 Realtests	37
8 Zusammenfassung und Ausblick	38

Literaturverzeichnis	V
-----------------------------	----------

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

Abbildungsverzeichnis

3.1	Hauptbestandteile des LEGO Minstorms NXT-Systems	7
3.2	Hauptbestandteile des LEGO Minstorms EV3-Systems	8
3.3	Kameramodul des Nexus 5	10
4.1	Von Lego bereitgestellter Roboteraufbau	11
4.2	NXT-Stein	12
4.3	Tastsensor des NXT-Systems	13
4.4	Farbsensor des NXT-Systems	14
4.5	Ultraschallsensor des NXT-Systems	14
4.6	Servo-Motor des NXT-Systems	15
4.7	Bluetooth-Verbindung zwischen NXT und Nexus 5	16
5.1	Beispiel einer Aufspaltung in RGB-Kanäle (links) und HSV-Kanäle (rechts) . .	19
5.2	Beispiel der Binarisierung des Sättigungskanals	19
5.3	Beispiel einer Segmentierung des binarisierten Sättigungskanals	20
6.1	Petrinetz des Arbeitsablaufs	23
6.2	Auffindung eines Fokusobjekts	24
6.3	Objektverfolgung durch Kriterium der lokalen Nähe	25
6.4	Objektverfolgung durch Größenkriterium	26
6.5	Objektverfolgung durch farbliche Ähnlichkeit	27
6.6	Verzerrte Aufnahme durch Bewegung des Roboters	28
6.7	Zentrierung eines Objekts	29
6.8	Stereokamerabasierte Entfernungsschätzung	30

6.9	Aufbau der monokamerabasierten Entfernungsschätzung	31
6.10	Exemplarische monokamerabasierte Entfernungsschätzung	32
6.11	Aufnahme aus der monokamerabasierten Entfernungsschätzung	33
6.12	Screenshot der Applikation bei Rückkehr zur Ausgangsposition	36

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 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 auftraten, sowie Designentscheidungen zur Lösung dieser. Kapitel ?? 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 Fokus-

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



Abbildung 3.1: Hauptbestandteile des LEGO Minstorms NXT-Systems

An den Hauptstein können bis zu drei Motoren und bis zu vier Sensoren angeschlossen werden. Kommuniziert wird hierbei im I²C-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 I²-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

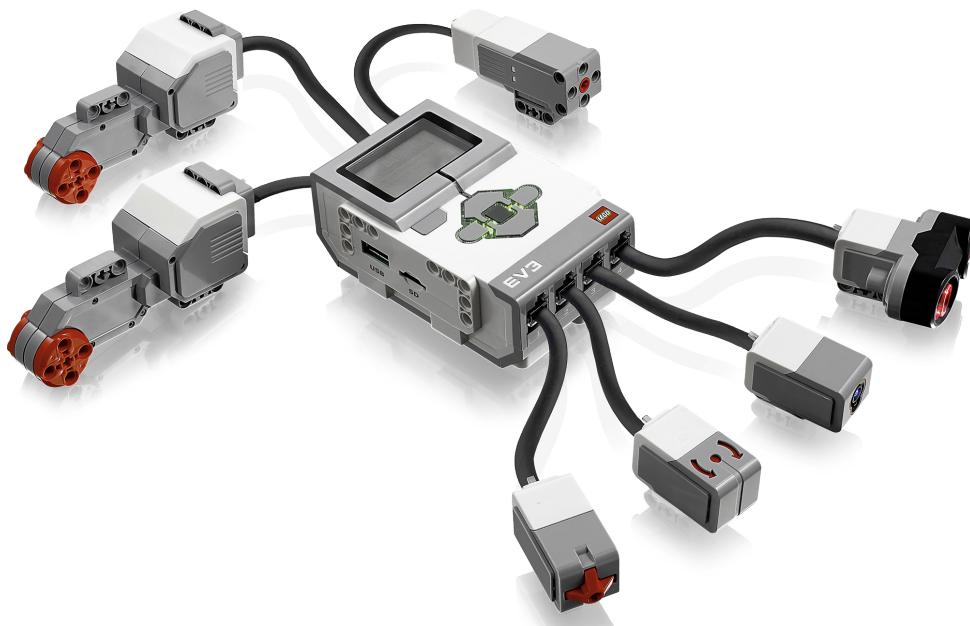


Abbildung 3.2: Hauptbestandteile des LEGO Minstorms EV3-Systems

EV3 stellt die Nachfolge-Kollektion von NXT dar. Die Hauptänderung besteht in Ersetzen des ARM7-Prozessor durch einen ARM9-Prozessor, auf dem eine Linuxumgebung ausgeführt wird.

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 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.

Ein Problem bei der Kamera des Nexus ist, dass 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.



Abbildung 3.3: Kameramodul des Nexus 5

4 Hardwareumsetzung

4.1 Entwurf des NXT-Roboters

Der Roboter hat einige hardwareseitigen Voraussetzungen. 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.

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.

1.Bild von Standard Roboter einfügen

Abbildung 4.1: Von Lego bereitgestellter Roboteraufbau

Da der Aufbau jedoch nicht allen Anforderungen genüge tut, wurde dieser abgeändert.

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

2.hier Bild des Roboters einfügen

4.1.1 NXT-Stein



Abbildung 4.2: NXT-Stein

Der NXT-Stein bildet die Recheneinheit und damit die Hauptkomponente des NXT-Robotersystems. Er besitzt oben drei Ausgänge für Motoren über die gleichzeitig die Rotationssensoren in den Servo-Motoren ausgelesen werden.

Unten 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.

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 auf die Beine stellen zu können. Im Fall des CLEEN-R-Aufbaus wird jedoch lediglich die interne Bluetooth-Verbindung genutzt.

Vorne befindet sich ein 100 mal 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.

4.1.2 Sensoren

Tastsensor



Abbildung 4.3: Tastsensor des NXT-Systems

Der berührungsempfindliche Sensor vorne diente zum Detektieren von Gegenständen im Bereich des Greifarms, woraufhin dieser geschlossen werden kann. Er wurde durch den Ultraschallsensor ersetzt, der nicht nur die unmittelbare Berührung bemerkt sondern auch ein herannahendes Objekt während der Fahrt detektiert.

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.

Farbsensor

Der RGB-Sensor am Boden des NXT dient dazu, die Farbe des Bodens herauszufinden. Diese kann sowohl dazu genutzt werden, bei der Bildverarbeitung die Hintergrundfarbe herauszurechnen und so die Qualität der Objekterkennung zu erhöhen, als auch eine farblich markierte Zielzone im Raum finden zu können, auf der ein getragenes Objekt abgelegt werden kann.



Abbildung 4.4: Farbsensor des NXT-Systems

Ultraschallsensor



Abbildung 4.5: Ultraschallsensor des NXT-Systems

Der Ultraschallsensor dient zur Messung der Distanz vom Roboter zum nächsten soliden Objekt. Hier wird er genutzt um zu detektieren, wie weit ein Objekt vom Greifarm entfernt ist, um diesen im geeigneten Moment zu schließen und so das Objekt mitnehmen zu können. Auch ermöglicht er ein geregeltes Heranfahren an ein Objekt. Die Präzision des Distanzsensors beträgt $\pm 3\text{cm}$.

4.1.3 Aktoren

Antriebsmotoren

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



Abbildung 4.6: Servo-Motor des NXT-Systems

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. Der NXT-Roboter stellt hierzu als Bluetooth-Gerät das Serial Port Profile (SPP) bereit, welches quasi 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[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.



Abbildung 4.7: Bluetooth-Verbindung zwischen NXT und Nexus 5

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 im Bereich von -100% bis 100% (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.

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 [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.

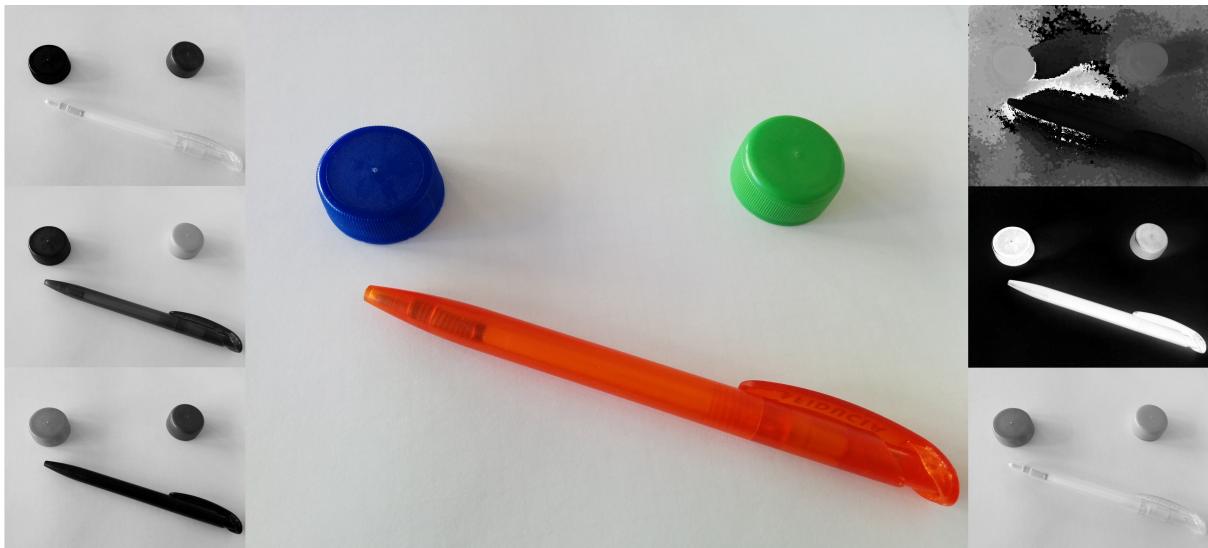


Abbildung 5.1: Beispiel einer Aufspaltung in RGB-Kanäle (links) und HSV-Kanäle (rechts)

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.

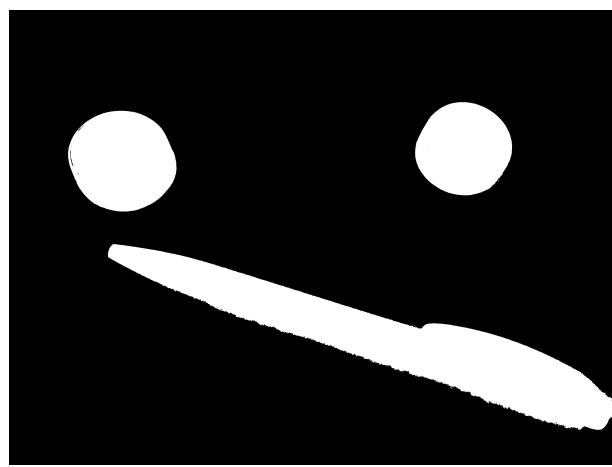


Abbildung 5.2: Beispiel der Binarisierung des Sättigungskanals

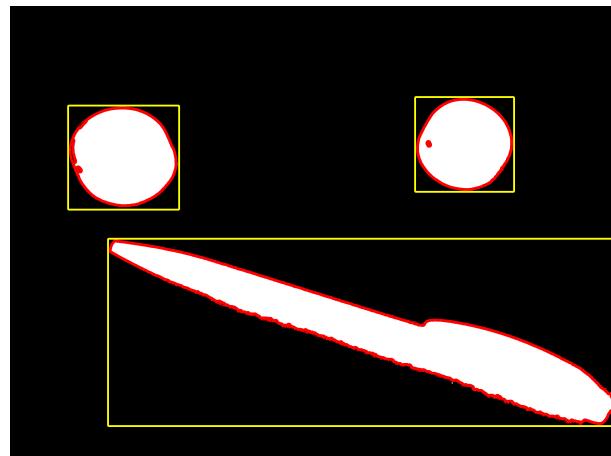


Abbildung 5.3: Beispiel einer Segmentierung des binarisierten Sättigungskanals

5.3 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 Arten der Positionsbestimmung

Mitteilung von Beobachterkomponente

Ist im Raum ein weiteres System vorhanden, dessen Position konstant ist und welche den Roboter stets im Blick behält (über Kameras, Ultraschall, o. Ä.), ist eine simple und doch genaue Positionsbestimmung möglich. Der Roboter muss hierbei keine Berechnungen anstellen und verlässt sich für die korrekte Positionsbestimmung ganz auf den Beobachter, von welcher jederzeit die Lage des Roboters im Raum abrufbar ist.

Raumanalyse

Hierbei werden die Kamerabilder analysiert und aus ihnen Fixpunkte/-geraden des Raumes extrahiert, mit welchen sich der Roboter bei Bewegung im Raum zu orientieren versucht. Dies setzt jedoch relativ viel Rechenzeit bei der Bildverarbeitung und Implementierungsarbeit voraus.

Auch müssen im Raum genügend Orientierungspunkte vorhanden sein. Verliert der Roboter die Orientierung, ist es nicht ohne weiteres möglich diese wiederzuerlangen.

Positionsverfolgung

Besitzt der Roboter einen möglichst exakten Sensor zur Beobachtung der eigens zurückgelegten Strecken in jede Richtung, kann er seine Position bestimmen, indem er jede Bewegung aufzeichnet und diesen Bewegungsvektor auf die letzte Position aufaddiert.

Bei Rotationssensoren in den Rädern wird die Genauigkeit dieses Verfahrens durch das Spiel der Räder und der Auflösung der Sensoren beeinträchtigt.

Da bei diesem Projekt keine Beobachterkomponente vorhanden ist und die zusätzliche Auswertung der Bilder zu aufwändig wäre, wurde sich für die Variante der Positionsverfolgung entschieden. Die Auflösung der Rotationssensoren ist mit 1° ausreichend genau. Die automatische Geschwindigkeitsanpassung der NXT-Motoren abhängig des Untergrunds fördern zusätzlich eine exakte Ortsbestimmung über die Positionsverfolgung.

5.4 Struktur der Android-App

5.4.1 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 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.

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.

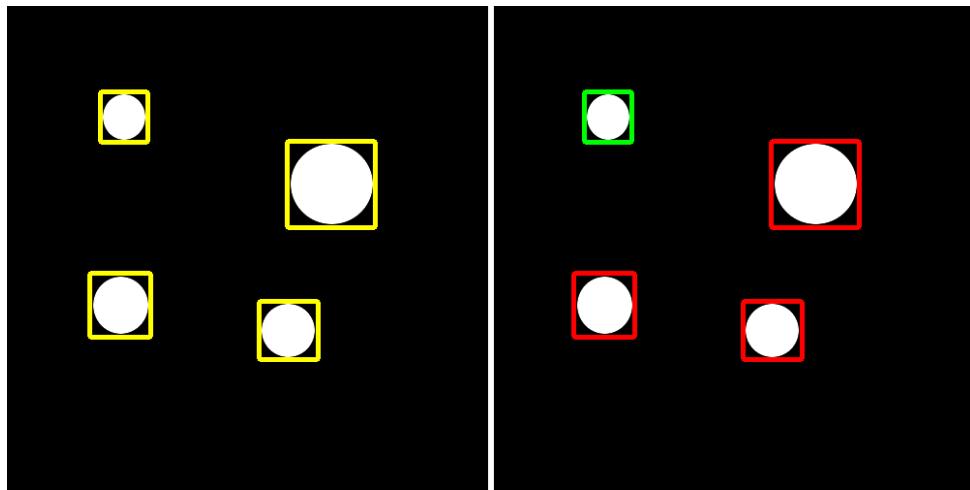


Abbildung 6.2: Auffindung eines Fokusobjekts

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 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 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 6.3 zeigt das beschriebene Verhalten anschaulich am Beispiel einer nachfolgenden Aufnahme aus Abbildung 6.2.

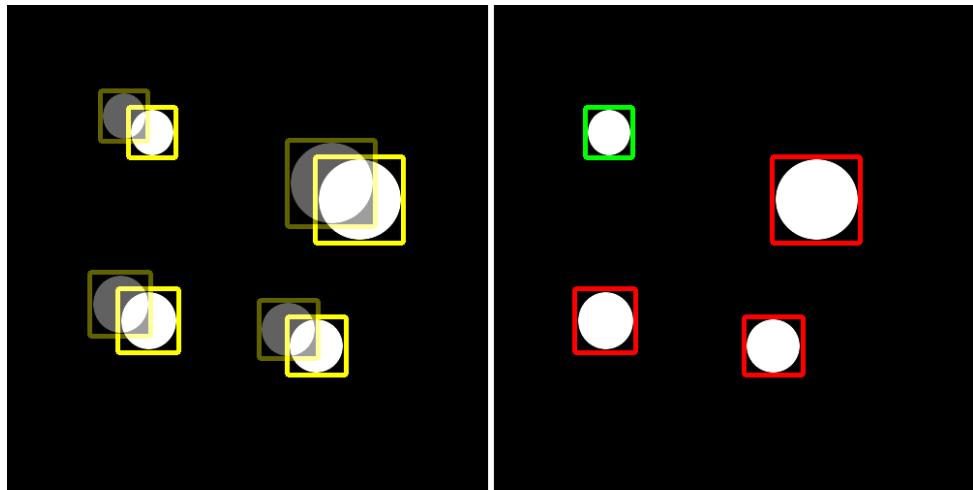


Abbildung 6.3: Objektverfolgung durch Kriterium der lokalen Nähe

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 6.4 zeigt exemplarisch die Verfolgung eines Objektes dessen Größe sich im Verlauf der Aufnahmeserie verändert.

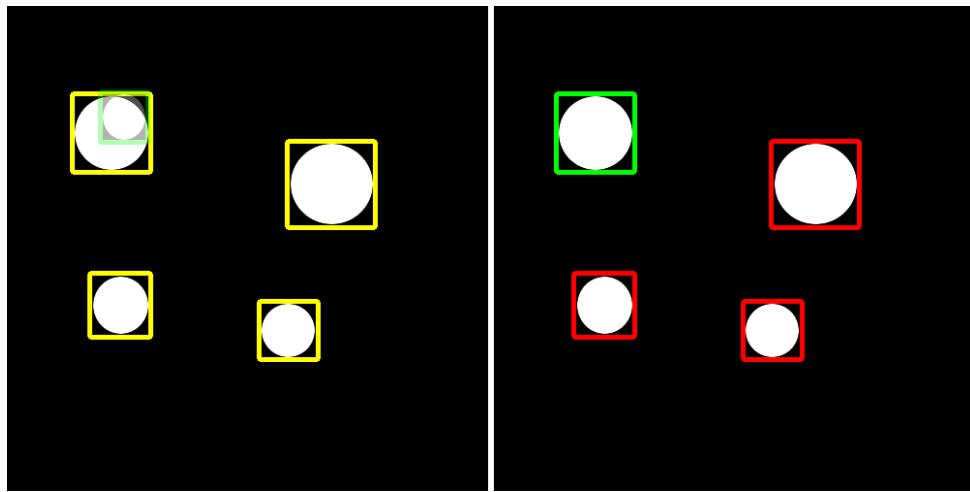


Abbildung 6.4: Objektverfolgung durch Größenkriterium

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 6.5 zeigt die farbliche Ähnlichkeit zweier Objekte und die dadurch resultierende Wiederfindung des Objektes in der Folgeaufnahme.

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 apprupter starker Ver-

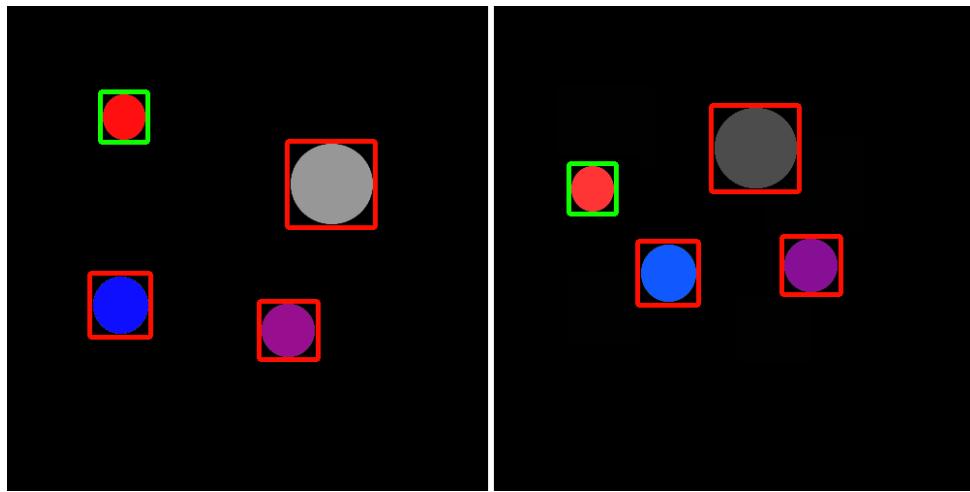


Abbildung 6.5: Objektverfolgung durch farbliche Ähnlichkeit

änderung 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 6.1.2 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 6.6 zeigt eine solche verzerrte Aufnahme.

Abbildung 6.6: Verzerrte Aufnahme durch Bewegung des Roboters

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.

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

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 auf-

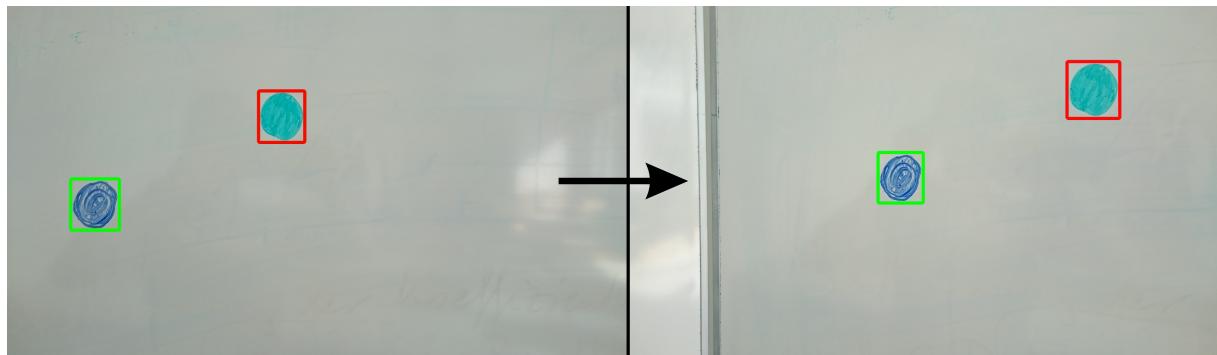


Abbildung 6.7: Zentrierung eines Objekts

weisen. Die folgenden Abschnitte gehen auf verschiedene Methoden der Entfernungsmessung ein.

6.2.1 Entfernungsschätzung

Die Entfernungsschätzung ist für mehrere Teile der Roboteroutine 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 6.8 zeigt schematisch eine solche Entfernungsschätzung [8].

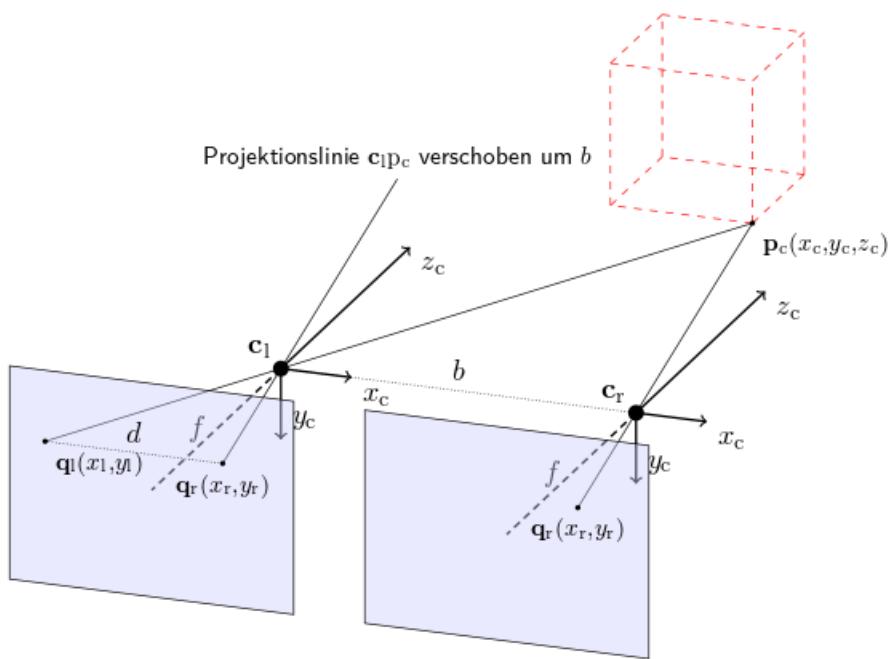


Abbildung 6.8: Stereokamerabasierte Entfernungsschätzung

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 Öffnungswinkels ein virtuelles Blickfeld erstellt. Dieses Blickfeld breitete sich von der Kamera aus pyramidenförmig aus und trifft den als planar angenommenen Boden. Abbildung 6.9 zeigt den Aufbau eines solchen Systems.

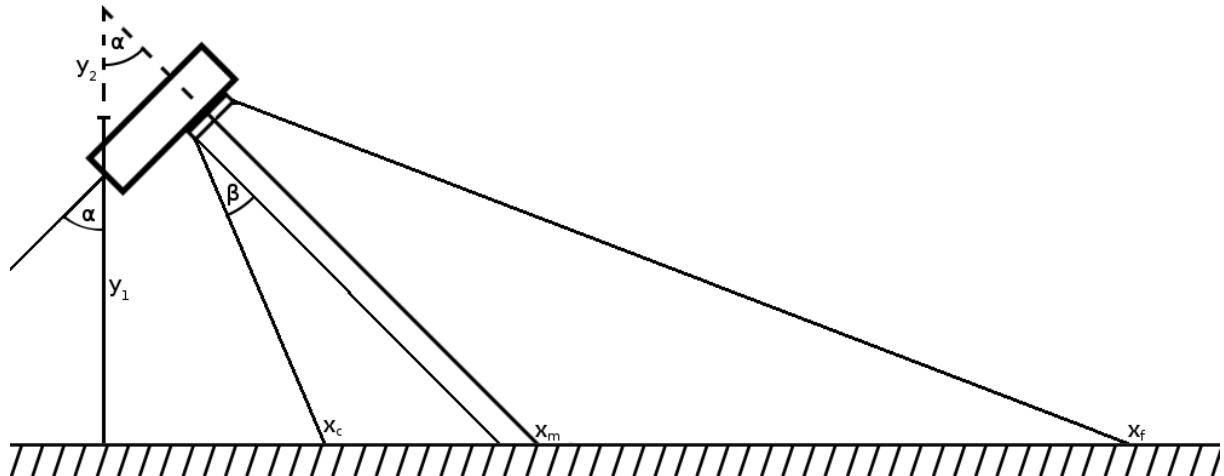


Abbildung 6.9: Aufbau der monokamerabasierten Entfernungsschätzung

Parameter	Wert	Erklärung
y_1	5cm	Kamera „schwebt“ 5cm in der Luft
α	45°	Kamera ist im 45° Winkel zum Boden geneigt
β	40°	Kamera hat 40° Öffnungswinkel

Tabelle 6.1: Parameter der Positionsschätzung

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 6.10 näher ersichtlich.

Abbildung 6.11 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.

Entfernungen lassen sich über einfache trigonometrische Verfahren berechnen.

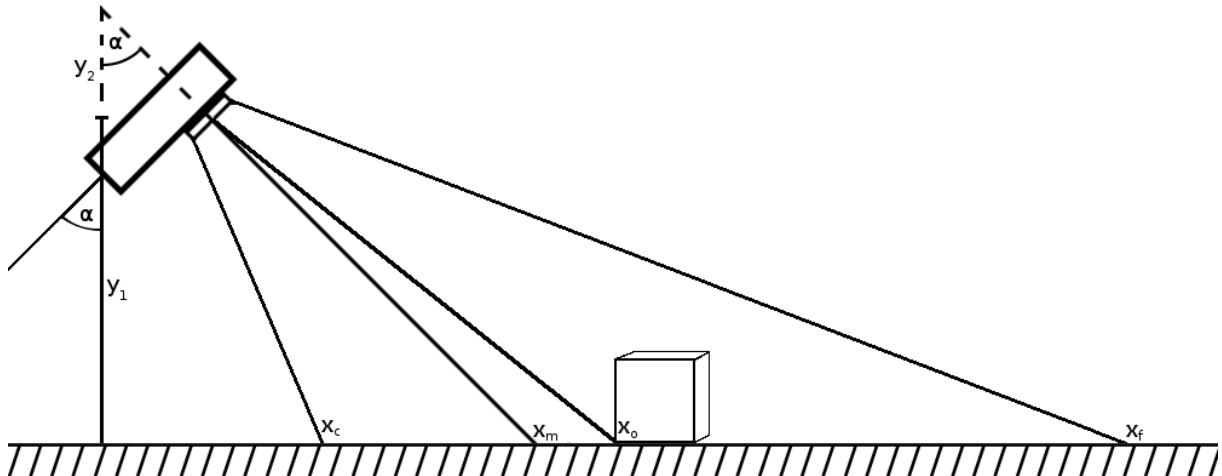


Abbildung 6.10: Exemplarische monokamerabasierte Entfernungsschätzung

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.

Ultraschallbasiert

Die ultraschallbasierte Entfernungsschätzung kann auch über große Distanzen von bis zu zwei Metern zentimetergenaue Entfernungsmessungen durchführen. 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 LEGOTM erworben

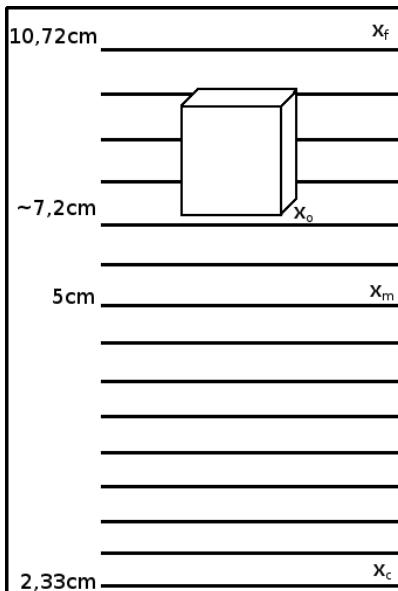


Abbildung 6.11: Aufnahme aus der monokamerabasierten Entfernungsschätzung

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.

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.

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 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 6.2 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

TODO

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° . 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 der X- und der Y-Anteil berechnet und auf die Roboterposition aufaddiert.

Bei Drehung um die eigene Achse wird

3.Skizze der Positionsverfolgung

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 beschrieben Verfahren kann sich der Roboter im Raum orientieren und den Startpunkt ausfindig machen und ansteuern. Abbildung 6.12 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 6.12: Screenshot der Applikation bei Rückkehr zur Ausgangsposition

7 Tests des Robotersystems

7.1 Tests im gesicherten Rahmen

7.2 Realtests

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.

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.