

Inhalt

1	Einleitung	1
2	Grundlagen	4
2.1	Robotik im Bereich teilautonomer technischer Assistenzsysteme	4
2.2	Mustererkennung als Teilgebiet des maschinellen Sehens	7
2.3	Visual Servoing	12
3	Konzeption	15
3.1	Manipulator mit Bildverarbeitungseinheit	15
3.1.1	Anforderungen an das System	15
3.1.2	Auswahl der Komponenten	18
3.1.3	Aufbau des Systems	21
3.2	Objekt- und Texterkennung	25
3.2.1	Maschinelles Detektionsverfahren mit vorgegebenem Merkmalsraum	25
3.2.2	Deep Learning Ansatz zur Detektion ohne Merkmalsraumvorgabe	31
3.2.3	Auslesen der Tasterbeschriftung	35
3.3	Objektverfolgung	38
3.3.1	Echtzeit-Objektverfolgung in Videosequenzen	39
3.3.2	Merkmalsabgleich und perspektivische Transformation zwischen Einzelbildern	40
3.4	Entfernungsberechnung	42
3.4.1	Berechnung durch Änderung der Bildgröße des Ziels	43
3.4.2	Berechnung durch Änderung des Winkels zum Ziel	43
3.5	Bewegungssteuerung	46
3.6	Integration der Teillösungen	49
4	Prototypische Umsetzung	53
4.1	Hard- und Software	53

4.2	Implementierung und Ergebnisse der Teilaufgaben	55
4.2.1	Tasterdetektion	55
4.2.2	Textdetektion und -erkennung	57
	Lokalisierung von Tasterbeschriftungen	57
	Auslesen von Tasterbeschriftungen	59
4.2.3	Objektverfolgung	61
4.2.4	Distanzermittlung	63
4.2.5	Firmware zur Bewegungssteuerung	63
4.3	Gesamtsystem	66
5	Fazit und Ausblick	70
6	Literaturverzeichnis	73
7	Abbildungsverzeichnis	79
8	Tabellenverzeichnis	81
Anhang A	Nachrichtenaustausch	82
Anhang B	Ergebnisbeispiele	83
Anhang C	Materialliste	85

1 Einleitung

Medizinische sowie technische Assistenzsysteme und Hilfsmittel erleichtern Menschen mit Behinderung ihren Alltag. Bei körperlicher Einschränkung findet oft ein elektrischer Rollstuhl Anwendung, welcher eine gewisse Autonomie in der Mobilität ermöglicht. Viele Tätigkeiten bzw. Aufgaben des täglichen Lebens, wie bspw. selbständiges Trinken oder das Betätigen von Fahrstuhl-tastern, bleiben aber weiterhin für viele Betroffene schwierig oder unmöglich. In diese Lücke stoßen seit einiger Zeit Assistenzsysteme aus dem Bereich der Robotik. Durch die Verwendung eines Roboterarms, den die eingeschränkte Person bspw. per Joystick steuert, können zahlreiche Tätigkeiten selbständig durchgeführt werden. Leider sind diese Systeme in der Regel teuer und durch die manuelle Steuerung sehr langsam. Darüber hinaus ist eine manuelle Bedienung nicht für jeden Menschen möglich. Aktuelle Forschungsprojekte haben zum Ziel, durch die Entwicklung von teilautonomen Fähigkeiten für Assistenzsysteme deren Anwendungsgebiet zu erweitern, die Nutzbarkeit zu verbessern sowie den potentiellen Anwenderkreis zu vergrößern.

Aufgrund der Verbreitung von additiven Fertigungsverfahren sowie des Open-Source-Gedankens ist es heutzutage möglich, relativ komplexe technische Systeme kostengünstig zu entwickeln. Z. B. existiert eine Reihe von offenen Entwürfen und Konstruktionsplänen zum Bau eines Roboterarms. So ist es durchaus denkbar, dass in Zukunft „intelligente“ Assistenzsysteme für einen größeren Kreis von Betroffenen bezahlbar werden.

Gegenstand der Arbeit ist die Konzeption eines teilautonomen technischen Assistenzsystems am Beispiel einer ausgewählten Funktion. Die zu bewältigende Aufgabe ist das automatische Betätigen eines Fahrstuhlknopfes, nachdem vom Nutzer ein Zielstockwerk ausgewählt wurde. Um die Einsatzumgebung nicht unnötig einzuschränken, sollen Fahrstühle mit Tastern unterschiedlichen Typs bedient werden können. Eine prototypische Implementierung eines Demonstrators dient zur Funktionsüberprüfung.

Inhalt der Arbeit ist ausschließlich die technische Betrachtung des Systems, rechtliche Rahmenbedingungen wie etwa das Medizinproduktegesetz oder Sicherheitsanforderungen durch die Mensch-Maschine-Interaktion werden nicht berücksichtigt. Der Schwerpunkt soll auf der Entwicklung der Funktionen für ein eingebettetes System liegen, welches theoretisch in ein anderes technisches System, z. B. einen elektrischen Rollstuhl mit Leichtbauroboterarm, integriert werden kann. Ebenfalls im Fokus steht die Verwendung freier Software sowie möglichst günstiger Hardwarekomponenten.

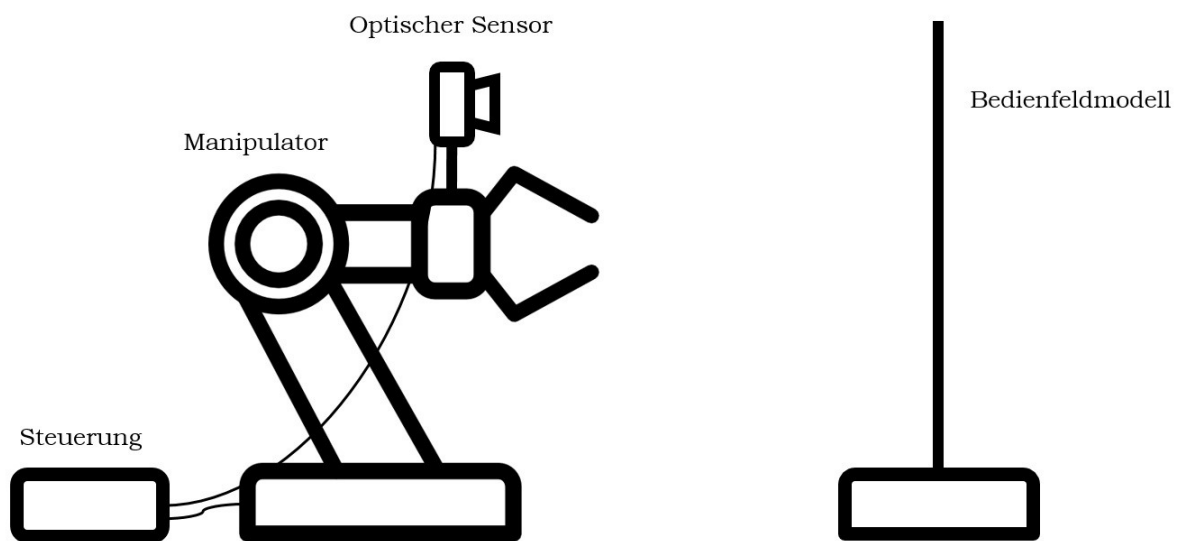


Abbildung 1: Möglicher Aufbau eines Prototyps

Die zu bearbeitende Aufgabe lässt sich untergliedern in die optische Erkennung von Fahrstuhltastern, das Auslesen der zugehörigen Beschriftungen, das Ermitteln der Lage des Ziels in Relation zum Assistenzsystem sowie das Ansteuern und Betätigen des gewünschten Tasters durch einen Aktor. Bei der Umsetzung der Funktionalität für ein vorhandenes Assistenzsystem ist unter Umständen ein Manipulator vorhanden und kann genutzt werden. Für den in dieser Arbeit zu entwickelnden Demonstrator kann allerdings nicht auf eine vorhandene Plattform zurückgegriffen werden, daher ist ein System zur Positionierung und Manipulation so auszuwählen oder zu entwickeln, dass die gestellte Aufgabe erfüllt werden kann. Für die Teilaufgaben sind Lösungswege basierend auf dem Stand der Technik zu entwerfen und abschließend zu einem Gesamtsystem zusammenzuführen. Abbildung 1 zeigt einen möglichen

vereinfachten Aufbau des zu entwickelnden Demonstrators mit Komponenten für Sensorik und Aktorik. Die Fahrstuhltaster werden in diesem Versuchsaufbau durch Modelle repräsentiert, dies können bspw. Fotos von echten Fahrstuhlbedienfeldern sein.

Im folgenden Kapitel werden die Grundlagen der für diese Arbeit relevanten Themenfelder betrachtet. Anschließend erfolgt in Kapitel 3 aufbauend auf der technologischen Ausgangssituation zunächst die Konzeption der Lösungen für die Teilaufgaben, bevor diese in ein Gesamtsystem zur Umsetzung der geplanten Funktion integriert werden. Details zur Implementierung sowie erzielte Ergebnisse werden in Kapitel 4 beschrieben. Abschließend erfolgen eine Beurteilung der geleisteten Arbeit und ein Ausblick auf sich durch offene Fragestellungen und Optimierungspotential ergebende zukünftige Forschungsprojekte.

2 Grundlagen

In diesem Kapitel erfolgt eine kurze Einführung in die theoretischen Grundlagen der vorliegenden Arbeit, zudem wird die Thematik auf relevante Bereiche eingegrenzt. Der aktuelle Stand der Technik dieser wissenschaftlichen Themengebiete dient als Ausgangspunkt für die Konzeption im folgenden Kapitel.

Da sich die Arbeit mit der Konzeption einer teilautonomen Funktion für ein technisches Assistenzsystem befasst, soll zunächst die technologische Ausgangssituation in diesem Bereich ermittelt werden. Anschließend erfolgt eine Einführung in die für die Umsetzung der Arbeit wichtigen Themengebiete Mustererkennung sowie Visual Servoing.

2.1 Robotik im Bereich teilautonomer technischer Assistenzsysteme

Technische Assistenzsysteme sind allgemein Systeme, die dem Menschen bei der Bewältigung einer Aufgabe helfen bzw. eine bestimmte Tätigkeit erst ermöglichen. Eine einheitliche Definition des Begriffes fehlt bisher, stattdessen erfolgen Abgrenzungen je nach Themengebiet und Ziel. So definiert das Bundesministerium für Gesundheit in einer Studie technische Assistenzsysteme als Hilfsmittel zur Unterstützung pflegebedürftiger Menschen auf der Basis von Informations- und Kommunikationstechnologien, wozu auch mechatronische Systeme gezählt werden [1]. In dieser Arbeit sollen Systeme aus der Robotik betrachtet werden, welche hilfsbedürftige Menschen teilautonom bei der Bewältigung von Alltagsaufgaben unterstützen.

Roboter lassen sich nach Einsatzgebiet bspw. in Industrie-, Service-, Medizin-, Erkundungs- oder Militärroboter unterteilen [2]. Serviceroboter werden zur Unterstützung des Menschen derzeit hauptsächlich im Haushalt eingesetzt, dringen aber zunehmend auch in den Bereich der Pflege vor. Nach DIN EN ISO 8373 wird auch bei gleicher Bauform und Funktionsweise zwischen Service- und Industrierobotern nach Einsatzgebiet unterschieden [3]. Im Folgenden sollen Roboter in Anlehnung an die VDI Richtlinie 2860 als frei program-

mierbare Bewegungsautomaten verstanden werden, die mit Hilfe eines Effektors Handhabungsaufgaben erfüllen können [4]. Bei teilautonomen Systemen sind in der Regel Sensoren zur selbsttätigen Programmadaption vorhanden.

Ein Beispiel für einen in der Rehabilitation verwendeten Roboter ist der JACO Roboterarm der Firma Kinova Robotics. Dieser ist für eine Montage am Rollstuhl vorgesehen, um Menschen mit eingeschränkter Mobilität mehr Selbstständigkeit im Alltag zu ermöglichen [5]. Gesteuert wird der JACO manuell per Joystick, Touchpad bzw. Kopf- oder Kinnsteuerung. Einfache Bewegungsabläufe können auch über eine USB-Schnittstelle programmiert werden. Es fehlt jedoch eine Sensorik für eine Adaption zur Laufzeit, daher sind teilautonome Funktionen mit dem JACO nicht zu realisieren.

Im aktuellen Forschungsprojekt SMART-Assist entwickelt das Institut für Robotik und Mechatronik des Deutschen Zentrums für Luft- und Raumfahrt (DLR) das Assistenzrobotiksystem EDAN (EMG-controlled daily assistant) [6]. Es besteht aus einem handelsüblichen elektrischen Rollstuhl, auf den ein DLR-Leichtbauroboterarm montiert ist. Schwerpunkte des Projekts sind eine auf Elektromyographie basierende Steuerung, sichere Mensch-Maschine-Interaktion sowie adaptives Greifen. Der Sicherheitsaspekt wird durch den Einsatz von flexiblen, adaptiven Materialien (Soft Robotics Konzept) sowie eine Drehmomentregelung umgesetzt. Integrierte Teilautonomie soll Unterstützung bei Aufgaben des Alltagslebens bieten¹. Veröffentlichungen zur Vorgehensweise oder Ergebnissen bei der Entwicklung von teilautonomen Funktionen existieren bislang nicht.

Ein ähnliches System ist der an der Universität Bremen entwickelte Assistenzroboter FRIEND (Functional robot arm with user-friendly interface for disabled people), welcher als Basis verschiedener Forschungsprojekte dient. Der Aufbau des Systems ist in Abbildung 2 zu sehen.

¹ Als Beispiel einer teilautonomen Funktion dient die Hilfe beim Trinken.

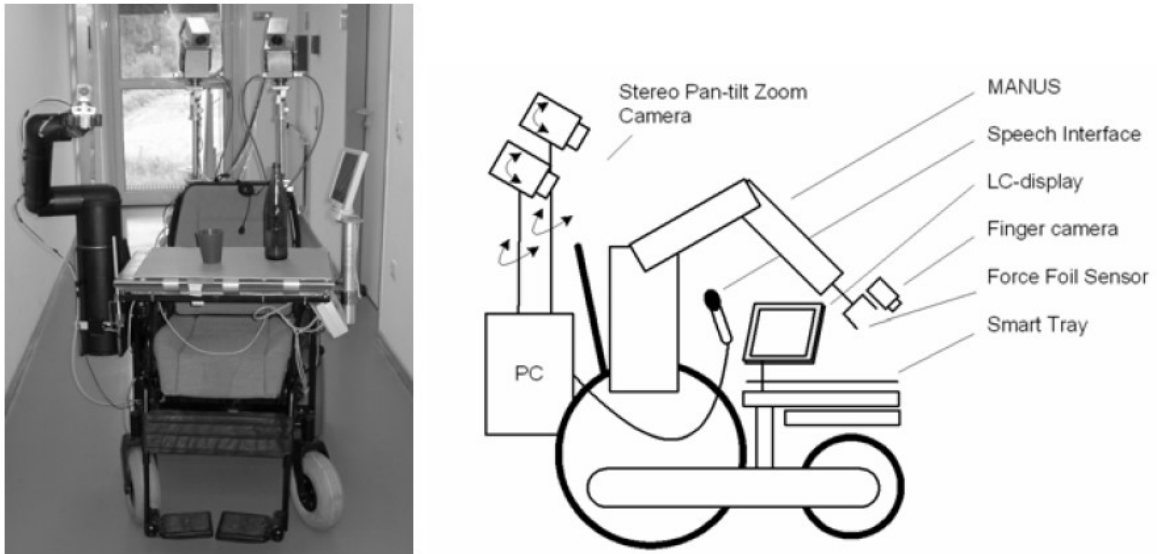


Abbildung 2: Assistenzsystem FRIEND der Universität Bremen [7]

Ein Schwerpunkt des ersten Forschungsprojekts ist das Ermöglichen eines teilautomatischen Greifvorgangs. Anstatt wie bspw. beim JACO jede Bewegung vorzugeben, wird hier der komplette Greifprozess mit zwei bis drei Nutzerkommandos initiiert und durchgeführt [7]. Dazu wird zunächst die Kamera am Effektor vom Nutzer in die Nähe des gewünschten Objekts manövriert. Anschließend erfolgt mit Hilfe von Visual Servoing (vgl. Abschnitt 2.3) die teilautomatische Ansteuerung des Ziels. Als Einschränkung müssen die zu greifenden Objekte auf einer zum System gehörenden Ablage platziert werden (siehe Smart Tray in Abbildung 2), welches über Sensoren Gewicht und Position des Objekts bestimmt. Im aktuellen Modellprojekt ReIntegraRob wird durch Verbesserung der Bildverarbeitung ein Arbeiten ohne Smart Tray ermöglicht. Ziel ist die Reintegration einer tetraplegisch gelähmten Frau in ihren Beruf als Bibliothekarin [8]. Hierzu soll der Roboter Funktionen wie das Aufnehmen und Ablegen des Buches oder das Umblättern von Seiten autonom durchführen.

Es existieren zahlreiche weitere Industrie- und Forschungsprojekte, welche sich mit Reha- bzw. Assistenzrobotern für Menschen mit Behinderung befassen². Bezüglich teilautonomer Funktionalität scheinen die hier vorgestellten

² Ein Beispiel für ein erwerbbares System ist der iARM der Firma Assistive Innovations [54], welcher ähnliche Funktionen wie der JACO bietet.

Systeme EDAN und FRIEND am aktuellsten bzw. weitesten fortgeschritten. Forschungsbedarf ist allerdings weiterhin vorhanden, da bisher Unterstützung nur für wenige spezielle Tätigkeiten angeboten wird. Weiterhin ist die Verfügbarkeit der genannten Systeme problematisch, da sie entweder für den normalen Nutzer kaum erschwinglich³ sind oder ausschließlich in Forschungsprojekten zur Verfügung stehen.

2.2 Mustererkennung als Teilgebiet des maschinellen Sehens

„Computer Vision, auch maschinelles Sehen genannt, umfasst verschiedene Methoden zur Erfassung, Verarbeitung, Analyse und Interpretation von Bildern.“ Mustererkennung in der Bildverarbeitung ist der Prozess zur Identifikation von Objekten. Teilschritte sind Bildaufnahme, Vorverarbeitung, Segmentierung, Merkmalsextraktion, Klassifizierung und Aussage bzw. Interpretation [9]. Heutige Standardmethoden zur Mustererkennung verfolgen in der Regel einen statistischen Ansatz, bei dem Wahrscheinlichkeiten für die Zugehörigkeit eines Objektes zu verschiedenen Kategorien ermittelt werden.

Die Bildvorverarbeitung hat zur Aufgabe, unerwünschte Signalbestandteile zu reduzieren (z. B. Rauschreduktion). Mit Hilfe von Segmentierung werden zusammenhängende bzw. zusammengehörende Regionen in Bildern identifiziert. Anhand von Merkmalen⁴ wie bspw. Kanten oder Eckpunkten wird in der Bildanalyse versucht, auf Bildinhalte zu schließen. Das Auffinden verschiedener Merkmale übernimmt die Merkmalsextraktion. Diese liefert für ein Bild einen Merkmalsvektor im Merkmalsraum. Die Klassifikation soll abschließend anhand des Merkmalsvektors die segmentierten Regionen oder Objekte in verschiedene Klassen einteilen [11].

In der visuellen Objekterkennung muss zwischen der Bildklassifikation und der Objektlokalisierung unterschieden werden. Die Bildklassifikation prüft, ob bzw. mit welcher Wahrscheinlichkeit in einem Bild mindestens ein Beispiel einer oder mehrerer Klassen existiert. Das ganze Bild wird der Klasse bzw.

³ Die Preise von JACO und iARM liegen im Bereich von mehreren zehntausend Euro.

⁴ Ein Merkmal ist eine numerische oder qualitative Information, die aus Bildpunkten einer Region (lokal) oder des gesamten Bildes (global) berechnet wird. Merkmale können bspw. farb-, form- oder texturbasiert sein, aber auch durch abstrakte Operationen entstehen [10].

Kategorie mit der höchsten Wahrscheinlichkeit zugeordnet. Ziel der Objektklassifizierung (auch Objektdetektion genannt) ist das Auffinden von Objekten verschiedener Klassen inklusive der Bestimmung von Position und Ausdehnung. Für beide Aufgabenstellungen werden heutzutage häufig Verfahren des maschinellen Lernens (Machine Learning) eingesetzt, bei denen ein vorgegebener Algorithmus zur Extraktion vorbestimmter Merkmale verwendet wird. Aufgrund der Leistungsfähigkeit moderner Hardware finden zunehmend auch Deep Learning Methoden Einsatz in der Objekterkennung. Durch tiefe neuronale Netze sollen die für die Klassifizierung und Detektion relevanten Merkmale selbständig erlernt werden. Deep Learning Verfahren zur Objekterkennung erzielen häufig hohe Erkennungsraten, benötigen hierzu allerdings eine große Menge an Trainingsdaten, lange Trainingsphasen und hohe Detektionszeiten. Verkürzt werden kann das Training durch das sogenannte Transfer Learning, bei welchem nur die letzten Schichten eines auf verschiedene Objektklassen spezialisierten neuronalen Netzes neu trainiert werden. Je nach Größe und der Anzahl verschiedener Klassen des verwendeten vortrainierten Netzwerks können die Ergebnisse in Bezug auf Erkennungsrate und Ausführungszeit unterschiedlich ausfallen [11].

Ein bekanntes Beispiel für ein maschinelles Lernverfahren zur Objektdetektion ist die Methode nach Viola und Jones [12]. Als mögliche Merkmale für einen gegebenen Bildausschnitt werden hier die Differenzen von Grauwertsummen innerhalb verschiedener Rechtecktypen verwendet, sogenannte Haar-ähnliche Merkmale (vgl. Abbildung 3). Da die Rechtecktypen in verschiedener Größe und Skalierung verwendet werden, ergibt sich eine hohe Anzahl an möglichen Merkmalen für die Klassifikation. Um die Rechenzeit zu reduzieren, wird eine Auswahl relevanter Merkmale mit Hilfe eines sogenannten Boosting-Verfahrens anhand von positiven und negativen Trainingsdaten erlernt. Ein weiterer wichtiger Schritt zur Effizienzsteigerung ist die Verwendung einer Kaskade von Klassifikatoren, welche negative Bildausschnitte schnell verwirft [12].

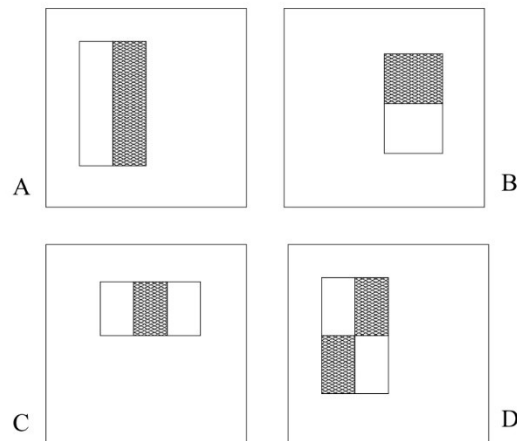


Abbildung 3: Vier Grundtypen zur Merkmalsberechnung nach Viola und Jones mit zwei (A und B), drei (C) und vier (D) Rechtecken in verschiedener Skalierung und Position [12]

Auf der Viola-Jones-Methode basieren zahlreiche weitere Ansätze zur Objektdetektion, bspw. erreicht das Verfahren von Ahonen et al. durch die Verwendung von Local Binary Patterns (LBP) als texturbasierte Merkmale eine schnellere Trainings- und Detektionszeit [13]. Ebenfalls beliebt ist die Verwendung der Orientierung der Kanten im Bild, welche für jeden Teilbereich in einem Histogramm gespeichert und so für die Merkmalsbeschreibung genutzt werden (Histogram Of Oriented Gradients). Dieses Verfahren führt teilweise zu besseren Erkennungsraten, hat aber ein schlechteres Laufzeitverhalten als die Verwendung von Haar-ähnlichen oder LBP-Merkmalen [14].

In einer für die vorliegende Themenstellung interessanten Arbeit werden zunächst verschiedene Merkmale in einem ähnlichen Verfahren wie der Viola-Jones-Methode zur Detektion von unbekannten Fahrstuhltastern verwendet. In anschließenden Nachbearbeitungsschritten werden mögliche Falschdetektionen und nicht erkannte Taster weitgehend korrigiert [15]. Diese Schritte basieren auf der Einschränkung auf solche Taster, die den ADA Richtlinien [16] entsprechen⁵. Unter anderem muss sich dafür die Beschriftung immer - zumindest zusätzlich - links außerhalb des Tasters befinden.

⁵ In Deutschland entspricht nur ein Teil der Fahrstuhltaster diesen Richtlinien.

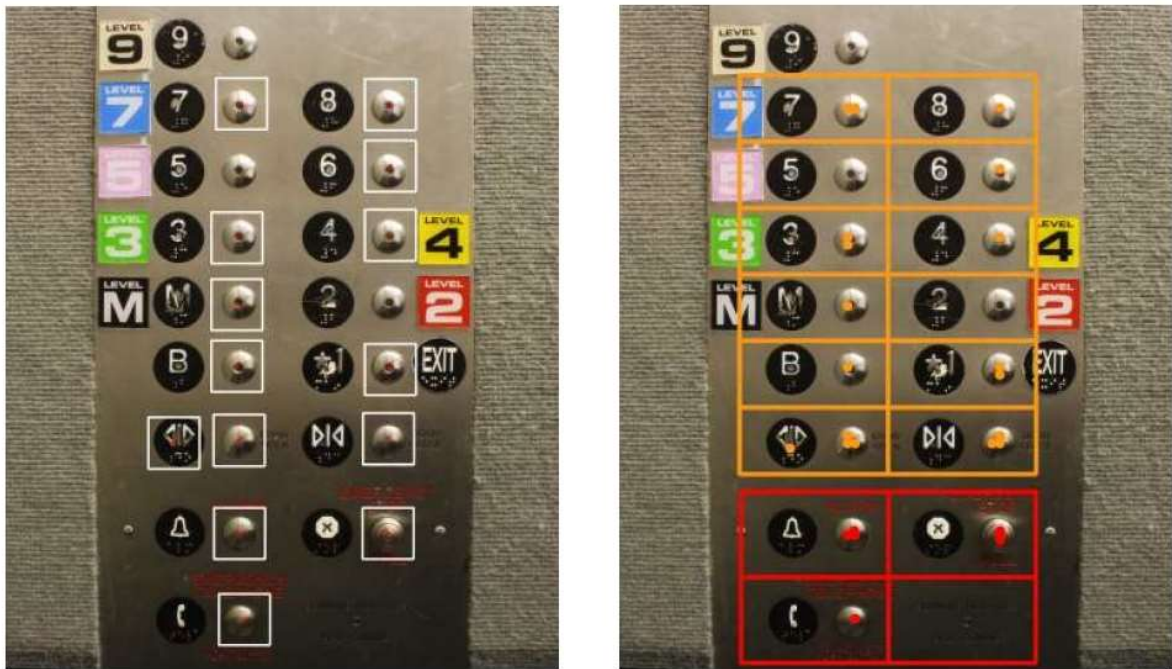


Abbildung 4: Erkannte Taster (links) und ermitteltes Gitter (rechts) [15]

Auf der Bedientafel eines Fahrstuhls ergibt sich somit eine Art Gittermuster, welches für die Erkennung genutzt wird (vgl. Abbildung 4 rechts). Weiterhin wird im Systemaufbau der Arbeit ein Laserscanner verwendet, mit welchem die zu erwartende Tastergröße (bekannt aus den ADA Richtlinien) geschätzt werden kann. Bei einem Testdatensatz mit 150 Bildern von Bedientafeln wird eine Erkennungsrate von 86,2 Prozent erreicht, Angaben zur Ausführungs-dauer fehlen allerdings [15].

Seit dem ersten Sieg von Krizhesvsky et al. beim ImageNet Wettbewerb sind Convolutional Neural Networks (CNNs) bei der Bildklassifikation ungeschlagen [17]. Bei diesen handelt es sich um „faltende“ neuronale Netze, bei denen in jeder Faltungsschicht die Werte eines Filterkernels bzw. einer Faltungsmatrix selbständig durch Fehlerrückführung erlernt werden. Nachteil tiefer neuronaler Netze ist das Verhalten als Black Box. Es ist kaum ersichtlich, nach welchen Kriterien Entscheidungen getroffen werden. Eine detaillierte Beschreibung der Funktionsweise kann bspw. in [18] gefunden werden. Auch der komplexere Bereich der Objektdetektion wird mittlerweile von verbesserten CNN-Architekturen dominiert. Ein erster Durchbruch gelang Girshick et al. mit sogenannten Region-based Convolutional Neural Networks (R-CNNs)

[19]. Während vorherige Ansätze meist nach dem Sliding-Windows Prinzip⁶ vorgehen, erfolgt bei R-CNNs die Anwendung der neuronalen Netze zur Klassifikation nur auf einer wesentlich geringeren Anzahl von vorgefilterten Regionen. Dadurch kann die Trainings- und Detektionszeit erheblich verkürzt werden⁷.

Nach dem ersten Erfolg des R-CNN-Ansatzes zur Objektllokalisierung versuchen Wissenschaftler die Laufzeiten oder die Performanz im Sinne von Erkennungs- und Fehlerraten zu verbessern. Die bislang erfolgreichsten Ansätze sind Faster R-CNNs, You Only Look Once (YOLO) und Single Shot Detector (SSD) mit MobileNet-Architektur⁸. Während YOLO bislang die schnellsten Ausführungszeiten vorweisen kann, erzielt SSD bei ebenfalls geringen Laufzeiten höhere Erkennungsraten [21].

Zur Detektionszeit von Objekterkennungsverfahren in eingebetteten Systemen bzw. auf Low-Power-Hardware existieren nur wenige Veröffentlichungen. Eine aktuelle Arbeit vergleicht Laufzeiten und Erkennungsraten einiger der oben genannten sowie eines selbst entwickelten Deep Learning Verfahrens bei Ausführung auf einem Raspberry Pi 3 Einplatinencomputer. Bei allen getesteten Verfahren liegt die erreichbare Bildrate bei unter einem Bild pro Sekunde⁹. Höhere Erkennungsraten korrelieren mit schlechteren Laufzeiten [22]. Für echtzeitfähige Systeme zur Objektdetektion ist je nach Einsatzgebiet und Anforderungen noch weitere Forschungs- und Entwicklungsarbeit notwendig.

Texterkennung in Bildern gliedert sich in die Detektion von Bereichen, die Text oder einzelne Zeichen enthalten, sowie die optische Zeichenerkennung,

⁶ Beim Sliding-Window-Prinzip wird ein Fenster vordefinierter Größe Pixel für Pixel über das gesamte Bild verschoben und an jeder Position eine Klassifikation vorgenommen [11].

⁷ Resultat ist eine Reduktion der Detektionszeit auf ca. 50 Sekunden auf einer High-End-CPU des Jahres 2013 [19].

⁸ Insbesondere hat auch die Architektur der verwendeten tiefen neuronalen Netze, wie z. B. Anzahl und Typ der Schichten, einen Einfluss auf Erkennungsrate und Ausführungszeiten. MobileNets eignen sich für mobile Geräte und erzielen trotz kompakter Größe dennoch hohe Erkennungsraten [20].

⁹ Die Detektionsrate liegt im Bereich von 0,1 fps bis 0,7 fps (frames per second)

welche aus diesen Bereichen Buchstaben, Zahlen oder Wörter extrahiert. Zum Auffinden von Regionen, welche möglicherweise Zeichen enthalten, kann bspw. nach verbundenen Konturen oder einheitlichen Texturen gesucht werden. Durch eine Klassifikation anhand vorgegebener Merkmale werden diese anschließend in Textbereiche oder Nicht-Textbereiche unterteilt. Die darauf folgende Segmentierung trennt Zeichen vom Bildhintergrund¹⁰ und liefert zusammengehörige Blöcke von Zeichen. Erst jetzt erfolgt die eigentliche Erkennung des Textes, welche zeichenbasiert (Optical Character Recognition) oder wortbasiert erfolgen kann. Auch hier werden zunächst Merkmale extrahiert bevor die Zeichen in Kategorien unterteilt werden [23]. Auch in den Bereich der Texterkennung dringen vermehrt Verfahren des Deep Learnings ein. So genannte Ende-zu-Ende-Verfahren übernehmen alle Schritte der Textdetektion und -erkennung und liefern zunehmend gute Ergebnisse [24].

2.3 Visual Servoing

Aufgabe von Visual Servoing ist es, mit Hilfe von Bildverarbeitungsdaten die Bewegungen oder die gewünschte Position des Effektors eines Roboters zu steuern. Diese Daten können von einer am Effektor befestigten Kamera stammen (eye-in-hand), wobei Bewegungen des Roboters auch Bewegungen der Kamera auslösen. Eine weitere Möglichkeit ist die Platzierung der Kamera außerhalb des Roboters, sodass dessen Bewegungen von einem festen Punkt aus beobachtet werden (eye-to-hand). Unterschieden wird weiterhin zwischen bildbasierter Regelung (Image Based Visual Servoing), bei welcher Bildinformationen zur direkten Steuerung der Gelenke des Roboters verwendet werden, und positionsbasierter Regelung (Position Based Visual Servoing), bei der zunächst eine geometrische Interpretation der Bilddaten erfolgt und daraus ein Steuerbefehl im kartesischen Raum erzeugt wird [25]. Bei der bildbasierten Regelung wird die Regelabweichung aus der Lageabweichung eines vorher aufgenommenen Referenzbilds und der aktuellen Kameraaufnahme ermittelt. Für die positionsbasierte Regelung wird die Stellung der kinematischen Kette

¹⁰ Dies kann unter anderem durch Binarisierung erfolgen.

im Raum anhand von Bilddaten geschätzt und ein Positionsfehler als Regelabweichung verwendet [26].

Ein stabiler geschlossener Regelkreis zur Steuerung der Gelenke benötigt eine hohe Regelfrequenz. Daten aus Bildverarbeitungssystemen stehen allerdings häufig mit geringerer Frequenz zur Verfügung, sodass die meisten Visual Servoing Systeme eine positionsbasierte Regelung anwenden [26].

Sowohl zur Positionsschätzung (positionsbasierte Regelung) als auch zur Bestimmung der Lageabweichung zum Referenzbild (bildbasierte Regelung) werden Merkmale (vgl. Abschnitt 2.2) aus der Bildverarbeitung verwendet. Anhand ausgewählter Merkmalspunkte lässt sich die Transformation zwischen zwei Bildern berechnen. Werden diese Punkte markerbasiert¹¹ ermittelt, führt dies zu einer schnellen und robusten Merkmalsextraktion. Dagegen ist die Verwendung von Merkmalen, die aus dem Zielobjekt generiert werden, zwar aufwendiger aber auch wesentlich flexibler einsetzbar. Zur Bestimmung der Merkmalspunkte liefert zunächst ein Merkmalsdetektor markante Positionen im Bild, anschließend versieht sie ein Merkmalsdeskriptor mit zusätzlichen Informationen, wie bspw. der Größe und Orientierung der Gradienten in der Umgebung des Merkmalspunkts. Brauchbare Merkmale sollten unter anderem lokal, möglichst invariant gegenüber Skalierung, Translation und Rotation, wenig anfällig für Rauschen und Verwischen, gut unterscheidbar und effizient zu berechnen sein [27].

Ein bekanntes Verfahren zur Detektion und Beschreibung lokaler Merkmale ist SIFT (Scale-Invariant Feature Transform), welches die oben genannten Kriterien weitgehend erfüllt. Eine effiziente Abwandlung von SIFT ist der SURF-Algorithmus (Speeded Up Robust Features), welcher ähnlich robuste und invariante Merkmale in kürzerer Berechnungszeit liefert [28].

Ein Vergleich zwischen verschiedenen Bildern desselben Objekts liefert zusammengehörige Merkmalspunkte (auch Feature Matching genannt). Sind die Bilder aus unterschiedlichen Perspektiven entstanden, kann anhand dieser

¹¹ Dies kann z. B. mit QR-Codes erfolgen.

Merkmalspunkte die Homographie¹² und damit die Bewegung der Kamera oder des Objektes berechnet werden [27].

¹² Zu geometrischen Transformationen und homogenen Koordinaten siehe bspw. [11].

3 Konzeption

In diesem Abschnitt wird auf der Grundlage der in Kapitel 2 beschriebenen technologischen Ausgangssituation ein Entwurf für die Funktion der teilautomatischen Tastererkennung und -betätigung erarbeitet. Da die Funktionalität an einem prototypischen Demonstrator gezeigt werden soll, sind zunächst Grundlagenentscheidungen bezüglich dessen Aufbau und verwendeter Komponenten zu treffen. Aus diesem Auswahlprozess können sich Restriktionen für die möglichen Lösungswege der Teilaufgaben ergeben.

3.1 Manipulator mit Bildverarbeitungseinheit

Aufgabe des Manipulators ist die physikalische Interaktion mit der Umgebung, hier also die Betätigung des gewünschten Fahrstuhltasters. Dazu muss er Steuerbefehle entgegennehmen und eine gewünschte Position im Raum anfahren. Die Steuerbefehle stammen aus einer Bildverarbeitungseinheit, welche die anzusteuern Position aus Kameradaten berechnet.

Da zahlreiche Möglichkeiten des Systemaufbaus bestehen, wie z. B. Anzahl und Art der Achsen, werden aus den funktionalen und nicht-funktionalen Anforderungen zunächst Kriterien für die Festlegung des Aufbaus und die Auswahl der verwendeten Komponenten abgeleitet.

3.1.1 Anforderungen an das System

Die Anforderungen an das zu entwickelnde System ergeben sich aus den zu lösenden Teilaufgaben und den Bedingungen der Einsatzumgebung. Im Zielszenario soll die teilautonome Assistenzfunktion durch ein – z. B. in einem elektrischen Rollstuhl – eingebettetes System geleistet werden. Daher ist auch für den Demonstrator möglichst kompakte, energiesparende und günstige Hardware zu verwenden, welche dennoch leistungsstark genug für eine echtzeitnahe Ausführung der Programmlogik ist. Da bei dem zu entwickelnden Prototypen geringe Kosten eine hohe Priorität haben, wird die Antwortzeit in Form einer weichen Echtzeitanforderung auf zehn Sekunden festgelegt. D. h. das System muss in unter zehn Sekunden reagieren, nicht aber die vollständige Funktion ausführen.

Als Eingabe für die Bildverarbeitung in den Arbeitsschritten Objekt- und Texterkennung, Entfernungsberechnung sowie Objektverfolgung dienen Bilder im digitalen Format. Diese müssen in ausreichend hoher Auflösung, Bildqualität und Frequenz bereitgestellt werden. Die in Abschnitt 2.2 und 2.3 genannten Verfahren zur Mustererkennung nutzen Auflösungen, die alle modernen Digitalkameras und Kameramodule für Computer bieten. Da wie oben beschrieben Low-Power-Hardware für die Ausführung der Bildverarbeitung verwendet werden soll, wird die benötigte Eingangsbildrate durch die Rechenzeit der Objekt- und Texterkennung auf ein Bild pro Sekunde beschränkt (vgl. Kapitel 2.2).

Neben der Bildverarbeitung laufen die Programmlogik bzw. die Gesamtablaufsteuerung und die Verarbeitung von Benutzereingaben auf derselben Hardware (im Folgenden Hauptrechner genannt). Für letztere muss eine entsprechende Schnittstelle angeboten werden. Optional kann auch die Steuerung des Manipulators vom Hauptrechner übernommen werden. Sollen aber bspw. Bildverarbeitung und optische Regelung parallel laufen, reicht die Leistungsfähigkeit unter Umständen nicht aus. Wird für die Aktorsteuerung separate Hardware verwendet, muss eine Schnittstelle zur Kommunikation mit dem Hauptrechner berücksichtigt werden.

Aufgabe des Manipulators ist die Positionierung des Effektors innerhalb des Arbeitsbereichs im dreidimensionalen Raum, welche durch einen externen Steuerbefehl angestoßen wird. Da für die Entwicklung des Demonstrators die Grundfunktionalität wesentlich ist, spielen Reichweite und Arbeitsbereich nur eine untergeordnete Rolle. Zur Umsetzung des Positionierbefehls nimmt die Steuerung des Manipulators Koordinaten¹³ entgegen und berechnet daraus Steuerbefehle für die einzelnen Motoren der Achsen. Diese müssen ein ausreichendes Drehmoment für die Bewegung der Glieder sowie für die Tasterbetätigung besitzen, wobei letzteres kaum ins Gewicht fällt.

¹³ Die Form der Koordinaten (bspw. kartesisch) und das Bezugssystem sind noch zu definieren.

Da der Effektor keine Werkstücke transportiert und lediglich Druck auf einen Taster ausüben soll, sind weder Werkzeug noch Greifer erforderlich. Zur Begrenzung der ausgeübten Kraft ist zu überprüfen, ob bzw. wie viel Druck der Effektor auf einen Gegenstand ausübt. Es wird angenommen, dass nur solche Taster betätigt werden, die in einem kegelförmigen Bereich vor dem Manipulator liegen. Dieser Bereich wird durch das Sichtfeld der verwendeten Kamera bestimmt. Daher muss die Orientierung des Effektors im Raum nicht beliebig sein. Lediglich der Winkel des Vektors der Krafteinwirkung zur Normalen des Tasters sollte bei Betätigung nicht zu groß werden¹⁴.

Weiterhin wird angenommen, dass sich keine Hindernisse im Arbeitsraum bzw. zwischen der Ausgangsposition des Effektors und dem anzusteuernenden Taster befinden. Dadurch kann eine zeit- und ressourcenintensive Bahnplanung vermieden werden.

Aus den genannten Anforderungen kann abgeleitet werden, dass der Manipulator mindestens drei Achsen bzw. Freiheitsgrade benötigt, um die Positionier- bzw. Betätigungsaufgabe im Arbeitsraum zu erfüllen. Mehr Achsen können z. B. zu einer besseren Orientierung des Effektors im Raum oder zu einem größeren Arbeitsbereich beitragen, sind aber nicht zwingend erforderlich. Da in der Zieleinsatzumgebung wenig Platz zur Verfügung steht (bspw. ein an einem Rollstuhl befestigter Leichtbauroboterarm), ist maximal eine Translationsachse zu verwenden.

Die Energieversorgung des Systems ist für den Demonstrator nicht zwingend mobil einzuplanen. Nach Möglichkeit ist aber durch Verwendung energiesparender Komponenten und einer entsprechend dimensionierten Energiequelle die Voraussetzung für eine spätere mobile Einsetzbarkeit zu schaffen.

Da es sich bei dem zu entwickelnden System um einen Funktionsprototypen handelt, stehen Kriterien wie Zuverlässigkeit, Aussehen, Wartbarkeit, Sicherheitsanforderungen oder die Betriebs- und Umweltbedingungen nicht im Vordergrund. Daher werden diesbezüglich keine Anforderungen gestellt. Wichtig

¹⁴ Da an dieser Stelle keine genauen Aussagen getroffen werden können, wird der Grenzwert willkürlich auf 45° festgelegt.

hingegen sind eine einfache Bedienbarkeit des Systems, die Portierbarkeit der Funktionalität auf andere Systeme, aufgrund des Zeitrahmens der Arbeit eine einfache Implementierbarkeit und geringe Kosten. Durch die Verwendung von offener Hard- und Software sowie von weit verbreiteten Komponenten können die Portierbarkeit erhöht und die Kosten gesenkt werden. Durch eine übersichtlich und einfach gehaltene Benutzerschnittstelle ist die verständliche Bedienung zu ermöglichen.

3.1.2 Auswahl der Komponenten

Mit Blick auf die im vorherigen Kapitel beschriebenen Anforderungen und Kriterien erfolgt in diesem Abschnitt die Auswahl der für den Demonstrator zu verwendenden Komponenten. Da diese untereinander kompatibel sein müssen, wird zunächst die Hardware des Hauptrechners bestimmt und davon ausgehend die übrigen Bestandteile des Systems. Eine Materialliste der verwendeten Komponenten ist in Anhang C zu finden.

Hauptrechner

In [29] ist eine aktuelle Übersicht über gängige Einplatinencomputer mit ihren jeweiligen Vor- und Nachteilen wie bspw. verfügbare Anschlüsse, Leistungsfähigkeit und Preis zu finden. Nur aufgrund von Leistung und Preis würde die Wahl nicht auf den weit verbreiteten Raspberry Pi 3 fallen, allerdings sind noch andere Kriterien zu beachten. Um den Entwicklungsaufwand möglichst zu reduzieren ist eine gute Verfügbarkeit von Bibliotheken für Bildverarbeitungsaufgaben und Kommunikation über die vorhandenen Schnittstellen erforderlich. Die Größe der Anwendergemeinde bestimmt, wie schnell und ob überhaupt Hilfestellungen erhalten werden können. Weiterhin ist eine Plattform, die seit Jahren weiterentwickelt wird, im Allgemeinen zuverlässiger als eine Neuentwicklung. Diese Punkte sprechen in Summe für den Raspberry Pi 3 als Hauptrechner für das vorliegende Projekt.

Sensoren

Für die Bildverarbeitung wird eine Kamera benötigt, an die wie oben erwähnt keine besonderen Ansprüche bestehen. Die erste Version des Raspberry Pi Kameramoduls ist günstig, klein, leicht, bietet eine Auflösung von maximal 2592 mal 1944 Pixeln und einen Videomodus mit 30 Bildern pro Sekunde bei

FullHD-Auflösung [30]. Vor allem aufgrund der Kompaktheit ist dieses Modul für den zu entwickelnden Prototypen interessant. Es existiert eine neuere Version des Moduls mit besseren Leistungsdaten, welche allerdings nicht benötigt werden. Zudem ist der Preis höher, sodass hier die erste Version des Moduls ausgewählt wird.

Für die Abstandsberechnung des Effektors zum Ziel kann ein Abstandsensor, wie z. B. ein Time-of-Flight-Sensor, zum Einsatz kommen. Mit dieser dedizierten Hardware ist eine genaue und schnelle Bestimmung des Abstands zum Ziel möglich. Da aber bei dem zu entwickelnden Prototyp zunächst die Kosten im Vordergrund stehen und schon eine Kamera vorhanden ist, soll die Entfernungsberechnung mit dieser durchgeführt werden.

Zur Vermeidung von Schäden ist am Effektor ein Sensor zu verwenden, welcher die Überschreitung eines Grenzwerts der Krafteinwirkung erkennt. Für den Prototyp soll der Prozess der Tastererkennung und -betätigung zunächst anhand von Fotos von Fahrstuhlknöpfen simuliert werden. Daher ist ein einfacher mechanischer Endlagenschalter mit einer kleinen Feder ausreichend.

Manipulator

In der Zieleinsatzumgebung kommt im Allgemeinen ein Roboterarm zum Einsatz, daher soll nach Möglichkeit auch für den Prototyp ein solcher Aktor verwendet werden. Aus Zeitgründen ist eine Neukonstruktion in der vorliegenden Arbeit nicht machbar. Zur Auswahl stehen die Anschaffung eines fertigen Manipulators oder eines Bausatzes sowie der Aufbau nach frei zugänglichen Bauplänen. Sowohl für fertige Roboter als auch für Bausätze gilt, dass sie entweder zu teuer sind oder ein geschlossenes System bilden, dass nur manuell steuerbar ist. Daher erfolgt ein Nachbau einer kinematischen Struktur, welcher zudem die Flexibilität bei der Wahl der Motoren, der Steuerungshardware und damit auch der Umsetzung der Steuerungssoftware erhöht.

Walter [31] ist ein Open Source Projekt für einen Gelenkarmroboter mit sechs Rotationsachsen. Baupläne für die Steuerungshardware werden ebenso zur Verfügung gestellt wie die Steuerungssoftware, welche bspw. eine Bahnpla-

nung mit Interpolation mittels Bézier-Kurven bietet. Der Preis der Komponenten und der Aufwand zum Aufbau sprengen allerdings den Rahmen dieser Arbeit¹⁵.

Die Auswahl an günstig herzustellenden Robotern mit frei verfügbaren Bauplänen ist beschränkt, insbesondere wenn sie von mehr als nur einem Anwender erprobt bzw. getestet sind. Umsetzbare Vorlagen beinhalten kleinere kinematische Strukturen, die Steuerung geht häufig nicht über die manuelle Bewegung der einzelnen Achsen hinaus. Eine Berechnung der inversen Kinematik zur Positionierung des Effektors erfolgt bspw. in einem Projekt mit einem 3-Achs-Roboter mit zwei gleich langen Armen. Umfangreichere Berechnungen sowie eine Bahnplanung bietet von den hier betrachteten Projekten nur der oben genannte Roboter Walter, diese sind aber für den Demonstrator nicht zwingend erforderlich.

Geeignet für die Anwendung in der vorliegenden Arbeit scheint besonders das Modell eines ABB IRB460 im Maßstab 1:7, welches eine umfassende Anleitung bietet und von zahlreichen Nutzern erfolgreich nachgebaut wurde [32]¹⁶. Zudem ist der Roboterarm durch die Verwendung 3D-gedruckter Teile preiswert und einfach aufzubauen. Im Gegensatz zum Original wird hier auf die letzte Achse verzichtet. Drei Achsen bzw. Freiheitsgrade beschränken zwar den Arbeitsbereich und die Möglichkeiten zur Orientierung des Effektors, dafür ist die inverse Kinematik über den Kosinussatz bestimmbar. Somit muss kein iteratives Verfahren oder Ähnliches implementiert werden. Zudem stellen weniger Gelenke auch weniger Fehlerquellen in der Positionierung dar. Der Effektor wird durch die Verwendung einer verknüpften Kinematik immer in der Horizontalen gehalten.

¹⁵ Der Preis für Walters Komponenten beträgt ca. 800 Euro.

¹⁶ Von dem hier beschriebenen Modell existieren zwei Varianten, von denen eine Schrittmotoren anstelle von Servomotoren verwendet. Da das verwendete Schrittmotormodell ein höheres Drehmoment und vermutlich eine höhere Positioniergenauigkeit gegenüber dem eingesetzten Servomotor bietet, wird hier die erste Variante gewählt.

Bei allen erschwinglichen Systemen handelt es sich um Roboter mit Modellbaucharakter und nicht um professionelle Industrieroboter. Daher muss mit geringer Präzision, Wiederholgenauigkeit und Robustheit gerechnet werden.

Steuerung des Manipulators

Durch den Einsatz dedizierter Hardware für die Steuerung des Manipulators können Ressourcenprobleme vermieden werden. Die Trennung der Software für die Robotersteuerung von der restlichen Programmlogik erhöht die Flexibilität; so ist eine andere Programmiersprache einsetzbar, der Code übersichtlicher und die Fehlersuche einfacher.

Die Verwendung eines verbreiteten Mikrocontrollers wie dem Arduino Uno bietet eine große Auswahl an Softwarebibliotheken und Programmierbeispielen. Das hierfür erhältliche Erweiterungsmodul CNC-Shield in Verbindung mit Schrittmotortreibern des Typs A4988 vereinfacht die Steuerung der Motoren. Grundsätzlich sind auch andere Mikrocontroller oder Treiber einsetzbar, der günstige Preis und die Eignung für die Steuerungsaufgabe führen hier zu einer willkürlichen Entscheidung für die bewährte Kombination aus Mikrocontroller, Erweiterungsmodul und Motortreibern. Die Kommunikation mit dem Hauptrechner kann über die serielle Schnittstelle unter Verwendung eines USB-Kabels erfolgen. Der zusätzliche Energiebedarf ist gering und kann für den Prototypen vernachlässigt werden.

3.1.3 Aufbau des Systems

Der Manipulator besteht aus einer drehbaren Basis sowie zwei Armen, welche ebenfalls durch Rotationsachsen verstellbar sind. Die Halterung für den Effektor wird durch Gelenkstangen stets in der Horizontalen bzw. parallel zur xy-Ebene gehalten. Abbildung 5 zeigt den kinematischen Aufbau des verwendeten Modells nach VDI 2861, dabei handelt es sich um eine Kombination aus serieller und paralleler Kinematik. Eine Seitenansicht des Originals, bei welcher die Gelenkstangen zu erkennen sind, ist in Abbildung 6 zu sehen. Der Effektor besteht aus einer Halterung sowie dem daran fest installierten Endlagenschalter.

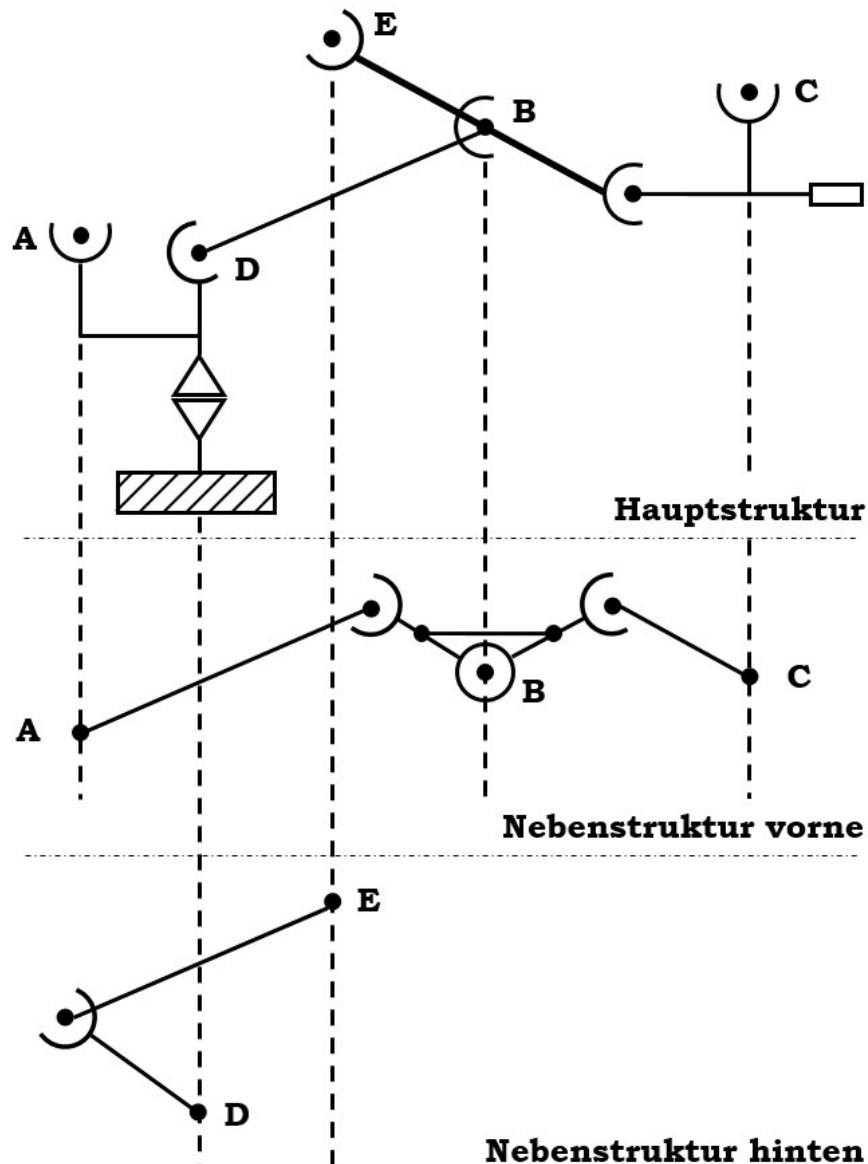


Abbildung 5: Kinematischer Aufbau des Manipulators nach VDI 2861 [33]

Da kein Abstandssensor verwendet wird, kann die Entfernung mit Hilfe der Kamera durch die Veränderung der geometrischen Eigenschaften des Ziels infolge von Bewegungen ermittelt werden. Dafür wird die Kamera an der Effektorhalterung platziert (eye-in-hand). Somit kann die Entfernung der Kamera zum Ziel abzüglich des Abstands zwischen Kamera und Endlagenschalter für die Ansteuerung des Ziels verwendet werden. Ein Nachteil dieser Befestigungsart ist das Entstehen von Bewegungsunschärfe im gesamten Bild bei Bewegung des Roboters.

Durch den eingeschränkten Sichtbereich der Kamera kann der in Abschnitt 3.1.1 vorgegebene Grenzwert für den Winkel zur Normalen des Ziels systembedingt nicht überschritten werden. Durch die oben angesprochene waagerechte Haltung des Effektors beträgt zudem der Winkel zur durch den Taster verlaufenden Horizontalen stets 0° .

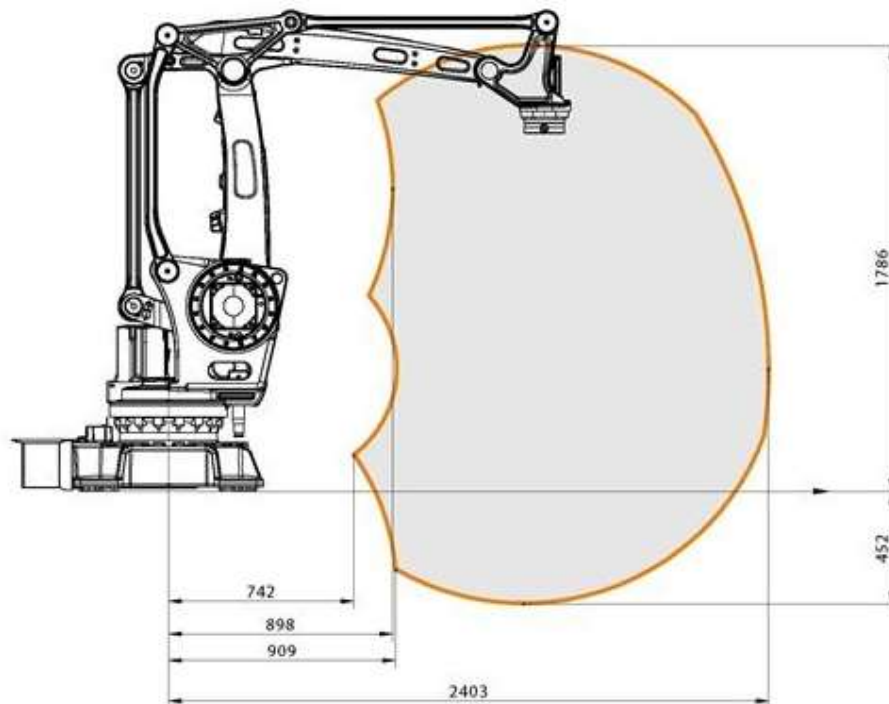


Abbildung 6: Seitenansicht eines ABB IRB460 [34]

Abbildung 7 zeigt den geplanten Aufbau des zu entwickelnden Prototyps mit allen verwendeten Komponenten und ihren Verbindungen untereinander. Zu den letzteren zählen mechanische Verbindungen bspw. zur Drehmomentübertragung sowie Leitungen zur Stromversorgung und Kommunikation. Der Antrieb besteht aus drei Motoren, die über Motortreiber vom Mikrocontroller gesteuert werden und über Zahnradgetriebe das Drehmoment auf die mechanische Struktur übertragen. Die Energieversorgung stellt dem Hauptrechner, dem Mikrocontroller und der Antriebseinheit Energie zur Verfügung, teilweise versorgen diese wiederum an sie angeschlossene Komponenten.

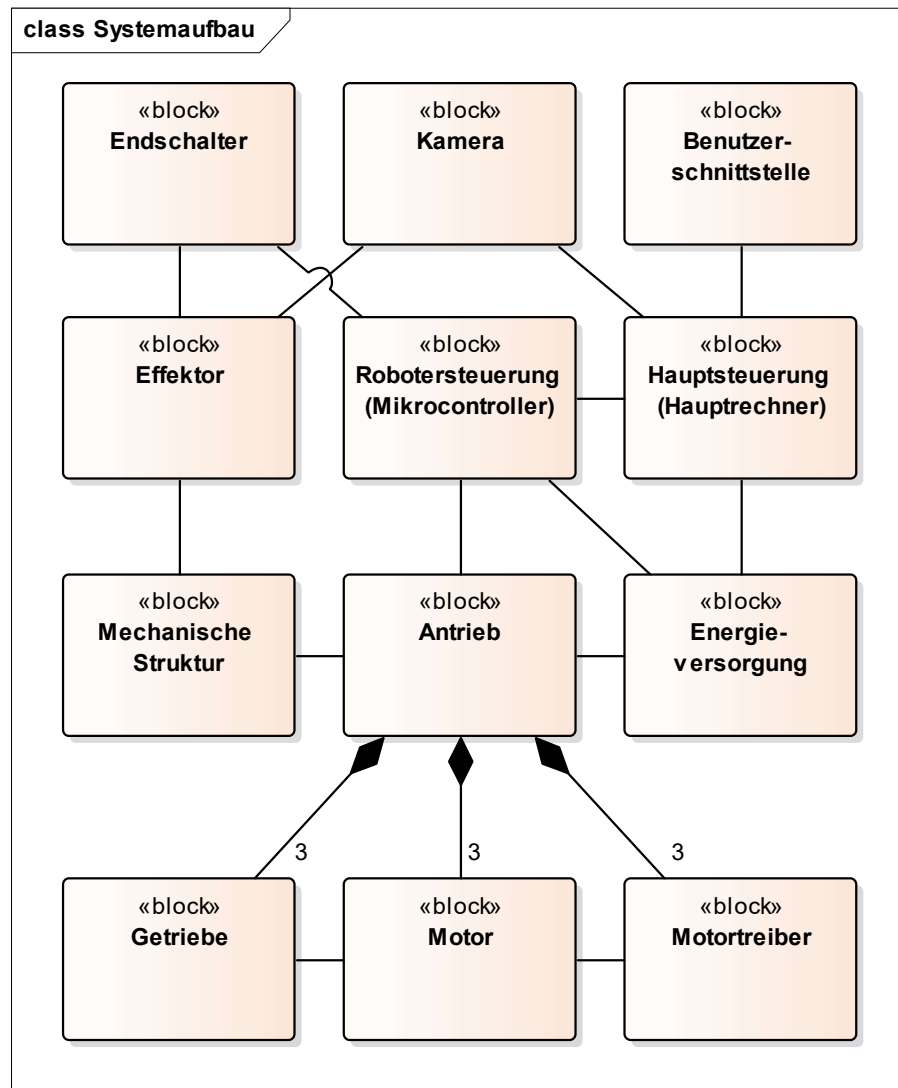


Abbildung 7: Aufbau des Prototyps mit Schnittstellen und Verbindungen zwischen den physischen Komponenten

3.2 Objekt- und Texterkennung

In Kapitel 2.2 wurde eine Auswahl an Verfahren zur Objekt- und Texterkennung vorgestellt. Zur Ausführungsgeschwindigkeit von Objektdetektionsverfahren auf Kleinstcomputern existieren nur wenige Untersuchungen. Daher sind Detektionsgeschwindigkeit und -rate verschiedener Verfahren auf solcher Hardware schwer vorhersagbar. In diesem Abschnitt werden aus diesem Grund zwei Methoden zur Erkennung und Lokalisierung von Fahrstuhltafeln entworfen und in Kapitel 4 evaluiert. Während klassische Verfahren aus dem Bereich des maschinellen Lernens häufig schnellere Detektionszeiten erlauben, erzielen Deep Learning Ansätze meist bessere Erkennungsraten. Daher dienen beide Bereiche als Grundlage für jeweils eins der Verfahren.

Nach erfolgter Detektion eines Tasters soll dessen Beschriftung ausgelesen werden. Dafür muss diese zunächst lokalisiert werden, bevor eine Zeichenerkennung erfolgen kann.

3.2.1 Maschinelles Detektionsverfahren mit vorgegebenem Merkmalsraum

In Abschnitt 2.2 wurde die Viola-Jones-Methode zur Objektdetektion als erster Vertreter der sogenannten Cascade Classifier (zu Deutsch etwa Kaskadenklassifikator) vorgestellt. Verfahren dieses Typs bieten ein gutes Laufzeitverhalten, sind bewährt und werden mittlerweile von freien Bibliotheken wie OpenCV unterstützt. Daher soll in dieser Arbeit ein Detektionsansatz nach dieser Methode entworfen werden. Das in [15] beschriebene Verfahren verwendet eine ähnliche Methode, nutzt aber zusätzlich einen Laserscanner für den Erkennungsprozess. Zusätzlich werden Eigenschaften von Fahrstuhltafeln nach ADA-Richtlinien vorausgesetzt, welche in Deutschland häufig nicht zutreffen. Daher ist dieses Verfahren in dieser Arbeit nicht anwendbar.

Bei Cascade Classifiern durchläuft jeder Bildausschnitt¹⁷ solange eine Kaskade von binären Klassifikatoren, bis er entweder zurückgewiesen oder der

¹⁷ Die Bildausschnitte entstehen nach dem Sliding-Windows-Prinzip durch Verschieben eines Suchfensters fester Größe über das Eingangsbild (vgl. Abschnitt 2.2).

gesuchten Klasse zugeordnet wird (vgl. Abbildung 8). Um die Effizienz des Verfahrens zu erhöhen, werden in den ersten Stufen möglichst viele Hintergrundregionen, also solche Bildausschnitte ohne Objekt der gesuchten Klasse, mit möglichst wenigen Berechnungsschritten aussortiert. Wichtig ist hier eine niedrige Falsch-Negativ-Rate, da falsch aussortierte Objekte der Klasse in späteren Stufen nicht wieder korrigiert werden können. Daraus resultiert eine zunächst hohe Falsch-Positiv-Rate, welche in späteren Stufen durch die Verwendung komplexerer Klassifikatoren reduziert wird [12].

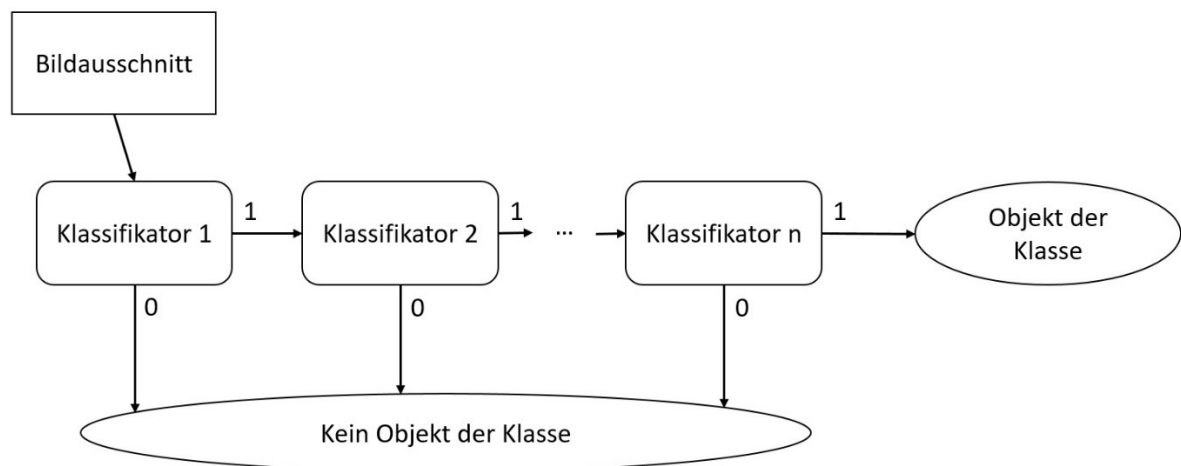


Abbildung 8: Kaskade von binären Klassifikatoren nach [12]

Die binäre Klassifikation in den einzelnen Stufen der Kaskade erfolgt anhand von Merkmalen des jeweiligen Bildausschnitts. Z. B. wird bei der Verwendung Haar-ähnlicher Merkmale nach Viola und Jones der Merkmalswert eines Rechtecktyps durch einfache Addition und Subtraktion der Pixelwerte im Grauwertbild unter den weißen und schwarzen Flächen des Rechtecks ermittelt¹⁸. Durch Rotation, Skalierung und Verschieben der Rechtecke ergeben sich für Bildausschnitte von 24 mal 24 Pixeln Größe, wie in der Arbeit von Viola und Jones verwendet, über 180.000 verschiedene Merkmale. Von diesen trägt aber nur ein geringer Teil zur Verbesserung der Klassifikation bei.

Zur Auswahl relevanter Merkmale und zum Training der Klassifikatoren der Kaskade setzen Viola und Jones eine Modifikation des AdaBoost-Verfahrens

¹⁸ Viola und Jones führen zur effizienten Berechnung der Merkmalswerte die Technik des Integralbilds ein, auf die hier aus Zeitgründen nicht näher eingegangen wird [12].

ein, welches eine geringe Laufzeit und gute Generalisierungseigenschaften verspricht [35]. Für jedes Merkmal wird anhand der Trainingsdaten ein Schwellwertklassifikator trainiert, welcher positive und negative Beispiele möglichst gut trennt. So entsteht eine Reihe von sogenannten schwachen Klassifikatoren, welche besser als der Zufall agieren, also mehr als 50 Prozent der Beispiele richtig klassifizieren. Die Schwellwertfunktion für das Merkmal f_j mit Grenzwert θ_j lautet für einen Bildausschnitt x :

$$h_j(x) = \begin{cases} 1 & \text{wenn } p_j f_j(x) < p_j \theta_j \\ 0 & \text{sonst} \end{cases} \quad (3.1)$$

Die Parität p_j bestimmt das Vorzeichen, gibt also an, ob der Grenzwert θ_j über- oder unterschritten werden muss. Eine vorgegebene Anzahl T der besten dieser schwachen Klassifikatoren wird vom AdaBoost-Verfahren iterativ zu einem starken Klassifikator kombiniert:

$$h(x) = \begin{cases} 1 & \text{wenn } \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{sonst} \end{cases} \quad (3.2)$$

Die Gewichtung α_t der schwachen Klassifikatoren entsteht durch die Fehlerfunktion ϵ_t , welche die Abweichung der Vorhersage über die ebenfalls gewichteten Beispielbilder x_i mit $y_i = 0$ für negative und $y_i = 1$ für positive Bilder summiert:

$$\epsilon_t = \sum_i w_i |h_j(x_i) - y_i| \quad (3.3)$$

In jedem Schritt des iterativen Verfahrens wird die Wichtung w_i des Bildes x_i verringert, wenn es korrekt klassifiziert wurde. Dadurch konzentrieren sich nachfolgende Schritte auf schwierig zu klassifizierende Beispiele.

Um nun wie oben erwähnt möglichst viele Hintergrundregionen mit wenig Rechenaufwand zu verwerfen, werden die starken Klassifikatoren der ersten Stufen aus möglichst wenigen Merkmalen gebildet. In der Originalarbeit von Viola und Jones zur Gesichtserkennung beträgt die Zahl T der Merkmale für die ersten fünf Stufen bspw. 1, 10, 25, 25 und 50. Weiterhin wird der von

AdaBoost verwendete Standardgrenzwert für den Klassifikator jeder Stufe¹⁹ so angepasst, dass die Falsch-Negativ-Rate minimiert wird. Die Falsch-Positiv-Rate soll mit jeder Stufe der Kaskade reduziert werden. Dies wird durch Vorgabe einer minimal akzeptierten Sensitivität²⁰ d_i und einer maximal akzeptierten Falsch-Positiv-Rate f_i je Stufe umgesetzt. Beim Training der Klassifikatoren jeder Stufe werden solange Merkmale, also schwache Klassifikatoren, hinzugefügt, bis durch eine Veränderung des Grenzwerts aus Formel 3.2 die Werte für d_i und f_i eingehalten werden. Für eine Kaskade mit K Stufen gilt dann für die Falsch-Positiv-Rate:

$$F = \prod_{i=1}^K f_i \quad (3.4)$$

Die Sensitivität bzw. Detektionsrate lässt sich ermitteln zu:

$$D = \prod_{i=1}^K d_i \quad (3.5)$$

Es ist zu beachten, dass diese Werte für den Trainingsdatensatz gelten und nicht verallgemeinert werden können. Das Training der Kaskade kann für eine bestimmte Anzahl Stufen oder solange erfolgen, bis ein vorgegebener Wert für die Falsch-Positiv-Rate unterschritten wird.

Während Viola und Jones Haar-ähnliche Merkmale für ihr Verfahren zur Objektdetektion nutzen, kann dieses leicht angepasst werden, um auch andere Merkmalstypen wie HOG oder LBP zu unterstützen. Je nach Problemstellung kann ein anderer Merkmalstyp zu besseren Ergebnissen führen. Wie in [14] zu lesen, führen HOG-Merkmale zu höheren Rechenzeiten und scheiden daher für die vorliegende Arbeit aus. Da das Training durch die Unterstützung von Bibliotheken wie OpenCV ohne großen Mehraufwand für LBP und Haar-ähnliche Merkmale durchgeführt werden kann, werden für die Implementierung in Kapitel 4 beide Typen eingesetzt und verglichen.

¹⁹ $\frac{1}{2} \sum_{t=1}^T \alpha_t$ aus Formel 3.2

²⁰ Die Sensitivität ist die Richtig-Positiv-Rate $\frac{r_p}{r_p + f_n}$. Es gilt $\frac{r_p}{r_p + f_n} + \frac{f_n}{r_p + f_n} = 1$. Bei Maximierung der Sensitivität wird also die Falsch-Negativ-Rate minimiert.

Um als Ergebnis des Trainingsprozesses gute Klassifikatoren zu erhalten, ist die Qualität der Trainingsdaten entscheidend. Zum einen ist eine hohe Zahl an positiven und negativen Beispielen erforderlich²¹, zum anderen ist die Bildqualität der Beispiele bezüglich Kriterien wie bspw. Auflösung, Belichtung und Bildrauschen wichtig. Das Aufbereiten der Trainingsdaten beinhaltet das Separieren und Auflisten der Positiv- und Negativbeispiele. Damit die Positivbeispiele als Input für das Training fungieren können, müssen diese manuell annotiert werden. D. h. für alle Positivbilder muss für jedes Objekt der gesuchten Klasse die Lage und Größe im Bild in Form eines Rechtecks angegeben werden. Für Fahrstuhltaster existiert kein Datensatz mit Beispielbildern, sodass diese selbst zusammengetragen werden müssen. Da der Trainingsprozess sehr rechenintensiv ist, kann dieser nicht auf einem Einplatinencomputer erfolgen, sondern muss auf einem Desktop-Rechner durchgeführt werden.

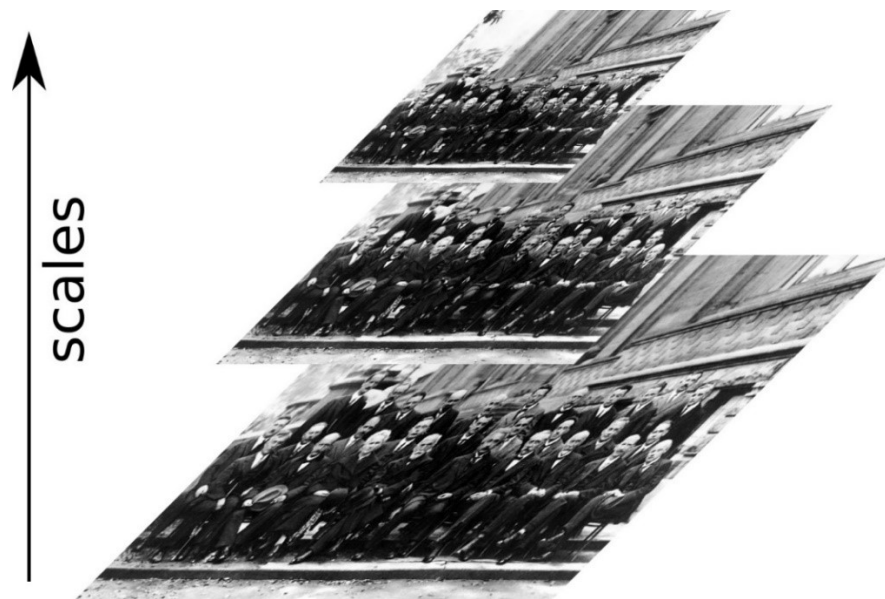


Abbildung 9: Bildpyramide eines Beispielbildes mit drei Skalierungen [36]

Das Ergebnis des Trainingsprozesses ist ein Kaskadenklassifikator, der für ein Eingabebild für alle detektierten Objekte der gesuchten Klasse die jeweilige Lage und Größe im Bild ausgibt. Dazu wird nach dem Sliding-Window-Prinzip ein Suchfenster fester Größe über das Bild verschoben und an jeder Position

²¹ Nach Möglichkeit mehrere hundert Positiv-Beispiele und das fünffache an Negativ-Beispielen.

der Klassifikator für den aktuellen Bildausschnitt aufgerufen. Da Objekte unterschiedlicher Größe erkannt werden sollen, wird das Eingabebild in Form einer Bildpyramide skaliert und der Suchprozess für jede Skalierung vorgenommen (vgl. Abbildung 9). Die Anzahl verschiedener Skalierungen kann über einen Faktor vorgegeben werden.

Um den Klassifikator einsetzen zu können, ist zunächst eine Vorverarbeitung der aufgenommenen Bilder notwendig. Da die Merkmalsextraktion mit Grauwerten arbeitet, muss das Eingabebild in ein Grauwertbild umgewandelt werden. Eine Größenanpassung kann unter Umständen die Detektionsdauer reduzieren, während Bildverbesserungsmaßnahmen wie z. B. Rauschreduzierung die Detektionsrate erhöhen können.

Abbildung 10 zeigt den Detektionsprozess von der Erzeugung der Bildausschnitte für den Klassifikator bis zur Ausgabe der Liste mit Positionen und Ausdehnungen der detektierten Objekte.

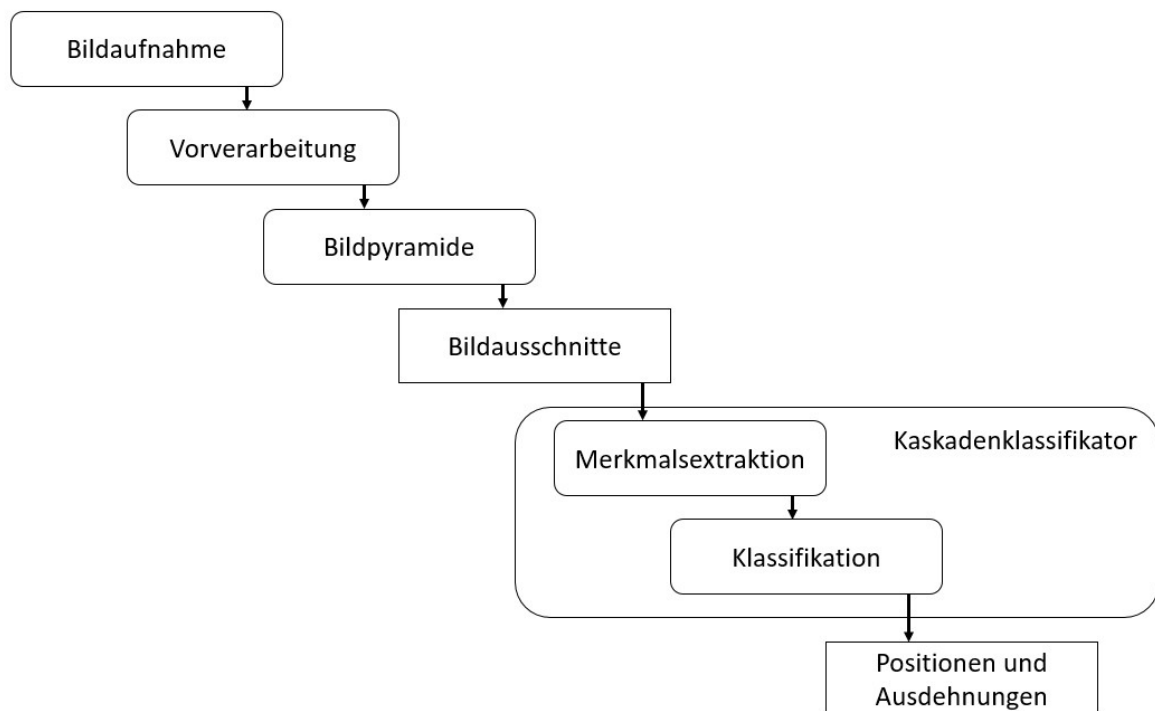


Abbildung 10: Ablauf der Tasterdetektion mit Kaskadenklassifikator

3.2.2 Deep Learning Ansatz zur Detektion ohne Merkmalsraumvorgabe

Ihren Namen tragen Convolutional Neural Networks aufgrund der Hauptoperation zur Berechnung der Neuronenaktivität. Durch diskrete Faltung (Convolution) wird in jeder Faltungsschicht (Convolutional Layer) der Input der Neuronen berechnet. Die erste Schicht verarbeitet das Eingabebild, welches als zwei- oder dreidimensionale Matrix vorliegt (Grauwert- oder Farbbild). Abbildung 11 zeigt die Faltungsoperation für den Input des zweiten Neurons, wobei durch sogenanntes Padding das Bild so erweitert wird, dass die Faltungsmatrix bzw. der Filterkernel auch in Randbereichen angewendet werden kann. Im Grunde handelt es sich bei den Eingangswerten für die Neuronen um die durch die Faltung entstandenen Pixelwerte. Durch die Anwendung von Faltungsmatrizen begrenzter Größe sind die Schichten lokal miteinander verbunden, d. h. ein Neuron reagiert nur auf Reize einer lokalen Umgebung der vorherigen Schicht (das sogenannte rezeptive Feld) [37].

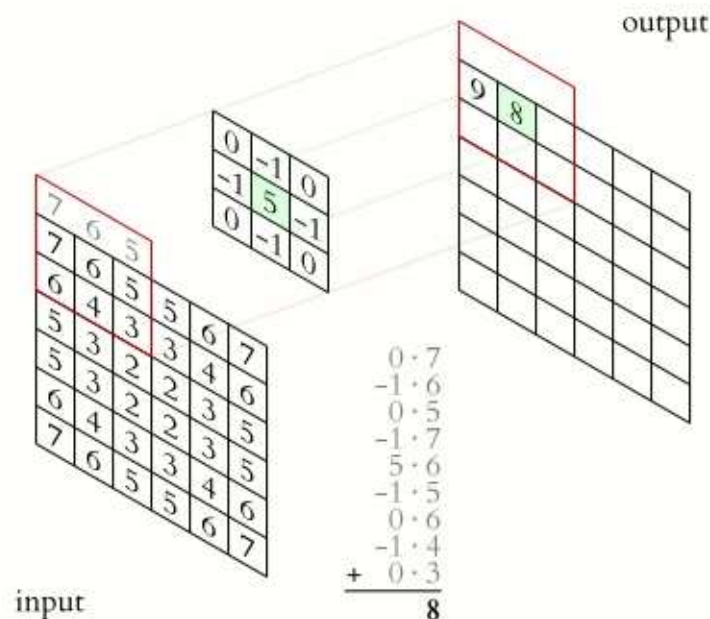


Abbildung 11: Faltung einer zweidimensionalen Eingangsmatrix mit einem 3x3-Filterkernel [38]

Im Anschluss an die Faltung wird durch Anwendung einer Aktivierungsfunktion auf die Eingangswerte der Neuronen deren Ausgangswert ermittelt. Häufig wird eine Funktion genutzt, welche negative Werte auf null setzt, um so das Schwellenpotential echter Neuronen zu simulieren²².

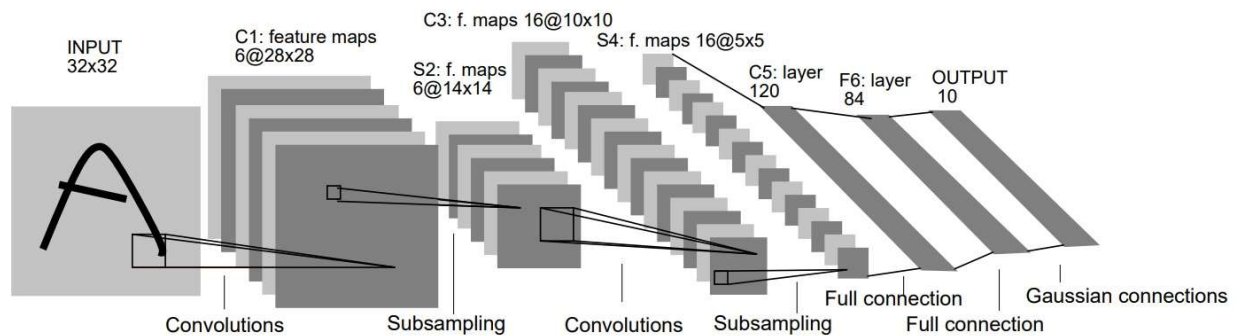


Abbildung 12: Architektur von LeNet-5, einem CNN zur Zeichenerkennung [18]

In der Regel wird der Filterkernel einer Faltungsschicht mit jeweils verschiedenen Gewichten mehrfach auf die Eingangsdaten angewendet, wodurch mehrere Merkmalsmatrizen, sogenannte Feature Maps, pro Schicht entstehen. In Abbildung 12 ist der Aufbau eines Convolutional Neural Networks beispielhaft dargestellt. Es besteht im Allgemeinen aus mehreren Faltungsschichten, auf welche Reduzierungsschichten (Pooling Layer) folgen. Diese werfen überflüssige Informationen, indem die Feature Maps bspw. in 2x2-Felder unterteilt und eine Mittelung der Werte durchführt (Mean-Pooling) oder nur das aktivste Neuron jedes Feldes behält (Max-Pooling). Abschließend folgen wie bei klassischen neuronalen Netzen vollständig miteinander verbundene Schichten (Fully-connected Layer), wobei die Anzahl der Neuronen der letzten Schicht in der Regel der Anzahl der verschiedenen Klassen entspricht. Im Gegensatz zu binären Klassifikatoren (vgl. Abschnitt 3.2.1) sind hier mehrere Objektkategorien möglich. Die Ausgabe der letzten Schicht sind die Wahrscheinlichkeiten, mit denen das Eingabebild zu den verschiedenen Klassen gehört. Das Training der Netze erfolgt überwacht mittels Fehlerrückführung (Back Propagation). Dabei wird der Ausgabewert des Netzes mit dem Sollwert

²² Statt der sogenannten Rectified Linear Unit $f(x) = \max(0, x)$ wird aus Gründen der Differenzierbarkeit (siehe Backpropagation weiter unten) meist die Approximation $f(x) = \ln(1 + e^x)$ genutzt.

verglichen und der Fehler bis zur Eingabeschicht zurück propagiert. Abhängig von ihrem Einfluss auf den Fehler wird die Gewichtung der Neuronen verändert.

Bei der bis hierhin beschriebenen Anwendung von Convolutional Neural Networks wird jeweils das gesamte Eingangsbild einer Klasse zugeordnet. Der intuitivste Weg, mehrere Objekte in einem Eingabebild zu detektieren, ist eine Suche nach dem Sliding-Window-Prinzip in mehreren Skalierungsstufen des Bildes (vgl. Abschnitt 3.2.1). Da bei diesem Ansatz aber sehr viele Bildausschnitte an das neuronale Netz übergeben werden, ist die Laufzeit entsprechend hoch. In Kapitel 2.2 wurden Ansätze zur Verbesserung der Detektionszeit genannt, z. B. das Vorfiltern von Regionen, die Objekte enthalten könnten. Den bislang besten Kompromiss aus Detektionsrate und -dauer liefern Single Shot Detektoren (SSD), welche in einem Durchlauf Objektregionen vorschlagen und diese klassifizieren. Daher soll für den Deep Learning Ansatz zur Tasterdetektion ein solcher Detektor eingesetzt werden. Eine detaillierte Beschreibung der Funktionsweise von SSDs ist für den geneigten Leser in [39] zu finden.

Einen entscheidenden Einfluss auf die Erkennungsrate und Laufzeit hat die Architektur des verwendeten Netzes, also die Anzahl, Reihenfolge, Art und Größe der miteinander verbundenen Schichten. Abbildung 13 zeigt einen aktuellen Vergleich populärer Netzwerkarchitekturen bezüglich ihrer Vorhersagegenauigkeit bei der Klassifikation von Bildern, der benötigten Rechenoperationen und des Speicherbedarfs. Da in der vorliegenden Arbeit eine geringe Rechenzeit Priorität hat, ist die Verwendung einer MobileNet-Architektur der vielversprechendste Ansatz. In Kombination mit SSD zur Detektion mehrerer Objekte erzielen sie nach [21] die schnellsten Detektionszeiten. MobileNet dient hierbei zur Merkmalsextraktion, während SSD zur Inferenz verwendet wird.

Da das Training eines tiefen neuronalen Netzes von Grund auf sehr viele Beispieldaten benötigt und auch auf leistungsfähiger Hardware mitunter Wochen dauern kann, soll in dieser Arbeit mittels Transfer Learning ein vortrainiertes Netz angepasst werden. Unter [40] findet sich eine Auswahl an mit dem

COCO-Datensatz (Common Objects in Context) trainierten Netzen. Dieser bietet ca. 1,5 Millionen Objektinstanzen in 80 Kategorien. Unter den vortrainierten Netzen befindet sich ein SSD-MobileNet Modell, welches in dieser Arbeit verwendet wird.

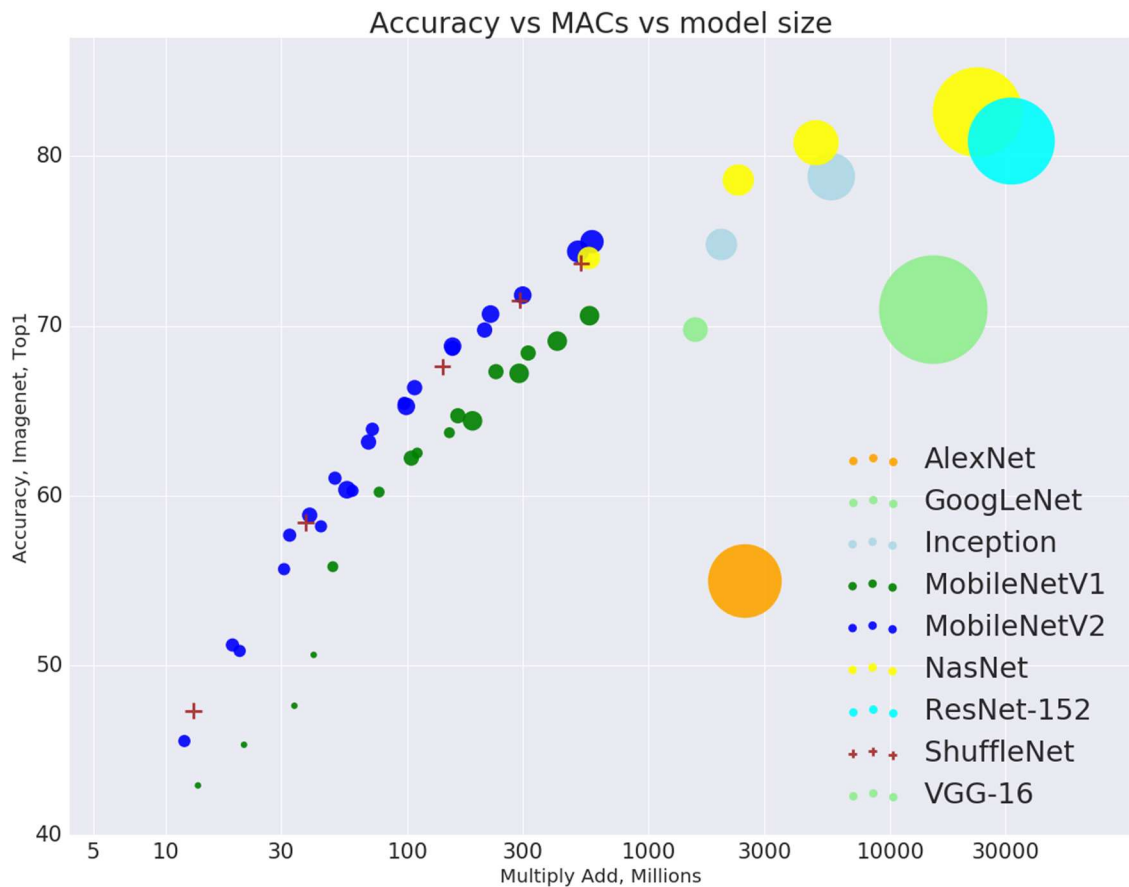


Abbildung 13: Vorhersagegenauigkeit, benötigte Rechenoperationen und Speicherbedarf (hier als Ausdehnung dargestellt) verschiedener CNN-Architekturen [40]

Im Unterschied zum in Abschnitt 3.2.1 beschriebenen Verfahren werden für das Training des ausgewählten Netzes nur Positivbeispiele benötigt, Negativbeispiele werden selbständig aus Hintergrundregionen generiert. Ansonsten sind die Ansprüche an die Trainingsdaten ähnlich, mit geringen Modifikationen können diese auch für den hier beschriebenen Deep Learning Ansatz genutzt werden.

3.2.3 Auslesen der Tasterbeschriftung

Um die Beschriftung von Fahrstuhltastern mittels Bildverarbeitung auszulesen, sind zwei Teilaufgaben zu lösen. Zunächst müssen Zeichen im Bild detektiert, also Größe und Position in Form von Begrenzungsboxen ermittelt werden. Anschließend ist die Text- bzw. Zeichenerkennung für die einzelnen Begrenzungsboxen durchzuführen. Eine Erkennung von Symbolen, wie bspw. „Tür schließen“ wird aus Zeitgründen in dieser Arbeit nicht betrachtet.

Textdetektion

Im Rahmen der International Conference on Document Analysis and Recognition (ICDAR) finden regelmäßig Wettbewerbe zur Textdetektion sowie -erkennung in Bildern natürlicher Umgebung statt. Zu den am häufigsten genannten Methoden zur Auffindung von Text zählen Extremal Regions (ER) bzw. Maximally Stable Extremal Regions (MSER), welche verbundene Regionen erkennen, die heller oder dunkler als ihre Umgebung sind [41]. Ebenfalls oft genannt wird der SWT Algorithmus (Stroke Width Transform), durch den die wahrscheinliche Strichstärke für jedes Pixel des Bildes bestimmt wird. Anhand dieser Strichstärken werden verbundene Komponenten bzw. zusammengehörende Regionen ermittelt [42]. Die beiden genannten Verfahren werden häufig mit anderen Bildbearbeitungsalgorithmen, wie z. B. Kantendetektion (oft Canny-Algorithmus), Weichzeichnungsfiltern (z. B. Gaussian Blur) oder Nicht-Maxima-Unterdrückung kombiniert. Da einige Implementierungen dieser Verfahren frei verfügbar sind, sollen sie für die Detektion von Tasterbeschriftungen eingebunden und getestet werden. Für diesen spezifischen Anwendungsfall liegen für kein Detektionsverfahren Untersuchungsergebnisse vor, daher ist ein weiterer auf die Einsatzumgebung abgestimmter Lösungsansatz als Alternative zu entwickeln. Dieser wird nachfolgend beschrieben.

Da Tasterbeschriftungen in verschiedenen Farben und Materialien möglich sind, ist hier eine texturbasierte Segmentierung wenig sinnvoll. Um Schrift und Bildhintergrund zu trennen, ist eine pixelorientierte bzw. histogrammbasierte Segmentierung vorzunehmen. Dazu wird zunächst das Bildrauschen durch einen Gauß-Filter verringert und anschließend ein Schwellwertverfah-

ren zur Binarisierung des Bildes angewendet. Gute Ergebnisse liefert das Verfahren nach Otsu, welches einen optimalen globalen Schwellwert ermittelt [9]. Eine rudimentäre Schräglauferkennung (Skew Detection) mittels Kantendetektion (Canny-Algorithmus) und Hough-Transformation wird zur Begradigung der Schrift eingesetzt. Im Anschluss sind zusammengehörende Regionen zu identifizieren, die durch sie umschließende Konturen beschrieben werden können. Zum Auffinden solcher Konturen dient bspw. das Verfahren nach Suzuki und Abe [43], welches in quelloffenen Bibliotheken zur Bildverarbeitung implementiert ist und hier eingesetzt wird. Durch modellbasierte Informationen können die gefundenen Regionen vorgefiltert werden. Dazu dienen die ungefähre Form bzw. das Seitenverhältnis einzelner Zeichen sowie eine für die spätere Klassifikation benötigte Mindestgröße. Die so ermittelten Regionen stellen einzelne Zeichen dar, für die im Anschluss die eigentliche Zeichenerkennung durchgeführt werden kann. Dazu werden rechteckige Begrenzungsboxen um die Zeichen erstellt und an die Zeichenerkennung übergeben. Enthält die Beschriftung eines Tasters mehr als ein Zeichen, ist dies an der Distanz und Lage der Begrenzungsboxen der Zeichen zu erkennen. Abbildung 14 zeigt die Arbeitsschritte des beschriebenen Verfahrens zur Zeichendetektion.

Wird die Detektion im ganzen Bild durchgeführt, ist eine Zuordnung von Beschriftungen zu Tastern über deren Position bzw. Distanz zu den Tastern möglich. Durch eine zu hohe Falsch-Positiv-Rate wird dies allerdings erschwert. Sollte die Suche im ganzen Bild zu schlechten Ergebnissen führen, ist für die vorliegende Arbeit die Beschränkung auf eine Detektion innerhalb der Begrenzungsboxen der gefundenen Taster in Betracht zu ziehen.

Als zusätzliche Möglichkeit zur Detektion der Fahrstuhltasterbeschriftungen ist noch das Trainieren eines neuronalen Netzes zu erwähnen. Da es sich prinzipiell um eine Objektdetektion handelt, kann ein Verfahren wie in Abschnitt 3.2.2 beschrieben eingesetzt werden. Allerdings ist hierzu das zeitintensive Erstellen eines Beispieldatensatzes notwendig²³. Weiterhin ist für die hier zu

²³ Dazu zählen das Zusammentragen von Positivbeispielen sowie das Annotieren der Position und Größe der Beschriftungen.

bearbeitende Aufgabe die Größe der entstehenden Modelle sowie deren Ausführungszeit zu beachten, da nur begrenzte Ressourcen zur Verfügung stehen. Um für den zu entwickelnden Prototypen eine möglichst gute Detektion von Tasterbeschriftungen zu erhalten, soll dieses Verfahren dennoch implementiert und evaluiert werden. Bei guten Ergebnissen ist ein Zusammenfassen von Taster- und Beschriftungserkennung in einem neuronalen Netz eine mögliche Option für zukünftige Entwicklungen.

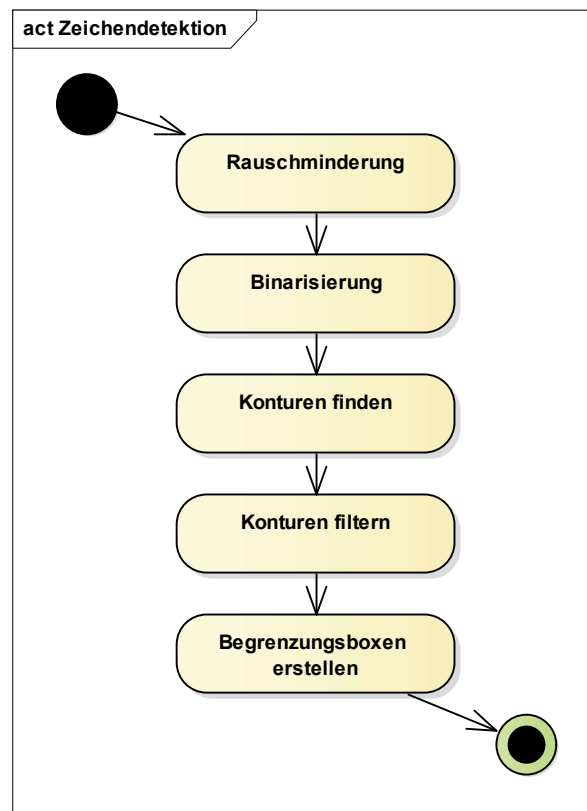


Abbildung 14: Arbeitsschritte des entworfenen Verfahrens zur Zeichendetektion

Zeichenerkennung

Da es sich bei der Beschriftung von Fahrstuhltastern meist um einzelne Zeichen oder Zahlen aus maximal zwei Ziffern handelt, ist eine zeichenbasierte Texterkennung anzuwenden. Dabei handelt es sich um eine Bildklassifikation, da jede Begrenzungsbox ein Zeichen enthält und einer Klasse zugeordnet wird. In dieser Arbeit werden die möglichen Klassen eingegrenzt und durch das Alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, E\}$ dargestellt. Dies schließt einige Symbole und Zeichen aus. Die daraus zusammensetzbaren Kombinationen decken aber die meisten Möglichkeiten zur Stockwerkswahl ab.

Der Chars74K Datensatz wurde für die Zeichenerkennung in natürlichen Bildern angelegt und umfasst ca. 74.000 Bilder, wovon die meisten allerdings Buchstaben zeigen [44]. Da einige mit diesem Datensatz trainierte neuronale Netze frei verfügbar sind, kann eines davon mit wenig Aufwand für den zu entwickelnden Prototyp eingesetzt und evaluiert werden.

Durch die Textdetektion liegen die Begrenzungsboxen binarisiert vor, somit kann auch die frei verfügbare, seit langem bewährte und in vielen Anwendungen genutzte Bibliothek Tesseract [45] eingesetzt werden. In neueren Versionen arbeitet auch diese mit neuronalen Netzen.

In Kapitel 4 erfolgt eine Auswertung der Detektionszeit und -rate der beiden Verfahren. Anhand dieser Daten wird eines für den Demonstrator ausgewählt.

3.3 Objektverfolgung

Nach der Detektion des gewünschten Tasters soll dieser vom Manipulator betätigt werden. Da die verwendete Kamera am Effektor befestigt ist, verändert sich das Bild des Ziels beim Verfahren des Effektors durch die geometrischen Transformationen Skalierung, Translation und Rotation. Zumindest zur Entfernungsberechnung muss mit dem hier entworfenen Verfahren ein Zwischenbild nach erfolgter Bewegung analysiert werden. Somit ist ein sofortiges direktes Ansteuern des Ziels nicht möglich und eine Objektverfolgung notwendig. Dabei sollen ähnlich wie bei der Objektdetektion Lage und Größe des Ziels ermittelt werden. Dies kann durch die Untersuchung der Bilder einer Videosequenz auch während der Bewegung oder nur anhand von Einzelbildern erfolgen, welche vor und nach der Bewegung aufgenommen wurden. Grundsätzlich könnte auch eine erneute Detektion des Ziels mit den im Abschnitt 3.2 entworfenen Verfahren erfolgen. Weil diese aber zeitintensiv sind, sollen hier schnellere Methoden betrachtet werden, die unter Umständen auch eine höhere Erfolgsquote bieten.

Die meisten Verfahren zur Objektverfolgung konzentrieren sich auf bewegte Ziele, welche von einer fest montierten Kamera beobachtet werden, sodass der

Bildhintergrund überwiegend unverändert bleibt. Da sich bei dem zu entwickelnden Prototyp die Kamera und somit auch der Hintergrund bewegt, sind diese hier nicht anwendbar.

Durch die Bewegung der Kamera entsteht Bewegungsunschärfe. Da die Aufhängung der Kamera nicht gedämpft ist, kann die Bildqualität durch Vibrationen weiter verschlechtert werden. Dies kann zu Problemen bei der Objektverfolgung anhand von während der Bewegung aufgenommenen Videosequenzen führen. Daher soll auch die Objektverfolgung mittels Einzelbildern betrachtet werden.

3.3.1 Echtzeit-Objektverfolgung in Videosequenzen

Aktuelle Methoden zur Objektverfolgung in Videosequenzen lernen, Veränderungen in der Erscheinung des Zielobjekts über die Zeit vorherzusagen. In einer Untersuchung aus dem Jahr 2017 werden mehrere Methoden aus dem aktuellen Stand der Technik hinsichtlich ihrer Erfolgsrate bei der Verfolgung, aber auch der Laufzeit auf einem Einplatinencomputer verglichen. Die erreichbare Bildrate während der Objektverfolgung liegt zwischen 2 und 15 Bildern pro Sekunde [46]. Damit ist eine bildbasierte Regelung des Manipulators (vgl. Abschnitt 2.3) nicht möglich.

Das Verfahren mit der höchsten Erfolgsquote bei den in der Studie verwendeten Videosequenzen ist Kernelized Correlation Filter (KCF), welches die Bilder mittels Fourier-Transformation zunächst in den Frequenzraum überträgt und anschließend Übereinstimmungen anhand von Optimalfiltern sucht. Bei Größenänderungen des Ziels versagt KCF allerdings häufig. Von den untersuchten Methoden ist nur Tracking-Learning-Detection (TLD) in der Lage, sich an Veränderungen der Größe des Ziels anzupassen. Dies wird durch die Verwendung von HOG-Merkmalen erreicht, da sie invariant gegenüber geometrischen Transformationen sind.

Um eines der Verfahren für den Demonstrator einsetzen zu können, muss diesem eine Videosequenz beginnend mit dem Start der Kamerabewegung zur Verfügung gestellt werden. Einerseits kann die Bildrate des Videos so ange-

passt werden, dass die Berechnungen parallel zur Bewegung des Manipulators ausgeführt werden. Dies führt zu schnelleren Antwortzeiten, kann aber in einer geringeren Erfolgsrate der Verfolgung resultieren. Andererseits ist es möglich, zunächst ein Video mit höherer Bildrate aufzunehmen und die Objektverfolgung im Anschluss an die Bewegung durchzuführen. Auf diese Weise dauert der Verfolgungsprozess länger, da mehr Bilder berechnet werden, dafür ist eine Verbesserung der Erfolgsquote möglich. Da Implementierungen der genannten sowie einiger weiterer²⁴ Verfahren für die verwendete Hardware verfügbar sind, können diese ohne wesentlichen Mehraufwand getestet werden.

3.3.2 Merkmalsabgleich und perspektivische Transformation zwischen Einzelbildern

Erfolgt die Aufnahme von Bildern ausschließlich vor und nach einer Bewegung der Kamera, führt dies in der Regel zu stärkeren geometrischen Transformationen, wie bspw. einer Translation oder Skalierung des Ziels zwischen den Einzelbildern. Die in Abschnitt 3.3.1 genannten Verfahren basieren aufgrund der angenommenen hohen Bildrate auf geringen Transformationen und sind je nach zurückgelegter Distanz zwischen den Aufnahmen nicht anwendbar.

Wie in Kapitel 2.3 beschrieben, kann die Transformation zwischen zwei Bildern anhand von Merkmalen ermittelt werden, die gegenüber Skalierung, Translation und Rotation möglichst invariant sind. Der erste Schritt zur Verfolgung eines Zielobjekts ist das Auffinden und die Beschreibung solcher lokaler Merkmale im Bildausschnitt der Zielregion vor der Bewegung und im Gesamtbild nach der Bewegung. Als Algorithmus für die Merkmalsdetektion und -beschreibung wird ein erprobtes Verfahren eingesetzt, für das eine Implementierung in einer freien Bildverarbeitungsbibliothek vorhanden ist. SURF liefert ähnlich gute Ergebnisse wie SIFT, bietet aber ein besseres Laufzeitverhalten. Weitere Verfahren wie ORB, BRISK oder BRIEF können bei Verwendung einer entsprechenden Bibliothek ebenfalls getestet werden.

²⁴ Dazu gehören MIL, Median Flow, Boosting und Mosse.

Der nächste Schritt ist ein Abgleich der Merkmale, wobei die Wahrscheinlichkeit der Übereinstimmung anhand der Nächste-Nachbarn-Klassifikation bestimmt wird. Als Maß für die Übereinstimmung gilt die euklidische Distanz zwischen den Merkmalsvektoren. Mit dem vom Autor des SIFT-Verfahrens vorgeschlagenen Verhältnistest²⁵ können Merkmale schlechter Qualität aussortiert werden [47].

Nach dem Merkmalsabgleich wird aus den übereinstimmenden Punkten in den beiden Aufnahmen mittels RANSAC-Methode [48] die Homographie-Matrix berechnet, mit welcher jeder Punkt aus dem ersten Bild einem Punkt im zweiten Bild zugeordnet werden kann:

$$\mathbf{x}' = H\mathbf{x} \quad (3.6)$$

In Abbildung 15 ist die Transformation eines Bildpunktes von der Kameraebene der ersten Aufnahme in die Ebene der zweiten, aus einem anderen Winkel erstellten Aufnahme angedeutet.

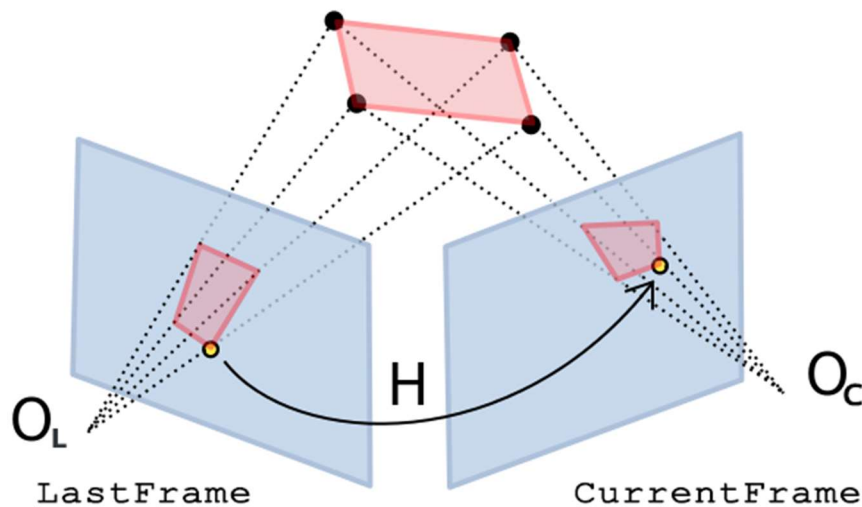


Abbildung 15: Homographie zur Transformation von Bildpunkten zwischen den Kameraebenen O_L und O_C bei Aufnahmen aus verschiedenen Winkeln [49]

²⁵ Hierbei werden die durch die Nächste-Nachbarn-Klassifikation ermittelten euklidischen Distanzen paarweise ins Verhältnis gesetzt. Für zwei Merkmale m_1 und m_2 mit den Distanzen d_1 und d_2 muss $d_1/d_2 < f$ gelten, damit m_1 nicht aussortiert wird. Der Faktor f beträgt häufig 0,8, kann aber frei gewählt werden.

Mit Hilfe dieser Matrix wird anschließend die ursprüngliche Bildregion des Zielobjekts auf das neue Bild mittels perspektivischer Transformation projiziert. Abbildung 16 zeigt das Ergebnis der Schritte Merkmalsdetektion, -beschreibung und -abgleich sowie perspektivischer Transformation anhand zweier Beispielbilder.

Die nach den hier beschriebenen Schritten erhaltene Zielregion im zweiten Bild dient wiederum als Ausgangspunkt für das Auffinden des Ziels in der nachfolgenden Aufnahme. Der Prozess der Objektverfolgung besteht somit aus dem Markieren des Ziels in einem Ursprungsbild und anschließender Wiederholung von Bildaufnahme und den hier angeführten Berechnungen.

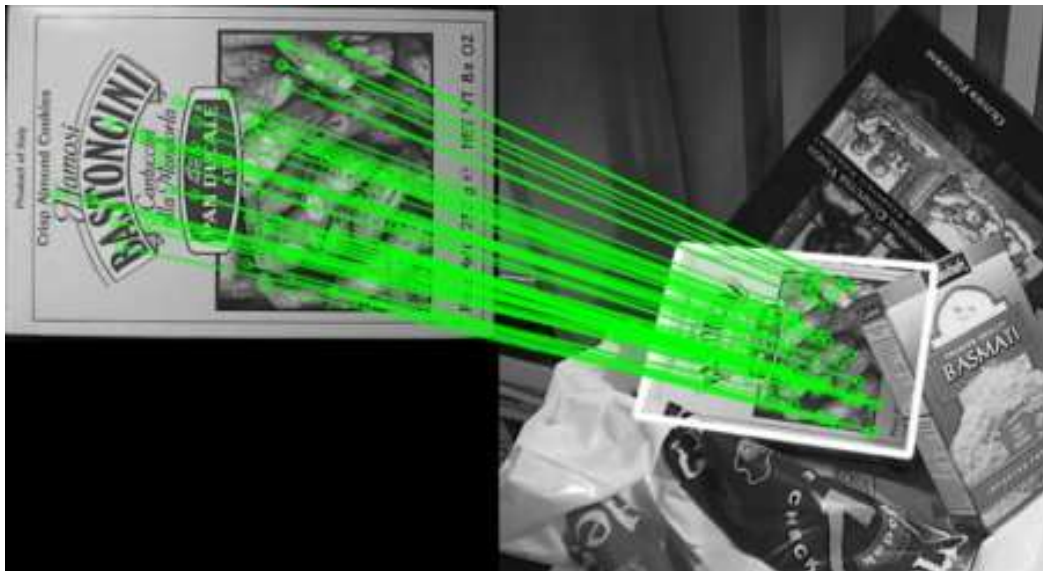


Abbildung 16: Beispiel für Merkmalsabgleich und perspektivische Transformation [50]

3.4 Entfernungsberechnung

Der in dieser Arbeit verwendete prototypische Aufbau beinhaltet nur eine Kamera, Sensoren für eine Abstandsmessung sind nicht vorgesehen. Da somit nur zweidimensionale Informationen vorliegen, ist eine Entfernungsberechnung nur über eine Bewegung der Kamera und anschließende Betrachtung geänderter Werte wie Bildgröße oder Winkel vom Bildzentrum zum Zielobjekt möglich. Nachfolgend werden zwei Methoden zur Berechnung der Distanz zum Ziel beschrieben. Aufgrund der in Kapitel 3.1 angesprochenen zu erwartenden Ungenauigkeiten durch die Beschaffenheit des verwendeten Roboterarms,

z. B. durch Spiel in den Gelenken und unpräzise Stellbewegungen der Motoren, sowie möglicher Abbildungsfehler der Kamera ist eine Kombination der Verfahren bzw. eine erneute Überprüfung nach dem Fahren einer Teilstrecke unter Umständen sinnvoll.

3.4.1 Berechnung durch Änderung der Bildgröße des Ziels

Befindet sich das Ziel zu Beginn im Bildzentrum, ist ein Verfahren des Effektors in dessen Richtung auf gerader Linie möglich. Mit Hilfe der Abbildungsgleichung und der veränderten Bildgröße kann die Objektentfernung ermittelt werden (Formel 3.7) [51].

$$g_2 = d \cdot \frac{B_1}{B_2 - B_1} \quad (3.7)$$

Hierbei entspricht d der zurückgelegten und g_2 der noch zurückzulegenden Entfernung sowie B_i der Bildgröße des Ziels vor und nach der Bewegung. Abbildung 17 verdeutlicht die Begriffe.

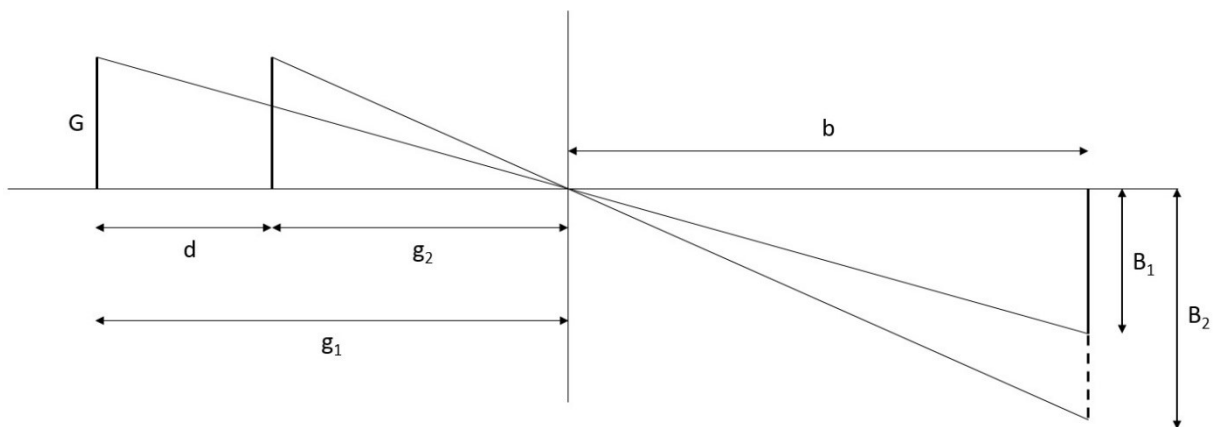


Abbildung 17: Projektionsverhältnisse bei Bewegung des Gegenstands G um die Distanz d

3.4.2 Berechnung durch Änderung des Winkels zum Ziel

Liegt das Ziel außerhalb des Bildzentrums, müssen zunächst die Winkel zur horizontalen und vertikalen Mittellinie bestimmt werden. Sind Sichtfeld der Kamera und Bildbreite bzw. -höhe bekannt, können die Winkel mit Hilfe der horizontalen und vertikalen Distanz zur Bildmitte berechnet werden. Abbildung 18 zeigt ein Beispiel für die Bestimmung des horizontalen Winkels von

der Bildachse der Kamera zum Ziel. Dargestellt ist der erfasste Bildausschnitt im Abstand l zur Kamera. Mit der Bildbreite W , der horizontalen Distanz zum Ziel d_h und dem horizontalen Sichtfeld der Kamera $HFOV$ gilt:

$$\tan \alpha = \frac{d_h}{l} \text{ und } \tan \frac{HFOV}{2} = \frac{\frac{W}{2}}{l} \quad (3.8)$$

Somit lässt sich der gesuchte Winkel α berechnen:

$$\tan \alpha = 2 \frac{d_h}{W} \cdot \tan \frac{HFOV}{2} \quad (3.9)$$

In Abbildung 18 sind die reale Bildbreite und Distanz zur Bildachse angegeben. Da in der Formel aber das Verhältnis der beiden betrachtet wird, können auch die Werte der Bildebene in Pixeln verwendet werden. Es wird vereinfachend angenommen, dass keine Verzerrungen oder andere Abbildungsfehler existieren.

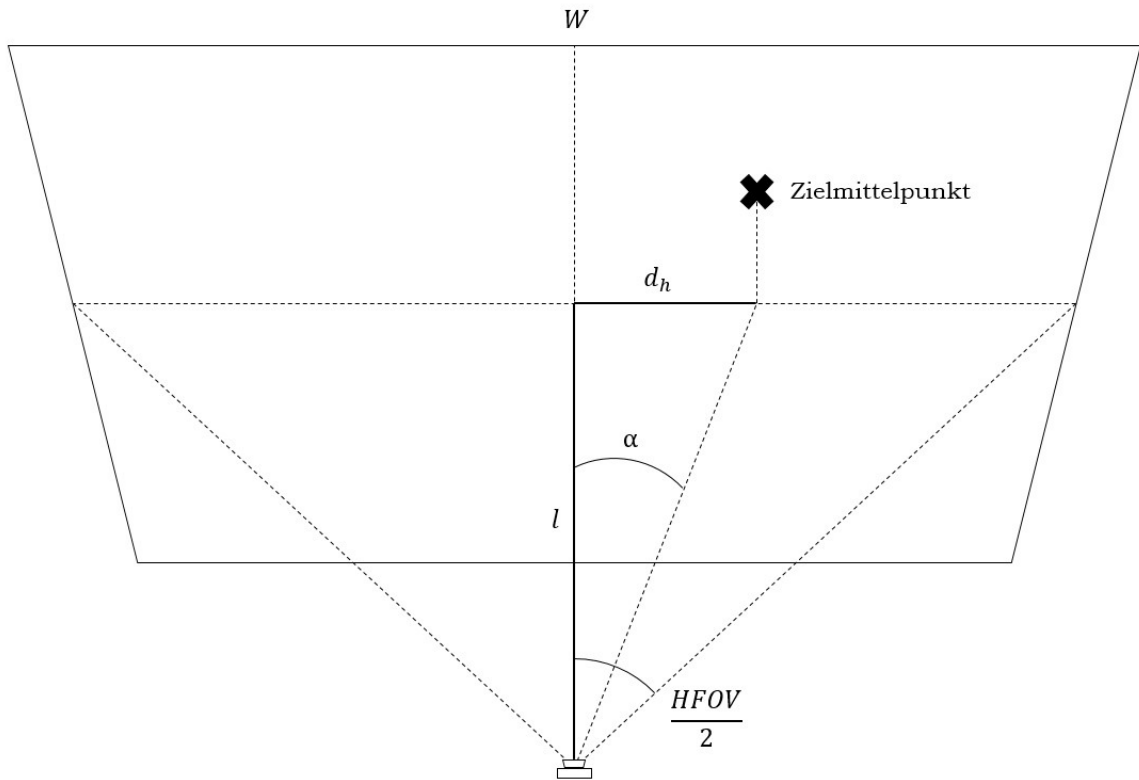


Abbildung 18: Ermittlung des horizontalen Winkels von der Bildachse zum Ziel

Für die folgenden Betrachtungen wird ein kartesisches Koordinatensystem in die Basis des Roboters gelegt. Es wird angenommen, dass das Ansteuern des Ziels aus der bekannten Startposition des Roboters erfolgt.

Die Objektentfernung lässt sich durch seitliches Verfahren auf der x- und z-Achse berechnen, wenn die y-Achse in der Startposition einer Vorwärtsbewegung entspricht (vgl. Abbildung 23). Durch die Veränderung des Winkels zwischen Zielzentrum und Kameraachse lassen sich die noch zurückzulegenden Distanzen auf allen drei Achsen ermitteln. Diese nachfolgend als α_1 und α_2 bezeichneten Winkel werden mittels Formel 3.9 berechnet.

Unter der Annahme eines Verfahrens auf der x-Achse um die Strecke x_1 ohne Drehung der Kamera und der Positionierung der Kamera am Tool Center Point (TCP), wie in Abbildung 19 links dargestellt, ergibt sich für die restliche auf der x-Achse zurückzulegende Distanz:

$$x_2 = \frac{\tan \alpha_2}{\tan \alpha_1 - \tan \alpha_2} \cdot x_1 \quad (3.10)$$

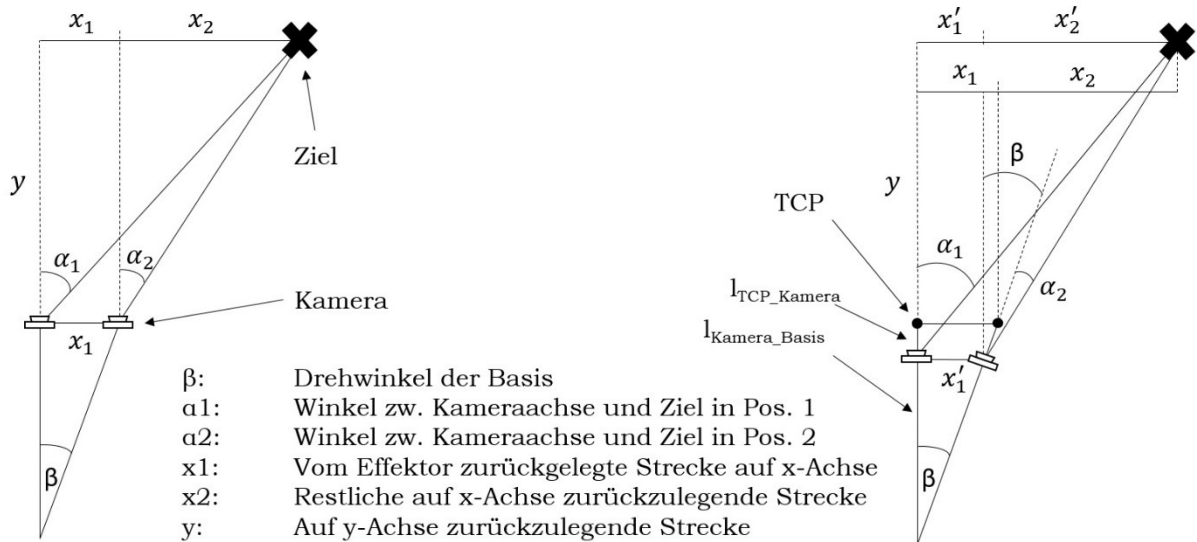


Abbildung 19: Schema zur Berechnung der Objektentfernung aus der Veränderung des Winkels zum Ziel (hier in der xy-Ebene), links ohne Kameradrehung und TCP

Da der Effektor des verwendeten Roboterarms allerdings nicht drehbar ist, erfolgt eine Drehung der Kamera, sodass mit einem anderen Winkel gerechnet werden muss (Abbildung 19 rechts):

$$\alpha'_2 = \beta + \alpha_2 \quad (3.11)$$

Weiterhin ist die Kamera am realen Manipulator ein Stück hinter dem Ende des Effektors bzw. dem TCP platziert, somit gilt für die von der Kamera zurückgelegte Distanz auf der x-Achse

$$x'_1 = \frac{l_{Kamera_Basis}}{l_{TCP_Kamera} + l_{Kamera_Basis}} \cdot x_1 \quad (3.12)$$

sowie

$$x'_2 = \frac{\tan \alpha'_2}{\tan \alpha_1 - \tan \alpha'_2} \cdot x'_1 \quad (3.13)$$

Hier ist x'_1 die von der Kamera zurückgelegte Distanz auf der x-Achse und x'_2 die Entfernung der Kamera zum Ziel auf der x-Achse. Die vom TCP noch zurückzulegende Strecke auf der x-Achse lässt sich dann ermitteln zu:

$$x_2 = x'_2 - (x_1 - x'_1) \quad (3.14)$$

Für die z-Achse erfolgen die Berechnungen analog zu den obigen Ausführungen. Für die auf der y-Achse zurückzulegende Distanz gilt:

$$y = \frac{x_1 + x_2}{\tan \alpha_1} - l_{TCP_Kamera} \quad (3.15)$$

3.5 Bewegungssteuerung

Die Entwicklung einer Steuerung für einen Roboter ist nicht Schwerpunkt dieser Arbeit. Da allerdings nicht mit einem fertigen System gearbeitet werden kann, ist sie zumindest so weit umzusetzen, dass die Hauptaufgabe des zu entwickelnden Prototypen die Betätigung von Fahrstuhltastern bzw. Modellen derselben erfüllt werden kann. Für den Manipulator wird eine offene Steuerung anstelle eines geschlossenen Regelkreises entwickelt, da weder Motoren noch Gelenke über Positionssensoren verfügen.

Die Bewegung der Gelenke des Manipulators erfolgt über Zahnradgetriebe, die von Schrittmotoren angetrieben werden. Um möglichst kleine Bewegungen durchführen zu können, werden diese im Mikroschrittbetrieb angesteuert. Für die Umrechnung eines einzustellenden Winkels für eines der Drehgelenke in die benötigte Anzahl an Schritten wird der Faktor $\mu_{Schritt}$ aus der Zähnezahl von An- und Abtriebsrad sowie der Anzahl Vollschriffe pro Umdrehung und Mikroschritte pro Vollschrift berechnet:

$$\mu_{Schritt} = \frac{Z_{Abtrieb}}{Z_{Antrieb}} \cdot n_{Voll} \cdot n_{Mikro} \cdot \frac{1}{2\pi} \quad (3.16)$$

Soll ein Winkel θ eingestellt werden, lässt sich die Anzahl an Schritten für den Motor wie folgt ermitteln:

$$n = \mu_{\text{Schritt}} \cdot \theta \quad (3.17)$$

Die mechanisch maximal einstellbaren Winkel der Gelenke sind experimentell zu bestimmen und in der Ansteuerung der Motoren zu berücksichtigen.

Die in Abschnitt 3.4 aufgeführten Berechnungen zur Ermittlung der Distanz zwischen Werkzeug und Ziel basieren auf kartesischen Koordinaten. Auch für die Steuerung des Roboters wird ein kartesisches Koordinatensystem verwendet, wobei das Weltkoordinatensystem in die Roboterbasis gelegt wird.

Da der verwendete Roboter nur drei angetriebene Achsen nutzt, kann die inverse Kinematik über den Kosinussatz bestimmt werden. Die Lage des als Werkzeug verwendeten Endschafters stimmt mit dem TCP überein, sodass der TCP die Zielposition (x, y, z) im Weltkoordinatensystem (x_0, y_0, z_0) anfahren soll. Die nachfolgenden Berechnungen basieren auf [52].

Der Drehwinkel θ der Basis zur y_0 -Achse (Abbildung 20 links) kann mit Hilfe der $\arctan2$ -Funktion ermittelt werden:

$$\theta = \text{atan2}(x, y) \quad (3.18)$$

Für die folgenden Berechnungen wird die Distanz r von der Basis zum TCP in der xy -Ebene benötigt:

$$r = \sqrt{x^2 + y^2} \quad (3.19)$$

Die gesuchten Winkel der Roboterarme zur Horizontalen α und β werden anhand der Seitenansicht durch die rz -Ebene ermittelt (Abbildung 20 rechts). l_1 und l_2 stellen die Armlängen dar, d die Länge der Effektorhalterung, welche stets horizontal gehalten wird. In dieser Ansicht beträgt die Distanz R von der Basis bis zum Beginn der Effektorhalterung

$$R = \sqrt{(r - d)^2 + z^2}. \quad (3.20)$$

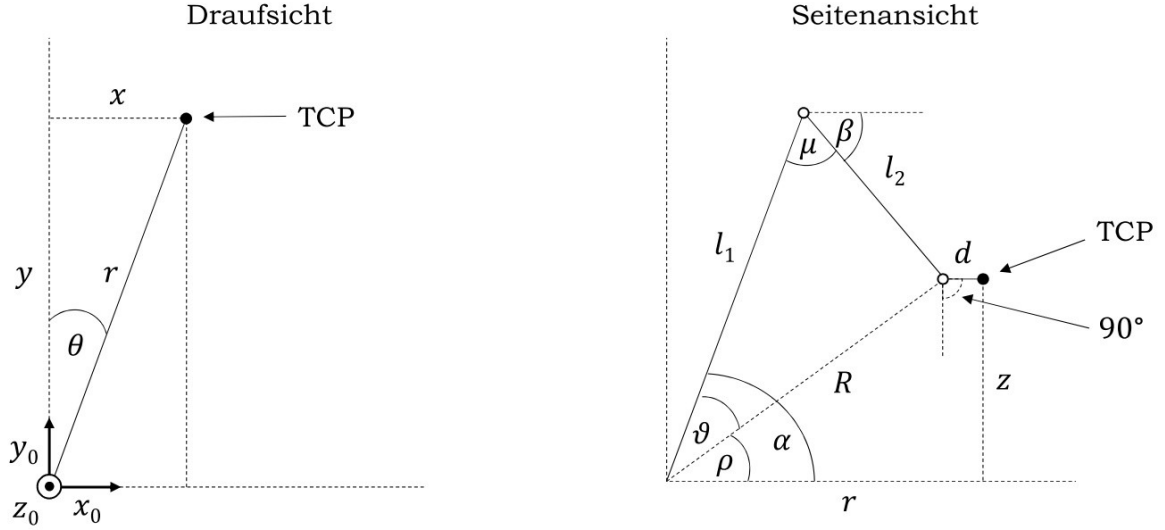


Abbildung 20: Ansicht der xy - und rz -Ebenen zur Berechnung der inversen Kinematik

Weiterhin gilt:

$$\rho = \text{atan2}(z, r - d) \quad (3.21)$$

$$\vartheta = \cos^{-1} \left(\frac{l_1^2 + R^2 - l_2^2}{2l_1 R} \right) \quad (3.22)$$

$$\mu = \cos^{-1} \left(\frac{l_1^2 + l_2^2 - R^2}{2l_1 l_2} \right) \quad (3.23)$$

Daraus lassen sich die Winkel der Arme zur Horizontalen ermitteln:

$$\alpha = \vartheta + \rho \quad (3.24)$$

$$\beta = \mu + \alpha - \pi \quad (3.25)$$

Diese können nun nach Formel 3.16 in die Positionen der Schrittmotoren umgerechnet werden. Die theoretisch mögliche zweite Lösung der Armstellungen ist durch die Restriktionen des mechanischen Aufbaus nicht umsetzbar.

Für die Bewegungssteuerung genügt in der vorliegenden Arbeit eine Punkt-zu-Punkt-Steuerung (PTP), da davon ausgegangen wird, dass sich keine Hindernisse im Arbeitsbereich befinden. Da an das Drehmoment und die Drehzahl der Motoren aufgrund des leichten Aufbaus des Manipulators nur geringe Ansprüche bestehen, werden Beschleunigungs- und Bremsrampen der Einfachheit halber nicht berücksichtigt. Daher ist eine synchrone PTP-

Bahnplanung einfach umzusetzen. Für den Motor mit dem größten zurückzulegenden Winkel φ_m wird die benötigte Zeitdauer t anhand einer vorgegebenen Winkelgeschwindigkeit ω_0 ermittelt:

$$t = \frac{\varphi_m}{\omega_0} \quad (3.26)$$

Für die übrigen Motoren kann die zu wählende Geschwindigkeit dann wie folgt berechnet werden:

$$\omega_i = \frac{\varphi_i}{t} \quad (3.27)$$

3.6 Integration der Teillösungen

Die in diesem Kapitel entworfenen Teillösungen werden in die Steuerung des Gesamtsystems integriert. Dazu erfolgt zunächst eine Zuordnung von Teilaufgaben zu Komponenten des Systems und anschließend eine Festlegung der Kommunikationsabläufe zwischen den Komponenten.

Die Robotersteuerung nimmt absolute und relative Positionierbefehle entgegen und versucht diese umzusetzen. Dazu wird die inverse Kinematik bestimmt und in Motorwinkel umgerechnet. Ist das Ziel außer Reichweite oder die Stellung der Gelenke nicht realisierbar, erfolgt eine Fehlermeldung. Andernfalls wird die Bewegung durch Steuerung der Motoren ausgeführt und nach erfolgter Bewegung eine Erfolgsmeldung zurückgegeben. Eine Aktivierung des Endlagenschalters führt zur Fahrt in die Startposition und wird ebenfalls gemeldet. Die Ausführung der Robotersteuerung erfolgt auf dem Mikrocontroller (vgl. Abbildung 7).

Aufgaben der Hauptsteuerung sind die Verarbeitung von Nutzereingaben, die Taster- und Beschriftungserkennung, die Entfernungsberechnung zum Ziel, die Objektverfolgung, die Kommunikation mit der Robotersteuerung und schließlich die Rückmeldung an den Nutzer. Die Steuerung des Gesamtsystems verläuft wie die Robotersteuerung in einem offenen Wirkungskreis. Eine Einbindung der Kamera in einen Regelkreis wäre zwar machbar, würde den Prozess der Zielansteuerung aufgrund wiederholt durchzuführender Berech-

nungen aber deutlich verlangsamen und zu benutzerunfreundlichen Antwortzeiten führen. Zudem ist die Objektverfolgung in der Nähe des Ziels nicht mehr realisierbar, da dort nur noch ein Ausschnitt des Fahrstuhl-tasters von der Kamera erfasst wird. Weiterhin wird durch den Endschalter zwar eine Betätigung erkannt, nicht aber ob das richtige Ziel getroffen wurde. Hierzu müsste eine eventuelle Rückmeldung des Fahrstuhls berücksichtigt werden. Ausgeführt wird die Hauptsteuerung auf dem verwendeten Einplatinencomputer. Der Ablauf des Hauptsteuerungszyklus ist in Abbildung 21 dargestellt.

Abbildung 22 zeigt die Abfolge der Kommunikation bzw. der Befehlsaufrufe im Gesamtsystem am Beispiel eines erfolgreichen Ablaufs von Eingabe, Erkennung und Ansteuerung des gewünschten Tasters. Die Kommunikation zwischen den Komponenten erfolgt synchron, da die richtige Reihenfolge eingehalten werden muss. So bedingt bspw. die Objektverfolgung des Ziels, dass die Bewegung des Manipulators abgeschlossen ist, da erst dann die gültige Entfernung zum Ziel berechnet werden kann.

Die Benutzerschnittstelle soll möglichst einfach gehalten werden und eine Eingabe von einzelnen Zeichen oder maximal zweistelligen Zahlen vorsehen. Dazu wird ein einfaches Tastenfeld an den Hauptrechner angeschlossen. Durch eine LED werden die Bereitschaft, das Bearbeiten einer Eingabe sowie Fehler signalisiert.

Der Mikrocontroller nimmt die Positionierungsbefehle über die serielle Schnittstelle entgegen. Ein beliebtes Format im 3D-Druck aber auch zunehmend bei Modellbau-Roboterarmen sind Varianten des Gerber-Formats (RS274 oder G-Code), welches ursprünglich für die Steuerung von Photoplottern und NC-Maschinen entwickelt wurde. Die Steuerbefehle enthalten hierbei in der Regel kartesische Koordinaten, können aber auch Befehle wie z. B. „Motor aus“ übermitteln.

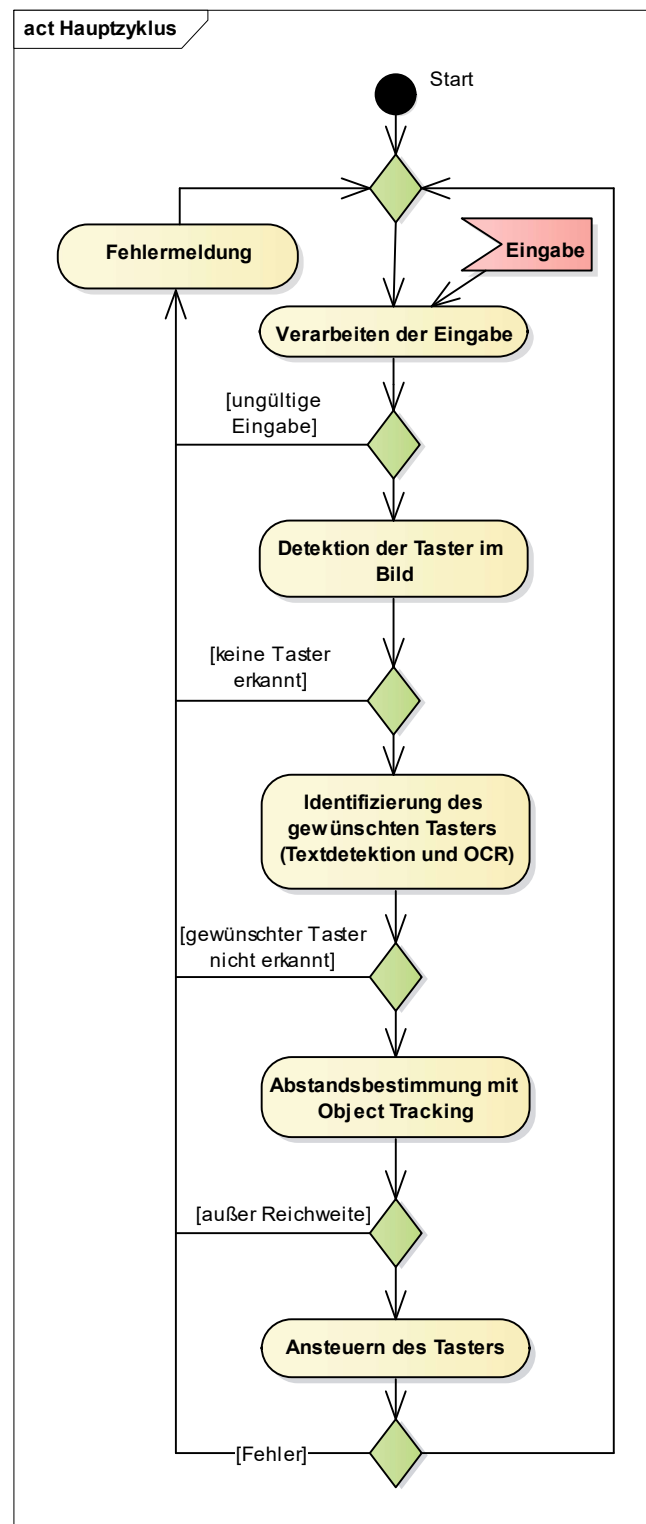


Abbildung 21: Zyklus der Hauptsteuerung

Für die vorliegende Arbeit ist ein geeigneter Befehlssatz zu definieren, der absolute und relative Positionsveränderungen berücksichtigt. Dieser Befehlssatz sowie die Rückmeldungen an die Hauptsteuerung sind in Anhang A aufgeführt.

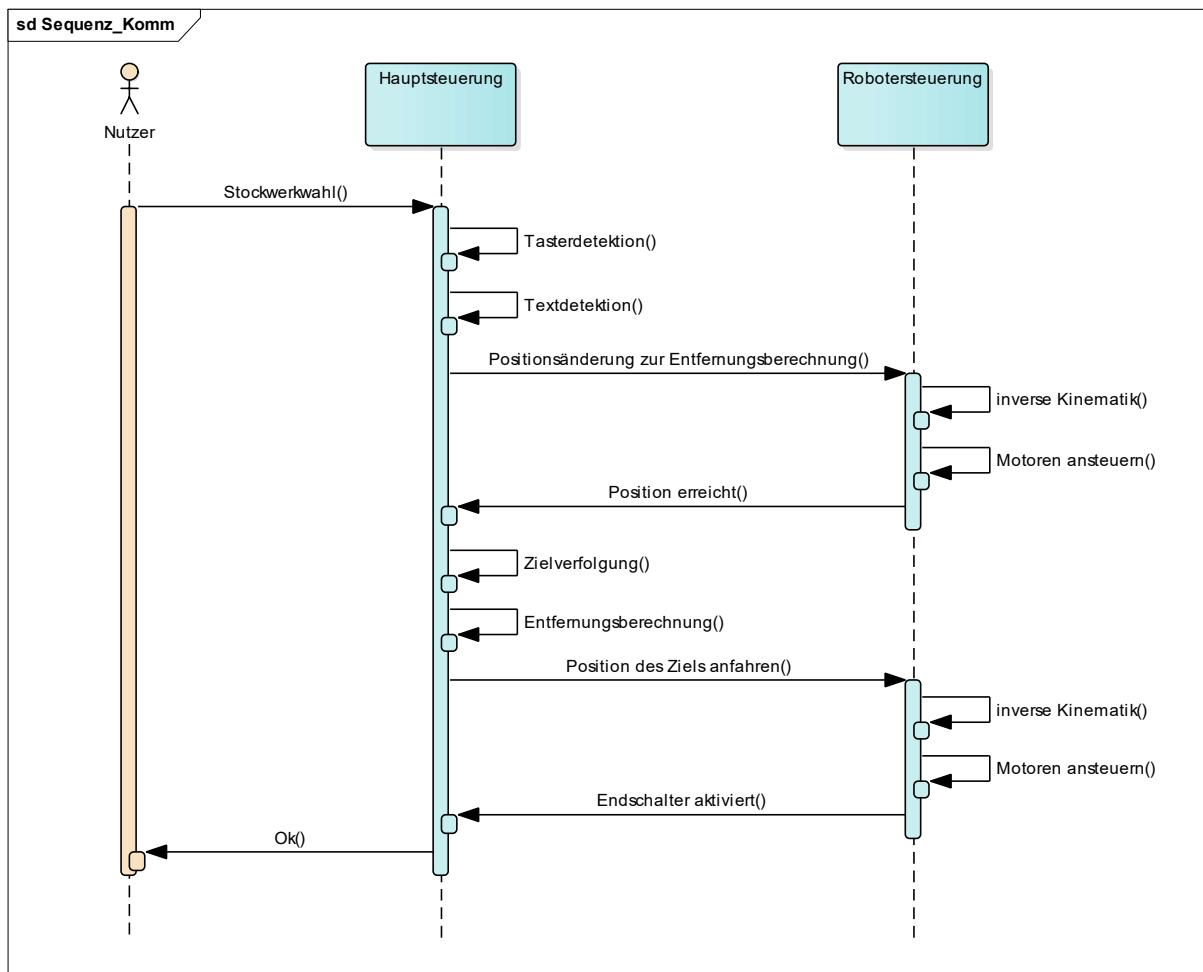


Abbildung 22: Kommunikationsablauf bei erfolgreicher Ansteuerung des Ziels bis zum Betätigen des Endlagenschalters

4 Prototypische Umsetzung

Das folgende Kapitel beschreibt die Implementierung des Prototypen, welcher die Assistenzfunktion an einem Modell demonstrieren soll. Nach grundlegenden Informationen zur verwendeten Hard- und Software werden Umsetzung und Ergebnisse der Teilaufgaben besprochen. Abschließend erfolgt eine Beschreibung der Integration der Teillösungen zum funktionsfähigen Gesamtsystem.

4.1 Hard- und Software

Die Auswahl der verwendeten Komponenten erfolgte in Abschnitt 3.1.2 vor dem Entwurf der Lösungen für die Teilaufgaben, da die vorhandene Hardware Einfluss auf die Möglichkeiten zur Umsetzung hat. An dieser Stelle werden einige Details bzw. Besonderheiten des physischen Aufbaus beschrieben. Ebenso wird auf einige Aspekte der Softwareimplementierung eingegangen. Der vollständige Quellcode dieser Arbeit ist der beigelegten CD zu entnehmen.

Da der Greifer der Originalversion des Robotermodells nicht benötigt wird und dessen Halterung eine stabile Montage ermöglicht, werden Kamera und Endschalter mittels Schrauben dort befestigt. Weiterhin wird eine an die Form der Motorwelle der eingesetzten Schrittmotoren angepasste Version der Antriebszahnräder verwendet, um ein Durchdrehen zu vermeiden.

Der zur Steuerung des Manipulators eingesetzte Mikrocontroller (Arduino Uno) ist über die serielle Schnittstelle per USB-Kabel mit dem Hauptrechner (Raspberry Pi 3) verbunden. Die Verbindung zu dem als Öffner eingesetzten Endlagenschalter erfolgt über zwei digitale Eingangspins. Über das Erweiterungsmodul (CNC-Shield) werden die Motortreiber zur Ansteuerung der Schrittmotoren über 4-adrige Kabel angesprochen. Das Erweiterungsmodul wird über digitale Ausgangspins mit dem Mikrocontroller verbunden. Die an der Effektorhalterung montierte Kamera ist über ein spezielles Flachbandkabel mit ausreichender Länge mit dem Hauptrechner verbunden. Das Tastenfeld zur Stockwerkeingabe wird über digitale Eingangspins ebenfalls an den Hauptrechner angeschlossen. Die Stromversorgung der Motoren erfolgt über

das CNC-Shield, welches von einem Netzteil gespeist wird. Hauptrechner und Mikrocontroller werden über separate Netzteile versorgt.

Da die Reichweite des Roboterarms beschränkt ist, erfolgt die Detektion und Betätigung von Fahrstuhltastrn innerhalb des Modellaufbaus anhand von Fotos. Diese werden an einer beweglichen Halterung in Lotrichtung befestigt. Abbildung 23 zeigt den verwendeten Modellaufbau mit einem Beispiel eines Bedienfeldmodells. Der Ursprung des verwendeten Koordinatensystems wird in die Achse des ersten Gelenks gelegt. Die y-Achse zeigt in Richtung Bedienfeldmodell, die z-Achse nach oben.

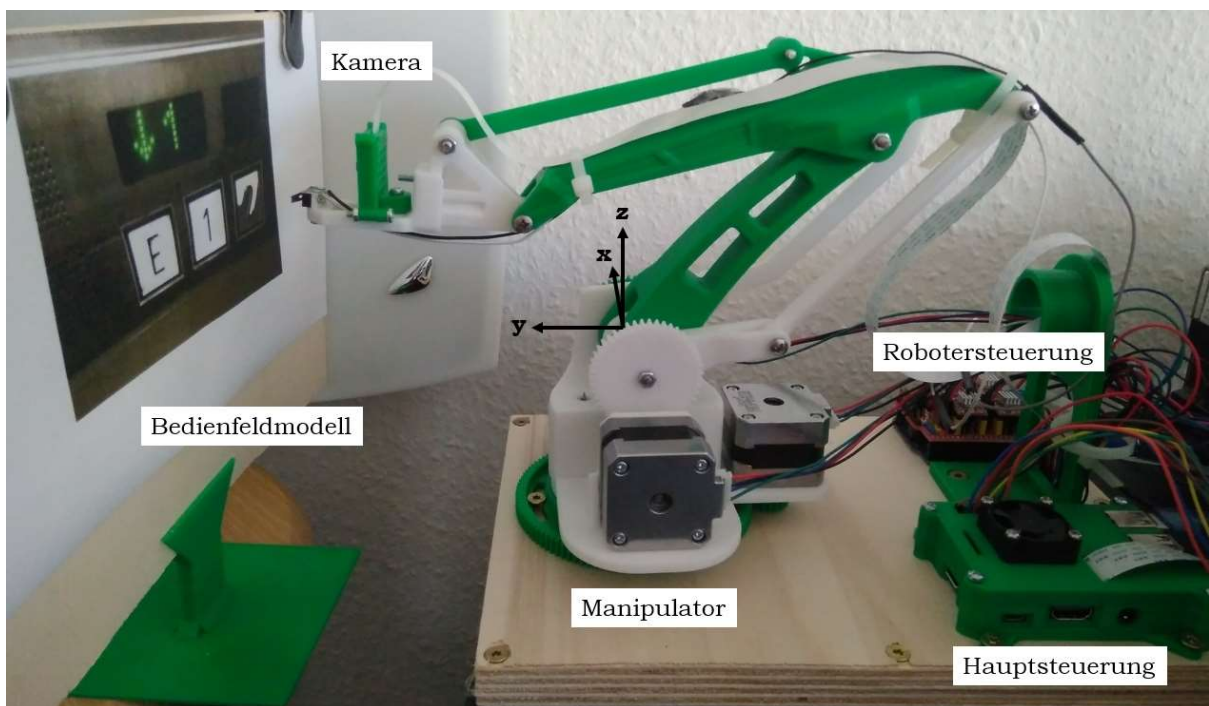


Abbildung 23: Aufbau des Prototyps mit Hauptkomponenten

Die Implementierung der Robotersteuerung erfolgt in der Programmiersprache C++, da die Anwendung von C bzw. C++ Standard für AVR Mikrocontroller ist und viele Bibliotheken in diesen Sprachen zur Verfügung stehen (bspw. für die Ansteuerung von Schrittmotoren). Die Verwendung von C++ bietet eine etwas größere Auswahl an Bibliotheken sowie zusätzlich die Möglichkeit zur objektorientierten Programmierung.

Die Hauptsteuerung wird in der Sprache Python entwickelt, da auch hier viele Bibliotheken insbesondere zur Bildverarbeitung und zum maschinellen Lernen verfügbar sind. Zwar resultiert die Verwendung von Python in einer minimal langsameren Programmausführung als bei C/C++, führt aber meist zu weniger Fehlern und erlaubt eine schnellere Entwicklung. Für Bildverarbeitungsaufgaben wird die Bibliothek OpenCV genutzt, da diese sehr umfangreich ist und zahlreiche Hilfestellungen existieren. Für Lösungsansätze mit tiefen neuronalen Netzen kommt die Bibliothek Tensorflow zum Einsatz, da diese viele Funktionen und vortrainierte Netze sowie zum Zeitpunkt der Entwicklung die beste Unterstützung für den Raspberry Pi bietet. Zu beachten ist, dass nicht alle Versionen der Bibliotheken und Programmiersprachen untereinander kompatibel sind²⁶.

4.2 Implementierung und Ergebnisse der Teilaufgaben

In diesem Abschnitt werden einige Details der Implementierung der Lösungen für die Teilaufgaben betrachtet. Weiterhin erfolgt eine Auswertung der erzielten Ergebnisse.

4.2.1 Tasterdetektion

Das in Abschnitt 3.2.1 entworfene Verfahren zur Tasterdetektion mittels Cascade Classifier wird mit Hilfe der Bibliothek OpenCV für Haar-ähnliche und LBP-Merkmale umgesetzt. Das Annotieren der Trainingsdaten erfolgt mit dem von OpenCV bereitgestellten Werkzeug²⁷. Für das Training²⁸ stehen ca. 550 aufbereitete Positivbilder mit insgesamt 2.100 Fahrstuhltastern sowie ca. 10.000 Negativbilder zur Verfügung. Als minimal akzeptierte Sensitivität wird $d_i = 0,999$ und als maximal akzeptierte Falsch-Positiv-Rate $f_i = 0,5$ gewählt. Die Anzahl der Kaskadenstufen wird auf 20 begrenzt und die Größe des Suchfensters auf 24 mal 24 Pixel festgesetzt.

²⁶ Als lauffähig hat sich bspw. die Kombination aus Tensorflow 1.4.1, OpenCV 3.4 und Python 3.5 herausgestellt.

²⁷ *opencv_annotation*

²⁸ Mittels *opencv_traincascade*

Nach einigen Stunden liegt der trainierte Cascade Classifier in Form einer xml-Datei vor, welche für die Detektion von Tastern genutzt werden kann. Vor dem Start des Suchprozesses wird das jeweilige Bild in ein Grauwertbild umgewandelt. Der Skalierungsfaktor für die Berechnung der Bildpyramide wird auf 1,05 gesetzt, d. h. die Bildgröße wird solange nach jedem Suchdurchlauf um 5 % reduziert, bis die Größe des Suchfensters erreicht bzw. unterschritten wird.

Für das Transfer Learning wird zunächst ein mittels Tensorflow vortrainiertes neuronales Netz heruntergeladen²⁹ und eine vorhandene Konfigurationsdatei für den Trainingsprozess angepasst. Insbesondere ist die Anzahl zu unterscheidender Objektklassen auf eins zu setzen, weiterhin wird die initiale Lernrate auf 0,004 festgelegt. Der Trainingsfortschritt kann mittels der Software TensorBoard überwacht werden. Während des Trainings werden in festen Abständen Kontrollpunkte erstellt, sodass bei einsetzender Verschlechterung des Netzes, z. B. durch eine Überanpassung, der beste Stand ausgewählt werden kann. Anhand des gewählten Kontrollpunkts wird anschließend der Inferenzgraph exportiert und steht für den Detektionsprozess zur Verfügung.

Die Ergebnisse der Trainingsprozesse für die oben genannten Verfahren sind in Tabelle 1 zusammengefasst. Zur Auswertung dienen ca. 50 separate Bilder mit insgesamt 120 Fahrstuhltastern. Einige Beispielbilder mit Markierungen der verschiedenen Verfahren sind in Anhang B zu sehen. Zu erwähnen ist, dass viele Testbilder, ähnlich wie die Trainingsbilder, in schlechter Qualität vorliegen. Insbesondere ist die Auflösung häufig gering.

Auffällig ist die Präzision des Deep Learning Verfahrens. Bei diesem werden keine falsch-positiven Klassifizierungen vorgenommen. Hier liegt allerdings auch die Gefahr, dass das Netzwerk schlecht generalisieren kann (also von den Trainingsdaten abweichende gesuchte Objekte nicht identifiziert, sogenanntes Overfitting). Dies kann auch durch ein zu langes Training begünstigt werden. Der mit Haar-ähnlichen Merkmalen trainierte Cascade Classifier erzielt zwar eine hohe Sensitivität, ist aber aufgrund der vielen falsch-positiven

²⁹ Zum Zeitpunkt der Implementierung ist die aktuellste Version SSD MobileNet v1.

Treffer ungeeignet. Die schnellste Detektionszeit erzielt der LBP-Classifier. Aufgrund der gezeigten Ergebnisse wird das Deep Learning Verfahren für den Demonstrator verwendet.

Tabelle 1: Ergebnisse der Verfahren zur Tasterdetektion

Detektionsmethode	Trainingsdauer ³⁰ [h]	Mittlere Detektionsdauer ³¹ [s]	Sensitivität (Richtig-Positiv-Rate) $r_p/(r_p + f_n)$ ³²	Genauigkeit (Präzision) $r_p/(r_p + f_p)$
Haar-Cascade	20	2,20	0,91	0,56
LBP-Cascade	2	0,60	0,74	0,81
SSD MobileNet v1	12	0,75	0,82	1,00

4.2.2 Textdetektion und -erkennung

Lokalisierung von Tasterbeschriftungen

Die Verfahren zur Textdetektion (vgl. Abschnitt 3.2.3) können außer dem Deep Learning Ansatz, für welchen TensorFlow eingesetzt wird, mit der Bibliothek OpenCV umgesetzt werden. Während MSER- und SWT-Algorithmus direkt verfügbar sind, existieren für alle Schritte des entworfenen konturbasierten Verfahrens wie z. B. Binarisierung nach Otsu oder Konturauffindung nach Suzuki und Abe Methoden in der Bibliothek. Für das Training des neuronalen Netzes erfolgt zunächst die Annotation der Bilder, wobei in 255 Positivbildern 879 Tasterbeschriftungen vorliegen. Training und Inferenz verlaufen analog zu dem Verfahren in Abschnitt 4.2.1.

Tests zur Detektion von Tasterbeschriftungen in Bildern mit mehreren Tastern ergeben für keines der Verfahren ein gutes Ergebnis. Während die SWT Implementierung auf einem Mittelklasse Desktop-Computer mehrere Sekunden für die Berechnung benötigt und somit für einen Kleincomputer nicht geeignet

³⁰ Auf einem Intel Core i5-6600K mit 16 GB Arbeitsspeicher und GTX 950 Grafikkarte

³¹ Auf einem Raspberry Pi 3

³² r_p : richtig Positive, f_n : falsch Negative, f_p : falsch Positive

ist, erkennt die Methode mit MSER kaum eine Beschriftung korrekt und scheidet ebenfalls aus. Das einfache Verfahren zur Konturauffindung ist das schnellste, es benötigt inklusive Vorverarbeitung des Eingabebildes ca. 30 ms auf dem Raspberry Pi 3. Sensitivität und Genauigkeit sind zwar besser als bei Verwendung des MSER-Algorithmus, aber bei Anwendung innerhalb des Gesamtbildes dennoch unbrauchbar. Ebenso verhält es sich für das Deep Learning Verfahren mit einer Detektionszeit von ca. 750 ms. Daher erfolgt an dieser Stelle keine genauere Betrachtung der Ergebnisse.

Aufgrund der Resultate im Gesamtbild wird die Textdetektion auf gefundene Taster beschränkt, d. h. es wird jeweils innerhalb der Begrenzungsboxen der Fahrstuhlknöpfe nach der Beschriftung gesucht. Dadurch können zwar Taster mit einer seitlich außerhalb liegenden Aufschrift nicht betätigt werden, für die übrigen wird die Erkennung aber vereinfacht. Wird nur in den von der Tastendetektion gefundenen Bildausschnitten gesucht, werden die Zeichen deutlich häufiger erkannt. Es erfolgt eine Anpassung des Verfahrens zur Konturauffindung, bei der bezogen auf die Tastergröße zu kleine und zu große Konturen herausgefiltert werden. Für die zu behaltenden Regionen gilt bezogen auf die Höhe h :

$$0,3 \cdot h_{Taster} < h_{Region} < 0,9 \cdot h_{Taster} \quad (4.1)$$

Abbildung 24 zeigt an einem Bildausschnitt eines vorher detektierten Tasters die Vorverarbeitung, die Schräglaufererkennung und schließlich die vom Verfahren zur Konturauffindung bestimmte Begrenzungsbox im rotierten Ursprungsbild.

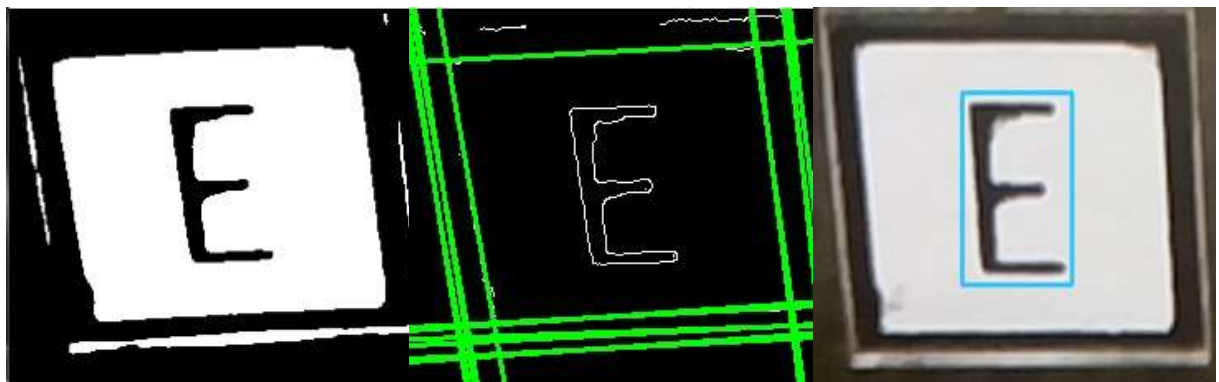


Abbildung 24: Vorverarbeitung, Schräglaufererkennung und rotiertes Ausgangsbild mit gefundenem Zeichen

Da diese Verarbeitungsschritte für jeden gefundenen Taster durchgeführt werden, erhöht sich die Gesamtdauer des Verfahrens proportional zur Anzahl der Taster³³. In einem Test mit 102 beschrifteten Fahrstuhltastern in 32 Bildern erreichen die beiden noch zur Auswahl stehenden Verfahren die in Tabelle 2 dargestellten Ergebnisse.

Tabelle 2: Ergebnisse der Verfahren zur Beschriftungsdetektion

Detektionsverfahren	Mittlere Detektionsdauer ³⁴ [ms]	Sensitivität (Richtig-Positiv-Rate) $r_p / (r_p + f_n)$	Genauigkeit (Präzision) $r_p / (r_p + f_p)$
Konturauffindung	90	0,80	0,76
Deep Learning	2400	0,83	0,78

Während das in Abschnitt 3.2.3 entworfene Verfahren zur Konturauffindung bei Anwendung auf innerhalb von Tastern liegende Bereiche immer einzelne Zeichen detektiert, liefert das Deep Learning Verfahren meistens die gesamte Beschriftung, teilweise aber auch Einzelzeichen. Somit wäre bei Wahl des Deep Learning Verfahrens eine zusätzliche Nachbearbeitung für einheitliche Ergebnisse erforderlich. Da aber das Verfahren zur Konturauffindung bei ähnlichen Resultaten deutlich schneller ist, wird dieses für den Demonstrator verwendet.

Auslesen von Tasterbeschriftungen

Die im letzten Abschnitt beschriebenen Begrenzungsboxen (vgl. Abbildung 24 rechts) dienen als Eingang für die Zeichenerkennung. Liegen mehrere Zeichen auf einem Taster, werden diese anhand ihrer Position sortiert und nach dem Erkennungsprozess konkateniert. Das Ergebnis wird mit dem gewünschten Ziel verglichen. Das auf dem Chars74K Datensatz basierende neuronale Netz wurde mit quadratischen Eingangsbildern trainiert, die an sie übergebenen

³³ Bei drei Tastern mit der Methode zur Konturauffindung z. B. auf $3 \cdot 30 \text{ ms} = 90 \text{ ms}$, mit dem Deep Learning Verfahren auf 2,25 s.

³⁴ Berechnungszeit für alle Taster eines Bildes auf einem Raspberry Pi 3.

Bilder müssen ebenfalls dieses Format besitzen. Daher treten teilweise Verzerrungen auf, die einen Einfluss auf die Prognose haben können. Die verwendete Tesseract-Implementierung (pytesseract als Wrapper für die Programmiersprache Python) liefert ohne weitere Konfiguration nur schlechte Ergebnisse. In ersten Tests wurde die Einstellung einiger Parameter nicht übernommen³⁵. Eine Recherche im Internet ergab, dass die Dokumentation der Bibliothek fehlerhaft ist und die Parameter anders als dargestellt übergeben werden müssen. Mit korrekter Konfiguration³⁶ sind mit Tesseract gute Prognosen erzielbar. Eine weitere Verbesserung ist zu erzielen, wenn ein gewisser Innenabstand des Zeichens zum Rahmen der Begrenzungsbox vorliegt (sogenanntes Padding). Ist dieser nicht vorhanden, wird er durch Vergrößern der Begrenzungsbox hinzugefügt.

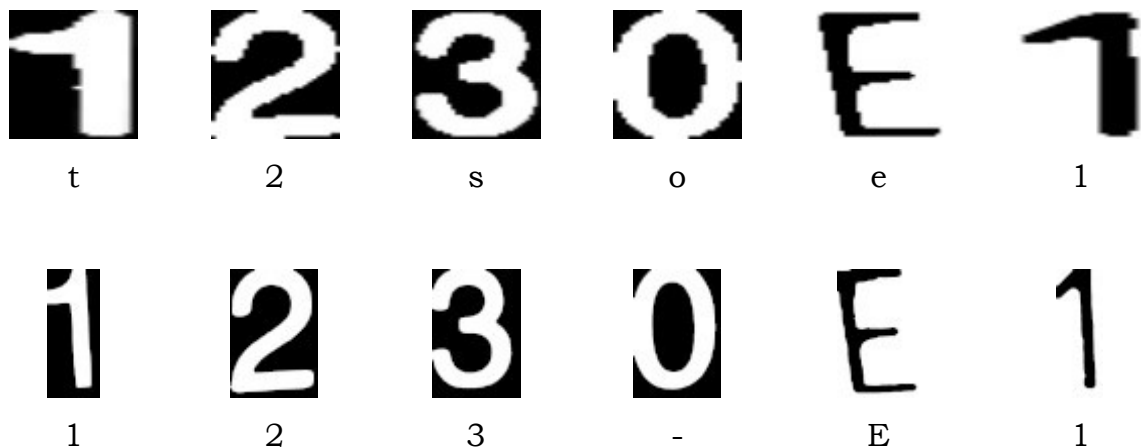


Abbildung 25: Zeichen mit zugehörigen Prognosen des Chars74K Modells (oben) und der Tesseract-Bibliothek ohne Padding (unten)

Mit hinzugefügtem Innenabstand beträgt die Erkennungsquote von Tesseract bei ca. 50 Beispielen nahezu 100 Prozent. Abbildung 25 zeigt die Resultate beider Verfahren anhand einiger Beispiele, bei Hinzufügen des Innenabstands verändert sich bei dem Chars74K Modell nichts, die Tesseract Bibliothek erkennt dann auch die Null korrekt. Es ist zu erkennen, dass das auf dem Chars74K basierende Erkennungsverfahren unbrauchbar ist. Nachteilig an der Tesseract Bibliothek ist die lange Ausführungszeit, welche pro Zeichen zwischen 1,5 und 1,7 Sekunden liegt. Bei der Verarbeitung von Bildern mit

³⁵ Bspw. kann das Eingabebild als einzelnes Zeichen, Wort oder Textzeile behandelt werden.

³⁶ Einzelzeichenerkennung und Vorgabe des Alphabets $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, E\}$.

mehreren Tastern kann dies zu erheblichen Verzögerungen führen. Um die Detektionszeit zu verkürzen, wird die Texterkennung bei der ersten Übereinstimmung mit dem gewünschten Ziel abgebrochen.

4.2.3 Objektverfolgung

Zunächst ist anzumerken dass das Sichtfeld der verwendeten Kamera nicht gut auf den Arbeitsbereich des verwendeten Manipulators abgestimmt ist, da es nur ein relativ kleines Betätigungsfeld erlaubt. Mit dem verwendeten Robotermodell können nur kurze Distanzen überbrückt werden, somit wäre hier ein größeres Sichtfeld vorzuziehen.

Nach dem Lokalisieren des gewünschten Tasters werden die Verfahren zur Objektverfolgung (vgl. Abschnitt 3.3) mit dessen Begrenzungsbox im Ausgangsbild initialisiert. Bei einem Aktualisierungsprozess versuchen sie, das Ziel in einem veränderten Bild erneut zu lokalisieren.

Von den in der OpenCV Bibliothek implementierten Verfahren zur Objektverfolgung KCF, MIL, TLD, Median Flow, Boosting und Mosse erreicht nur Median Flow bei der Verarbeitung einer von der Kamera bei Bewegung erstellten Videosequenz brauchbare Resultate. Allerdings ist mit diesem Verfahren die Ermittlung einer veränderten Zielgröße nicht möglich. Die anderen Verfahren verlieren aufgrund der Bewegungsunschärfe schnell das Ziel, ein Wiederauffinden ist in der Regel nicht möglich. Werden Einzelbilder zwischen Bewegungen aufgenommen, versagen alle genannten Verfahren bei etwas größeren zurückgelegten Distanzen. Insgesamt ist keines dieser Verfahren für den Prototypen geeignet.

Für das Verfahren mit Merkmalsextraktion und -abgleich wird nach einer Bewegung des Roboterarms eine vorgegebene Zeitspanne für ein Ausschwingen berücksichtigt, bevor das nächste Einzelbild erstellt wird. Erfolgt die Aufnahme zu schnell nach der Bewegung, entstehen Fehler durch unscharfe Aufnahmen, wie in Abbildung 26 oben zu sehen.

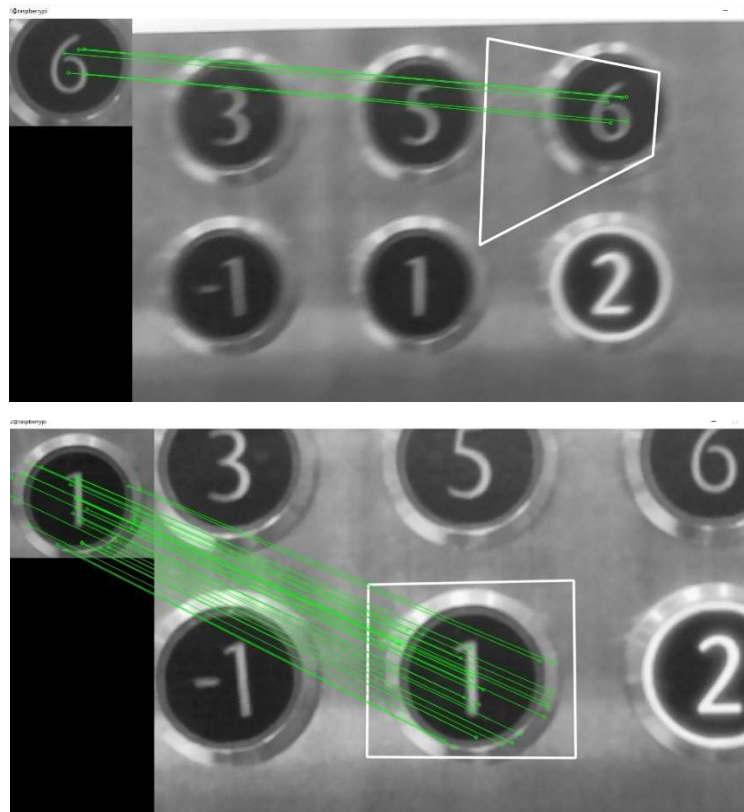


Abbildung 26: Beispiele des Verfahrens mit Merkmalsabgleich mittels SURF-Algorithmus (oben Fehlschlag durch Bewegungsunschärfe, unten erfolgreicher Versuch bei verkürzter Distanz zum Ziel)

Von den getesteten Methoden zur Merkmalsdetektion und -extraktion erzielen SIFT und SURF bei guter Ausleuchtung und scharfen Aufnahmen gute Ergebnisse, insbesondere findet eine veränderte Bildgröße Berücksichtigung. Der SURF-Algorithmus berechnet die Ergebnisse in den Tests schneller als der SIFT-Algorithmus. Beim Merkmalsabgleich erreichen die getesteten Verfahren Brute Force Matching und FLANN Based Matching gleich gute Resultate bei übereinstimmender Berechnungsdauer. Abbildung 26 unten zeigt die erfolgreich abgeglichenen Merkmale zwischen dem Zielausschnitt aus dem ursprünglichen Bild und dem aktuellen Bild bei Verwendung des SURF-Algorithmus in Verbindung mit Brute Force Matching. Im Mittel benötigt das Verfahren 2,2 Sekunden für das Lokalisieren des Ziels. Schnellere Detektions- und Extraktionsverfahren wie ORB, BRIEF oder BRISK liefern keine stabilen Resultate. Da bei entsprechenden Lichtverhältnissen die Wiederauffindungsquote nahezu 100 Prozent beträgt, wird trotz der längeren Laufzeit das hier vorgestellte Verfahren mit SURF und Brute Force Matching verwendet, anstatt

eine erneute Detektion mit dem in Abschnitt 4.2.1 gewählten Verfahren anzustoßen.

4.2.4 Distanzermittlung

Laut Datenblatt beträgt das horizontale Sichtfeld 53,5 Grad [30]. Die Bildbreite wird auf 1296 Pixel eingestellt. Die horizontale Distanz des Ziels zur Bildmitte ergibt sich aus der x-Koordinate des Mittelpunkts der zuvor ermittelten Begrenzungsbox. Auch die Bildgröße der in Abschnitt 3.4.1 beschriebenen Methode wird anhand dieser Begrenzungsbox ermittelt.

Die Berechnung der Entfernung zum Ziel basiert, wie in Kapitel 3.4 beschrieben, auf einer Relativbewegung der Kamera zum Ziel. Wird diese Bewegung manuell auf abgemessenen Strecken durchgeführt, liefern beide Methoden zur Entfernungsberechnung Abweichungen von wenigen Millimetern auf den einzelnen Achsen. Beim Verfahren über die Robotersteuerung treten teilweise größere Abweichungen und Ausreißer auf, da hier insbesondere bei kurzen Strecken Ungenauigkeiten auftreten (vgl. Abschnitt 4.2.5). Auch bei Erreichbarkeit des Ziels wird dieses sporadisch außerhalb der Reichweite des Aktors vermutet. Bei Verwendung eines zusätzlichen Zwischenstopps zur Nachberechnung der Objektentfernung wird diese häufig zu groß berechnet. Allgemein ist in den Tests keine Verbesserung der Ergebnisse durch Verwendung eines Zwischenstopps erkennbar. Durch die relativ geringe Positionier- und Wiederholgenauigkeit des verwendeten Roboterarms sowie eventuelle Abbildungsfehler der Kamera sind die unterschiedlichen Ergebnisse der Entfernungsberechnung erklärbar. Die Laufzeit der beiden Methoden liegt im einstelligen Millisekundenbereich und ist daher vernachlässigbar.

4.2.5 Firmware zur Bewegungssteuerung

Für die Steuerung von Roboterarmen sind einige einfache Projekte frei verfügbar, diese bieten aber nur eine manuelle Ansteuerung einzelner Gelenke. Eine Ausnahme bildet [53], welches eine Positionierung eines ähnlichen Robotermodells im kartesischen Raum erlaubt. Die Berechnung der inversen Kinematik basiert allerdings auf zwei gleich langen Armteilen, und ist daher in dieser Arbeit nicht verwendbar. Daher wird die Software um die in Abschnitt 3.5

entworfenen Berechnung der inversen Kinematik erweitert. Die Überprüfung der Erreichbarkeit eines Ziels wird neu implementiert, eine entsprechende Rückmeldung wird über die serielle Schnittstelle ausgegeben. Zu den verwendeten G-Code Befehlen werden Relativbewegungen und eine Startposition hinzugefügt. Weiterhin wird der verwendete Endschalter eingebunden und die Grenzwinkel der Gelenke nach einer Messung am Modell hinzugefügt. Da das Projekt ein anderes Erweiterungsmodul für die Ansteuerung der Motoren verwendet³⁷, erfolgt eine Anpassung an das CNC-Shield. Die Übersetzung der Getriebe wird nach Formel 3.15 berechnet. Die Motoren werden im Mikroschrittbetrieb angesteuert. Dabei wird ein Vollschrift in 16 Mikroschritte unterteilt. Im Vollschriftbetrieb benötigen die Motoren 200 Schritte für eine Umdrehung. Der Umrechnungsfaktor für den Motor des ersten Gelenks bzw. der drehbaren Basis berechnet sich bspw. wie folgt:

$$\mu_{Schritt_Arm1} = \frac{Z_{Abtrieb}}{Z_{Antrieb}} \cdot n_{Voll} \cdot n_{Mikro} \cdot \frac{1}{2\pi} = \frac{47}{19} \cdot 200 \cdot 16 \cdot \frac{1}{2\pi} = 1295,8$$

Für einen Winkel von 90 Grad ergibt sich so z. B. die Anzahl Schritte:

$$n_{Arm} = \mu_{Schritt_Arm1} \cdot \theta = 1295,8 \cdot \frac{\pi}{2} = 2035,4$$

Weitere Details, wie bspw. die Grenzwinkel der Gelenke, sind dem beigefügten Quellcode zu entnehmen.

Abbildung 27 zeigt die Softwarekomponenten der Robotersteuerung. Durch den modularisierten Entwurf sind funktional zusammengehörige Elemente in eigenen Komponenten gekapselt. Diese werden über Schnittstellen angesprochen, so greifen z. B. sowohl die Berechnung der Kinematik als auch die Interpolation der geplanten Bewegung auf das Modul Roboterstatus zu. Dieses stellt die aktuelle Stellung der Gelenke sowie die kartesischen Koordinaten bereit und gibt Auskunft über die Erreichbarkeit von Zielpositionen. Das Modul Kommunikation ist für die Verbindung zur Außenwelt bzw. zur Haupt-

³⁷ Es wird ein RAMPS Board verwendet.

steuerung über die serielle Schnittstelle zuständig. Die Hauptroutine übernimmt die Initialisierung des Programms sowie die zyklische Kontrolle des Ablaufs.

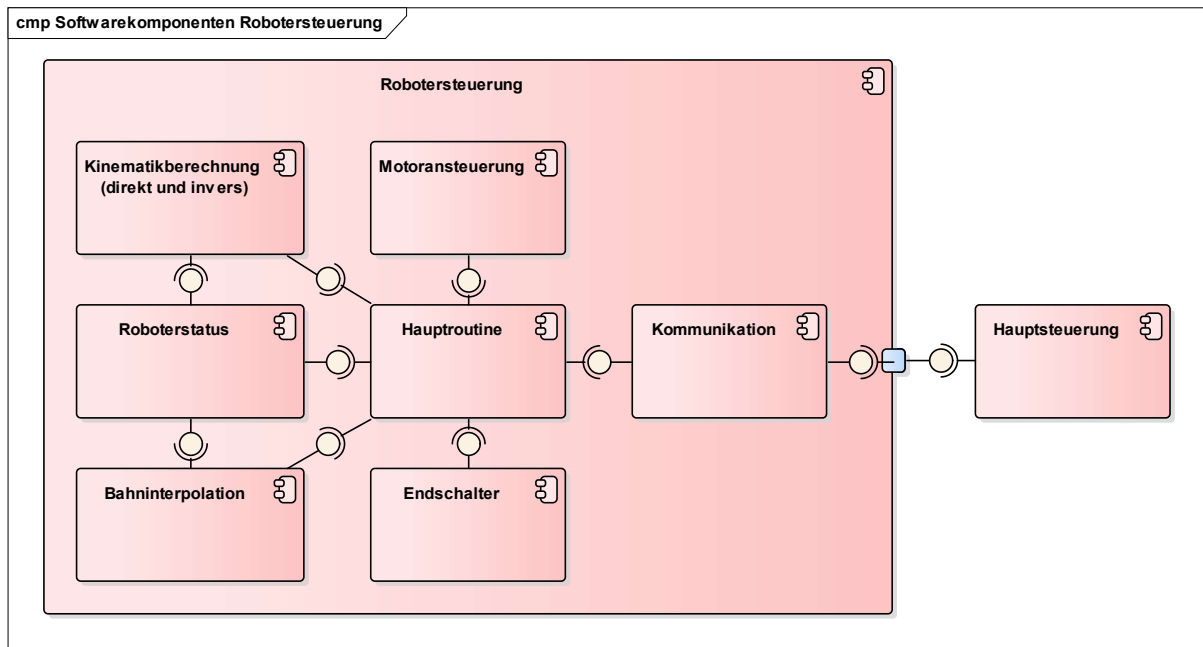


Abbildung 27: Komponenten der Robotersteuerung mit Schnittstellen

Als Startposition wird die in Abbildung 23 gezeigte Lage der Gelenke gewählt, da sich diese Position bedingt durch die Schwerkraft und den Anschlag der Gelenke einfach und reproduzierbar manuell einstellen lässt. Die Startkoordinaten werden durch Ausmessen ermittelt und lassen sich dem beigefügten Quellcode entnehmen.

Vor Start des Systems muss der Roboter manuell in die Startposition gebracht werden. Bei Messungen der Verfahrswege zeigt sich, dass die Positionier- und Wiederholgenauigkeit nahe an der Startposition sehr gering ist³⁸. Beginnt die Bewegung von einer Position in der Nähe der Startposition, reduziert sich die Abweichung stark. Daher wird als Ausgangspunkt der Tasterdetektion und -betätigung eine Position abweichend von der Startposition definiert³⁹.

³⁸ Abweichungen von ca. 4 mm bei einem Verfahrsweg von 10 mm kommen häufig vor. Hier scheint es sich um einen Trend bzw. Bias zu handeln, da immer eine zu kurze Strecke zurückgelegt wird.

³⁹ Hier genügt eine Positionierung ausgehend von der Startposition 20 mm weiter vorne auf der y-Achse und 15 mm weiter oben auf der z-Achse.

Da der Fokus dieser Arbeit nicht auf der Entwicklung eines genauen Positioniersystems liegt, erfolgt an dieser Stelle keine detaillierte Auswertung von Wiederhol- und Positioniergenauigkeit. Die Abweichung bei wiederholtem Anfahren derselben Position liegt unter ± 1 mm bei Fahrwegen von 150 mm auf einer Achse ausgehend von der oben erwähnten Ausgangsposition. Die absolute Positionsabweichung beträgt allerdings bis zu 5 mm bei Wegen von 150 mm auf einer Achse. Bei Verfahren in die Ausgangsposition kann es zu relativ großen Abweichungen kommen. Die Ursachen können unter anderem ein falsches Ausmessen der Startposition, unpräzise Motoren, Schrittverluste durch Weglassen von Beschleunigungs- und Bremsrampen, ungleichmäßig bzw. ungenau gedruckte Teile oder das vorhandene Spiel in den Gelenken sein. Für die Demonstration der in dieser Arbeit entwickelten Funktionen ist die Genauigkeit des Manipulators ausreichend.

4.3 Gesamtsystem

Abbildung 28 zeigt die Komponenten der Hauptsteuerung, welche auf die externe Robotersteuerung zugreift. Das Modul Roboterbewegung übersetzt die gewünschte Zielposition bzw. die auszuführende Bewegung in einen G-Code ähnlichen Befehl (vgl. Anhang A) und sendet diesen über die serielle Schnittstelle an die Robotersteuerung. In der Komponente Tastenfeld werden Benutzereingaben verarbeitet und das gewünschte Stockwerk ausgegeben. Die Tasterdetektion nutzt das neuronale Netz zum Auffinden aller Fahrstuhltaster im Bild, während die Beschriftungserkennung innerhalb der gefundenen Taster Zeichen lokalisiert und erkennt. Sämtliche Funktionen zur Bildverarbeitung wie Bildaufnahme, Vorverarbeitungsfunktionen oder Merkmalsabgleich werden vom gleichnamigen Modul angeboten, welches seinerseits für geometrische Berechnungen auf die entsprechende Komponente zugreift. Die Funktionen der Module Zielverfolgung und Distanzberechnung entsprechen ebenfalls ihrer Namensgebung. Die Hauptroutine ist wie auch bei der Robotersteuerung für die Steuerung des logischen Ablaufs zuständig (vgl. Abbildung 21).

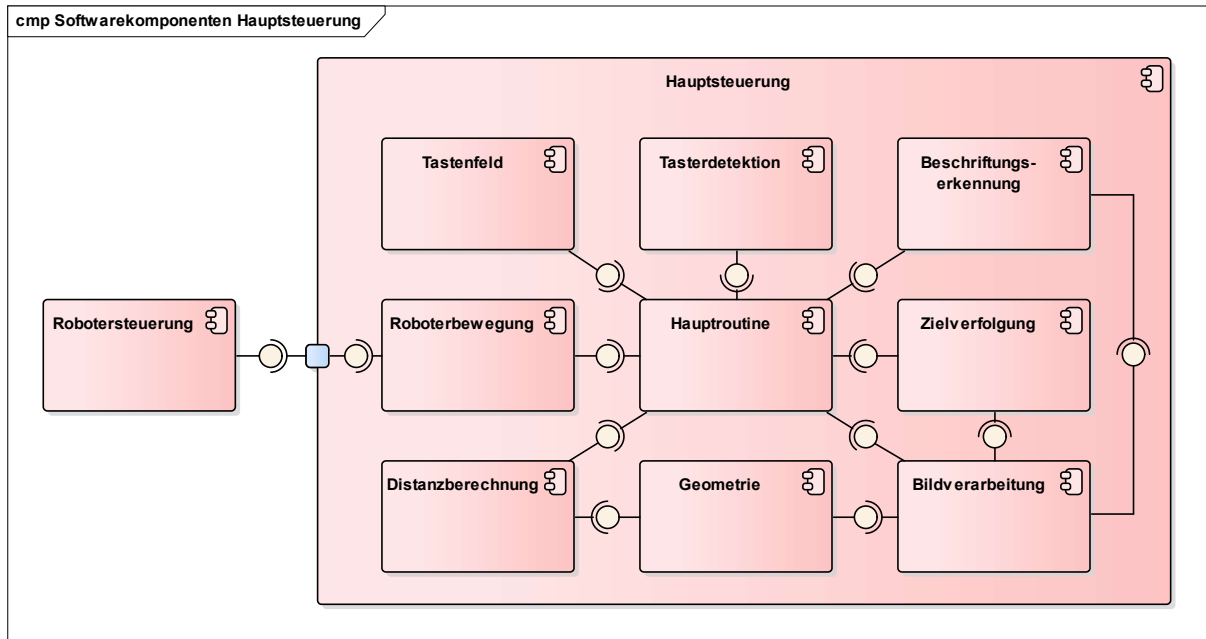


Abbildung 28: Softwarekomponenten der Hauptsteuerung

Für die Bildaufnahme ist die Bildautomatik bzw. automatische Helligkeitsanpassung der Kamera zu deaktivieren, da es sonst zu deutlichen Unterschieden in der Belichtung zwischen einzelnen Aufnahmen kommen kann. Diese Belichtungsunterschiede wiederum können einen negativen Einfluss auf die Tasterdetektion und die Zielverfolgung haben.

Für die Stockwerkswahl werden solange Tasteneingaben entgegengenommen, bis eine Ziffer und das Rautezeichen oder zwei Ziffern registriert wurden. Durch die Sterntaste wird die Eingabe zurückgesetzt. Da das Tastenfeld den Buchstaben E nicht unterstützt und im Allgemeinen E und Null nicht gleichzeitig auf einem Bedienfeld vorkommen, wird dieser durch die Null repräsentiert. Die Tasteneingabe ist ausschließlich während der Phase der Stockwerkswahl aktiv.

Wie in Abschnitt 4.2 dargestellt, kommt für die Tasterdetektion das Deep Learning Verfahren mit einer Laufzeit von ca. 0,75 s pro Eingabebild zum Einsatz. Die Beschriftungslokalisierung mittels Konturauffindung und die Zeichenerkennung mit der Tesseract Bibliothek werden für jeden gefundenen Taster ausgeführt, bis das gewünschte Stockwerk erkannt wurde. Im schlechtesten Fall bedingt dies bspw. für ein Bild mit sechs Tastern eine Ausführungszeit

von ca. $6 \cdot 1,7 \text{ s} = 10,2 \text{ s}$, im Mittel von $3 \cdot 1,7 \text{ s} = 5,1 \text{ s}$ für die Beschriftungslokalisierung und -erkennung. Nach dem Verfahren zur Entfernungsberechnung erfolgt zunächst die Zielverfolgung mittels Merkmalsextraktion und -abgleich. Diese benötigt ca. 2,2 s für das erneute Auffinden des Zieltasters. Für das Beispiel mit sechs Tastern ergibt sich also im Mittel eine Laufzeit von ungefähr 8 s für die Bildverarbeitungsfunktionen. Die erste Bewegung für die Distanzberechnung benötigt mit Ausschwingen ca. 0,4 s. Die Bewegung zum Ziel ist zwar entfernungsabhängig, dauert aber in der Regel nicht länger als 1,6 s, so dass der gesamte Prozess von der Eingabe des Nutzers bis zum Betätigen des Tasters bei dem Beispiel mit sechs Tastern mit einstelliger Beschriftung im Mittel ungefähr 10 s dauert.

Der Erfolg beim Auffinden des gewünschten Fahrstuhltasters hängt hauptsächlich von den Schritten Tasterdetektion und Beschriftungslokalisierung ab, da hier die Fehlerquoten deutlich höher sind als bei den anderen zum Einsatz kommenden Verfahren. Unter der vereinfachten Annahme, dass nur die Trefferquoten bzw. Sensitivitäten der Taster- und Beschriftungsdetektion Einfluss haben, beträgt die Wahrscheinlichkeit, einen vorhandenen Taster inklusive Beschriftung korrekt zu erkennen:

$$P_{\text{Gesamt}} = P_{\text{Taster}} \cdot P_{\text{Beschriftun}} = 0,82 \cdot 0,80 = 0,66^{40}$$

Diese Berechnung basiert auf den weiter oben genannten Testergebnissen, die mit relativ wenigen Testdaten erzielt wurden. Für eine fundierte Aussage ist unter Umständen eine größere Testmenge erforderlich.

Die Berechnung der Entfernung zum Ziel ist in großem Maß abhängig von der Positioniergenauigkeit des verwendeten Roboterarms. Bei einer Abweichung der tatsächlich zurückgelegten Strecken auf den einzelnen Achsen von den angenommenen Werten ist das Ergebnis entsprechend ungenau. Gerade bei kurzen Strecken, die für die Distanzberechnung verwendet werden, ist der Positionierfehler des verwendeten Roboterarms bemerkbar, sodass hier mit

⁴⁰ P_{Taster} steht hier für die Sensitivität der Tasterdetektion (vgl. Tabelle 1), $P_{\text{Beschriftun}}$ für die Sensitivität der Beschriftungslokalisierung (vgl. Tabelle 2).

Abweichungen gerechnet werden muss. Ebenfalls abhängig von der Positioniergenauigkeit des Roboters ist die Betätigung des gewünschten Tasters. Zwar ist die Abweichung auf längeren Strecken geringer, die anzusteuende Position wird aber durch die Entfernungsberechnung ermittelt.

Zur Auswertung bzw. Beurteilung des entwickelten Prototyps dient eine Testreihe mit fünf verschiedenen Modellen von Fahrstuhlbedienfeldern mit insgesamt 26 Tastern. Ausgewählt wurden solche Bilder, die überwiegend von der Tasterdetektion erkannt werden, da hier die Integration der Teilfunktionen beurteilt werden soll. Aufgrund des eingeschränkten Sichtfelds der verwendeten Kamera muss das Modell des Bedienfelds relativ mittig zur Kamerachse platziert werden. Für jedes Bild eines Bedienfeldes sind jeweils zehn Versuche vorgesehen. Im Ergebnis wird zu ca. 65 Prozent der gewünschte Taster betätigt. Im Falle eines Fehlschlags ist dies zu 70 Prozent einer Fehlberechnung der Entfernung geschuldet, die das Ziel fälschlicherweise außer Reichweite vermutet. Bei 23 Prozent der Misserfolge wird der Effektor falsch positioniert und bei 7 Prozent der Taster nicht erkannt. Bei fehlerhafter Distanzermittlung ist häufig schon mit bloßem Auge zu erkennen, dass die Fahrwege für die anschließenden Berechnungen zu kurz ausfallen. Messungen bestätigen diesen Eindruck. Diese Ergebnisse zeigen, dass die Erfolgsquote des Demonstrators bei der Tasterbetätigung hauptsächlich von der Positioniergenauigkeit des verwendeten Roboterarms abhängen. Diese ist maximal zu Demonstrationszwecken geeignet.

5 Fazit und Ausblick

Ziel der vorliegenden Arbeit war die Konzeption eines mechatronischen Assistenzsystems zur automatischen Erkennung und Betätigung von Aufzugstastern. Anhand eines Prototyps sollte untersucht werden, ob die Umsetzung der Zielstellung unter Verwendung von freier Software und günstiger Hardware machbar ist.

Nach der Ableitung von Teilaufgaben aus der Problemstellung wurden basierend auf dem Stand der Technik Lösungswege für diese entworfen. Die Integration der Teillösungen erforderte eine Festlegung des Ablaufs der Hauptsteuerung und der Kommunikationswege und -abläufe zwischen den Komponenten. Da teilweise mehrere alternative Verfahren für die Teilaufgaben entwickelt wurden, erfolgte nach deren Evaluation eine Auswahl für den Prototyp.

Der entwickelte Demonstrator zur Hilfe bei der Fahrstuhlbedienung findet und betätigt nach der Eingabe des gewünschten Stockwerks überwiegend den richtigen Taster, wenn dieser in Reichweite des Roboterarms liegt. Für einen Einsatz in der Praxis besteht allerdings Optimierungspotential bezüglich der Antwortzeit und der Erfolgsquote des entwickelten Systems.

Die Antwortzeit des Prototyps ist im Wesentlichen abhängig von der Zeichenerkennung und der Verfolgung des gewünschten Tasters bei Bewegung des Manipulators. Die verwendete Bibliothek zur Zeichenerkennung bietet zwar eine hohe Erkennungsrate, ist aber durch den großen Umfang an unterstützten Sprachen und die Spezialisierung auf das Auslesen ganzer Textseiten langsam in der Ausführung. Hier ermöglicht ein selbst trainiertes Verfahren mit Einschränkung auf Ziffern und wenige ausgewählte Zeichen wahrscheinlich ein besseres Laufzeitverhalten. Weiterer Untersuchungsbedarf besteht auch bei der Objektverfolgung mittels Merkmalsdetektion und -abgleich. Andere Algorithmen zur Extraktion und Beschreibung von Merkmalen wie bspw. BRISK bieten schnellere Laufzeiten, erzielten allerdings in der Einsatzumgebung des Prototyps keine robusten Ergebnisse.

Für einen erfolgreichen Betätigungsprozess ist zunächst der gewünschte Fahrstuhltaster inklusive Beschriftung zu erkennen, bevor dieser von einem

Effektor angefahren und betätigt wird. Da Zielverfolgung und Zeichenerkennung zuverlässig funktionieren, ist die Erfolgsquote für die Lokalisierung des Ziels im Wesentlichen von den Verfahren zur Detektion von Tastern und deren Beschriftung abhängig. Die beiden hierfür ausgewählten Methoden erzielen zwar für den Demonstrator brauchbare Resultate, für einen Einsatz in der Praxis müssen die Erkennungsraten allerdings weiter verbessert werden. Da die Lokalisierung aller Tasterbeschriftungen im Gesamtbild in dieser Arbeit nicht zuverlässig umgesetzt werden konnte, erfolgte eine Einschränkung auf solche Beschriftungen, die innerhalb von vorher detektierten Tastern liegen. Für zukünftige Arbeiten ist eine Erweiterung für Beschriftungen außerhalb von Fahrstuhl Tastern vorzunehmen. Denkbar ist bspw. eine Kombination mit den Nachbearbeitungsschritten aus [15] oder eine Verbesserung des entworfenen Deep Learning Verfahrens zur Beschriftungslokalisierung. Zur Optimierung der Detektionsmethoden, die auf maschinellen Lernverfahren bzw. Deep Learning basieren, existiert eine Reihe von Verbesserungsmöglichkeiten. Insbesondere hat der Input, also die Qualität sowie die Anzahl der verwendeten Bilder, einen großen Einfluss auf das Training der verschiedenen Verfahren. Kann die Anzahl der zur Verfügung stehenden Positivbilder nicht erhöht werden, bietet sich eine künstliche Vergrößerung des Ausgangsmaterials an (Data Augmentation). Dies kann bspw. durch zufällige Rotation, Verzerrung oder Kontraständerung erfolgen. Weiterhin steht für den Trainingsvorgang eine Vielzahl von Parametern zur Verfügung, durch welche das Ergebnis unter Umständen weiter verbessert werden kann.

Nach erfolgreicher Lokalisierung des gewünschten Tasters erfolgen die Wegberechnung und das Verfahren des Effektors zum Ziel. Eine Sonderstellung nimmt hierbei der verwendete Manipulator ein, da sowohl die Ermittlung der Entfernung zum Ziel als auch das eigentliche Betätigen des Tasters von dessen Positioniergenauigkeit abhängen. Der für den Prototyp verwendete Roboterarm dient lediglich zu Demonstrationszwecken, aufgrund des sehr günstigen Aufbaus und der damit einhergehenden Ungenauigkeit ist dieser für einen Einsatz in der Praxis nicht geeignet. Für ein Produktivsystem ist entweder ein vorhandener Manipulator, wie bspw. der Leichtbauroboterarm des Systems

EDAN, zu nutzen oder es ist ein Aufbau mit qualitativ höherwertigen Komponenten durchzuführen, um eine bessere Positioniergenauigkeit zu erreichen. Letzteres erhöht zwar die Kosten, wie das Projekt Walter⁴¹ zeigt, ist damit aber immer noch ein für Endanwender erschwingliches System realisierbar. Denkbar ist auch die Einführung eines geschlossenen Regelkreises, wofür allerdings u. a. die Geschwindigkeit der Berechnungen zur Objektverfolgung erhöht werden müsste.

Vom Autor wurde nur eine andere Arbeit gefunden, welche sich mit der Bedienung von Fahrstühlen unterschiedlichen Typs befasst. Diese bietet zwar etwas bessere Erkennungsraten, ist allerdings nur bei Bedienfeldern nach den ADA-Richtlinien anwendbar, welche in Deutschland seltener Anwendung finden. Zusätzlich wird ein Laserscanner in den Erkennungsprozess eingebunden. Eine Integration bzw. Kombination der Methoden beider Arbeiten bietet unter Umständen Verbesserungspotential. Ebenso könnte die Einbindung weiterer Sensoren, wie z. B. eines Time-of-Flight-Entfernungssensors, ein Untersuchungsgegenstand künftiger Projekte sein.

Insgesamt wurde ein modulares System entwickelt, in welches neue Funktionen leicht integriert werden können. Für den entwickelten Prototyp wurden ausschließlich freie Software sowie günstige Hardwarekomponenten verwendet. Die geforderten Aufgaben wurden größtenteils erfüllt, einzelne Teilfunktionen wie die Detektion von Tastern und deren Beschriftung bieten Raum für Verbesserungen. Für einen Einsatz in der Praxis könnten zusätzlich Aspekte wie die Sicherheit bei der Mensch-Maschinen-Interaktion oder das Medizinproduktegesetz Gegenstand zukünftiger Arbeiten sein.

⁴¹ vgl. Abschnitt 3.1.2

6 Literaturverzeichnis

- [1] Bundesministerium für Gesundheit, „Abschlussbericht zur Studie: Unterstützung Pflegebedürftiger durch technische Assistenzsysteme,“ VDI/VDE Innovation + Technik GmbH, IEGUS – Institut für Europäische Gesundheits- und Sozialwirtschaft GmbH, Berlin, 2013.
- [2] T. Christaller, M. Decker, J. Gilsbach und G. Hirzinger, Robotik - Perspektiven für menschliches Handeln in der zukünftigen Gesellschaft, Berlin Heidelberg: Springer-Verlag , 2001.
- [3] DIN Deutsches Institut für Normung e.V., *Roboter und Robotikgeräte - Wörterbuch (ISO/DIS 8373:2010); Entwurf November 2010*, Berlin: Beuth Verlag GmbH, 2010.
- [4] VDI, „VDI Richtlinie 2860: Montage- und Handhabungstechnik; Handhabungsfunktionen, Handhabungseinrichtungen; Begriffe, Definitionen, Symbole,“ VDI, Düsseldorf, 1990.
- [5] Maheu et al., „Evaluation of the JACO robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities,“ in *IEEE international conference on rehabilitation robotics (ICORR)*, Zürich, 2011.
- [6] Vogel et al., „An assistive decision-and-control architecture for force-sensitive hand-arm systems driven by human-machine interfaces,“ *The International Journal of Robotics Research (IJRR)* , vol. 34, no. 6, pp. 763-780, Mai 2015.
- [7] C. Martens, O. Prenzel und A. Graese, „The Rehabilitation Robots FRIEND-I & II: Daily Life Independency through Semi-Autonomous Task-Execution,“ in *Rehabilitation Robotics*, Sashi S Kommu, IntechOpen, 2007.
- [8] Universität Bremen - Institute of Automation, „Assistenzroboter FRIEND,“ 16 Januar 2014. [Online]. Available: <http://www.iat.uni-bremen.de/sixcms/detail.php?id=555>. [Zugriff am 18 Mai 2018].
- [9] L. Priese, Computer Vision - Einführung in die Verarbeitung und Analyse digitaler Bilder, Berlin Heidelberg: Springer Vieweg, 2015.
- [10] M. Hassaballah, A. Abdelmgeid und A. Hammam, „Image Features Detection, Description and Matching,“ in *Image Feature Detectors and*

- Descriptors: Foundations and Applications - Studies in Computational Intelligence (Book 630)*, Switzerland, Springer International Publishing, 2016, pp. 11-45.
- [11] H. Süße und E. Rodner, Bildverarbeitung und Objekterkennung - Computer Vision in Industrie und Medizin, Wiesbaden: Springer Vieweg, 2014.
 - [12] P. Viola und M. Jones, „Robust Real-time Object Detection,“ in *SECOND INTERNATIONAL WORKSHOP ON STATISTICAL AND COMPUTATIONAL THEORIES OF VISION – MODELING, LEARNING, COMPUTING, AND SAMPLING*, Vancouver, 2001.
 - [13] A. T., H. A. und P. M., „Face Recognition with Local Binary Patterns,“ in *Computer Vision - ECCV 2004. ECCV 2004. Lecture Notes in Computer Science, vol 3021*, Berlin, Heidelberg, Springer-Verlag, 2004.
 - [14] J. Cruz, E. Shiguemori und L. Guimarães, „A comparison of Haar-like, LBP and HOG approaches to concrete and asphalt runway detection in high resolution imagery,“ *Journal of Computational Interdisciplinary Sciences*, Bd. 6, 2016.
 - [15] E. Klingbeil, B. Carpenter, O. Russakovsky und A. Y. Ng, „Autonomous operation of novel elevators for robot navigation,“ in *2010 IEEE International Conference on Robotics and Automation*, Anchorage, 2010.
 - [16] American with Disability Act, „ADA Compliance Directory - ELEVATORS,“ 2018. [Online]. Available: <http://www.ada-compliance.com/ada-compliance/ada-elevators.html>. [Zugriff am 12 Mai 2018].
 - [17] A. Krizhevsky, I. Sutskever und G. E. Hinton, „ImageNet Classification with Deep Convolutional Neural Networks,“ in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou und K. Q. Weinberger, Hrsg., Curran Associates, Inc., 2012, pp. 1097-1105.
 - [18] Y. Lecun, L. Bottou, Y. Bengio und P. Haffner, „Gradient-based learning applied to document recognition,“ in *Proceedings of the IEEE*, 1998.
 - [19] R. B. Girshick, J. Donahue, T. Darrell und J. Malik, „Rich feature hierarchies for accurate object detection and semantic segmentation,“ *CoRR*, Bd. abs/1311.2524, 2013.

- [20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto und H. Adam, „MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,“ *CoRR*, Bd. abs/1704.04861, 2017.
- [21] J. Hui, „Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3),“ 27 März 2018. [Online]. Available: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359. [Zugriff am 20 April 2018].
- [22] Y. Zhang, H. Peng und P. Hu, „CS341 Final Report: Towards Real-time Detection and Camera Triggering,“ in *Project in Mining Massive Data Sets*, 2017.
- [23] C. R. Kulkarni und A. B. Barbadekar, „Text Detection and Recognition: A Review,“ in *International Research Journal of Engineering and Technology (IRJET) Volume: 04 Issue: 06*, IRJET, 2017, pp. 179-185.
- [24] T. Wang, D. J. Wu, A. Coates und A. Y. Ng, „End-to-end text recognition with convolutional neural networks,“ in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, Tsukuba, 2012.
- [25] F. Chaumette und S. Hutchinson, „Visual servo control, Part I: Basic approaches,“ *IEEE Robotics and Automation Magazine*, Bd. 13, pp. 82-90, 2006.
- [26] G. Palmieri, M. Palpacelli, M. Battistelli und M. Callegari, „A Comparison between Position-Based and Image-Based Dynamic Visual Servoings in the Control of a Translating Parallel Manipulator,“ *Journal of Robotics*, Bd. 2012, pp. 1-11, 2012.
- [27] R. Pieters, *Visual Servo Control - Vorlesung*, Eindhoven: Eindhoven University of Technology, 2012.
- [28] E. Karami, S. Prasad und M. S. Shehata, „Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images,“ *CoRR*, Bd. abs/1710.02726, 2017.
- [29] 1&1 Internet SE, „Vier aktuelle Raspberry-Pi-Alternativen – Einplatinenrechner im Vergleich,“ 2018 Januar 25. [Online]. Available: <https://hosting.1und1.de/digitalguide/server/knowhow/raspberry-pi-alternativen-einplatinenrechner-im-check/>. [Zugriff am 29 Januar 2018].

- [30] J. Hughes, „Raspberry Pi Camera Module,“ 3 März 2017. [Online]. Available: <https://www.raspberrypi.org/documentation/hardware/camera/README.md>. [Zugriff am 6 Januar 2018].
- [31] J. Alt, „Walter,“ 2 März 2017. [Online]. Available: <https://walter.readthedocs.io/en/latest/>. [Zugriff am 20 Februar 2018].
- [32] J. Le, „Robot Arm MK2 Plus,“ 6 September 2017. [Online]. Available: <https://www.thingiverse.com/thing:2520572>. [Zugriff am 5 Januar 2018].
- [33] M. Venhaus, *Modellierung ABB IRB 460*, Iserlohn, 2018.
- [34] ABB Ltd, „Technische Daten IRB 460,“ 2018. [Online]. Available: <https://new.abb.com/products/robotics/de/industrieroboter/irb-460/daten>. [Zugriff am 20 April 2018].
- [35] Y. Freund und R. E. Schapire, *A Short Introduction to Boosting*, 1999.
- [36] Z. Ye, „Viola-Jones Face Detection,“ 1 Dezember 2012. [Online]. Available: <https://sites.google.com/site/5kk73gpu2012/assignment/viola-jones-face-detection>. [Zugriff am 10 03 2018].
- [37] „CS231n: Convolutional Neural Networks for Visual Recognition,“ 2018. [Online]. Available: <http://cs231n.stanford.edu/>. [Zugriff am 21 Februar 2018].
- [38] Wikipedia, „Datei:3D_Convolution_Animation.gif,“ 4 Februar 2013. [Online]. Available: https://de.wikipedia.org/wiki/Datei:3D_Convolution_Animation.gif. [Zugriff am 15 Mai 2018].
- [39] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu und A. C. Berg, „SSD: Single Shot MultiBox Detector,“ in *Computer Vision -- ECCV 2016*, Cham, 2016.
- [40] TensorFlow, „MobileNetV2,“ 2 Mai 2018. [Online]. Available: <https://github.com/tensorflow>. [Zugriff am 5 Juni 2018].
- [41] H. Chen, S. S. Tsai, G. Schroth, D. M. Chen, R. Grzeszczuk und B. Girod, „Robust text detection in natural images with edge-enhanced Maximally Stable Extremal Regions,“ in *2011 18th IEEE International Conference on Image Processing*, 2011.
- [42] B. Epshtein, E. Ofek und Y. Wexler, „Stroke Width Transform,“ 2010.

- [43] S. Suzuki und K. Abe, „Topological structural analysis of digitized binary images by border following.,” *Computer Vision, Graphics, and Image Processing*, Bd. 30, pp. 32-46, September 1985.
- [44] T. de Campos, „The Chars74K dataset,” 15 Oktober 2012. [Online]. Available: <http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/>. [Zugriff am 13 Januar 2018].
- [45] R. Smith, „Tesseract OCR,” 3 April 2018. [Online]. Available: <https://github.com/tesseract-ocr/tesseract>. [Zugriff am 5 April 2018].
- [46] V. Lehtola, H. Huttunen, F. Christophe und T. Mikkonen, „Evaluation of Visual Tracking Algorithms for Embedded Devices,” in *Image Analysis*, Springer International Publishing, 2017, pp. 88-97.
- [47] W. Burger und M. J. Burge, *Digitale Bildverarbeitung*, Springer Berlin Heidelberg, 2015.
- [48] M. A. Fischler und R. C. Bolles, „Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Commun. ACM*, Bd. 24, pp. 381-395, Juni 1981.
- [49] C. Yin, S. Yang, X. Yi, Z. Wang, Y. Wang, B. Zhang und Y. Tang, „Removing dynamic 3D objects from point clouds of a moving RGB-D camera,” in *2015 IEEE International Conference on Information and Automation*, 2015.
- [50] A. Mordvintsev und K. Abid, „Feature Matching + Homography to find Objects,” 2013. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html. [Zugriff am 22 Januar 2018].
- [51] M. Venhaus, „Notizen zur Bestimmung der Objektentfernung bei unbekannter Objektgröße,” Iserlohn, 2018.
- [52] M. Venhaus, „Robotertechnik - Lehrbrief,” Fachhochschule Südwestfalen, Iserlohn, 2014.
- [53] F. Tobler, „robotArm - Software for a 3D Printed Robot Arm,” 9 Juni 2017. [Online]. Available: <https://github.com/ftobler/robotArm/>. [Zugriff am 6 Januar 2018].

- [54] Assistive Innovations, „iARM,“ Assistive Innovations, [Online]. Available: <https://www.assistive-innovations.com/de/roboterarme/iarm-de>. [Zugriff am 18 Mai 2018].

7 Abbildungsverzeichnis

Abbildung 1: Möglicher Aufbau eines Prototyps	2
Abbildung 2: Assistenzsystem FRIEND der Universität Bremen [7]	6
Abbildung 3: Vier Grundtypen zur Merkmalsberechnung nach Viola und Jones mit zwei (A und B), drei (C) und vier (D) Rechtecken in verschiedener Skalierung und Position [12]	9
Abbildung 4: Erkannte Taster (links) und ermitteltes Gitter (rechts) [15]	10
Abbildung 5: Kinematischer Aufbau des Manipulators nach VDI 2861 [33]	22
Abbildung 6: Seitenansicht eines ABB IRB460 [34]	23
Abbildung 7: Aufbau des Prototyps mit Schnittstellen und Verbindungen zwischen den physischen Komponenten	24
Abbildung 8: Kaskade von binären Klassifikatoren nach [12]	26
Abbildung 9: Bildpyramide eines Beispielbildes mit drei Skalierungen [36]	29
Abbildung 10: Ablauf der Tasterdetektion mit Kaskadenklassifikator	30
Abbildung 11: Faltung einer zweidimensionalen Eingangsmatrix mit einem 3x3- Filterkernel [38]	31
Abbildung 12: Architektur von LeNet-5, einem CNN zur Zeichenerkennung [18] ...	32
Abbildung 13: Vorhersagegenauigkeit, benötigte Rechenoperationen und Speicherbedarf (hier als Ausdehnung dargestellt) verschiedener CNN- Architekturen [40]	34
Abbildung 14: Arbeitsschritte des entworfenen Verfahrens zur Zeichendetektion..	37
Abbildung 15: Homographie zur Transformation von Bildpunkten zwischen den Kameraebenen O_L und O_C bei Aufnahmen aus verschiedenen Winkeln [49] ...	41
Abbildung 16: Beispiel für Merkmalsabgleich und perspektivische Transformation [50]	42
Abbildung 17: Projektionsverhältnisse bei Bewegung des Gegenstands G um die Distanz d	43
Abbildung 18: Ermittlung des horizontalen Winkels von der Bildachse zum Ziel...	44
Abbildung 19: Schema zur Berechnung der Objektentfernung aus der Veränderung des Winkels zum Ziel (hier in der xy-Ebene), links ohne Kameradrehung und TCP	45
Abbildung 20: Ansicht der xy- und rz-Ebenen zur Berechnung der inversen Kinematik	48
Abbildung 21: Zyklus der Hauptsteuerung	51

Abbildung 22: Kommunikationsablauf bei erfolgreicher Ansteuerung des Ziels bis zum Betätigen des Endlagenschalters	52
Abbildung 23: Aufbau des Prototyps mit Hauptkomponenten	54
Abbildung 24: Vorverarbeitung, Schräglauferkennung und rotiertes Ausgangsbild mit gefundenem Zeichen	58
Abbildung 25: Zeichen mit zugehörigen Prognosen des Chars74K Modells (oben) und der Tesseract-Bibliothek ohne Padding (unten)	60
Abbildung 26: Beispiele des Verfahrens mit Merkmalsabgleich mittels SURF-Algorithmus (oben Fehlschlag durch Bewegungsunschärfe, unten erfolgreicher Versuch bei verkürzter Distanz zum Ziel)	62
Abbildung 27: Komponenten der Robotersteuerung mit Schnittstellen.....	65
Abbildung 28: Softwarekomponenten der Hauptsteuerung	67

8 Tabellenverzeichnis

Tabelle 1: Ergebnisse der Verfahren zur Tasterdetektion.....	57
Tabelle 2: Ergebnisse der Verfahren zur Beschriftungsdetektion.....	59

Anhang A Nachrichtenaustausch

Absolute Zielpositionen werden mittels dem Text $G0\ X\{xmm\}\ Y\{ymm\}\ Z\{zmm\}$ an die Robotersteuerung übergeben, wobei die geschweiften Klammern durch Millimeterwerte für die jeweilige Achse ersetzt werden. Also z. B. $G0\ X20\ Y100\ Z15$.


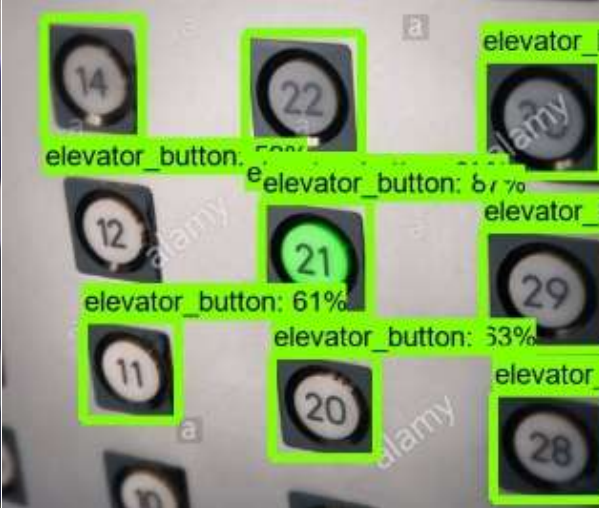




Für relative Bewegungen wird das Format $G1\ X\{xmm\}\ Y\{ymm\}\ Z\{zmm\}$ verwendet. Der Befehl für das Ansteuern der Startposition lautet $H0$. Mit $M17$ werden die Motoren aktiviert, mit $M18$ deaktiviert.

Die Rückgabewerte bestehen aus zwei Buchstaben und sind in folgender Tabelle aufgeführt:




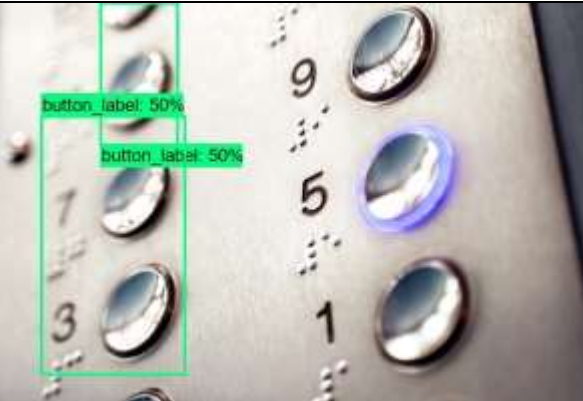


Rückgabewert	Bedeutung
ok	Befehl akzeptiert
or	Ziel außer Reichweite
cf	Kommando ausgeführt
er	Endschalter betätigt
//...	Kommentartext

Anhang B Ergebnisbeispiele

Beispiele der Tasterdetektion

Cascade Classifier (LBP-Merkmale)	CNN
	
	
	

Beispiele der Detektion von Beschriftungen im Gesamtbild

Methode mit Konturauffindung	Deep Learning Verfahren
	
	
	

Anhang C Materialliste

Bezeichnung	Anzahl
Raspberry Pi 3B	1
Raspberry Pi Kameramodul v1	1
Arduino Uno	1
CNC-Shield (Erweiterungsmodul für Arduino Uno)	1
A4988 Motortreiber	3
Schrittmotor Typ Nema 17 No Name	3
Endlagenschalter mechanisch mit Feder	1
Membrantastatur 12 Tasten	1
Raspberry Pi 3 Netzteil schwarz - 2,5A/5V	1
Raspberry Pi Flexkabel 50 cm	1
4-Draht-Kabel für Schrittmotor 50 cm	3
Jumper Kabel (männlich / weiblich)	7
USB-Kabel (A-Stecker auf B-Stecker)	1
Regelbares Netzteil (max. 30V/2,5A)	1
Schraube M4x25	4
Schraube M4x40	4
Schraube M3x20	2
Schraube M3x15	12
Gewindestab M4x80	2
Mutter M4	12
Mutter M3	2