…

Inhalt

[1 Einleitung 1](#_Toc515626847)

[1.1 Ausgangssituation und Problemstellung 1](#_Toc515626848)

[1.2 Abgrenzung der Themenstellung 1](#_Toc515626849)

[1.3 Aufbau der Arbeit 2](#_Toc515626850)

[2 Grundlagen 3](#_Toc515626851)

[2.1 Robotik im Bereich teilautonomer technischer Assistenzsysteme 3](#_Toc515626852)

[2.2 Mustererkennung als Teilgebiet des maschinellen Sehens 6](#_Toc515626853)

[2.3 Visual Servoing 11](#_Toc515626854)

[3 Konzeption 14](#_Toc515626855)

[3.1 Manipulator mit Bildverarbeitungseinheit 14](#_Toc515626856)

[3.1.1 Anforderungen an das System 14](#_Toc515626857)

[3.1.2 Auswahl der Komponenten 17](#_Toc515626858)

[3.1.3 Aufbau des Systems 20](#_Toc515626859)

[3.2 Objekt- und Texterkennung 23](#_Toc515626860)

[3.2.1 Maschinelles Detektionsverfahren mit vorgegebenem Merkmalsraum 23](#_Toc515626861)

[3.2.2 Deep Learning Ansatz zur Detektion ohne Merkmalsraumvorgabe 23](#_Toc515626862)

[3.2.3 Auslesen der Tasterbeschriftung 24](#_Toc515626863)

[3.3 Objektverfolgung 24](#_Toc515626864)

[3.3.1 Echtzeit-Objektverfolgung in Videosequenzen 24](#_Toc515626865)

[3.3.2 Merkmalsabgleich und perspektivische Transformation zwischen Einzelbildern 25](#_Toc515626866)

[3.4 Entfernungsberechnung 25](#_Toc515626867)

[3.4.1 Berechnung durch Änderung des Winkels zum Ziel 25](#_Toc515626868)

[3.4.2 Berechnung durch Änderung der Bildgröße des Ziels 25](#_Toc515626869)

[3.5 Bewegungssteuerung 25](#_Toc515626870)

[3.5.1 Wahl des Koordinatensystems 25](#_Toc515626871)

[3.5.2 Positionierung des Effektors im Raum 25](#_Toc515626872)

[3.6 Integration der Teillösungen 26](#_Toc515626873)

[3.6.1 Benutzerschnittstelle 26](#_Toc515626874)

[3.6.2 Interaktion der Komponenten 26](#_Toc515626875)

[3.6.3 Ablauf des Erkennungs- und Betätigungsprozesses 26](#_Toc515626876)

[4 Prototypische Umsetzung 26](#_Toc515626877)

[4.1 Hardware 27](#_Toc515626878)

[4.1.1 Verwendete Komponenten 27](#_Toc515626879)

[4.1.2 Schnittstellen 27](#_Toc515626880)

[4.1.3 Aufbau des Prototypen und des Tastermodells 27](#_Toc515626881)

[4.2 Software 27](#_Toc515626882)

[4.2.1 Realisierung der Bildverarbeitungsfunktionen 27](#_Toc515626883)

[4.2.2 Firmware zur Bewegungssteuerung 27](#_Toc515626884)

[4.2.3 Schnittstellen 27](#_Toc515626885)

[4.2.4 Programmstruktur 28](#_Toc515626886)

[4.3 Ergebnisse der Prototypenentwicklung 28](#_Toc515626887)

[5 Fazit und Ausblick 28](#_Toc515626888)

[6 Literaturverzeichnis 30](#_Toc515626889)

[7 Abbildungsverzeichnis 35](#_Toc515626890)

[8 Anhang A: Materialliste 37](#_Toc515626891)

# Einleitung

Evtl. ohne unterkapitel

## Ausgangssituation und Problemstellung

Einsatzumgebung

z. B. ein an Rollstuhl befestigter Leichtbauroboter

## Abgrenzung der Themenstellung

…

Grenzen Stand der Technik

Ziele was wie womit

Was soll genau gemacht werden

Einsatzumgebung

Mensch-Maschinen-Interaktion hier nicht

Auch nicht Mobilität (autonome Stromversorgung bzw. so konzipiert, dass Stromverbrauch gering und später autonom möglich)

Begrenzte Energieversorgung

Skizze Aufbau! (oder in 3)

Hier Skizze Roboter, Kamera, Tastermodell???

Arbeitsschritte

Erkennung verschiedenartiger Taster!

## Aufbau der Arbeit

# 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 einen Assistenzroboter befasst, soll zunächst die technologische Ausgangsituation in diesem Bereich ermittelt werden. Anschließend erfolgt eine Einführung in die für die Umsetzung der Arbeit wichtigen Themengebiete Mustererkennung sowie Visual Servoing.

## 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 programmierbare 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 Selbständigkeit 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 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 welchen ein DLR-Leichtbauroboterarm montiert ist. Schwerpunkte des Projekts sind eine auf Elektromyographie (EMG) 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 erreicht. Integrierte Teilautonomie soll Unterstützung bei Aufgaben des Alltagslebens bieten[[1]](#footnote-1). 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 1 zu sehen.



Abbildung 1: 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 (vgl. Smart Tray in Abbildung 1), 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 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 befassen[[2]](#footnote-2). Bezüglich teilautonomer Funktionalität scheinen die hier vorgestellten Systeme EDAN und FRIEND am aktuellsten bzw. weitesten fortgeschritten. Forschungsbedarf ist allerdings weiterhin vorhanden, da bisher Unterstützung nur für wenige spezielle Aufgaben bzw. Tätigkeiten angeboten wird. Weiterhin ist die Verfügbarkeit der genannten Systeme problematisch, da sie entweder für den normalen Nutzer kaum erschwinglich[[3]](#footnote-3) sind oder ausschließlich in Forschungsprojekten zur Verfügung stehen.

## 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](https://de.wikipedia.org/wiki/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. [10]

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 Merkmalen[[4]](#footnote-4) 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. Kategorie mit der höchsten Wahrscheinlichkeit zugeordnet. Ziel der Objektlokalisierung (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 neuro-nale Netze sollen die für die Klassifizierung und Detektion relevanten Merkmale selbständig erlernt werden. Deep Learning Methoden zur Objekterkennung erzielen häufig hohe Erkennungsraten, benötigen hierzu allerdings auch 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 Machine Learning Verfahren 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 2). 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 2: 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 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. Diese Schritte basieren auf der Einschränkung auf solche Taster, die den ADA Richtlinien [15] entsprechen[[5]](#footnote-5). Unter anderem muss sich dafür die Beschriftung immer - zumindest zusätzlich - links außerhalb des Tasters befinden.



Abbildung 3: Erkannte Taster (links) und zugehöriges ermitteltes Gitter (rechts) [16].

Auf der Bedientafel eines Fahrstuhls ergibt sich somit eine Art Gittermuster, welches für die Erkennung genutzt wird (vgl. Abbildung 3). 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 % erreicht, Angaben zur Ausführungsdauer fehlen allerdings [16].

Seit dem ersten Sieg von Krizhesvky 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 Prinzip[[6]](#footnote-6) 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 werden[[7]](#footnote-7). [20]

Nach dem ersten Erfolg des R-CNN-Ansatzes zur Objektlokalisierung 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-Architektur[[8]](#footnote-8). 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 Sekunde[[9]](#footnote-9). 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, welche aus diesen Bereichen Buchstaben, Zahlen oder Wörter extrahiert. Zum Auffinden von Regionen, welche möglicherweise Zeichen enthalten, kann bspw. nach verbundenen Konturen gesucht werden. Durch eine Klassifikation anhand vorgegebener Merkmale werden diese anschließend in Textbereiche oder Nicht-Textbereiche unterteilt. Die darauf folgende Segmentierung trennt Zeichen vom Bildhintergrund[[10]](#footnote-10) 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. Sogenannte Ende-zu-Ende-Verfahren übernehmen alle Schritte der Textdetektion und -erkennung und liefern zunehmend gute Ergebnisse [24].

## 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), wo 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 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 (Bildrate) zur Verfügung, so dass 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. Die Transformation zwischen zwei Bildern kann anhand ausgewählter Merkmalspunkte berechnet werden. Werden die Merkmalspunkte markerbasiert 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 Merkmale im Bild, anschließend versieht sie ein Merkmalsdeskriptor mit zusätzlichen Informationen, wie bspw. Orientierung und Umgebung des Punkts. Brauchbare Merkmalspunkte 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 Merkmalen ist SIFT (Scale-Invariant Feature Transform), welches die oben genannten Kriterien weitgehend erfüllt. Eine effiziente Abwandlung von SIFT ist das SURF-Verfahren (Speeded Up Robust Features), welches 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 Merkmalspunkte die Homographie[[11]](#footnote-11) und damit die Bewegung der Kamera oder des Objektes berechnet werden [27].

# 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.

## Manipulator mit Bildverarbeitungseinheit

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

### 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 einen 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 läuft die Programmlogik (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. Soll 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 Koordinaten[[12]](#footnote-12) entgegen und berechnet daraus Steuerbefehle für die einzelnen Motoren der Achsen. Diese müssen ausreichend Drehmoment für die Bewegung der Glieder sowie für die Tasterbetätigung besitzen, wobei letzteres kaum ins Gewicht fällt.

Da der Effektor keine Arbeitsvorgänge am Zielort ausführt und keine Werkstücke transportiert, 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 bei Betätigung des Tasters zur Normalen des Tasters sollte nicht zu groß werden[[13]](#footnote-13).

Weiterhin wird angenommen, dass sich keine Hindernisse im Arbeitsraum bzw. zwischen der Ausgangsposition des Effektors und dem anzusteuernden 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 eine spätere mobile Einsetzbarkeit zu erleichtern.

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 hingegen sind eine einfache Bedienbarkeit des Systems, die Portierbarkeit der Funktionalität auf andere Systeme, aufgrund des kleinen Zeitrahmens der Arbeit eine einfache Implementierbarkeit und geringe Kosten. Durch die Verwendung von offener Hard- und Software sowie die Verwendung 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.

### 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 A 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 darüber, 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 Bahnplanung mit Interpolation mittels Bézier-Kurven bietet. Der Preis der Komponenten und der Aufwand zum Aufbau sprengen allerdings den Rahmen dieser Arbeit.

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 (momentan über 80) erfolgreich nachgebaut wurde [32][[14]](#footnote-14). 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. Freiheitsgerade 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.

Bei allen erschwinglichen Systemen handelt es sich um Roboter mit Modellbaucharakter und nicht um professionellen Industrieroboter. Daher muss mit einer geringen 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.

### 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 4 zeigt den kinematischen Aufbau des verwendeten Modells nach VDI 2861 sowie ein Schema des Originals, bei dem die Gelenkstangen zu erkennen sind.

Der Effektor besteht aus einer Halterung sowie dem daran fest installierten Endlagenschalter. Die Federung des Schalters ist für den Demonstrator ausreichend, um eine Betätigung eines Tasters zu simulieren.



Abbildung 4: Kinematischer Aufbau des Manipulators nach VDI 2861 (links) und Schema des Originals (rechts)

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 Verfahren 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 5 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 5: Aufbau des Prototyps mit Schnittstellen und Verbindungen zwischen den physischen Komponenten

## 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 Fahrstuhltastern 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.

### 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 [16] beschriebene Verfahren verwendet eine ähnliche Methode, nutzt aber zusätzlich einen Laserscanner für den Erkennungsprozess. Zusätzlich werden Eigenschaften von Fahrstuhltastern nach ADA-Richtlinien vorausgesetzt, welche in Deutschland häufig nicht zutreffen. Daher ist dieses Verfahren in dieser Arbeit nicht anwendbar.

Bei Cascade Classifiern durchläuft jeder Bildausschnitt[[15]](#footnote-15) solange eine Kaskade von binären Klassifikatoren, bis er entweder zurückgewiesen oder der gesuchten Klasse zugeordnet wird (vgl. Abbildung 6). 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 6: 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 ermittelt[[16]](#footnote-16). 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 ein, welches schnelle Laufzeitverhalten und gute Generalisierungseigenschaften verspricht [33]. 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 % der Beispiele richtig klassifizieren. Die Schwellwertfunktion für das Merkmal mit Grenzwert lautet für einen Bildausschnitt :

|  |  |
| --- | --- |
|  | (3.1) |

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

|  |  |
| --- | --- |
|  | (3.2) |

Die Gewichtung der schwachen Klassifikatoren entsteht durch die Fehlerfunktion , welche die Abweichung der Vorhersage über die ebenfalls gewichteten Beispielbilder mit für negative und für positive Bilder summiert:

|  |  |
| --- | --- |
|  | (3.3) |

In jedem Schritt des iterativen Verfahrens wird die Wichtung des Bildes 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 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 Stufe[[17]](#footnote-17) 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 Sensitivität[[18]](#footnote-18) und einer maximal akzeptierten Falsch-Positiv-Rate 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 und eingehalten werden. Für eine Kaskade mit Stufen gilt dann für die Falsch-Positiv-Rate:

|  |  |
| --- | --- |
|  | (3.4) |

Die Sensitivität bzw. Detektionsrate lässt sich ermitteln zu:

|  |  |
| --- | --- |
|  | (3.5) |

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.

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 erforderlich[[19]](#footnote-19), 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.

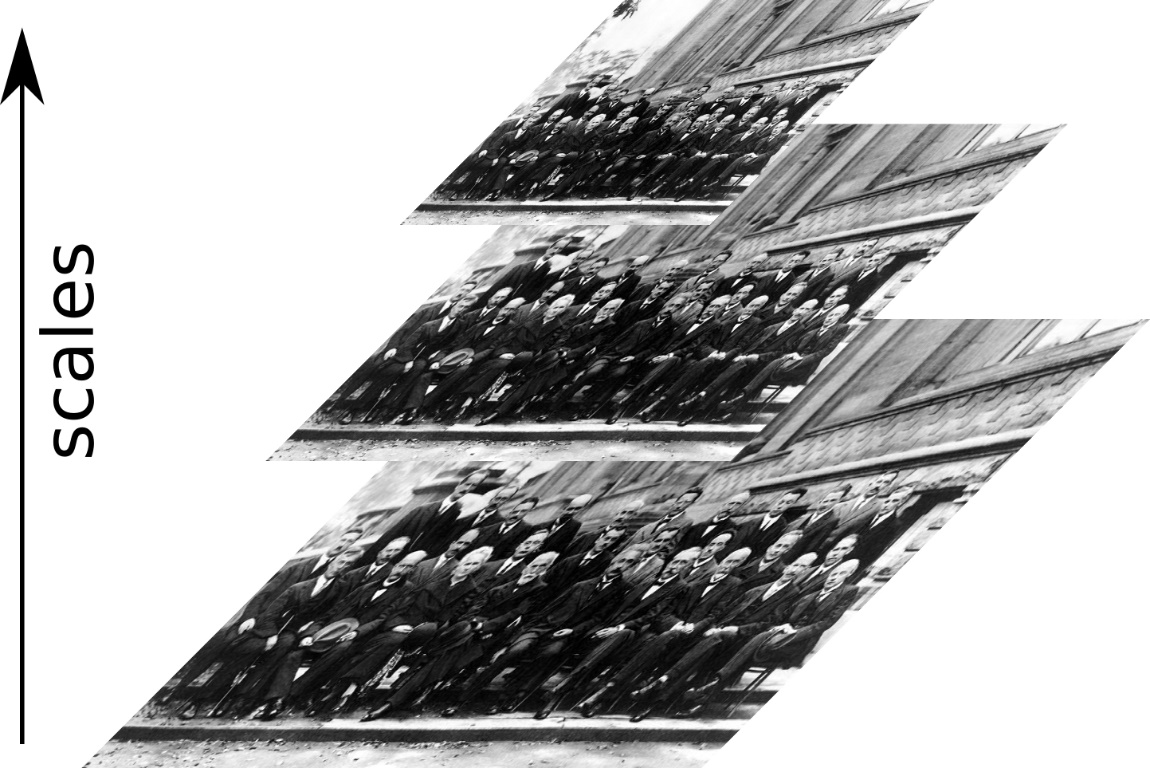


Abbildung 7: Bildpyramide eines Beispielbildes mit drei Skalierungen [34]

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 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 7). 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 8 zeigt den Detektionsprozess von der Erzeugung der Bildausschnitte für den Klassifikator bis zur Ausgabe der Liste mit Positionen und Ausdehnungen der detektierten Objekte.



Abbildung 8: Ablauf der Objektdetektion mit Kaskadenklassifikator

### 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 9 zeigt die Faltungsoperation für den Input des zweiten Neurons, wobei durch sogenanntes Padding das Bild so erweitert wird, das 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) [35].



Abbildung 9: Faltung einer zweidimensionalen Eingangsmatrix mit einem 3x3-Filterkernel [35]

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 simulieren[[20]](#footnote-20).



Abbildung 10: 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 10 ist der Aufbau eines Convolutional Neural Networks beispielhaft dargestellt. Es besteht im Allgemeinen aus mehreren Faltungsschichten, auf welche Reduzierungsschichten (Pooling Layer) folgen. Diese verwerfen 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 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 [37] 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 11 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.

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 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 hier verwendet wird.



Abbildung 11: Vorhersagegenauigkeit, benötigte Rechenoperationen (Multiply-Accumulate) und Speicherbedarf (Ausdehnung) verschiedener CNN-Architekturen [38]

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.

### Auslesen der Tasterbeschriftung

Textdetektion (nicht recognition) als Herausforderung (in the wild)

Alternative zur Detektion: Selbes CNN wie oben mit zwei Klassen trainieren (dann aber wieder viele Trainingsdaten notwendig)

Recognition im Prinzip Klassifikation (jedes Zeichen ein Bild)

Hier zeichenbasiert, da meist Ziffern oder Zahlen

Vorverarbeitung wichtig!

Segmentierung beschreiben

MNIST, char74k etc.

Vorgefertigte Bibliotheken

Alle Ansätze beschreiben

## Objektverfolgung

Ziel / Unterschied bewegter Hintergrund

Hier Teil des Visual Servoing

Nur Translation interessant?

Kamera nicht „gedämpft“! (aber Kamera bewegt)

Laufzeit / mögliche Frequenz untersuchen

### Echtzeit-Objektverfolgung in Videosequenzen

Wäre einfacher (?) bzw. vorzuziehen (warum?)

Normalerweise unbewegter Hintergrund -> zu verfolgende Objekte bewegen sich durch das Bild (Bsp. Fußgänger oder Autos)

Hier aber bewegter Hintergrund, da Kamera bewegt

Bei höherer Frequenz Steuerbefehle schneller anpassbar

### Merkmalsabgleich und perspektivische Transformation zwischen Einzelbildern

## Entfernungsberechnung

2D Bild -> 3D Informationen

### Berechnung durch Änderung des Winkels zum Ziel

### Berechnung durch Änderung der Bildgröße des Ziels

## Bewegungssteuerung

Hier Teil des Visual Servoing

Erst Mechanik, Übersetzung Zahnräder etc.

IK

Denavit… nicht notwendig

PTP ausreichend, keine kartesische Bahnsteuerung notwendig

PTP asynchron oder synchron?

Dann Software

### Wahl des Koordinatensystems

Bezugspunkt?

Bezugssystem! (s. Robotik.pdf)

kartesisch

### Positionierung des Effektors im Raum

Inverse Kinematik…

Bewegungssteuerung -> PTP (?)

## Integration der Teillösungen

Zustandsdiagramm (Zustände der Steuerung) wohin?

### Benutzerschnittstelle

Hier Sequenzdiagramm?

### Interaktion der Komponenten

Kommunikation (Diagramm, Komm.diagr. und/oder Sequenz)

Befehlssatz (oder in 4) (dann im Anhang)

Sequenzdiagramm

Endschalter: Rückmeldung nicht bei jedem Aufzug. Wenn dann unterschiedlich? Z. B. Taste beleuchtet.

### Ablauf des Erkennungs- und Betätigungsprozesses

# Prototypische Umsetzung

Implementierung und Auswertung der Teillösungen

Der zur Steuerung des Manipulators eingesetzte Mikrocontroller ist über die serielle Schnittstelle mit dem Hauptrechner verbunden. Die Verbindung zu dem als Öffner eingesetzten Endlagenschalter erfolgt über zwei digitale Eingangspins. Über das Erweiterungsmodul werden die Motortreiber zur Ansteuerung der Motoren angesprochen.

Nur ein Modell!

Verfahren zur Tasterdetektion

Verwendete Bibliothek(en)

Wie womit implementiert

Auswertung / Gegenüberstellung der Verfahren

Zielverfolgung bei Kamerabewegungen

Firmware zur Bewegungssteuerung des Manipulators

Ansteuerung der Motoren

Kommunikation und Befehlssatz

Erkennung einer Tasterbetätigung

Endlagenschalter

Optional „nachgiebiger Finger“, der erst ab bestimmter Krafteinwirkung auslöst

Implementierung

Firmware Manipulator

Tests etc.

Test des Gesamtsystems

## Hardware

### Verwendete Komponenten

Endlagenschalter

### Schnittstellen

### Aufbau des Prototypen und des Tastermodells

## Software

### Realisierung der Bildverarbeitungsfunktionen

### Firmware zur Bewegungssteuerung

Mit Befehlssatz (oder in nächsten Abschnitt)

Erkennung Tasterdetektion

### Schnittstellen

Benutzer und Systemintern (nach außen und innen)

### Programmstruktur

## Ergebnisse der Prototypenentwicklung

Nachteil Kaskadenklassifikatoren: manuell viele Parameter bestimmen, festlegen für jedes Problem(typ)

# Fazit und Ausblick

# 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] | American with Disability Act, „ADA Compliance Directory - ELEVATORS,“ 2018. [Online]. Available: http://www.ada-compliance.com/ada-compliance/ada-elevators.html. [Zugriff am 12 05 2018]. |
| [16] | 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*, 2010. |
| [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 04 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 2 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] | Y. Freund und R. E. Schapire, *A Short Introduction to Boosting,* 1999. |
| [34] | Assistive Innovations, „iARM,“ Assistive Innovations, [Online]. Available: https://www.assistive-innovations.com/de/roboterarme/iarm-de. [Zugriff am 18 Mai 2018]. |

# Abbildungsverzeichnis

[Abbildung 1: Assistenzsystem FRIEND der Universität Bremen [7] 5](#_Toc515959680)

[Abbildung 2: 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] 8](#_Toc515959681)

[Abbildung 3: Erkannte Taster (links) und zugehöriges ermitteltes Gitter (rechts) [16]. 9](#_Toc515959682)

[Abbildung 4: Kinematischer Aufbau des Manipulators nach VDI 2861 (links) und Schema des Originals (rechts) 21](#_Toc515959683)

[Abbildung 5: Aufbau des Prototyps mit Schnittstellen und Verbindungen zwischen den physischen Komponenten 22](#_Toc515959684)

[Abbildung 6: Kaskade von binären Klassifikatoren nach [12] 24](#_Toc515959685)

# Anhang A: Materialliste

1. Als Beispiel in Schaubildern oder Videos dient die Hilfe beim Trinken. [↑](#footnote-ref-1)
2. Ein Beispiel für ein verfügbares System ist der iARM der Firma Assistive Innovations [34], welcher ähnliche Funktionen wie der JACO bietet. [↑](#footnote-ref-2)
3. Die Preie von JACO und iARM liegen im Bereich von mehreren Zehntausend Euro. [↑](#footnote-ref-3)
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]. [↑](#footnote-ref-4)
5. In Deutschland entspricht nur ein Teil der Fahrstuhltaster diesen Richtlinien. [↑](#footnote-ref-5)
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]. [↑](#footnote-ref-6)
7. Resultat ist eine Reduktion der Detektionszeit auf ca. 50 Sekunden auf einer High-End-CPU des Jahres 2013 [19]. [↑](#footnote-ref-7)
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]. [↑](#footnote-ref-8)
9. Die Detektionsrate liegt im Bereich von 0,1 fps bis 0,7 fps (frames per second) [↑](#footnote-ref-9)
10. Dies kann unter anderem durch Binarisierung erfolgen. [↑](#footnote-ref-10)
11. Zu geometrischen Transformationen und homogenen Koordinaten siehe bspw. [11]. [↑](#footnote-ref-11)
12. Die Form der Koordinaten (bspw. kartesisch) und das Bezugssystem sind noch zu definieren. [↑](#footnote-ref-12)
13. Da an dieser Stelle keine genauen Aussagen getroffen werden können, wird der Grenzwert willkürlich auf 45° festgelegt. [↑](#footnote-ref-13)
14. Von dem hier beschriebenen Modell existieren zwei Varianten, von denen eine Schrittmotoren anstelle von Servomotoren verwendet. Da das verwendete Schrittmotormodell ein höheres Drehmoment und eine höhere Positioniergenauigkeit gegenüber dem eingesetzten Servomotor bietet, wird hier die erste Variante gewählt. [↑](#footnote-ref-14)
15. Die Bildausschnitte entstehen nach dem Sliding-Windows-Prinzip durch Verschieben eines Suchfensters fester Größe über das Eingangsbild (vgl. Abschnitt 2.2). [↑](#footnote-ref-15)
16. Viola und Jones führen zur effizienten Berechnung der Merkmalswerte die Technik des Integralbilds ein, auf die hier aus Zeitgründen nicht näher eingegangen wird [12]. [↑](#footnote-ref-16)
17. aus Formel 3.2 [↑](#footnote-ref-17)
18. Die Sensitivität ist die Richtig-Positiv-Rate . Es gilt . Bei Maximierung der Sensitivität wird also die Falsch-Negativ-Rate minimiert. [↑](#footnote-ref-18)
19. Nach Möglichkeit mehrere hundert Positiv-Beispiele und das fünffache an Negativ-Beispielen. [↑](#footnote-ref-19)
20. Statt der sogenannten Rectified Linear Unit wird aus Gründen der Differenzierbarkeit (siehe Backpropagation weiter unten) meist die Approximation f genutzt. [↑](#footnote-ref-20)