Programmierung eines Industrieroboters für ein Kunden Showcase

1.Projektarbeit

vorgelegt am 1. Januar 2023

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs IMBIT 2021

von

Jon Eckerth

|  |  |
| --- | --- |
| Betreuer in der Ausbildungsstätte: | DHBW Stuttgart: |
| Hewlett Packard Enterprise  Andreas Fuchs CTC Technology and Operations Lead | Prof. Dr. Thomas Kessel |

**Inhaltsverzeichnis**

[Abkürzungsverzeichnis IV](#_Toc123085619)

[Abbildungsverzeichnis V](#_Toc123085620)

[1 Einleitung 1](#_Toc123085621)

[2 Grundlagen 4](#_Toc123085622)

[2.1 Robotik 4](#_Toc123085623)

[2.2 Showcases 6](#_Toc123085624)

[2.3 Computer Vision 7](#_Toc123085625)

[3 Prototyping nach Cole at. al. 8](#_Toc123085626)

[3.1 Problemidentifikation: 8](#_Toc123085627)

[3.2 Intervention: 8](#_Toc123085628)

[3.3 Evaluation: 8](#_Toc123085629)

[3.4 Reflektion und Lernen: 9](#_Toc123085630)

[4 Problem Identifizierung 10](#_Toc123085631)

[5 Intervention 12](#_Toc123085632)

[5.1 ABB Yumi IRB14000 12](#_Toc123085633)

[5.1.1 Routinenprogrammierung mit RAPID 13](#_Toc123085634)

[5.2 Visionsystem 15](#_Toc123085635)

[5.2.1 Kinect 15](#_Toc123085636)

[5.2.2 pylibfreenect2 15](#_Toc123085637)

[5.2.3 Numpy & OpenCV 16](#_Toc123085638)

[5.2.4 Mediapipe Pose 17](#_Toc123085639)

[5.2.5 Gestenerkennungs Logik 17](#_Toc123085640)

[5.3 Kontrollserver 18](#_Toc123085641)

[6 Evaluation 19](#_Toc123085642)

[6.1 Evaluierung der Programmierten Routine 19](#_Toc123085643)

[6.2 Interaktivität und Zuverlässigkeit der Demo 22](#_Toc123085644)

[6.3 Entwickelung der API 25](#_Toc123085645)

[6.4 Kinect 26](#_Toc123085646)

[6.5 Erweiterbarkeit und leichte Wartung 27](#_Toc123085647)

[7 Learning 28](#_Toc123085648)

[Zusammenfassung 28](#_Toc123085649)

[Reflektion 28](#_Toc123085650)

[Fazit und Ausblick: 30](#_Toc123085651)

[Literaturverzeichnis 31](#_Toc123085652)

# Abkürzungsverzeichnis

ROS = Robot Operating System

HPE = Hewlett Packard Enterprise

CTC = Customer Technology Center

ML = Machine Learning

IEEE = Institute of Electrical and Electronics Engineers

ToF = Time-of-Flight

Kinect = Microsoft Kinect V2

Yumi = ABB Yumi IRB14000

# Abbildungsverzeichnis

[Abb. 1: Robotics and mechatronics 2](#_Toc31092940)

[Abb. 2: Kollaborierender Betrieb Arten Gemäß ISO 10218-1.-2 und ISO/TS 15066 4](#_Toc31092940)

[Abb. 3: ABB Yumi](#_Toc31092940) 10

[Abb. 4: Screenshot Robotstudio](#_Toc31092940) 11

[Abb. 5: Erkennen einer Geste während eines Testlaufs im CTC](#_Toc31092940) 13

[Abb. 6: Aufteilung der Orientierungspunkte 14](#_Toc31092940)

[Abb. 7: Punkte in Einflussbereich des Yumis](#_Toc31092940) 16

[Abb. 8: Greifer in Abholposition](#_Toc31092940) 16

[Abb. 9: Greifer in Ablagepostition](#_Toc31092940) 16

[Abb. 10: Powerbank Position A korrekt](#_Toc31092940) 18

[Abb. 11: Powerbank Position A fehlerhaft](#_Toc31092940) 18

[Abb. 12: Framerate mit Kinect](#_Toc31092940) 20

[Abb. 13: FPS der Gestenerkennung](#_Toc31092940) 21

[Abb. 14: Manuele Analyse des Tesvideos](#_Toc31092940) 22

[Abb. 15: Automatische Analyse des Testvideos](#_Toc31092940) 22

Tabellenverzeichnis

[Tab. 1: Fehlerzustaende des ABB Yumi 14](#_Toc31092943)

1. Einleitung

Die intuitive Kollaboration zwischen Mensch und Maschine bietet ein hohes Potential in der Weiterentwicklung der industriellen Produktion. Denn sie ermöglicht gleichzeitig die Nutzung der hohen Flexibilität und strategischen Entscheidungsfähigkeit eines menschlichen Akteurs, ohne dabei auf die Effizienz und Geschwindigkeit eines Roboters verzichten zu müssen. Bisher war es aufgrund der limitierten Wahrnehmungsfähigkeit von maschinellen Systemen und dem daraus resultierenden Unfallpotential nur beschränkt möglich Mensch und Robotik direkt miteinander interagieren zu lassen[[1]](#footnote-2). Durch große Fortschritte in der Echtzeitverarbeitung von Sensordaten, insbesondere im Bereich der Computer-Vision, rückt die gestenbasierte Interaktion zwischen Mensch und Maschine langsam in den Bereich des Möglichen[[2]](#footnote-3). Da der Themenbereich der Robotik auch für die Kunden der HPE interessant ist, wurde im Rahmen des Customer Technologie Center (CTC) eine einfache Kundendemonstration mit einem ABB 1400 YuMi entwickelt. Das Ziel dieser Demonstration ist es, in Form eines semipermanenten Ausstellungsstücks im CTC, bei Kunden einen positiven Eindruck zu hinterlassen und als Konversationsstarter zu fungieren. Zusätzlich ist geplant die Demo mit auf Messen zu nehmen, um die Aufmerksamkeit von potenziellen Bewerbern auf sich zu ziehen.

Die bisherige Demonstration implementierte den Roboter nur auf eine primitive Weise, denn dieser muss manuell gestartet werden und hat kaum Möglichkeiten auf Veränderungen in seinem Umfeld zu reagieren. Darüber hinaus ist die Routine, die er durchläuft, kein geschlossener Zyklus, weshalb nach jedem Durchlauf das Objekt, welches er bewegt wieder per Hand in die Ursprungskonfiguration gebracht werden muss.

Um diese Demonstration zum einen näher an die Thematik der Industrie 4.0 zu bringen und zum anderen als Kundendemonstration interaktiver zu gestalten, wird nun eine Microsoft Kinect V2 angeschafft und die Anforderung gestellt mit dieser eine Weiterentwicklung des Showcase umzusetzen.

Um dies zu erreichen, wird via des Prototyping Prozesses nach Cole et. Al. ein Artefakt erstellt. In dem ersten Schritt der Prototyping Methode, der Problemdefinition, wird das Problem als solches definiert und analysiert. Aus dieser Analyse entspringt auch die zentrale Forschungsfrage, welche die Arbeit leitet: Wie kann ein Showcase basierend auf einer Microsoft Kinect V2 eine Gestenerkennung implementiert, welche automatisch Bewegungsroutinen des Roboters auslösest um die Aufmerksamkeit von Kunden zu erregen? Um diese Frage zu beantworten, wird im Rahmen dieser Arbeit eine Iteration des Prototyping-Prozesses durchgeführt und als Resultat ein Prototyp des Showcases als Artefakt erstellt.

Durch die limitierten Ressourcen in der Praxis ergibt sich, dass sich die Implementierung der Gestenerkennung auf eine einfache Geste beschränken muss. Da die Verarbeitung von komplexen Bewegungen und drei-dimensionalen Bildern aufgrund von mangelnder Rechenkapazität nicht möglich ist, können diese im Rahmen dieser Arbeit nicht betrachtet werden. Im Spezifischen wurde sich hierbei für die Geste des Winkens entschieden, da dies eine simple und leicht zu erkennende Geste mit viel Aussagekraft ist. Die Implementierung der Gestenerkennung ist auf die Microsoft Kinect V2 beschränkt, da diese die einzige Kamerahardware ist, welche von der Abteilung zur Verfügung gestellt wird. Dabei soll die Implementierung in Python erfolgen, da dieses den besten Kompromiss zwischen Leistung und einfacher Programmierung darstellt und die Lernkurve für zukünftige Weiterentwicklungen gering ist. Ausnahme bildet hierbei die Implementierung der Bewegungsroutinen, welche in RAPID oder ROS umgesetzt werden muss, da die ABB Yumi nur mit diesen Sprachen programmiert werden kann. Da die bereits bestehende Demo in RAPID programmiert ist, soll die Implementierung der Bewegungsroutinen auch in dieser Sprache erfolgen.

Nach der Einleitung und der Problemdefinition, wird in Kapitel 2 die theoretische Grundlage der Arbeit beschrieben. Hierbei wird zuerst die Robotik und im spezifischen die Subkategorie der kooperativen Robotik beschrieben. Darauf folgt eine Einführung in das Marketingthema des Showcases und eine Begriffsdefinition dessen. Schließlich werden noch die Grundlagen der Computer Vision beschrieben.

Nachdem die theoretische Grundlage beschrieben wurde, wird in Kapitel 3 die Methode der Arbeit beschrieben. Hierbei wird das Prototyping nach Cole et. Al. erklärt und die einzelnen Schritte der Arbeit erläutert. Ab diesem Punkt orientiert sie die Struktur der Arbeit an den in der Methodik beschrieben Prozessen.

Denn in Kapitel 4 wird die bestehende Problematik welche es durch den Prototypen zu lösen gilt, definiert und analysiert. Die Kriterien, die in diesem Schritt formuliert werden, bilden die Grundlage, sowohl für die Implementierung des Lösungsansatzes als auch für die Evaluation der Arbeit.

Die Implementierung des Lösungsansatzes erfolgt in Kapitel 5 in Form der Intervention. Hierbei wird zuerst die Programmierung des Roboters mithilfe von Robot Studio beschrieben. Darauf folgt die Implementierung des Kinect basierten Computer Vision Modules und dem Algorithmus zur Erkennung des Winkens. Schließlich wird die Integration der beiden Module durch den zentralen Kontrollserver beschrieben.

Die Evaluation der Arbeit erfolgt in Kapitel 6. Hierbei wird die in Kapitel 5 beschriebene Intervention anhand der in Kapitel 4 formulierten Kriterien evaluiert. Zusätzlich werden Lösungsansätze für die Probleme, die sich bei der Implementierung ergeben haben beschrieben, welche in einer zweiten Iteration des Prototyping adressiert werden könnten.

Im Kapitel 7 wird die Arbeit abschließend zusammengefasst und die Ergebnisse der Arbeit diskutiert. Dabei wird auch die Struktur der Arbeit und die Methodik der Arbeit kritisch reflektiert.

1. Grundlagen
   1. Robotik

Da es sich bei der Robotik um ein interdisziplinäres Feld handelt, welches universell einsetzbar ist, gilt es hier den relevanten Bereich einzugrenzen. Dem Institute of Electrical and Electronics Engineers (IEEE) zu folge bilden industrielle Roboter eine eigene Untergruppe der Robotik, da sie sich durch ihr hohes Maß an Spezialisierung und einer starken Abhängigkeit der Umgebung von den anderen Bereichen der Robotik abgrenzen[[3]](#footnote-4). Auf diese im Spezifischen wird sich diese Projektarbeit fokussieren.

So ist zum Beispiel ein selbstfahrendes Auto, welches zur Untergruppe der autonomen Roboter gehört, darauf ausgelegt auf unerwartete Situationen in Echtzeit zu reagieren. Während ein industrieller Roboter klassischerweise darauf ausgelegt ist, dass Variationen seiner Arbeitsumgebung so gering bleiben wie möglich, um ein reproduzierbares Ergebnis zu liefern[[4]](#footnote-5).

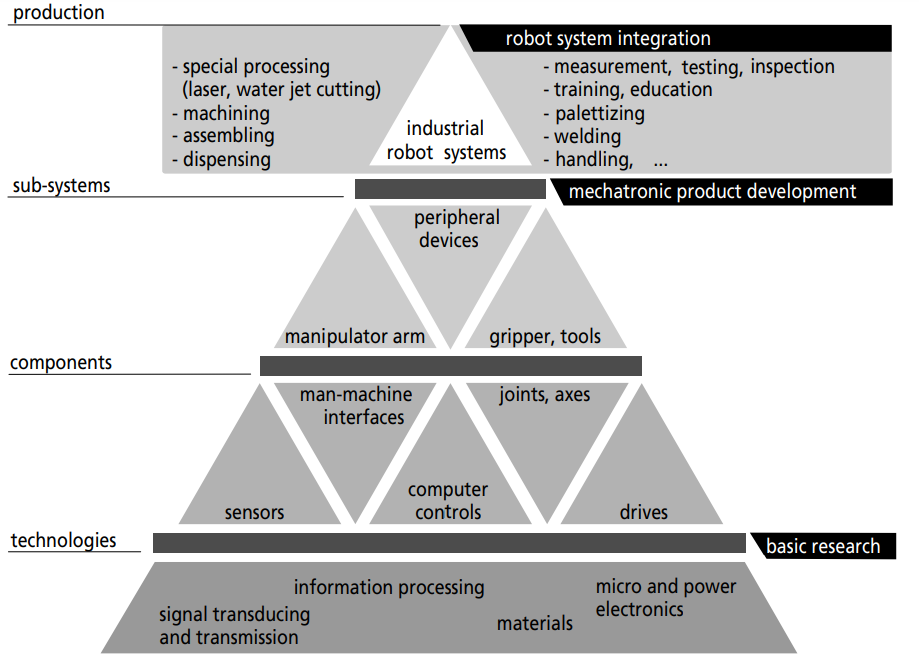


Abb. 1 Robotics and mechatronics[[5]](#footnote-6)

Wie auf der Abbildung (siehe Abb. 1) zu erkennen ist besteht ein Roboter aus drei Subsystemen von Komponenten. Dem Manipulator-Arm, welcher aus Gelenken und Motoren besteht und dafür sorgt, dass das Werkzeugmodul physisch dort platziert wird, wo es seine Arbeit verrichten kann. Die besagten Werkzeugmodule, welche meist anwendungsspezifisch wie etwa eine Sprühdüse für das Auftragen von Lacken oder in Lichtbogenschweißer sind und können bei Änderung der Anforderung schnell für ein anderes ausgetauscht werden.

Koordiniert wird der Roboter oder der Zusammenschluss von mehreren Robotern von einer Kontrolleinheit, welche ihrer Programmierung entsprechend den einzelnen Modulen genau getaktete Aktions-Instruktionen gibt und gleichzeitig Feedback von eingebauten Sensoren entgegennimmt und verarbeitet.

Zusätzlich kann es dann noch Peripherie wie Drehscheiben oder Laufbänder geben, welche nicht den Manipulator zu seinem Ziel bringen, sondern das zu bearbeitende Objekt selbst repositionieren.

**Kooperative Roboter**

Da industrielle Roboter in der Fabrikation für physisch hoch belastende Arbeiten genutzt werden und gleichzeitig auf maximale Produktionseffizienz, bzw. Geschwindigkeit optimiert sind, besteht ein hohes Gefahrenpotential[[6]](#footnote-7), da die kinetische Energie von einem solchen Roboterarm schwere körperliche Schäden verursacht, wenn eine Person in der Gefahrenzone der Maschine erfasst wird. In der Vergangenheit hatten diese Systeme ein sehr beschränktes Verständnis ihrer Umgebung, sog. „no-see no-feel“ Roboter und liefen starr ihre immer gleichen Routinen ab, ohne auf sensorischen Input zu reagieren, weder innerhalb des Fabrikationsprozesses, noch um die Kollision mit Fremdkörpern zu vermeiden[[7]](#footnote-8). Deshalb war die Nutzung dieser Roboter hauptsächlich auf Bereiche mit hohen Stückzahlen von geringer individueller Variation beschränkt und der Zugang von menschlichen Mitarbeitern in den Einflussbereich der Maschinen möglichst zu vermeiden[[8]](#footnote-9). Doch mit dem Aufkommen neuer und besserer Sensortechnik müssen sich Roboter neuerdings nicht mehr nur auf repetitive Arbeiten fernab von menschlichen Arbeitern fokussieren.Zum einen sind sie durch bessere Echtzeit Videoverarbeitung dazu in der Lage dynamischer mit den zu bearbeitenden Objekten interagieren (z.B. Operationen relativ zur Objektorientierung)[[9]](#footnote-10). Zum anderen hat die erweiterte Sensorik es ermöglicht das gefahrenpotential dieser Roboter signifikant zu minimieren. Diese Möglichkeiten, für den kooperativen Betrieb, sind in der ISO-Norm 10218-1[[10]](#footnote-11), -2[[11]](#footnote-12) formalisiert. Diese normiert 4 verschiedene Betriebsmodi für kollaborative Roboter (siehe Abb. 1).

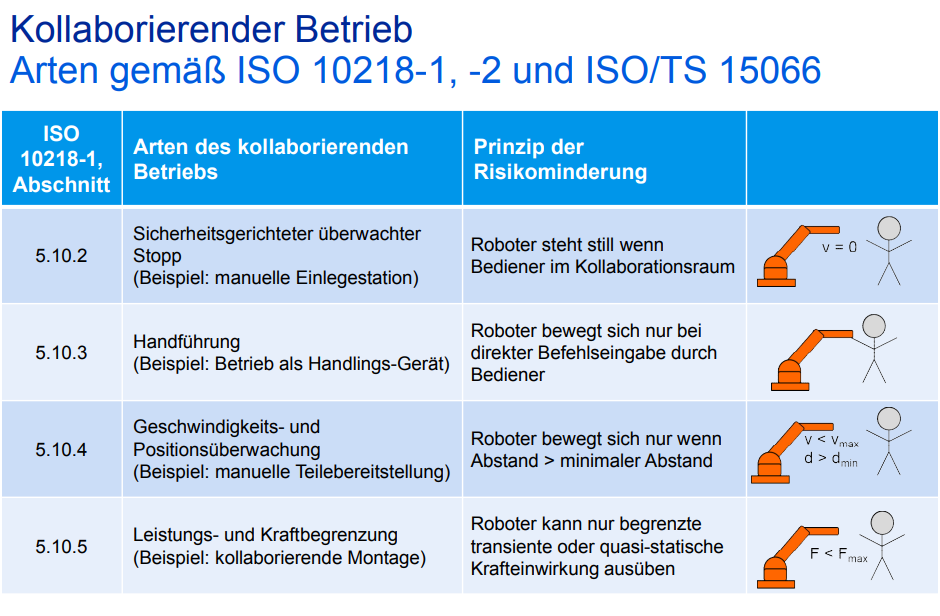


Abb. 2 Kollaborierender Betrieb arten gemäß ISO 10218-1.-2 und ISO/TS 15066[[12]](#footnote-13)

* 1. Showcases

Der Begriff des Showcase ist im Marketing ein sehr weitgefasster Begriff und wird von dem Cambridge Dictionary als Eine Situation oder ein Event dass es ermöglichst die besten Eigenschaften von etwas darzustellen definiert[[13]](#footnote-14). Diese Arbeit fokussiert sich jedoch auf diesen Begriff im Kontext des Erlebnis- und Eventmarketings. Also dem Teil der Marketingtheorie, welcher sich auf das Generieren von Eindrücken und Erlebnissen bei den Interessenten konzentriert. Denn sowohl bei Ständen auf einer Messe als auch bei der Führung von Geschäftspartnern durch den eigenen Ausstellungsraum geht es im Kern nicht nur um die Vermittlung von Fakten über die eigenen Produkte, sondern vor allem darum einen langfristigen positiven Eindruck der Produkte und der Marke beim Publikum zu hinterlassen[[14]](#footnote-15). Ein wichtiger Teil davon ist die Partizipation der Interessenten, denn eine Demonstration bei der Kunden teilhaben können und nicht nur passive Betrachter des Geschehens sind, hinterlässt einen tieferen Eindruck[[15]](#footnote-16). Ein noch tieferer Eindruck kann hinterlassen werden, wenn bei den Interessenten die Erfahrungen mit Emotionen gekoppelt werden können, Beispielseise wenn ein Event Begeisterung hervorruft oder eine besonders anregende Gruppendynamik im Publikum entsteht und damit ein Gefühl der Zugehörigkeit oder des persönlichen Interesses mit der Marke verbunden wird[[16]](#footnote-17).

Dabei dienen Exponate als zentraler Punkt der Interaktion., Sie funktionieren als Publikumsmagnet und bieten dabei die Möglichkeit zur Interaktion mit dem Projekt und dem Personal. So kann ein Exponat als ein Anreiz für einen Kundenshowcase dienen. Denn wenn eine Person Interesse an dem Exponat als solches bekundet hat, kann der Präsentator darauf eingehen und anhand dessen den Nutzen, die Geschichte und die Vorteile des darunter liegenden Produktes darstellen, in dem er die spezifische Implementierung des Exponates erklärt.

* 1. Computer Vision

Computervision beschreibt eine algorithmische Erfassung, Analyse, Verarbeitung und Interpretation von visuellen Daten[[17]](#footnote-18). In dieser Arbeit wird der Begriff der Computer Vision vor allem im Kontext der Objekterkennung verwendet werden. Also dem Bereich der Computer Vision, welche versucht anhand von Fotos oder Videos automatisch ein abstraktes Wissen über die reale Welt zu generieren, in dem sie physische Objekte anhand von zweidimensionalen Bildern zu identifizieren versucht.

In der Objekt Detektion gibt es essenziell zwei zentrale Paradigmen, zum einen die klassische „Featue based“ Objekt Detektion und die Maschine Learning Computer Vision. Die Erstere befasst sich mit manuell programmierten Heuristiken welche nach spezifischen Ansammlungen von gewissen „Features“ wie Graden oder Kurven in Bildern Objekte erkennt. Die zweite Methode basiert auf dem Konzept des maschinellen Lernens, also automatischem Trainieren von neuronalen Netzten[[18]](#footnote-19). Diese neuronale Netzte sind an der Struktur von Gehirnen angelehnt und bestehen aus einer Vielzahl an miteinander verbundenen Knotenpunkten. Im Gegensatz zu der Feature basierten Objekterkennung werden die Erkennungsalgorithmen jedoch nicht manuell programmiert, sondern durch das neuronale Netz anhand von Trainingsdaten an die Problemstellung angelernt.[[19]](#footnote-20) Die Struktur der Neuronalen Netzwerke ermöglich es ihnen komplexe Zusammenhänge zu abstrahieren und daraus Schlüsse zu ziehen.[[20]](#footnote-21)

1. Prototyping nach Cole at. al.

Die in dieser Arbeit von Cole et al. (2005) konzeptionierten Methode des Prototyping, welche versucht die Elemente des Design-Science-Ansatzes mit der Aktionsforschung zu verbinden. Ziel dabei ist es nicht nur, einen Prototypen zu entwickeln, welcher ein gegebenes Problem den Anforderungen entsprechend löst, sondern auch durch eine detaillierte Analyse des entwickelten Prototyps und dessen Einflüsse auf seine Umwelt ein Verständnis über das System als solches zu erlangen. Die Methode nach Cole et al. (2005) ist grundsätzlich ein iterativer Prozess, der solange durlaufen werden kann, bis man sich der idealen Lösung adequat angenähert hat. Dieser setzt sich aus vier zentralen Schritten zusammen:

* 1. Problemidentifikation:

 In diesem Schritt gilt es das Problem als solches zu identifizieren und zu verstehen, welche Auswirkungen eine Lösung desselben auf das entsprechende Environment haben wird. Zudem ist es wichtig, die Interessen der beteiligten Personen zu verstehen und zu berücksichtigen, um eine Lösung zu finden, die auf praktischer Ebene einen Kompromiss zwischen den Interessen der beteiligten Personen darstellt.

* 1. Intervention:

 In diesem Schritt wird der Prototyp entwickelt und in Form einer Intervention in die entsprechende Umgebung eingesetzt. Dabei ist es wichtig die Lösung des Problems zu berücksichtigen und die Auswirkungen auf das Umfeld zu verstehen. Ziel ist es, eine Lösung zu finden, welche die Anforderungen des Problems erfüllt und eine zielgerichtete Veränderung im Umfeld bewirkt.

* 1. Evaluation:

n diesem Schritt wird die Lösung des Problems und die Veränderung im Umfeld evaluiert. Dabei ist es wichtig zu prüfen ob die Lösung des Problems und die Veränderung im Umfeld die gestellten Ziele erfüllen. Zudem ist es wichtig, die Lösung des Problems und die Veränderung im Umfeld zu verstehen und zu analysieren. Zusätzlich werden Schritte formuliert, wie in einer nächsten Iteration die Lösung des Problems und die Veränderung im Umfeld verbessert werden kann und auf welche Weise die Lösung des Problems und die Veränderung im Umfeld in die Umgebung integriert werden kann.

* 1. Reflektion und Lernen:

Dies ist einer der wichtigsten Schritte, da diese der Punkt ist, an welchem neues wissen, sowohl theoretisch als auch praktisch, generiert wird . Dabei werden die Ergebnisse der Evaluation genutzt um ein besser Verständnis des Problems und dessen Umwelt zu erlangen aber auch um abstraktes Wissen über den Lösungsansatz und -prozess zu generieren. Zudem wird das Verständnis der beteiligten Personen über das Problem und dessen Umwelt verbessert und die Lösung des Problems und die Veränderung im Umfeld in die Umgebung integriert.

Als Methodik ist das Prototyping unter dem konstruktivistischen Ansatz zu subsumieren, da die Methode darauf abzielt, die Gegebenheiten der Realität zu verstehen und zu verändern[[21]](#footnote-22). Jedoch bietet sie das Potential, nicht nur qualitative Erkenntnisse zu dem spezifischen und ähnlichen Problemen zu generieren, sondern kann durch die grundlegend iterative Natur auch einen empirischen Ansatz verfolgen.

1. Problem Identifizierung

Die konkrete Problematik, welche im Kontext des CTC der HPE existiert, besteht im Kern aus drei verschiedenen Schichten.

Zum ersten besteht der Bedarf an einer Kundendemonstration, welche auf Messen und als permanentes Exponat im Ausstellungsraum Aufmerksamkeit auf sich ziehen soll und bei potenziellen Kunden oder Bewerbenden ein Interesse für die HPE generieren soll. Aus diesem Grund wurde eine Kundendemonstration entwickelt, welche den ABB Yumi IRB14000 verwendet, um eine HPE gebrandete Powerbank von einer Hand in die Nächste und wieder zurück an den Ursprungsort zu legen. Dabei soll die gesamte Bewegungsfreiheit des Roboters demonstriert werden. Da diese vorprogrammierte Routine bisher jedes Mal manuell gestartet werden muss und die Powerbank per Hand wieder an ihren Ursprungsort zurückgelegt werden muss, besteht der Bedarf einer automatisierten Lösung.

Daraus resultiert die zweite Stufe des Problems. Der Roboter soll nicht ununterbrochen laufen, da dies in seiner Funktion als permanentes Exponat unnötige Strom- und Wartungskosten generieren würde. Andererseits soll diese Routine automatisch ausgeführt werden sollte sich ein potenzieller Interessent in der Nähe befindet. Um dies zu bewerkstelligen, wurde für die Weiterentwicklung der Kundendemonstration ein Microsoft Kinect V2 Sensorsystem zur Verfügung gestellt welches nun den Kern der Problematik ausmacht: Wie ist, basierend auf einer Microsoft Kinect V2, ein Prototyp aufzubauen, welcher in der Lage ist bei Anwesenheit eines Interessenten eine Bewegungsroutine des ABB Yumi IRB14000 auszulösen.

Aus jeder dieser Schichten resultieren unterschiedliche Anforderungen an den Prototypen. Eine gute Kundendemonstration hat die Eigenschaft, die Aufmerksamkeit der Zielgruppe anzuziehen und daraufhin einen längerfristigen Eindruck zu hinterlassen. Eine Möglichkeit diese Anforderung zu erfüllen ist die Initiierung einer sozialen Interaktion durch den Roboter[[22]](#footnote-23).

Aus der Problematik der Automatisierung entsteht andererseits der Bedarf für eine technische Lösung welche zuverlässig dazu in der Lage ist nicht nur die Anwesenheit eines potenziellen Interessenten wahrzunehmen, sondern auch direkt mit dem Roboter kommunizieren können, ohne das manuelle Intervention nötig ist. Zusätzlich besteht bei vollautomatischen unbeaufsichtigten Robotern ein Unfallrisiko, welches es zu minimieren gilt. Diese Probleme legen die Implementierung eines Computer-Vision Systems nahe, dies geht auch mit der Anforderung zur Verwendung einer Kinect einher, da diese für diesen Zweck entwickelt wurde. Jedoch hat auch die erfolgreiche Implementierung eines Computer Vision Systems seine eigenen Problemstellungen, da es in erster Linie schnell und zuverlässig funktionieren muss.

Zusätzlich muss das Gesamtsystem leicht zu warten und weiterentwickeln sein, da voraussichtlich zukünftige Studentenp Projekte darauf aufbauen werden.  Weshalb bei der Auswahl der einzusetzenden Softwarebausteine auch Wert auf leichte Bedienung und gute Dokumentaiton gelegt werden sollte, um die Einarbeitungszeit für zukünftige Studenten zu minimieren.

1. Intervention

Die Problematik, dass jeder Durchlauf der Bewegungsroutine des Roboters einen manuelle Intervention benötig, soll nun auf zwei Arten adressiert werden. Zum einen muss die bereits bestehende Routine so zu Ende programmiert werden, dass sie einen geschlossenen Zyklus bildet. Also der Endzustand und der Anfangszustand der Routine identisch sind, so dass die Routine theoretisch beliebig oft ausgeführt werden kann, ohne dass ein manueller Eingriff in den Ablauf des Roboters nötig ist. Zusätzlich wird eine API umgesetzt, welche externen System die Möglichkeit gibt den Roboter zu starten, um das Betreuungspersonal ein Computer System zu ersetzen.

Alternativ hätte er Roboter auch in einen Dauer-Durchlauf-Nodus versetzen werden können, aber um die damit verbundenen Nachteile zu minimieren wird, nun anhand einer Microsoft Kinect V2 (Kinect) eine Möglichkeit entwickelt die Anwesenheit von Interessenten zu erkennen. Ursprünglich war die Idee nur eine Personenerkennung zu implementieren, womit die Kinect eine ähnliche Funktion wie einen Bewegungsmelder erfüllt hätte.

Um aber gleichzeitig auch die Anforderung eines ansprechenden Exponates zu erfüllen wurde sich dafür entschieden den Ansatz weiterzuentwickeln, dass das System dazu in der Lage ist, das Winken einer Person zu erkennen und dementsprechend mit einer Reaktion des Roboters zu antworten. Darüber hinaus wird auf einem Monitor neben dem Roboter ein Livestream der verarbeiteten Aufnahmen der Kinect gezeigt, um dem Anwender Feedback zu geben.

Damit besteht die Anwendung aus drei Hauptteilen, einmal dem Roboter mit seiner Steuereinheit, dem Sensoriksystem welches die Kinect, den Monitor und einen kleinen Computer beinhaltet und dem Kontrollserver, welcher die Kommunikation zwischen beiden Systemen ermoeglicht und als zentraler Wartungspunkt dient.

* 1. ABB Yumi IRB14000

Bei dem ABB Yumi IRB14000 (Yumi) handelt es sich um einen industriellen Roboter, der spezifisch für die Kollaboration mit Menschen entwickelt wurde.[[23]](#footnote-24) Er ist mit zwei Armen ausgestattet, welche jeweils in sieben unabhängige Aktuatoren unterteilt sind, die wiederum frei rotiert werden können. Da es sich bei dem Yumi um einen kollaborativen Roboter handelt, braucht er keinen separaten Bereich, sondern ist von Mitarbeitern und Passanten frei zugänglich, ohne ein Risiko dazustellen[[24]](#footnote-25). Sollte sich eine Person im Weg des Roboters befinden würde er dies beim ersten Kontakt erkennen und sich in selbst in den Not-Aus Modus versetzen.[[25]](#footnote-26)

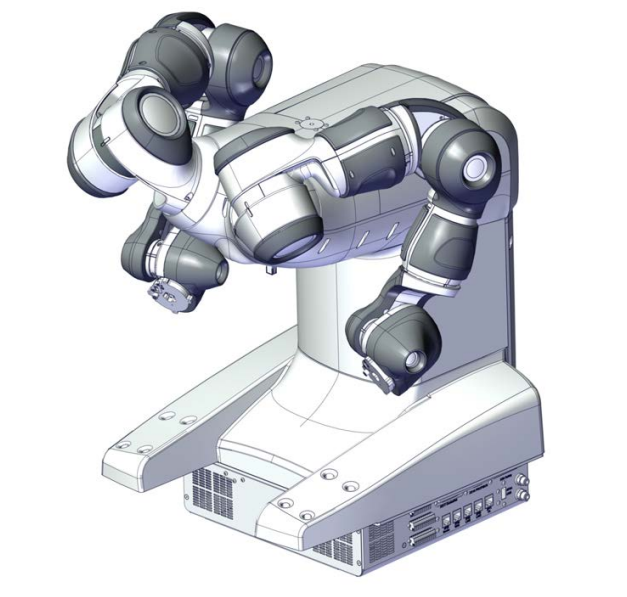


Abb. 3 ABB Yumi[[26]](#footnote-27)

* + 1. Routinenprogrammierung mit RAPID

Bei der Programmiersprache RAPID handelt es sich um eine proprietäre high-level Programmiersprache von ABB, welche spezifisch für das Steuern von Robotik Systemen entwickelt wurde.

Strukturiert ist der Code dabei für jeden Arm separat in so genannten Tasks, welche jeweils ihre eigene Mainmethode haben und bei Aktivierung immer parallel ausgeführt werden. Das essenzielle Elemente bei der Programmierung ist das Objekt mit dem Namen *robtarget*, welches den Zustand eines Armes genau beschreibt. Das Objekt definiert genau den Winkel von jedes Aktuators und die Position des Werkzeugmodules im Koordinatensystem relativ zum Roboter selbst. Dementsprechend gibt es die *Move* Funktion, welche dem Arm die Instruktion gibt sich in besagte Position zu bewegen. Welche Bewegungen der Aktuatoren tatsächlich durchführen muss, um diese Position zu erreichen wird von dem integrierten Controller automatisch durch inverse Kinematik berechnet. Ebenso wichtig sind die Synchronisationsfunktionen, welche dafür sorgen, dass beide Arme in Einklang miteinander agieren. Denn die Synchronisationsfunktionen ermöglichen die Kommunikation zwischen den separat ablaufenden Routinen der Arme. So kann ein Arm die Exekution seiner Routine so lange pausieren, bis er von dem anderen das Signal zum Weitermachen erhält. Prozesse, die sehr von der zeitlichen Koordinierung beider Arme abhängen, wie etwa die Übergabe eines Objektes von der einen in die nächste Hand, werden dadurch leichter zu entwickeln.

Die Programmierung, fand auf zwei unterschiedliche Methoden statt, denn zum einen kann der integrierten Controller des YuMis direkt mittels eines eingebauten Eingabegeräts programmiert werden, Zum anderen stellt ABB aber auch eine Entwicklungsplattform namens RobotStudio zur Verfügung[[27]](#footnote-28). Im Gegensatz zu der Programmierung mittels des FlexPendants also dem Eingabegerät, programmiert Robot Studio nicht den Roboter selbst, sondern eine virtuelle Version der Kontrolleinheit welche eine simulierte Version des Roboters steuert[[28]](#footnote-29) (siehe Abb 4 ).

Diese virtualisierte Version des Kontrollers kann dann bei Bedarf mit einem realen Kontroller synchronisiert werden. Dabei wird der gesamte Code eines Kontrollers mit dem des Anderen spielsweise von einem echten Controller auf Robot Studio übertragen, angepasst und wieder auf den echten Controller gesendet werden.

Der Vorteil bei RobotStudio ist dabei die deutlich erleichterte Manipulation des Programmcodes anhand des Integrierten Softwareentwickelung-Toolset, welches für RAPID konzipiert ist.

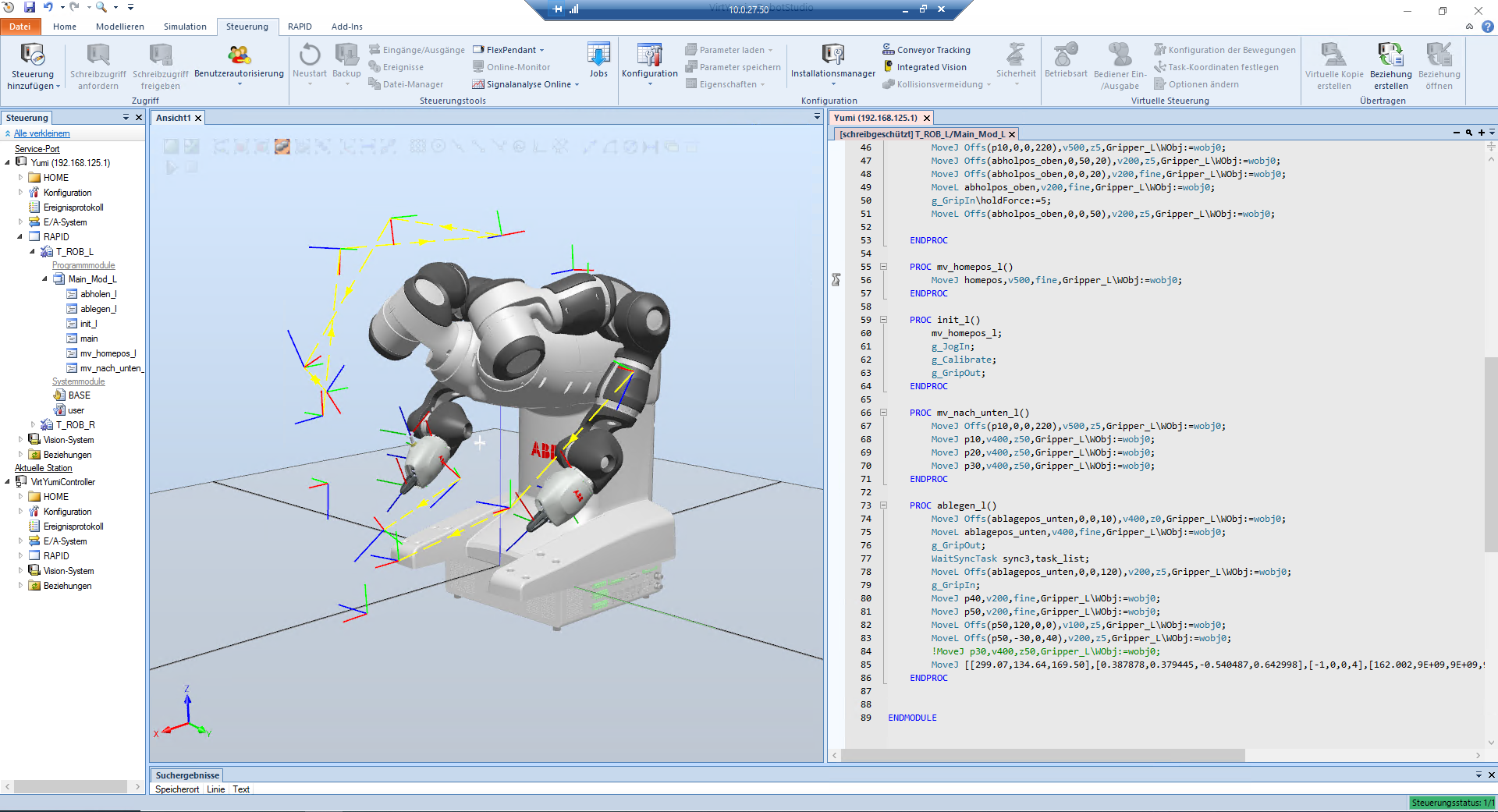
`

Abb. 4 Screenshot Robotstudio

Technisch gesehen ist das Flex Pendant in seinen Funktionen zwar genau so fähig wie Robot Studio, doch ist durch den Touchscreen als primäre Eingabe stark in seiner Anwendbarkeit zum Programmieren begrenzt, jedoch ermöglicht sie den Roboter in einen Programmiermodus zu versetzen. Dabei werden die Gelenke des physischen Roboters frei bewegbar[[29]](#footnote-30). Wenn der Roboter dann so bewegt wurde, dass er sich an der wichtigen Stelle befindet, kann diese Position als robtarget abspeichert und weiterverwendet werden.[[30]](#footnote-31)

* 1. Visionsystem

Zur Erhebung der Daten wurde das Kamerasystem “Kinect V2“ mit Hilfe des proprietären USB Adapters mit einem NUC8i7HN Computer verbunden. Dieser wurde dann mit der neusten Version von Ubuntu und den für die Hardware nötigen Treibern aufgesetzt dann mit dem Netzwerk CTC und dem Kontrollserver verbunden.

* + 1. Kinect

Bei der Kinect handelt es sich um ein RGB-D Kamerasystem von Microsoft, dass ursprünglich für die Spielekonsole Xbox entwickelt wurde, um anstelle eines Handheld-Controllers Nutzereingaben via Motion Capture zu registrieren und diese in Spiele zu integrieren. Dafür verbindet sie eine 1080p Farbkamera mit einem aktiven Infrarot Time-of-Flight (ToF) Sensor und erstellt damit ein Umgebungsbild, welches neben den Rot Grün Blau und Infrarot Farbwerten auch noch die Distanz zum Sensor selbst enthält[[31]](#footnote-32). Diese Werte können dann von Computersystemen, wie der Xbox oder einem PC entgegengenommen werden und anhand von verschiedenen Mustererkennungsalgorithmen analysiert werden. In diesem Anwendungfall wurde die Kinect via des proprietären Adapterstecker von Microsoft and besagten NUC8i7HN angeschlossen auf welchem die im Folgenden beschriebene Datenverarbeitung stattfindet.

* + 1. pylibfreenect2

Die direkte Kommunikation mit der Kinect fand mittels des OpenSource Treiber Libfreenect2 und dem entsprechenden Wrapper für die Programmiersprache Python, namens pylibfreenect2[[32]](#footnote-33). Es ermöglicht direkten Zugriff auf die Videostreams der Kamera und stellt diese anderen Softwaremodulen zur Verfügung. Als Alternative wurde hier auch das Treiber- und Software packet des Herstellers[[33]](#footnote-34), namens „Kinect for Windows SDK 2.0“ betrachtet, doch da dieses offiziell nicht mehr weiterentwickelt[[34]](#footnote-35) wird und ausschließlich mit Windows verwendbar ist, besteht die Befürchtung, dass die aktuelle Kompatibilität mit einem zukünftigen Windowsupdate verloren gehen könnte[[35]](#footnote-36). Im Gegensatz dazu hat die Libfreenect2 Bibliothek den Vorteil, dass diese mit allen gängigen Betriebssystemen kompatibel ist, sprich Linux, Mac und Windows und zusätzlich durch ihre Open-Source Lizensierung jederzeit auf neue Gegebenheiten angepasst werden kann, sollte ein Kompatibilitätsproblem entstehen. Obwohl der Funktionsumfang der „Kinect for Windows SDK 2.0“ beachtlich größer ist, wurde deshalb die Libfreenect2 Bibliothek als Mittel der Wahl ausgewählt und die benötigten Features separat implementiert.

* + 1. Numpy & OpenCV

Bei Numpy handelt es sich um eine Open-Source Bibliothek für Python welche u.a. die Manipulation von multidimensionalen Matrizen erleichtern soll. Diese wird benutz um die Bilder welche der Libfreenect2 Treiber von der Kinect als Frame-datentyp bekommt in die Form einer leicht zu manipulierenden dreidimensionalen Matrix zu übersetzten. Diese Matrix enthält dann für jeden der 1920x1080 Pixel jeweils 3 nicht vorzeichenbehaftete Ganzzahl, einen für Rot einen für Blau einen für Grün. Die weiterer Verarbeitung findet via OpenCV und MediaPipe statt.

OpenCV ist eine Open-Source Bibliothek für Python welche sich auf die Echtzeitverarbeitung von Bildern spezialisiert hat[[36]](#footnote-37). In dieser Implementierung wird sie dazu genutzt Informationen, wie das aus Mediapipe gewonnenen Skelett[[37]](#footnote-38), die berechneten Winkel der Arme und die Meldung wenn ein winken erkannt wurde auf das Ausgabebild zu projizieren, zu Rendern und als eigenes Fenster oder als Video stream dazustellen[[38]](#footnote-39). Dieser Videostream wird gleichzeitig auch auf einem Monitor als Teil des Exponates gezeigt (siehe Abb. 5).

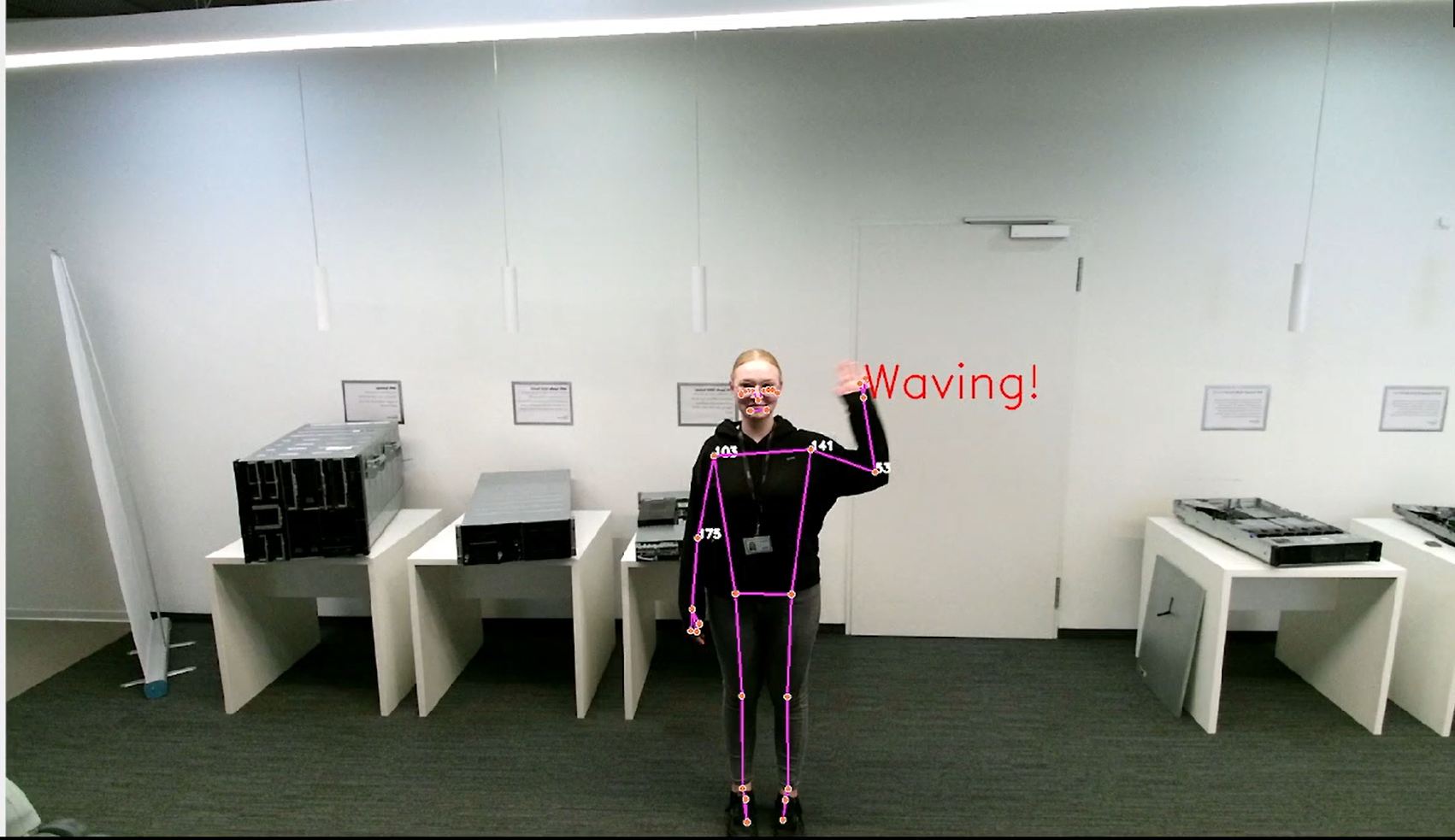


Abb. 5 Erkennen einer Geste während eines Testlaufs im CTC

* + 1. Mediapipe Pose

Bei Mediapipe handelt es sich um eine Sammlung an Bibliotheken welche von Google entwickelt wurde, um das verarbeiten von Sensordaten via ML Algorithmen zu erleichtern[[39]](#footnote-40), im spezifischen wurde hier die „Pose solution“ verwendet, welche darauf trainiert wurde Menschen innerhalb eines Bildes zu erkennen und die die Position gewisser orientierungspunkte der Gliedmaßen relativ zum Bild zu lokalisieren[[40]](#footnote-41) .(siehe Abb 6)

Abb. 6 Aufteilung der Orientierungspunkte[[41]](#footnote-42)

* + 1. Gestenerkennungs Logik

Die Erkennung von spezifischen Gesten anhand eines Skelettes könnte mit maschinellem Lernen umgesetzt werden, Jedoch besteht bei dem Usecase einer Kundendemonstration der Bedarf an der zusätzlichen Genauigkeit, welche möglicherweise durch ML erreicht werden würde, da die Konsequenzen einer erhöhten Fehlerrate bei einer Kundendemonstration relativ gering wäre und die Kosten, die Komplexität und den Aufwand einer ML-Lösung vorerst nicht legitimieren. Deshalb wurde die Gestenerkennung konventionell implementiert und basiert auf folgender Approximation einer Wink-Geste: Eine Person wird als winkend klassifiziert, wenn die Variation des Winkels zwischen Oberarm (Strecke zwischen Punkt 11 und 13 siehe Abb. 6) und Unterarm (Strecke zwischen Punkt 13 und 15 siehe Abb. 6) innerhalb eines bestimmten Zeitabschnittes einen gewissen Grenzwert übertritt, unter der Voraussetzung, dass zwischen dem winkenden Oberarm und der gegenüberliegenden Schulter (Strecke zwischen 11 und 12 siehe Abb. 6) ein gewisser Winkel eingehalten wird.

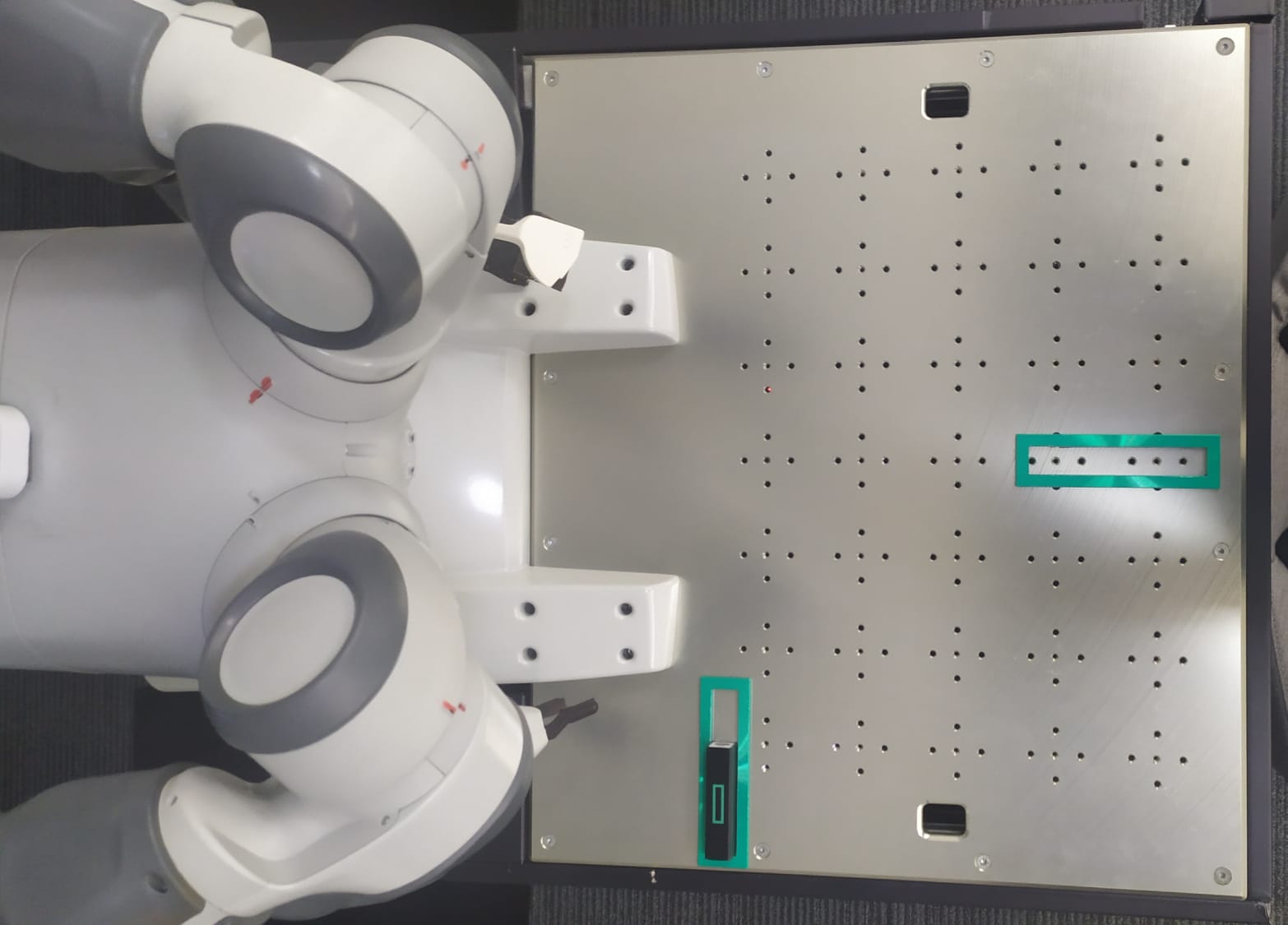
In der Anwendung ist dies so implementiert, dass es für jeden der Arme eine Liste des Typs Warteschlange erstellt. Für jedes Frame der Kamera wird dann der Wert des Winkels zwischen Oberarm und Unterarm an das Ende der entsprechenden Liste geschrieben und der älteste Wert daraus gelöscht, es seidedn der Oberarm und die gegenüberliegende Schulter haben einen Winkel kleiner gleich 100°. Sollte dies der Fall sein, oder kann kein Arm gefunden werden, wird statt dem Winkel ein Nullwert in die Liste geschrieben. Dies passiert, um sicherzustellen, dass die Person tatsächlich ihren Arm zum Winken gehoben hat.

Nachdem der aktuelle Zustand festgehalten wurde, beginnt die Analyse der Listen. Für beide Arme ist der Prozess wieder gleich. Aus jeder Liste werden die fünf größten und fünf kleinsten Werte genommen wobei die Nullwerte ignoriert werden. Wenn die Differenz zwischen diesen normalisierten Extremwerten den Grenzwert 30 übersteigt, wird dieser Arm als winkend eingestuft. Wenn dies geschehen ist, wird via eines HTTP 200 Statuscodes das Signal an den Kontrollserver gegeben, dass eine winkende Person erkannt wurde.

* 1. Kontrollserver

Die Implementierung des Kontrollservers ist relativ simpel gehalten und dient hauptsächlich als zentraler Eingangspunkt für Wartungs- und Entwickelungszwecke. In der Demonstration nimmt er jedoch eine Rolle als zentrales Bindeglied zwischen dem Vision System und der Schnittstelle des Roboters ein. Dies ist nötig, da die Kontrolleinheit vom Hersteller permanent so konfiguriert ist, dass sie ausschließlich Kommunikationen von Geräten mit der IP Adresse 192.168.125.50 annimmt. Da dies mit dem lokalen Netzwerk unvereinbar ist, wurde der Kontrollserver als eine Art Relai konfiguriert und wandelt die simplen HTTP Status codes des Vision Systems in den für den Roboter verständlichen Post-Request[[42]](#footnote-43). Dies wurde in Form einer kleinen Flask API implementiert, welche auf Port 80 auf Seiten des Visionsystems einen Webserver präsentiert[[43]](#footnote-44). Die API nimmt Vier verschiedene Arten von Anfragen an, eine Statusabfrage, ein einfaches Startsignal, ein Stoppsignal und eine Anfrage zum Starten einer Endlosschleife.

1. Evaluation
   1. Evaluierung der Programmierten Routine



2

B

1

A

C

Abb. 7 Punkte in Einflussbereich des Yumis

Das Ziel, die Bewegungsroutine so weiter zu programmieren, dass der Endzustand und der Anfangszustand der Routine übereinstimmen, wurde technisch gesehen erfolgreich umgesetzt und ist auf Programmebene korrekt.

Ablagepos [[175.47,-339.51,144.93],[0.0307155,-0.999354,-0.0114965,-0.014691],[1,0,1,5],[170.958,9E+09,9E+09,9E+09,9E+09,9E+09]]

Abholpos: [[177.39,-304.01,142.06],[0.0256908,0.999255,0.0286698,-0.00283161],[1,0,1,5],[-170.612,9E+09,9E+09,9E+09,9E+09,9E+09]]

Zwar sind die Variablen für die Position des Greifarms von an Punkt A (siehe Abb. 7) für Abholung und Ablage nicht exakt identisch, dies ist aber dem geschuldet, dass der Roboter an Stelle B (siehe Abb. 7) die Powerbank nicht mittig greifen kann, ohne das er vorher mit der Bodenplatte kollidieren würde, deshalb wird er gezwungen die Powerbank im oberen Drittel zu greifen.

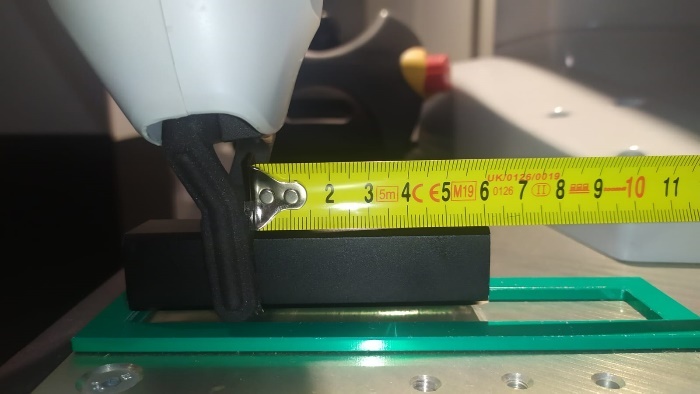
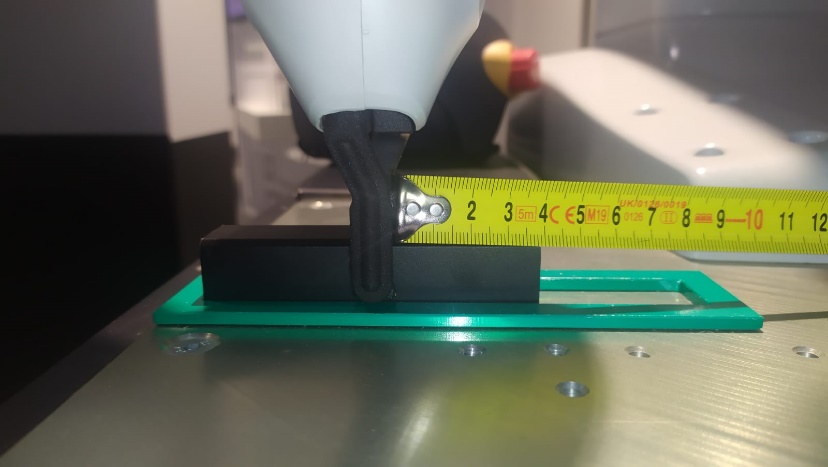


Abb. 8 Greifer in Abholposition Abb. 9 Greifer in Ablageposition

Anhand dieser Abbildung ist der Unterschied zwischen dem Roboter in der Abholposition Abb. 8 und in der Ablageposition Abb. 9 So sind die Positionsvariablen für den Arm zwar unterschiedlich, jedoch befindet sich die Powerbank an derselben Stelle.

Obwohl der Kreislauf geschlossen ist, lässt sich in der Praxis trotzdem erkennen, dass der Roboter nach einiger Zeit, ohne offensichtliche aus Einwirkungen von außen in einen kritischen Fehlerzustand gelangt.

Um besser einschätzen zu können, wie oft ein solcher Fehlerzustand tatsächlich erreicht wird, wurde der Roboter in den Dauerschleifemodus versetzt und die Zeit gestoppt bis ein solcher Zustand erreicht wurde. Im Vergleich dazu, eine Iteration, so wie sie einmalig im normalen Modus ausgeführt werden würde, beträgt in etwa 37 Sekunden

|  |  |  |
| --- | --- | --- |
| Zeit | Anzahl der Iterationen | Fehlerart |
| 1h 21 min 27s | 132 | Vorwärtssturz an Punkt B |
| 1h 08 min 34s | 111 | Vorwärtssturz an Punkt B |
| 46 min 42s | 75 | Vorwärtssturz an Punkt B |
| 2h 14 min 25s | 217 | Kollision auf weg zu Punkt B |
| 1h 50min 57s | 178 | Vorwärtssturz an Punkt B |

Tab. 1 Fehlerzustaende des ABB Yumi

Der Roboter kann auf zwei Arten in einen Fehlerzustand gelangen.

Der erste Fehlerzustand entsteht durch das Vorwärtsfallen der Powerbank, während sie von Punkt B1 (siehe Abb. 7) zu Punkt B2 (siehe Abb. 7) geschoben wird. An dieser Stelle der Routine befindet sich die Powerbank in einem besonders anfälligen Zustand, dies ist zurückzuführen auf die folgenden zwei Gründe.

Denn erstens steht sie auf einer ihrer Seitenflächen während sie von Punkt zu Punkt geschoben. Gleichzeitig liegt aber auch der Schwerpunkt der Powerbank aufgrund von ungleicher Verteilung der internen Komponenten nicht auf dem visuellen Mittelpunkt der Powerbank. Es könne die Powerbank zwar auf die andere Seitenfläche gelegt werden um damit den Schwerpunkt nach unten zu verlegen, was die Powerbank stabiler machen würde, jedoch sind auf dieser Seite die USB-A und USB-Micro-A, Lade- und Entladebuchsen verbaut, weshalb die Oberfläche auf dieser Seite deutlich unebener ist und fast sofort zu einem Fehlerzustand führt. Potenzielle Lösungsmöglichkeiten, wären eine präparierte Powerbank mit niedrigerem Schwerpunkt einzusetzen, womit die benötigte Kraft um diese Umzuwerfen erhöht werden wurde. Allerdings könnte auch der abrupte Stopp der Powerbank zu einer graduellen Entschleunigung umprogrammiert werden. Dadurch würde die das vorwärts Momentum langsam reduziert und ein äußerer Einfluss müsste stärker sein, um einen Sturz zu verursachen.

Zweitens ist die Oberfläche auf der Strecke B1-2 mit Löchern und Gewinden für die einfache Montage von stationären Elementen versehen. Diese Löcher führen manchmal zu einem Sturz, wenn sich die Kante der Powerbank mit einem der Senkränder trifft. Eine potenzielle Lösungsmöglichkeit hierbei kann aus einer Kombination des Auffüllens und Glätten der Oberfläche der Strecke B1-2 und einem Abrunden der Ränder der Powerbank sein um das Wahrscheinlichkeit einer kritischen Verkantung zu minimieren.

Der zweite Fehlerzustand entsteht durch eine ungeplante Kollision mit der Umgebung. Also wenn der Roboter im Zuge der Bewegung mit der Powerbank in der Hand eine der Kanten des HPE Symboles berührt und der dabei erzeugte Impuls höher ist als die von ABB zugelassenen Werte. Dies hat zur Folge, dass der Sensor im Motor den Not-Aus des Roboters aktiviert. Das eine solche Kollision zustande kommt, obwohl die Routine sich nicht verändert hat ist zum einen kumulativem Error zuzuschreiben, bei dem sich kleine Varianzen durch Umwelteinflüsse sich so lange ansammeln, bis sie zu einem katastrophalen Fehlerzustand führen. Eine Vorstufe für einen solchen kumulativen Fehlerzustand, ist beispielsweise eine fehlerhafte Position beim Zurücklegen der Powerbank an Punkt A siehe Abb 9. Welche den korrekten Zustand zeigt, während Abb 10 den fehlerhaften Zustand zeigt.

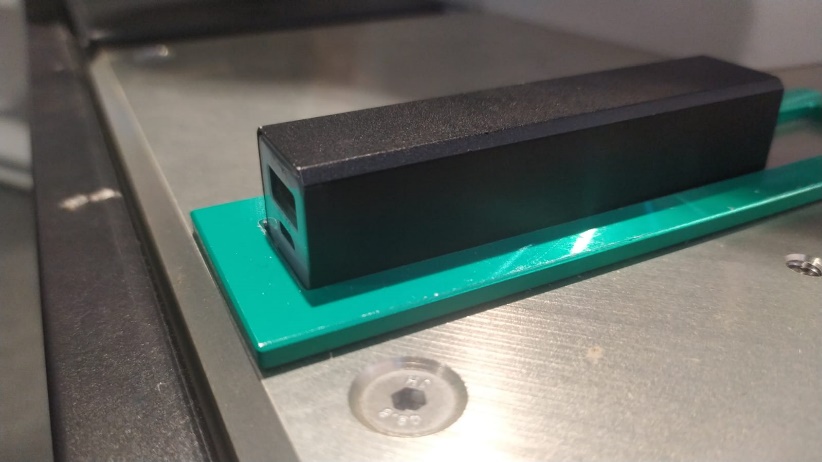
 

Abb. 11 Powerbank Position A korrekt Abb. 12 Powerbank Position A fehlerhaft

Die Vorstufe bei dem die Powerbank nicht innerhalb des HPE Symboles, sondern auf der Kante liegt, entsteht durch ein leichtes Rutschen der Powerbank während des Transportes von Punkt B nach Punkt A. Zu einem kritischen Fehlerzustand, kommt es in diesem Fall jedoch nicht, auch wenn dies oft ein Indikator eines Zukünftigen ist.

Der tatsächliche Fehlerzustand entsteht meist während der Bewegung von Punkt C (siehe Abb. 7) zu Punkt B, wenn die Unterseite der Powerbank mit dem Rand des HPE Symboles kollidiert.

Eine Möglichkeit das Problem mit dem Abrutschen der Powerbank zu adressieren wäre die Innenflächen der Greifer mit einem besser haftenden Material auszukleiden, um die Reibung zu erhöhen und das Rutschen zu verringern. Um diese Art von Problemeb systematisch anzugehen, besteht auch die Möglichkeit mit einer aktiven Fehlerkorrektur gegen den kumulativen Error vorzugehen, in dem man sich der Kameras in den Händen des Yumi bedient. Diese ermöglichen es statt an einen immer gleichen Punkt zu fahren, die Powerbank zu erkennen und die kleinen Variationen in der Positionierung zu kompensieren.

Zusammenfassend sind diese Fehlerzustände jedoch selten (siehe Tab. 1), vor allem wenn die Routine nur bei Bedarf statt kontinuierlich ausgeführt wird, entsteht ein Fehler nur in 0.81%der Fälle.

* 1. Interaktivität und Zuverlässigkeit der Demo

Um die Demo interaktiv zu machen wurde eine Gestenerkennung in Python auf der Basis von Numpy, OpenCV und Mediapipe entwickelt. Ziel war es dabei die Anwesenheit einer Person zu erkennen und dann festzustellen, ob diese Person gestikuliert.

Die Umsetzung als solche erkennt Personen via eines neuronalen Netzwerkes und bestimmt die Körperhaltung als der Winkel der Gliedmaßen zueinander. Basierend auf diesen Winkelangaben wird dann anhand von Heuristiken festgestellt, ob die Person am Winken ist.

Um den Erfolg der Gestenerkennung zu analysieren, müssen zwei Metriken betrachtet werden. Zum einen die Fehlerrate, da diese die Zuverlässigkeit des Systems bestimmt. Zum andren False Positives, also dem fälschlichen Auslösen einer Routine, obwohl eigentlich keine Person tatsächlich am Winken ist. Und die False-Negative-Rate, bei dem eine Person versucht ein Winken zu vermitteln, dieses aber vom System nicht erkannt wird.

Zusätzlich ist auch die Latenz wichtiges Kriterium, da es sich um eine Echtzeitinteraktion zwischen Maschine und Interakteur handelt. Dabei gilt es zwischen zwei Arten der Latenz zu unterscheiden. Denn zum einen ist eine gewisse Latenz dadurch gegeben ist, dass die Erkennung des Winkens in Form eines gleitenden Durchschnitts implementier ist, was eine inhärente Latenz zur Folge hat. Andererseits kann Latenz auch durch die Bearbeitungszeit pro Frame beeinflusst werden.

**Framerate Analyse**

Die Messung der Framerate funktioniert anhand einer Methode, welche bei jeder Iteration des Programmes die vergangene Zeit misst und alle zehntel Sekunde die aktuellen Iterationen pro Sekunde misst. Jede Iteration ist dabei die Verarbeitung eines einzelnen Bildes. Unter praktischen Bedingungen wird der maximalen Bildrate ein Limit durch die Geschwindigkeit der Kamera gesetzt, wie auch in Abb 11 zu erkennen. Denn während des gesamten Prozesses, selbst wenn Personen sich Personen in das Sichtfeld der Kamera bewegen, bleibt die Anzahl der Bilder pro Sekunde bei circa 30 FPS. Dies ergibt sich daraus, dass die Kinect V2 laut Datenblatt nur in der Lage ist 30 Bilder pro Sekunde zu liefern[[44]](#footnote-45).

Abb. 12 Framerate mit Kinect

Die Anomalien in den ersten 3 Sekunden sind durch den Initialiserungsprozess der Kamera zu erklären, da sie erst gar kein und dann leere Bilder sendet, bis sie vollkommen einsatzfähig ist.

Da diese Messung durch die maximale Bildrate der Kamera limitiert ist und deshalb wenig Aussagekraft über die tatsächliche Umsetzung der Gestenerkennung hat wird zusätzlich auch getestet, wie sich der Anwendungscode verhält, wenn er nicht von der Kamera begrenzt ist. Um dies zu erreichen, wird mit der Kinect ein Video aufgenommen und die Gestenerkennung damit gespeist. Dies hat zu folge, dass die einzelnen Frames so schnell zur Verfügung stehen, wie sie die Festplatte liefern kann.

Abb. 13 FPS der Gestenerkennung

Diese beiden Graphen basieren auf demselben Video und an den Bereichen wo die Framerate absackt ist es deutlich zu erkennen wann sich eine Person im Sichtbarlich der Kamera befindet und wann nicht. Dies lässt sich dadurch erklären, dass der Teil des Codes, an dem die Position und Winkel der Extremitäten bestimmt wird, schlichtweg übersprungen wird, wenn sich keine Person im Bild befindet. Der obere Testverlauf, in dem die bearbeiteten Videodaten nur auf den Bildschirm gestreamt werden und der Untere wo das resultierende Video zusätzlich noch auf der Festplatte gespeichert wurde. Aus den gewonnenen Daten zeigt sich, dass die Anwendung mit einer ausreichenden Geschwindigkeit agiert, bei komplexen Erweiterungen der Anwendung sollte jedoch in Betracht gezogen werden, ob die Leistung der Hardware ausreichend ist.

**Analyse der Fehlerrate und Latenz**

Um die Fehlerrate und die Latenz der Gestenerkennung zu messen, wurde das bereits aufgenommene Video manuell analysiert und die die vorlaufende Nummer des Frames, in welchen eine oder beide Arme am Winken sind notiert. Als Anfang der Wink-Aktion wird dabei der Frame genommen, ab welchem sich die Hand oberhalb der Brust befindet, jedoch nur wenn es danach zu einem Winken kommt. Der Fall am Ende, bei welchem die Hand nur gehoben, aber nicht winkt wird also nicht als eine Wink-Aktion gewertet, auch wenn sich die Hand oberhalb der Brust befindet. Aus dieser manuellen Analyse geht dann diese Musterlösung hervor, (siehe Abb 13). Bei diesem befindet sich die Nummer des Frames auf der X-Achse und auf der Y-Achse die Information, ob der Arm als Winkend erkannt wurde. Für eine bessere Übersicht wird der Linke arm in der oberen Hälfte dargestellt und der rechte Arm in der Unteren. Bei nicht winkendem arm bleibt die entsprechende Linie in der Mitte und schlägt bei Winken je nach arm nach oben oder unten aus.

Abb. 14 Manuele Analyse des Tesvideos

Daraufhin wurde das selbe Video der Softwareimplementierung zu Verfügung gestellt, welche folgendes Ergebnis generierte (siehe Abb. 14). Aus diesem ergibt sich zwar, dass jedes Mal, wenn gewunken wurde, dies tatsächlich erkannt wurde. Also auch eine entsprechende Routine beim Roboter ausgelöst worden wäre. Jedoch lasst sich erkennen, das das System trotzdem Probleme damit hat ein Winken in seiner ganzen Länge zu erkennen und in insgesamt AchtAbschnitten fehlschlägt ein Winken zu erkennen. Darüber hinaus lässt sich gegen Ende eine erhöhte False-Positive Rate erkennen. Also dem Detektieren einer Geste, wo in Realität keine ist. Dies kommt daher, dass gegen Ende des Videos, um die Limitationen des Modells hervorzuheben, jeweils beide Hände gehoben werden, jedoch ohne zu winken. Die Bewegung der Hand wird dabei jedoch trotzdem als winken erkannt, was zwei False-Positives auslöst.

Abb. 15 Automatische Analyse des Testvideos

Die Latenz der Gestenerkennung hält sich jedoch dabei in Grenzen. Im Schnitt wird die Gesten zwei bis drei Frames länger erkannt als sie eigentlich da wäre. Da sich dies jedoch nur bei Beendigung der Geste bemerkbar macht und nicht bei Beginn, fällt dies in der Praxis nicht auf, denn ab diesem Moment ist der Roboter schon gestartet und mit Ausnahme eines Indikators auf dem Bildschirm ist es von außen nicht erkennbar, dass deine Latenz in der Erkennung besteht.

Alles in allem ist die Lösung nicht perfekt jedoch adäquat für die Anforderungen. Weitere Schritte könnten hierbei die Entwickelung oder Implementierung eines spezifisch auf Winken trainiertes neuronales Netzwerk sein.

* 1. Entwickelung der API

Das Ziel der API war es via eines Webservers eine Schnittstelle im CTC Netzwerk zur Verfügung zu stellen, welche in der Lage ist eine Routine des Roboters bei Anfrage zu starten.

Die dazu entwickelte Lösung erfüllt dabei die Anforderungen, in dem ein einfaches Get  
-request angenommen wird, welches als Postrequests an die Kontrolleinheit des Roboters weitergegeben wird. Über den Umfang der Anforderung hinaus besteht nun auch noch die Möglichkeit den Roboter in den Enlosschleifen-modus zu versetzen und diese zu unterbrechen.

Verbesserungspotential besteht hier bei der Sicherheit. Denn die schnittstelle des Roboters erfordert eine Authentifizierung. Diese wird von der API automatisch für jede anfrage erteilt und das Password ist im Programmcode hardgecodet. Zwar ist das das CTC N etzt sowieso nur für Administratoren des CTC zugänglich und bei der benötigten Authentifizierung handelt es sich auch nur um das vom Werk vorkonfigurierte Standardpassword. Trotzdem wäre die Implementation einer Authentifizierung auf Seiten der API selbst und einer Abänderung des Passwortes auf Seiten des Roboters von Vorteil, um unbefugten Zugriff zu verhindern.

Darüber hinaus besteht noch hohes Potential für die Weiterentwicklung der API selbst, beispielsweise eine Möglichkeit zu finden automatisch arbiträren RAPIDCode auf den Roboter aufzuspielen, ohne diesen Manuell via RobotStudio hochladen zu müssen. Die Möglichkeit anhand eines ROS Interfaces live Bewegungssequenzen auf dem Roboter ausführen zu lassen, ohne diese erst als Methode aufspielen zu müssen bietet hier auch ein hohes Potential für dynamische Anwendungen wie das Nachmachen einer Körperhaltung.

* 1. Kinect

Das Ziel der Kinect war es eine zuverlässige Erkennung von Personen zu implementieren, um das manuelle Auslösen der Roboterroutine zu automatisieren.

Die dazu entwickelte Lösung basierend auf dem Maschine Learning Modell von Google namens Mediapipe funktioniert zuverlässig und schnell. Die Auslastung des Intel NUC8i7HN beträgt jedoch während des betriebs durchgehend 60 Prozent. Dies ist hauptsächlich dem fakt zuzuschreiben ist, dass das Mediapipe nicht von GPU Beschleunigung profitieren kann, da der NUC8i7HN keine dedizierte Grafik Einheit hat, sondern eine Iris Pro Graphics 580 verwendet, also eine im Prozessor integrierte Grafik Einheit, welche nicht für diese Art von Berechnungen ausgelegt ist. Diese sind für sehr geringe Anforderungen wie das Darstellen einer Webseite oder dem Transcodieren eines Videos adäquat. Zwar ist die Rechenleistung des Geräts für die Anwendung ausreichend, jedoch ist der Preis für diese Leistung mit 714 Euro hoch[[45]](#footnote-46).

Eine Alternative, welche mehr Leistung für Komplexere Aufgaben zur Verfügung stellen könnte und dabei noch ein besseres Preis-Leistungs-Verhältnis bieten würden wäre die Verwendung von dedizierter Beschleunigungshardware spezifisch ausgelegt für neuronale Netzwerke. Da diese nur für diesen auf der Silicium Ebene dafür entwickelt wurden neuronale Netze zu berechnen, haben diese einen massiven Vorteil in Effizienz und Leistung gegenüber einer generalistischen Einheit wie einer CPU[[46]](#footnote-47). So hat ein Raspberry pi4 in Verbindung mit einem Coral USB-Akzelerator das Potential komplexere Aufgaben zu erfüllen, wie beispielsweise das Tracking von mehr als einer Person zur selben Zeit.

* 1. Erweiterbarkeit und leichte Wartung

Gleichzeitig war es auch Ziel es zukünftigen Studenten es möglichst leicht zu machen die Demo zu verstehen, zu verändern und zu erweitern.

Um einen möglichst schnelle und einfachen Eingewöhnung Phase zu ermöglichen, wurde auf dem Kontrollserver die benötigten Software Tools vorinstalliert und mit den entsprechenden Einstellungen vorkonfiguriert. So wurden wichtige entwickelunsumgebungen wie VSCode, RobotStudio und Postman installiert aber auch die SSH Verbindung zu dem Visonsystem wurde mit SSH schlüsseln und einer festen XServer-Session-weiterleitung über SSH so eingerichtet, dass diese automatisch funktionieren.

Außerdem wurde darauf geachtet, dass hauptsächlich weitverbreitete Software genutzt wurde. Zum einen um die Möglichkeit zu erhöhen das zukünftige studierende mit der Software schon bekannt sein könnten und zum anderen für diese Studenten den Nutzen den sie für sich selbst ziehen zu maximieren. So hätte sich beispielsweise angeboten die proprietäre Personenerkennung von Microsoft für die Kinect zu nutzen. Doch ist diese nicht nur nichtmehr offiziell unterstützt, sondern ist vor allem ausschließlich für die Kinect anwendbar. Der alternative Lösungsansatz mit OpenCV und Mediapipe ist nicht nur modularer und damit leichter weiterzuentwickeln sondern auch hardwareagnostisch, also nicht an die Kinect gebunden, womit das gelernte direkt in den meisten Computer Vision Anwendungen genutzt werden kann.

Auf Seiten des Roboters gibt es hier jedoch noch Verbesserungspotential, da die Programmierung via der ABB Spezifischen Robot Studio Plattform stattfindet, obwohl diese eigentlich auch die generalistische Plattform „Robto Operating System“ unterstützt, doch der Mehraufwand die bestehende Routine in ROS zu übertragen hätte in diesem Fall Zeit von der Entwickelung der Gestenerkennung in Anspruch genommen, welche jedoch eine höhere Priorität hatte. Jedoch wäre die Flexibilität von ROS eine wertvolle Ressource für zukünftige Projekte und sollte bei Weiterentwicklung des Projektes in Betracht gezogen werden.[[47]](#footnote-48)

1. Learning

Zusammenfassung:

Ziel dieser Arbeit war es zu untersuchen, inwiefern ein Kundenshowcase basierend auf einer Microsoft Kinect V2, eine Gestenerkennung implementiert werden kann, welche automatisch Bewegungsroutinen des Roboters auslöst, um die Aufmerksamkeit von Kunden zu erregen. Dazu wurde eine Proof of Concept (POC) erstellt, welche die Grundfunktionalität einer solchen Gestenerkennung demonstriert.

In dem Kapitel Grundlagen wurde dementsprechend die Grundkonzeption der Robotik, die Showcases und der Computervision erläutert. In dem Kapitel der Methodik wurde dann die wissenschaftliche Methode des Prototyping nach Cole et. al. vorgestellt und dementsprechend das Vorgehen der Arbeit beschrieben.

Das Kapitel der Problem Identifizierung skizziert den Kontext, die Umgebung und die Probleme, welche durch die Prototypen adressiert werden sollen.

Das darauffolgende Kapitel der Intervention beschreibt die Umsetzung der drei Zentralen Komponenten des Prototyps und deren Integration. Zuerst wurde die Programmierung der Bewegungsroutine des Roboters mit Hilfe von Robot Studio und der Programmiersprache RAPID beschrieben. Danach wurde die Entwickelung der Gestenerkennung mithilfe der Microsoft Kinect V2, der Programmiersprache Python und den entsprechenden Computer Vision Bibliotheken beschrieben. Um die Geste des Winkens zu erkennen, wurde auch ein heuristischer Algorithmus entwickelt. Schließlich wurde die Integration der beiden Komponenten durch die Entwicklung einer Flask basierten Schnittstelle beschrieben, welche mit einem Separaten Kontrollserver umgesetzt wurde.

Im Letzten Kapitel findet eine genaue Analyse der Schwächen des Prototyps statt, dabei wird nach den Anforderungen vorgegangen, welche im ersten Schritt der Methode des Prototyping identifiziert wurden. Dabei wird auch die Frage beantwortet, ob der Prototyp die Anforderungen erfüllt und ob er die Probleme löst, welche in dessen Umgebung identifiziert wurden.

Reflektion:

Die vorliegende Arbeit zeigt, dass erfolgreich ein Prototyp erstellt worden ist, der die grundlegenden Bedingungen erfüllt, jedoch von weiteren Iterationen der Entwickelungen profitieren würde. Die dabei aus der Praxis stammende Forschungsfrage konnte durch die Erstellung des Artefaktes als solches beantwortet werden.

Für die Lösung der praktischen Probleme, war die Forschungsfrage sehr gut geeignet, hätte jedoch von weiteren begleitenden wissenschaftlichen Fragestellungen ergänzt werden können, da sie als rein praktische Fragestellung in einem akademischen Kontext einen zu weiten Fokus hat, welcher ohne weitere Forschungsfragen nur schwierig ausgeschöpft werden konnte.

Ähnliches gilt für die Methode des Prototyping, welche zwar sehr gut geeignet war, um die Anforderungen der Praxis zu lösen, jedoch durch den Einsatz von einer zusätzlichen wissenschaftlichen Methodik, um die Anforderungen fundierter Erfassung zu können, hätte ergänzt werden könnte. In einer zweiten Iteration des Prototyping würde es sich daher lohnen, das bisherige Methodenspektrum um die der Design Science Research zu ergänzten, um die Anforderungen der Praxis noch besser zu erfassen.

Auch hat die Evaluation eine starke Orientierung auf die praktischen Anforderung, wobei auch hier eine zusätzliche wissenschaftliche Methode die Arbeit gut ergänzt hätte.

Jedoch darf dabei der Zeitliche Aufwand nicht unterschätzt werden, da die Arbeit aufgrund der Komplexität der Thematik und der damit verbundenen Probleme bereits sehr zeitintensiv war. Basieren auf dem bisher erarbeiteten Prototypen, besteht die Möglichkeit die Entwicklung in eine weitere Iteration zu überführen, und in dieser durch zusätzlich Methoden die Erfassung der Anforderungen zu ergänzen. Dies hätte jedoch in der ersten Iteration und auch in der vorliegenden Arbeit den Zeitlichen Rahmen überschritten.

Weitere einschränkende Faktoren neben dem zeitlichen Rahmen, welcher durch die Komplexität der Thematik und der damit verbundenen Probleme bereits ausgeschöpft war, war die begrenze Auswahl an Hard- und Software.

Die durch Firma zur Verfügung gestellte Hardware war für die Entwicklung des Prototyps ausreichend, grenzte jedoch die Auswahl an Entwicklungsmöglichkeiten ein. Beispielsweise war die zugrundeliegende Bewegungsroutine des Roboters in Robot Studio entwickelt worden, was eine Verwendung eines offenen Frameworks wie ROS ausschloss. Auch war die Entwickelung der Gestenerkennung Hardware technisch durch die Firma auf die Microsoft Kinect V2 beschränkt und lies bei der Programmierung auch keine Wahlfreiheit, da der Hersteller die Software Entwickelung Kit nicht mehr supportet, weshalb die Wahl eines offenen Frameworks wie pylibfreenect2 alternativlos war. Die Entwicklung des Prototyps war somit stark von der Hardware und der Software abhängig, welche von externen vorgegeben war.

Zusätzlich sollte das Komplexität Niveau des gesamten Projektes möglichst klein gehalten werden, um den Zeitlichen Rahmen für nachfolgende Entwickler gering zu halten. Dies war jedoch nicht immer möglich, da die Komplexität der Thematik bereits sehr hoch war.

Fazit und Ausblick:

Die vorliegende Arbeit zeigt, dass das der Prototyp die Anforderungen erfüllt und die Probleme löst. Dabei wurde die Grundfunktionalität einer Gestenerkennung demonstriert, welche automatisch Bewegungsroutinen des Roboters auslöst, um die Aufmerksamkeit von Kunden zu erregen. Der Prototyp wurde in der Firma Hewlett Packard Enterprise (HPE) entwickelt und ist in der Praxis bereits im Einsatz. Weitere Entwickelung in Form sind wie von vornherein geplant, weshalb die einfache Wartung ein wichtiger Aspekt dieser Arbeit war. Die Entwicklung des dieses Artefaktes hat gezeigt, dass die Methode des Prototyping nach Cole et al. sehr gut geeignet ist um in der einen Prototypen zu entwickeln, welcher die Anforderungen der Praxis erfüllt, jedoch von weiteren wissenschaftlichen Methoden ergänzt werden sollte, um die Anforderungen der Praxis fundierter erfassen zu können.

# Literaturverzeichnis

**Gorecky, D. /Schmitt, M. /Loskyll, M. /Zühlke, D. (2014):** Human-Machine-Interaction in the Industry 4.0 Era, Kaiserslautern

**Rautaray S.S. /Agrawal A. (2012):** Vision based hand gesture recognition for human computer interaction: a survey, Allahabad India

**Schlenoff C. /Prestes E. /Madhavan R. /Goncalves P. /Li H. /Balakirsky S. /Kramer T. /Miguelanez E, (2012):** An IEEE Standard Ontology for Robotics and Automation, Vilamoura, Algarve, Portugal

**Warnecke H.-J. /Schraft R. D. /Haegele M. /Barth O. /Schmierer G. (1999):** MANIPULATOR DESIGN in Handbook of Industrial Robotics, Second Edition, New York John Wiley & Sons, Inc.

**Clack D. R. / Letho M. R. (1999):** RELIABILITY, MAINTENANCE, AND SAFETY OF ROBOTS in Handbook of Industrial Robotics, Second Edition, New York John Wiley & Sons, Inc.

**Asfahl C.R. (1999):** SENSORS FOR ROBOTICS in Handbook of Industrial Robotics, Second Edition, New York John Wiley & Sons, Inc.

**ISO 10218-1 (2006):** Robots for industrial environments - Safety requirements - Part 1: Robot; ISO Copyright Office, Geneva.

**ISO 10218-2 (2008):**, Robots for industrial environments - Safety requirements - Part 2: Robot system and Integration ISO Copyright Office, Geneva

**Bjoern M. (2014):** Kollaborierende Roboter – Stand der Safety-Normung und Umsetzungsbeispiel YuMi®, 18. IFF-Wissenschaftstage, Fachtagung Assistenzrobotik und Mensch-Roboter-Kollaboration, Magdeburg, Germany

**Cambridge Dicitionary. (o. J.) a:** Bedeutung von showcase im Englischen, <https://dictionary.cambridge.org/de/worterbuch/englisch/showcase>, abgerufen am 04.09.2022

**Thinius J. /Untiedt J. (2017**): Events – Erlebnismarketing für alle Sinne, 2. Auflage, Springer Gabler, Essen/Dortmund , Deutschland

**Duncan T. /Moriarty S. E. (1998):** A Communication-Based Marketing Model for Managing Relationships, Journal of Marketing

**Saad E. /Neerincx M. A. / Hindiriks K. V. (2019):** Welcoming Robot Behaviors for Drawing Attention, Delft, The Netherlands

**Zhong-Qiu Z. / Peng Z. /Xu S. T. /Wu X. (2019):** Object Detection With Deep Learning: A Review, IEEE

**Dresch, A./Lacerda, D. P./Antunes, J. A. V. (2014**): Design science research, A .method for science and technology advancement, Cham: Springer

**ABB (2015):** Product manual IRB 14000-0.5/0.5 IRC5, Revision: Q

**Xiang L. /Echtler F. /Kerl C. u.a. (2016):** libfreenect2: Release 0.2 (v0.2). Zenodo. https://doi.org/10.5281/zenodo.50641

**Microsoft (2022):** Kinect for Windows, <https://docs.microsoft.com/en-us/windows/apps/design/devices/kinect-for-windows> 04.09.2022

**Fankhauser P. /Bloesch M. /Rodriguez D. /Kaestner /Hutter M. /Siegwart R. (2015**): "Kinect v2 for mobile robot navigation: Evaluation and modeling," 2015 International Conference on Advanced Robotics (ICAR)

**Lugaresi C. /Tang J. /Nash H. (2019):** MediaPipe: A Framework for Building Perception Pipelines, arXiv:1906.08172v1, Google

**Mediapipe (o.J**.): MediaPipe Pose, <https://google.github.io/mediapipe/solutions/pose>, abgerufen am 04.09.2022

**Lachat E./Macher H. /Landes T. /Grussenmeyer P. (2015):** Assessment and Calibration of a RGB-D Camera (Kinect v2 Sensor) Towards a Potential Use for Close-Range 3D Modeling - Scientific Figure on ResearchGate.

o. V. (o. J.) b: Amazon NUC8i7HN, <https://www.amazon.com/Intel-NUC-Performance-G-Kit-NUC8i7HVK/dp/B07BR5GK1V?language=de_DE&currency=EUR>, 04.09.202

**Hesse C. N. (2021)**: Analysis and Comparison of Performance and Power Consumption of Neural Networks on CPU, GPU, TPU and FPGA

**Mak. A (2020):** HandBook Guidance on the programming of ABB YuMi IRB 14000

ABB Robotics (2021): RobotStudio tutorial l Recording Playback and Stop Position,

<https://www.youtube.com/watch?time_continue=5&v=JGKTlhS6Kvs> Abruf: 24.12.2022

**Werner Hampel - Der Roboterkanal (2020):** Erstes BEWEGUNGSPROGRAMM mit einem ABB Roboter: Step by Step, <https://www.youtube.com/watch?v=Qmn47UTq8xI> Ab-  
ruf: 24.12.2022

**Roman Parak (2021):** ABB RobotStudio - Basic control of the YuMi (IRB 14000) collaborative robot, <https://github.com/rparak/ABB-RobotStudio-YUMI> Abruf: 24.12.2022

ABB Robotics **(2018):** ABB ROBOTICS IRB YuMi. Produktpräsentation, https://docplayer.org/76043385-Abb-robotics-irb-yumi-produktpraesentation.html Abruf 24.12.2022 S. 5

**Thomas C. /Matthias B. /Kuhlenkötter B. (2016):** Human‐Robot Collaboration – New Applications in Industrial Robotics : International Conference in Competitive Manufacturing 2016 (COMA'16)

**Zanchettin A. M. / Croft E. /Ding H. /Li M. (2018):** "Collaborative Robots in the Workplace [From the Guest Editors]," in IEEE Robotics & Automation Magazine, , vol. 25, no. 2, S16-17

**Malik A. A. / Bilberg A. (2019):** Collaborative robots in assembly: A practical approach for tasks Distribution in in: 52nd CIRP Conference on Manufacturing Systems ,S. 665-671

**Schou C. /Andersen R. S. /Chrysostomou D.** **(2018)** : Making collaboration with robots easier, in Robotics and Computer-Integrated Manufacturing, Volume 53, S 72-80

**Wang F. /Li Y.:** Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, S. 596-603

**Raju A. / Mahendiran M. / Ahmed S. S (2022):** "Newfangled 3d human pose estimation using MediaPipe with foreground object detection", AIP Conference Proceedings 2640,

**Bharath K (2020)** OpenCV: Complete Beginners Guide To Master Basics Of Computer Vision With Codes!, <https://towardsdatascience.com/opencv-complete-beginners-guide-to-master-the-basics-of-computer-vision-with-code-4a1cd0c687f9> , Abruf: 24.12.2022

**Rosebrock A. (2018):** OpenCV Tutorial: A Guide to Learn OpenCV, <https://pyimagesearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv/> Ab-  
ruf: 24.12.2022

**Hassan M. (2021):** Advanced Computer Vision with Python - Full Course, <https://www.youtube.com/watch?v=01sAkU_NvOY> , Abruf 24.12.2022

**Winfuture** (2017): Kinect ist offiziell tot, Microsoft beendet die Produktion des Zubehörs, <https://winfuture.de/news,100258.html>, Abruf 24.12.2022

**sa125 (2011):** [Configure Flask dev server to be visible across the network](https://stackoverflow.com/questions/7023052/configure-flask-dev-server-to-be-visible-across-the-network), <https://stackoverflow.com/questions/7023052/configure-flask-dev-server-to-be-visible-across-the-network>, Abruf 25.12.2022

**Rosebrock A. (2018)**: YOLO object detection with OpenCV, <https://pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/> Abruf 25.12.2022

**OpenCV (2022):** Video I/O with OpenCV Overview, <https://docs.opencv.org/3.4/d0/da7/videoio_overview.html> Abruf 25.12.2022

**Dyouri A. (2020)**: How To Make a Web Application Using Flask in Python 3, <https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3> Abruf 25.12.2022

**Blankemeyer, S. / Göke, J. / Grimm, T. / Meier, B. / Raatz, A. :** Gesture-Based Robot Programming Using Microsoft Kinect , Precision assembly (International seminar) IN Precision assembly (International seminar) ; 72-81, International Federation for Information Processing -Publications- IFIP ; 530

**Webb, J., & Ashley, J. (2012).** Beginning Kinect programming with the Microsoft Kinect SDK ; [create cool Kinect applications using the Microsoft Kinect SDK]. In The expertś voice in Microsoft. Apress.

**Yeo, HS / Lee, BG. / Lim, H. (2015):** Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. Multimed Tools Appl 74, 2687–2715

.

**Erklärung**

Ich versichere hiermit, dass ich meine Projektarbeit Programmierung eines Industrieroboters für ein Kunden Showcase selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Heilbronn, 28.12.2022 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ort, Datum) (Unterschrift)

1. Vgl. Gorecky u.a. 2014 S 1-2 [↑](#footnote-ref-2)
2. Vgl. Rautaray/Agrawal 2012 S 41-43 [↑](#footnote-ref-3)
3. Vgl. Schlenoff u.a 2012 S 1 [↑](#footnote-ref-4)
4. Vgl. Schlenoff u.a 2012 S 3-4 [↑](#footnote-ref-5)
5. Mit Änderungen entnommen aus: Warneckeu.a. 1999 S 44 [↑](#footnote-ref-6)
6. Vgl. Clark/Letho 1999 S 735 [↑](#footnote-ref-7)
7. Vgl. Asfahl 1999 S 245 [↑](#footnote-ref-8)
8. Vgl. Malik/Bilberg (2019) S 666 [↑](#footnote-ref-9)
9. Vgl. Asfahl 1999 S 246 u. Thomas/Matthias/Kuhlenkötter 2016 [↑](#footnote-ref-10)
10. Vgl. ISO 2006 10218-1 [↑](#footnote-ref-11)
11. Vgl. ISO 2008 10218-2 [↑](#footnote-ref-12)
12. Mit Änderungen entnommen aus: Bjoern 2015 S 6 [↑](#footnote-ref-13)
13. Vgl Cambridge Dictionary. o.J. [↑](#footnote-ref-14)
14. Vgl Thinius/Untiedt 2017 S 67-68 [↑](#footnote-ref-15)
15. Vgl Thinius/Untiedt 2017 S 157 [↑](#footnote-ref-16)
16. Vgl. Duncan/Moriarty 1998 S 8 [↑](#footnote-ref-17)
17. Vgl. Priese 2015 S. V [↑](#footnote-ref-18)
18. Vgl. Zhao u.a 2019. S. 1 u Rosebrock (2018) [↑](#footnote-ref-19)
19. Vgl. Zhao u.a. 2019 S. 2 [↑](#footnote-ref-20)
20. Vgl. Wang/Li 2013 [↑](#footnote-ref-21)
21. Wilde/Hess 2006 S. 8 [↑](#footnote-ref-22)
22. Vgl. Saad/Neerincx/Hindriks (2019) S 1-2 [↑](#footnote-ref-23)
23. Vgl. ABB 2015 S 37 [↑](#footnote-ref-24)
24. Vgl. ABB Robotics 2018 S. 5 [↑](#footnote-ref-25)
25. Vgl. ABB 2015 S 34 [↑](#footnote-ref-26)
26. Entnommen aus ABB 2015 S 57 [↑](#footnote-ref-27)
27. ABB Robotics (2022) [↑](#footnote-ref-28)
28. Vgl. Parak (2021) [↑](#footnote-ref-29)
29. Vgl. Schou/Andersen/Chrysostomou (2018) [↑](#footnote-ref-30)
30. Vgl. Hampel 2020 [↑](#footnote-ref-31)
31. Vgl. Frankhauser u.a. S 2-3 [↑](#footnote-ref-32)
32. Xiang/Echtler/Kerl 2016 [↑](#footnote-ref-33)
33. Webb/Ashley 2012 [↑](#footnote-ref-34)
34. Vgl. Microsoft 2022 [↑](#footnote-ref-35)
35. Vgl. Winfuture 2017 [↑](#footnote-ref-36)
36. Vgl. OpenCV 2022 [↑](#footnote-ref-37)
37. Vgl Yeo/Lee/Lim 2015 [↑](#footnote-ref-38)
38. Vgl. Bharath 2020 u Hassan 2021 [↑](#footnote-ref-39)
39. Vgl. Lugaresi/Tang/Nash 2019S 1 ff. [↑](#footnote-ref-40)
40. Vgl. Raju/Mahendiran/Ahmed 2022 [↑](#footnote-ref-41)
41. Entnommen aus Mediapipe o. J. [↑](#footnote-ref-42)
42. Vgl. Dyouri (2020) [↑](#footnote-ref-43)
43. Vgl. sa125 (2011) [↑](#footnote-ref-44)
44. Vgl. Lachat u. a. 2015 [↑](#footnote-ref-45)
45. Vgl. o. V. o.J. b [↑](#footnote-ref-46)
46. Vgl. Hesse 2021S. 77 [↑](#footnote-ref-47)
47. Vgl. Mak 2020 S. 6 [↑](#footnote-ref-48)