

# Programmierung eines Industrieroboters für ein Kunden Showcase

## 1. Projektarbeit

vorgelegt am 05. September 2022

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs IMBIT 2021

von

Jon Eckerth

Betreuer in der Ausbildungsstätte:

Hewlett Packard Enterprise

Andreas Fuchs

CTC Technology and Operations Lead

DHBW Stuttgart:

Prof. Dr. Thomas Kessel

## Inhaltsverzeichnis

Abkürzungsverzeichnis .....	IV
Abbildungsverzeichnis.....	V
1 Einleitung .....	1
2 Grundlagen .....	2
2.1 Robotik.....	2
2.2 Showcases.....	4
2.3 Computer Vision.....	5
3 Prototyping nach Cole et. al. ....	6
4 Problem Identifizierung.....	7
5 Intervention .....	9
5.1 ABB Yumi IRB14000 .....	9
5.1.1 Routinenprogrammierung mit RAPID .....	10
5.2 Visionsystem.....	12
5.2.1 Kinect.....	12
5.2.2 pylibfreenect2.....	12
5.2.3 Numpy & OpenCV.....	13
5.2.4 Mediapipe Pose .....	13
5.2.5 Gestenerkennungs Logik .....	14
5.3 Kontrollserver.....	15
6 Evaluation .....	16
6.1 Evaluierung der Programmierten Routine.....	16
6.2 Interaktivität und Zuverlässigkeit der Demo.....	19
6.3 Entwicklung der API .....	22
6.4 Kinect.....	23
6.5 Erweiterbarkeit und leichte Wartung.....	24
7 Learning .....	25
Literaturverzeichnis .....	26



**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
Abb. 2: Kollaborierender Betrieb Arten Gemäß ISO 10218-1.-2 und ISO/TS 15066 .....	4
Abb. 3: ABB Yumi .....	10
Abb. 4: Screenshot Robotstudio .....	11
Abb. 5: Erkennen einer Geste während eines Testlaufs im CTC .....	13
Abb. 6: Aufteilung der Orientierungspunkte .....	14
Abb. 7: Punkte in Einflussbereich des Yumis.....	16
Abb. 8: Greifer in Abholposition .....	16
Abb. 9: Greifer in Ablageposition .....	16
Abb. 10: Powerbank Position A korrekt .....	18
Abb. 11: Powerbank Position A fehlerhaft .....	18
Abb. 12: Framerate mit Kinect.....	20
Abb. 13: FPS der Gestenerkennung.....	21
Abb. 14: Manuele Analyse des Tesvideos .....	22
Abb. 15: Automatische Analyse des Testvideos .....	22

## Tabellenverzeichnis

Tab. 1: Fehlerzustaende des ABB Yumi.....	14
---	----

## 1 Einleitung

Die intuitive Kollaboration zwischen Menschen und Maschine bietet ein hohes Potential in der Weiterentwicklung der industriellen Produktion. Denn sie ermöglicht gleichzeitig die Nutzung der hohen Flexibilität und strategische 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 resultierendem Unfallpotential nur beschränkt möglich Mensch und Robotik direkt miteinander interagieren zu lassen<sup>1</sup>. Durch große Fortschritte in der Echtzeitverarbeitung von Sensordaten, insbesondere im Bereich von Computer-Vision, rückt die gestenbasierte Interaktion zwischen Mensch und Maschine langsam in den Bereich des Möglichen.<sup>2</sup>

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 semi-permanenten Ausstellungsstücks im CTC, bei Kunden einen positiven Eindruck zu hinterlassen und als Konversationsstarter zu funktionieren. 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 Ursprungs Konfiguration 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 machen, wurde nun eine Microsoft Kinect V2 angeschafft und die Anforderung gestellt mit dieser eine Weiterentwicklung der Showcase umzusetzen.

Das Ziel der Projektarbeit ist es, basierend auf einer Microsoft Kinect V2, einen Prototyp zu entwickeln, welche in der Lage ist, anhand einer Gestenerkennung, Bewegungsroutinen des Roboters auszulösen.

Um dies zu erreichen, wird via des Prototyping Prozesses nach Cole et. Al. ein Artefakt erstellt. Zuerst werden die Theoretischen Konzepte erklärt, dann besagte Prototyping Methode erklärt und angewandt. Dies geschieht in dem erst die Teilprobleme identifiziert werden, dann via einer Intervention adressiert werden und diese Intervention dann letzten Endes evaluiert wird.

---

<sup>1</sup> Vgl. Gorecky u.a. 2014 S 1-2

<sup>2</sup> Vgl. Rautaray/Agrawal 2012 S 41-43

## 2 Grundlagen

### 2.1 Robotik

Da es sich bei der Robotik um ein sehr 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<sup>3</sup>. 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<sup>4</sup>.

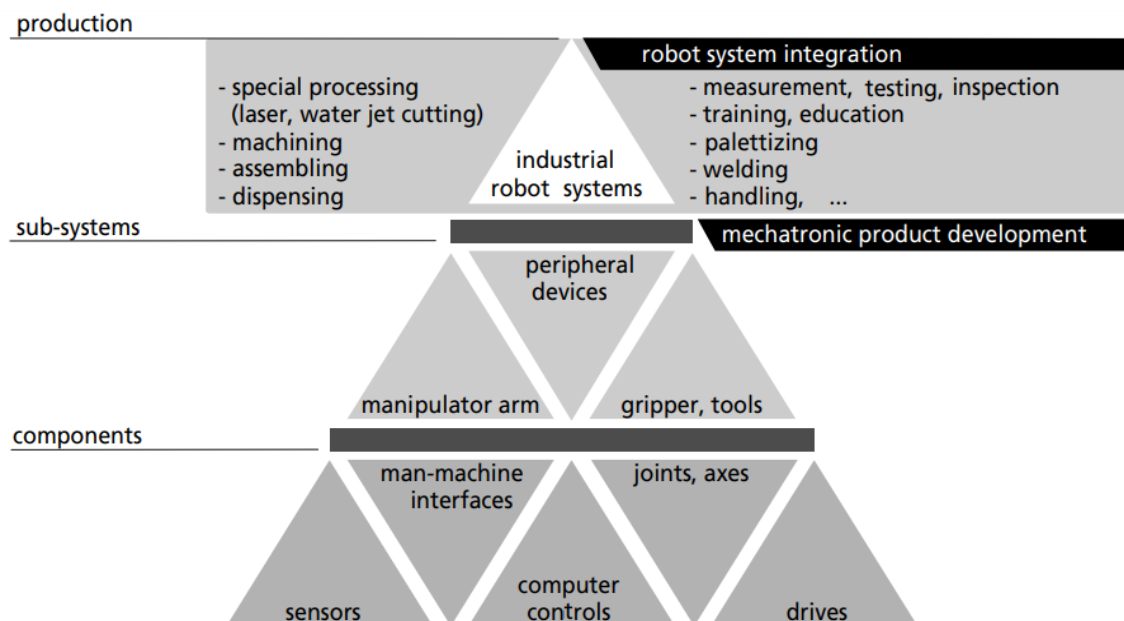


Abb. 1 Robotics and mechatronics<sup>5</sup>

Wie auf der Abbildung (siehe Abb. 1) zu erkennen ist besteht ein Roboter aus 3 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 Werkzeug-module, welche meist Anwendung spezifisch wie etwa eine Sprühdüse für das auftrage von Lacken oder in Lichtbogen Schweißer sind und können bei Änderung der Anforderung schnell für ein anderes ausgetauscht werden.

<sup>3</sup> Vgl. Schlenoff u.a 2012 S 1

<sup>4</sup> Vgl. Schlenoff u.a 2012 S 3-4

<sup>5</sup> Mit Änderungen entnommen aus: Warnecke u.a. 1999 S 44



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<sup>6</sup>, da die Kinetische Energie von solch einem Roboterarm schwere körperliche Schäden verursacht, wenn eine Person in der Gefahrenzone von 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 achten, weder innerhalb des Fabrikationsprozesses, noch um die Kollision mit Fremdkörpern zu vermeiden<sup>7</sup>. 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 vermeiden. Doch mit dem Aufkommen neuer und besserer Sensortechnik müssen sich Roboter neuerdings nicht mehr nur auf repetitive Arbeiten fernab von menschlichen Arbeitern fokussieren, sondern sind zum einen durch bessere Echtzeit Videoverarbeitung dazu in der Lage dynamischer mit den zu bearbeitenden Objekten interagieren (z.B. Operationen relativ zur Objektorientierung)<sup>8</sup>, aber 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 10218-1<sup>9</sup>, -2<sup>10</sup> Norm formalisiert. Diese Normiert 4 verschiedene Betriebsmodi für kollaborative Roboter (Siehe Abb. 1).

---

<sup>6</sup> Vgl. Clark/Letho 1999 S 735

<sup>7</sup> Vgl. Asfahl 1999 S 245

<sup>8</sup> Vgl. Asfahl 1999 S 246

<sup>9</sup> Vgl. ISO 2006 10218-1

<sup>10</sup>Vgl. ISO 2008 10218-2

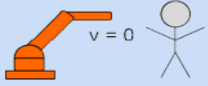
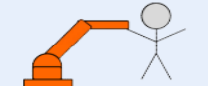
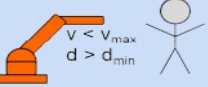
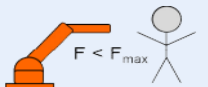
ISO 10218-1, Abschnitt	Arten des kollaborierenden Betriebs	Prinzip der Risikominderung	
5.10.2	Sicherheitsgerichteter überwachter Stopp (Beispiel: manuelle Einlegestation)	Roboter steht still wenn Bediener im Kollaborationsraum	
5.10.3	Handführung (Beispiel: Betrieb als Handlings-Gerät)	Roboter bewegt sich nur bei direkter Befehlseingabe durch Bediener	
5.10.4	Geschwindigkeits- und Positionsüberwachung (Beispiel: manuelle Teilebereitstellung)	Roboter bewegt sich nur wenn Abstand > minimaler Abstand	
5.10.5	Leistungs- und Kraftbegrenzung (Beispiel: kollaborierende Montage)	Roboter kann nur begrenzte transiente oder quasi-statische Krafteinwirkung ausüben	

Abb. 2 Kollaborierender Betrieb Arten Gemäß ISO 10218-1.-2 und ISO/TS 15066<sup>11</sup>

## 2.2 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 ermöglicht die besten Eigenschaften von etwas darzustellen definiert<sup>12</sup>. 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<sup>13</sup>. Ein wichtiger Teil davon ist die Partizipation der Interessenten, denn eine Demonstration bei der Kunden teilhaben können und nicht nur passive Betrachter des Geschehens oder Zuhörer eines Monologes sind, hinterlässt einen tieferen Eindruck<sup>14</sup>. Ein noch tieferer Eindruck kann hinterlassen werden, wenn bei den Interessenten die Erfahrungen mit Emotionen gekoppelt werden können, sprich 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<sup>15</sup>.

Dabei dienen Exponate als zentraler Punkt der Interaktion, denn zum einen können sie als Publikums Magnet funktionieren, bieten aber zum anderen auch die Möglichkeit, für das Personal, mit dem Publikum in kontakt zu treten. So kann ein Exponat als ein Anreiz fuer einen

<sup>11</sup> Mit Änderungen entnommen aus: Bjoern 2015 S 6

<sup>12</sup> Vgl Cambridge Dictionary. o.J.

<sup>13</sup> Vgl Thinius/Untiedt 2017 S 67-68

<sup>14</sup> Vgl Thinius/Untiedt 2017 S 157

<sup>15</sup> Vgl. Duncan/Moriarty 1998 S 8

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 Implementation des Exponates erklärt.

### 2.3 Computer Vision

Computervision jegliche beschreibt algorithmische Erfassung, Analyse, Verarbeitung und Interpretation von visuellen Daten<sup>16</sup>. In dieser Arbeit wird der Begriff der Computervision vor allem im Kontext der Objekt Erkennung 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 sucht versucht Objekte zu erkennen. Die Zweite Methode basiert auf dem Konzept des Maschine Learning, also automatischen trainieren von neuronalen Netzten<sup>17</sup>. Diese neuronale Netzten sind an der Struktur von Gehirnen angelehnt und bestehen aus einer Vielzahl an miteinander verbundenen Knotenpunkten. Im Gegensatz zu der Feature basierten Objekt Erkennung werden die Erkennung Algorithmen jedoch nicht manuell programmiert, sondern das neuronale Netz wird anhand von Trainingsdaten an die Problemstellung angelernt.<sup>18</sup> Die Struktur der Neuronalen Netzwerke ermöglich es ihnen komplexe Zusammenhänge zu abstrahieren und daraus Schlüsse zu ziehen.

---

<sup>16</sup>Vgl. Priese 2015 S. V

<sup>17</sup>Vgl. Zhao u.a 2019. S. 1

<sup>18</sup>Vgl. Zhao u.a. 2019 S. 2

### 3 Prototyping nach Cole et. al.

In der Methodologie nach Cole et. al. besteht das Prototyping aus vier unterschiedlichen Phasen: Der Problem Identifikation, der Intervention, der Evaluation und schließlich der Reflexion und des Bewusstmachens der gewonnenen Erkenntnisse, also dem „Learning“. Während der Identifikationsphase werden zwei Kernaspekte im Besonderen betrachtet.<sup>19</sup> Zum einen die Problematik als solche und zum anderen die Interessen derer Personen, welche sich mit der Problemlösung beschäftigen, um die tatsächliche praktische Relevanz für alle Involvierten besser zu erfassen. Die zweite Phase, die sogenannte Intervention, beschäftigt sich mit dem konstruieren eines Artefaktes, welches dazu dienen soll, die beschriebene Problematik zu adressieren und zu lösen, gleichzeitig beinhaltet es aber auch die Inbezugnahme des Umfeldes, welches auch entsprechend angepasst werden kann, um das Problem zu beheben. Darauf folgt die Evaluation der durchgeführten Intervention. Also eine Einschätzung, inwiefern die diese gesetzten Ziele erfüllen konnten und welchen Effekt die Intervention auf das Umfeld hatte.

---

<sup>19</sup> Dresch/Lacerda/Antunes 2014, S82

## 4 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, denn zum einen soll der Roboter 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 entstehen 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<sup>20</sup>.

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 kann, 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 schließlich für diesen Zweck entwickelt wurde. Jedoch hat auch die erfolgreiche Implementierung eines Computervision Systems seine eigenen Problemstellungen, da es in erster Linie schnell und zuverlässig funktionieren muss.

---

<sup>20</sup> Vgl. Saad/Neerincx/Hindriks (2019) S 1-2

Zusätzlich muss das Gesamtsystem leicht zu warten und weiterentwickeln sein, da voraussichtlich zukünftige Studenten Projekte darauf aufbauen werden. Weshalb bei der Auswahl der einzusetzenden Softwarebausteine auch Wert darauf gelegt werden sollte, dass diese leicht zu verwenden und weit verbreitet sind, um die Einarbeitungszeit für zukünftige Studenten zu minimieren.

## 5 Intervention

Die Problematik, dass jeder Durchlauf der Bewegungsroutine des Roboters einen Manuelle Intervention benötigt, soll nun auf zwei weisen 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 man 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 dritten System die Möglichkeit gibt den Roboter zu starten, um das Personal was jedes Mal benötigt wird, um den Roboter zu starten durch ein Computer System zu ersetzen.

Alternativ hätte er Roboter auch in einen dauer-durchlauf-modus 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 mit der Kinect so 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 ermöglicht und als zentraler Wartungspunkt dient.

### 5.1 ABB Yumi IRB14000

Bei dem ABB Yumi IRB14000 (Yumi) handelt es sich um einen Industriellen Roboter, welcher spezifisch für die Kollaboration mit Menschen entwickelt wurde.<sup>21</sup> 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. 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.<sup>22</sup>

---

<sup>21</sup> Vgl. ABB 2015 S 37

<sup>22</sup> Vgl. ABB 2015 S 34

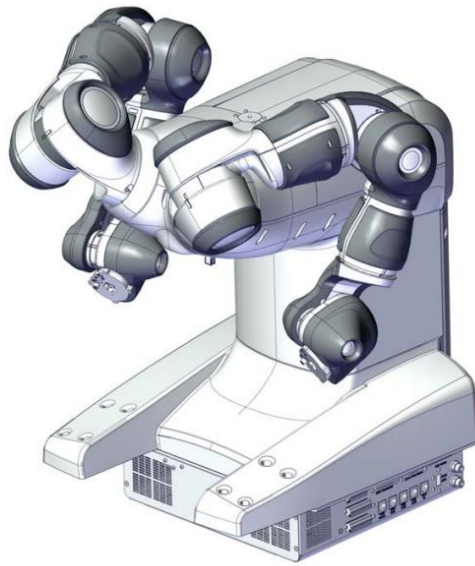


Abb. 3 ABB Yumi<sup>23</sup>

### 5.1.1 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 Element bei der Programmierung ist das Objekt mit dem Namen *robtarget*, welches den Zustand eines Armes genau beschreibt. Das Objekt definiert genau den Winkel von jeder Aktuator 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 er 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.

<sup>23</sup> Entnommen aus ABB 2015 S 57



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. 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 (siehe Abb 4 ).

Diese virtualisierte Version des Controllers kann dann bei Bedarf mit einem realen Controller synchronisiert werden. Dabei wird der gesamte code eines Controllers mit dem des anderen überschrieben. Diese Synchronisation ist in beide Richtungen möglich. So kann der Code beispielsweise 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 Programm-codes anhand des Integrierten Softwareentwicklung-toolset, welches fuer RAPID konzipiert ist.

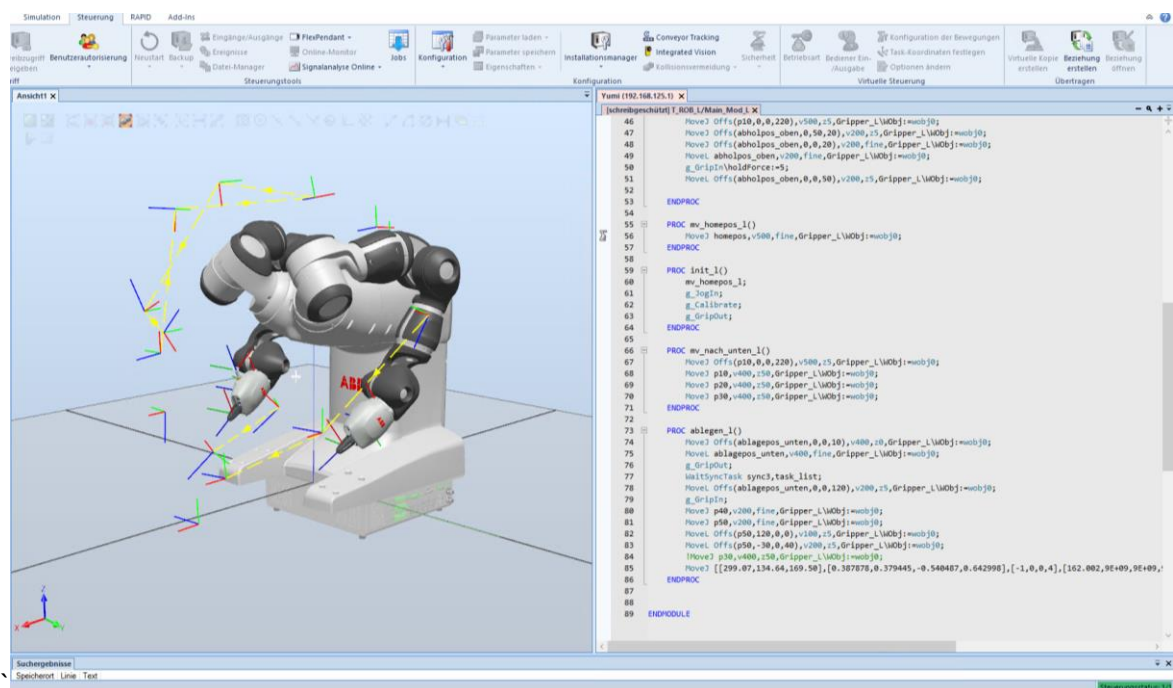


Abb. 4 Screenshot Robotstudio

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. Wenn der Roboter dann so bewegt wurde, dass er sich an der wichtigen stelle befindet, kann diese Position als robtarget abspeichert und weiterverwendet werden.

## 5.2 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.

### 5.2.1 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 nutzer-eingaben via Motion Captur 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<sup>24</sup>. Diese Werte können dann von Computersystemen, wie der Xbox oder einem PC entgegengenommen werden und anhand von verschiedenen Mustererkennungsalgorithmen analysiert werden. In diesem Anwendung Fall wurde die Kinect via des Proprietären Adapterstecker von Microsoft and Besagten NUC8i7HN angeschlossen auf welchem die im folgenden beschriebene Datenverarbeitung stattfindet.

### 5.2.2 pylibfreenect2

Die direkte Kommunikation mit der Kinect fand mittels des OpenSource Treiber Libfreenect2 und dem entsprechenden wrapper für die Programmiersprache Python, namens pylibfreenect2<sup>25</sup>, statt. 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, namens „Kinect for Windows SDK 2.0“ betrachtet, doch da dieses offiziell nicht mehr weiterentwickelt<sup>26</sup> wird und ausschließlich mit Windows verwendbar ist, besteht die Befürchtung, dass die aktuelle Kompatibilität mit einem Zukünftigen Windowsupdate verloren gehen könnte. 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 Opensource Lizenzierung jederzeit auf neue Gegebenheiten angepasst werden kann, sollte ein Kompatibilitäetsproblem entstehen. Obwohl der Funktionsumfang der „Kinect for Windows SDK 2.0“ beachtlich grösser ist, wurde deshalb die

---

<sup>24</sup> Vgl. Frankhauser u.a. S 2-3

<sup>25</sup> Xiang/Echtler/Kerl 2016

<sup>26</sup> Vgl. Microsoft 2022

Libfreenect2 Bibliothek als mittel der Wahl ausgewählt und die benötigten Features separat implementiert.

### 5.2.3 Numpy & OpenCV

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

OpenCV ist eine opensource Bibliothek für Python welche sich auf die Echtzeitverarbeitung von Bildern spezialisiert hat. In dieser Implementierung wird sie dazu genutzt Informationen, wie das aus Mediapipe gewonnenen Skelett, 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. Dieser Videostream wird gleichzeitig auch auf einem Monitor als Teil des Exponates gezeigt (siehe Abb. 5).

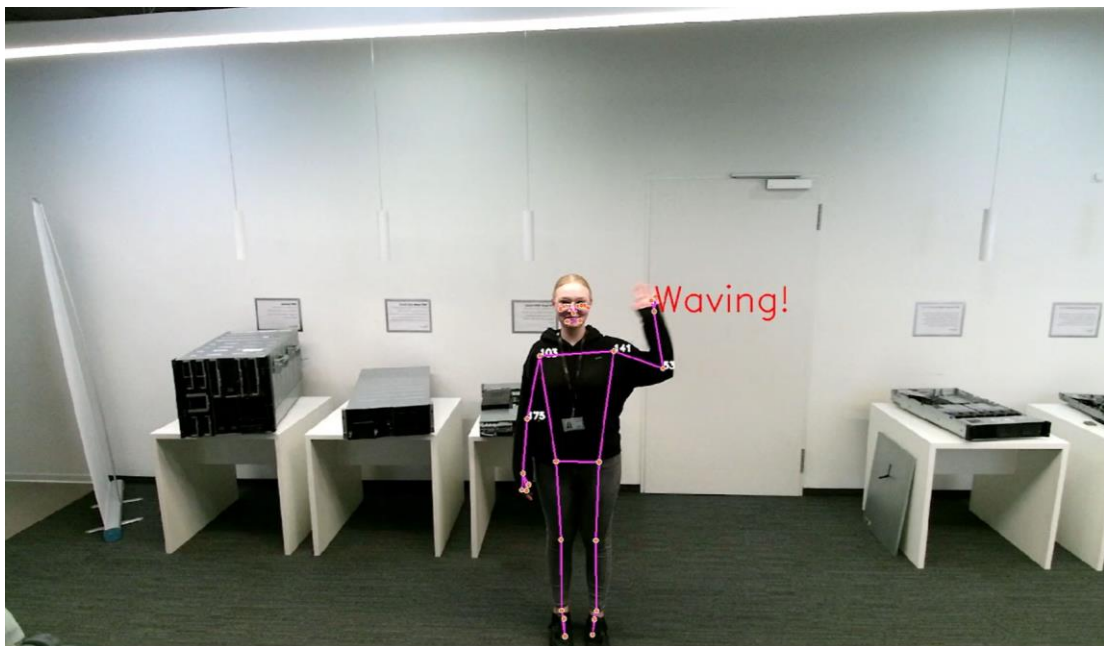
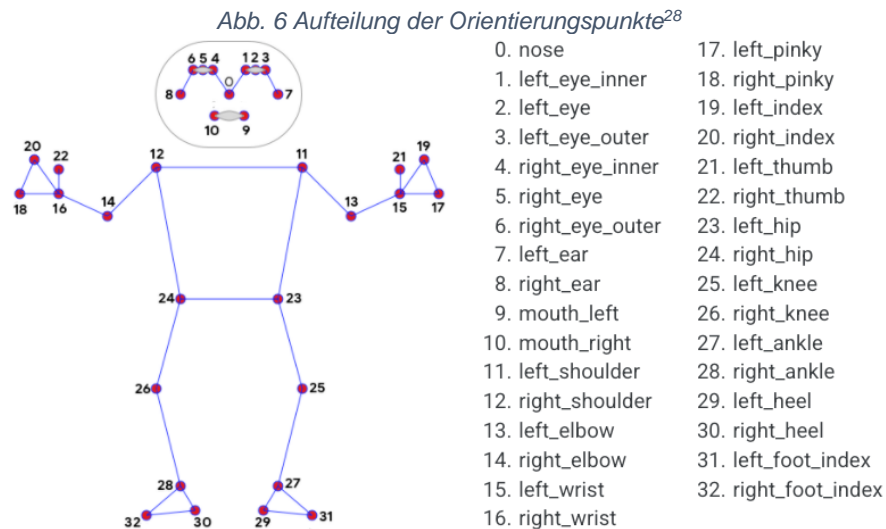


Abb. 5 Erkennen einer Geste während eines Testlaufs im CTC

### 5.2.4 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<sup>27</sup>, 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.(siehe Abb 6)



### 5.2.5 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 nicht, da die Konsequenzen einer erhöhten Fehlerrate bei einer Kundendenmonstration 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 Typen 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

<sup>27</sup> Vgl. Lugaresi/Tang/Nash 2019 S 1 ff.

<sup>28</sup> Entnommen aus Mediapipe o. J.

Wert daraus gelöscht, es seien der Oberarm und die gegenüberliegende Schulter haben einen Winkel kleiner oder gleich  $100^\circ$ , sollte dies der Fall sein, oder kann kein Arm gefunden werden, wird statt einem Winkel ein Nullwert in die Liste geschrieben. Dies geschieht, 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 Statuscode das Signal an den Kontrollserver gegeben, dass eine winkende Person erkannt wurde.

### **5.3 Kontrollserver**

Die Implementierung des Kontrollservers ist relativ einfach gehalten und dient hauptsächlich als zentraler eingangspunkt für Wartungs- und Entwicklungszwecke. 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. Dies wurde in form einer kleinen Flask API implementiert, welche auf Port 80 auf seiten des Visionsystems einen Webserver präsentiert. Die API nimmt 4 verschiedene Arten von Anfragen an, eine Statusabfrage, ein einfaches Startsignal, ein Stoppsignal und eine Anfrage zum Starten einer Endlosschleife.

## 6 Evaluation

### 6.1 Evaluierung der Programmierten Routine

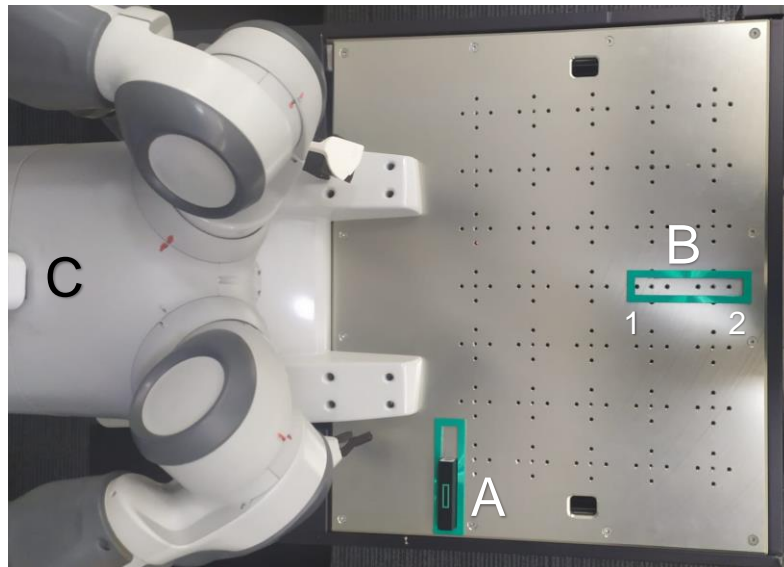


Abb. 7 Punkte in Einflussbereich des Yumis

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

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

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 dass er vorher mit der Bodenplatte kollidieren würde, deshalb wird er gezwungen die Powerbank im oberen Drittel zu greifen.

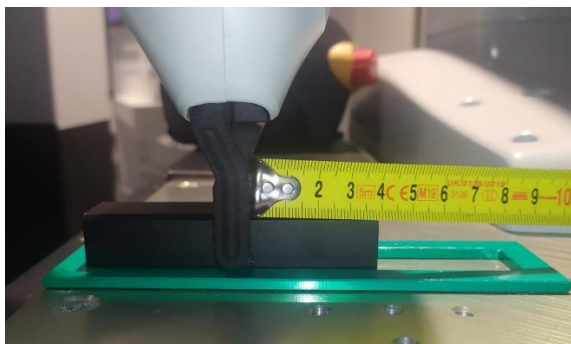


Abb. 8 Greifer in Abholposition



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 Fehlerzustände 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.

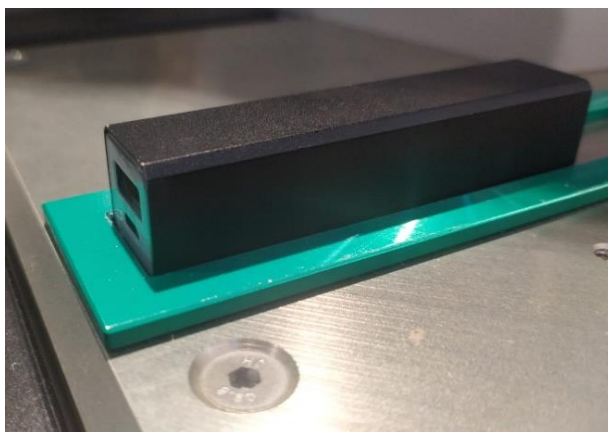
Denn erstens steht sie auf einer ihrer Seitenflächen während sie von Punkt zu Punkt geschoben. Gleichzeitig liegt aber auch der Schwerpunkt der Powerbank aufgrund von ungleicher Verteilung der internen Komponenten dem Mittelpunkt der Powerbank. Es könnte die Powerbank zwar auf die andere Seitenfläche 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 Entlade Buchsen 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 würde. 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 Senkenränder 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 Symbolen berührt und der dabei erzeugte Impuls höher ist als die von ABB zugelassenen werte, dies hat zur Folge, dass der Sensor im Motor den Notaus 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 Symbolen liegt, sondern stattdessen auf der Kante liegen bleibt, 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 Symbolen 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 Problem 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.

## **6.2 Interaktivität und Zuverlässigkeit der Demo**

Um die Demo Interaktiv zu machen wurde eine Gestenerkennung in Python auf der Basis von Numpy, OpenCV und vor allem 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 in Form der Winkel der Gliedmaßen zueinander. Basierend auf diesen Winkelangaben wird dann anhand von klassischen programmieren und Heuristiken festgestellt, ob die Person am Winken ist.

Um den Erfolg der Gestenerkennung zu analysieren, müssen zwei Metriken betrachtet werden. Zum einen auf die Fehlerrate, da diese die Zuverlässigkeit des Systems bestimmt. Zum einen durch 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 entsteht eine gewisse Latenz dadurch gegeben ist, dass die Erkennung des Winkens in form eines gleitenden Durchschnitts implementiert 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 selbst gesetzt, dies ist 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<sup>29</sup>.

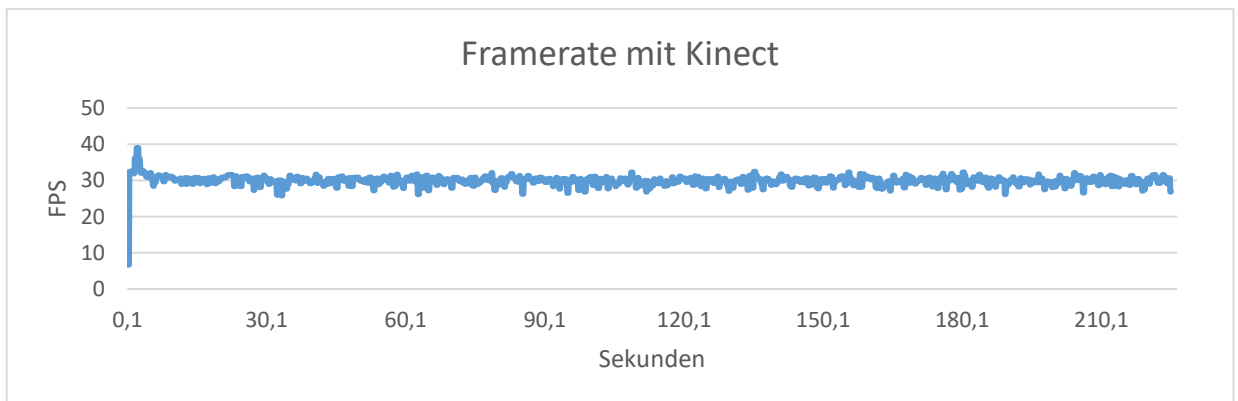


Abb. 12 Framerate mit Kinect

Die Anomalien in den ersten 3 Sekunden sind durch den Initialisierungsprozess 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 wurde zusätzlich auch getestet, wie sich der Anwendungscode verhält, wenn er nicht von der Kamera begrenzt ist. Um dies zu erreichen, wurde 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.

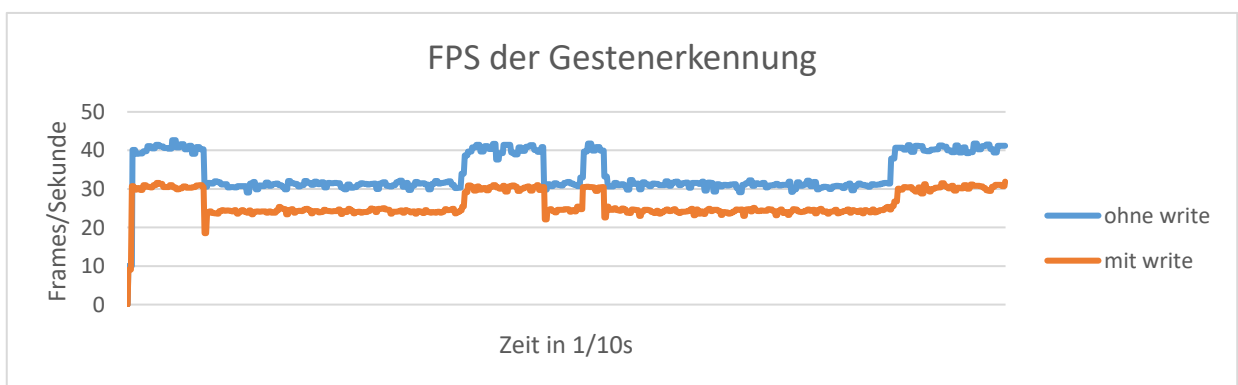


Abb. 13 FPS der Gestenerkennung

<sup>29</sup> Vgl. Lachat u. a. 2015

Diese beiden Graphen basieren auf demselben Video und in den Bereichen, in denen die Frame-Rate absackt, ist es deutlich zu erkennen, wann sich eine Person im Sichtfeld 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. Testverlauf, in dem die bearbeiteten Videodaten nur auf den Bildschirm gestreamt werden und dem, 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 vorlaufende Nummer des Frames, in denen 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.

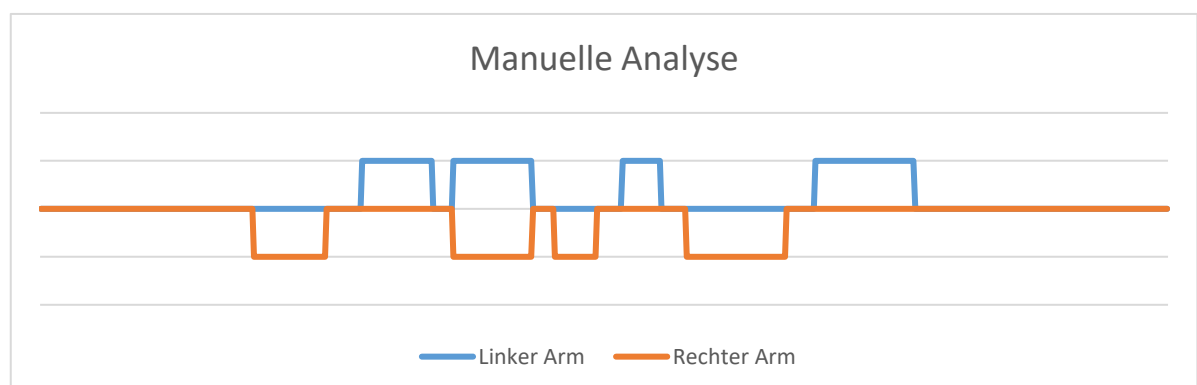


Abb. 14 Manuelle Analyse des Testvideos

Daraufhin wurde das selbe Video der Software-Implementierung zur Verfügung gestellt, welches 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 lässt sich erkennen, dass das System trotzdem Probleme damit hat ein Winken in seiner Ganzen Länge zu erkennen und in insgesamt 8 abschnitten 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.

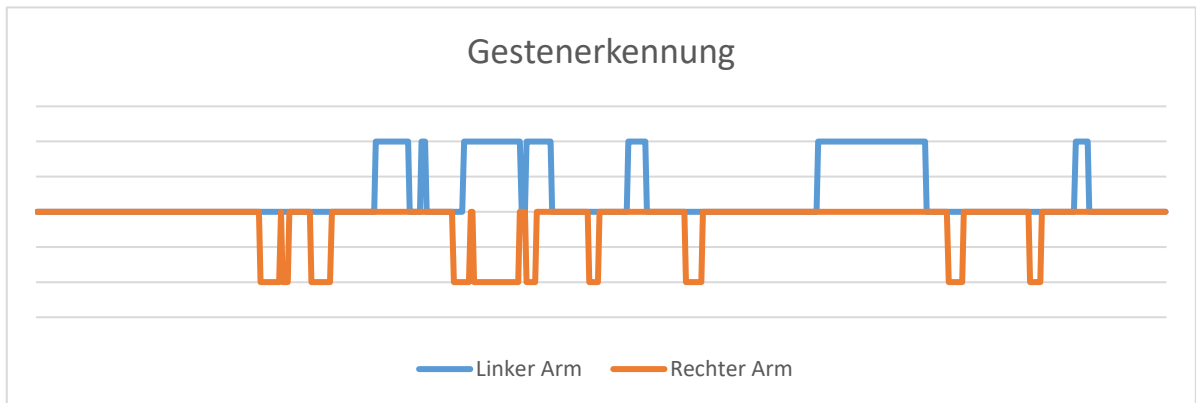


Abb. 15 Automatische Analyse des Testvideos

Die Latenz der Gestenerkennung hält sich jedoch dabei in Grenzen. Im Schnitt wird die Geste 2-3 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 keine Latenz in der Erkennung besteht.

Alles in allem ist die Lösung nicht perfekt jedoch adäquat für die Anforderungen. Weitere Schritte könnten hierbei die Entwicklung oder Implementierung eines spezifisch auf Winken trainierten Neuronalen Netzwerks sein.

### 6.3 Entwicklung 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, indem 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.

#### 6.4 Kinect

Das Ziel der Kinect war es eine zuverlässige Erkennung von Personen zu implementieren, um das manuelle Auslösen der Roboteroutine 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<sup>30</sup>.

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<sup>31</sup>. 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.

---

<sup>30</sup> Vgl. o. V. o.J. b

<sup>31</sup> Vgl. Hesse 2021 S. 77

## 6.5 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 Visionsystem 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 Entwicklung 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.<sup>32</sup>

---

<sup>32</sup> Vgl. Mak 2020 S. 6

## 7 Learning

Ziel dieses Projektes war die Entwicklung eines Prototyps auf Basis eines ABB YuMi Roboters, dieser sollte in der Lage sein bei Anwesenheit von Interessenten automatisch Routinen starten zu können, wofür eine Kinect V2 zur Verfügung gestellt wurde.

Zuerst wurden die zugrunde liegenden Konzepte und Technologien vorgestellt, welche in diesem Projekt von Relevanz waren. Also die Grundlagen der Robotik und der Kollaborativen Robotern, die Grundkonzepte von Showcases und die der Computervision.

Danach wurde die Methodik des Prototyping nach Cole et. Al. Erklärt und in den folgenden Schritten angewandt. Erst wurden die Problemstellungen und die relevanten Kriterien identifiziert, welche während des Intervention Teils adressiert wurden. Und schlussendlich wurde diese Intervention untersucht und reflektiert.

Das Artefakt, dass aus diesem Projekt hervorging wurde mittlerweile im Rahmen des CTCs mehrere Wochen erfolgreich eingesetzt. Für einen permanenten Einsatz fehlt nur noch eine Evaluation durch die Datenschutzbeauftragte, welche mit Zuversicht erwartet wird.

Von den im evaluationsteil festgestellten Problemen fällt in der Praxis hauptsächlich die Fehlerzustände des Roboters auf, diese sind zwar vergleichsweise selten, aber da sie erst manuell behoben werden müssen, bevor die Demonstration wieder funktionsfähig ist.

Deshalb sehe ich dort das höchste Potential für eine Verbesserungen des bestehenden Projektes. Für die Weiterentwicklung oder einem alternativen Projekt wäre der Bereich der Computervision von besonderem Interesse.

## 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 Dictionary. (o. J.) a:** Bedeutung von showcase im Englischen, <https://dictionary.cambridge.org/de/worterbuch/englisch/showcase>, abgerufen am 04.09.2022
- Thinus 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](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

## 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.

Boeblingen, 4.9.2022

(Ort, Datum)

---

(Unterschrift)