

Fachbereich VI - Informatik und Medien

# **Konzeption und prototypische Implementierung von innovativen Interaktionsmöglichkeiten mit Industriemaschinen in der virtuellen Realität im Kontext von Industrie 4.0**

Ibrahim Khaled Reguieg

Matrikelnummer 813812  
Studiengang Bachelor Medieninformatik  
6. Fachsemester  
Beuth Hochschule für Technik Berlin  
E-Mail: Khaled.Reguieg@gmail.com

*Betreuer*      Prof. Dr. Kristian Hildebrand  
Fachbereich VI - Informatik und Medien  
Beuth Hochschule für Technik Berlin

*Gutachter*      Prof. Dr. Felix Gers  
Fachbereich VI - Informatik und Medien  
Beuth Hochschule für Technik Berlin

28. Februar 2017

# Inhaltsverzeichnis

<b>1 Einführung</b>	<b>1</b>
1.1 Motivation und Kurzbeschreibung . . . . .	2
1.2 Aufbau . . . . .	3
<b>2 Grundlagen</b>	<b>5</b>
2.1 Industrie 4.0 . . . . .	5
2.1.1 Herausforderungen . . . . .	7
2.1.2 Effekte . . . . .	9
2.2 Robotik . . . . .	11
2.2.1 Überblick, Entwicklung und Forschungsstand . . . . .	11
2.2.2 Fertigungsroboter . . . . .	15
2.2.3 Lego Mindstorms EV3 . . . . .	17
2.3 Augmented Reality . . . . .	18
2.3.1 Überblick, Entwicklung und Forschungsstand . . . . .	19
2.3.2 Microsoft HoloLens . . . . .	26
2.3.3 Interaktionen in AR . . . . .	32
<b>3 Konzeption</b>	<b>38</b>
3.1 Spezifikation . . . . .	38
3.1.1 Zielsetzung . . . . .	39
3.1.2 Anwendungsfall . . . . .	39
3.1.3 Muss-Kriterien . . . . .	42
3.1.4 Kann-Kriterien . . . . .	43
3.1.5 Abgrenzungskriterien . . . . .	43
3.1.6 Nicht funktionale Kriterien . . . . .	44
3.1.7 Anwendungsumgebung . . . . .	45
3.1.8 Anwendungsdaten . . . . .	46
3.1.9 Anwendungsszenario . . . . .	47
3.2 Systementwurf und entwicklungsspezifische Technologien . .	48
3.2.1 Unity 5.5f . . . . .	48
3.2.2 HoloToolkit . . . . .	50
3.2.3 MonoBrick API . . . . .	51

<b>4 Prototypische Implementierung einer Augmented Reality Applikation zur Steuerung eines Roboters</b>	<b>53</b>
4.1 Kalibrierung der Szene . . . . .	53
4.2 Realisierung der Interaktionen . . . . .	54
4.2.1 User Interface . . . . .	54
4.2.2 Befehlsvermittlung an die Maschine . . . . .	55
4.2.3 Implementierung der Gesten . . . . .	58
4.2.4 Implementierung der Sprachkommandos . . . . .	62
<b>5 Usability Test</b>	<b>65</b>
5.1 Think Aloud Test . . . . .	65
5.2 Auswertung . . . . .	66
<b>6 Fazit und Ausblick</b>	<b>69</b>
<b>A Anlagen</b>	<b>70</b>
A.1 GazeManager.cs . . . . .	70
A.2 CursorManager.cs . . . . .	70
A.3 InteractableManager.cs . . . . .	72
A.4 GestureAction.cs . . . . .	73
A.5 VoiceManager.cs . . . . .	75
A.6 EV3Manager.cs . . . . .	76
A.7 Usability Test Transskript . . . . .	78
<b>Literatur</b>	<b>81</b>

# Abbildungsverzeichnis

1	Datenbrille in »Batman of the Future«. Bildquelle: [Vim]. . . . .	3
2	Verlauf Industrieller Revolutionen ab 1800 bis heute. Bildquelle: [Xin]. . . . .	6
3	Durchschnittliche Arbeitskosten pro Stunde in Europa, 2015. Bildquelle: [eur]. . . . .	7
4	Bevölkerungsentwicklung in Deutschland von 1950-2050. Bildquelle: [bpb]. . . . .	8
5	Skizze von Heron von Alexandria »Automata«. Bildquelle: [Wikc].	12
6	Die Droiden von Jaquet Droz und seinem Sohn: Der Zeichner (links) und seine Zeichnung (links unten), Der Schreiber (mitte) und dessen Text (rechts unten) und die Musikerin (rechts). Bildquelle: [Wikd], [Wikf], [his], [Wike], [Mer]. . . . .	12
7	Patentzeichnung für den ersten Industrieroboter. Bildquelle: [Dev].	13
8	Unterteilung von Handhabungsgeräten in Untergruppen. Bildquelle: [Lin16]. . . . .	13
9	Schematische Darstellung eines Industrieroboters. Bildquelle: [Lin16]. . . . .	14
10	Fließbandarbeiter um 1913 in der Automobilfabrikation von Henry Ford. Bildquelle: [Run]. . . . .	15
11	Industrieroboter in einem Daimler-Werk. Bildquelle: [Dai]. . . . .	16
12	Hauptkomponente des <i>Lego Mindstorms</i> der EV3-Stein. Bildquelle: [Leg]. . . . .	17
13	<i>Reality-Virtuality Continuum</i> von Paul Milgram und Fumio Kishino. Bildquelle: [res]. . . . .	19
14	Charles Wheatstones Spiegelstereoskop. Bildquelle: [kin]. . . . .	20
15	David Brewsters Linsenstereoskop. Bildquelle: [Wika]. . . . .	20
16	Albert Pratts Patent für eine am Kopf montierte Ziel- und Schießeinrichtung, 1915. Bildquelle: [pra]. . . . .	21
17	Morton Heiligs Erfindung das »Sensorama« zeigt Filmerlebnisse, die auch andere Sinne ansprechen. Bildquelle: [mor]. . . . .	22

18	Ivan Sutherlands Erfindung »Sword of Damocles«, 1968. Bildquelle: [aug]. . . . .	22
19	HMD(links) und augmentierte Hilfestellung(rechts) des Projekts »KARMA«. Bildquelle: [col]. . . . .	23
20	Schaubild zur Epipolargeometrie. Bildquelle: [Wikb]. . . . .	24
21	Spielfigur macht Bewegung des Spielers durch Kameratracking im Spiel <i>EyeToy3</i> nach. Bildquelle: [Pla]. . . . .	25
22	Gartners Hypecycle Diagramm für das Jahr 2016. Bildquelle: [Gar]. . . . .	26
23	<i>Microsofts</i> Datenbrille, die <i>HoloLens</i> . Bildquelle: [Micb]. . . . .	27
24	Von der <i>HoloLens</i> erfasst: Schreibtisch und Wände als 3D-Objekt. Bildquelle: Eigenes Werk. . . . .	29
25	Von der <i>HoloLens</i> erfasste Räume aus der Außenansicht als 3D-Objekt. Bildquelle: Eigenes Werk. . . . .	29
26	<i>HoloLens – Mesh</i> des erfassten Raumes in mehreren Stufen. Bildquelle: Eigenes Werk. . . . .	30
27	<i>Microsoft HoloLens</i> in dunklem Raum. Bildquelle: Eigenes Werk. . . . .	31
28	<i>Microsoft HoloLens</i> Sensorenleiste. Bildquelle: [Micc]. . . . .	31
29	Infrarot Leuchten der <i>Microsoft HoloLens in Betrieb</i> . Bildquelle: Eigenes Werk. . . . .	32
30	Erkennungsbereich für Gesten der Frontkameras. Bildquelle: [Mica]. . . . .	33
31	<i>Prepare Bloom Gesture (Fist)</i> (a), <i>Perform Bloom Gesture</i> (b). Bildquelle: Eigenes Werk. . . . .	34
32	Die <i>Air-Tap</i> -Geste. Initiiert mit der <i>Gesture-Ready</i> -Geste (a), ausgeführt mit dem <i>Tap</i> selbst (b) und Endposition erneute <i>Gesture-Ready</i> -Geste (c). Bildquelle: Eigenes Werk. . . . .	34
33	<i>Scroll</i> -Geste in X- und Z-Richtung ( <i>Tap-And-Hold + Translation</i> ). Bildquelle: Eigenes Werk. . . . .	35
34	<i>Voice-Command</i> : Am Beispiel »Grab«. Bildquelle: Eigenes Werk. . . . .	36
35	Informationsfluss in der Applikation <i>EV3ControllAR</i> . Bildquelle: Eigenes Werk. . . . .	38
36	UML Use-Case-Diagramm – CommandRobot. Bildquelle: Eigenes Werk. . . . .	39
37	Versuchsaufbau für die Applikation <i>EV3ControllAR</i> . Bildquelle: Eigenes Werk. . . . .	46
38	<i>Unity</i> Multiplattform-Unterstützung – Übersicht der unterstützten Plattformen. Bildquelle: [unity]. . . . .	48

39	<i>Unity</i> Entwicklungsumgebung – Übersicht des Editors. Bildquelle: Eigenes Werk.	49
40	HoloInterface – <i>Userinterface</i> als <i>Wireframe</i> Entwurf. Bildquelle: Eigenes Werk.	54
41	HoloInterface – <i>User Interface</i> für die Applikation »EV3ControllAR«. Bildquelle: Eigenes Werk.	55
42	UDP-Paket des <i>EV3</i> im Netzwerk, welches die Seriennummer enthält. Bildquelle: Eigenes Werk.	56

# **Tabellenverzeichnis**

3.1	Anwendungsfall 1 – »Roboter bewegen«. . . . .	40
3.2	Anwendungsfall 2 – »Objekt aufnehmen«. . . . .	41
3.3	Anwendungsfall 3 – »Objekt platzieren«. . . . .	41
3.4	Anwendungsfall 4 – »Notstopp«. . . . .	42
5.1	Testperson 1 – Büromanagement, männlich. . . . .	66
5.2	Testperson 2 – Büromanagement, weiblich. . . . .	66
5.3	Testperson 3 – Gestaltung, männlich. . . . .	66
5.4	Testperson 4 – Gestaltung, weiblich. . . . .	67
5.5	Testperson 5 – Projektmanagement, männlich. . . . .	67
5.6	Testperson 6 – Projektmanagement, weiblich. . . . .	67
5.7	Testperson 7 – Entwicklung, weiblich. . . . .	68
5.8	Testperson 8 – Entwicklung, männlich. . . . .	68

# Einführung

Die rasanten Entwicklungen der Technologisierung und Digitalisierung haben branchenübergreifend einen Paradigmenwechsel eingeläutet. Die Konsequenz: Geräte wie beispielsweise Smartphones sind rund um die Uhr in sämtlichen Lebens- und Arbeitsbereichen aktiv, online und miteinander vernetzt. Digitale Geräte finden Einsatz im täglichen Leben und Arbeiten und so steht die Gesellschaft derzeit an der Schwelle zur vierten Industriellen Revolution.

Damit einhergehend dringen neue Technologien, wie *Virtual Reality* (VR) und *Augmented Reality* (AR) in die globalen Märkte ein und beginnen die Entertainmentindustrie und die Arbeitswelt nachhaltig zu verändern. Durch die zunehmende Medienkompetenz und Offenheit der Gesellschaft ist die Bereitschaft vorhanden, neue Technologien zu nutzen. Es besteht daher ein hohes Interesse daran, geeignete Interaktionsmöglichkeiten für Aktionen im virtuellen Raum zu finden und zu erproben. Das folgende Szenario verdeutlicht den Sachverhalt anhand einer Baumaschine und eines Maschinenführers im augmentierten Raum und stellt einen Rahmen für die prototypische Implementierung dar.

**Szenario der Anwendung** Ein Maschinenführer befähigt eine Baumaschine mit Hilfe der Standard-Gesten der *Microsoft HoloLens*, schwere reale Objekte von einer Position A zu Standort B zu befördern. Maschinen können heutzutage ohne weiteres Befehle von außen empfangen. Somit könnte der Maschinenführer die Maschine bequem und sicher in einer in 3D abstrahierten Replikation der Baustelle, die er in seiner Datenbrille sieht, bedienen. Er interagiert dann mit virtuellen Replikationen statt mit den echten Objekten auf der Baustelle. Die Interaktion mit den virtuellen Objekten wird über eine Netzwerkverbindung auf den Roboter übertragen, welcher dann das echte Objekt auf der realen Baustelle bewegt. Als Einsatzorte bieten sich menschenfeindliche Arbeitsumgebungen an, beispielsweise die Atmosphäre eines anderen Planeten, verseuchte oder unwegsame Gebiete der Erde oder das All. In einer Arbeitsumgebung, in der nur Maschinen anwesend sind, kann kein Mensch zu Schaden kommen. Die Arbeiter wären keinen Witterungen oder gesundheitsgefährdenden Arbeitsumfeldern ausgesetzt. Resultat dessen

wäre zwangsläufig weniger körperliche Erschöpfung, was wiederum weniger krankheitsbedingte Ausfälle der Maschinenführer bedeutet. Die Interaktionsmöglichkeit mit den Maschinen wäre stets gegeben, so bleibt der menschliche Faktor, der alle Prozesse individuell der Situation anpassen kann, weiterhin gegeben und die Baustelle weiterhin wirtschaftlich.

**Fragestellung** Aus diesem Szenario resultieren verschiedene Fragestellungen. Allen voran die Frage danach, wie der Mensch »Herr der Maschinen« bleiben kann. Welche Technologien eignen sich dazu, die Maschinenprozesse zu überwachen und zu beobachten? Und welche Interaktionen eignen sich am besten, einer Maschine Befehle zu erteilen, sowohl als Teleoperator als auch in enger räumlicher Zusammenarbeit mit der Maschine?

Um diese Fragen beantworten zu können, werden im Rahmen der vorliegenden Arbeit ein Konzept und eine prototypische Implementierung einer Industriemaschine im virtuellen Raum erstellt. Die Applikation wird mit innovativen Interaktionen bestückt und die Anwendung im Kontext der Industrie 4.0 diskutiert.

## 1.1 Motivation und Kurzbeschreibung

Innovative Ideen, Konzepte und Produkte zeichnen die Informatik und IT-Branche aus wie keine andere. So gilt es stets, neue Technologien zu erforschen, zu erlernen und frühzeitig Anwendungen und Konzepte auszuarbeiten, welche die Gesellschaft nachhaltig verändern können. Fast kein Industriezweig bleibt unberührt von diesen neuartigen, agilen Methoden und Konzepten. Datenbrillen, seien es nun VR oder AR *Head-Mounted Display* (HMD) sind in ihrer *Consumerversion* beziehungsweise als *Development-Kit* auf den Märkten verfügbar. Noch sind sie zwar teuer, allerdings haben erste Hersteller bereits Preissenkungen angekündigt. Steigende Verkaufszahlen deuten darauf hin, dass es der Traum vieler Menschen ist, eine Datenbrille zu besitzen, sei es zu Entertainmentzwecken oder, um Vorteile im Alltag und im Arbeitsleben zu genießen. Was im Comic-Band »Batman of the Future« noch futuristisch und utopisch klang, nämlich einen Roboter mit einer Datenbrille und Gesten fernzusteuern, ist heute Gegenstand zahlreicher Forschungen, wie auch dieser Arbeit. In einer Episode des Bandes benutzt einer der Akteure einen riesigen Roboter als Baumaschine. Er steuert ihn über eine Datenbrille

und Gesten, wie in Abbildung 1 zu sehen. Um die Gesellschaft auf künftige



**Abb. 1.:** Datenbrille in »Batman of the Future«. Bildquelle: [Vim].

Technologien vorzubereiten bedarf es Anwendungen, die diese Neuerungen erforschen, in ansehnlichen Szenarien darstellen und mit gutem Design und vorausschauenden Konzepten die Berührungsängste mit ihnen minimieren.

## 1.2 Aufbau

Ziel der vorliegenden Arbeit soll es sein, mit Hilfe eines Prototyps einen Ausblick darauf zu geben, wie Menschen in Zukunft mit Maschinen interagieren, was dabei zu beachten ist und welche Interaktionsformen sich für industrielle Zwecke besonders eignen. Im Anschluss an eine ausführliche Definition der grundlegenden Begrifflichkeiten im Kapitel 2 erfolgt eine Analyse des aktuellen Forschungsstandes sowie eine Erörterung der Plausibilität der Fragestellung dieser Arbeit. Unter anderem wird in diesem Zuge auch erläutert, weshalb *Microsofts HoloLens* und *Legos EV3* geeignete Geräte für die prototypische Implementierung sind. Im Anschluss werden die Relevanz und die Potentiale der Technologien Robotik, Datenbrillen und AR für Forschungseinrichtungen, Industrieunternehmen und Technik-Enthusiasten erläutert und außerdem darauf eingegangen, inwieweit die »Industrie 4.0« als ganzheitliches Konzept dahinter steht.

Danach wird in Kapitel 3 ein Konzept erarbeitet und präsentiert, welches sich damit auseinandersetzt, wie über eine Datenbrille in Form einer *Microsoft HoloLens* eine Maschine, im prototypischen Fall ein Roboter aus *Lego*, gesteuert und bei seiner Arbeit mit einem *User Interface* augmentiert wird. Besagtes Vorhaben gilt es in Kapiel 4 in einer Anwendung minimalisiert darzustellen und einen konzeptionellen Beweis dafür zu liefern, dass Interaktionen im virtuellen Raum auf die reale Welt übertragen werden können, einen Mehrwert bedeuten und dass sie im Kontext der vierten Industriellen Revolution eine Berechtigung besitzen. Abschließend wird diese These in Kapitel 5 mit

einer benutzerbasierten, qualitativen Evaluation, dem *Thinking-Aloud* Test, überprüft und dessen Ergebnisse ausgewertet.

# Grundlagen

In diesem Kapitel wird erläutert, was der Begriff »Industrie 4.0« bedeutet, vor welche Herausforderungen sie uns stellt und welche Effekte mit ihr einhergehen. Des Weiteren behandelt dieses Kapitel grundlegende Sachverhalte bezüglich Fertigungsrobotern und dem hier als Substitution benutzten *Lego Mindstorms EV3* sowie Grundlagen zu AR und der genutzten Brille *Microsoft HoloLens*.

## 2.1 Industrie 4.0

Eine »Industrielle Revolution« bezeichnet »einen raschen Wandel von Produktionstechniken und, daraus abgeleitet, von wirtschaftlich-gesellschaftlichen Strukturen.<sup>1</sup> Die »Industrie 4.0« bezeichnet das Ergebnis der vierten *Industriellen Revolution*. Sie reiht sich an der Spitze von drei Vorgängern ein:

- Die erste *Industrielle Revolution*, etwa um 1750, bei der Arbeits- und Kraftmaschinen die Industrialisierung und das Verhindern von Hungerkatastrophen ermöglichte.
- Die zweite *Industrielle Revolution*, etwa um 1870, die Wohlstand durch arbeitsteilige Massenproduktion mit Hilfe elektrischer Energie schaffte.
- Und die dritte *Industrielle Revolution*, seit circa 1960, welche durch Elektronik und IT automatisierungsgetriebene Rationalisierung sowie die variantenreiche Serienproduktion ermöglichte.<sup>2</sup>

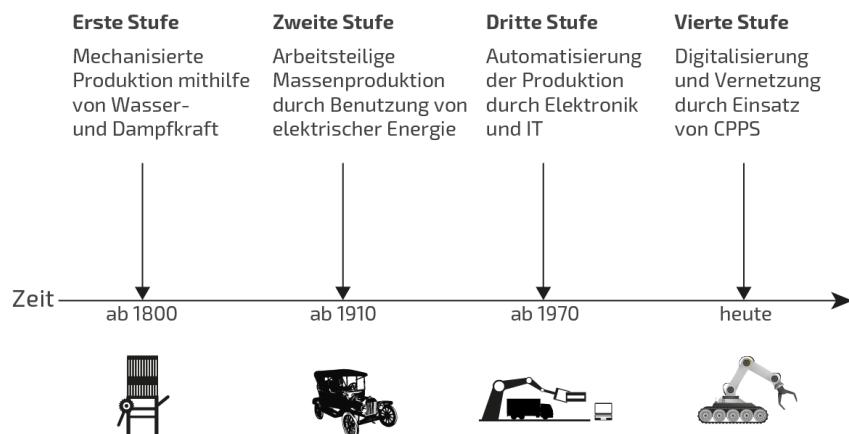
Diese Entwicklung ist zusätzlich schematisch in Abbildung 2 dargestellt. Der Begriff »Industrie 4.0« ist ursprünglich ein Projektname einer Arbeitsgruppe des Bundesministerium für Wirtschaft und Energie (BMWi). Es sind zahlreiche Beschreibungen zu finden, die sich sehr stark ähneln. Für die folgende Arbeit wird daher die Definition des BMWi beibehalten:

---

<sup>1</sup> Prof. Dr. Voigt, 2016, [Accessed: 06.01.2017].

<sup>2</sup> Vgl. Dipl.-Ing. Jäger et al, 2015.

## Stufen der industriellen Revolution



**Abb. 2.:** Verlauf Industrieller Revolutionen ab 1800 bis heute. Bildquelle: [Xin].

»In der Industrie 4.0 verzahnt sich die Produktion mit modernster Informations- und Kommunikationstechnik [...] (zu) intelligente(n) Fabriken (sogenannte „Smart Factories“) [...] Menschen, Maschinen, Anlagen, Logistik und Produkte kommunizieren und kooperieren in der Industrie 4.0 direkt miteinander. Produktions- und Logistikprozesse zwischen Unternehmen im selben Produktionsprozess werden intelligent miteinander verzahnt, um die Produktion noch effizienter und flexibler zu gestalten.

So können intelligente Wertschöpfungsketten entstehen, die zudem alle Phasen des Lebenszyklus des Produktes miteinschließen – von der Idee eines Produkts über die Entwicklung, Fertigung, Nutzung und Wartung bis hin zum Recycling. [...]«<sup>3</sup>

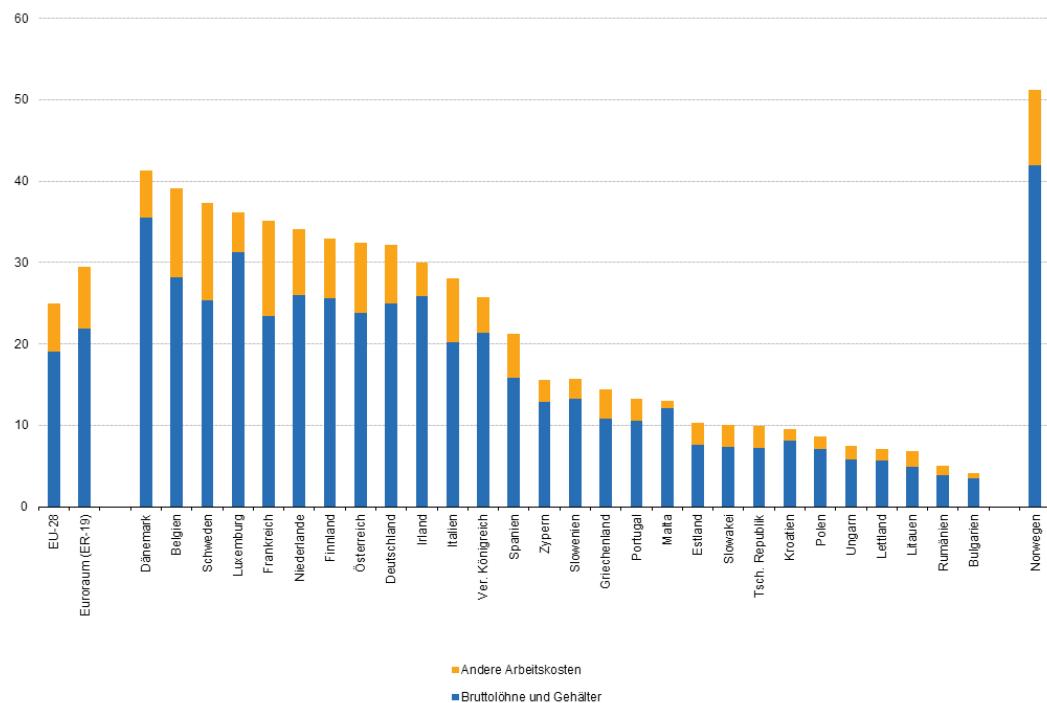
Momentan eher als Marketingbegriff benutzt, fasst der Ausdruck »Industrie 4.0« sehr gut zusammen, wie die Produktion in Deutschland, aber auch weltweit, in Zukunft aussehen kann. Ein hochgerechnetes Investitionsvolumen von jährlich 40 Mrd. Euro in den nächsten fünf Jahren zeigt deutlich, dass vernetzte und intelligente Produktionsstätten ein Hauptanliegen der deutschen Industrie und Wirtschaft darstellen<sup>4</sup>. Sie sind maßgeblich dafür verantwortlich, dass Deutschland seine Vorreiterrolle und sein Ansehen als Motor moderner, effizienter und qualitativ hochwertiger Industrieanlagen weltweit behaupten und weiter ausbauen kann.

<sup>3</sup> BMWi, Hrsg., 2016, [Accessed: 06.01.2017].

<sup>4</sup> Vgl. Dr. Geissbauer / Schrauf / Koch / et al. (2016), S.7.

## 2.1.1 Herausforderungen

Deutschland ist, wie in Abbildung 3 zu sehen, ein Hochlohnstandort. Eine maßgebliche Herausforderung ist daher: Die Produktion durch Innovationen profitabel im Inland zu betreiben, statt sie ins Ausland zu verlagern.



(\*) Unternehmen mit mindestens 10 Arbeitnehmern. NACE Rev. 2 Abschnitte B bis S ohne O. Einschließlich vorläufiger Daten.  
Quelle: Eurostat (Online-Datencode: lc\_lci\_lev)

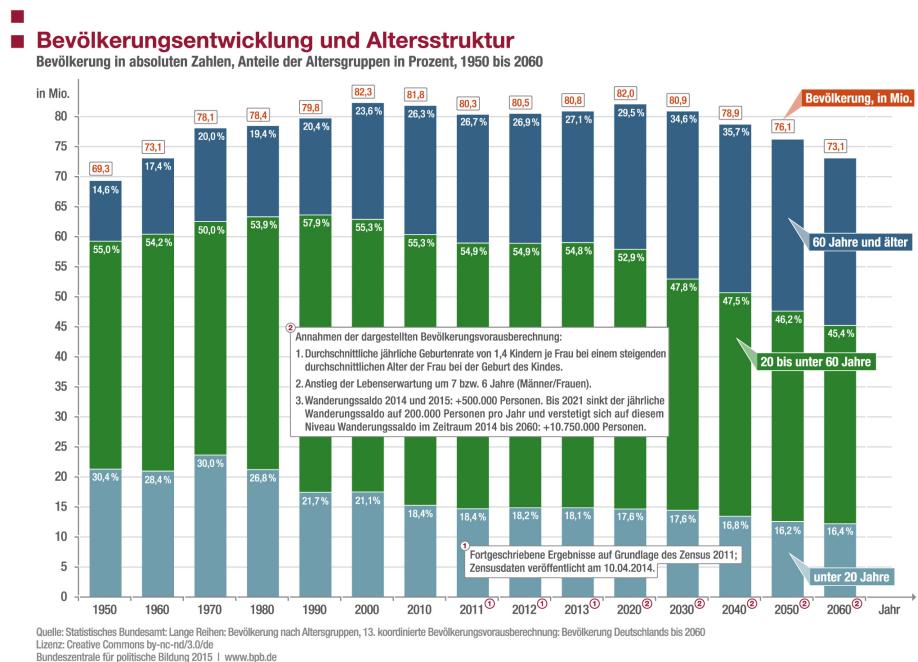
**Abb. 3.:** Durchschnittliche Arbeitskosten pro Stunde in Europa, 2015. Bildquelle: [eur].

Des Weiteren zählt die Bundesrepublik Deutschland zu einem der Länder, dessen Rohstoffverlangen bei weitem die Menge dessen übersteigt, welche im Inland abgebaut werden kann. Es bilden sich Abhängigkeiten, um die Rohstoffversorgung zu sichern. Beispielsweise importierte die Bundesrepublik im Jahr 2005 Waren mit dem Gesamtwert von 625 Mrd. Euro, darunter 26,5 Mio. Tonnen Steinkohle und 110 Mio. Tonnen Rohöl.<sup>5</sup> Um wettbewerbsfähig und unabhängig zu sein, müssen die Produktionsstätten der Zukunft ressourcensparend und vor allem energieeffizient produzieren. Hierzu gesellt sich auch das Schlagwort »Urbane Produktion«. Es bedeutet: „das Umfeld einer Produktionsstätte in den Bilanzierungsrahmen der energetischen Optimie-

<sup>5</sup> Topel / Seibel, Hrsg., o. J., [Accessed: 11.01.2017].

rung einzubeziehen“.<sup>6</sup> Dazu zählen sowohl Recycling als auch Abwärme und andere passive Rohstoffe.

Außerdem sehen sich Gesellschaften, die ein gewisses Maß an Wohlstand erreicht haben, mit einem demographischen Wandel konfrontiert.<sup>7</sup> In Ab-



**Abb. 4.:** Bevölkerungsentwicklung in Deutschland von 1950-2050. Bildquelle: [bpb].

bildung 4 ist die Bevölkerungsentwicklung in Deutschland von 1950 bis 2050 zu sehen. Abzuleiten daraus ist, dass der Anteil der Bevölkerung im arbeitsfähigen Alter kleiner, und der nicht arbeitsfähige Anteil<sup>8</sup> größer wird. Dieser Wandel spannt verschiedene Problemfelder auf, deren Lösung von verschiedenen Seiten gefordert wird. Welche Lösungsansätze durch die »4. Industrielle Revolution« entstehen, wird im Folgenden erläutert.

<sup>6</sup> Fraunhofer-Institut für Produktionstechnik und Automatisierung, Hrsg., o.J., [Accessed: 11.01.2017].

<sup>7</sup> Vgl. Roser / Ortiz-Ospina, 2017, [Accessed: 01.02.2017].

<sup>8</sup> Vgl. Statistisches Bundesamt, Bevölkerung nach Altersgruppen, Hrsg., [Accessed: 11.01.2017].

## 2.1.2 Effekte

Die aus den Herausforderungen resultierenden Effekte versucht die »Industrie 4.0« bewusst zu adressieren und anzugehen. Intelligente Fabriken können das Arbeiten am Hochlohnstandort wieder profitabel machen. Das BMWi beschreibt diesen Wandel wie folgt: „Industrie 4.0 bedeutet gute Arbeit, nicht menschenleere Fabriken. Vielmehr verändert sich das Arbeiten: Die Beschäftigten müssen stärker in die Prozesse eingebunden werden, zum Beispiel[sic] um Abläufe zu koordinieren, die Kommunikation zu steuern und eigenverantwortlich schnell Entscheidungen zu treffen“.<sup>9</sup>

Verzahnung von industrieller Produktion mit modernster Informations- und Kommunikationstechnik soll in hoher Individualität, Effizienz und Kostensparnis münden. Schon heute ermöglichen Innovationen, wie beispielsweise ein intelligenter Handschuh<sup>10</sup>, Zeitersparnisse und mehr Effizienz im Logistiksektor. Eine *Wearable*-Lösung wie diese begleitet einen Arbeiter auf dem Weg in die »Industrie 4.0«, ist jedoch eher in Logistik- und Zuliefererbetrieben anzusiedeln. Allerdings soll in Zukunft die Produktion durch *Cyber Physical Systems*<sup>11</sup> digitalisiert werden, damit dabei eine smarte Wertschöpfungskette entsteht, die ihren Energie- und Ressourcenbedarf optimal regulieren und ausschöpfen kann. Hinzukommend könnten Kreisläufe entstehen, die urbane Gegebenheiten, von Abwärme bis hin zu Recycling mit berücksichtigen und so die Produktionsprozesse noch umweltfreundlicher und effizienter gestalten.

Die Herausforderung einer alternden Gesellschaft bedeutet den Verlust von Arbeitskraft in menschlicher Form. Eine Industrielandschaft, in der Maschinen den Hauptteil der zu bewerkstelligenden Aufgaben erledigen können und nur ein vergleichsweise kleiner Teil der Gesellschaft durch fachliche Kompetenz diese Maschinen entwickeln und warten muss, kann einen solchen Wandel ausgleichen.

---

<sup>9</sup> BMWi, Hrsg., 2016, [Accessed: 09.01.2017].

<sup>10</sup> Handschuh der beispielsweise Produktions- und Logistikprozesse verbessert und vereinfacht.

<sup>11</sup> Cyber-Physische Systeme (CPS) sind Systeme mit eingebetteter Software und Elektronik, die über Sensoren und sogenannte Aktoren (Antriebselemente) mit der Außenwelt verbunden sind.

Diese Effekte durch gezielte Investitionen und Entwicklungen im Fachbereich der Sensorik und Robotik anzugehen, ist das Hauptanliegen der »Industrie 4.0«.

## 2.2 Robotik

Die interdisziplinäre Kategorie der Robotik bezeichnet ein Teilgebiet des Ingenieurwesens, welches sich aus den Hauptbestandteilen Maschinenbau, Elektrotechnik und Informatik zusammensetzt und dessen Wortursprung im slawischen Wort »roboṭa« liegt und soviel bedeutet wie »Arbeiten«.<sup>12</sup> 1920 wurde der Begriff vom tschechischen Schriftsteller Karel Čapeks in seinem Schauspiel »R. U. R.« (Rossums Universal-Roboter) im Sinne eines Maschinenmenschen erwähnt und dadurch geprägt. Die Robotik ist eine der Leitindustrien des 21. Jahrhunderts, woraus eine enge Verbindung zur Mensch-Maschine-Interaktion, Psychologie, Soziologie (soziale Robotik) und Philosophie (Maschinenethik) entsteht.<sup>13</sup>

### 2.2.1 Überblick, Entwicklung und Forschungsstand

Der Verband Deutscher Ingenieure (VDI) definiert in seiner *Richtlinie 2860* Roboter wie folgt: »Industrieroboter sind universell einsetzbare Bewegungskräfte mit mehreren Achsen, deren Bewegungen hinsichtlich Bewegungsfolge und Wegen bzw. Winkeln frei (d.h. ohne mechanischen bzw. menschlichen Eingriff) programmierbar und gegebenenfalls sensorgesteuert sind. Sie sind mit Greifern, Werkzeugen oder anderen Fertigungsmitteln ausrüstbar und können Handhabungs- und/oder Fertigungsaufgaben ausführen«.<sup>14</sup>

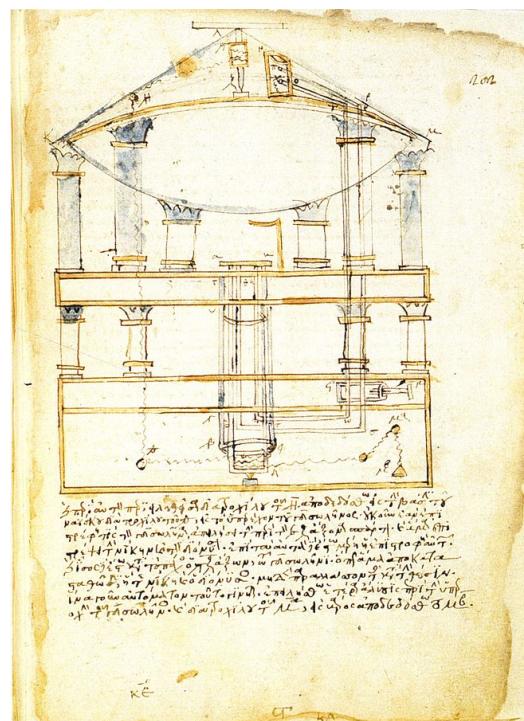
Angefangen mit Heron von Alexandria (Abb. 5) und seinen Entwürfen für ein Automatentheater um etwa 100 nach Christus, über die drei Droiden (Zeichner, Schriftsteller und Klavierspieler) aus Abbildung 6, gebaut 1774 von Jaquet Droz und dessen Sohn, bis zur Erfindung des Industrieroboters durch George Devol 1954 (Patentzeichnung in Abbildung 7) sind Automaten dafür erdacht worden, den Menschen Arbeiten abzunehmen und sie durch die Automatisierung von Arbeitsschritten zu entlasten.<sup>15</sup>

<sup>12</sup> Vgl. Siciliano / Khatib, S.1, 2008.

<sup>13</sup> Vgl. Prof. Dr. Bendel, Wirtschaftslexikon Stichwort: Robotik, o.J., o.S.

<sup>14</sup> VDI-Richtlinie 2860, Blatt 1.

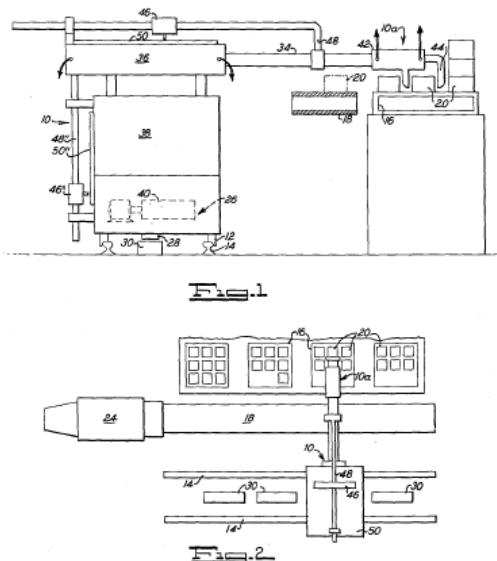
<sup>15</sup> Vgl. Prof. Dr.-Ing. Linemann, S.4 ff., 2016.



**Abb. 5.:** Skizze von Heron von Alexandria »Automata«. Bildquelle: [Wikc].

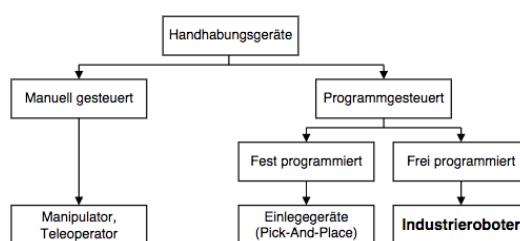


**Abb. 6.:** Die Droiden von Jaquet Droz und seinem Sohn: Der Zeichner (links) und seine Zeichnung (links unten), Der Schreiber (mitte) und dessen Text (rechts unten) und die Musikerin (rechts). Bildquelle: [Wikd], [Wikf], [his], [Wike], [Mer].



**Abb. 7.:** Patentzeichnung für den ersten Industrieroboter. Bildquelle: [Dev].

Laut der oben genannten VDI-Richtlinie sind Industrieroboter eine Untergruppe der Handhabungsgeräte. Abbildung 8 zeigt dabei ihre Unterteilung in verschiedene Untergruppen. Zu erwähnen ist, dass die Kategorien »Manipulator, Teleoperator« und »Industrieroboter« in Zukunft wieder zusammengeführt werden könnten. So zählt auch in dieser Abschlussarbeit die prototypische Implementierung mehr zur Kategorie der Teleoperator. Allerdings wird darauf hingewiesen, dass Industrieroboter durch das verschmelzen der Arbeitsbereiche von Mensch und Maschine immer mehr zu »Cobots«<sup>16</sup> werden könnten und somit ein hybrides Dasein zwischen Werkzeug und autonomer Maschine pflegen könnten.



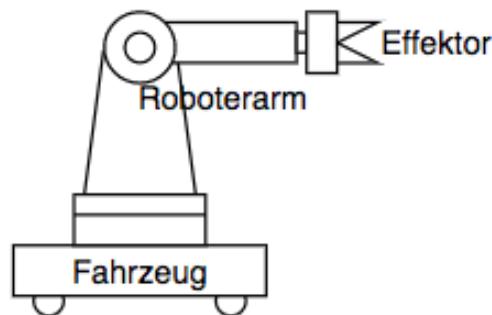
**Abb. 8.:** Unterteilung von Handhabungsgeräten in Untergruppen. Bildquelle: [Lin16].

<sup>16</sup> Kooperierende Roboter (Cobot), bezeichnen Roboter die gemeinsam mit Menschen Arbeit verrichten.

Man unterscheidet vier Hauptgruppen von Robotern:

- Industrieroboter
- Serviceroboter
- Geländeroboter
- Autonome Roboter / Agentensysteme

Betrachtet man die Gruppe der Industrieroboter genauer, ist es wichtig sich über die genaue Beschaffenheit des Roboters im Klaren zu sein, da sie sowohl seinen Arbeitsraum als auch die Komplexität seiner Bewegungen definieren. Die Hauptkomponenten sind, wie in Abbildung 9 gezeigt, das Fahrzeug, welches die Lokomotion bestimmt, der Roboterarm, der den Endeffektor führt, und der Endeffektor selbst, welcher mit der Umwelt interagiert.



**Abb. 9.:** Schematische Darstellung eines Industrieroboters. Bildquelle: [Lin16].

Alle genannten Komponenten haben Freiheitsgrade, welche die Komplexität der Dynamik und Kinematik des Roboters festlegen. Die folgende Formel bestimmt dabei die Bewegungsfreiheitsgrade eines Roboters:

$$F = \sum_{i=1}^n (F_{R_i} + F_{T_i}), \text{ mit } F \geq f \quad (2.1)$$

Dabei gibt  $F_R$  den Freiheitsgrad eines Rotationsgelenks an und nimmt einen Wert  $\leq 3$  an,  $F_T$  gibt den Freiheitsgrad eines Translationsgelenks an, der bei 1 liegt und  $n$  beschreibt die Anzahl der Gelenke eines Roboters.

## 2.2.2 Fertigungsroboter

Das Bild der Fertigungsstraßen des frühen letzten Jahrhunderts korreliert mit dem Verlauf der Industriellen Revolutionen aus Abbildung 2. Die Werkshallen waren bedingt durch Henry Fords Erfindung, der Bandfertigung, gefüllt mit Arbeitern, die zum Teil simple und monotone Arbeiten ausführen mussten (Abbildung. 10).



**Abb. 10.:** Fließbandarbeiter um 1913 in der Automobilfabrikation von Henry Ford.  
Bildquelle: [Run].

Mit anhaltendem technologischen Fortschritt stieg auch die Anzahl der Arbeiten, die von Maschinen bewerkstelligt werden konnten. Maschinen die eine solche Aufgabe erfüllen, nennt man Fertigungsroboter. Beim Wort Roboter, haben viele Menschen das Bild eines *Humanoiden Roboters*<sup>17</sup>, wie er in Filmen oft zum Einsatz kommt, im Kopf. Allerdings sind Roboter in menschlicher Form nur bedingt sinnvoll, wenn es um das Lösen spezifischer Aufgaben geht, denn unsere Anatomie hindert uns an vielen Bewegungen. Das mag für uns sehr sinnvoll sein und kann physiologische Gründe haben, für einen Roboter mit einer bestimmten Aufgabe ist dies jedoch eher hinderlich. Warum sollte sich ein Roboterarm beispielsweise nicht in allen Freiheitsgraden bewegen können? Blickt man in die Produktionshallen des 21. Jahrhunderts sind dort in vielen Bereichen immer mehr solcher Roboter anzutreffen. Es scheint ein reges Interesse an Maschinen dieser Art zu herrschen. Beispielsweise hat sich

<sup>17</sup> Roboter in Menschengestalt.



**Abb. 11.:** Industrieroboter in einem Daimler-Werk. Bildquelle: [Dai].

die Anzahl der Industrieroboter seit 2010 mehr als verdoppelt<sup>18</sup> und chinesische Investoren haben letztes Jahr den deutschen Industrieroboterhersteller *KUKA* gekauft.<sup>19</sup> Die Roboter erledigen ihre Aufgaben meist präziser und schneller als Menschen es je könnten, denn sie wurden für eine spezifische Aufgabe entwickelt und gebaut. Hinzu kommt, dass Roboter – im Gegensatz zu Lebewesen – nicht durch Krankheit oder persönliche Gründe ausfallen. So kann das ganze Jahr über eine gleichbleibend stabile Produktion und Qualität gewährleistet werden. Das führt allerdings auch dazu, dass Fertigungsroboter oft nur ihre eigenen Aufgaben kennen und nur wenige Sensoren zur Außenwelt besitzen. Betritt ein Mensch ihren Arbeitsbereich, ist höchste Vorsicht geboten, denn Industrieroboter sind schnell und kraftvoll; erkennt der Roboter den Menschen auf seinem Weg nicht, sind schwere Verletzungen bis hin zum Tod die Folge. Momentan arbeiten Roboter deswegen in einem Arbeitsbereich, der von dem des Menschen getrennt ist. In Zukunft soll sich dies durch immer leistungsfähigere, vernetzte und ihrer Umgebung gegenüber aufmerksamen Robotern ändern und das Arbeiten an der Fertigungsstraße von Mensch und Maschine kooperativ bewerkstelligt werden.

---

<sup>18</sup> Vgl. Dr. Schmidt, S.5, 2016.

<sup>19</sup> Süddeutsche Zeitung Hrsg., 2016, [Accessed: 16.02.2017].

## 2.2.3 Lego Mindstorms EV3

Der *Lego Mindstorms EV3* ist die dritte Generation von einer Spielzeugreihe, die sich mit der Verwirklichung von Robotern oder technischen Maschinen aus *Lego*-Steinen befasst. Es handelt sich dabei um einen größeren Stein, der als Rechner fungiert, und um Sensoren und Motoren, die per Kabel an diesen angeschlossen werden. Das Besondere daran ist: Man kann eigene Programme schreiben, welche die Motorenbewegung steuern und die Sensordaten auslesen. Mit grafischer Programmiersprache ausgeliefert, aber als Spielzeug für Kinder entwickelt und gedacht, erfreut sich der Stein auch in der Lehre und Forschung großer Beliebtheit. Tauscht man die Firmware des Steins aus, ist es möglich, diesen mit Programmen zu bespielen, die in *Java*, *C#* oder *Python* geschrieben wurden. Im Internet finden sich so unzählige Videos, Beiträge von Enthusiasten und Lehrbeispiele, welche die *Mindstorms*-Reihe nutzen.



**Abb. 12.:** Hauptkomponente des *Lego Mindstorms* der *EV3*-Stein. Bildquelle: [Leg].

Die, in Abbildung 12 dargestellte, Recheneinheit des im September 2013 auf den Markt gebrachten *EV3* ist ausgestattet mit einem 178x128 Pixel monochromen LCD, einem *Ti Sitara AM1808* Prozessor mit 300MHz, 64MB RAM

und 16MB Flash-Speicher. Außerdem verfügt er über einen SD-Kartenslot, mit dem der Speicher erweitert oder eigene Programme und Firmware auf den Roboter gespielt werden können. Ein *USB Host Port*, der zur Erweiterung der Funktionalität mit WLAN genutzt werden kann, ergänzt die Konnektivität des Steins. Hinzu kommen Ports für jeweils vier Servomotoren und Sensoren. Im Lieferumfang enthalten sind drei Motoren unterschiedlicher Bauart. Jeweils ein Berührungs-, ein Infrarot- und ein Farberkennungssensor vervollständigen das Set und ermöglichen erste Experimente.<sup>20</sup>

In dieser Arbeit wird der *EV3* als Substitution für einen echten Industrieroboter benutzt. In der folgenden Sektion 2.3 werden die Geschichte, der Forschungsstand sowie die Entwicklung von AR betrachtet und das in der vorliegende Arbeit genutzte HMD die *Microsoft HoloLens* vorgestellt.

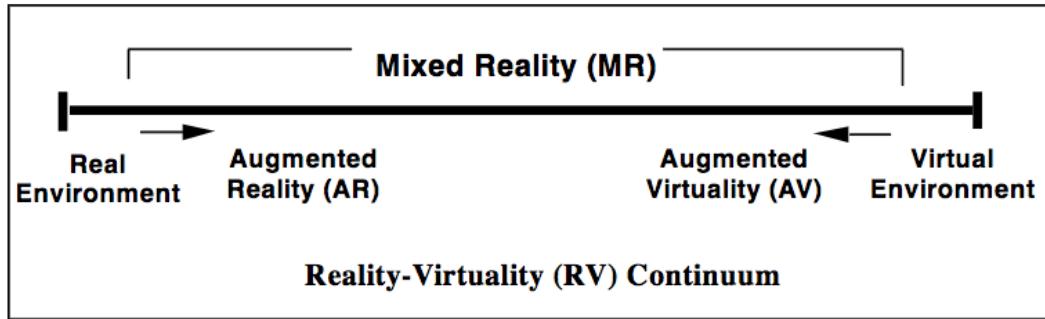
## 2.3 Augmented Reality

Der Begriff *Augmented Reality* bezeichnet »die computergestützte Erweiterung der Realitätswahrnehmung«.<sup>21</sup> Sie kann alle menschlichen Stimuli ansprechen. Es wird jedoch häufig nur die computergestützte Erweiterung der visuellen Darstellung gemeint. Sie ist eine Untermenge der virtuellen Realität. Man kann mit 3D-Objekten interagieren und diese beliebig in allen Freiheitsgraden translatieren, rotieren und skalieren. Bei nicht-statischen AR-Anwendungen werden Trackingdaten miteinbezogen, um die Erfahrung mit der realen Welt zu verschmelzen. Das von 1994 von Paul Milgram und Fumio Kishino definierte *Reality-Virtuality Continuum* aus Abbildung 13 umfasst eine Spanne von der echten Umgebung (links) bis zur virtuellen Umgebung (rechts); Es gehört damit auch zur Gruppe der virtuellen Realitäten, jedoch etwas weiter links Richtung echter Umgebung lokalisiert.<sup>22</sup>

<sup>20</sup> Anonymous, 31313 Mindstorms EV3, o.S., o.J., [Accessed: 21.02.2017].

<sup>21</sup> Wikipedia, 2017, [Accessed: 28.12.2016].

<sup>22</sup> Vgl. Carmignani, Furth, S.4., 2011.



**Abb. 13.:** Reality-Virtuality Continuum von Paul Milgram und Fumio Kishino. Bildquelle: [res].

Im Gegensatz zu VR-Anwendungen werden Nutzer allerdings nicht von der Außenwelt abgeschirmt. AR ergänzt beziehungsweise augmentiert die reale Welt und macht sie deshalb besonders wertvoll für Anwendungen im industriellen Kontext.<sup>23</sup>

### 2.3.1 Überblick, Entwicklung und Forschungsstand

AR wurde der breiten Öffentlichkeit insbesondere durch das Erscheinen des Spiels *Pokémon GO* im Sommer 2016 bekannt. Eine Anwendung wie diese benutzt die Kamera des Smartphones, um kleine Monster in das Bild der realen Welt auf dem Display zu projizieren. AR hat ihren Ursprung jedoch schon viel früher. Dazu ein passendes Zitat von Morton Heilig:

»When anything new comes along, everyone, like a child discovering the world, thinks that they've invented it, but you scratch a little and you find a caveman scratching on a wall is creating virtual reality in a sense. What is new here is that more sophisticated instruments give you the power to do it more easily.«  
– Morton Heilig

1833 erfand der britische Physiker Sir Charles Wheatstone das Spiegelstereoskop. Abbildung 14 zeigt Wheatstones Erfindung, die mit Hilfe zweier um 45° geneigter Spiegel, einem je Auge, zwei Teilbilder in den Blick des Betrachters spiegelt und so einen räumlichen Eindruck des Gesamtbildes erschaffte.<sup>24</sup>

<sup>23</sup> Vgl. Arnoldy / Bautz / Bruns et al, S.6, 2016.

<sup>24</sup> Vgl. Ph.D Jerald, S.15, 2016.

Darauf aufbauend entwickelte David Brewster um 1849 ein kleineres, handliches Stereoskop, welches in Abbildung 15 dargestellt ist. Es nutzte Linsen, statt Spiegel und ist so um ein vielfaches benutzerfreundlicher und kleiner als die Erfindung Wheatstones.



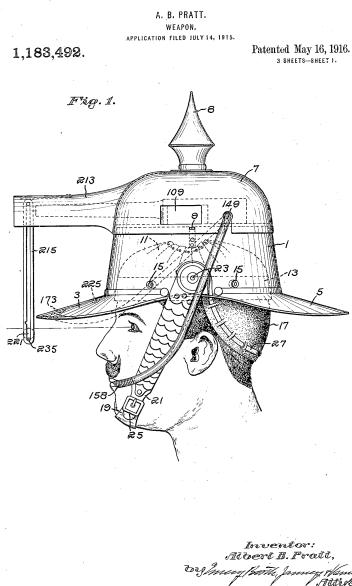
**Abb. 14.:** Charles Wheatstones Spiegelstereoskop. Bildquelle: [kin].



**Abb. 15.:** David Brewsters Linsenstereoskop. Bildquelle: [Wika].

Neue Interaktionskonzepte, wie Albert Pratts am Helm montierte Ziel- und Schießeinrichtung<sup>25</sup> aus Abbildung 16 von 1915, führten zu einem neuen Verständnis der Interaktion und Handhabung von Geräten.

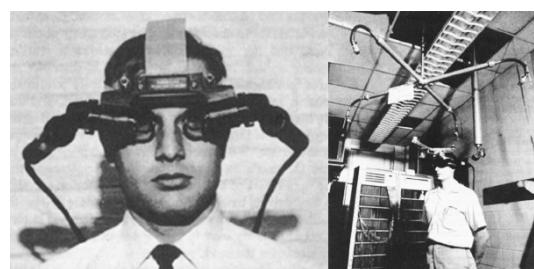
<sup>25</sup> Pratt, o.S., 1913, [Accessed: 24.01.2017].





**Abb. 17.:** Morton Heiligs Erfindung das »Sensorama« zeigt Filmerlebnisse, die auch andere Sinne ansprechen. Bildquelle: [mor].

Sechs Jahre später, 1968, entwickelte Ivan Sutherland<sup>27</sup> zusammen mit seinem Studenten Bob Sproull das erste, als funktionsfähig zu betrachtende HMD für AR und VR. Es trägt den Namen »The Sword of Damocles«, welcher darauf zurückzuführen ist, dass die Vorrichtung an der Decke befestigt über dem Nutzer schwebte. Seine Erfindung verfügte über Systeme zur Erfassung der Kopfbewegung und computergenerierter Grafik, die eine Welt aus Drahtgeometriedarstellungen erstellte.<sup>28</sup> 1985 präsentierte Myron Krueger



**Abb. 18.:** Ivan Sutherlands Erfindung »Sword of Damocles«, 1968. Bildquelle: [aug].

auf der *Special Interest Group on Computer-Human Interaction (SIGCHI)* sein 1970 gestartetes Labor für erweiterte Realität mit dem Namen – *Videoplace*.

<sup>27</sup>Amerikanischer Professor und Pionier für Computergrafik, oft auch als „Vater der Computergrafik“ bezeichnet

<sup>28</sup>Vgl. Jerald, S.22, 2016.

Es kombinierte die Silhouette eines Benutzers aus einer Kameraaufnahme mit Bildern einer computer-grafischen Welt. Man konnte sogar mit Objekten interagieren und so beispielsweise langsam mit der Hand schreiben oder Objekte bewegen.<sup>29</sup> Letztendlich führten 1992 zwei Wissenschaftler der Firma Boeing den Begriff *Augmented Reality* ein. Sie entwickelten ein HMD, das als Hilfestellung beim Verlegen von Kabeln in Flugzeugen dienen sollte.<sup>30</sup> Noch im selben Jahr veröffentlichte Dr. Louis B. Rosenberg als Ergebnis seiner Doktorarbeit ein *Virtual Fixture*-System, welches für die U.S. Air Force angefertigt wurde. Es wird als erstes immersives Augmented Reality-System, das jemals gebaut wurde, angesehen.<sup>31</sup>

Bereits 1993, nur ein Jahr später wurde das Projekt *KARMA*, was für *Knowledge-based Augmented Reality for Maintenance Assistance* steht und von Steven Feiner, Blair MacIntyre, Dorée Seligmann an der Columbia Universität entwickelt wurde, ins Leben gerufen. *KARMA* sollte es ermöglichen, eine Anleitung zum Reparieren von defekten Laserdruckern zu augmentieren und den Arbeitern so eine Anleitung zu bieten, wie sie beispielsweise das Papierfach leeren.<sup>32</sup> 1996 wurde dann von Dr. Rekimoto zum ersten Mal ein markerbasiertes



**Abb. 19.:** HMD(links) und augmentierte Hilfestellung(rechts) des Projekts »KARMA«. Bildquelle: [col].

AR-System eingeführt mit dem Namen *Matrix Code*. Laut Rekimoto war das besondere an seiner Technik, dass sie in der Lage war, simultan Objekte in der echten Welt zu identifizieren sowie deren Koordinatensystem sehr genau abzuschätzen. Zudem war sie sehr kostengünstig, da 2D-Matrixmarker, welche nichts anderes als quadratische Barcodes sind, benutzt wurden und man diese einfach ausdrucken und an geeigneter Stelle anbringen konnte. Außerdem waren die Marker verlässlicher und genauer als Ultraschall- oder Magnet-

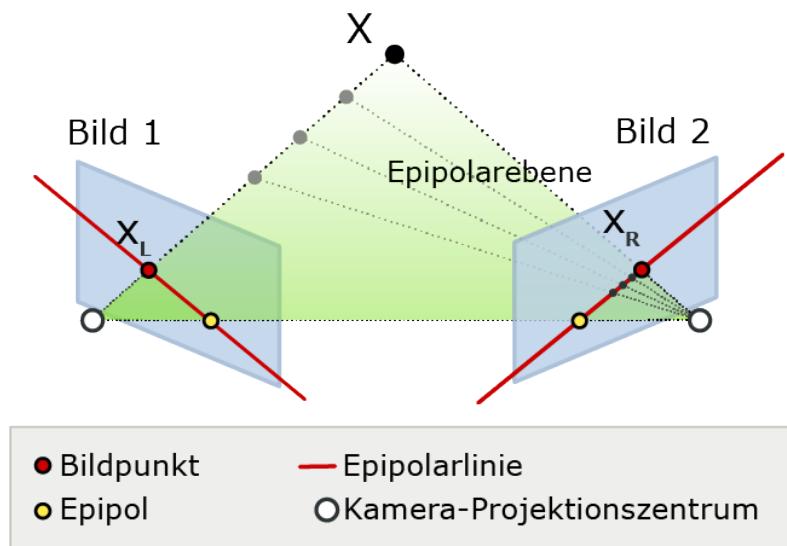
<sup>29</sup> Vgl. Krueger, S. 35f., 1985.

<sup>30</sup> Vgl. Arnoldy / Bautz / Bruns et al., S.7, 2016.

<sup>31</sup> Vgl. Wikipedia Virtual Fixtures.

<sup>32</sup> Vgl. Feiner / MacIntyre / Seligmann 93, [Accessed: 10.01.2017]

3D-Sensoren für die Positionsbestimmung.<sup>33</sup> Danach wurde viel Forschung im Bereich maschinelles Sehen betrieben und es konnte eine »Verschiebung der Forschungsschwerpunkte innerhalb der automatischen Bildanalyse von zweidimensionalen hin zu dreidimensionalen Problemstellungen festgestellt werden«.<sup>34</sup> Hervorzuheben ist dabei die Verwendung von *Stereoverfahren* zur Tiefenbestimmung. Dabei werden zwei oder mehrere Bilder der gleichen Szene aus verschiedenen Kamerapositionen aufgenommen. Durch Bekanntheit der Parameter, als auch der Anordnung der Kamera, kann durch ein solches Verfahren die Lage und räumliche Position eines bestimmten Punktes in jedem der Bilder identifiziert werden.<sup>35</sup> Besonders hervorzuheben sind dabei: »Die Wahl der Kalibrierung der Stereogeometrie, Wahl und Detektion der Merkmale, Korrespondenzanalyse, Tiefenbestimmung und Interpolation. Unter Stereogeometrie fällt zum einen das mathematische Modell jeder einzelnen Kamera (z.B. Lochkameramodell), zum anderen die relative Lage der beiden Kameras zueinander«.<sup>36</sup> Die Ergebnisse des Stereoverfahrens können durch Zuordnungseinschränkungen erheblich verbessert werden. Zu den am häufigsten verwendeten Zuordnungsverfahren zählt die *Epipolargeometrie*.



**Abb. 20.:** Schaubild zur Epipolargeometrie. Bildquelle: [Wikb].

Das Verfahren nutzt die Gegebenheit, dass der Bildpunkt  $X_L$  aus Abbildung 20 und alle möglichen auf seinem Weg zum Objektpunkt  $X$  liegenden Punkte auf einem Bild aus einem anderen Blickwinkel aufgenommen auf einer Gera-

<sup>33</sup> Vgl. Ph.D Rekimoto, S.1, 1998.

<sup>34</sup> Dr. Jiang / Prof. Dr. Bunke, o.S. (Vorwort), 1997.

<sup>35</sup> Vgl. Dr. Jiang / Prof. Dr. Bunke, S.7., 1997.

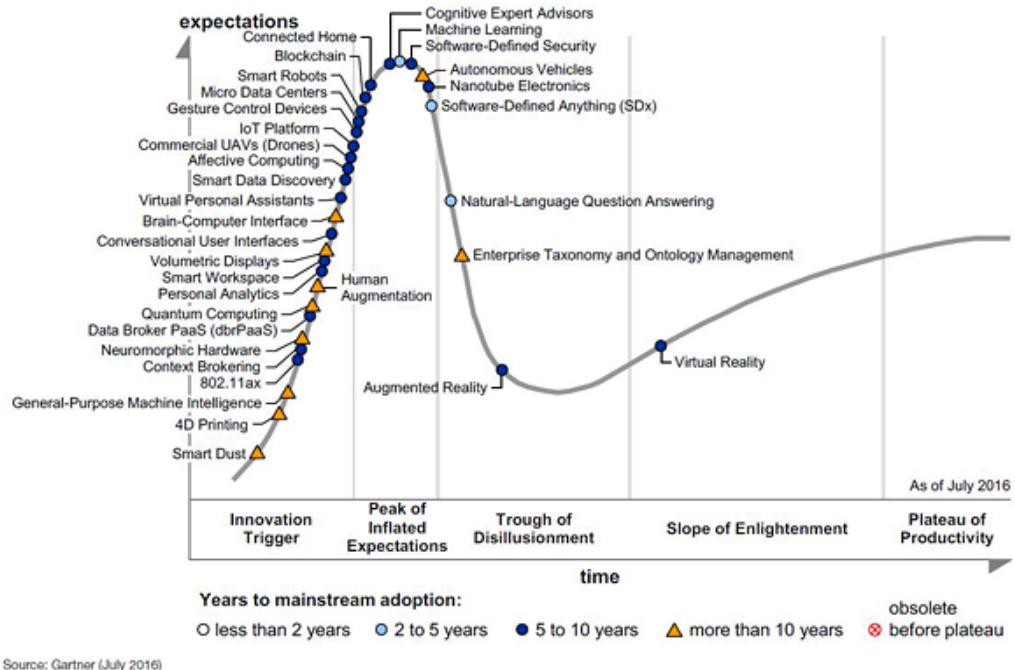
<sup>36</sup> Dr. Jiang / Prof. Dr. Bunke, S.7., 1997.

den liegen müssen und schränken somit den Suchbereich für den Bildpunkt  $X_R$  in Beziehung zum Bildpunkt  $X_L$  auf die Epipolarlinie ein. Diese und weitere mathematische und technische Fortschritte im Bereich maschinelles Sehen trieben auch die Entwicklung von AR voran. Angefangen mit *PlayStation* Zusatzgerät *EyeToy*, welches 2003 für die *PlayStation 2* erschien, bis hin zu *Microsofts Kinect*, erschienen im Jahr 2010, wurden Kamerasysteme für den Verbrauchermarkt immer beliebter. Mit ihnen hielten AR-Anwendungen, ähnlich des Projekts *Videoplacement* von Myron Krueger, Einzug in die Wohnzimmer. Abbildung 21 zeigt eine solche Applikation, in der die Bewegung des Spielers mit Hilfe des Trackings der Kamera auf eine Spielfigur übertragen werden.



**Abb. 21.:** Spielfigur macht Bewegung des Spielers durch Kameratracking im Spiel *EyeToy3* nach. Bildquelle: [Pla].

Gartners 2016 Hype Cycle of Emerging Technologies aus Abbildung 22 verdeutlicht die momentane Position von AR.



**Abb. 22.:** Gartners Hypecycle Diagramm für das Jahr 2016. Bildquelle: [Gar].

Das Diagramm zeigt, dass die Technologie 2016 im *Trough of Disillusionment* (z.Dt. Tal der Enttäuschung) lag und die Aufregung die Technologie betreffend bereits am verfliegen ist, da die hohen Erwartungen, welche zu Beginn des *Hypes* an AR adressiert waren, nicht erfüllt wurden. Und da man mittlerweile weiß, wo die Knackpunkte von AR und dementsprechend auch Vor- und Nachteile liegen, wird jetzt stattdessen eher im Stillen in der Phase des *Slope of Enlightenment* (z.Dt. Pfad der Erleuchtung) weiter an ihr gearbeitet. Dennoch muss es AR jetzt auf das *Plateau of Productivity* (z.Dt. Plateau der Produktivität) und damit den Absprung auf den Massenmarkt schaffen, um akzeptiert zu werden. Dies macht AR zu einem spannenden Feld für die Erprobung kleinerer Applikationen und Konzepte mit neuen Geräten und echten Testprobanden.

### 2.3.2 Microsoft HoloLens

Am 21. Januar 2015 stellte *Microsoft* im Rahmen des *Windows 10 Events* sein *Augmented Reality Headmounted Display* »HoloLens« vor.<sup>37</sup> Es handelt sich um ein sehr neues Gerät, welches erst seit 12.10.2016 in Europa von

<sup>37</sup> Vgl. Microsoft Windows 10 Event January 2015 Präsentationsvideo ab 1:37:40 [Accessed: 13.01.2017].

Endkunden in der *Development-Kit*-Version vorbestellt werden konnte<sup>38</sup> und ab Anfang Dezember 2016 verschifft wurde. Alle tiefergreifenden technischen Informationen sind bisher unter Verschluss beziehungsweise nur sehr grob dokumentiert. Trotzdem, oder besser, genau deswegen, wird dieses Kapitel die »Hardware-Details« genauer beleuchten und auf einzelne Bauteile tiefer eingehen.



**Abb. 23.:** Microsofts Datenbrille, die *HoloLens*. Bildquelle: [Micb].

Die Daten-Brille (Abb. 23) verfügt über folgende Merkmale:

### Optik

- Transparente holographische Linsen (Lichtleiter)<sup>39</sup>
- Zwei *High-Definition 16:9 Light-Engines*
- Automatische Interpupillardistanz Kalibrierung
- Holografische Auflösung von 2,3M Lichtpunkte insgesamt
- Holografische Dichte von >2,5T radians (Lichtpunkten pro Radian)

<sup>38</sup> Vgl. Microsoft Pressemitteilung [Accessed: 14.01.2017].

<sup>39</sup> Als Lichtleiter werden transparente Bauteile wie Fasern, Röhren oder Stäbe bezeichnet, die Licht über kurze oder lange Strecken transportieren. (Vgl. Wikipedia Link)

## Sensoren

- Eine Inertiale Messeiheit(IMU)<sup>40</sup>
- Vier umgebungswahrnehmende Kameras
- Eine Tiefenkamera
- Eine 2MP Foto- / HD Videokamera
- Vier Mikrofone
- Einen Umgebungslichtsensor

## Mensch-Maschine Kommunikation

- Räumlicher Klang durch zwei eingebaute Lautsprecher
- Tracking der Kopfbewegung (Gaze)
- Gestenerkennung
- Spracherkennung
- Webinterface

## Weitere technische Details

- Intel 32-Bit Architektur mit *Trusted Platform Modul 2.0*<sup>41</sup>
- Spezialangefertigte *Microsoft Holographic Processing Unit (HPU 1.0)*
- 64GB Flash-Speicher

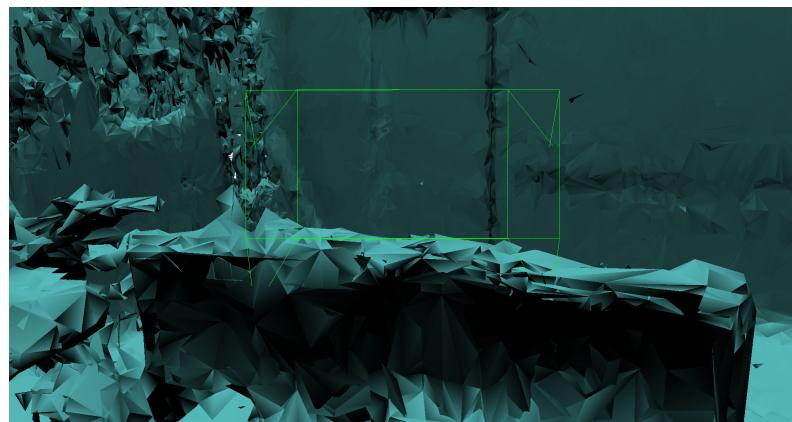
---

<sup>40</sup> Chipset mehrerer Sensoren, wie Gyroskop, Beschleunigungssensor, Drehratensensoren und Magnetometer, das der Bewegungsdetektion dient.( Vgl. Wikipedia Link) In der HoloLens dient sie vermutlich zur Bildstabilisierung.

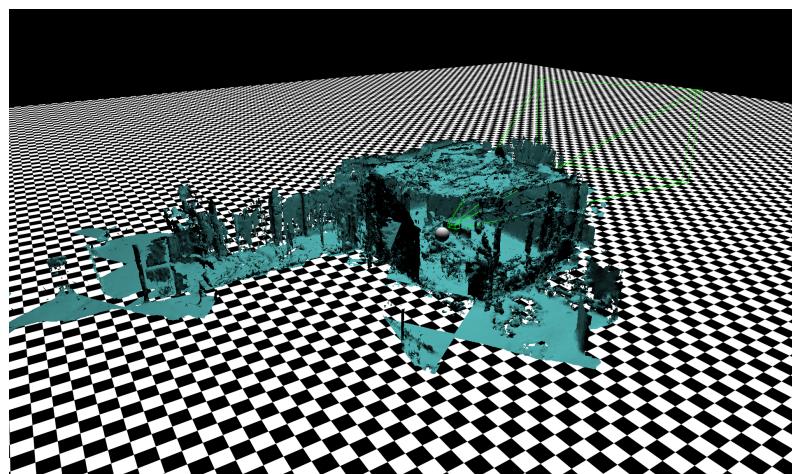
<sup>41</sup> Ein Chip nach der TCG-Spezifikation, der einen Computer oder ähnliche Geräte um grundlegende Sicherheitsfunktionen erweitert. [https://de.wikipedia.org/wiki/Trusted\\_Platform\\_Module](https://de.wikipedia.org/wiki/Trusted_Platform_Module)

- 2GB RAM
- Wi-Fi 802.11ac
- Bluetooth 4.1 LE
- Gesamtgewicht: 579g

Das Beeindruckende an Microsofts HoloLens ist die Geschwindigkeit und Genauigkeit ihres »Mappings«. In Abbildung 24 und 25 sieht man beispielsweise Büroräumlichkeiten von der HoloLens erfasst und als fertiges 3D-Objekt dargestellt sowie in Abbildung 26 die erfassten Daten generiert als *Mesh*.

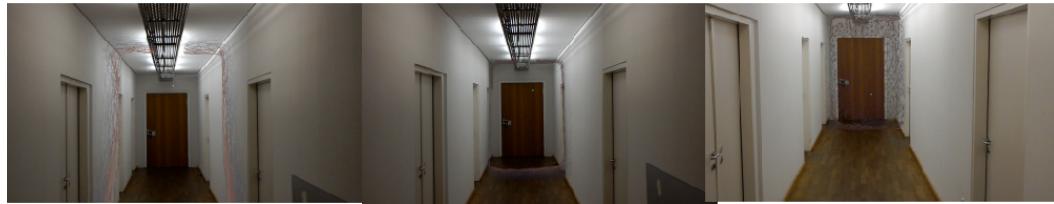


**Abb. 24.:** Von der *HoloLens* erfasst: Schreibtisch und Wände als 3D-Objekt. Bildquelle: Eigenes Werk.



**Abb. 25.:** Von der *HoloLens* erfasste Räume aus der Außenansicht als 3D-Objekt. Bildquelle: Eigenes Werk.

Doch wie funktioniert die Raumerkennung genau? Betrachten wir zunächst



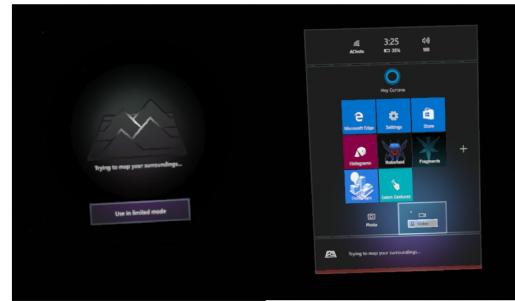
**Abb. 26.:** *HoloLens – Mesh* des erfassten Raumes in mehreren Stufen. Bildquelle: Eigenes Werk.

die Tiefenkamera der HoloLens. Es ist stark davon auszugehen, dass eine sogenannte *Time Of Flight* (ToF)-Tiefenkamera, wie sie auch in der »Kinect v2«<sup>42</sup> vorzufinden ist, verwendet wird. Beim ToF-Verfahren wird ein Lichtpuls in die Szene gegeben, um diese auszuleuchten. Danach misst die Kamera die Zeit, die das Licht benötigt, um zu einem Objekt und wieder zurück zu gelangen. Somit lässt sich im Sensor der ToF-Kamera ein Tiefenwert für jeden Bildpunkt errechnen. Auch bei der HoloLens wird Licht im Infrarotspektrum per Laserdiode oder LED in die Szene, wie in Abbildung 29 zu sehen ist, geschossen, um Anwender nicht mit ständigen Lichtblitzen zu stören. Die ToF-Kamera in Verbindung mit den in stereoskopischer Lage angebrachten Frontkameras führt bei der Abtastung des Raumes zu einer hohen Anzahl von Punkten und Tiefenwerten. Diese ermöglichen es der *HoloLens* in kurzer Zeit die gewonnenen Daten in der *Holographic Processing Unit* zu verarbeiten und den Raum, mit einem auf ihn abgestimmten *Mesh* versehen, anzuzeigen. Außerdem erlauben es die *Inertiale Messeinheit* und das »Bewusstsein« über die Beschaffenheit des Raumes, den Träger der Datenbrille exakt im Raum zu lokalisieren. Diese Gegebenheit macht es möglich, dass der Anwender mit seiner Umgebung und Hologrammen in AR interagiert. Diese Interaktionen und deren genaue Funktionsweise werden in Sektion 2.3.3 beschrieben.<sup>43</sup>

**Tests** Im Rahmen der vorliegenden Arbeit wurden drei Funktionstest durchgeführt, um die Funktionsweise der *HoloLens* weiter zu ergründen. Als erstes wurde versucht die *HoloLens* in einem nicht, oder nur schwach beleuchteten Raum zu verwenden. Das Ergebnis beläuft sich darauf, dass sie ihr *Tracking* verliert und den Anwender auffordert den *limited mode* zu nutzen, welcher ohne *spatial mapping* auskommt. Zu sehen ist das Ergebnis aus der *HoloLens* heraus aufgenommen in Abbildung 27.

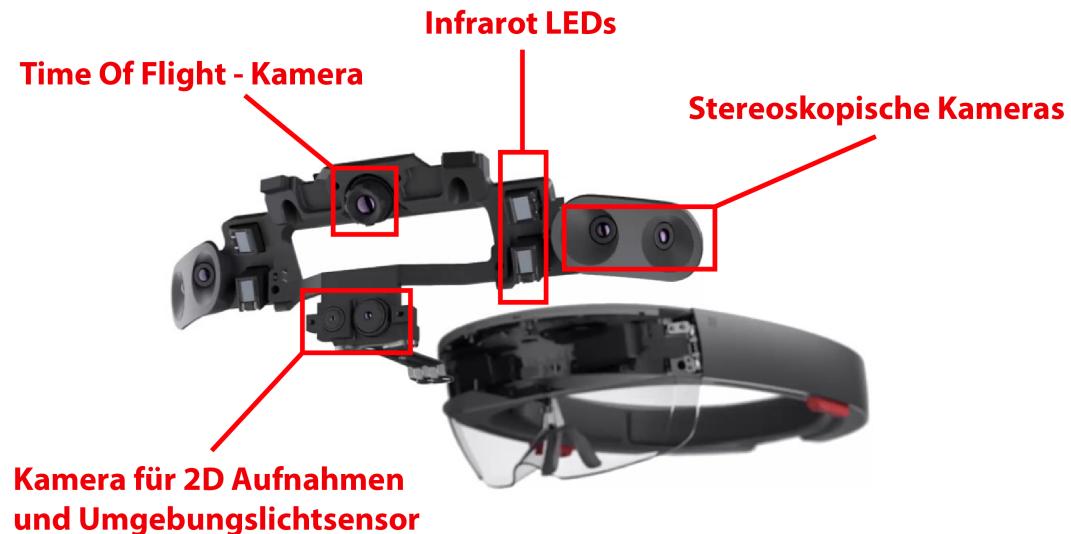
<sup>42</sup> „Kinect (abgeleitet vom englischen kinetic connect, deutsch Kinetische Verbindung) ist eine Hardware zur Steuerung der Videospielkonsole Xbox 360“ <https://de.wikipedia.org/wiki/Kinect>

<sup>43</sup> Vgl. *HoloLens Hardware Details*, [Accessed: 14.01.2017].



**Abb. 27.:** Microsoft HoloLens in dunklem Raum. Bildquelle: Eigenes Werk.

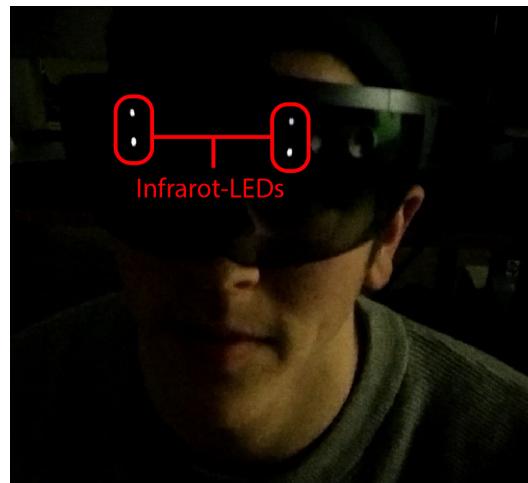
Des Weiteren wurden die Komponenten aus Abbildung 28 Kameras abgedeckt und beobachtet wie die Datenbrille reagiert.



**Abb. 28.:** Microsoft HoloLens Sensorenleiste. Bildquelle: [Micc].

So verhindert ein abgedeckter *ToF*-Sensor, dass die Gesten erkannt werden, was darauf schließen lässt, dass die Daten aus dem *ToF*-Sensor ebenfalls für das *Gesture-Tracking* benutzt werden. Außerdem können einzelne Kameras der vier räumlichen Kameras abgedeckt werden, ohne einen Funktionsverlust zu erleiden. Deckt man allerdings, wenn auch nur kurzzeitig, alle vier ab, verliert die Brille sofort ihr *Tracking*. Als letzter Funktionstest wurde die *HoloLens* mit einem Infrarotstrahler des Typs *rayTEC RM25-F-120 RAYMAX 25*

*Fusion- 120deg Illumination- 850nm* bestrahlt. Erstaunlicherweise konnte das Tracking der *HoloLens* nicht durch den Beschuss des Infrarotstrahlers gestört werden.



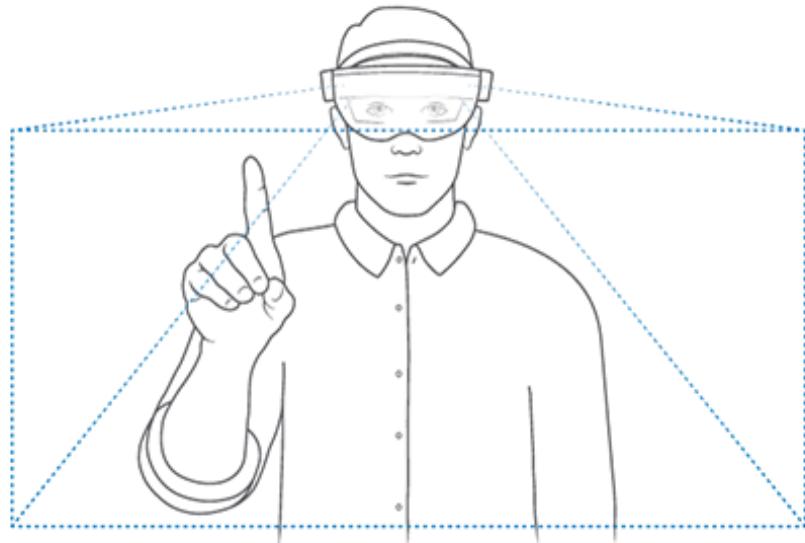
**Abb. 29.:** Infrarot Leuchten der *Microsoft HoloLens* in Betrieb. Bildquelle: Eigenes Werk.

### 2.3.3 Interaktionen in AR

Die *HoloLens* bietet insgesamt drei Hauptmöglichkeiten zur Interaktion:

- Gaze – Kopfbewegung
- Gestures – Gesten
  - Tap-Geste
  - Bloom-Geste
  - Tap-And-Hold-/Hold-And-Release-Geste
    - \* Scroll
    - \* Drag
    - \* Zoom
- Voice Commands – Spracherkennung

Die Gesten werden von zwei der vier Kameras, die am Frontchassis angebracht sind, in einem Frustum, wie es in Abbildung 30 zu sehen ist, erkannt.



**Abb. 30.:** Erkennungsbereich für Gesten der Frontkameras. Bildquelle: [Mica].

**Gaze** Bezeichnet die Bewegung des Kopfes in allen Freiheitsgraden meint ist die Hauptform der Interaktion in der *HoloLens*. Sie steuert mit Hilfe des *Face-Vectors* und des Gyroskops den *Cursor* und somit die aktuelle Bearbeitungsposition im *ViewPort* beziehungsweise den *Point Of Interest*. Mit Hilfe des Gaze kann mit virtuellen Objekten interagiert werden. Trifft der *Raycast* des Gaze-Vektors auf ein Objekt, wird dieses zurückgegeben und man kann im Code programmatisch Aktionen auslösen.

**Gestures** Damit sind alle Gesten gemeint, die von den Kameras der *HoloLens* erkannt werden können. Die **Bloom**-Geste ist das Equivalent zur *Escape*-Taste am Computer. Sie bringt einen immer zum Hauptbildschirm zurück; Außerdem ruft sie das Start-Menü auf und schließt es auch wieder.

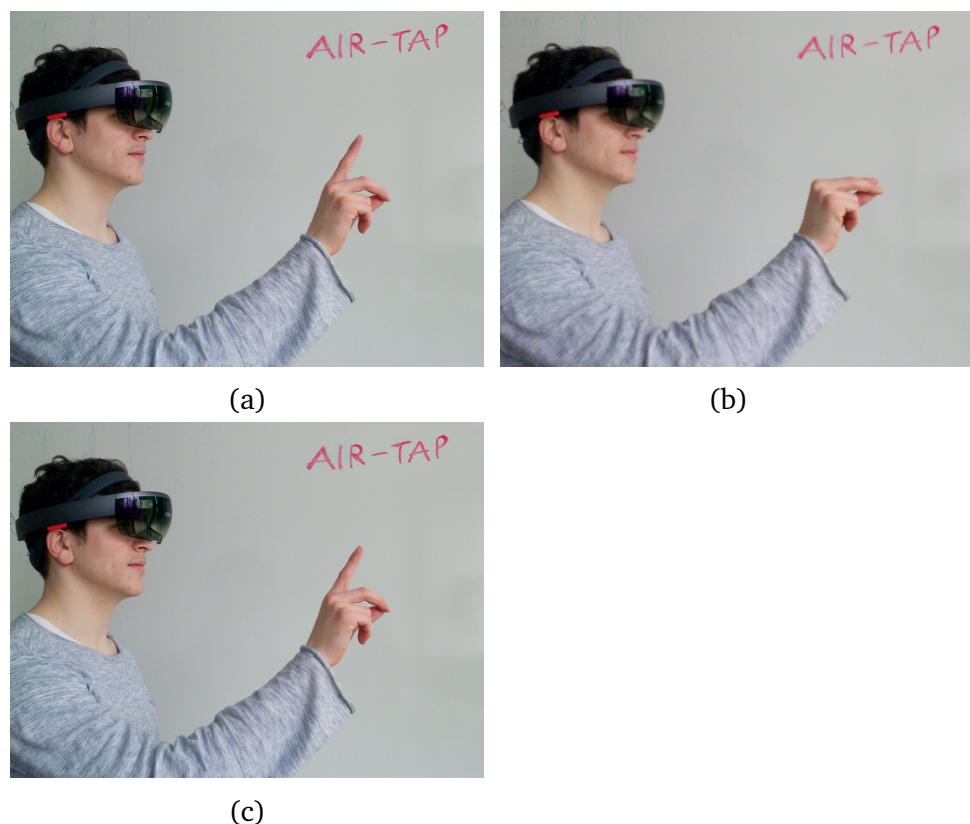
Die **Air-Tap**-Geste ist das Gegenstück zum klassischen Klick mit einer Maus am Computer. In Verbindung mit der *Gaze*-Geste des Kopfes können somit Hologramme ausgewählt und Aktionen ausgelöst werden.

Die Geste setzt sich aus einer schnellen Abfolge der **Tap-And-Hold**-Geste sowie der **Hold-And-Release**-Geste zusammen. Mit diesen lassen sich auch bekannte Manöver wie die *Drag-And-Drop*-Interaktion am Computer imitieren. Beispielsweise kann man Hologramme in allen Freiheitsgraden bewegen,

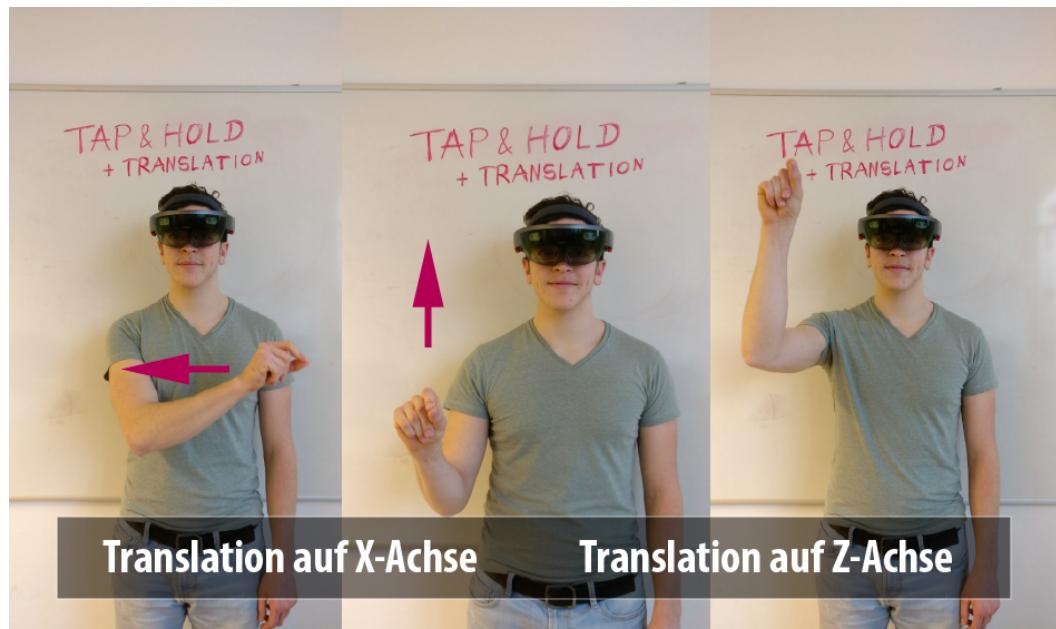


**Abb. 31.:** Prepare Bloom Gesture (Fist) (a), Perform Bloom Gesture (b). Bildquelle: Eigenes Werk.

Bildschirminhalte verschieben (Scrollen) und andere Regler-Interaktionen, wie *Zoomen* bedienen. Abbildung 33 zeigt, wie die Kombination aus diesen Gesten zusammen mit einer Translation der Hand auf der X-Achse und Y-Achse zum Scrollen verwendet werden kann.



**Abb. 32.:** Die Air-Tap-Geste. Initiiert mit der *Gesture-Ready*-Geste (a), ausgeführt mit dem *Tap* selbst (b) und Endposition erneute *Gesture-Ready*-Geste (c). Bildquelle: Eigenes Werk.

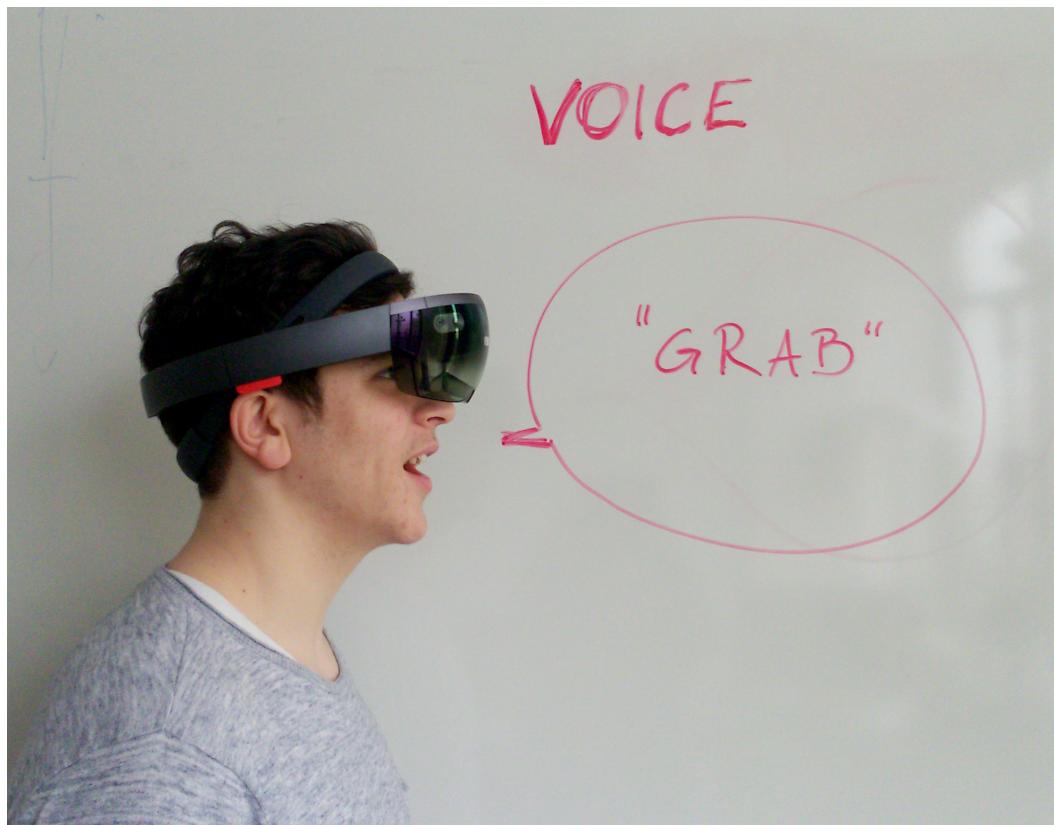


**Abb. 33.:** Scroll-Geste in X- und Z-Richtung (*Tap-And-Hold + Translation*). Bildquelle: Eigenes Werk.

**Voice-Commands** Die Spracherkennung der *HoloLens* ist eine Interaktionsform zusätzlich zur Gesteninteraktion. In Menüs der *HoloLens* sind alle interaktiven Schaltflächen auch mit Sprachkommandos aktivierbar. Dies trägt zur Barrierefreiheit bei und ist bei einer Spracherkennung mit einer Wort-zu-Fehlerrate von 5,9 Prozent durchaus als ein verlässliches Werkzeug anzusehen.<sup>44</sup>

---

<sup>44</sup> Vgl. Xiong / Droppo / Huang / et al, o.S., 2016.



**Abb. 34.:** Voice-Command: Am Beispiel »Grab«. Bildquelle: Eigenes Werk.

Die Gesten- und Sprachinteraktion sind den HCI-Kriterien der ISO 9241-110<sup>45</sup> für Soft- und Hardwaresysteme nachempfunden. Sie tragen abgekürzt den Namen »ASSEFIL«-Kriterien:

1. Aufgabenangemessenheit:  
»Ein Dialog ist aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen.«
2. Selbstbeschreibungsfähigkeit:  
»Wenn eine Eingabe verlangt wird, sollte das Dialogsystem dem Benutzer Informationen über die zu erwartete Eingabe geben.«
3. Steuerbarkeit:  
»Ein Dialog steuerbar, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist«

---

<sup>45</sup>DIN 9241-110, Hrsg., o.S., 2008.

4. Erwartungskonformität:

»Ein Dialog ist erwartungskonform, wenn er konsistent ist und den Merkmalen des Benutzers entspricht, zum Beispiel seinen Kenntnissen aus dem Arbeitsgebiet und seinen Erfahrungen sowie den allgemein anerkannten Konventionen.«

5. Fehlertoleranz:

»Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbarer Fehleingaben entweder mit keinem oder mit minimalem Korrekturaufwand seitens des Benutzers erreicht werden kann.«

6. Individualisierbarkeit:

»Ein Dialog ist individualisierbar, wenn Benutzer die Mensch-System-Interaktion und die Darstellung von Informationen ändern können, um diese an ihre individuellen Fähigkeiten und Bedürfnisse anzupassen.«

7. Lernförderlichkeit:

»Ein Dialog ist lernförderlich, wenn er den Benutzer beim Erlernen der Nutzung des interaktiven Systems unterstützt und anleitet.«

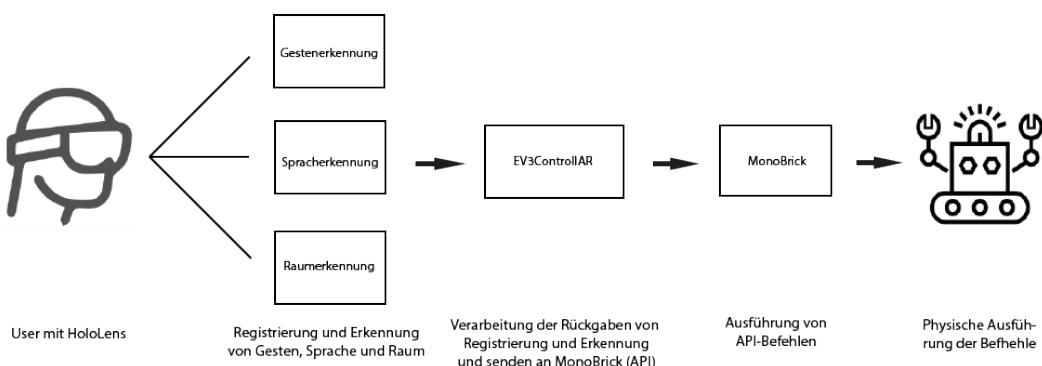
Im nächsten Kapitel wird, mit den im Kapitel 2 vorgestellten Technologien und Interaktionen, das Konzept für die prototypische Implementierung vorgestellt.

# Konzeption

In diesem Kapitel werden zunächst Kriterien definiert, die das Anwendungssystem im Rahmen dieser Abschlussarbeit erfüllen soll. Hierbei wird auf die Zielsetzung geachtet und Anwendungsfälle, sogenannte *Use Cases*, entwickelt, die ein Anwender mit der prototypischen Implementierung bestreiten können soll. Des Weiteren wird der Systementwurf genauer betrachtet. Dazu zählen insbesondere das Zusammenwirken der einzelnen Komponenten, welche Schichten abstrahiert werden und wie der Informationsfluss in den einzelnen Komponenten ist.

## 3.1 Spezifikation

Bei der zu entwickelnden Software handelt es sich um ein *User Interface*, dass den Anwender dazu befähigt einen Roboter mit Hilfe der Gesten- und Stimmenerkennung der *Microsoft HoloLens* über das Netzwerk fernzusteuern. Für die Verwendung wird ein Roboter, der eine Netzwerkschnittstelle besitzt, eine *Microsoft HoloLens*, die die Gestensteuerung erkennt und dem Anwender Hilfestellung in grafischer Form bietet sowie eine aktive, kabellose Netzwerkverbindung benötigt. Abbildung 35 beschreibt schematisch den Informationsfluss in der Anwendung.



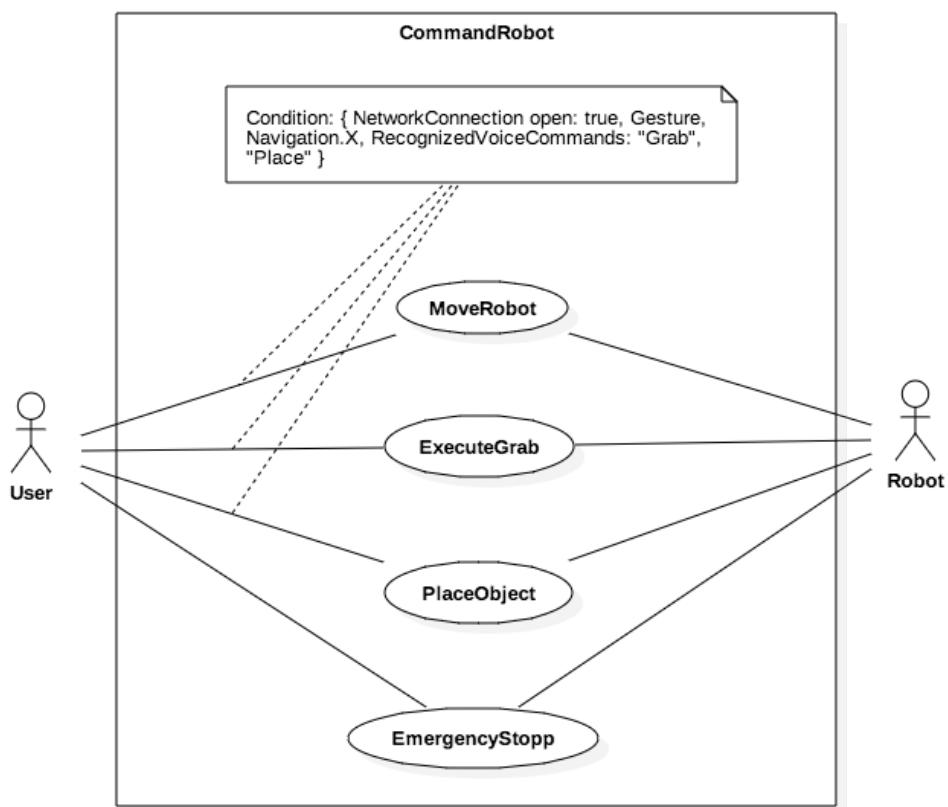
**Abb. 35.:** Informationsfluss in der Applikation *EV3ControlAR*. Bildquelle: Eigenes Werk.

### 3.1.1 Zielsetzung

Das Ziel der Applikation ist es, zu erfassen, ob und inwieweit Gesten und Sprachkommandos eine natürliche Interaktionsform und Schnittstelle für die in Zukunft verschmelzenden Welten von Robotern und Menschen sind und ob Anwender diese beim Lösen kleiner Aufgaben als intuitiv empfinden.

### 3.1.2 Anwendungsfall

Die funktionalen Anforderungen der Anwendung werden mit Hilfe von Anwendungsfällen ermittelt. Diese abstrahieren die Interaktion zwischen Benutzer und Anwendung und machen deutlich, was die Applikation aus Sicht des Benutzers leisten muss und inwiefern sie den Benutzer unterstützen kann. In Abbildung 36 sieht man das Anwendungsfalldiagramm in der *Unified Modeling Language*(UML) modelliert. Es zeigt die einzelnen Fälle der Applikation gebündelt in einem Diagramm.



**Abb. 36.:** UML Use-Case-Diagramm – CommandRobot. Bildquelle: Eigenes Werk.

Aus diesem wurden dann drei explizitere untergeordnete Anwendungsfälle aufgegliedert. Diese wurden nach dem Prinzip der *User Story* entwickelt. Die *User Story* beschreibt aus Sicht des Anwenders oder Kundens, was eine Software leisten soll und was der Anwender sich diesbezüglich wünscht.<sup>46</sup>

### Anwendungsfall 1 – MoveRobot – Roboter bewegen

»Als Anwender möchte ich den Roboter auf seiner Schiene frei bewegen können; dabei ist es mir wichtig, dass ich die volle Kontrolle über Geschwindigkeit und Position des Roboters habe.«

<b>Name</b>	Roboter bewegen (MoveRobot)
<b>Ziel im Kontext</b>	Roboter bewegt sich von Position A zu Position B
<b>Akteure</b>	Anwender und Roboter
<b>Trigger</b>	Anwender verschiebt 3D-Objekt in Szene
<b>Essentielle Schritte</b>	Richtiges 3D-Objekt auswählen Korrekte Geste zum Translatieren in X-Richtung ausführen 3D-Objekt freigeben
<b>Mögliche Erweiterungen</b>	Freies Translatieren ohne 3D-Objekt

**Tab. 3.1.:** Anwendungsfall 1 – »Roboter bewegen«.

### Anwendungsfall 2 – GrabObject – Objekt aufnehmen

»Als Anwender möchte ich mit Hilfe des Roboters Gegenstände in Reichweite aufheben; dabei ist es mir wichtig, dass der Roboter die Gegenstände präzise aufnimmt und bei Bewegungen in seinen Greifern behält.«

---

<sup>46</sup> Vgl. Wells, o.S., 1999, [Accessed: 04.01.2017].

<b>Name</b>	Mit Roboter Objekt greifen (GrabObject)
<b>Ziel im Kontext</b>	Greift ein Objekt mit Greifern
<b>Akteure</b>	Anwender und Roboter
<b>Trigger</b>	Anwender verschiebt 3D-Objekt in Szene
<b>Essentielle Schritte</b>	Richtiges 3D-Objekt auswählen Korrekte Geste zum Translatieren in Z-Richtung ausführen 3D-Objekt freigeben
<b>Mögliche Erweiterungen</b>	Freies Translatieren ohne 3D-Objekt

**Tab. 3.2.:** Anwendungsfall 2 – »Objekt aufnehmen«.

### Anwendungsfall 3 – PlaceObject – Objekt platzieren

»Als Anwender möchte mit Hilfe des Roboters einen aufgenommenen Gegenstand an einer beliebigen Position ablegen; dabei ist es mir wichtig, dass der Roboter die Gegenstände präzise, durch öffnen der Greifer, absetzt und sie zur Aufnahme neuer Objekte offen hält.«

<b>Name</b>	Mit Roboter Objekt platzieren (PlaceObject)
<b>Ziel im Kontext</b>	Roboterarm bewegt sich gen Boden und öffnet Greifer
<b>Akteure</b>	Anwender und Roboter
<b>Trigger</b>	Anwender aktiviert Sprachbefehl mit Stimme
<b>Essentielle Schritte</b>	Richtigen Sprachbefehl Korrekte Abfolge von Worten aussprechen
<b>Mögliche Erweiterungen</b>	Translatieren eines 3D-Objekts in Z-Richtung als Trigger

**Tab. 3.3.:** Anwendungsfall 3 – »Objekt platzieren«.

### Anwendungsfall 4 – EmergencyStop – Notstopp

»Als Anwender möchte ich den Roboter immer und mit sofortiger Wirkung anhalten können; dabei ist es mir wichtig, dass alle Motoren anhalten, Greifer und Arm sollten in ihrer bisherigen Position verweilen und der Bewegungsapparat zum Stehen kommen.«

<b>Name</b>	Notstopp (Stop)
<b>Ziel im Kontext</b>	Roboter stoppt sofort alle Motoren
<b>Akteure</b>	Anwender und Roboter
<b>Trigger</b>	Anwender aktiviert Sprachbefehl mit Stimme
<b>Essentielle Schritte</b>	Richtigen Sprachbefehl oder <i>Bloom</i> -Geste
<b>Mögliche Erweiterungen</b>	Notknopf am Körper.

**Tab. 3.4.:** Anwendungsfall 4 – »Notstopp«.

### 3.1.3 Muss-Kriterien

Die Anwendungsfälle resultieren in folgenden Funktionen, die dem Nutzer in der Applikation sofort und immer zur Verfügung stehen müssen:

- Gestenerkennung
- Kenntlichmachung, wenn Gesten angewendet werden können (Gesture-Ready)
- Unterscheidung von Gesten:
  - Tap-Geste um 3D-Objekte auszuwählen
  - Tap-And-Hold-Geste um 3D-Objekte als *draggable* auszuwählen
  - Translatieren von 3D-Objekten in X-Richtung
- Symbolisierung der oben genannten Gesten mit Icons
- Interface zur Übersicht des Roboters und dessen Funktionen
- **Notstopp**, sofortiger Stopp aller Motoren

### 3.1.4 Kann-Kriterien

Kann-Kriterien bezeichnen die Kriterien, deren Umsetzung sehr erstrebenswert wären. Allerdings sprengen sie im Kontext von Bachelorarbeiten meistens den zeitlichen Rahmen.

- Roboter kann sich in allen Freiheitsgraden bewegen
  - Rotationsgesten, ähnlich dem Drehen eines Drehreglers
  - *Eyetracking* zur schnelleren Lokalisierung von Punkten im Raum
  - Verbindung mit zusätzlichem *Wearables* für mehr Interaktionsspielraum und Nutzung von bereits bekannten Gesten, wie *Touch* oder Tasten
- Roboter bewegt sich frei im Raum, nicht auf Schiene
  - Wegfindung
  - Visualisierung des Pfades im HMD
  - Setzen von Wegpunkten aus dem HMD heraus
  - Erweiterte Routenoptionen
- Weitere Anwendungsszenarien

### 3.1.5 Abgrenzungskriterien

Die Abgrenzungskriterien stellen Kriterien dar, welche aufzeigen, was das System nicht leisten soll, um es klar von anderen Systemen abzugrenzen.

Die Grenzen dieses Produkts belaufen sich im Allgemeinen darauf, dass die Applikation nur ein Prototyp ist, welcher die Machbarkeit überprüfen und

das Gefühl der kontrollierten Steuerung übertragen soll. Sie belaufen sich des Weiteren auf folgende Punkte:

- Verwendbarkeit unter echten Produktionsbedingungen
- Verwendung in Echtzeit
- Synchronität von Gesten und Bewegung des Roboters
- Sicherheit
- Skalierbarkeit
- Simultane Verwendbarkeit eines Roboters durch mehrere Anwender
- Generische Nutzung von Maschinen

### 3.1.6 Nicht funktionale Kriterien

Was »nicht funktionale Kriterien« sind, ist nicht einheitlich definiert, jedoch ist allen Definitionen gemein, dass »nicht funktionale Kriterien« mit Metriken beschreiben, wie gut/zuverlässig ein System funktionieren soll.<sup>47</sup>

In der vorliegenden Arbeit wird besonderer Wert darauf gelegt, dass die Applikation *EV3ControllAR* folgende Kriterien erfüllt:

**Aussehen** Ebenso ist es für die Anwendung der vorliegenden Arbeit wichtig, dass sowohl das *Interface Design* als auch die Code-Struktur sauber, modern, übersichtlich und gut strukturiert ist. Dazu zählt auch die Wartbarkeit des Codes durch Lesbarkeit und fundierte Dokumentation.

**Benutzbarkeit** In der vorliegenden Arbeit wird ebenfalls hoher Wert darauf gelegt, dass die Anwendung eine leichte Verständlichkeit bietet, sodass sich dem Anwender die Benutzbarkeit erschließt und durch visuelle Signale gestützt wird. Außerdem sollte sie, soweit dies bei innovativen Interaktionsformen möglich ist, durch sich wiederholende, bekannte und bereits erlernte

---

<sup>47</sup> Vgl. Rupp, S. 268f, 2014.

Muster und Metaphern gefördert werden. Hinzu kommt die Reaktionszeit des Roboters, die unter einer Sekunde liegen soll.

### 3.1.7 Anwendungsumgebung

In diesem Bereich wird erläutert, in welcher Anwendungsumgebung die Applikation ihren Sinn findet und unter welchen Bedingungen sie läuft. Zunächst sei dargelegt, dass die prototypische Version nur unter Laborbedingungen arbeitet, wohingegen das Konzept sowohl für Werkshallen, als auch menschenfeindliche Arbeitsumgebungen, beispielsweise die Atmosphäre eines anderen Planeten, verseuchte oder unwegsame Gebiete der Erde, das All oder auch alle Umgebungen, bei denen ein ferngesteuerter Roboter einen Sinn ergibt, entwickelt wurde.

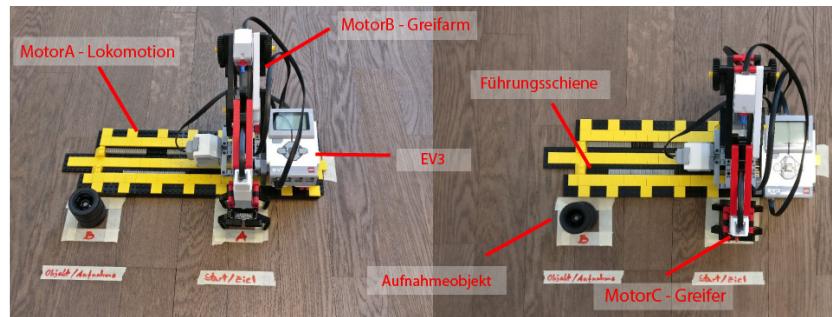
Die Anwendung an sich kann dank *Unity* für verschiedene Betriebssysteme kompiliert werden. Allerdings macht es nur Sinn, sie als Windows-App für die *HoloLens* zu *builden*, da die Schnittstellen des *HoloToolkit* nur die Datenbrille von Windows ansprechen. Für die Ausführung unter Laborbedingungen benötigt man folgende Hardware:

- *Microsoft HoloLens*
- *Lego Mindstorms EV3 mit WiFi USB-Dongle*
- Roboterarm aus *Lego*
  - MotorA - Translation im Raum
  - MotorB - Rotation um Armgelenk
  - MotorC - Greifer auf/zu
  - Führungsschiene für den Roboter
  - Objekt, das bewegt werden soll
- Router für ein lokales Netzwerk

Der Softwareteil beschränkt sich auf die Applikation *EV3ControllAR*.

- Client Applikation auf HoloLens

Abbildung 37 zeigt den Versuchsaufbau für die prototypische Implementierung.



**Abb. 37.:** Versuchsaufbau für die Applikation *EV3ControllAR*. Bildquelle: Eigenes Werk.

### 3.1.8 Anwendungsdaten

Dieser Abschnitt beschäftigt sich mit den Daten, die vom Anwender ins System eingepflegt werden müssen. Auch hier wird wieder zwischen Prototyp und Konzept unterschieden. So muss ein User beim Prototyp keine spezifischen Produktdaten eingeben. Allerdings ist es von höchster Priorität, dass ein Maschinenführer sich bei wertvollem Baustellenequipment und schweren Baumaschinen zur Nutzung autorisiert. Heutzutage läuft dieser Prozess meistens noch wie folgt ab: Eine Autorisierung erfolgt über optionale Schlüsselqualifikationen, wie beispielsweise einem Baggerführerschein, Staplerführerschein oder Drohnenführerschein. Die Scheine sind meistens kein »Muss«, begünstigen allerdings den Eindruck und die versicherungsspezifischen Bedingungen des Arbeiters. Die Scheine werden in Schulungen erworben. Die Qualifikation wird beim Arbeitgeber vorgelegt, welcher dann den Arbeiter zum Maschinenführer ernennt und ihm einen Schlüssel aushändigt, der den Arbeiter zum Starten der Maschine befähigt. Eine Schulung, wie oben beschrieben, soll es auch für virtuelle Interaktion geben. Im ausführten Thinkaloud-Test bekommen die Probanden beispielsweise eine Einführung in die Interaktionen. Allerdings könnte die Autorisierung an der Maschine über Stimme, Fingerabdruck, Iris-Scan oder andere innovative Authentifizierungsmechanismen stattfinden.

### 3.1.9 Anwendungsszenario

Dieser Abschnitt beschäftigt sich mit dem Anwendungsszenario der prototypischen Implementierung, aber auch mit konzeptionellen Erweiterungen.

Das Szenario hat als Hauptaufgabe, die Anwendung in den Kontext der Mensch-Maschine-Interaktion zu setzen. Deswegen bekommen die Testprobanden Hilfestellung in Form eines Tutoriums, welches die Gesten erklärt, und danach folgende Aufgabe, die es zu bewältigen gilt:

#### **Aufgabenstellung an die Testprobanden**

1. Orientieren Sie sich in der Szene mit Hilfe des Cursors und zeigen Sie mit diesem auf das Steuerelement.
2. Heben Sie eine Hand und bringen Sie sie in die *Gesture-Ready*-Position.
3. Bewegen Sie nun den Roboter mit Hilfe des Interaktionselements und der *Tap-And-Hold*- und *Translation*-Geste von Position A zum Aufnahmearm des Objektes.
4. Am Objekt angekommen aktivieren Sie über den Sprachbefehl »GRAB« die Aufnahme des Objekts durch den Roboterarm.
5. Bewegen Sie den Roboterarm nun zurück zu Position A.
6. Dort angekommen setzen Sie das Objekt ab, indem Sie den Sprachbefehl »PLACE« aussprechen.

Weitere mögliche Szenarien sind, dass Anwender versuchen Objekte zu stapeln oder zu sortieren. Außerdem könnte man erproben, das Objekt während der Fahrt aufzunehmen oder abzusetzen. Verändert man die Führungsschiene des Roboters oder gibt ihm einen anderen Antriebsmodus, wie Räder oder Ketten, kann man den Prototyp auch unter freieren Bedingungen testen. Tests im Gelände und Ferntests sind aus Sicht des Autors der nächste logische Schritt.

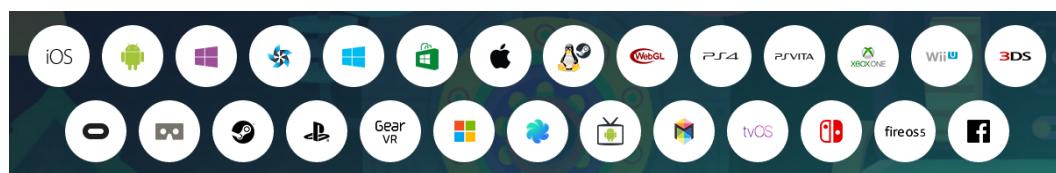
## 3.2 Systementwurf und entwicklungsspezifische Technologien

Diese Sektion gibt in erster Linie Einblicke in die grundlegende Interaktion mit dem System. Dazu werden die wichtigsten Funktionalitäten anhand von *Wireframes* und Systementwürfen erläutert. Im Anschluss folgt eine Einführung in die verwendeten Technologien und Bibliotheken. Der Teil dieser Applikation, der sich mit der Benutzerseite beschäftigt, ist stark von Interaktion und visuellen Reizen geprägt. Wie soll beispielsweise deutlich gemacht werden, welches Szenenobjekt wann bewegt werden soll und welche Möglichkeiten der Interaktion es gibt?

### 3.2.1 Unity 5.5f

Diese Sektion beschäftigt sich mit der *Game Engine Unity5.5f* und mit ihrer Beschaffenheit, sowie damit, welche Vorteile *Unity* bietet und weshalb die genannte Version gewählt wurde.

**Unity** »ist eine Laufzeit- und Entwicklungsumgebung für Spiele (Spiel-Engine) des Unternehmens Unity Technologies mit Hauptsitz in San Francisco«.<sup>48</sup> Das Unterstützen von Multiplattform-Kompilierung (siehe Abbildung 38) und einer breit gefächerten Entwickler- und Unterstützergemeinschaft verhalf *Unity* in den letzten Jahren dazu, die beliebteste Spiele-Engine der Welt zu werden.<sup>49</sup>

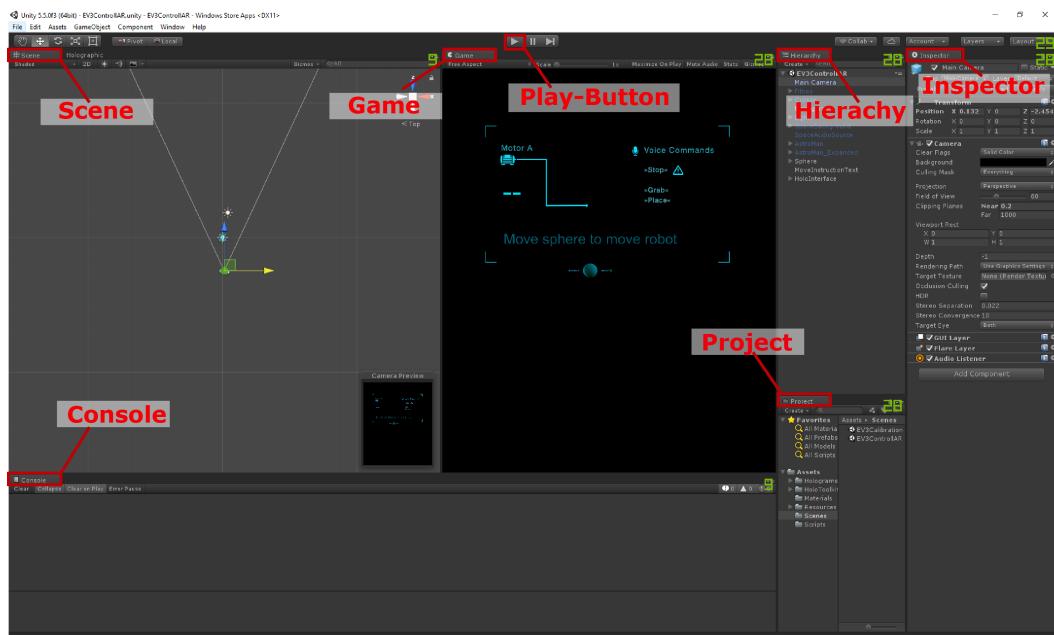


**Abb. 38.:** Unity Multiplattform-Unterstützung – Übersicht der unterstützten Plattformen. Bildquelle: [unity].

<sup>48</sup> Wikipedia, Unity (Spiel-Engine), [Accessed: 14.02.2017].

<sup>49</sup> Unity - Anonymous, Unternehmensfakten, 2016, [Accessed: 14.02.2017].

**Entwicklungsumgebung** Die Hauptansicht von *Unity*, der *Editor*, beinhaltet die Unterfenster *Scene*, in der die 3D-Szene durch *GameObjects* manipuliert werden kann; diese *GameObjects* werden in einem Szenengraph organisiert, der im Panel *Hierarchy* dargestellt wird und dessen Attribute wiederum im Panel *Inspector* angezeigt werden und dort auch verändert werden können. Sie werden mit *Components* bestückt, um ihnen bestimmte Funktionen zu geben. Sobald der *Play-Button* aktiviert wurde, zeigt das Fenster *Game* das Spiel aus der Sicht der aktiven Kamera. Man nutzt dieses zum Testen seiner Applikation, es ermöglicht schnelle Iterationszyklen, da das Programm nicht nach jeder Änderung kompiliert und auf das Zielgerät übertragen werden muss. In Abbildung 39 sieht man ein solches Unity-Editor-Fenster und seine Paneele auf einen Blick. Mit dem *Scene*- und *Hierarchy*-Fenster lassen sich durch bereits vorgefertigte *Components* und viele, von anderen Leuten bereits angefertigte *Assets* aus dem *Assetstore*, auf Anhieb kleinere Spiele, auch ohne Programmierkenntnisse, fertigen.



**Abb. 39.:** Unity Entwicklungsumgebung – Übersicht des Editors. Bildquelle: Eigenes Werk.

**Skripts** Möchte man allerdings seiner Applikation mehr Funktionalität geben, tut man dies durch das Anhängen eigener *Scripts* als *Component* an die *GameObjects*. Die Skriptsprache ist *C#* oder *Unity-Script*, welches an *JavaScript* angelehnt ist. Der *Inspector* von *Unity* interagiert mit den Variablen aus den Skripten. Deklariert man beispielsweise eine Variable mit der Zugriffs-

klasse *public* wird diese im *Inspector* angezeigt und man kann sie direkt aus *Unity* per *Drag and Drop* initialisieren.

**Grafik** *Unitys Grafikpipeline* bedient die *Rendering-Pfade*:

- *Forward Rendering*
- *Deferred Shading*
- *Legacy Deferred Lighting*
- *Legacy Vertex Lit*
- *Shadows*

*Shader*, welche in der *High-Level-Shader-Language*(HLSL) geschrieben werden, bestimmen wie Objekte in der Szene aussehen, indem sie die Materialeigenschaften und die Beleuchtung dazu berechnen.<sup>50</sup>

Das besondere an der Version 5.5f ist, dass sie *HoloLens Ready* ist. In ihr wurden die Versionen *Unity 5.4.0f3-HTP*, was für den Entwicklungszweig die *HoloLens* betreffend steht, sowie der ursprüngliche Entwicklungspfad von *Unity* zusammengeführt und zusätzliche Zusatzfunktionen implementiert, die das Entwickeln mit und für die *HoloLens* erheblich vereinfachen und beschleunigen. Dazu zählen beispielsweise das Starten des entwickelten Projekts direkt aus dem *Editor*, was die Iterationszyklen, gegenüber dem Kompilieren und Übertragen des Programms per *USB* oder *WiFi* auf die *HoloLens*, erheblich beschleunigt, sowie das Bereitstellen von Funktionalitäten des *HoloToolkit*, welches in der folgenden Sektion 3.2.2 genauer betrachtet wird.

### 3.2.2 HoloToolkit

Das *HoloToolkit* ist »eine Kollektion von Skripten und Komponenten, welche beabsichtigt die Entwicklung von holografischen Applikationen für Win-

---

<sup>50</sup> Unity, Rendering Pipeline <https://docs.unity3d.com/Manual/SL-RenderPipeline.html>.

*dows Holographics* zu beschleunigen.«<sup>51</sup> Das *Toolkit* beinhaltet viele nützliche Schnittstellen<sup>52</sup> für die *HoloLens*, grob gegliedert in:

1. Input
2. Sharing
3. Spatial Mapping
4. Spatial Understanding
5. Spatial Sound
6. Utilities
7. Build

Die vorliegende Arbeit beschäftigt sich und nutzt hauptsächlich die Inhalte der Sektion *Input*, im folgenden Abschnitt werden seine Bestandteile genauer betrachtet.

**Input** Das *Input*-System beinhaltet ein voll funktionsfähiges Eingabe-Modul, dies erlaubt dank einer Implementation basierend auf Schnittstellen ein erweiterbares Modul, welches die Hauptcharakteristiken *Gaze*, *Gesture* und *Voice* abdeckt. Es liefert vorgefertigte Objekte und Skripte mit, die zum einen dafür sorgen, dass Grundfunktionen wie *Gaze* immer gleich, beziehungsweise ähnlich implementiert werden und zum anderen die Entwicklung beschleunigen.

### 3.2.3 MonoBrick API

Die *MonoBrick API* ist ein Paket, das aus der *MonoBrick EV3 Firmware* entstanden ist, welche es erlaubt den *EV3* mit dem *opensource .Net Framework* namens *Mono* programmiert und *debuggt* werden kann.<sup>53</sup> Sie wurde von Anders Soborg und Lars Jeppesen entwickelt und macht sie für ein *Unity*-Projekt

---

<sup>51</sup> Microsoft, HoloToolkit Repository, 2016, [Accessed: 15.02.2017].

<sup>52</sup> Microsoft, HoloToolkit Repository, 2016, [Accessed: 15.02.2017].

<sup>53</sup> Vgl. Soborg / Jeppesen, o.J., [Accessed: 22.02.2017].

besonders wertvoll, da durch die .Net-Umgebung in C# entwickelt werden kann. Die Schnittstelle *MonoBrick Communication Library* agiert dabei als *Wrapper* für die *LEGO Mindstorms Communication Library* in C#, welche mit der Standard-Firmware des EV3 funktioniert. Allerdings musste für die Nutzung mit *Unity* die Bibliothek zunächst herunterkompiliert werden, da *Unity* momentan nur die .Net 2.0-Umgebung unterstützt, die *MonoBrick Communication Library* allerdings für die .Net 4.0-Umgebung entwickelt wurde. Die Entwicklungsumgebung *Microsoft Visual Studio 2015 professional* lässt ein solches Abstufen von .dll Paketen zu und unterstützt den Entwickler dahingehend im *Build*-Menü ein Ziel-Framework auszuwählen.

# Prototypische Implementierung einer Augmented Reality Applikation zur Steuerung eines Roboters

Dieses Kapitel beschäftigt sich mit der prototypischen Implementierung einer *Augmented Reality* Applikation, die als Teleoperator für einen Roboter dient. Es wird auf die Architektur der Anwendung eingegangen, welche sowohl die Assoziationen der einzelnen Systeme aufzeigt, als auch die konkrete Implementierung der Komponenten: *User Interface*, Befehlsvermittlung an den Roboter, Implementierung der Aktionen der Maschine sowie Implementierung der Gesten und Sprachkommandos.

## 4.1 Kalibrierung der Szene

Die Kalibrierung der Szene erfolgt über eine manuelle Kalibrierung. Der Anwender setzt mit Hilfe des *Cursors* zwei Punkte an Anfang und Ende der Führungsschiene des Roboters. Die dabei entstehenden Objekte sind gleichzeitig Ankerpunkte für die Translationsgeste. Zwischen ihnen kann später das Interaktionsobjekt hin und her bewegt werden. Die Positionen des linken und rechten Ankerpunkts werden gespeichert und der Mittelpunkt der Strecke zwischen ihnen bildet den Ausrichtungspunkt für das *User Interface* »HoloInterface«.

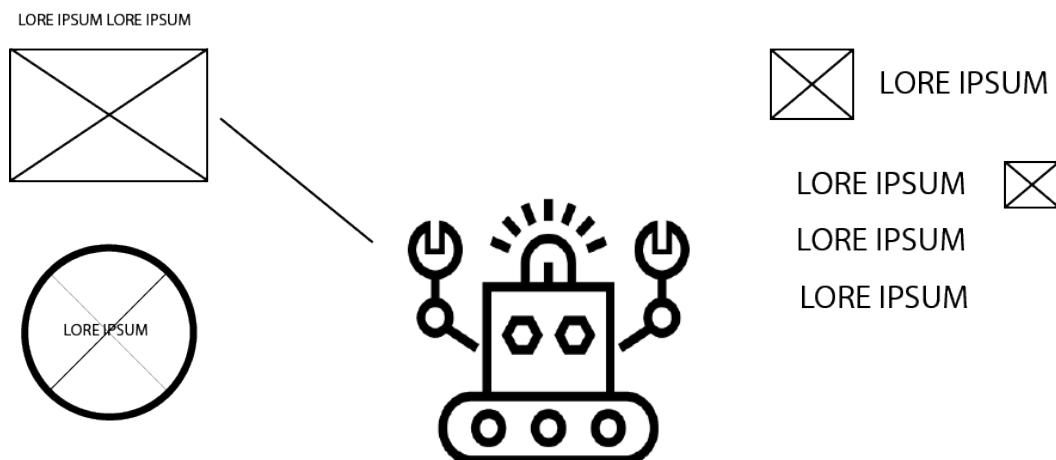
## 4.2 Realisierung der Interaktionen

Die Interaktionen werden mit Hilfe des *HoloToolkit* realisiert. Es bietet Schnittstellen für alle vorgestellten Gesten. Der weitere Vorgang orientiert sich grob an der *Pipeline* aus Abbildung 35.

Startet man die Applikation und durchläuft die Kalibrierungsszene erfolgreich, werden im Hauptobjekt der Steuerungsszene, welches den simplen Namen »Manager« trägt, die Raum-, Sprach- und Gestenerkennung registriert sowie der *Wrapper* für die EV3-Kommunikationsbibliothek gestartet. Sie wurden jeweils in die Skripte »Voice«-, »Gesture«- und »EV3«-Manager modularisiert.

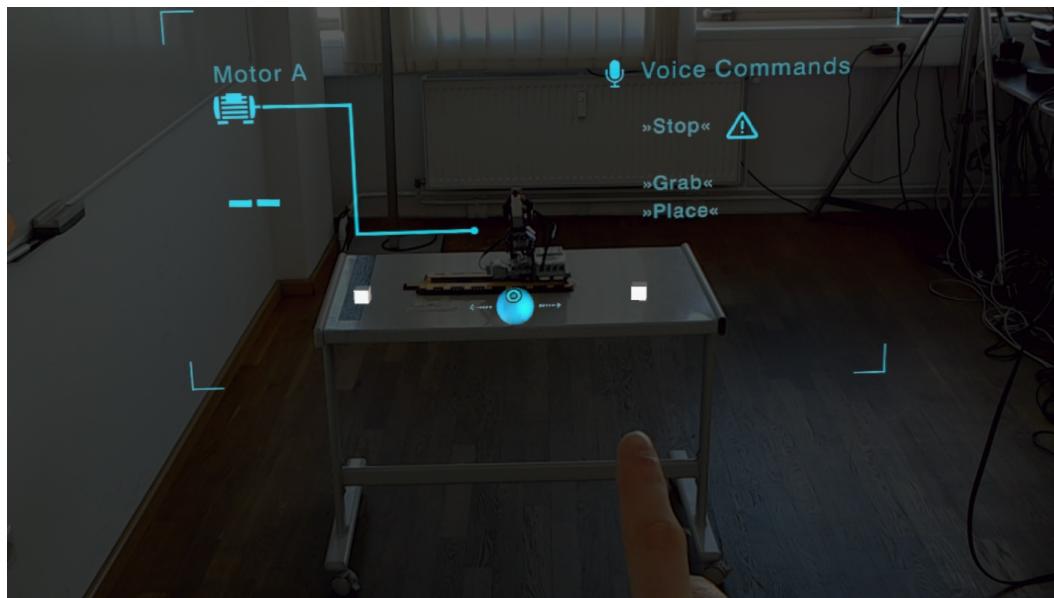
### 4.2.1 User Interface

Das *User Interface* wurde ebenfalls nach den, bereits aus Sektion 2.3.2 bekannten, »ASSEFIL-Kriterien« als *Wireframe* gestaltet. Der Entwurf dazu ist in Abbildung 40 zu sehen.



**Abb. 40.:** HoloInterface – *Userinterface* als *Wireframe* Entwurf. Bildquelle: Eigenes Werk.

Er wurde in Anbetracht dessen gestaltet, dass ein Anwender die wichtigsten Informationen den Roboter betreffend während der Interaktion mit ihm vor sich in einem *Headup-Display* wahrnehmen kann, ohne dabei zu sehr von seiner Aufgabe abgelenkt zu werden.



**Abb. 41.:** HoloInterface – *User Interface* für die Applikation »EV3ControllAR«. Bildquelle: Eigenes Werk.

Das *User Interface* beinhaltet die wichtigsten Informationen auf einen Blick. Zu den essentiellen Informationen zählen: Wie schnell sich »MotorA« dreht. Wobei das Vorzeichen die Drehrichtung des Motors angeibt und dabei angibt, ob der Roboter auf seiner Schiene nach rechts oder links fährt. Zusätzlich wird eine Liste der Sprachkommandos angezeigt, die der Anwender benötigt, um den Greifarm des Roboters zu kontrollieren. Es wurde das HCI-Kriterium *Chunking* beachtet, welches vorgibt, nicht mehr als sieben Elemente in einer Gruppierung zu platzieren, da das menschliche Gehirn nur maximal sieben *Chunks* gleichzeitig aufnehmen kann.<sup>54</sup>

#### 4.2.2 Befehlsvermittlung an die Maschine

Die Befehlsvermittlung an die Maschine wurde mit der Kommunikationsbibliothek *MonoBrick* realisiert. Was *MonoBrick* ist und wie es aufgebaut ist, kann in der Sektion 3.2.3 nachgeschlagen werden. *MonoBrick* erlaubt es eine Verbindung zum *EV3*-Baustein über Netzwerk zu realisieren, wenn ein kompatibler *WiFi-Dongle* im *USB-Port* des *EV3*-Baustein eingesteckt wurde. Allerdings muss der *EV3* zunächst überhaupt in die Lage versetzt werden

---

<sup>54</sup> Vgl. Gobet / Fernand / Lane et al., S. 236 ff., 2001.

Befehle von außen zu erhalten und eine TCP/IP (Transmission Control Protocol/Internet Protocol) Verbindung herzustellen:<sup>55</sup>

1. Mit einem Protokollprogramm (*Wireshark*) auf Port 3015 lauschen, denn der EV3 sendet ungefähr im 10-Sekudentakt eine UDP (User Datagram Protocol) Übertragung.
2. Diese enthält die Seriennummer des EV3 (siehe Abbildung 42).
3. Hat man die Seriennummer notiert, muss man eine beliebige UDP-Nachricht an den EV3 senden.
4. Danach ist der Stein bereit TCP/IP-Nachrichten zu empfangen.
5. Jetzt folgt die Freigabe-Nachricht via TCP/IP:
  - GET /target?sn=0016534e474f VMTP1.0 Protocol: EV3
6. Antwortet der Stein mit einer 16 Byte langen TCP/IP-Nachricht mit dem Inhalt Accept:EV340 ist der Stein freigeschaltet und der obige Code lässt die Kommunikation zwischen EV3 und Applikation via TCP/IP zu.

```

> Frame 223: 109 bytes on wire (872 bits), 109 bytes captured (872 bits) on interface 0
> Ethernet II, Src: EdimaxTe_02:71:fa (74:da:38:02:71:fa), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 192.168.0.103, Dst: 192.168.0.255
> User Datagram Protocol, Src Port: 55061, Dst Port: 3015
  Data (67 bytes)
    Data: 53657269616c2d4e756d6265723a20303031363533346534...
    [Length: 67]

0000 ff ff ff ff ff ff 74 da 38 02 71 fa 08 00 45 00 .....t. 8.q...E.
0010 00 5f 00 00 40 00 40 11 b7 d7 c0 a8 00 67 c0 a8 ..-.@. ....g..
0020 00 ff d7 15 0b c7 00 4b 16 c6 53 65 72 69 61 6c .....K .Serial
0030 2d 4e 75 6d 62 65 72 3a 20 30 31 36 35 33 34 -Number: 0016534
0040 65 34 37 34 66 0d 0a 50 6f 72 74 3a 20 35 35 35 e474f..P ort: 555
0050 35 0d 0a 4e 61 6d 65 3a 20 45 56 33 0d 0a 50 72 5.Name: EV3..Protocol: EV3..
0060 6f 74 6f 63 6f 6c 3a 20 45 56 33 0d 0a

```

**Abb. 42.:** UDP-Paket des EV3 im Netzwerk, welches die Seriennummer enthält.  
Bildquelle: Eigenes Werk.

Ist erst einmal eine Verbindung geöffnet und initialisiert, kann mit dem *EV3*-Baustein über die Schnittstelle von *MonoBrick* kommuniziert werden. Folgen-

<sup>55</sup> Vgl. MonoBrick, Guides, 2016. [Accessed: 15.01.2017].

der Codeabschnitt zeigt die Deklaration, Initialisierung und Aktivierung der Verbindung zwischen dem *EV3* und der *Wrapper-Klasse »EV3-Manager« mit Hilfe der *MonoBrick*-Kommunikationsbibliothek:*

```
1 using UnityEngine;
2 using MonoBrick.EV3;
3
4 public class EV3Manager : MonoBehaviour {
5
6     public Brick<TouchSensor, Sensor, Sensor, Sensor>;
7     [...]
8
9     void Start() {
10         ev3 = new Brick<TouchSensor, Sensor, Sensor, Sensor>("WiFi");
11         ec3.Connection.Open();
12         [...]
13     }
14     [...]
15 }
```

---

Die Bewegungsabfolgen des *EV3* wurden ebenfalls im *EV3-Manager* implementiert. Sie orientieren sich an den vier *User Stories* aus Sektion 3.1.2.

**MoveRobot** Übermittelt die Geschwindigkeit des MotorsA, welcher für die Translation auf der Schiene zuständig ist als Wert vom Typ *sbyte*. Die Methode kann einen Wert zwischen -100 und 100 annehmen, was die maximale Geschwindigkeit des Servomotors in beide Richtungen angibt.

```
1 public void MoveRobot(sbyte speed) {
2     ev3.MotorA.On(speed);
3 }
```

---

**ExecuteGrab** Beläuft sich auf drei Aktionen des Roboterarms: Den Arm nach unten bewegen; Greifer schließen; Arm nach oben bewegen. Wird dann der Berührungssensor am Roboterarm aktiviert hält der Arm seine Position.

```
1 public void GrabObject() {
2     WaitForMotorToStop();
3     ev3.MotorB.Off();
4     WaitForMotorToStop();
5     ev3.MotorC.MoveTo(10, 80, true);
```

```
6     WaitForMotorToStop();
7     ev3.MotorB.On(-25);
8     MotorBlocked = true;
9 }
```

---

Die *Update*-Methode prüft dabei den Zustand des Berührungssensors.

```
1 if (ev3.Sensor1.Read() == 1 && MotorBlocked == true) {
2     ev3.MotorB.Brake();
3     MotorBlocked = false;
4 }
```

---

**PlaceObject** Diese Methode verhält sich analog zur *Grab*-Methode und bedarf keiner weiterer Erklärung.

**EmergencyStopp** Die Methode stoppt sofort alle Motoren und lässt den Roboter in seiner Position verharren.

```
1 ev3.MotorA.Off();
2 ev3.MotorB.Off();
3 ev3.MotorC.Off();
```

---

### 4.2.3 Implementierung der Gesten

Gesten sind, wie in Kapitel aufgezeigt, eine der drei Hauptformen, um mit Hologrammen zu interagieren. Zunächst muss jedoch ein Hologramm anvisiert werden. In der Applikation *EV3ControllerAR* ist dafür der *GazeManager* verantwortlich. Er generiert einen *Raycast*<sup>56</sup>, welcher aus der Mitte der Datenbrille abgefeuert wird und somit der Kopfbewegung und im weiteren Sinne dem Blick des Trägers folgt. Wurde ein Hologramm mit dem *Cursor* anvisiert, wird die Position und Normale des vom *Raycast* des *Cursors* getroffenen *GameObjects* zurückgegeben und die Ergebnisse können weiterverarbeitet werden. Folgender Codeausschnitt aus Anhang A.1 zeigt die Implementierung des *Cursors*:

```
1 [...]
2     gazeOrigin = Camera.main.transform.position;
```

<sup>56</sup> Raycasting ist der Prozess des Schießens eines unsichtbaren Strahls von einem Punkt aus in eine spezifische Richtung und des Detektierens von Schnittpunkten des Strahls mit *Collidern* in der Szene.

```

3   gazeDirection = Camera.main.transform.forward;
4   gazeStabilizer.UpdateHeadStability(gazeOrigin, Camera.main
5       .transform.rotation);
6   gazeOrigin = gazeStabilizer.StableHeadPosition;
7   UpdateRaycast();
8
9   /// <summary>
10  /// Calculates the Raycast hit position and normal.
11  /// </summary>
12  private void UpdateRaycast() {
13      RaycastHit hitInfo;
14      Hit = Physics.Raycast(gazeOrigin, gazeDirection, out
15          hitInfo, MaxGazeDistance, RaycastLayerMask);
16      HitInfo = hitInfo;
17      if (Hit) { // If raycast hit a hologram...
18          Position = hitInfo.point;
19          Normal = hitInfo.normal;
20      }
21  [...]

```

---

Wurde ein Hologramm getroffen, ist es sinnvoll, die Gestalt des *Cursors* zu ändern und eine Nachricht an das getroffene Hologramm zu senden, sodass der Benutzer ein *User Feedback* bekommt, welches weitere Aktionsmöglichkeiten visualisiert sowie Befehle an das getroffene Hologramm gesendet werden können. Dafür zuständig sind die Skripte **InteractableManager** und **CursorManager** sowie **Interactable**. Die Hauptaufgabe des *InteractableManagers* ist es ein *GameObject* zu erstellen und dieses mit der *hitInfo* des *GazeManagers* anzureichern. Die folgenden Codeabschnitte zeigen dieses Verhalten und beleuchten kurz die Änderung des *Cursor-Icons*. Wie im Anhang A.2 ab Zeile 29 zu sehen, entscheidet diese Klasse darüber, welcher *Cursor* aktiv sein soll, basierend darauf, ob ein Hologramm getroffen wird oder nicht. Des Weiteren übernimmt sie das setzen der *Cursor-Position*:

```

1  [...]
2  if (GazeManager.Instance == null || CursorOnHolograms ==
3      null || CursorOffHolograms == null) {
4      return;
5
6  if (GazeManager.Instance.Hit) {
7      CursorOnHolograms.SetActive(true);
8      CursorOffHolograms.SetActive(false);
9  } else {
10      CursorOffHolograms.SetActive(true);

```

```

11     CursorOnHolograms.SetActive(false);
12 }
13
14 gameObject.transform.position = GazeManager.Instance.
15     Position;
16 gameObject.transform.up = GazeManager.Instance.Normal;
17 [...]

```

---

Der Codeausschnitt aus Anhang A.3 ist für das Umschalten des Cursors und das Senden von Nachrichten an das getroffene *GameObject* zuständig.

```

1 [...]
2 if (hitInfo.collider != null) {
3     // Assign the hitInfo's collider gameObject to the
4     // FocusedGameObject.
5     FocusedGameObject = hitInfo.collider.gameObject;
6 }
7 [...]
8 if (FocusedGameObject != null) {
9     if (FocusedGameObject.GetComponent<Interactable>() != null
10         ) {
11         // Send a GazeEntered message to the FocusedGameObject.
12         FocusedGameObject.SendMessage("GazeEntered");
13     }
14 }
15
16 private void ResetFocusedInteractable() {
17     if (oldFocusedGameObject != null) {
18         if (oldFocusedGameObject.GetComponent<Interactable>() != null) {
19             // Send a GazeExited message to the
20             // oldFocusedGameObject.
21             oldFocusedGameObject.SendMessage("GazeExited");
22         }
23     }
24 }
25 [...]

```

---

Ist der *Cursor* und dessen Verhalten implementiert, folgt als nächstes die Gestenerkennung. Zur Gestenerkennung gehören die Skripte **HandsManager**, **GestureManager** und **GestureAction**. Im *Handsmanager* meldet man die Ereignisse *SourceDetected* und *SourceLost* mit folgenden Codezeilen an und wechselt das *Cursor-Asset* dementsprechend aus:

```

1 [...]
2 InteractionManager.SourceDetected +=
    InteractionManager_SourceDetected;
3 InteractionManager.SourceLost +=
    InteractionManager_SourceLost;
4
5 /// Register for SourceManager.SourcePressed event.
6 InteractionManager.SourcePressed +=
    InteractionManager_SourcePressed;
7
8 /// Register for SourceManager.SourceReleased event.
9 InteractionManager.SourceReleased +=
    InteractionManager_SourceReleased;
10
11 [...]
12 private void InteractionManager_SourceDetected(
    InteractionSourceState hand) {
13     HandDetected = true;
14 }
15
16 private void InteractionManager_SourceLost(
    InteractionSourceState hand) {
17     HandDetected = false;
18     ResetFocusedGameObject();
19 }
```

---

Danach wird im *GestureManager* festgelegt, welche Gesten erkannt werden sollen; in Fall von *EV3ControllerAR* Applikation sind das die Navigation auf X-Ebene zur Verschiebung des Interaktionselements und die *Tap*-Geste:

```

1 /// Instantiate the NavigationRecognizer.
2 NavigationRecognizer = new GestureRecognizer();
3
4 /// Add Tap and NavigationX GestureSettings to the
5 // NavigationRecognizer's RecognizableGestures.
6 NavigationRecognizer.SetRecognizableGestures(GestureSettings
    .Tap | GestureSettings.NavigationX);
```

---

Das an das Interaktionsobjekt angehängte Skript *GestureAction* behandelt den Sachverhalt, dass der Roboter bewegt werden soll, wann immer das Interaktionsobjekt translatiert wird. Dazu rechnet man den Translationsfaktor in die Handbewegung ein und überträgt eine daraus passende Geschwindigkeit an den Motor des Roboters, welcher für die Lokomotion zuständig ist;

wird während der Translation die Navigationsgeste unterbrochen, stoppt der Roboter.

```
1 [...]
2 public float TranslationSensitivity = 10.0f;
3 private float translationFactor;
4 [...]
5
6 void Update(){
7     PerformTranslation();
8 }
9
10 private void PerformTranslation() {
11     if (GestureManager.Instance.IsNavigating && HandsManager.
12         Instance.FocusedGameObject == gameObject) {
13         // Calculate translationFactor based on GestureManager's
14         // NavigationPosition.X and multiply by
15         // TranslationSensitivity.
16         // This will help control the amount of translation.
17         translationFactor = GestureManager.Instance.
18             NavigationPosition.x;
19
20         // Restrict the translationFactor to a value between -1
21         // and 1, so the robot doesn't get too fast.
22         float clampTranslation = Mathf.Clamp(translationFactor, -1
23             , 1f);
24         positionManager.SetPosition(clampTranslation / 2 + 0.5f);
25         ev3.MoveRobot((sbyte)(clampTranslation * 10));
26     } else {
27         positionManager.SetPosition(0.5f);
28         ev3.StopRobot();
29     }
30 }
```

---

Zusätzlich wurden Sprachkommandos implementiert, die den Greifarm des Roboters steuern. Sie werden zusammen mit ihrer Implementation genauer in der nächsten Sektion 4.2.4 behandelt.

#### 4.2.4 Implementierung der Sprachkommandos

Drei der vier Anwendungsfälle wurden mit Sprachkommandos realisiert: Dabei handelt es sich um die Anwendungsfälle *GrabObject* aus Tab. 3.2,

*PlaceObject* aus Tab. 3.3 sowie *Stop* aus Tab. 3.4. Die Realisierung von Sprachbefehlen in *Unity* kann auch drei Art und Weisen gestaltet werden:

1. PhraseRecognizer
2. GrammaerRecognizer
3. DictationRecognizer

Bei der vorliegenden Arbeit wurde der *KeywordRecognizer* als Spracheingabe gewählt und im Skript **VoiceManager** implementiert, da er sich für die Anwendungsfälle, welche klare Befehle an eine Maschine übertragen sollen, am besten eignet, da nur einzelne Schlüsselwörter erkannt und auf diese reagiert werden soll. Zunächst muss spezifiziert werden, auf welche Schlüsselwörter der *KeywordRecognizer* hören soll und die Aktion definiert werden, die nach Erkennung durchgeführt werden soll. Folgender Code aus Anlage A.5 ist dafür zuständig.

```
1 [...]  
2 /// Initialize...  
3 KeywordRecognizer keywordRecognizer;  
4 Dictionary<string, System.Action> keywords = new Dictionary<  
    string, System.Action>();  
5 [...]  
6 /// Create keywords and add them to dictionary...  
7 keywords.Add("grab", () => {  
8     ev3.GrabObject();  
9 });  
10 keywords.Add("place", () => {  
11     ev3.PlaceObject();  
12 });  
13 keywords.Add("stop", () => {  
14     ev3.Stop();  
15 });  
16 [...]  
17 /// Tell KeywordRecognizer which words to recognize.  
18 keywordRecognizer = new KeywordRecognizer(keywords.Keys.  
    ToArray());  
19 /// Register for OnPhraseRecognized event  
20 keywordRecognizer.OnPhraseRecognized +=  
    KeywordRecognizer_OnPhraseRecognized;  
21 /// Start recognizing...  
22 keywordRecognizer.Start();  
23
```

```
24 private void KeywordRecognizer_OnPhraseRecognized(
25     PhraseRecognizedEventArgs args) {
26     Debug.Log("Keyword Recognized..." + args.text);
27     System.Action keywordAction;
28     /// Fire action depending on which keyword was recognized.
29     if (keywords.TryGetValue(args.text, out keywordAction)) {
30         keywordAction.Invoke();
31     }
32 }
```

---

# Usability Test

## 5.1 Think Aloud Test

Der *Think Aloud*-Test, manchmal auch *Thinking Aloud*-Test genannt, ist ein simples *Usability*-Testverfahren, welches die Testprobanden dazu anhält: »das System zu benutzen, während kontinuierlich laut gedacht wird — also die Gedanken zu verbalisieren, die aufkommen, während sich der Nutzer durch das *User Interface* bewegt.«<sup>57</sup> Der *Think Aloud*-Test ist ein qualitatives Testverfahren, das heißt es werden nur wenige und dafür ausgewählte Testprobanden benötigt.

**Testaufbau** Für den Test in der vorliegenden Arbeit wurden folgende Testprobanden ausgesucht:

- Zwei Testprobanden aus dem Bereich des Büromanagements
- Zwei Testprobanden aus dem Bereich der Gestaltung
- Zwei Testprobanden aus dem Bereich des Projektmanagements
- Zwei Testprobanden aus dem Bereich der Entwicklung

Die Testpersonen hatten keine bis wenig Vorkenntnisse im Umgang mit der *Microsoft HoloLens* und es wurde darauf geachtet jeweils eine männliche und eine weibliche Testperson auszusuchen. Als nächstes muss für Ruhe gesorgt werden, den Testern einzeln die Aufgabe aus Sektion 3.1.9, die es zu erfüllen gilt, erklärt werden sowie die Gesten einmalig vorgeführt werden.

---

<sup>57</sup>Nielsen, S. 196, 1993, [Accessed: 21.02.2017].

## 5.2 Auswertung

Der Test aus Sektion 5.1, welcher sich transkribiert im Anhang A.7 befindet, liefert im Kern die folgenden Ergebnisse:

<b>Testperson</b>	männlich
<b>Bereich</b>	Büromanagement
<b>Vorkenntnisse</b>	keine
<b>Eindruck</b>	Gut
<b>Hauptanmerkung</b>	Deutlicher machen, wie genau man den Roboter bewegt

**Tab. 5.1.:** Testperson 1 – Büromanagement, männlich.

<b>Testperson</b>	weiblich
<b>Bereich</b>	Büromanagement
<b>Vorkenntnisse</b>	keine
<b>Eindruck</b>	schlecht
<b>Hauptanmerkung</b>	<i>Cursor-Hand-Koordination schwierig. Frustration, weil Geste nicht funktioniert</i>

**Tab. 5.2.:** Testperson 2 – Büromanagement, weiblich.

<b>Testperson</b>	männlich
<b>Bereich</b>	Gestaltung
<b>Vorkenntnisse</b>	keine
<b>Eindruck</b>	sehr gut
<b>Hauptanmerkung</b>	Wenig Latenz; funktioniert erstaunlich gut; »wie Magie«; Interface übersichtlich; <i>Gaze</i> zusammen mit Hand schwierig

**Tab. 5.3.:** Testperson 3 – Gestaltung, männlich.

<b>Testperson</b>	weiblich
<b>Bereich</b>	Gestaltung
<b>Vorkenntnisse</b>	keine
<b>Eindruck</b>	sehr gut
<b>Hauptanmerkung</b>	Schnelle Reaktion; funktioniert besser als gedacht; Interface geordnet

**Tab. 5.4.:** Testperson 4 – Gestaltung, weiblich.

<b>Testperson</b>	männlich
<b>Bereich</b>	Projektmanagement
<b>Vorkenntnisse</b>	wenige Vorkenntnisse
<b>Eindruck</b>	sehr gut
<b>Hauptanmerkung</b>	Nicht intuitivste Steuerung; hatte angenommen das Interaktionsobjekt bildet absolute Positionswerte ab, nicht relative; wie Magie; Interface übersichtlich; <i>Gaze</i> zusammen mit Hand schwierig

**Tab. 5.5.:** Testperson 5 – Projektmanagement, männlich.

<b>Testperson</b>	weiblich
<b>Bereich</b>	Projektmanagement
<b>Vorkenntnisse</b>	keine Vorkenntnisse
<b>Eindruck</b>	sehr gut
<b>Hauptanmerkung</b>	Hatte Spaß bei der Bedienung; erstaunt, dass Roboter sich durch Geste bewegt hat; Interface ansprechend; Stimme leider nicht gut erkannt

**Tab. 5.6.:** Testperson 6 – Projektmanagement, weiblich.

<b>Testperson</b>	weiblich
<b>Bereich</b>	Entwicklerin
<b>Vorkenntnisse</b>	keine Vorkenntnisse
<b>Eindruck</b>	sehr gut
<b>Hauptanmerkung</b>	<i>HoloLens</i> -Gesten allgemein schwer und ungewohnt; Feedback für <i>Tap-And-Hold</i> -Geste gewünscht; wenn Geste bekannt Aufgabe einfach zu bewältigen; eventuell Aufgabe in <i>Interface</i> Schritt für Schritt abbilden

**Tab. 5.7.:** Testperson 7 – Entwicklung, weiblich.

<b>Testperson</b>	männlich
<b>Bereich</b>	Entwickler
<b>Vorkenntnisse</b>	keine Vorkenntnisse
<b>Eindruck</b>	gut
<b>Hauptanmerkung</b>	Einfaches, gut verständliches Interface; Kontrollelement jedoch zu tief; Interaktionsobjekt auf Höhe des HoloInterface gewünscht

**Tab. 5.8.:** Testperson 8 – Entwicklung, männlich.

Fasst man die Testergebnisse zusammen, lässt sich erschließen, dass bei technisch problemfreiem Ablauf, das heißt keine Blockade der Motoren auf Grund von Ungenauigkeiten im Tacho des *Lego Servomotors*, die Testprobanden ihre Aufgabe gut erfüllen konnten. Manchen Testern musste mit erneuter Ansage der Instruktionen oder Hilfestellung die Gegensteuerung betreffend geholfen werden, sodass sie ihre Aufgabe erfüllen konnten. Heraus gestochen ist, dass zwei Tester die Bewegung des Interaktionsobjekts als absolute Position des Roboters auf der Schiene wahrgenommen haben. Außerdem wurde das *User Interface* fast einstimmig als übersichtlich und angemessen beschrieben. Fünf der acht Testpersonen haben sich mehr eine optische Rückmeldung für die *Tap-And-Hold*-Geste gewünscht, um in Erfahrung bringen zu können, ob das Interaktionselement erfolgreich »angefasst« wurde. Außerdem waren unverkennbar ein Staunen und Freude erkennbar, sobald der Roboter sich durch Gestik bedingt in Bewegung gesetzt hat. Ebenfalls erkennbar war, angefangen bei Testprobanden aus dem Bereich des Büromanagements, über Projektleitung und Gestaltung hin zum Bereich der Entwicklung, eine Zunahme der *Usability*.

# Fazit und Ausblick

Das Ergebnis hat gezeigt, dass es mit Technologien wie AR, Gesten- und Sprachsteuerung möglich ist, eine Maschine, das heißt einen Spielzeugroboter, zu teleoperieren und dass augmentierte *User Interfaces* dabei hilfreich sind. Die Fertigstellung des funktionalen Prototypen wirft neue Fragestellungen auf. So stellt sich beispielsweise die Frage, wie ein solches System am besten für den Betrieb mit echten Industriemaschinen und unter betrieblichen Bedingungen entworfen und implementiert sein soll. Muss die Anwendung auf Seiten des Anwenders erheblich verbessert werden oder ist die Verbesserung der technischen, das heißt der *Hardware*-Seite viel wichtiger und ein minimalistisches *User Interface* vor den Augen des Anwenders reicht soweit aus? Wie schützt man eine solche Anwendung vor Fremdeinfluss und können Datenbrillen den Maschinenführern zusätzliche Sicherheit durch Sensorik bereitstellen? Wie stellt man sicher, dass das System generisch, mit einer Vielzahl von Robotern, funktioniert und ist eine Plattform mit der auf eine Vielzahl von Robotern zugegriffen werden kann sinnvoll oder gar gefährlich? Die Verwendung von *Unity*, speziell der Version 5.5f, der *Microsoft HoloLens*, dem *HoloToolkit* und *MonoBrick* hat gezeigt, dass man mit einem solchen *Framework* in der Lage ist, bei der Realisierung eines solchen Prototypen auf allen Ebenen die Programmiersprache *C#* verwendet werden kann auch wenn es mit Problemen der Kompatibilität von Schnittstellen einhergeht. Der Sachverhalt, dass sich viele der benutzten Technologien noch in den Kinderschuhen befinden, macht Hoffnung auf eine Umgebung, in der ohne größere Hindernisse und aus einer Hand, AR Anwendungen entwickelt werden können, und dass zukünftige Versionen von *Unity* und der *Microsoft HoloLens* dieses Vorhaben noch stärker vereinfachen und somit die AR Erfahrungen noch verbessern. Allgemein lässt sich zusammenfassen, dass alle wesentlichen Ziele dieser Arbeit erreicht wurden und einer erfolgversprechenden Weiterentwicklung der Anwendung nichts im Wege steht, vor allem jetzt, da aufschlussreiche Ansatzpunkte aus dem *Usability*-Test extrahiert werden konnten, welche die *Usability* erneut verbessern.

# Anlagen

## A.1 GazeManager.cs

Listing A.1: GazeManager.cs

```
1 using UnityEngine;
2
3 public class GazeManager : Singleton<GazeManager> {
4
5     /// Declaration of variables, setters and getters.
6     [...]
7
8     private void Update() {
9         gazeOrigin = Camera.main.transform.position;
10        gazeDirection = Camera.main.transform.forward;
11        gazeStabilizer.UpdateHeadStability(gazeOrigin, Camera.main
12            .transform.rotation);
13        gazeOrigin = gazeStabilizer.StableHeadPosition;
14        UpdateRaycast();
15    }
16
17    /// <summary>
18    /// Calculates the Raycast hit position and normal.
19    /// </summary>
20    private void UpdateRaycast() {
21        RaycastHit hitInfo;
22        Hit = Physics.Raycast(gazeOrigin, gazeDirection, out
23            hitInfo, MaxGazeDistance, RaycastLayerMask);
24        HitInfo = hitInfo;
25        if (Hit) { // If raycast hit a hologram...
26            Position = hitInfo.point;
27            Normal = hitInfo.normal;
28        }
29    }
}
```

## A.2 CursorManager.cs

### Listing A.2: CursorManager.cs

```
1 using UnityEngine;
2
3 /// <summary>
4 /// CursorManager class takes Cursor GameObjects.
5 /// One that is on Holograms and another off Holograms.
6 /// Shows the appropriate Cursor when a Hologram is hit.
7 /// Places the appropriate Cursor at the hit position.
8 /// Matches the Cursor normal to the hit surface.
9 /// </summary>
10 public class CursorManager : Singleton<CursorManager> {
11
12     [Tooltip("Drag the Cursor object to show when it hits a
13         hologram.")]
14     public GameObject CursorOnHolograms;
15
16     [Tooltip("Drag the Cursor object to show when it does not
17         hit a hologram.")]
18     public GameObject CursorOffHolograms;
19
20     void Awake() {
21         if (CursorOnHolograms == null || CursorOffHolograms ==
22             null) {
23             return;
24         }
25
26         // Hide the Cursors to begin with.
27         CursorOnHolograms.SetActive(false);
28         CursorOffHolograms.SetActive(false);
29     }
30
31     void Update() {
32         if (GazeManager.Instance == null || CursorOnHolograms ==
33             null || CursorOffHolograms == null) {
34             return;
35         }
36
37         if (GazeManager.Instance.Hit) {
38             CursorOnHolograms.SetActive(true);
39             CursorOffHolograms.SetActive(false);
40         } else {
41             CursorOffHolograms.SetActive(true);
42             CursorOnHolograms.SetActive(false);
43         }
44     }
45 }
```

```
41     gameObject.transform.position = GazeManager.Instance.  
        Position;  
42     gameObject.transform.up = GazeManager.Instance.Normal;  
43     }  
44 }
```

---

## A.3 InteractableManager.cs

**Listing A.3: InteractableManager.cs**

```
1 using UnityEngine;  
2  
3 /// <summary>  
4 /// InteractableManager keeps tracks of which GameObject  
5 /// is currently in focus.  
6 /// </summary>  
7 public class InteractableManager : Singleton<  
    InteractableManager> {  
8  
9     public GameObject FocusedGameObject { get; private set; }  
10    private GameObject oldFocusedGameObject = null;  
11  
12    void Start() {  
13        FocusedGameObject = null;  
14    }  
15  
16    void Update() {  
17        oldFocusedGameObject = FocusedGameObject;  
18  
19        if (GazeManager.Instance.Hit) {  
20            RaycastHit hitInfo = GazeManager.Instance.HitInfo;  
21            if (hitInfo.collider != null) {  
22                //Assign the hitInfo's collider gameObject to the  
                FocusedGameObject.  
23                FocusedGameObject = hitInfo.collider.gameObject;  
24            } else {  
25                FocusedGameObject = null;  
26            }  
27        } else {  
28            FocusedGameObject = null;  
29        }  
30  
31        if (FocusedGameObject != oldFocusedGameObject) {  
32            ResetFocusedInteractable();  
33        }
```

```

34     if (FocusedGameObject != null) {
35         if (FocusedGameObject.GetComponent<Interactable>() != null) {
36             // Send a GazeEntered message to the FocusedGameObject
37             .
38             FocusedGameObject.SendMessage("GazeEntered");
39         }
40     }
41 }
42
43 private void ResetFocusedInteractable() {
44     if (oldFocusedGameObject != null) {
45         if (oldFocusedGameObject.GetComponent<Interactable>() != null) {
46             //Send a GazeExited message to the
47             oldFocusedGameObject.
48             oldFocusedGameObject.SendMessage("GazeExited");
49         }
50     }
51 }

```

---

## A.4 GestureAction.cs

**Listing A.4: GestureAction.cs**

```

1 using System;
2 using UnityEngine;
3 using MonoBrick.EV3;
4 /// <summary>
5 /// GestureAction performs custom actions based on
6 /// which gesture is being performed.
7 /// </summary>
8 public class GestureAction : MonoBehaviour {
9     [Tooltip("Translation max speed controls amount of
10      translation.")]
11     public float TranslationSensitivity = 10.0f;
12
13     private Vector3 manipulationPreviousPosition;
14     private float translationFactor;
15     private EV3Manager ev3;
16     PositionManager positionManager;
17
18     void Start() {

```

```

18     ev3 = GameObject.Find("Managers").GetComponent<EV3Manager>();
19     Debug.Log(ev3 + "loaded!\nGetting in ready position...");
20     ev3.Beep();
21     ev3.MotorBlocked = false;
22     ev3.MoveRobotArmToStartingPosition();
23     positionManager = GetComponent<PositionManager>();
24 }
25
26 void Update(){
27     PerformTranslation();
28     PerformGrab();
29 }
30
31 private void PerformTranslation() {
32     if (GestureManager.Instance.IsNavigating && HandsManager
33         .Instance.FocusedGameObject == gameObject) {
34         // Calculate translationFactor based on GestureManager
35         // 's NavigationPosition.X and multiply by
36         // TranslationSensitivity.
37         // This will help control the amount of translation.
38         translationFactor = GestureManager.
39             Instance.NavigationPosition.x;
40
41         float clampTranslation = Mathf.Clamp(translationFactor
42             , -1f, 1f);
43         positionManager.SetPosition(clampTranslation / 2 + 0.5
44             f);
45         ev3.MoveRobot((sbyte)(clampTranslation * 10));
46     } else {
47         positionManager.SetPosition(0.5f);
48         ev3.StopRobot();
49     }
50 }
51
52 private void PerformGrab() {
53     if (GestureManager.Instance.IsNavigating && HandsManager.
54         Instance.FocusedGameObject == gameObject) {
55         // Calculate translationFactor based on GestureManager's
56         // NavigationPosition.X and multiply by
57         // TranslationSensitivity.
58         // This will help control the amount of translation.
59         translationFactor = GestureManager.
60             Instance.NavigationPosition.z;
61         transform.Translate(new Vector3(0, 0, translationFactor
62             / TranslationSensitivity));
63 }

```

```
50     transform.position = new Vector3(transform.position.x,
51         transform.position.y, Mathf.Clamp(transform.position
52             .z, - 1f, 1f));
53 }
```

---

## A.5 VoiceManager.cs

**Listing A.5: VoiceManager.cs**

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.Windows.Speech;
5 using System.Linq;
6 using System;
7
8 public class VoiceManager : MonoBehaviour {
9     KeywordRecognizer keywordRecognizer;
10    Dictionary<string, System.Action> keywords = new
11        Dictionary<string, System.Action>();
12    private EV3Manager ev3;
13
14    // Use this for initialization
15    void Start () {
16        ev3 = GetComponent<EV3Manager>();
17        keywords.Add("Grab", () => {
18            ev3.GrabObject();
19        });
20
21        keywords.Add("Place", () => {
22            ev3.PlaceObject();
23        });
24
25        keywords.Add("Stop", () => {
26            ev3.Stop();
27        });
28
29        keywordRecognizer = new KeywordRecognizer(keywords.Keys.
    ToArray());
        keywordRecognizer.OnPhraseRecognized +=
    KeywordRecognizer_OnPhraseRecognized;
        keywordRecognizer.Start();
    }
```

```

30
31     private void KeywordRecognizer_OnPhraseRecognized(
32         PhraseRecognizedEventArgs args) {
33     Debug.Log("Keyword Recognized...");
34     System.Action keywordAction;
35     if (keywords.TryGetValue(args.text, out keywordAction)) {
36         keywordAction.Invoke();
37     }
38
39     // Update is called once per frame
40     void Update () {
41
42     }
43 }
```

---

## A.6 EV3Manager.cs

**Listing A.6: EV3Manager.cs**

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine; using MonoBrick.EV3;
4 using System; using System.Threading;
5
6 public class EV3Manager : MonoBehaviour {
7     public Brick<TouchSensor, Sensor, Sensor, Sensor> ev3;
8     private bool motorBlocked;
9
10    public bool MotorBlocked {
11        get { return motorBlocked; }
12        set { motorBlocked = value; }
13    }
14
15    // Use this for initialization
16    void Start () {
17        ev3 = new Brick<TouchSensor, Sensor, Sensor, Sensor>("WiFi
18            ");
19        ev3.Connection.Open();
20        ev3.Sensor1.Mode = TouchMode.Boolean;
21        ev3.MotorA.ResetTacho();
22        ev3.MotorB.ResetTacho();
23        ev3.MotorC.ResetTacho();
24        MotorBlocked = false;
25    }
26}
```

```

25
26 // Update is called once per frame
27 void Update () {
28     Debug.Log(ev3.Sensor1.Read());
29     Debug.Log("Motor Blocked?=" + motorBlocked);
30     Debug.Log("TACHO B= " + ev3.MotorB.GetTachoCount());
31     Debug.Log("TACHO C= " + ev3.MotorC.GetTachoCount());
32
33     if (ev3.Sensor1.Read() == 1 && MotorBlocked == true) {
34         ev3.MotorB.Brake();
35         MotorBlocked = false;
36     }
37 }
38
39 public void MoveRobot(sbyte speed) {
40     ev3.MotorA.On(speed);
41 }
42
43 public void StopRobot() {
44     ev3.MotorA.Off();
45 }
46
47 public void MoveRobotArmToStartingPosition() {
48     ev3.MotorB.On(-25);
49     MotorBlocked = true;
50 }
51
52 public void Beep() {
53     ev3.Beep(10, 500);
54 }
55
56 public void GrabObject() {
57     WaitForMotorToStop();
58     ev3.MotorB.Off();
59     WaitForMotorToStop();
60     ev3.MotorC.MoveTo(10, 80, true);
61     WaitForMotorToStop();
62     ev3.MotorB.On(-25);
63     MotorBlocked = true;
64 }
65
66 internal void PlaceObject() {
67     MotorBlocked = false;
68     ev3.MotorB.Off();
69     WaitForMotorToStop();
70     ev3.MotorC.MoveTo(25, 0, true);
71     WaitForMotorToStop();

```

```

72     ev3.MotorB.On(-25);
73     MotorBlocked = true;
74 }
75
76 void OnDestroy() {
77     ev3.MotorA.Off();
78     ev3.MotorB.Off();
79     ev3.MotorC.Off();
80 }
81
82 void WaitForMotorToStop() {
83     Thread.Sleep(500);
84     while (ev3.MotorA.IsRunning() || ev3.MotorB.IsRunning() ||
85             ev3.MotorC.IsRunning())
86         Thread.Sleep(500);
87 }
88
89 public int GetSpeed() {
90     return ev3.MotorA.GetSpeed();
91 }
```

---

## A.7 Usability Test Transskript

### **Tester1, männlich, keine Vorkenntnisse, Officemanagement**

- deutlicher machen wie man Roboter bewegt

### **Tester2, weiblich, keine Vorkenntnisse, Officemanagement**

- Nicht Muttersprachler
- Hart die Ready-Gesture zu verstehen.
- Augen/Kopf-Hand-Koordination schnellere Ausführung der Gesten nötig Frustration, weil die Gesten nicht Funktionieren.
- Konnte »Grab«Aktion nicht ausführen
- Fehlfunktion schwierig/Motorblockade behindert

### **Tester3, männlich, keine Vorkenntnisse, Gestaltung**

- Hat versucht Cursor mit Auge oder Finger zu lenken
- Aussage: »Funktioniert erstaunlich gut.«
- Gaze schwierig
- Wenig Latenz!
- Aussage: »Wie Magie!«

### **Tester4, weiblich, keine Vorkenntnisse, Gestaltung**

- Icon-Feedback fürs Tippen gewünscht
- Frustum für Handdetection anzeigen
- Cyan blau gut
- Reagiert schnell
- Nur die wichtigsten Dinge auf Interface

### **Tester5, männlich, keine Vorkenntnisse, Projektmanagement**

- Anwendung hat direkt funktioniert
- Es hat Spaß gemacht.
- Erstaunlich das Roboter sich bewegt hat.
- Hat Sprachbefehl zweimal sprechen müssen
- Nicht intuitivste Steuerung
- Absolute Werte Positionsweite

### **Tester6, weiblich, keine Vorkenntnisse, Projektmanagement**

- Das Holointerface ist ansprechend
- Gut, dass die Befehle nochmals angezeigt werden
- Es hat Spaß gemacht.
- Leider Stimme nicht sehr gut erkannt

### **Tester7, weiblich, keine Vorkenntnisse, Entwicklung**

- Motorblockade nervt
- »Tap and Hold« zunächst nicht verstanden
- Wenn Gesten verstanden dann ist die Bedienung total leicht. Das gefällt
- Microsoft Gesten an für sich schwierig und ungewohnt
- Aufgabe in Interface Schritt für Schritt anzeigen
- Gut, dass die Befehle nochmals angezeigt werden

### **Tester8, männlich, keine Vorkenntnisse, Entwicklung**

- Interaktionsobjekt auf Höhe des Holointerface
- Einfaches, gut verständliches Interface
- Alles wichtige in einem Blick!
- Der Aufgabe angemessen
- Kontrollelement zu tief, dadurch Unsicherheit beim Bedienen
- Aufgabe in Interface Schritt für Schritt anzeigen
- Gut, dass die Befehle nochmals angezeigt werden

# Literatur

- [08] ? *Ergonomie der Mensch-System-Interaktion - Teil 110: Grundsätze der Dialoggestaltung (ISO 9241-110:2006); DeutscheFassung EN ISO 9241-110:2006.* Norm. Sep. 2008.
- [aut05] Dr. Oliver Schreer (auth.) *Stereoanalyse und Bildsynthese*. 1. Aufl. Springer-Verlag Berlin Heidelberg, 2005.
- [Bek08] George A. Bekey. *Robotics: state of the art and future challenges*. Imperial College Press ; Singapore ; Hackensack, NJ, 2008.
- [Bru08] Oussama Khatib Bruno Siciliano. *Springer Handbook of Robotics*. 1. Aufl. Springer, 2008.
- [DMG08] Carl Diercke, Thomas Michael und Wiebke Gehring. *Diercke - Weltatlas*. übergeordnete Quelle: gbv.559587341. [Accessed: 17.01.2017 11:08]. Braunschweig Westermann, 2008.
- [Gle07] Thomas Gleis. „(Die neue) DIN EN ISO 9241-110 („Grundsätze der Dialoggestaltung“)“. In: *Förderverein Usability-Netzwerk Bonn/Rhein-Sieg e.V mit Fraunhofer FIT* (2007).
- [Gob+01] Fernand Gobet, Peter CR Lane, Steve Croker et al. „Chunking mechanisms in human learning“. In: *Trends in cognitive sciences* 5.6 (2001), S. 236–243.
- [Jul11] Borko Furht (eds.) Julie Carmigniani Borko Furht (auth.) *Handbook of Augmented Reality*. 1. Aufl. Springer-Verlag New York, 2011.
- [KGH85] Myron W. Krueger, Thomas Gionfriddo und Katrin Hinrichsen. „VIDEO-PLACE: an Artificial Reality“. In: *SIGCHI Bull.* 16.4 (Apr. 1985), S. 35–40.
- [Kru10] Steve Krug. *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems*. 1. Aufl. New Riders Press, 2010.
- [Nie93] Jakob Nielsen. *Usability Engineering*. 1st. Morgan Kaufmann, 1993.
- [PhD16] Jason Ph.D. Jerald. *The VR Book*. 1. Aufl. San Rafael, CA: ACM Books, 2016.

- [Rup14] Chris Rupp. *Requirements-Engineering und -Management: Aus der Praxis von klassisch bis agil*. Carl Hanser Verlag GmbH Co KG, 2014.
- [SK08] Bruno Siciliano und Oussama Khatib, Hrsg. *Springer Handbook of Robotics*. Springer, 2008.
- [Xio+16] Wayne Xiong, Jasha Droppo, Xuedong Huang et al. „Achieving Human Parity in Conversational Speech Recognition“. In: *CoRR* abs/1610.05256 (2016).

## Online-Quellen

- [15] *Industrie 4.0 – Chancen und Perspektiven für Unternehmen der Metropolregion Rhein-Neckar*. [Accessed: 06.01.2017 15:43]. 2015. URL: [http://www.ipa.fraunhofer.de/fileadmin/user\\_upload/Publikationen/Studien/Studientexte/Studie\\_Industrie\\_4.0\\_IHK\\_Fraunhofer\\_IPA.pdf](http://www.ipa.fraunhofer.de/fileadmin/user_upload/Publikationen/Studien/Studientexte/Studie_Industrie_4.0_IHK_Fraunhofer_IPA.pdf).
- [Ano] Anonymous. *31313 Mindstorms EV3*. [Accessed: 21.02.2017 19:04]. URL: <https://www.lego.com/de-de/mindstorms/products/mindstorms-ev3-31313>.
- [Ano16] Anonymous. *Project Website*. [Accessed: 16.02.2017 09:02]. 2016. URL: <http://www.sueddeutsche.de/wirtschaft/roboerbauer-aus-augsburg-chinesische-investoren-kaufen-mehrheit-an-roboerbauer-kuka-1.3062929>.
- [aug] augmentedreality. URL: [http://de.augmentedreality.wikia.com/wiki/Datei:Sword\\_of\\_damocles.jpg](http://de.augmentedreality.wikia.com/wiki/Datei:Sword_of_damocles.jpg) (zitiert auf Seite 22).
- [BMW16] Hrsg. BMWi. *Was ist Industrie 4.0? - Die vierte industrielle Revolution: Auf dem Weg zur intelligenten und flexiblen Produktion*. [Accessed: 06.12.2016 09:34]. 2016. URL: <http://www.plattform-i40.de/I40/Navigation/DE/Industrie40/WasIndustrie40/was-ist-industrie-40.html>.
- [bpb] bp. URL: [http://www.bpb.de/system/files/datei/SOZ\\_01\\_04\\_Grafiken.zip](http://www.bpb.de/system/files/datei/SOZ_01_04_Grafiken.zip) (zitiert auf Seite 8).
- [col] columbia. URL: <http://www.augmentedrealityvisor.com/images/ar-karma.jpg> (zitiert auf Seite 23).
- [Dai] Genius - Daimler. URL: [https://www.genius-community.com/wp-content/uploads/2016/05/Roboter\\_Werk.jpg](https://www.genius-community.com/wp-content/uploads/2016/05/Roboter_Werk.jpg) (zitiert auf Seite 16).
- [Dev] Devol. URL: <http://cyberneticzoo.com/wp-content/uploads/Devol-patent-1954.PNG> (zitiert auf Seite 13).

- [Die J] Diercke, Topel / Seibel. *Deutschland - Rohstoffabhängigkeit Erde - Welt-handel*. o. J. URL: %5Curl%7Bhttp://www.diercke.de/content/deutschland-rohstoffabh%C3%A4ngigkeit-978-3-14-100700-8-244-2-0%7D.
- [eur] europa.eu. URL: http://ec.europa.eu/eurostat/statistics-explained/images/c/cf/Estimated\_hourly\_labour\_costs%2C\_2015\_%28%C2%B9%29\_%28EUR%29\_YB16-de.png (zitiert auf Seite 7).
- [Gar] Gartner. URL: http://na2.www.gartner.com/imagesrv/newsroom/images/emerging-tech-hc-2016.png;wa59f7b006c484099e (zitiert auf Seite 26).
- [his] history-computer. URL: http://history-computer.com/Dreamers/images/Jaquet-Droz\_Musician.jpg (zitiert auf Seite 12).
- [IPA] Fraunhofer IPA. *Urbane Produktion*. URL: %5CUR1%7Bhttp://www.ipa.fraunhofer.de/urbane-produktion.html%7D.
- [kin] kingscollections. URL: http://www.kingscollections.org/media/exh\_spc/images/006850/D01\_Stereoscope.jpg (zitiert auf Seite 20).
- [Leg] Lego. URL: https://sh-s7-live-s.legocdn.com/is/image/LEGO/45500\_alt1?id=fb7Qx3&fmt=jpg&fit=constrain,1&wid=568&hei=426&qlt=80,1&op\_sharpen=0&resMode=sharp2&op\_usm=1,1,6,0&iccEmbed=0&printRes=72 (zitiert auf Seite 17).
- [Lin16] Prof. Dr.-Ing. Heinrich Linemann. *Vorlesungsskript - Robotik*. Version 16/17. Accessed: 2017-02-08 19:28. 2016. URL: https://prof.beuth-hochschule.de/fileadmin/user/linemann/PDF-Dateien/Robotertechnik/Roboter\_Technik\_Vorlesung\_Alles.pdf (zitiert auf den Seiten 13, 14).
- [Mer] Galerie Merveilles. URL: http://galeriedesmerveilles.jaquet-droz.com/sites/musee/files/styles/large\_desktop/public/D\_WRITER-MESSAGE.jpg (zitiert auf Seite 12).
- [Mica] Microsoft. URL: https://support.microsoft.com/en-us/help/12644/hololens-use-gestures (zitiert auf Seite 33).
- [Micb] Microsoft. URL: https://1.f ix.de/scale/geometry/695/q75/imgs/18/1/7/3/6/1/7/9/908a2bb5-1509-47e8-ac9f-f48f719846be-edb717f1bd3f6c1c.png (zitiert auf Seite 27).
- [Micc] Microsoft. URL: https://az835927.vo.msecnd.net/sites/holographic/resources/images/Sensor\_bar.jpg (zitiert auf Seite 31).
- [mor] mortonheilig. URL: http://www.mortonheilig.com/sensorama-1.jpg (zitiert auf Seite 22).

- [Pla] PlayStation. URL: [https://psmedia.playstation.com/is/image/psmedia/eyetoplay3\\_sc004?MediaCarousel\\_SmallImage\\$](https://psmedia.playstation.com/is/image/psmedia/eyetoplay3_sc004?MediaCarousel_SmallImage$) (zitiert auf Seite 25).
- [pra] pratt. URL: <http://patentimages.storage.googleapis.com/pages/US1183492-0.png> (zitiert auf Seite 21).
- [Pra16] A.B. Pratt. *Weapon*. [Accessed: 2017-01-24 10:49]. Mai 1916. URL: <http://www.google.com/patents/US1183492>.
- [Rek98] Ph.D Rekimoto. [Accessed: 2017-01-30 16:40]. 1998. URL: <https://www.sonycs1.co.jp/person/rekimoto/papers/apchi98.pdf>.
- [res] researchgate. URL: [https://www.researchgate.net/profile/Mark\\_Billinghurst/publication/29487174/figure/fig1/AS:3099679670149380\\_1450913530882/Figure-2-Milgram's-Reality-Virtuality-Continuum.png](https://www.researchgate.net/profile/Mark_Billinghurst/publication/29487174/figure/fig1/AS:3099679670149380_1450913530882/Figure-2-Milgram's-Reality-Virtuality-Continuum.png) (zitiert auf Seite 19).
- [Ros17] Ortiz-Ospina Roser. *World Population Growth*. Accessed: 2017-02-01 12:48. 2017. URL: <https://ourworldindata.org/world-population-growth/#fertility>.
- [Run] Bayrischer Rundfunk. URL: [http://www.br.de/themen/wissen/henry-ford-automobile-110\\_v-img\\_\\_16\\_\\_9\\_xl\\_-d31c35f8186eb80b0cd843a7c267a0e0.jpg?version=998de](http://www.br.de/themen/wissen/henry-ford-automobile-110_v-img__16__9_xl_-d31c35f8186eb80b0cd843a7c267a0e0.jpg?version=998de) (zitiert auf Seite 15).
- [S93] Feiner / MacIntyre / und Seligmann. *Project Website*. [Accessed: 10.01.2017 10:31]. 1993. URL: <http://monet.cs.columbia.edu/projects/karma/karma.html>.
- [Sch16] Ilse Schmiedecke. *Vorlesungsskript HCI*. [Accessed: 07.01.2017 08:34]. 2016. URL: <http://www.schmiedecke.info/HCI/Folien/HCI-17-Zusammenfassung.pdf>.
- [Sob] Jeppesen Soborg. *MonoBrick EV3 Firmware*. [Accessed: 21.02.2017 15:52]. URL: <http://www.monobrick.dk/software/ev3firmware/>.
- [Uni] Unity. URL: <https://unity3d.com/de/unity/multiplatform>.
- [Vim] Vimeo. URL: [http://i.vimeocdn.com/video/547137267\\_1280x720.jpg](http://i.vimeocdn.com/video/547137267_1280x720.jpg) (zitiert auf Seite 3).
- [Voi16] Prof. Dr. Voigt. *Gabler Wirtschaftslexikon, Stichwort: industrielle Revolution*. [Accessed: 06.01.2017 13:37]. 2016. URL: <http://wirtschaftslexikon.gabler.de/Archiv/73533/industrielle-revolution-v4.html>.
- [Wel99] Don Wells. *Extreme Programming - User Stories*. [Accessed: 04.01.2017 12:38]. 1999. URL: <http://www.extremeprogramming.org/rules/userstories.html>.
- [Wika] Wikimedia. URL: <https://commons.wikimedia.org/w/index.php?curid=48038451> (zitiert auf Seite 20).

- [Wikb] Wikipedia. URL: <https://de.wikipedia.org/wiki/Epipolargeometrie#/media/File:Epipolargeometrie3.svg> (zitiert auf Seite 24).
- [Wikc] Wikipedia. URL: [https://commons.wikimedia.org/wiki/File:Hero\\_of\\_Alexandria,\\_Automata,\\_Venice,\\_Gr.\\_516.jpg](https://commons.wikimedia.org/wiki/File:Hero_of_Alexandria,_Automata,_Venice,_Gr._516.jpg) (zitiert auf Seite 12).
- [Wikd] Wikipedia. URL: <https://commons.wikimedia.org/w/index.php?curid=481573%20Zeichner> (zitiert auf Seite 12).
- [Wike] Wikipedia. URL: <https://commons.wikimedia.org/w/index.php?curid=481534%20Zeichnung> (zitiert auf Seite 12).
- [Wikf] Wikipedia. URL: <https://commons.wikimedia.org/w/index.php?curid=481571%20Schreiber> (zitiert auf Seite 12).
- [Wikg] Wikipedia. *Microsoft HoloLens Wikipedia Entry*. [Accessed: 2016-12-14 11:25]. URL: [https://de.wikipedia.org/wiki/Microsoft\\_HoloLens.html](https://de.wikipedia.org/wiki/Microsoft_HoloLens.html).
- [Wikh] Wikipedia. *Unity (Spiele-Engine)*. [Accessed: 14.02.2017 14:35]. URL: [https://de.wikipedia.org/wiki/Unity\\_\(Spiel-Engine\)](https://de.wikipedia.org/wiki/Unity_(Spiel-Engine)).
- [Wik17] Wikipedia. *Augmented Reality*. Accessed: 2017-02-01 13:07. 2017. URL: [https://de.wikipedia.org/wiki/Erweiterte\\_Realit%C3%A4t](https://de.wikipedia.org/wiki/Erweiterte_Realit%C3%A4t).
- [Xin] Xinfo. URL: <http://xinfo.xinfogmbh.netdna-cdn.com/sites/default/files/Industrie-4.0-Definition-Stufen-Revolution-01.png> (zitiert auf Seite 6).

# Erklärung

Hiermit erkläre ich eidesstattlich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, welche anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

*Berlin, 28. Februar 2017*

---

Ibrahim Khaled Reguieeg