DLR-IB-SL-HF-2022-61

Erfassung dynamischer menschlicher Parameter in kollaborativen Arbeitsprozessen mit Augmented Reality

Masterarbeit

Florian Beckert

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Dokumenteigenschaften

Titel	Erfassung dynamischer menschlicher Parameter in kollaborativen Arbeitsprozessen mit Augmented Reality
Betreff	Masterarbeit
Institut	Institut für Systemarchitekturen in der Luftfahrt (SL)
Erstellt von	Florian Beckert
Beteiligte	
Geprüft von	M.Sc. Mara Fuchs (DLR-SL)
Freigabe von	Prof. Dr.-Ing. Thorsten Schüppstuhl (TUHH)
Datum	02.05.2022
Version	1.0
Dateipfad	

Institut für Flugzeug-
Produktionstechnik
Prof. Dr.- Ing. Thorsten Schüppstuhl

Masterarbeit

Name: Florian Beckert

Matr.-Nr.: 21866230

Thema: Erfassung dynamischer menschlicher Parameter in kollaborativen Arbeitsprozessen mit Augmented Reality

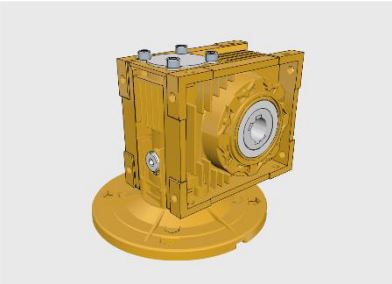
Erstprüfer: Prof. Dr.-Ing. Thorsten Schüppstuhl
Zweitprüfer: Prof. Dr.-Ing. Hermann Lödding

Betreuer: Constantin Deneke, M.Sc. (IFPT)
Mara Fuchs, M.Sc. (DLR)
Dr.-Ing. Jörn Biedermann (DLR)

Hamburg, den 02. Mai 2022

Masterarbeit

Erfassung dynamischer menschlicher Parameter in kollaborativen Arbeitsprozessen mit Augmented Reality



Gearbox zur Validierung des entwickelten Ansatzes.

Eingesetztes Augmented-Reality-Headset: Microsoft HoloLens 2

Hintergrund und Motivation

Die Luftfahrt ist vom Luftfahrzeugentwurf über die Produktion bis zum Betrieb digitalisiert. Diese durchgehende Verfügbarkeit aller Daten wird als Digitaler Faden bezeichnet. Das Institut für Systemarchitekturen in der Luftfahrt erforscht, wie aus diesen umfangreichen Daten praktisch nutzbares Wissen abgeleitet werden kann, um noch leistungsfähigere, effizientere und sichere Luftfahrtprodukte zu entwerfen. Zur Validierung von neu entwickelten Methoden dient am Institut für Systemarchitekturen in der Luftfahrt unter anderem ein Versuchsstand für kollaborative Arbeit zwischen Menschen und Robotern. Die Versuchsanlage ermöglicht es, Szenarien zu testen, bei denen die Daten aus dem digitalen Faden verwendet werden um Montageprozesse zu automatisieren und zu optimieren. Herausforderungen hierbei sind die Aggregierung und Verarbeitung von vielfältigen, statischen und dynamischen Parametern.

Für die vollständige Digitalisierung eines kollaborativen Prozesses ist es notwendig, zusätzlich zu den Daten der beteiligten Maschinen und Roboter auch Parameter zur menschlichen Arbeit zu erfassen. Die Daten können dann beispielsweise genutzt werden, um Kollisionen zu verhindern und Arbeitsabläufe zu optimieren. Die menschlichen Parameter unterliegen naturgemäß einer starken Dynamik und können daher nur mit großen Unsicherheiten simuliert werden. Des Weiteren ist es auf Grund der offensichtlich fehlenden digitalen Schnittstelle nur mit Hilfe zusätzlicher Geräte möglich, Realdaten aufzuzeichnen und zu digitalisieren. Um eine solche Schnittstelle zu schaffen haben Forscher beispielsweise gezeigt, wie mit Kameras und Sensoren menschliches Verhalten beobachtet und analysiert werden kann [1].

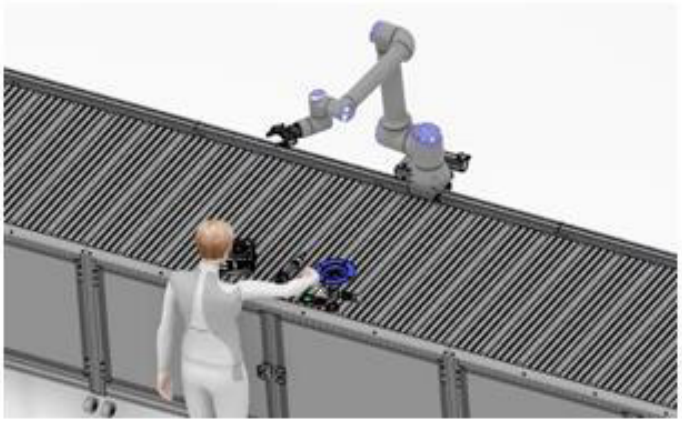
In den letzten Jahren machte die Entwicklung von Augmented Reality (AR) große Fortschritte. AR-Headsets wurden technisch ausgereifter und kostengünstiger und kommen nun auch für die breite industrielle Anwendung in Frage. Es hat sich gezeigt, dass auch der Einsatz von AR in der Mensch-Roboter-Interaktion vielversprechend ist. Es wurde beispielsweise gezeigt, wie AR den Menschen dabei unterstützen kann, Status und Intention von Robotern zu erkennen [2] oder Roboter zu steuern und zu programmieren [3]. In dieser Arbeit soll ein neuer Ansatz entwickelt werden, dynamische Parameter zur menschlichen Arbeit mit einem Augmented-Reality-Headset aufzuzeichnen und diese im digitalen Zwilling zu verarbeiten.

[1] Kousi, Niki et al. "Digital Twin for Designing and Reconfiguring Human–Robot Collaborative Assembly Lines." Applied Sciences 11 (2021): 4620.
[2] Hietanen, A. et al. "AR-based interaction for human-robot collaborative manufacturing." Robotics Comput. Integr. Manuf. 63 (2020): 101891.
[3] Mueller, Fabian et al. "Intuitive Welding Robot Programming via Motion Capture and Augmented Reality." IFAC-PapersOnLine 52 (2019): 294-299.

DLR

Ziele der Arbeit

Ziel der Arbeit ist es, eine Methode zu entwickeln, mit der Parameter wie Prozesszeiten für einzelne menschliche Arbeitsschritte sowie Übergangszeiten zwischen Einzelschritten mit Hilfe der HoloLens2 gemessen werden können. Anhand der entwickelten Methode soll untersucht werden, inwieweit sich der Einsatz von Augmented Reality und der zugehörigen AR-Hardware eignet, dynamische Parameter für den digitalen Zwilling des Menschen in kollaborativen Arbeitsprozessen aufzuzeichnen. Der Fokus soll hierbei auf der Genauigkeit und Zuverlässigkeit der Datenerfassung sowie der industriellen Anwendbarkeit liegen.
Als Anwendungsbeispiel dient die kollaborative Montage eines Getriebes. Die aggregierten Daten sollen automatisiert an einen Server übertragen und dort verarbeitet werden. Anschließend sollen die übertragenen Daten algorithmisch analysiert und beispielhaft dazu verwendet werden, die Reihenfolge der einzelnen kollaborativen Arbeitsschritte zu optimieren.



Visualisierung des Versuchsaufbaus:

Roboter und Mensch führen die Getriebemontage in einem kollaborativen Prozess durch. Der Mensch trägt dabei die HoloLens. Die Position des Arbeiters und seiner Hände wird mit der HoloLens erfasst und permanent an den digitalen Zwilling übertragen. Die Bearbeitungszeiten für einzelne Prozessschritte sowie Übergangszeiten zwischen Prozessschritten werden anhand von den übertragenen Positionsdaten errechnet und zur Optimierung der Prozessreihenfolge genutzt.

Vorgehensweise

- Recherche zum Stand der Wissenschaft.
- Festlegung eines beispielhaften kollaborativen Arbeitsprozesses zwischen Mensch und Roboter zur Getriebemontage.
- (Geometrische) Modellierung des Prozesses mit Hilfe einer geeigneten Prozessmodellierungssprache.
- Programmierung einer AR-Anwendung für die HoloLens 2:
  - ➔ Entwicklung einer Methode zur räumlichen Kalibrierung der HoloLens mit Hilfe von Markern.
  - ➔ Entwicklung einer Methode zur permanenten automatisierten Erfassung, Übertragung und Speicherung von Kopf- und Handpositionen an Server.
  - ➔ Durchführung von Versuchsreihen mit Hilfe der entwickelten Anwendung.
- Programmierung einer Serverumgebung zur Verarbeitung der Daten:
  - ➔ Analyse der übertragenen Daten: Berechnen von Zeiten für einzelne Prozessschritte und Übergangszeiten zwischen Einzelprozessen auf Grundlage des Prozessmodelles.
  - ➔ Zusammenführen der berechneten Parameter mit vorab gemessenen Daten zu Roboter-Prozesszeiten.
  - ➔ Anwendung eines Algorithmus zur Optimierung der Prozessschrittreihenfolge mit dem Ziel der Minimierung der Gesamtprozesszeit (z.B. mit Hilfe von Prozessgraphen).
  - ➔ Rückübertragung und Darstellung des optimierten Ablaufs für den Arbeiter über die HoloLens.
- Bewertung des Ansatzes in Bezug auf Genauigkeit und Zuverlässigkeit der Datenerfassung und der industriellen Anwendbarkeit; Diskussion und Dokumentation der Ergebnisse.

Erklärung

Ich, Florian Beckert (Student Flugzeug-Systemtechnik an der Technischen Universität Hamburg, Matrikelnummer 21866230), versichere an Eides Statt, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner Prüfungskommission vorgelegt.

Hamburg, 02. Mai 2022

Inhaltsverzeichnis

Kurzfassung

Der Begriff „Industrie 4.0“ fasst Technologien und Methoden zusammen, mit denen Produktionssysteme sich durch eine systematische Erfassung und Analyse von Daten selbstständig steuern und optimieren können. Insbesondere dort, wo Tätigkeiten innerhalb des Produktionsprozesses von Menschen ausgeführt werden, entsteht die Herausforderung, dass für die Erfassung menschlicher Daten in der Regel eine komplexe technische Infrastruktur notwendig ist. Die vorliegende Arbeit befasst sich mit der Fragestellung, ob der Prozess, menschliche Daten zu erfassen, durch die Anwendung eines AR-Headsets vereinfacht werden kann und auf ortsfeste Infrastruktur verzichtet werden kann. Zur Überprüfung des Ansatzes wird eine AR-Anwendung entwickelt, die kontinuierlich die Position eines Arbeiters in der Produktionsanlage bestimmt. Grundlage hierfür ist das dauerhaft vom AR-Headset durchgeführte SLAM-Tracking in Verbindung mit einem optischen, markerbasierten Kalibrierungsverfahren. Im Rahmen einer Messreihe wird die Präzision der Positionsbestimmung überprüft und mit anderen Verfahren zur Lokalisierung in Innenräumen verglichen. Für die Analyse der Positionsdaten wird zusätzlich eine Serveranwendung entwickelt, welche die Positionsdaten empfängt, speichert und mit Hilfe einer vorher festgelegten Prozessbeschreibung automatisiert Prozesszeiten aus den ermittelten Positionsdaten berechnet. Anhand eines praktischen Anwendungsfalls wird aufgezeigt, wie die ermittelten Prozesszeiten genutzt werden können, um die Reihenfolge des Prozesses zu optimieren.

Abstract

The term Industry 4.0 summarizes technologies and methods with which production systems can autonomously control and optimize themselves through the systematic collection and analysis of data. In cases where activities within the production process are carried out by humans, the challenge arises that a complex technical infrastructure is usually required for the acquisition of human data. This paper focuses on the question whether the process of capturing human data can be simplified by the application of an AR headset and fixed infrastructure can be avoided. To test the approach, an AR application is developed that continuously tracks the position of a worker in the production line. The foundation for this is SLAM tracking performed permanently by the AR headset in combination with an optical marker-based calibration procedure. The accuracy of the position determination is verified as part of a measurement series and compared with other indoor localization methods. In addition, a server application is developed for the analysis of the position data. This application receives and stores the position data and automatically calculates process times from the acquired position data using a predefined process description. A practical use case is used to demonstrate how the determined process times can be used to optimize the order of the process.

II Abkürzungen

Abkürzung	Beschreibung
2D	zweidimensional
3D	dreidimensional
AR	Augmented Reality
AV	Augmented Virtuality
Cobot	Collaborative robot
CP	Constraint Programming
CPS	Cyber-physisches System
DB	Datenbank
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V.
FPS	Frames per Second
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HMD	Head-Mounted Display
HMI	Human Machine Interface
ID	Identifikation
IMU	Inertial Measurement Unit
IoT	Internet of Things
IPS	Indoor Positioning System
JSON	JavaScript Object Notation
KV	Kontrollvolumen
LIDAR	Light Detection And Ranging
MR	Mixed Reality
MRTK	Mixed Reality Toolkit
NP	nichtdeterministisch polynomielle Zeit
RGB	Rot Grün Blau
SLAM	Simultaneous Localization and Mapping
UWB	Ultra-wideband
VR	Virtual Reality
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

III Abbildungsverzeichnis

IV Tabellenverzeichnis

1 Einleitung

1.1. Hintergrund und Motivation

Die Anforderungen an moderne industrielle Produktionssysteme erhöhen sich stetig. Ein globalisierter Markt fordert immer personalisiertere und regional angepasste Produkte, was zu einer Verringerung von Produktionsmengen einzelner Varianten führt. Bei gleichzeitigem ökonomischen Druck und sich erhöhenden Nachhaltigkeitsanforderungen ergibt sich für Unternehmen hieraus eine deutliche Vergrößerung der Komplexität. Um dieser Komplexität zu begegnen und wettbewerbsfähig zu bleiben, müssen Unternehmen ihre Produktionssysteme anpassen und die Prozesseffizienz steigern. Angestrebt wird eine Optimierung entlang der gesamten Wertschöpfungskette, die vor allem durch eine vollständige Digitalisierung aller Phasen des Produktlebenszyklus erreicht werden soll. Für diese Zukunftsvision hat sich hierzulande die Bezeichnung „Industrie 4.0“ durchgesetzt, eine Anspielung auf die weitreichenden Veränderungen, welche einer vierten industriellen Revolution ähnlich sind.

Bestandteil der Vision der Industrie 4.0 ist, dass Produktionssysteme idealerweise aus verschiedenen sogenannten cyber-phyischen Systemen (CPS) gebildet werden. Gemeint sind Systeme (z. B. Maschinen oder Roboter), die mit Aktuatoren auf die physische Welt einwirken können und gleichzeitig mit Sensoren Daten aus dieser erfassen können. Über Schnittstellen können diese Daten mit anderen Systemen ausgetauscht werden. Ein so vernetztes System bietet die Möglichkeit, den Produktionsprozess in vielfältiger Art und Weise zu optimieren.

Obwohl ein langfristiges Ziel die vollständige Automatisierung von Prozessen ist, bleibt der menschliche Arbeiter vorerst ein wichtiger Teil vieler Industrieanlagen. Er kommt überall dort zum Einsatz, wo Prozesse sich nicht automatisieren lassen oder dies nicht wirtschaftlich ist. In vielen Fällen lässt sich die Produktivität des Arbeiters jedoch deutlich steigern, indem Teilaufgaben von Robotern übernommen werden. Um eine enge räumliche Zusammenarbeit zwischen Mensch und Roboter zu ermöglichen, werden kollaborative Roboter (engl. Cobots) eingesetzt. Auch solche kollaborativen Prozesse bieten ein großes Potenzial für Optimierung. So können beispielsweise durch eine vorausschauende Ablaufplanung Wartezeiten reduziert oder Kollisionen vermieden werden. Folgt man der Idee der Industrie 4.0, dann lässt sich diese Prozessoptimierung durch den gezielten Austausch von Daten zwischen Mensch und Roboter erreichen. Mensch und Roboter bilden demnach jeweils ein CPS, welches Daten und Parameter bereitstellt und Anweisungen empfängt (siehe Abb. 1.1). Während moderne kollaborative Roboter meist schon über digitale und vernetzte Steuereinheiten verfügen, ist das Schaffen von digitalen Schnittstellen beim Menschen mit Herausforderungen verbunden. Um digitale Informationen zu empfangen, ist der Mensch auf externe Geräte wie Bildschirme oder Lautsprecher angewiesen. Komplexer noch ist die Erfassung von menschlichen Daten,

da hierfür entweder manuelle Eingaben oder die Nutzung aufwendiger Sensorsysteme notwendig sind.

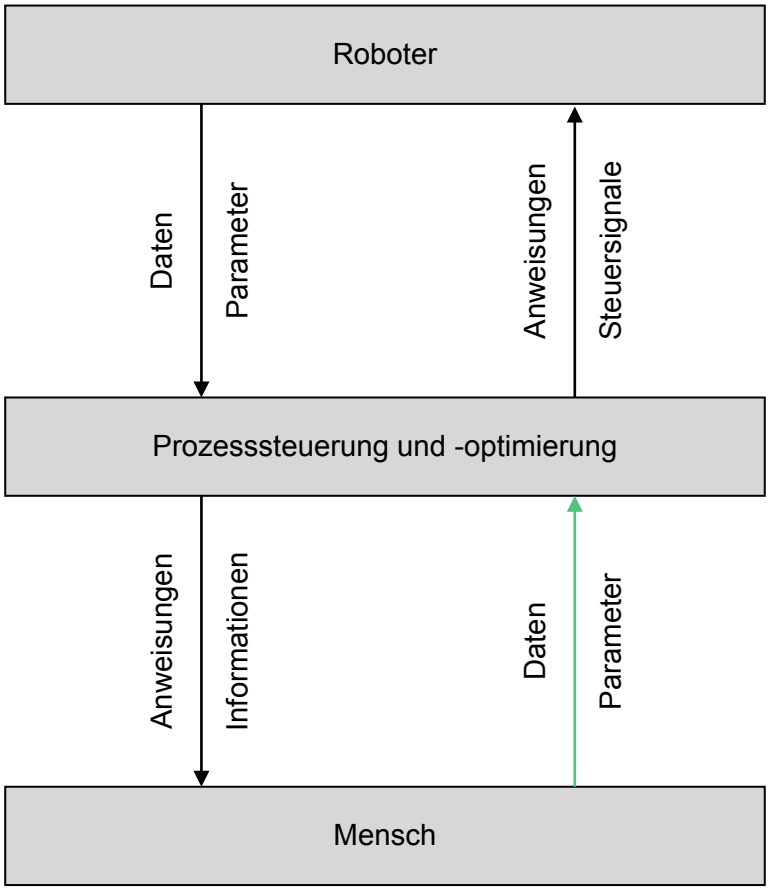


Abbildung 1.1.: Schematische Darstellung des Datenflusses bei der digitalen Steuerung und Optimierung eines kollaborativen Arbeitsprozesses

Ein Beispiel für menschliche Daten, die für die Prozesssteuerung und -optimierung von kollaborativen Mensch-Roboter-Prozessen genutzt werden können, ist die räumliche Position des Arbeiters. Mit diesen Positionsdaten können z. B. Durchführungszeiten berechnet oder der Prozess effizient gesteuert werden. Um einen Menschen kontinuierlich zu lokalisieren, ist jedoch in der Regel eine komplexe und ortsfeste Infrastruktur mit kamera- oder funkbasierten Sensoren erforderlich. Als Folge dieser Komplexität wird in der Praxis häufig auf die Erfassung der menschlichen Parameter verzichtet und für die Prozessplanung auf empirische Tabellenwerte oder Computersimulationen zurückgegriffen, welche das dynamische Verhalten des Menschen jedoch nur eingeschränkt abbilden können.

Ein vielversprechender Ansatz, um den Menschen mit einer digitalen Schnittstelle auszurüsten, sind Augmented Reality (AR) Systeme, insbesondere AR-Headsets, die auf dem Kopf getragen werden und den Arbeiter möglichst wenig in seiner Bewegungsfreiheit einschränken. Während Nutzereingaben durch Handgesten oder Sprachbefehle erfolgen können, werden digitale Informationen als Hologramme über halbtransparente Bildschirme in das

Sichtfeld des Nutzers eingeblendet. Um eine möglichst natürliche Darstellung der Hologramme zu gewährleisten, erfasst das AR System kontinuierlich mit Sensoren seine räumliche Umgebung und berechnet die Bildschirmanzeige neu. Ziel ist es, dass der Eindruck entsteht, das Hologramm bliebe immer an der gleichen Stelle im Raum. Hieraus ergibt sich der Ansatz, die bei diesem Prozess anfallenden Sensordaten systematisch zu nutzen, um Positionsdaten des menschlichen Arbeiters passiv, kontinuierlich und ohne ortsfeste Infrastruktur zu erfassen.

1.2. Ziel der Arbeit

Diese Arbeit verfolgt den in der Literatur bislang wenig beachteten Ansatz, die vom AR-Headset temporär zur Berechnung der Bildschirmanzeige genutzten Sensordaten systematisch zu verarbeiten, zu übertragen und zu analysieren. Im Rahmen dieser Arbeit liegt der Fokus hierbei zunächst auf der Bestimmung von Positionsdaten des Arbeiters. Anhand dieses Beispiels soll untersucht werden, ob das AR-Headset neben seiner Funktion als Aus- und Eingabemedium es zusätzlich ermöglichen kann, relevante Prozessparameter eines Produktionsarbeiters zu bestimmen, die dann für die Optimierung des Prozesses verwendet werden können (siehe Abb. 1.2).

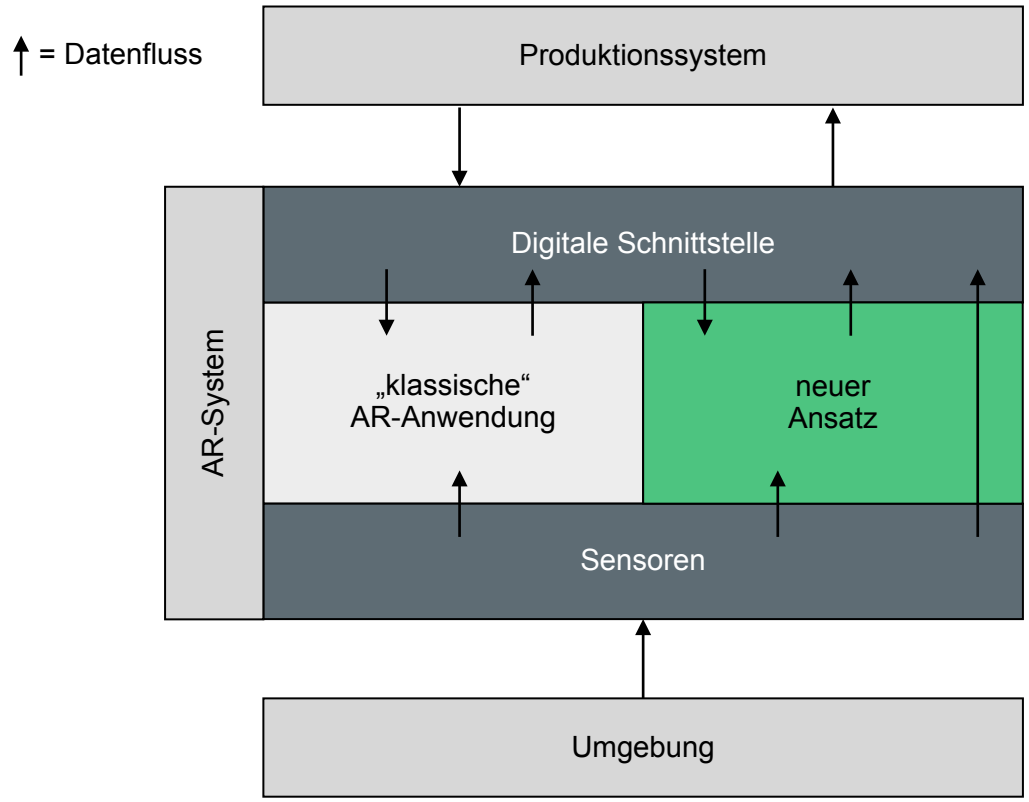


Abbildung 1.2.: Schematische Darstellung des Datenflusses einer AR-Anwendung und des neuen Ansatzes

Zu diesem Zweck wird im Rahmen dieser Arbeit eine AR-Anwendung für die Microsoft HoloLens 2 entwickelt, die die Position des Headset-Trägers im Raum kontinuierlich bestimmt und diese serverbasiert speichert und analysierbar macht. Die Lokalisierung erfolgt hierbei in einem raumfesten Koordinatensystem, welches in einem Kalibrierungsverfahren unter Nutzung eines QR-Markers mit bekannter Position an das Koordinatensystem des AR-Systems angepasst wird. Durch ein photogrammetrisches Messverfahren wird die Genauigkeit der Positionserfassung durch die Anwendung bestimmt. Die ermittelte Abweichung wird dann mit Literaturwerten anderer Lokalisierungssysteme verglichen. Zusätzlich erfolgt ein Vergleich in Bezug auf die Systemkomplexität, die Infrastrukturanforderungen und die Kosten.

Anschließend erfolgt die Erprobung der entwickelten Anwendung anhand eines praktischen Anwendungsfalls. Als Beispiel dient hierbei die Montage eines Getriebes, welche kollaborativ von Mensch und Roboter durchgeführt wird. Anhand des Anwendungsfalls soll aufgezeigt werden, wie aus den erfassten Positionsdaten relevante Prozessparameter wie Bearbeitungs- und Transferzeiten bestimmt werden können, welche dann zur Optimierung des Prozesses verwendet werden können. Auf Grundlage des Anwendungsfalls wird schließlich die industrielle Anwendbarkeit der Methode qualitativ bewertet.

1.3. Aufbau der Arbeit

Nach der Einleitung folgen in **Kapitel zwei** die **Grundlagen**. Zunächst wird der Begriff der Industrie 4.0 definiert und auf die aktuellen Herausforderungen der Industrieproduktion eingegangen. Anschließend werden der Aufbau und die Funktion der für die Industrie 4.0 charakteristischen cyber-physischen Systeme beschrieben. Es wird zudem auf die Rolle des Menschen in der Industrieproduktion der Zukunft eingegangen und beschrieben, welche Rolle die Kollaboration von Mensch und Roboter hierbei hat. Anschließend wird erläutert, wie der Industriearbeiter selbst zum cyber-physischen System werden kann, welche menschlichen Daten relevant für die Optimierung von Prozessen sind und wo die Herausforderungen bei der Datenerfassung liegen. Zuletzt erfolgt eine Einführung in AR-Systeme, deren technische Grundlagen und das für diese Arbeit verwendete AR-Headset HoloLens 2.

In **Kapitel drei** wird der **Stand der Technik** beschrieben. Es wird zunächst ein Einsatzszenario definiert und anschließend verschiedene in der Industrie eingesetzte Innenraum-Positionierungssyteme und deren charakteristische Parameter vorgestellt. Zusätzlich werden die Vor- und Nachteile dieser Systeme erläutert und dargestellt, welche Erwartungen mit dem neuen Ansatz verbunden sind. Zuletzt erfolgt eine Darstellung von wissenschaftlichen Ansätzen zur Nutzung von Positionsdaten für die Analyse und Optimierung von Prozessen.

Es folgt in **Kapitel vier** die **Anforderungsanalyse** der zu entwickelnden Anwendung, beginnend mit einer Einteilung des Systems in verschiedene Teilsysteme. Im Anschluss werden die Anforderungen für die jeweiligen Subsysteme festgelegt.

Im nachfolgenden **Kapitel fünf** wird der **Detailentwurf** der einzelnen Teilsysteme unter Berücksichtigung der zuvor festgelegten Anforderungen beschrieben. Es werden Konzepte, Methoden und Programmabläufe entwickelt und festgelegt, welche Programmiersprachen und welche Code-Bibliotheken für die Anwendungsentwicklung eingesetzt werden.

Kapitel sechs befasst sich mit der **Implementierung** der entworfenen Anwendung. Es wird die Umwandlung der vorher entwickelten Systeme, Methoden und Strukturen in Programmcode dokumentiert.

In **Kapitel sieben** folgt die **Validierung**. Zunächst wird die Durchführung der photogrammetrischen Messreihe zur Bestimmung der Lokalisierungsgenauigkeit beschrieben und deren Ergebnisse dokumentiert. Zusätzlich wird die Anwendung anhand des beschriebenen Beispiels der kollaborativen Getriebemontage praktisch erprobt und gezeigt, wie Bearbeitungs- und Übergangszeiten aus den hierbei erfassten Positionsdaten berechnet werden können.

Zuletzt erfolgt in **Kapitel acht** die **Diskussion** der Ergebnisse. Das entwickelte System wird mit den zuvor ausgewählten Alternativsystemen in Bezug auf Lokalisierungsgenauigkeit, Implementierungsaufwand und Kosten verglichen und überprüft, ob die erwarteten Vorteile eingetroffen sind. Anhand der durchgeführten praktischen Erprobung wird zudem die industrielle Anwendbarkeit qualitativ bewertet.

Die Arbeit wird abgeschlossen durch **Kapitel neun**. Dieses enthält eine **Zusammenfassung** und zeigt mögliche wissenschaftliche Anknüpfungspunkte auf.

2 Grundlagen

2.1. Industrie 4.0

2.1.1. Geschichte, Definition und aktuelle Entwicklung

Wirft man einen Blick in die Geschichte der industriellen Produktion, so finden sich im Laufe der Jahrhunderte mehrere große, durch technische Innovationen ausgelöste Umbrüche. Die Entwicklungen waren so bedeutsam, dass sie nicht nur die Industrie an sich, sondern gleichzeitig auch ganze Gesellschaften veränderten. Mitte des 18. Jahrhunderts lösten die Entwicklung von Maschinen zur Mechanisierung handwerklicher Tätigkeiten in Verbindung mit der Erfindung der Dampfmaschine die erste sogenannte industrielle Revolution aus. Während die Wertschöpfung vorher vor allem in dezentralen Manufakturen stattfand, verlagerte diese sich im Zuge der industriellen Revolution immer mehr in zentralisierte Fabriken, wo die Arbeit unter Einsatz der neu entwickelten Maschinen stattfinden konnte. Neben einer erhöhten Produktivität sorgte diese Entwicklung auch für die Entstehung einer neuen urbanen Industriegesellschaft. Ab ca. 1870 begann die zweite industrielle Revolution. Durch neue Formen der Arbeitsorganisation (Fließbandarbeit, Taylorismus) und den Einsatz von elektrischer Energie wurde eine Massenproduktion von Gütern ermöglicht, um der Nachfrage einer wachsenden Bevölkerung gerecht zu werden. In den 60er-Jahren des 20. Jahrhunderts folgte die nächste und damit dritte industrielle Revolution. Durch Fortschritte im Bereich der Elektronik und Informationstechnologie wurde ein immer höherer Grad der Automatisierung erreicht, was eine effiziente Serienproduktion mit großem Variantenreichtum ermöglichte [1].

Eine Möglichkeit, die Veränderungen der drei industriellen Revolutionen zu charakterisieren, ist deren Einfluss auf die Produktvielfalt und das Produktionsvolumen je Variante. In Abb. 2.1 ist dargestellt, wie sich diese beiden Größen im Laufe der Zeit entwickelt haben. Bis zur dritten industriellen Revolution Mitte des 20. Jahrhunderts nahm die Vielfalt an Produkten deutlich ab, während das Produktionsvolumen einzelner Varianten immer weiter stieg. Dies lässt sich durch die voranschreitende Mechanisierung erklären, welche zwar die Produktion größerer Mengen ermöglichte, jedoch auch eine gewisse Standardisierung der Produkte erforderte. Sinnbildlich für diese Entwicklung ist der viel zitierte Satz aus den Memoiren von Henry Ford (1863-1947) „Jeder Kunde kann ein Auto in jeder gewünschten Farbe haben, so lange es schwarz ist“ [2]. Erst mit Beginn der dritten industriellen Revolution kehrte sich dieser Trend um. Durch die Möglichkeit, Prozesse elektronisch zu steuern und zu automatisieren, wurde es möglich, vielfältigere Produktvarianten in kleineren Stückzahlen herzustellen und so auf individuelle Kundenanforderungen einzugehen. Auch in der Gegenwart dauert diese Entwicklung an und wird durch die fortschreitende Globalisierung und die wachsende internationale Nachfrage, welche eine Regionalisierung von Produkten erfordert, nochmals vergrößert. In diesem sich wandelnden Marktumfeld steigt die Komplexität für

Unternehmen deutlich an, was sich negativ auf die Wirtschaftlichkeit der Produktion auswirkt und den in Industrieländern wie Deutschland bereits vorhandenen ökonomischen Druck einer globalisierten Wirtschaft nochmals verstärkt. [1]

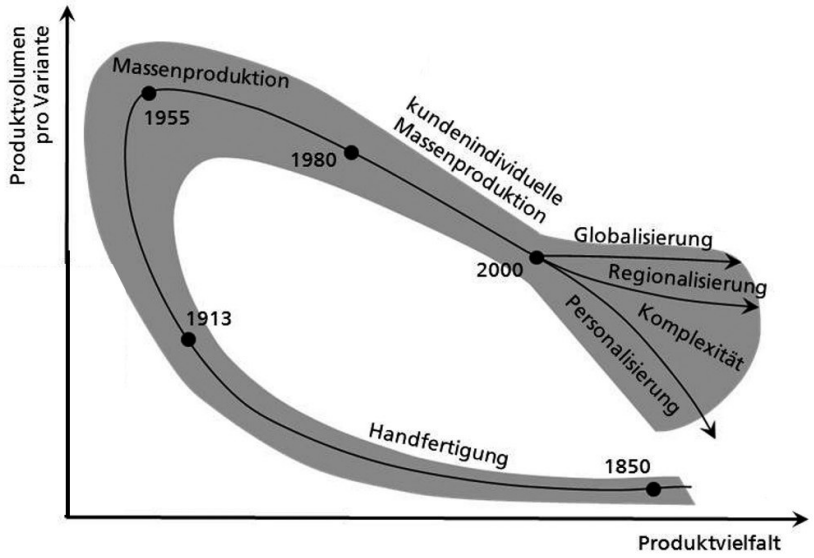


Abbildung 2.1.: Zeitliche Veränderung von Produktvielfalt und Produktionsmengen nach [1]

Um weiterhin wettbewerbsfähig zu bleiben, müssen neue Technologien und Methoden entwickelt und eingesetzt werden. Insbesondere die konsequente Digitalisierung über den gesamten Produktlebenszyklus hinweg wird hierbei als notwendig angesehen. Kern der Vision der Industrie 4.0 bildet ein vollständig digitalisiertes und vernetztes Produktionssystem, welches sich selbst steuert, optimiert und damit die Wirtschaftlichkeit der Produktion erhöht [1]. Um diese Zukunftsvision zu beschreiben, hat sich in Deutschland der Begriff „Industrie 4.0“ etabliert. Die Zahl Vier spielt hierbei auf eine potenzielle vierte industrielle Revolution an, welche durch die weitreichenden Veränderungen der Industrie entstehen könnte. Die Schreibweise mit Dezimalpunkt verdeutlicht die Verbindung des Begriffs zu modernen digitalen Technologien, da auch Softwareversionen üblicherweise in dieser Form notiert werden. [3]

2.1.2. Cyber-physische Systeme (CPS), Smarte Fabriken und das Internet der Dinge

Den Kern der Industrie 4.0 bildet ein vernetztes und sich selbst optimierendes Produktionssystem. Oftmals wird hierfür der Begriff Smarte Fabrik (engl.: Smart Factory) verwendet. Maschinen und Roboter innerhalb einer solchen smarten Fabrik sind nicht bloß Automaten, die eine festgelegte Aufgabe ausführen, sondern entscheiden durch die gezielte Auswertung von Daten und Informationen selbst, wie der Produktionsprozess optimal ausgeführt werden kann [4]. Um dies zu erreichen, kommunizieren die Maschinen kontinuierlich miteinander. Für die Kommunikation zwischen einzelnen Komponenten einer Produktionsanlage hat sich

die Bezeichnung Industrielles Internet der Dinge (engl.: Industrial Internet of Things, IIoT) durchgesetzt [5].

Damit Roboter und Maschinen im Internet der Dinge kommunizieren und Daten austauschen können, benötigen sie entsprechende digitale Schnittstellen. Physische Systeme, die über eine solche Schnittstelle verfügen, werden als cyber-physisches System (engl.: cyber-physical system, CPS) bezeichnet. Der schematische Aufbau eines CPS ist in Abb. 2.2 dargestellt. Das CPS erfasst über Sensoren Daten aus der physikalischen Welt und verarbeitet diese mit sogenannten eingebetteten Systemen (engl.: embedded systems), also eingebauten Computersystemen, bestehend aus Hard- und Software. Diese eingebetteten Systeme ermöglichen dann ebenfalls die Kommunikation mit anderen Bestandteilen des Produktionssystems über das Internet der Dinge. Gesendet und empfangen werden je nach Anwendungsfall Sensor-Rohdaten, daraus abgeleitete Informationen oder konkrete Anweisungen. Über Aktuatoren kann das CPS die physische Welt beeinflussen und somit den eigentlich wertschöpfenden Akt ausführen. Gesteuert werden die Aktuatoren von den digitalen Systemen auf Grundlage der erfassten und empfangenen Daten [1].

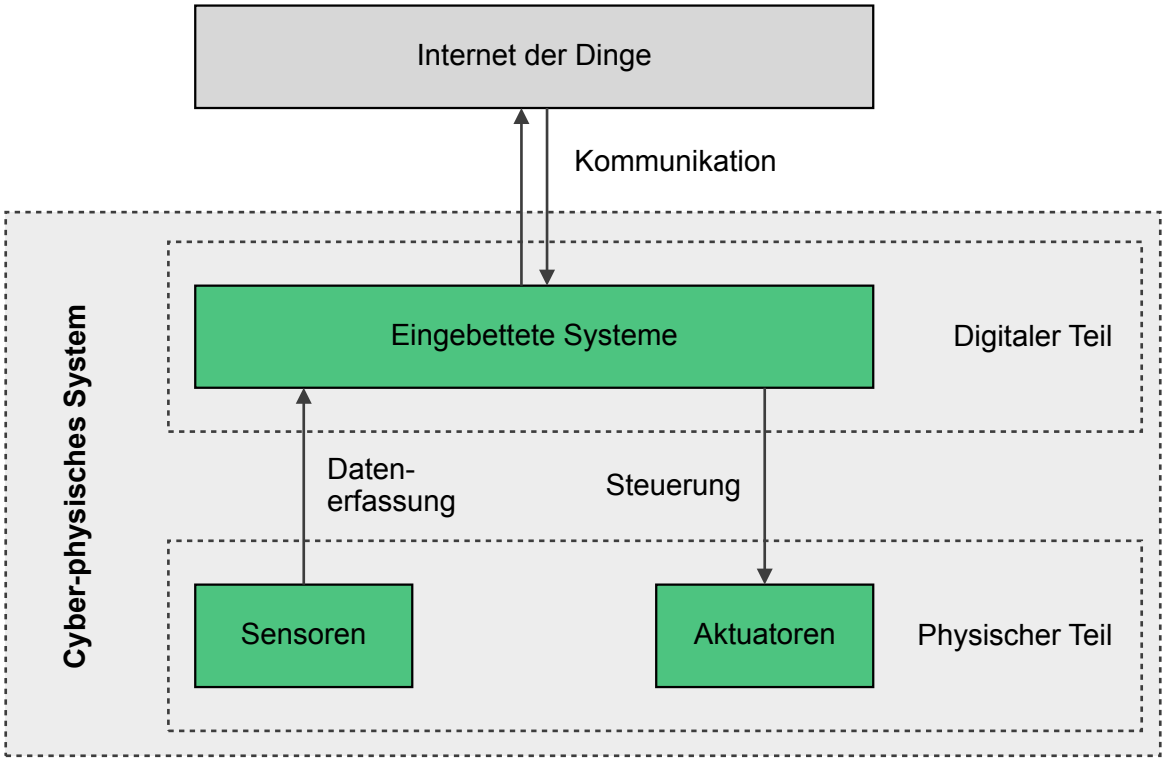


Abbildung 2.2.: Schematischer Aufbau eines cyber-physischen Systems nach [1]

2.2. Mensch-Roboter-Kollaboration

Trotz der fortschreitenden Automatisierung von Produktionssystemen spielt der Mensch weiterhin eine entscheidende Rolle in modernen Produktionsprozessen. Während der Mensch auf der einen Seite eine immer weiter übergeordnete Rolle einnimmt und beispielsweise die

Produktionsstrategie vorgibt, die von den dezentral organisierten CPS umgesetzt wird, wird jedoch mittelfristig auch der direkte Einsatz von Menschen als Arbeiter in Produktionssystemen von Nöten sein. Nach wie vor existiert eine Vielzahl von Anwendungsfällen, bei denen die vollständige Automatisierung technisch unmöglich oder unwirtschaftlich ist [6].

Insbesondere in der Kleinserien- oder Einzelteilfertigung sowie bei komplexen Montageprozessen kann es zielführend sein, einzelne Prozessschritte durch Roboter zu automatisieren und so eine Entlastung des Arbeiters oder eine Verringerung der Fehlerrate zu erreichen. Man spricht hierbei von Mensch-Roboter-Kollaboration (MRK). Da herkömmliche Industrieroboter nur in abgetrennten Bereichen operieren dürfen, sind für die effiziente Kollaboration von Mensch und Roboter in nächster Nähe zueinander spezielle, für diesen Einsatzzweck entwickelte Roboter erforderlich. Diese oft als kollaborative Roboter (engl.: collaborative robots, Cobots) bezeichneten Assistenten werden durch eine möglichst leichte Bauweise mit geringer Massenträgheit und eine empfindliche Sensorik zur Erkennung von Kollisionen charakterisiert. So können Verletzungen des Arbeiters durch Zusammenstöße vermieden werden. Zusätzlich verfügen viele Modelle über möglichst einfache und intuitive Konzepte für die direkte Programmierung und Steuerung der Roboter am Arbeitsplatz, um dem Einsatzzweck in flexiblen Produktionsszenarien gerecht zu werden [7].

Das Institut für Systemarchitekturen in der Luftfahrt des Deutschen Zentrums für Luft- und Raumfahrt (DLR) betreibt eine kollaborative Montagelinie als Versuchsanlage und erforscht am Beispiel der Montage von Flugzeugkabinenkomponenten Automatisierungsansätze. Im Vordergrund steht hierbei insbesondere der Digitale Faden, also die Durchgängigkeit von Daten in verschiedenen Phasen des Produktlebenszyklus vom Entwurf bis hin zur Fertigung. Zur Verfügung stehen insgesamt drei verschiedene kollaborative Robotertypen. Für das in dieser Arbeit gezeigte Anwendungsbeispiel kommt das Modell UR10e der Firma Universal Robots zum Einsatz [8].

2.3. Der Industriearbeiter als cyber-physisches System

Für die gesamtheitliche Umsetzung des Ansatzes der Industrie 4.0 lässt sich das Prinzip der CPS auch auf den menschlichen Arbeiter übertragen. Das Grundprinzip ist das gleiche: Es wird eine technische Schnittstelle geschaffen, um dem Menschen relevante Daten und Anweisungen des Produktionssystems zur Verfügung zu stellen und gleichzeitig Daten und Anweisungen des Menschen entgegen zu nehmen. Für die Kommunikation von Maschine zu Mensch ist eine solche Schnittstelle technisch einfach umsetzbar. Durch die Verwendung von Bildschirmen oder Lautsprechern können dem Industriearbeiter Informationen übermittelt werden. Auch der Einsatz von AR-Systemen ist ein vielversprechender Ansatz zur Informationsdarstellung, der bereits in zahlreichen Veröffentlichungen diskutiert und erprobt wurde [9].

Für die Kommunikation in entgegengesetzter Richtung, also von Mensch zu Maschine, müssen zwei Fälle unterschieden werden. Zum einen die bewusste oder aktive und zum

anderen die unbewusste oder passive Kommunikation. Aktive Kommunikation kann durch für diesen Zweck ausgelegte technische Komponenten erfasst werden. Im einfachsten Fall sind dies Schalter, Tasten oder berührungsempfindliche Oberflächen. Modernere Ansätze umfassen den Einsatz von AR-Systemen, um Eingaben virtuell zu tätigen [9] sowie die Steuerung mit Gesten [10] oder Spracheingaben [11].

Ebenfalls relevant, jedoch weit weniger erforscht ist die Erfassung und Nutzung von passiven menschlichen Daten, also Daten, die ohne Mitwirkung des Arbeiters aufgezeichnet und an das Produktionssystem gesendet werden. Diese Arbeit befasst sich damit, wie solche passiven menschlichen Daten und Parameter mit möglichst simpler technischer Infrastruktur automatisiert mit einem AR-Headset erfasst werden können. Um sich der Thematik zu nähern, werden im folgenden Abschnitt zunächst die relevanten passiven menschlichen Datenquellen erläutert.

2.3.1. Passive menschliche Datenquellen

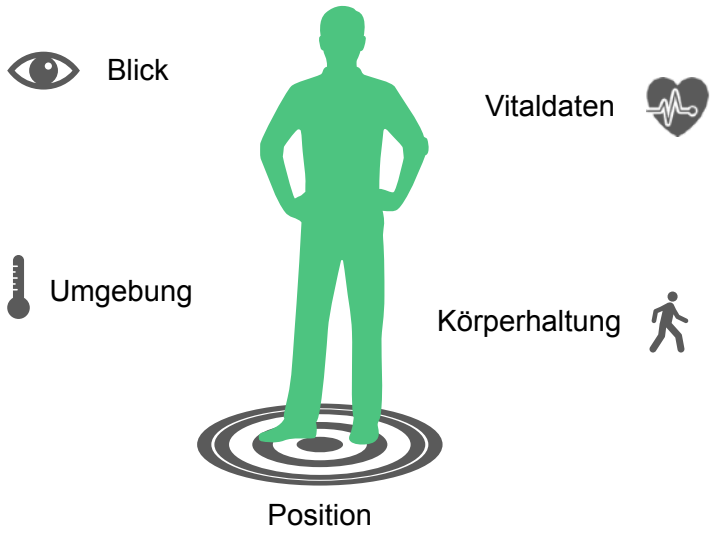


Abbildung 2.3.: Passive menschliche Datenquellen

Abb. 2.3 zeigt eine Übersicht der für die Industrieanwendung relevanten menschlichen Datenquellen, die im Folgenden erläutert werden:

- **Blickrichtung:** Die Blickrichtung dient als Indikator dafür, worauf die Aufmerksamkeit eines Menschen liegt. Zusätzlich kann sie ein Hinweis darauf sein, welche Intention ein Mensch hat.
- **Körperhaltung**: Unter Körperhaltung versteht man die Position und Stellung einzelner Körperteile in Relation zueinander. Durch die kontinuierliche Erfassung der Körper-

haltung können z. B. Bewegungsabläufe dokumentiert und ausgewertet, Kollisionen vermieden oder Belastungssituationen erkannt werden.

- **Umgebungsdaten**: Physikalische Parameter der Arbeitsumgebung haben einen direkten Einfluss auf das Wohlbefinden und die Produktivität eines Arbeiters. Beispiele für relevante Umgebungsdaten sind die Temperatur, Luftfeuchtigkeit, Lärm-, Strahlen- oder Schadstoffbelastung.
- **Vitaldaten**: Vitaldaten geben Auskunft über das körperliche Befinden des Menschen und können ein Indikator für gesundheitliche Gefahren und den Grad der Belastung sein. Beispiele für Vitaldaten sind Puls, Blutdruck oder Blutsauerstoffgehalt.
- **Position**: Die Position eines Arbeiters im Raum kann entweder in absoluten Koordinaten (2D oder 3D) bestimmt werden oder aber diskrete Messung (Anwesenheit/Abwesenheit).

Die Erfassung dieser menschlichen Daten ist deutlich komplexer als das Erfassen von Daten technischer Produktionssysteme, da hier naturgemäß keine elektronische Steuerung vorhanden ist, an der Daten gesammelt und verarbeitet werden können. Aus diesem Grund sind zusätzliche Sensoren notwendig, welche entweder fest im Raum verankert oder aber vom Arbeiter am Körper getragen werden (engl.: Wearables). Zu den Wearables zählen auch AR-Headsets, durch deren Sensoren eine Vielzahl an Daten erfasst werden kann. In Abschnitt 2.5.2 wird beschrieben, welche Sensoren in dem für diese Arbeit verwendeten AR-Headset der Microsoft HoloLens 2 verbaut sind und welche menschlichen Daten mit diesen erfasst werden können.

2.3.2. Anwendungsbeispiele aus der Wissenschaft

Im Folgenden werden Beispiele für die Anwendung passiver menschlicher Daten in der Wissenschaft vorgestellt:

Anwendungen für Positionsdaten

Menschliche Positionsdaten geben Aufschluss darüber, wo ein Arbeiter sich in der Positionsanlage befindet. In der Literatur lassen sich zahlreiche Anwendungsfälle für die Nutzung der Daten finden. So können diese beispielsweise genutzt werden, um Kollisionen zwischen Menschen und Robotern [12] oder Menschen und fahrerlosen Transportsystemen [13] zu verhindern oder deren Bewegungsabläufe vorausschauend umzuplanen. Positionsdaten können zudem genutzt werden, um Arbeitern interaktive Navigationsanweisungen zu geben und so Laufwege zu optimieren [14, 15] oder um die Ausführung von Prozessen zu überwachen und zu dokumentieren [16, 17]. Wird die Position eines Arbeiters dauerhaft erfasst, können aus den Positionsdaten Durchführungszeiten für Prozesse oder Wegezeiten

errechnet werden [18, 19]. Dieser Anwendungsfall dient auch im Rahmen dieser Arbeit als Anwendungsbeispiel für die Analyse und Nutzung von Arbeiter-Positionsdaten.

Anwendungen für Daten zur Körperhaltung

Die Gesundheit von Mitarbeitern rückt gegenwärtig immer weiter in den Fokus. Insbesondere bei körperlich belastenden Tätigkeiten bietet die Auswertung der Körperhaltung eines Arbeiters großes Potenzial für Optimierungen. So können durch eine ergonomische Analyse der Arbeitsprozesse besonders belastende Tätigkeiten mit erhöhter Verletzungsgefahr identifiziert und klassifiziert werden [20, 21] oder der Arbeiter direkt über unergonomische Körperhaltungen informiert werden [22].
Durch eine Erfassung der Körperhaltung und insbesondere durch die Verfolgung von Handbewegungen können Aktivitäten eines Arbeiters erkannt werden. Dies ermöglicht z. B. die Qualitätskontrolle von durchgeführten Prozessen [22, 23], die feingranulare Messung von Prozesszeiten [19] oder die Vermeidung von Kollisionen mit kollaborativen Robotern [24, 25]. Zusätzlich können einem Arbeiter aktivitätsbezogene Informationen wie beispielsweise eine Verfahrensanleitung für den derzeit ausgeführten Prozessschritt angezeigt werden [22, 26].

Anwendungen für Daten zur Blickrichtung

Die Blickrichtung kann ein aussagekräftiger Indikator dafür sein, worauf die Aufmerksamkeit eines Menschen liegt. Durch eine systematische Auswertung der Blickrichtung kann beispielsweise die Effizienz bei der Übergabe von Aufgaben zwischen Roboter und Mensch erhöht werden [27]. Zudem kann die Blickrichtung als Baustein für die Abschätzung menschlicher Intention dienen, was zur Vermeidung von Kollisionen bei der kollaborativen Arbeit genutzt werden kann [28].

Anwendungen für Vitaldaten

Der naheliegende Nutzen einer systematischen Auswertung von Vitaldaten wie der Herzfrequenz ist der Schutz vor Überlastungen des Industriearbeiters bei körperlich anstrengenden Tätigkeiten [29, 30]. Es existieren zudem Ansätze, mittels einer Analyse der Vitaldaten Anzeichen von Stress zu erkennen. Durch ein gezieltes Gegensteuern können so stressbedingte Arbeitsfehler vermieden werden [31].

Anwendungen für Umgebungsdaten

Die dauerhafte Erfassung von Zustandsdaten der Produktionsumgebung kann genutzt werden, um die Arbeitssicherheit für den Menschen zu erhöhen. Durch die gezielte Platzierung von Sensoren in Gefahrenbereichen oder in tragbaren Geräten können Arbeiter rechtzeitig gewarnt werden und die Daten genutzt werden, um die Prozesse in Bezug auf die Gefahreneindämmung zu optimieren [32].

Tabelle 2.1 gibt eine Übersicht über die beschriebenen Anwendungsfälle und listet die genannten wissenschaftliche Referenzen auf. In dieser Arbeit wird der Anwendungsfall untersucht, Prozess- und Wegezeiten aus mit dem AR-Headset gemessenen Positionsdaten zu bestimmen. In Kapitel 3 wird der Stand der Wissenschaft und Technik in Bezug auf diesen Anwendungsfall beschrieben und Verfahren zur Bestimmung von menschlichen Positionsdaten vorgestellt.

Tabelle 2.1.: Beispiele für wissenschaftliche Anwendungsfälle passiver menschlicher Daten

Datenquelle	Anwendung	Referenzen
Position	Kollisionsvermeidung	[12, 13]
	Messung von Prozess- und Wegezeiten	[18, 19]
	Laufwegoptimierung (Navigation)	[14, 15]
	Überwachung & Dokumentation (quantitativ)	[16, 17]
Körperhaltung	Ergonomische Auswertung und Optimierung	[20, 21, 22]
	Kollisionsvermeidung	[24, 25]
	Qualitätskontrolle	[22, 23]
	Aktivitätsbezogene Informationsdarstellung	[22, 26]
	Messung von Prozesszeiten	[19]
Blick	Optimierung Mensch-Roboter-Interaktion	[27]
	Arbeitssicherheit	[28]
Vitaldaten	Schutz vor Überlastung	[29, 30]
	Stresserkennung (Fehlervermeidung)	[31]
Umgebungsdaten	Arbeitssicherheit	[30, 32]

2.4. Augmented Reality

2.4.1. Definition

Die menschliche Wahrnehmung basiert auf der Kombination verschiedener Sinneseindrücke wie Hören, Sehen oder Riechen, die über spezialisierte Organe erfasst werden. Der mit Abstand wichtigste Sinn ist der über das Auge wahrgenommene Sehsinn. Schon seit einigen Jahrzehnten beschäftigen sich Wissenschaftler mit der Frage, wie visuelle Eindrücke des Menschen mithilfe von technischen Systemen beeinflusst oder vollständig durch künstliche Eindrücke ersetzt werden können. In diesem Zusammenhang sind die häufig gebrauchten Bezeichnungen Virtual Reality (VR; dt.: Virtuelle Realität), Augmented Reality (AR; dt.: Erweiterte Realität) und Mixed Reality (MR; dt.: Gemischte Realität) entstanden [33].

Für diese Begriffe existiert eine Vielzahl an Definitionen und Abgrenzungsansätzen. Viel beachtet ist das Konzept des Realitäts-Virtualitäts-Kontinuums von Paul Milgram, welches schematisch in Abb. 2.4 dargestellt ist. Dieses definiert die gemischte Realität als den

gesamten Zwischenraum zwischen der natürlichen Realität und der vollständig virtualisierten Realität. Die gemischte Realität wird nochmals unterteilt in die erweiterte Realität und die erweiterte Virtualität, welche theoretisch fließend ineinander übergehen können [34]. Als erweiterte Virtualität versteht man eine virtuelle Umgebung, in die einzelne reale Objekte eingeblendet werden. Ein Beispiel hierfür ist die bildliche Darstellung von realen Mitspielern bei Computerspielen.

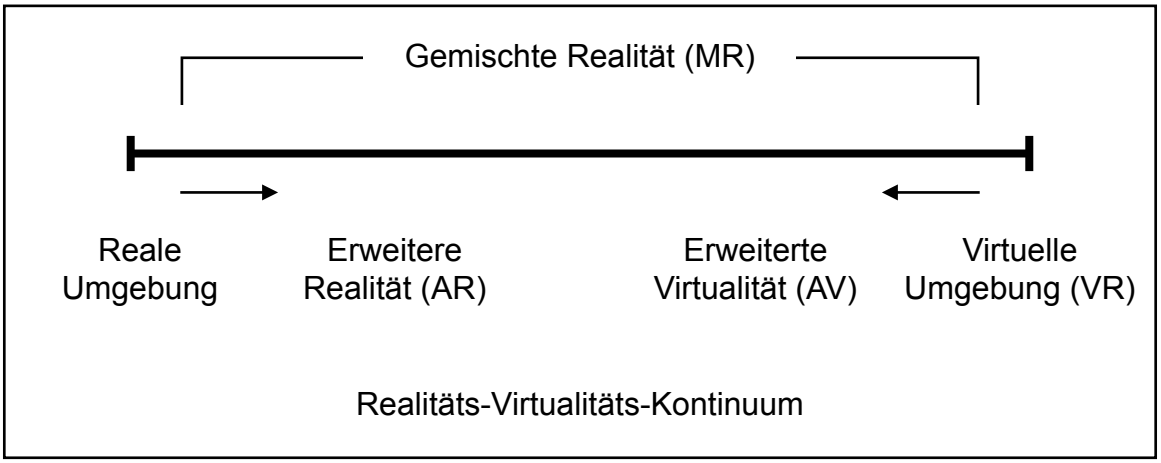


Abbildung 2.4.: Realitäts-Virtualitäts-Kontinuum nach Paul Milgram u. a. [34]

Bei der erweiterten Realität stehen reale Seheindrücke im Vordergrund. Diese werden durch die Einblendung einzelner virtueller Objekte ergänzt, mit denen der Nutzer oftmals auch interagieren kann. Charakteristisch für AR-Systeme ist das Ziel, dass virtuelle Objekte optisch wie reale Objekte dargestellt werden und beispielsweise bei Bewegung des Anwenders an der gleichen Stelle im Raum verbleiben. Diese Eigenschaft grenzt AR-Systeme von anderen Technologien wie z. B. einfachen Head-Up-Diplays oder sogenannten Smartglasses ab, bei denen die virtuellen Inhalte immer am gleichen Ort im Sichtfeld des Anwenders verbleiben [35, 36].

In den letzten Jahren hat sich die Verbreitung von Augmented Reality Systemen stark vergrößert, sodass inzwischen eine Vielzahl von AR-Anwendungen und -Systemen sowohl im Konsumentenbereich als auch im professionellen Bereich existieren. In den folgenden Abschnitten werden die technischen Grundlagen moderner AR-System erläutert und anhand von Beispielen industrielle Anwendungsfälle vorgestellt.

2.4.2. Technische Grundlagen von AR-Systemen

Gemäß der oben stehenden Definition ermöglichen es Augmented Reality Systeme, digitale Objekte in realen Umgebungen darzustellen und mit diesen zu interagieren. Um dies zu

erreichen, werden technische Geräte eingesetzt, die über Komponenten zur Datenausgabe und -eingabe verfügen. Die Ausgabe erfolgt vor allem optisch über Displays, mit denen der Anwender entweder nur die virtuellen Objekte oder aber die gesamte mit digitalen Objekten angereicherte Szene sehen kann. Die kontinuierliche Dateneingabe hat zwei Ziele. Zum einen gibt sie dem Anwender die Möglichkeit, mit den virtuellen Objekten zu interagieren, was beispielsweise durch Gesten, Controllereingaben oder die Bedienung berührungsempfindlicher Bildschirme erfolgen kann [35]. Auf der anderen Seite ist ein AR-System für die perspektivisch korrekte Darstellung der virtuellen Inhalte darauf angewiesen, die Umgebung dauerhaft sensorisch zu erfassen.

Ein Grundprinzip von allen AR-Systemen ist die kontinuierliche Erfassung der System-Position in Relation zu einem bestimmten Punkt oder dem gesamten Raum, das sogenannte Tracking (dt. Verfolgen). Für das Tracking existieren zwei Grundprinzipien, das markerbasierte und das markerlose Tracking. Markerbasiert bedeutet, dass das System über Kameras optische Marker erfasst und die virtuellen Objekte relativ zu diesen positioniert. Das markerlose Tracking ermöglicht die Positionsbestimmung auch ohne ortsfeste Marker, indem möglichst viele Punkte und Geometrien im Raum dauerhaft erfasst werden. Bei der Bewegung des Systems kann die relative Positionsveränderung aus der Verschiebung der Punkte und Geometrien bestimmt werden. Im Zusammenhang mit dem markerlosen Tracking steht auch das sogenannte SLAM-Verfahren, welches in Abschnitt 3.2 näher beschrieben wird. Für das Tracking verfügen viele AR-Systeme neben Kameras noch über andere Sensoren, die die Positionsbestimmung unterstützen. Beispiele sind Inertial-, Laser- oder Ultraschallsensoren [37].

AR-Systeme lassen sich in zwei Hauptgruppen unterteilen, welche den genannten Grundprinzipien unterliegen aber maßgebliche Unterschiede in der Art der visuellen Ausgabe und der Möglichkeit, Eingaben zu tätigen aufweisen. Man unterscheidet Handheld-Geräte und Head-Mounted-Displays(HMDs), welche in Abb. 2.5 dargestellt sind.

Handheld-Gerät

Head-Mounted-Display (HMD)

Abbildung 2.5.: Verschiedene Arten von AR-Systemen [38, 39]

Handheld-Geräte

Der Begriff Handheld beschreibt gemäß der Übersetzung aus dem Englischen Geräte, die der Anwender in der Hand hält, in die alle notwendigen technischen AR-Komponenten integriert sind. Zwischenzeitlich sind hierfür keine speziellen Geräte mehr notwendig, da moderne Konsumentengeräte wie Tablets oder Smartphones mit den verbauten Displays als Darstellungsmedium und Kameras als Umgebungssensoren bereits AR-Anwendungen ermöglichen. Nutzereingaben können vom Anwender direkt über die berührungsempfindlichen Bildschirme getätigt werden [35]. Insbesondere die großen Fortschritte bei der maschinellen Bildverarbeitung und der immer größeren Rechenleistung solcher Geräte beschleunigen diese Entwicklung zunehmend. Hersteller wie Apple richten ihre Produkte speziell für die Anwendung als AR-System aus, indem zusätzliche Sensoren z. B. basierend auf dem LIDAR-Verfahren (Light Detection And Ranging) [40] zur Unterstützung des Tracking verbaut werden und spezielle Programmierbibliotheken und -schnittstellen bereitgestellt werden [38]. Vorteil von AR-Anwendungen für diese Art von Systemen ist die ohnehin schon große Verbreitung von AR-fähigen Endgeräten, sodass AR-Anwendungen dem Nutzer einfach zugänglich gemacht werden können. Nachteilig sind die Limitierung der Darstellung auf das verbaute Display und die Notwendigkeit, das Gerät in der Hand zu halten, was es erschwert, AR-Anwendungen zu realisieren, die eine manuelle Tätigkeit unterstützen.

Head-Mounted-Displays (HMDs)

HMDs sind AR-Systeme, die vom Anwender ähnlich wie ein Helm auf dem Kopf getragen werden und integrierte Bildschirme direkt vor den Augen des Trägers positionieren. AR-HMDs werden oftmals als AR-Headsets oder AR-Brillen bezeichnet und weisen große Ähnlichkeit zu VR-Headsets auf. Auch VR-Headsets können durch spezielle Anwendungen zu AR-Headsets werden, indem die durch das VR-Headset verdeckte Sicht mit Kameras aufgezeichnet wird und mit virtuellen Inhalten ergänzt in Echtzeit auf die Bildschirmeinheit gespielt wird. Um unerwünschte Nebenwirkungen wie z. B. Cybersickness [41] zu minimieren, kommen für spezialisierte AR-Systeme halbtransparente Bildschirme zum Einsatz, auf Englisch als Optical See-Through-Displays (OST-Displays) bezeichnet, auf denen nur die virtuellen Objekte eingeblendet werden und somit die reale Wahrnehmung ergänzen. Dieser Kompromiss führt jedoch dazu, dass die virtuellen Objekte eine gewisse Transparenz aufweisen und in hellen Umgebungen meist schlechter wahrnehmbar sind. Hersteller behelfen sich hierfür mit verdunkelnden Scheiben vor den Displays, was die Anwendbarkeit der Systeme durch die insgesamt verminderte Sehleistung negativ beeinflussen kann [35]. Am Markt verfügbare Produkte aus dieser Kategorie sind beispielsweise die HoloLens der Firma Microsoft [42] oder die Magic Leap des gleichnamigen Herstellers [43].

Der entscheidende Vorteil von HMD-Systemen im Vergleich zu Handheld-Systemen liegt in der Möglichkeit, zusätzlich zur Nutzung des AR-Systems noch andere manuelle Tätigkeiten ausführen zu können. Um Eingaben mit HMD-Systemen tätigen zu können, setzen viele Hersteller deshalb auf die Steuerung mit Handgesten, die durch im Gerät verbaute Sensoren erfasst werden [37].

2.4.3. Einsatzszenarien für AR in der Industrie

Sowohl in der Forschung als auch in produktiven Anlagen existiert eine Vielzahl von Anwendungen, welche die Eigenschaften von Augmented Reality Systemen im Kontext industrieller Produktion nutzbar machen sollen. Nachfolgend werden Beispiele für den Einsatz von AR-Systemen in der Industrie vorgestellt.
Ein im Jahr 2019 veröffentlichtes wissenschaftliches Review kategorisiert Veröffentlichungen zu industriellen AR-Anwendungen anhand ihres Einsatzszenarios [44]. Die am häufigsten vertretenen Anwendungsfälle werden im folgenden Abschnitt beschrieben:

- **Montage**: AR-Systeme können menschliche Arbeiter dabei unterstützen, komplexe Montageprozesse effizient auszuführen. Insbesondere bei Montageprozessen bestehend aus vielen einzelnen Prozessschritten kann der Mensch durch AR unterstützt werden, indem z. B. die Montagereihenfolge oder Informationen zum Prozess virtuell im räumlichen Kontext dargestellt werden [45].
- **Wartung**: Bei der Wartung von Systemen können mittels AR beispielsweise digital verfügbare Daten und Informationen direkt im Sichtfeld des Anwenders angezeigt und somit Zeit eingespart werden [46].
- **Produktdesign**: Bei der Entwicklung neuer Produkte und Systeme können AR-Systeme angewendet werden, um digitale Modelle dreidimensional anzuzeigen und mit diesen interagieren zu können. Beispielhaft hervorzuheben ist die Möglichkeit, Produkte oder Einzelteile ohne die Herstellung von haptischen Prototypen im späteren physischen Umfeld zu betrachten [47].
- **Arbeitssicherheit**: AR-Systeme können zur Erhöhung der Arbeitssicherheit beitragen, indem beispielsweise Sicherheitsinformationen oder Warnungen direkt in das Sichtfeld des Anwenders eingeblendet werden. Ein Beispiel ist die Darstellung der geplanten Bewegungen eines Roboters oder fahrerlosen Transportsystems [28].
- **Fernunterstützung**: Um die physische Anwesenheit von Experten beispielsweise bei notwendigen Reparaturen zu verringern, wurden AR-basierte Ansätze entwickelt, um einen Nicht-Experten von der Ferne aus zu unterstützen. Beispielsweise existieren Anwendungen, bei denen ein Experte auf einem Monitor Hilfestellungen anhand eines dreidimensionalen Modells des zu reparierenden Systems geben kann, welche dem Nicht-Experten über ein AR-System angezeigt werden [48].
- **Training**: Große Kostenfaktoren für Unternehmen sind das Training und die Weiterbildung von Mitarbeitern. AR-Systeme können hierbei unterstützen, indem z. B. Trainingsobjekte virtuell eingeblendet werden und somit ein ortsunabhängiges Training mit einem verringerten Einsatz von physischen Demonstratoren möglich macht [49].

2.5. Microsoft HoloLens 2

Die im Rahmen dieser Arbeit entwickelte Anwendung wird mit dem AR-Headset HoloLens 2 der Firma Microsoft realisiert. Im folgenden Abschnitt werden allgemeine Informationen und

technische Spezifikationen des Systems dargestellt und eine Einführung in die Programmierung von AR-Anwendungen für die HoloLens gegeben.

2.5.1. Allgemein

Bei der HoloLens 2 der Firma Microsoft handelt es sich um ein HMD (siehe Abschnitt 2.4.2), welches im Februar 2019 als Nachfolger der HoloLens 1 vorgestellt wurde [50]. Die HoloLens vereint alle technischen Komponenten in einem Gerät, welches durch einen integrierten Akku ohne Kabelverbindungen betrieben werden kann. Während Akku und Recheneinheit an der Hinterseite platziert sind, befindet sich an der Vorderseite die Bildschirmeinheit inklusive der notwendigen Sensoren, welche der Nutzer wie ein Visier aufstellen und absenken kann. Um einen optimalen Tragekomfort und die richtige Position der Bildschirme relativ zu den Augen einzustellen, lässt sich das Headset mit einen Riemen und einem Verstellrad an den Kopf des Anwenders anpassen.

Am Gerät selbst befinden sich mit Ausnahme von Lautstärke- und Helligkeitsreglern keine Schaltflächen für Nutzereingaben. Diese kann der Nutzer ausschließlich durch die Verwendung von Handgesten und das Betätigen von virtuellen Schaltflächen machen, was durch ein permanentes Hand-Tracking ermöglicht wird. Zusätzliche Nutzerschnittstellen ergeben sich durch eingebaute Lautsprecher für die Audiowiedergabe und Mikrofone zur Erfassung von Spracheingaben [42].

Als Zielgruppe für das System lassen sich eindeutig Unternehmen und Forschungseinrichtungen identifizieren. So bietet Microsoft beispielsweise eine spezielle Variante für die Nutzung in Reinräumen oder ein in einen Schutzhelm integriertes Headset an (siehe Abb. 2.6). Auch der Preis von ca. 3800 Euro (Stand: Februar 2022) für die günstigste Variante lässt eine klare Ausrichtung auf institutionelle Kunden erkennen [51]. Für die Erprobung der im Rahmen dieser Arbeit entwickelten Anwendung wird die Standardvariante verwendet. Alle im Folgenden beschriebenen Eigenschaften beziehen sich ebenfalls auf diese Variante.

Abbildung 2.6.: Verschiedene Varianten der Microsoft HoloLens 2 [51]

2.5.2. Verbaute Sensoren

Wie bereits in Abschnitt 2.4.2 beschrieben, ist für die realistische und perspektivisch korrekte Darstellung von virtuellen Objekten in AR-Anwendungen eine relative Bestimmung der Systemposition in Echtzeit notwendig. Zu diesem Zweck besitzt die HoloLens 2 verschiedene Sensoren, deren Daten überlagert und in Echtzeit ausgewertet werden [52]. Ziel der Sensordatenfusion ist es, ein möglichst genaues Tracking in unterschiedlichen Umgebungen zu ermöglichen [53]. Abb. 2.7 zeigt die Position der verbauten Sensoren am Gerät, die im Folgenden beschrieben werden:

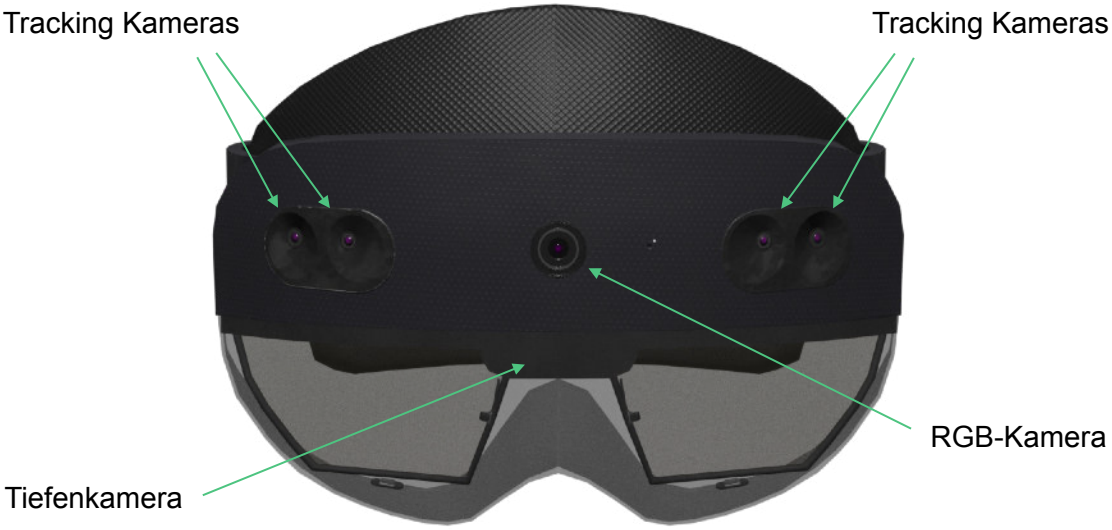


Abbildung 2.7.: Sensoren der HoloLens 2 für die Positionsbestimmung [52]

- **Tracking Kameras:** Die HoloLens 2 verfügt über insgesamt 4 Graustufen-Kameras für das Umgebungs-Tracking. Zwei davon sind in Stereoanordnung nach vorne gerichtet und zwei jeweils nach außen.
- **RGB Kamera:** Mittig über den Augen befindet sich eine Farb-Übersichtskamera. Diese ermöglicht es dem Nutzer, Videos oder Standbilder zu erstellen, in die vom System die gezeigten Hologramme eingefügt werden.
- **Tiefenkamera:** Unterhalb der Übersichtskamera findet sich die Infrarot-Tiefenkamera. Diese arbeitet in zwei verschiedenen Modi. Mit einer Bildrate von 45 FPS scannt sie den Nahbereich bis zu einer Entfernung von 1 m. Die hierbei erfassten Daten dienen vor allem dem Tracking der Hände. Mit einer Bildrate von 1-5 FPS scannt sie den weiter entfernten Bereich.
- **Trägheitsmesseinheit (Inertial Measurement Unit, IMU):** Die IMU besteht aus einem Beschleunigungssensor für die Registrierung von Bewegungen sowie einem Gyroskop und Magnetometer zur Lagebestimmung (IMU ist nicht in Abb. 2.7 sichtbar).

- **Eye-Tracking Kameras:** Zusätzlich sind zwei Infrarotkameras zur Verfolgung der Augen des Trägers in der HoloLens 2 verbaut. Diese ermöglichen dem System eine Bestimmung der Blickrichtung und damit eine perspektivisch korrekte Darstellung der virtuellen Objekte (Eye-Tracking-Kameras sind nicht in Abb. 2.7 sichtbar).

2.5.3. Entwicklung von Anwendungen für die HoloLens 2

Microsoft stellt für die Programmierung von Anwendungen für die HoloLens Programmierbibliotheken für die Entwicklungsumgebungen Unity und Unreal Engine zur Verfügung [54]. Beide Anwendungen wurden ursprünglich für die Entwicklung von Computerspielen entwickelt. Da viele Elemente aus Computerspielen wie die realistische Darstellung von 3D-Objekten oder die Verwendung von Physikmodellen auch für VR- und AR-Anwendungen außerhalb der Computerspiel-Branche relevant sind, ist die Verwendung von Spieleengines für die VR-/AR-Entwicklung weit verbreitet [55].

Für die in dieser Arbeit entwickelte Anwendung kommt die Entwicklungsengine Unity der Firma Unity Technologies zum Einsatz [56]. In Unity lassen sich HoloLens-Anwendungen mit der Sprache C# programmieren und nach einem Kompilierungsprozess auf das AR-Headset übertragen. Zusätzlich kann auf das sogenannte MRTK (Mixed Reality Toolkit) zurückgegriffen werden, ein Open-Source-Projekt, dass maßgeblich von Microsoft entwickelt wird und mit verschiedenen AR-Systemen kompatibel ist. Das MRTK beinhaltet Software-Bausteine für wiederkehrende Elemente und Prozesse in AR-Anwendungen und ermöglicht so kürzere Entwicklungszeiten [57].

3 Stand der Technik

Im Rahmen dieser Arbeit wird eine Methode entwickelt, um Daten zur menschlichen Arbeit in Industrieprozessen mittels eines AR-Headsets zu erfassen und zur Optimierung des Prozesses nutzbar zu machen. Der Fokus soll hierbei auf Positionsdaten des Arbeiters und der Bestimmung von Prozesszeiten aus diesen liegen. Im folgenden Kapitel wird der Stand der Technik in Bezug auf die Positionsbestimmung in Innenräumen dargestellt. Die in diesem Abschnitt beschriebenen Verfahren und deren Parameter und Eigenschaften werden später für die Bewertung des entwickelten AR-Positionsbestimmungssystems verwendet. Zusätzlich werden die Erwartungen an den Einsatz der HoloLens 2 für die Positionsbestimmung festgehalten und Ansätze zur Bestimmung von Prozesszeiten aus Arbeiter-Positionsdaten vorgestellt.

3.1. Positionsbestimmung in Innenräumen

Systeme zur Positionsbestimmung ermöglichen es, die Position eines Objekts in Relation zu einem definierten Bezugspunkt zu bestimmen. Die ältesten bekannten Instrumente zur Positionsbestimmung lassen sich mehrere tausend Jahre zurückdatieren und dienten der Schiffsnavigation auf hoher See. Bezugspunkt hierfür waren meist markante Sterne am Nachthimmel. Die heute wohl bekanntesten Positionierungssysteme, welche auch im Alltag der meisten Menschen eine Rolle spielen, sind unter der Abkürzung GNSS (engl. für global navigation satellite system oder dt. globales Navigationssatellitensystem) bekannt. Der bekannteste Vertreter ist das amerikanische Global Positioning System (GPS), dessen Namen oft auch synonym für GNSS genutzt wird. Grundlage eines GNSS sind um die Erde kreisende Satelliten, welche durch die Laufzeitmessung eines Signals ihren Abstand zu dem Objekt, für das die Position bestimmt werden soll, messen. Durch die Zusammenführung der Abstandsmessergebnisse mehrerer Satelliten kann die Position auf der Erdkugel eindeutig bestimmt werden. GNSS kommen auch in industriellen Anwendungsszenarien zum Einsatz. Ihr Einsatz beschränkt sich jedoch auf die Nutzung außerhalb von Gebäuden, da ansonsten die Satellitensignale nicht empfangen werden können [58].

Für viele industrielle Anwendungen die die Positionsbestimmung eines Objekts erfordern, kommen die weitverbreiteten satellitengestützten Verfahren (GNSS) nicht infrage. Entweder ist die Lokalisierung zu ungenau oder die Anwendung erfordert die Positionsbestimmung in einem Innenraum. Aus diesem Grund wurden eine Reihe an alternativen Verfahren entwickelt, die unter dem Begriff Indoor Positioning Systeme (IPS, dt.: Systeme zur Positionierung in Innenräumen) zusammengefasst werden [58].

Man unterscheidet insgesamt drei mögliche Systemkonfigurationen für IPS [59]:

- **Outside-In-Systeme**: Hierbei handelt es sich um die am weitesten verbreitete Systemkonfiguration, bei der die Positionsdaten mit ortsfesten Sensoren erfasst werden. Je nach Verfahren werden zur Unterstützung der Positionsbestimmung an den zu überwachenden Objekten teilweise Reflektoren oder Signalempfänger angebracht.
- **Inside-Out-Systeme**: Bei dieser Konfiguration befinden sich die Sensoren am Objekt selbst. Die Positionsbestimmung erfolgt durch die Erfassung der Umgebung und deren physikalischer Eigenschaften. Häufig werden verschiedene Sensortypen eingesetzt, deren Daten überlagert werden, um eine möglichst präzise Lokalisierung zu ermöglichen. Ein Vorteil dieser Systeme ist, dass auf feste Infrastruktur verzichtet werden kann und die Systeme flexibel einsetzbar sind.
- **Inside-In-Systeme**: Auch bei Inside-In-Systemen befinden sich die Sensoren am getrackten Objekt selbst. Die Sensoren erfassen ausschließlich inertiale Parameter des Objektes zum Beispiel durch Beschleunigungssensoren, Magnetometer oder Gyroskope. Nachteil dieser Systeme ist, das die Präzision der Positionsbestimmung nach erfolgter Kalibrierung mit der Zeit abnimmt und diese deshalb regelmäßig wiederholt werden muss [58]. Bei vielen Positionsbestimmungsverfahren werden Inside-In-Systeme deshalb unterstützend eingesetzt um schnelle Positionsänderungen erfassen zu können.

Im folgenden Abschnitt wird zunächst das für diese Arbeit maßgebliche Einsatzszenario der Positionsbestimmung dargestellt und Kriterien für die Auswahl geeigneter Positionsbestimmungssysteme definiert. Anhand der Kriterien werden geeignete IPS ausgewählt und deren technische Grundlagen sowie die charakteristischen Eigenschaften dargestellt. Insbesondere im Fokus stehen hierbei die erreichbare Präzision bei der Positionsbestimmung, die Komplexität der Infrastruktur und die ungefähren Anschaffungskosten. Die ermittelten Eigenschaften bilden schließlich die Grundlage um die in dieser Arbeit bestimmten Eigenschaften des AR-Lokalisierungssystems zu vergleichen.

3.1.1. Einsatzszenario und Auswahlkriterien

Es existiert eine Vielzahl von Verfahren zur Positionsbestimmung in Innenräumen, basierend auf unterschiedlichen physikalischen Prinzipien. So werden allein im Übersichtswerk von Samama [58] 40 verschiedene Methoden aufgelistet. Ein Vergleich dieser ist komplex, da je nach Anwendungsfall unterschiedliche technische Anforderungen berücksichtigt werden müssen. Für diese Arbeit wird von folgendem Szenario ausgegangen:

Ein Mensch und ein Roboter führen kollaborativ die Montage eines Getriebes durch. Der Prozess findet in einem fest definierten Raum statt, dessen Fläche ungefähr 50 m^2 beträgt. Die Aufgabe des Arbeiters besteht darin, einzelne Getriebekomponenten von unterschiedlichen

Lagerplätzen zu holen und gemeinsam mit einem Roboter den Zusammenbau durchzuführen. Nach der vollständigen Montage soll das Getriebe an einen weiteren Lagerplatz gebracht werden. Die Position des Arbeiters im Raum soll hierbei kontinuierlich bestimmt werden. Mit den erfassten Daten soll es möglich sein, in Verbindung mit einer geometrischen Prozessbeschreibung die Zeiten einzelner menschlicher Prozessschritte sowie Übergangszeiten zwischen Einzelschritten zu berechnen. Abb. 3.1 zeigt die schematische Draufsicht der Arbeitsumgebung.

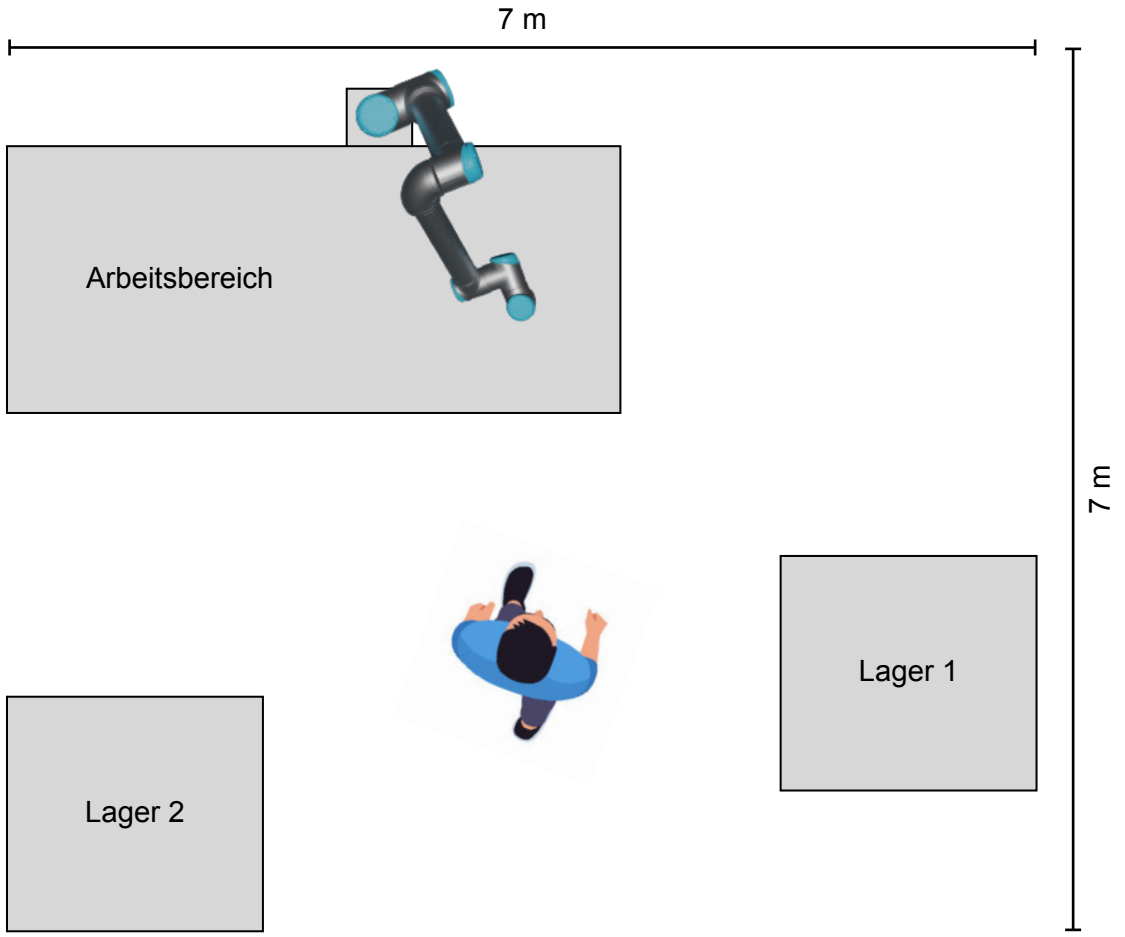


Abbildung 3.1.: Schematische Draufsicht der Arbeitsumgebung (nicht maßstabsgetreu)

Im Folgenden werden die Mindestanforderungen an ein in diesem Szenario verwendbares Positionierungssystem beschrieben:

Präzision

Da der Bewegungsradius des Arbeiters für das Holen und Ablegen von Komponenten begrenzt ist und in der Regel nur wenige Meter beträgt, ist eine gewisse Präzision der Positionierung erforderlich. Nur so lassen sich Einzelbewegungen in den Daten erkennen und eine hinreichende Präzision bei der Bestimmung der Prozesszeiten erreichen. Aus diesem Grund kommen nur Systeme infrage, die eine Positionsgenauigkeit von < 50 cm

erreichen. Generell lässt sich festhalten, dass eine höhere Genauigkeit der Positionierung auch genauere Berechnungen der Prozesszeiten ermöglicht, in der Regel aber zu einem komplexeren Positionierungssystem führt.

Reichweite

Um die Position des Arbeiters auf der gesamten Fläche bestimmen zu können, muss das Positionierungssystem eine gewisse Reichweite erreichen. Um sicher zu stellen, dass mit möglicht wenigen technischen Komponenten die gesamte Fläche abgedeckt werden kann, wird eine Mindestreichweite des Verfahrens von 5 m gefordert.

Passivität

Um den Arbeiter nicht von seiner Produktionsaufgabe abzulenken, wird gefordert, dass die Positionsbestimmung ohne aktive Mitwirkung des Menschen erfolgt. Unberücksichtigt sollen hierbei Kalibrierungsvorgänge bleiben, welche einmalig oder zyklisch durch den Arbeiter durchgeführt werden müssen.

Frequenz

Die Häufigkeit, mit der die Position bestimmt wird, entscheidet letztlich über die Genauigkeit der Prozesszeitberechnung. Da im beschriebenen Szenario davon auszugehen ist, dass sich viele Prozessschritte in einem Zeitrahmen unter 10 Sekunden bewegen, ist eine Messfrequenz von mindestens 1/s notwendig.

Etabliertheit

Um den Vergleich von Parametern zu ermöglichen sollen nur Verfahren berücksichtigt werden, für die am Markt bereits Systeme verfügbar sind. Alle technischen Kennzahlen beziehen sich ebenfalls auf etablierte Produkte im industriellen Einsatz. Verfahren, die bislang nur in der Forschung unter Laborbedingungen erprobt wurden, bleiben unberücksichtigt.

3.1.2. Ausgewählte Verfahren

Unter Berücksichtigung der definierten Kriterien kommen insgesamt vier unterschiedliche Positionierungsverfahren infrage. Im Folgenden werden die technischen Prinzipien dieser Verfahren erklärt und Produktbeispiele mit deren Kernparametern vorgestellt.

Kamerabasierte Systeme (Motion Capture)

Nimmt man mit einer Kamera Bilder des gleichen Objekts aus unterschiedlichen Perspektiven auf, so können aus diesen Bildern dreidimensionale Objekteigenschaften berechnet

werden. Dieses Prinzip nutzen auch kamerabasierte Positionierungssysteme. Der grundlegende Systemaufbau ist in Abb. 3.2 dargestellt. Um den zu trackenden Menschen sind Kameras in erhöhter Position fest angebracht. Da mit diesen Systemen viele Punkte im Raum gleichzeitig erfasst werden können, werden diese oft für das Tracking von menschlichen Bewegungen eingesetzt und dann als Motion Capture Systeme bezeichnet. Anwendung finden diese beispielsweise in der Film- und Computerspielbranche, um menschliche Bewegungen zu digitalisieren und diese dann auf virtuelle Charaktere zu übertragen [59]. Weitere Anwendungsfälle finden sich in Medizin und Sportwissenschaft bei der Analyse von Bewegungsabläufen [60, 61]. Generell unterscheidet man Systeme mit und ohne Marker. Diese Marker bestehen in der Regel aus einem reflektierenden Material und werden an Gegenständen oder Körperteilen angebracht. Sie ermöglichen dem System die zuverlässige Erkennung und Positionierung der Objekte mit den aufgenommenen Kamerabildern. Die Verwendung von Markern ermöglicht eine hohe Präzision der Positionsbestimmung, sorgt aber für einen erheblichen Aufwand für das Anbringen und die Kalibrierung. Die Reichweite eines solchen Systems ist bestimmt durch die Kameraauflösung. Durch den Einsatz einer größeren Anzahl an Kameras kann die überwachte Fläche jedoch fast beliebig vergrößert werden. Die Messfrequenz wird bestimmt durch die Belichtungsdauer des Kamerasensors. Die Firma OptiTrack erreicht mit ihrer leistungsfähigsten Motion Capture-Kamera eine Reichweite von 30-45 Metern und eine Bildrate von 180 Hz. Die Präzision wird vom Hersteller mit 0,1 mm angegeben [62]. Markerbasierte Motion Capture-Systeme werden in der Regel als Gesamtsysteme bestehend aus Kameras, Markern, Kalibrierungsequipment und Software angeboten. Einfache Systeme sind ab etwa 10 000 € erhältlich [63], während professionelle, für den großflächigen Einsatz vorgesehene Systeme bis über 1 Mio. € kosten können [59].

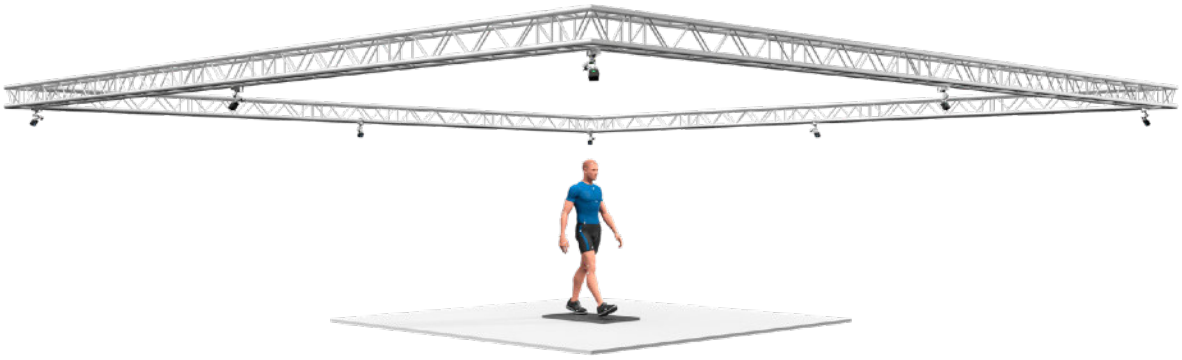


Abbildung 3.2.: Aufbau eines Motion Capture-Systems [63]

Deutlich preisgünstiger sind Systeme die auf den Einsatz von Markern verzichten. Diese arbeiten mit Bilddaten handelsüblicher Videokameras und berechnen die Position eines Menschen oder Gegenstandes mithilfe von Verfahren zur maschinellen Bilderkennung. Insbesondere die gestiegene Rechenleistung von Computern und Fortschritte im Bereich des maschinellen Lernens (engl. machine learning, ML) ermöglichten in den letzten Jahren eine stetige Weiterentwicklung der markerlosen Positionsbestimmung. Die Präzision von markerlosen Systemen reicht nicht an die von markerbasierten heran. In einer Versuchsreihe

japanischer Wissenschaftler mit der Open-Source-Software OpenPose wurde eine Positionierungsgenauigkeit von 3 cm ermittelt [64]. Am Markt verfügbar sind auch kommerzielle Softwarelösungen für die markerlose kamerabasierte Positionsbestimmung, beispielsweise von der deutschen Firma Captury [65]. Der Vorteil dieser Systeme besteht darin, dass die aufwendige Anbringung und Kalibrierung von Markern entfällt und diese somit flexibler und schneller einsetzbar sind.

Laser Tracker

Die größtmögliche Präzision bei der Bestimmung von Objektpositionen bieten Laser Tracker. Eine ortsfeste, aber dreh- und neigbare Basisstation sendet dauerhaft Laserimpulse aus, die von speziellen Reflektoren zurückgeworfen werden. Die Laufzeit des Lichts wird gemessen. Zusammen mit Dreh- und Neigungswinkel kann dann die Reflektorposition im Raum bestimmt werden. Wird der Reflektor bewegt, so kann dieser durch die kontinuierliche, inkrementelle Anpassung der Winkel verfolgt werden. Ein Beispiel für ein industriell eingesetztes System ist der Leica Absolute Tracker der Firma Hexagon. Dieses System erreicht eine Reichweite bis 80 m bei einer Präzision von bis zu 0,1 mm [66]. Basisstation und Reflektoren sind in Abb. 3.3 dargestellt. Die Anschaffungskosten für Laser Tracking-Systeme liegen in der Regel im Bereich von mehr als 20 000 € [66].

Abbildung 3.3.: Laser Tracking-System der Firma Hexagon - links: Basisstation, rechts: verschiedene Reflektoren [67, 68]

Ultra Wide Band (UWB)

Für die Ortung in Innenräumen kann die Ultrabreitband-Funktechnologie eingesetzt werden. Hierfür notwendig sind mehrere Basisstationen, die Signale im Frequenzbereich von 3 GHz bis 10 GHz zeitlich synchron aussenden. Ein am zu lokalisierenden Objekt befestigtes Gerät misst den Zeitpunkt, an dem die unterschiedlichen Signale eintreffen, wodurch die Position

des Objektes bestimmt werden kann [58]. Sind keine Hindernisse im Weg, können UWB-Signale eine Reichweite von bis zu 100 m erreichen und ermöglichen so eine großflächige Positionsbestimmung [69]. Für eine möglichst präzise UWB-Lokalisierung ist es wichtig, dass die ausgesendeten Signale möglichst zeitgleich gesendet werden. Des Weiteren wird die Positionierungsgenauigkeit reduziert, wenn die Ausbreitung der Signale von Hindernissen gestört wird. Theoretisch sind unter optimalen Bedingungen Fehlerwerte im Submillimeter-Bereich möglich [70]. Bei der Anwendung im industriellen Kontext ist die Lokalisierung mit UWB-Signalen jedoch weitaus ungenauer. Der Hersteller Sewio beispielsweise gibt für sein UWB-Positionierungssystem einen Fehlerwert von 30 cm an [71].

Die UWB-basierte Lokalisierung bietet insbesondere dort Vorteile, wo eine große Anzahl an Objekten getrackt werden soll, da eine einmal vorhandene Sender-Infrastruktur durch die Verwendung zusätzlicher, meist recht günstiger UWB-Empfänger erweitert werden kann. Ein zusätzlicher Vorteil ist die Möglichkeit, UWB-Ortung auch dann durchführen zu können, wenn kein direkter Sichtkontakt zwischen Sender und Empfänger besteht, wie dies beispielsweise in Lagern oftmals der Fall ist. Ein Nachteil des Systems ist die Notwendigkeit, die UWB-Empfänger mit elektrischer Energie zu versorgen, weshalb Akkus oder Batterien eingesetzt werden, welche dann regelmäßig geladen bzw. getauscht werden müssen [58].

Während Motion Capture-Systeme oder Laser-Tracking-Verfahren schon seit vielen Jahren vielfältig angewendet werden, ist die UWB-Lokalisierung erst seit einigen Jahren in den Fokus der Industrie gerückt. Basierend auf der UWB-Technologie entwickelt ein Zusammenschluss von Firmen den sogenannten Omlox-Standard, welcher die Integration verschiedener Lokalisierungstechnologien in industrielle Anwendungen sowie die gemeinsame Verwendung von Systemen unterschiedlicher Hersteller erleichtern soll [72]. Große Aufmerksamkeit für die UWB-Technologie erzeugte der Technologiekonzern Apple, welcher ankündigte, UWB-Module in alle kommenden Smartphone-Modelle zu verbauen. Zusätzlich wurde mit dem AirTag ein kleiner, batteriebetriebener UWB-Empfänger vorgestellt, welcher an analogen Gegenständen angebracht werden kann. Ziel ist es, Geräte und Gegenstände über kurze Distanzen präzise lokalisieren zu können. Die voranschreitende Verbreitung von UWB-fähigen Geräten könnte dazu führen, dass deren Einsatz sich auch im industriellen Kontext stärker als bisher verbreitet [73].

Alternative Positionierungsverfahren, die auf der Laufzeitmessung von Funkwellen basieren, nutzen unter anderem die WLAN oder Bluetooth-Technologie, sind aber deutlich ungenauer und werden deshalb nicht in den Vergleich einbezogen [58].

3.1.3. Vergleich der Eigenschaften

Aufgrund der vielfältigen Einsatzmöglichkeiten für Positionsbestimmungssysteme ist es äußerst komplex, diese miteinander zu vergleichen. Für diese Arbeit werden vier Vergleichdimensionen ausgewählt, welche schließlich einen Vergleich mit dem entwickelten

AR-Lokalisierungssystem ermöglichen sollen. Ausgewählt werden die erreichbare absolute Genauigkeit der Positionsbestimmung, der Umfang der notwendigen ortsfesten und beweglichen Infrastruktur sowie die Anschaffungskosten. Da insbesondere die Präzision und die Kosten ja nach betrachtetem Produkt stark schwanken wird versucht hier typische Werte für das jeweilie Verfahren anzugeben. Tabelle 3.1 zeigt eine Übersicht der ausgewählten Kenngrößen für die zuvor beschriebenen Positionierungssysteme.

Tabelle 3.1.: Übersicht ausgewählter Parameter der vorgestellten Systeme

Verfahren	Präzision	Ortsfeste Infrastruktur	Bewegliche Infrastruktur	Anschaffungs-kosten
Motion Capture mit Markern	< 1 mm	mehrere Kameras	passive Marker	> 10 000 €
Motion Capture ohne Marker	< 10 cm	mehrere Kameras	-	> 1000 €
Laser Tracker	< 1 mm	eine Basisstation	passive Reflektoren	> 20 000 €
UWB	< 50 cm	mehrere Sender	aktive Empfänger	> 5000 €

Die Erwartungen an das im Rahmen dieser Arbeit entwickelte Positionsbestimmungssystems auf Grundlage der HoloLens 2 und die erwartete Einordnung in Relation zu den genannten IPS werden in Abschnitt 3.2.4 formuliert.

3.2. Positionsbestimmung mit der HoloLens

Alle vorgestellten Systeme zur Positionsbestimmung arbeiten nach dem Outside-In-Prinzip, besitzen also ortsfeste Sensoren. AR-Systeme hingegen nutzen in der Regel die Inside-Out-Konfiguration, alle Sensoren sind also im AR-System selbst verbaut und die Lokalisierung des Systems geschieht auf Grundlage der sonsorisch erfassten Umgebung. Bezeichnet wird dieser Vorgang als SLAM, ein Akronym für Simultaneous Localization and Mapping (dt.: Simultane Lokalisierung und Kartierung). Wie der Verfahrensname bereits andeutet wird durch die Analyse der Sensordaten eine Karte der Umgebung erstellt und gleichzeitig die Systemposition in Relation zu dieser Karte bestimmt [37].

Während das Grundprinzip der SLAM-Lokalisierung deutlich komplexer ist als das der bereits vorgestellten Systeme bietet es jedoch einige Vorteile gegenüber diesen. So können alle technischen Komponenten an einem Ort gebündelt werden und auf ortsfeste Infrastruktur verzichtet werden, was den flexiblen Einsatz der Systeme ermöglicht. Neben der Positionsbestimmung kann durch die dauerhafte Erfassung der Umgebung auch auf Veränderungen dieser reagiert werden. So können beispielsweise autonome Roboter und Transportsysteme durch die Verwendung des SLAM-Verfahrens auf Hindernisse reagieren, ihnen ausweichen

und die Information für zukünftige Navigationsaufgaben einplanen [74].

Auch die für diese Arbeit verwendete HoloLens nutzt das SLAM-Verfahren um die Systemposition im Raum kontinuierlich zu bestimmen und mit den berechneten Informationen eine korrekte visuelle Darstellung der virtuellen Objekte für den Anwender zu erreichen. Microsoft bezeichnet diesen Vorgang als Spatial Mapping [75]. Die für das Spatial Mapping erstellt Karte wird von der HoloLens in ein Polygonnetz umgewandelt, welches in Abb. 3.4 dargestellt wird.

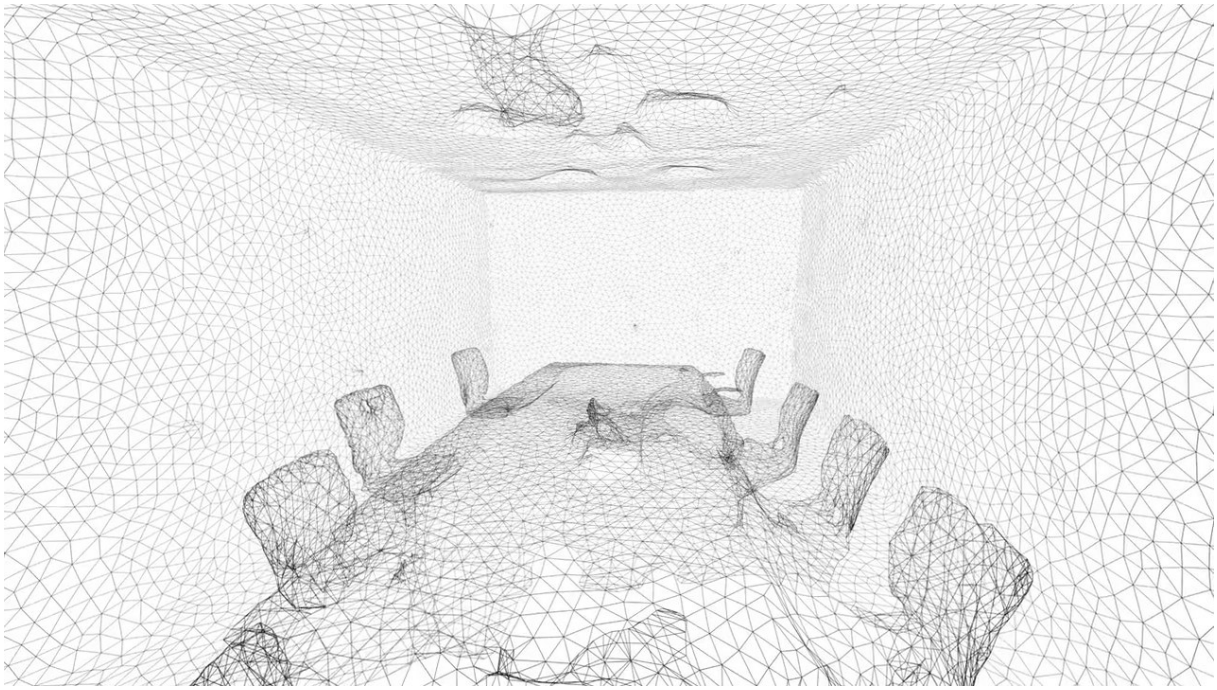


Abbildung 3.4.: Visualisierung des durch die HoloLens erstellten Polygonnetzes zur Umgebungskartierung [52]

Die Umgebungskartierung wird von der HoloLens dauerhaft im Hintergrund ausgeführt und kann vom Entwickler in aggregierter Form für die Entwicklung von AR-Anwendungen genutzt werden. So reicht es aus, Positionen oder Bewegungen von virtuellen Objekten im Koordinatensystem der HoloLens anzugeben. Die HoloLens-Anwendung sorgt dafür, dass die dem Nutzer als Hologramme eingeblendeten Objekte auch bei einer Bewegung des AR-Systems an der definierten Stelle verbleiben. Für spezielle Anwendungsfälle bietet der Hersteller jedoch auch die Möglichkeit an, direkt auf die Rohdaten des Kartierungsvorgangs zuzugreifen. Für diese Arbeit wird dies genutzt um die Position der HoloLens innerhalb der erstellten Umgebungskarte zu extrahieren.

Die Herausforderung bei der Positionsbestimmung mit der HoloLens ist, dass bei dem Start einer AR-Anwendung das HoloLens-Koordinatensystem stets neu ausgerichtet wird. Koordinatenursprung und die Orientierung der Koordinatenachsen orientieren sich an Position und Ausrichtung des Geräts zu diesem Zeitpunkt. Für visuelle AR-Anwendungen bei denen der Anwender frei mit virtuellen Objekten interagiert oder diese selbst im Raum

anordnet, stellt dies kein Problem dar, da die Verwendung von relativen Koordinatenwerten hierfür ausreicht. Für ein Positionsbestimmungssystem im Kontext von Industrieanlagen ist es jedoch erforderlich, Daten in absoluten, am Raum orientierten Koordinaten zu erhalten. Um dies zu erreichen ist eine initiale Kalibrierung des AR-Systems notwendig, sodass die Verschiebung zwischen dem jeweils neu definierten HoloLens-Koordinatensystem und dem raumfesten Koordinatensystem bekannt ist. Für den Vorgang, virtuelle Objekte an reale Objekte anzugleichen wird unter anderem der Begriff „Registrierung“ verwendet. Für die Angleichung des virtuellen an das raumfeste Koordinatensystem wird im weiteren Verlauf der Arbeit jedoch der Begriff „Kalibrierung“ verwendet.

3.2.1. Fehlerquellen bei der Positionsbestimmung

Die Präzision mit der die HoloLens ihre absolute Position im Raum bestimmen kann ist abhängig von zwei Faktoren, welche zusätzlich in Abb. 3.5 schematisch veranschaulicht sind:

1. Der Fähigkeit, die Positionsveränderung ausgehend von einer initialen Position zu verfolgen. Hierbei auftretende Abweichungen werden als Drift bezeichnet und können sich mit der Zeit verändern.
2. Die initiale Kalibrierung des Systems im Raumkoordinatensystem. Fehler bei der initialen Positionierung oder Ausrichtung wirken sich dauerhaft auf die Präzision der Positionsbestimmung aus.

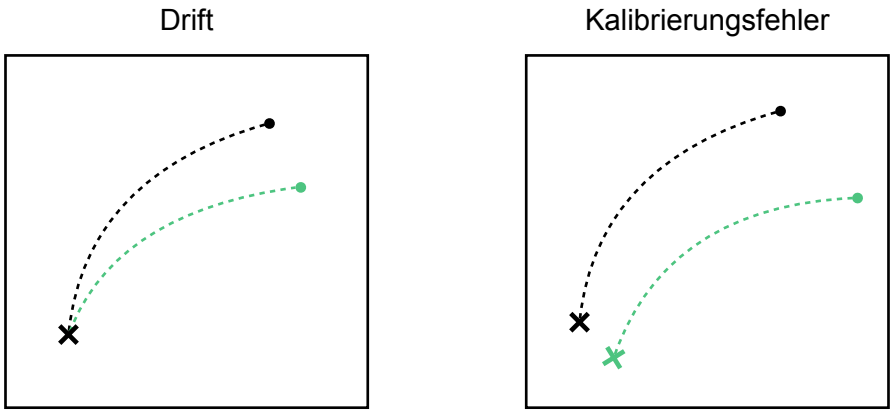


Abbildung 3.5.: Zwei maßgebliche Fehlerquellen für die Positionsbestimmung mit einem SLAM-System

In den folgenden Abschnitten wird anhand von wissenschaftlichen Referenzen dargestellt, welche Präzision sowohl in Bezug auf den Kalibrierungsvorgang als auch für das Positionstracking zu erwarten ist. Aufgrund der üblichen Verzögerung bei der Veröffentlichung wissenschaftlicher Arbeiten beziehen sich die meisten Werte auf den Vorgänger des für diese

Arbeit verwendeten Systems, die HoloLens 1. Da beide Systeme die gleichen technischen und sensorischen Grundprinzipien teilen, dürften die Veröffentlichungen trotzdem ausreichen, um eine grundlegende Einschätzung über das Potenzial der HoloLens 2 abgeben zu können.

3.2.2. Positionsfehler durch Drift

Die Trackinggenauigkeit des für die HoloLens verwendeten SLAM-Verfahrens lässt sich messen indem vom AR-System bestimmte Positionsdaten mit extern z.B. durch ein Motion-Capture-System ermittelte Daten verglichen werden. Die verfügbaren Referenzen messen hier lediglich die relativen Positionsveränderungen und verzichten auf eine initiale Kalibrierung. Der ermittelte Trackingfehler wird für die HoloLens 1 in [76] mit 40 mm und in [77] mit 20 mm angegeben. Für die HoloLens 2 wurde in [78] eine mittlere Abweichung von 30 mm gemessen. Die drei genannten Versuchsreihen beziehen sich jedoch auf kleine Versuchsflächen mit Abmessungen von wenigen Metern. Dem gegenüber steht die in [79] gemessene Abweichungen von bis zu 200 mm bei einer Größe der Versuchsfläche von 30 m x 50 m. Dies deutet darauf hin, dass bei einer Vergrößerung des Anwendungsbereichs auch der Positionierungsfehler durch Drift größer wird.

3.2.3. Räumliche Kalibrierung der HoloLens

Ziel der initialen Kalibrierung ist es, einmalig die exakte Position des AR-Systems relativ zu einem raumfesten Koordinatensystem zu erfassen, sodass gemessene Positionsveränderungen wieder zu raumfesten Koordinaten umgerechnet werden können. Verfahren für die exakte Lokalisierung werden im Folgenden beschrieben.

Die wohl präziseste Möglichkeit zur Kalibrierung der initialen Raumposition mit einem AR-System ist die Fixierung des Systems in einer Lage mit bekannten Raumkoordinaten während des Kalibrierungsvorgangs. Maßgeblich für die Präzision ist die Exaktheit mit der das System an diesem Punkt fixiert werden kann. Ein zusätzlicher Faktor hierbei ist die erreichbare Genauigkeit des zur Vermessung dieses Raumpunktes verwendeten Messsystems, welcher jedoch bei allen hier vorgestellten Kalibrierungsverfahren relevant ist. Während die Kalibrierungsgenauigkeit sehr hoch ist, hat dieses Verfahren jedoch auch Nachteile. So ist es bei AR-Headsets notwendig das Gerät abzusetzen. Da für den Anwender die AR-Nutzeroberfläche nun nicht mehr zur Verfügung steht muss der Kalibrierungsvorgang anderweitig gestartet werden, wofür zusätzliche Infrastuktur notwendig ist. Eine zusätzliche Herausforderung ist das fixieren des AR-Headsets in einer festen Position. Hierfür ist es notwendig eine passgenaue Aufnahmeform zu entwerfen und zu produzieren.

Deutlich effizienter ist der Kalibrierungsvorgang, wenn dieser durchgeführt werden kann

während der Nutzer das AR-Headset trägt. Hierbei lassen sich eine manuelle und eine automatisierte Variante unterscheiden. Bei der manuellen Kalibrierung sorgt der Anwender selbst für die korrekte Überlagerung von virtueller und realer Welt. Oftmals geschieht dies indem ein virtuelles Modell eines im Raum befindlichen Objekts z. B. eines Würfels über das AR-Display eingeblendet wird. Durch Gesten verschiebt der Nutzer das virtuelle Objekt so lange, bis dieses das reale Objekt vollständig überlagert. Anschließend kann die Transformationsvorschrift zwischen virtuellem und raumfestem Koordinatensystem bestimmt werden [80]. Während der hohe manuelle Aufwand ein Nachteil dieses Verfahrens ist, kann durch die Berücksichtigung des Blickpunkts des Nutzers eine präzise optische Überlagerung erreicht werden, was für viele AR-Anwendungen von Vorteil ist. Ein Review von Andrews u. a. [81] listet mehrere wissenschaftliche Veröffentlichungen auf, die den absoluten Fehler bei der Kalibrierung durch manuelle Überlagerung bestimmt haben. Bei einem Großteil der Versuchsreihen lag der Fehler im Bereich von etwa 5 mm.

Da für die im Rahmen dieser Arbeit entwickelte Anwendung nicht die optische Überlagerung sondern die absolute Lokalisierungsgenauigkeit entscheidend ist, wird auf ein Kalibrierungsverfahren zurückgegriffen, welches die initiale Systemposition durch die sensorische Erfassung markanter Raumpunkte mit bekannter Position bestimmen kann. Dies kann beispielsweise duch die automatisierte Erkennung einer bekannten Geometrie erfolgen [82]. Weitaus einfacher umsetzen lässt sich jedoch die Kalibrierung mit einem optischen Marker, dessen Raumposition bestimmt wird. Diese Methode wird auch für den im Rahmen dieser Arbeit implementierten Kalibrierungsvorgang verwendet. Die Präzision des markerbasierten Registrierungsverfahrens ist abhängig von der Größe des Anwendungsbereichs, also von der Entfernung zwischen HoloLens und Marker. Die in [81] aufgelisteten Messungen entstammen aus dem medizinischen Bereich und berücksichtigen nur einen kleinen Anwendungsbereich. Es ergibt sich eine Kalibrierungsgenauigkeit von 2 mm bis 4 mm. In [83] werden Versuche mit einem ganzen Raum als Anwendungsbereich durchgeführt für welche eine Präzision von durchschnittlich 20 mm mit einem Referenzmesssystem gemessen wurde.

3.2.4. Erwartungen an die zu entwickelnde Positionsbestimmungs-Anwendung

Auf Grundlage der genannten wissenschaftlichen Arbeiten, die sich mit der Positionierungsgenauigkeit der HoloLens auseinandersetzen und den definierten Rahmenbedingungen lässt sich eine Erwartung an die entwickelte AR-Anwendung ableiten. Da sowohl Fehler bei der Kalibrierung als auch bei der Positionsverfolgung durch das SLAM-Verfahren auftreten und sich addieren können wird von einer erreichbaren Präzision im Bereich von 100 mm ausgegangen. Abb. 3.6 zeigt eine qualitative Einordnung der Erwartung in Realation zu den vorgestellten IPS-Technologien.

Bezogen auf die Präzision der Lokalisierung bewegt sich die HoloLens zwischen der kamerabasierten Positionsbestimmung ohne Markern und der UWB-Lokalisierung. Während sich die Kosten dieser drei Systeme in einem ähnlichen Bereich bewegen, ist die Komplexität

für UWB- und kamerabasierte Systeme deutlich höher, da jeweils ortsfeste und kalibrierte Infrastruktur notwendig ist. Für das Positionierungsverfahren mit der HoloLens ist lediglich die einmalige Anbringung und Vermessung von gedruckten Markern notwendig, während das AR-System an sich nicht an einen Einsatzort gebunden ist. Mit laserbasierten Messsystemen oder kamerabasierten Systemen mit Markern lassen sich deutlich präzisere Positionsdaten erfassen. Die Kosten für die Anschaffung solcher Systeme ist jedoch deutlich höher, ebenso wie die Komplexität der Installation und der Kalibrierung aufgrund der notwendigen am Körper befestigten Marker bzw. Reflektoren.

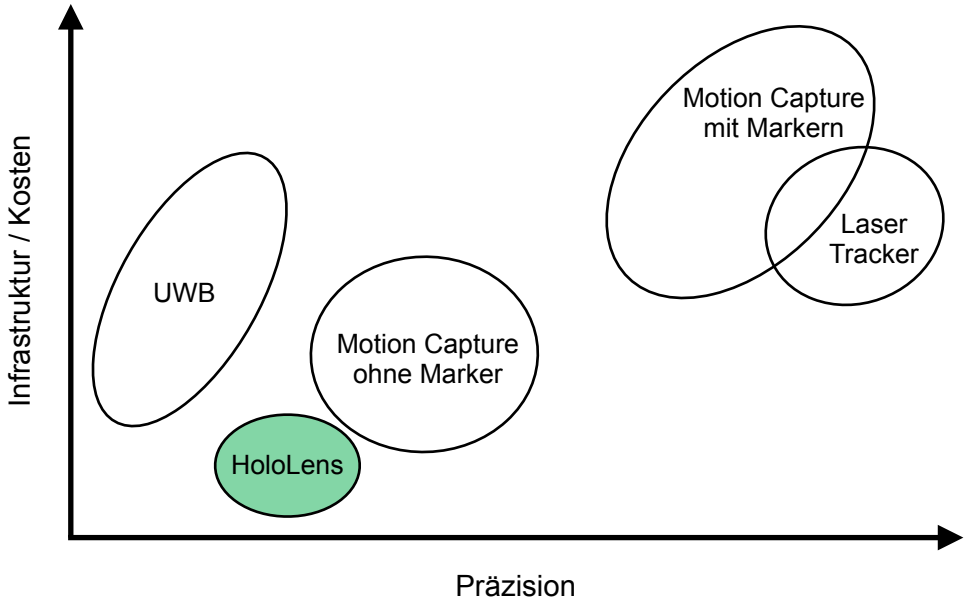


Abbildung 3.6.: Erwartete Einordnung des neuen Ansatzes in Bezug auf die erwartete Präzision, Infrastrukturanforderungen und Anschaffungskosten. Annahme: Eine getrackte Person auf einer Fläche von 50 m^2

3.3. Bestimmung von Prozesszeiten aus Positionsdaten

Wie bereits in Abschnitt 2.3.2 beschrieben existiert eine Vielzahl an Anwendungsbeispielen für die Verwendung von menschlichen Positionsdaten zur Optimierung von Industrieprozessen. Diese Arbeit beschränkt sich auf die Bestimmung von Prozesszeiten aus Arbeiterpositionsdaten. In der wissenschaftlichen Literatur findet sich nur eine Arbeit, die sich eingehend mit diesem Thema beschäftigt, nämlich die Arbeit der italienischen Forscher Faccio u. a., die den sogenannten „Human Factor Analyser“ entwickelten [19]. Grundlage bildet hier ein markerloses, kamerabasiertes Motion Capture System mit dem die Bewegungen verschiedener Körperteile eines Montagearbeiters erfasst werden. Um die Bewegungen zu analysieren werden Kontrollvolumen definiert. Ein Kontrollvolumen umschließt dabei beispielsweise einen Regal-Lagerplatz für ein bestimmtes Teil. Bei der Analyse werden diese Kontrollvolumen dann mit den erfassten Positionsdaten abgeglichen und auf diese Art und Weise errechnet, wann ein bestimmtes Körperteil in ein Kontrollvolumen eingetreten ist und

wann es dieses verlassen hat. So kann beispielsweise die Dauer eines Griffs in eine Kiste mit Komponenten gemessen oder die Anzahl der durchgeführten Aktionen bestimmt werden.

4 Systementwicklung

Im Folgenden wird die Entwicklung des im Rahmen dieser Arbeit entworfenen Systems zur Erfassung von Arbeiterpositionsdaten und der automatisierten Auswertung der Daten beschrieben. Nach einer Systemeinteilung folgt die Beschreibung des grundlegenden Systemaufbaus der einzelnen Komponenten sowie die Entwicklung der notwendigen Konzepte und Algorithmen.

4.1. Systemeinteilung

Ziel der Arbeit ist die Entwicklung und Erprobung einer AR-Anwendung, die die Position eines Arbeiters in einer Fertigungsanlage dauerhaft bestimmt. Zur Veranschaulichung soll zudem eine Anwendung entwickelt werden, um aus den gewonnenen Positionsdaten Prozesszeiten für einzelne manuelle Bearbeitungsschritte zu berechnen, welche dann zur Optimierung der Prozessreihenfolge genutzt werden können. Das entwickelte System wird in zwei Teilsysteme unterteilt, welche weitgehend unabhängig voneinander agieren können. Entsprechend wird auch die Beschreibung der Systementwicklung getrennt beschrieben. Die beiden Teilsysteme sind:

- Eine HoloLens-Anwendung zur kontinuierlichen Bestimmung der Position eines Arbeiters.
- Eine Serveranwendung zur Berechnung von Prozesszeiten aus den Positionsdaten und zur Durchführung der Prozessoptimierung.

Im Prinzip wäre es problemlos möglich, die Analyse-Anwendung direkt auf der HoloLens auszuführen. Um eine größere Modularität und Erweiterbarkeit der Anwendung zu erreichen und die Anbindung weiterer Systeme wie z. B. von Robotern zu ermöglichen, wird jedoch eine dedizierte Serveranwendung entwickelt. Vordergründig erhöht sich dadurch der Entwicklungsaufwand deutlich, da Kommunikationsschnittstellen geschaffen werden müssen, um beide Systeme zu verbinden. Die Abhängigkeit der Systeme voneinander sinkt durch die Aufteilung jedoch stark, sodass es zukünftig möglich sein wird, die Teilanwendungen getrennt voneinander weiter zu entwickeln oder neue Verfahren zu implementieren.

Abb. 4.1 zeigt eine schematische Übersicht des Gesamtsystems. Zusätzlich wird angegeben, in welchem Abschnitt die Entwicklung des jeweiligen Systembestandteils beschrieben wird.

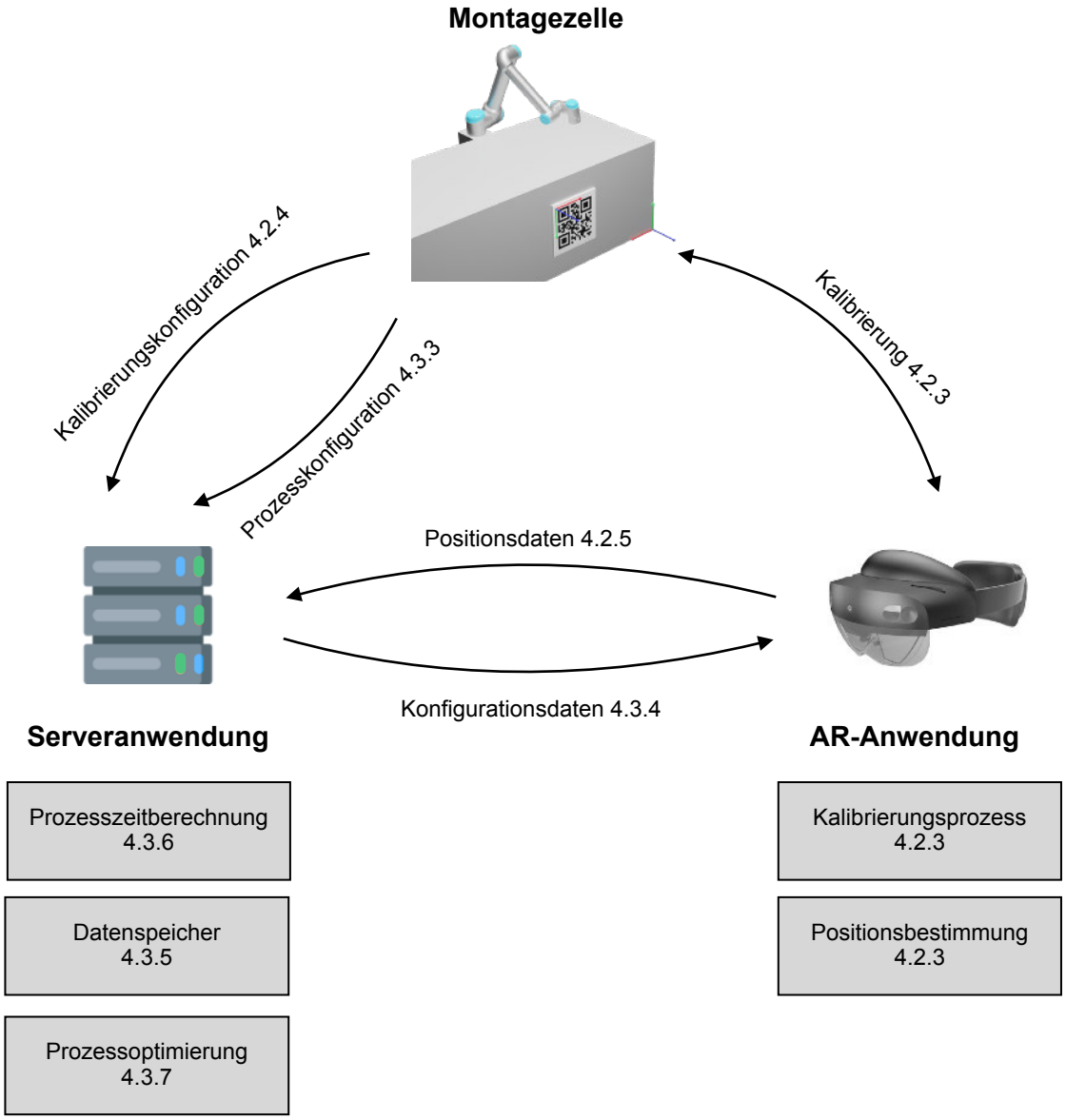


Abbildung 4.1.: Visualisierung der Gesamtsystemarchitektur mit Verweisen auf zugehörige Kapitel

4.2. HoloLens-Anwendung für Positionsbestimmung

Im Folgenden wird die Entwicklung der AR-Anwendung für die Microsoft HoloLens 2 beschrieben. Zunächst werden allgemeine Anforderungen festgelegt und der grundlegende Systemaufbau definiert. Anschließend folgt die Beschreibung des detaillierten Entwurfs der Systemprozesse und Schnittstellen.

4.2.1. Anforderungen an die AR-Anwendung

Die Hauptfunktion der zu entwickelnden AR-Anwendung ist die Bestimmung der Position des Arbeiters relativ zum Produktionssystem. Da es sich bei der für diese Arbeit verwendeten

HoloLens um ein AR-Headset handelt, ist die mit dem Gerät ermittelte Position gleichbedeutend mit der Kopfposition des Trägers. Wenn im weiteren Verlauf der Arbeit von Position gesprochen wird, so ist stets die Kopfposition gemeint.

Ziel eines Lokalisierungssystems ist stets die Ausgabe von Koordinaten in einem vorher festgelegten Bezugskoordinatensystem. Ausreichend für den in dieser Arbeit beschriebenen Anwendungsfall ist die Bestimmung von zwei Koordinatenwerten (X und Y). Um künftig weitere Anwendungsfälle betrachten zu können, wird jedoch entschieden, auch die Z-Koordinaten des Kopfes zu berücksichtigen und somit die Kopfposition in drei Dimensionen zu bestimmen. Um die Positionsbestimmung in raumfesten Koordinaten zu ermöglichen, ist eine Systemkalibrierung erforderlich (siehe Abschnitt 3.2.3). Damit diese so einfach und schnell wie möglich durchführbar ist, wird ein automatisiertes Kalibrierungsverfahren auf Basis eines optischen Markers entwickelt. Zuletzt müssen die bestimmten Positionsdaten an die ebenfalls entwickelte Serveranwendung übertragen werden, weshalb eine geeignete Datenschnittstelle in die Anwendung implementiert werden muss.

4.2.2. Systemaufbau der AR-Anwendung

Den Kern der AR-Anwendung bildet das Subsystem für die Positionsbestimmung, welches auch den notwendigen Kalibrierungsvorgang beinhaltet. Für die Erfassung der Position werden Daten aus zwei Quellen benötigt: Die von der HoloLens bereitgestellten Daten des SLAM-Prozesses und der verbauten Sensoren sowie die vom Nutzer generierten Konfigurationsdaten für den Kalibrierungsprozess.

Da die Analyse der Daten mit der ebenfalls entwickelten Serveranwendung erfolgen soll, muss eine Schnittstelle geschaffen werden, um die Positionsdaten an diese zu übertragen. Nebenbei kann die Schnittstelle dazu genutzt werden, um auch die für die Kalibrierung notwendigen geometrischen Konfigurationsdaten zu empfangen, was vor allem Vorteile bei einer späteren Skalierung der Anwendung mit sich bringt. Konfigurationsdaten können so an einem zentralen Ort gespeichert und verändert werden und von beliebig vielen AR-Systeme abgefragt werden.

Im Gegensatz zu vielen industriellen Anwendungsfällen für AR-Systeme, welche maßgeblich auf der Visualisierung und Eingabe von Daten und Informationen beruhen (siehe Abschnitt 2.4.3), ist für die entwickelte Anwendung zur Positionsbestimmung die Implementierung einer Benutzeroberfläche im Prinzip nicht notwendig. Dennoch sollen auch für die im Rahmen dieser Arbeit entwickelte Anwendung die Vorteile einer virtuellen Benutzeroberfläche genutzt werden, um eine effiziente und nutzerfreundliche Erprobung der AR-Anwendung möglich zu machen. So können beispielsweise über Nutzereingaben der Kalibrierungsprozess manuell gestartet werden oder dem Anwender durch virtuelle Einblendungen Informationen und Rückmeldungen zum Analyseprozess oder zur optimierten Prozessreihenfolge gegeben werden. Da die Gestaltung der Nutzeroberfläche keinen Ein-

fluss auf die Kernfunktionalität der Anwendung hat, wird auf eine ausführliche Beschreibung der Entwicklung und Implementierung verzichtet.

Der beschriebene Systemaufbau ist in Abb. 4.2 schematisch dargestellt. Pfeile symbolisieren den Fluss von Daten. Die im Rahmen der Systementwicklung nur am Rande betrachteten Komponenten und Datenflüsse sind mit gestrichelten Linien gekennzeichnet. Die Entwicklung der drei beschriebenen Komponenten, Positionsbestimmung inkl. Kalibrierung, Kalibrierungskonfiguration und Kommunikationsschnittstelle wird im Folgenden beschrieben.

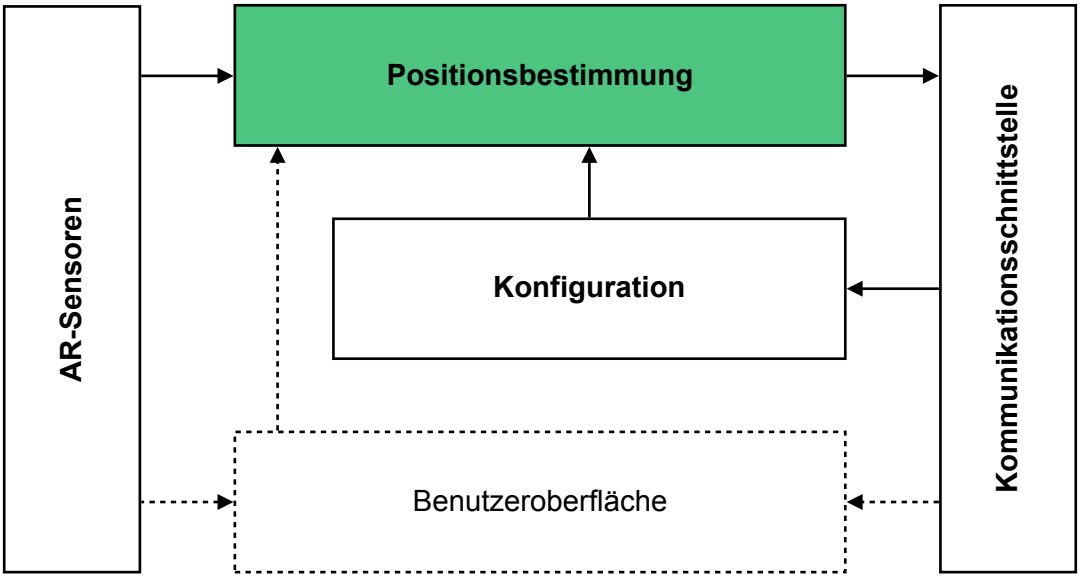


Abbildung 4.2.: Schematische Darstellung der AR-Anwendung

4.2.3. Positionsbestimmung und Kalibrierungsvorgang

Durch das von der HoloLens angewendete SLAM-Verfahren wird die Kopfposition des Headset-Trägers dauerhaft bestimmt und kann via Programmcode abgefragt werden. Da das AR-System jedoch sein eigenes Koordinatensystem festlegt, ist zunächst eine initiale Kalibrierung der HoloLens notwendig, um die Koordinaten in ein raumfestes System umwandeln zu können. Zu diesem Zweck wird im folgenden Abschnitt die Entwicklung eines automatisierten Kalibrierungsvorgangs beschrieben. Zur besseren Übersicht werden zunächst die zu unterscheidenden Koordinatensysteme definiert und benannt.

Für die Positionsbestimmung mit der HoloLens ist die Definition von insgesamt drei Koordinatensystemen notwendig. Durch die Transformation von Koordinaten kann sichergestellt werden, dass mit dem AR-Headset bestimmte Koordinaten für die Analyse in raumfesten Koordinaten vorliegen. Als Einheit für alle Koordinatensysteme werden Meter verwendet, da auch die HoloLens intern diese Einheit verwendet. Als Einheit für Rotationen werden

Grad festgelegt. Die definierten Koordinatensysteme werden in Abb. 4.3 visualisiert und im Folgenden beschrieben:

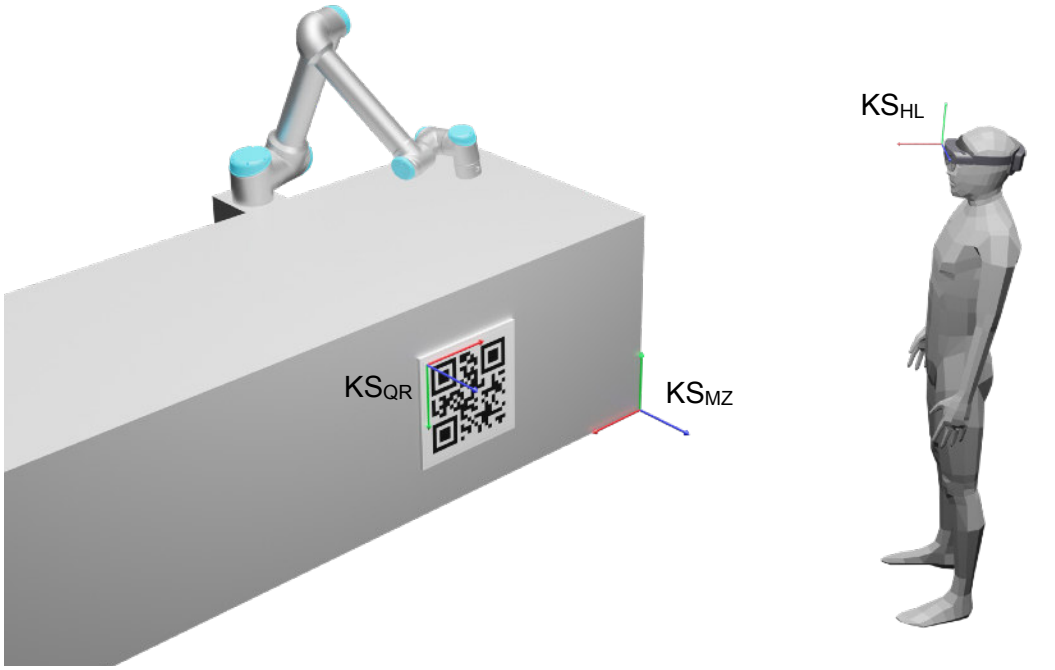


Abbildung 4.3.: Visualisierung der definierten Koordinatensysteme

- **HoloLens-Koordinatensystem KS_{HL}**: Beim Start einer AR-Anwendung wird von der HoloLens ein Koordinatensystem basierend auf der Position und Orientierung des Geräts zu diesem Zeitpunkt festgelegt. Mit dem SLAM-Verfahren bestimmt die HoloLens kontinuierlich ihre Position innerhalb dieses Koordinatensystems.
- **Raumfestes Koordinatensystem KS_{MZ}**: Das raumfeste Koordinatensystem soll sich in Bezug auf die Ursprungsposition und Achsenausrichtung an den festen Raumgeometrien orientieren. Als Bezeichnung wird die Abkürzung MZ für Montagezelle gewählt.
- **Marker-Koordinatensystem KS_{QR}**: Als Anker für die Transformation von HoloLens-Koordinaten in raumfeste Koordinaten dient ein Marker. Für die Transformation notwendig ist die Bestimmung der Marker-Ursprungsposition und Achsenorientierung sowohl in raumfesten als auch in HoloLens-Koordinaten.

In Abschnitt 3.2.3 wurden unterschiedliche Verfahren vorgestellt, um ein AR-System initial im Raum zu kalibrieren. Um einen möglichst zeiteffizienten Kalibrierungsvorgang zu erreichen, wird im Rahmen dieser Arbeit die Kalibrierung anhand eines optisch erfassten Markers mit bekannter Position untersucht. Für die Erfassung von Markern existiert eine Vielzahl an Erweiterungen und Softwarepaketen für die HoloLens, welche meist speziell für einen bestimmten Markertyp entwickelt wurden. Als Marker für den Kalibrierungsvorgang wird der QR-Code ausgewählt, da er viele für diesen Zweck hilfreiche Eigenschaften besitzt.

So lässt sich ein QR-Code aufgrund seiner asymetrischen Struktur und den insgesamt drei sogenannten „Finder Patterns“ eindeutig ausrichten und dessen Position und Lage vermessen. Zudem lassen sich im Code selbst vergleichsweise große Datenmengen z. B. in Form von Zeichenketten speichern [84]. Diese Eigenschaft kann genutzt werden, um verschiedenen Kalibrierungsmarkern eine eindeutige ID zuzuweisen, welche dann von der HoloLens ausgelesen und zur Referenzierung der hinterlegten Kalibrierungskonfiguration genutzt werden kann. Abb. 4.4 zeigt beispielhaft einen QR-Code mit dem eingezeichneten Koordinatenursprung des KS_{QR}. Microsoft stellt eine Programmbibliothek für die räumliche Erkennung von QR-Markern für die HoloLens bereit, welche auch im Rahmen dieser Arbeit verwendet wird. Der Hersteller verspricht eine Positionierungsgenauigkeit von wenigen Millimetern [85].

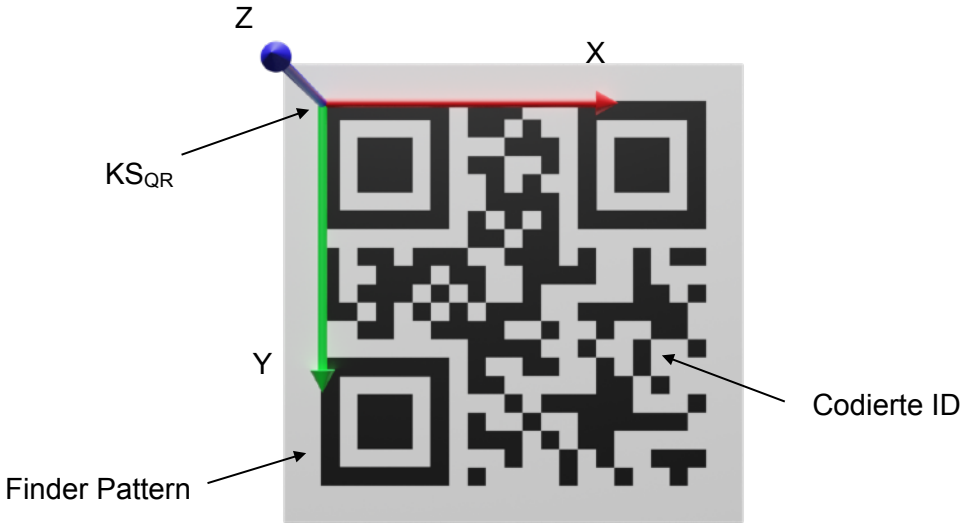


Abbildung 4.4.: Beispiel-QR-Code mit Visualisierung des Koordinatensystems KS_{QR}

Im Folgenden wird der Prozess für die Kalibrierung definiert. Ausgangspunkt für den Prozess ist ein QR-Code der an einer gut sichtbaren und gut beleuchteten Stelle in der Montagezelle angebracht ist. Der Vorgang startet, indem der Nutzer mit dem AR-Headset in Richtung des QR-Codes schaut. Die Anwendung erkennt den Code automatisch, bestimmt dessen Position und Orientierung relativ zum AR-Headset und liest die darin codierten Daten aus. Bei den Daten handelt es sich um die für den QR-Marker festgelegten ID, mit der die zugehörigen Kalibrierungskonfigurationsdaten vom Server abgerufen werden können. Aus den empfangenen geometrischen Konfigurationsdaten und der zuvor bestimmten Position und Lage des QR-Codes lässt sich die notwendige Transformationsvorschrift vom HoloLens-Koordinatensystem in das raumfeste Koordinatensystem ableiten. Abb. 4.5 zeigt den schematischen Ablauf des Kalibrierungsprozesses.

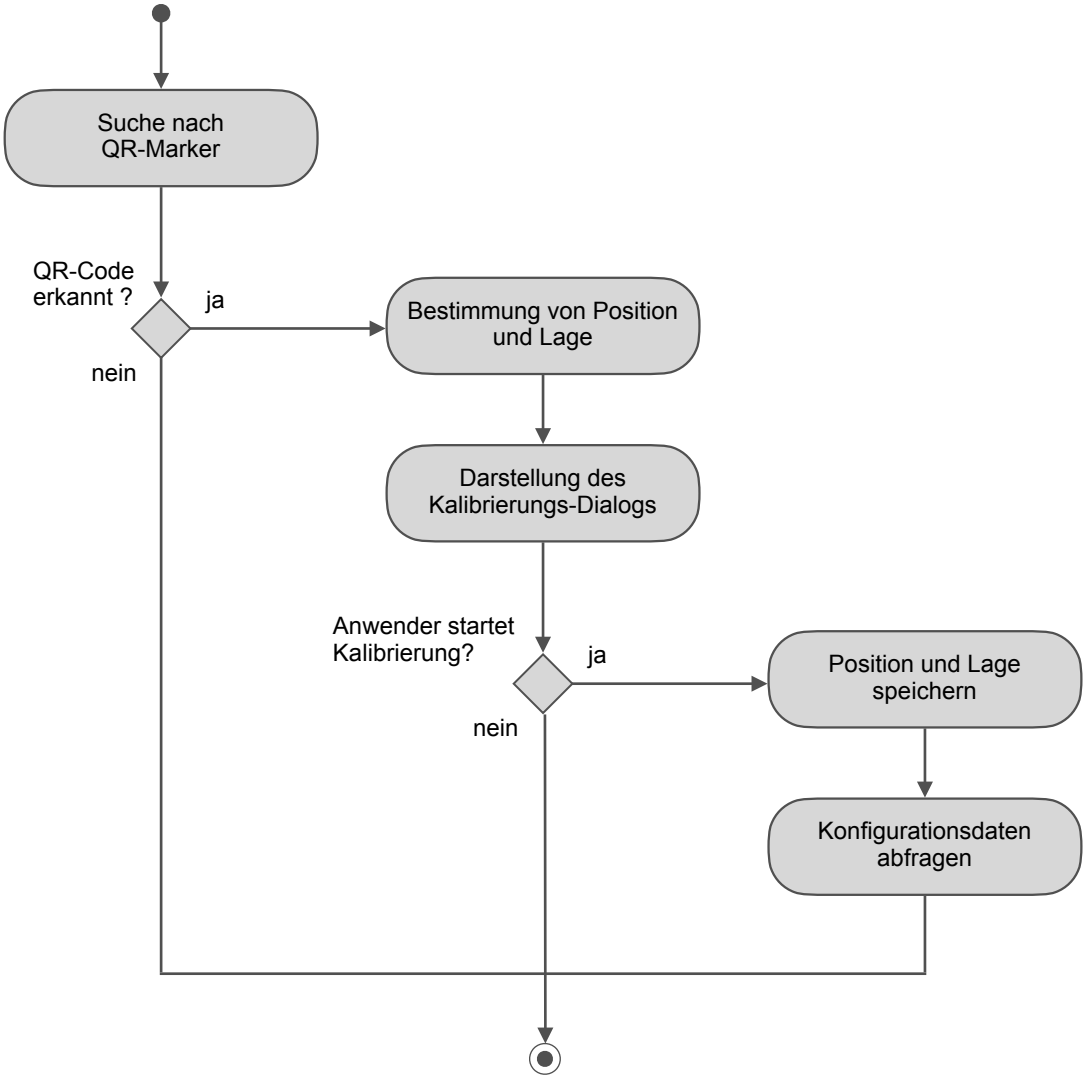


Abbildung 4.5.: Prozessschema für den automatisierten Kalibrierungsvorgang

4.2.4. Kalibrierungskonfiguration

Um die initiale Kalibrierung des AR-Systems mithilfe von QR-Codes zu ermöglichen, ist zusätzlich eine geometrische Konfiguration notwendig. In dieser wird festgehalten, welche relative Position und Orientierung ein QR-Code zum Koordinatenursprung des raumfesten Koordinatensystems KS_{MZ} hat. Der QR-Code fungiert somit als sogenannter Weltanker. Sowohl die Positions- als auch die Orientierungsverschiebung werden zu diesem Zweck jeweils in X-, Y- und Z-Wert angegeben. Um eine größtmögliche Modularität zu erreichen, wird die Kalibrierungskonfiguration als dedizierte Datei im JSON-Format gespeichert. JSON steht für JavaScript Object Notation und ist ein Datenformat, welches hauptsächlich in der Entwicklung von Webanwendungen eingesetzt wird. Da das JSON-Format sich gut für die einfache Strukturierung von Daten eignet und sowohl von Menschen als auch von Computern leicht erstellt und gelesen werden können, wird dieses für eine Vielzahl von Anwendungsfällen außerhalb der Webentwicklung eingesetzt. Aus diesem Grund sind JSON-Dateien mit allen gängigen Programmiersprachen kompatibel und können von diesen gelesen und geschrieben werden [86]. Ein JSON-Objekt wird von geschweiften Klammern umrahmt und

besteht aus einer mit Kommas separierten Liste aus Schlüsseln (engl.: keys) und Werten (engl.: values), die durch einen Doppelpunkt getrennt sind. Bei den Schlüsseln handelt es sich um Zeichenketten (engl.: strings), die mit doppelten Anführungszeichen eingerahmt sind. Bei den Werten kann es sich ebenfalls um Zeichenketten, aber auch um Zahlenwerte, Listen (engl.: arrays) oder wiederum um JSON-Objekte handeln. Dies ermöglicht eine beliebig komplexe Schachtelung der Daten im JSON-Dokument.

Die Verwendung einer solchen Konfigurationsdatei für die Kalibrierung bietet den Vorteil, dass diese zentral erstellt werden und auf eines oder mehrere AR-Headsets über die Kommunikationsschnittstelle übertragen werden kann. Um die Verwendung mehrerer Weltanker zu ermöglichen, wird eine Referenzierung über eine Anker-ID (engl.: anchor id) eingeführt. Dies ermöglicht es, in weitläufigen Produktionssystemen mehrere QR-Codes zu platzieren, um so eine Neukalibrierung an verschiedenen Orten zu ermöglichen. Listing 4.1 zeigt den Aufbau der JSON-Kalibrierungskonfiguration. Die Marker-IDs werden als JSON-Keys verwendet, als Werte werden Verschiebung und Rotation ebenfalls in der Schlüssel-Wert-Schreibweise festgehalten.

```
1  {
2    "anchorId": {
3      "transX": "-0.65",
4      "transY": "0.72",
5      "transZ": "0.60",
6      "rotX": "180",
7      "rotY": "0",
8      "rotZ": "0"
9    },
10   ...
11 }
```

Listing 4.1: JSON-Schema für die Kalibrierungskonfiguration mit Beispielwerten

4.2.5. Datenübertragung

Nachdem der Kalibrierungsvorgang abgeschlossen ist, können die Positionsdaten erfasst und in das raumfeste Koordinatensystem transformiert werden. Damit eine Analyse der Daten stattfinden kann, werden diese über eine Kommunikationsschnittstelle an die Serveranwendung übertragen. Zu diesem Zweck wird aus den transformierten Koordinaten ein JSON-Objekt generiert, welches neben den Positionskoordinaten und Orientierungswinkeln auch den Zeitstempel und einen Typen-Indikator mit der Bezeichnung „workerPosition“ enthält. Die Integration der Typenbezeichnung ermöglicht den Datenempfänger eine zielgerichtete Weiterverarbeitung. Listing 4.2 zeigt schematisch den Aufbau des JSON-Objekts. Der Server kann das Datenformat zusätzlich direkt für die Speicherung in einer Datenbank

verwenden. Die Speicherung der Rohdaten ermöglicht es, auch unabhängig vom im Rahmen dieser Arbeit gezeigten Analyseprozess, tiefer gehende Auswertungen vorzunehmen.

```
1 {
2   "dataType": "workerPosition",
3   "timestamp": "2021-12-15T14:59:18.844",
4   "posX": "1.2375",
5   "posY": "5.2241",
6   "posZ": "1.7542"
7 }
```

Listing 4.2: JSON-Objekt für Datenübertragung der Arbeiterposition mit Beispielwerten

4.3. Serveranwendung für Datenanalyse und Prozessoptimierung

Der zweite Bestandteil des entwickelten Gesamtsystems ist eine Serveranwendung zur strukturierten Verarbeitung und Analyse der Positionsdaten. Zudem soll der Server als zentraler Datenspeicher sowohl für die Positionsdaten und Analyseergebnisse als auch für die vom Anwender vorgegebenen Konfigurationsdaten dienen. Im folgenden Abschnitt werden zunächst die Anforderungen an die Serveranwendung gesammelt und daraus der grundlegende Systemaufbau abgeleitet. Anschließend wird die Entwicklung der einzelnen Systemkomponenten beschrieben.

4.3.1. Anforderungen an die Serveranwendung

Die zu entwickelnde Serveranwendung lässt sich in zwei Hauptkomponenten aufteilen: Zum einen die Kommunikationsschnittstelle zum Senden und Empfangen der Arbeiter-Positionsdaten und Konfigurationsdaten. Außerdem besitzt die Anwendung eine Einheit für die Analyse und Speicherung der Daten sowie die Prozessoptimierung.

Um für verschiedene Daten jeweils eine geeignete Form der Kommunikation anbieten zu können, soll die Kommunikationsschnittstelle aus zwei voneinander unabhängigen Subschnittstellen bestehen:

- **1-n Kommunikation nach dem Broadcast/Multicast-Prinzip:** In einem Kommunikationssystem nach diesem Schema können Daten kontinuierlich ohne die Festlegung eines Empfängers gesendet werden. Der Vorteil liegt darin, dass nicht explizit nach Daten „gefragt“ wird, sondern dauerhaft Daten empfangen werden und können dann bei Bedarf verarbeitet werden können [87]. Im Rahmen dieser Arbeit wird diese für die Übertragung der Positionsdaten verwendet.

- **1-1 Kommunikation nach dem Frage/Antwort-Prinzip:** Diese Kommunikationsform eignet sich für das gezielte Abfragen von Daten. Im Rahmen dieser Arbeit wird diese für die Übertragung von Konfigurationsdaten verwendet.

Neben der Bereitstellung der Kommunikationsschnittstelle soll der Server dazu verwendet werden, die erfassten menschlichen Positionsdaten zu verarbeiten, zu analysieren und diese zu speichern. Ziel ist die Bestimmung von Prozesszeiten mit denen beispielsweise die Prozessreihenfolge optimiert werden kann.

Entsprechend dem in Abschnitt 3.3 beschriebenen Ansatz zur Bestimmung von Prozesszeiten aus Positionsdaten ist hierfür die vorherige Definition einer geometrischen Prozessbeschreibung notwendig. Der Prozess wird darin als Abfolge von Arbeitsbereichen definiert. Die Prozesszeiten können dann berechnet werden indem die Aufenthaltsdauer des Arbeiters in einem Arbeitsbereich oder die Zeit für den Laufweg von einem Bereich in einen anderen berechnet werden. Für die Prozessbeschreibung ist ein geeignetes stukturiertes Datenformat notwendig, welches sowohl von Menschen als auch von Computern gelesen werden kann. Die auf diese Weise berechneten Prozesszeiten sollen zusammen mit den empfangenen Positionsdaten in einer geeigneten Datenbank gespeichert werden.

4.3.2. Systemaufbau der Serveranwendung

Aus den genannten Anforderungen lässt sich der für deren Erfüllung notwendige Systemaufbau ableiten, welcher in Abb. 4.6 dargestellt ist. Ausgehend vom AR-Headset lässt sich dieser wie folgt beschreiben:

Die AR-Anwendung fragt die für die Kalibrierung notwendigen Konfigurationsdaten über die entsprechende Schnittstelle ab. Anschließend wird die Arbeiterposition kontinuierlich bestimmt und über die Broadcast-Verbindung an den Server übertragen. Der Server verarbeitet jeden ankommenden Positionsdatensatz und prüft, ob dieser die Bedingungen für den Start oder das Ende eines Prozessschrittes erfüllt. Hierzu wird die vorher festgelegte geometrische Prozessbeschreibung verwendet. Wird eine der Bedingungen erfüllt, dann wird die berechnete Prozesszeit in der Datenbank gespeichert. Nachdem eine ausreichende Anzahl von Prozesszeiten errechnet wurde, werden diese verwendet um die Prozessreihenfolge zu optimieren. Die Ergebnisse der Optimierung werden als geänderte Prozessbeschreibung gespeichert. In die beschriebene Architektur lassen sich auch andere Systeme wie z. B. der kollaborative Roboter einbinden, welche sich ebenfalls über die Datenschnittstellen mit dem Server verbinden lassen. Eine Beschreibung der Roboter-Anbindung an die Serveranwendung findet sich in Anhang E.

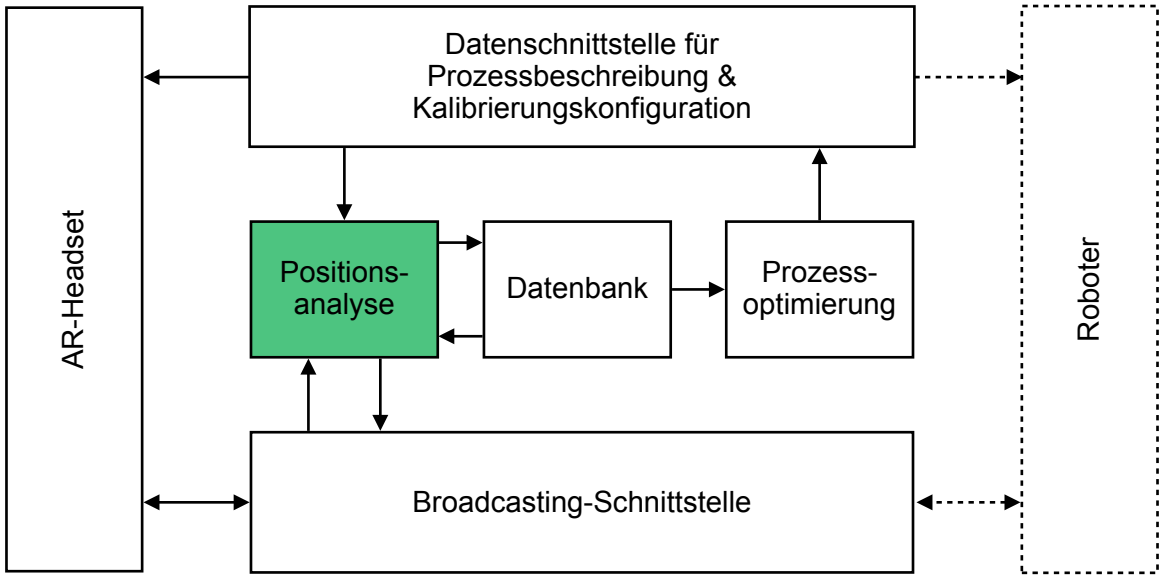


Abbildung 4.6.: Schematische Darstellung der Serveranwendung

4.3.3. Prozesskonfiguration

Für die geometrische Beschreibung des Prozesses wird ebenfalls das Datenformat JSON verwendet. Die Prozessbeschreibung besteht aus der geometrischen Definition der Kontrollvolumen sowie einer Definition der Prozessreihenfolge.

Um die Komplexität möglichst gering zu halten, wird ausschließlich die Verwendung von quaderförmigen Kontrollvolumen berücksichtigt. Diese lassen sich beschreiben, indem ein Mittelpunkt in X-,Y- und Z-Koordinaten definiert und die Länge, Breite und Höhe festgelegt wird. Als Einheit werden Meter verwendet. Zusätzlich wird eine ID zur Referenzierung des Kontrollvolumens festgelegt. Alle auf diese Art und Weise definierten Geometrien werden als Array unter dem Schlüssel „geometries“ im JSON-Dokument gespeichert.

Der Arbeitsprozess wird als Abfolge von Aufgaben (tasks) definiert, welche jeweils eine eindeutige ID besitzen. Um eine bessere Übersicht zu erreichen, werden Aufgaben nochmal in beliebig viele Prozesschritte (steps) unterteilt, welche als Array unter dem Schlüssel „steps“ im Dokument gespeichert werden. Jeder Prozessschritt besitzt ebenfalls eine eindeutige ID. Außerdem wird für jeden Schritt definiert, in welchem Kontrollvolumen der Schritt startet und in welchem er endet („startGeometry“ und „endGeometry“). Die Referenzierung der Kontrollvolumen erfolgt mit der jeweils festgelegten ID. Um eine größere Anzahl an Prozessvarianten abbilden zu können, lässt sich außerdem definieren, ob der Prozessschritt beim Betreten (enter) oder beim Verlassen (leave) des Kontrollvolumens gestartet bzw. beendet wird. Die zugehörigen JSON-Schlüssel sind „startConfType“ und „endConfType“.

Zuletzt wird die Möglichkeit vorgesehen, für jede Aufgabe festzulegen, von welchen anderen Aufgaben diese abhängt. Unter dem JSON-Schlüssel „dependencies“ kann ein Array mit IDs anderer Aufgaben definiert werden. Diese können dann bei der späteren Optimierung

der Prozessreihenfolge verwendet werden, um Aufgaben, welche zwingend nach einer anderen ausgeführt werden müssen, in der richtigen Reihenfolge zu planen. Die gesamte JSON-Prozessbeschreibung ist in Listing 4.3 dargestellt.

```
{
  "tasks": [{
    "id": "A",
    "steps": [{
      "id": "A1",
      "startGeometry": "123",
      "endGeometry": "123",
      "startConfType": "enter",
      "endConfType": "leave"},
      ...
    ],
    "dependencies": ["B"]
    },
    ...
  ],
  "geometries": [{
    "id": "123",
    "posX": "1",
    "posY": "0.2",
    "posZ": "-1",
    "length": "0.8",
    "width": "2",
    "height": "0.8",
    },
    ...
  ]
}
```

Listing 4.3: JSON-Schema für die geometrische Prozessbeschreibung mit Beispielwerten

4.3.4. Datenschnittstelle

Gefordert wird die Integration einer Datenschnittstelle die sowohl die 1-n Kommunikation nach dem Broadcast/Multicast-Prinzip als auch die 1-1 Kommunikation nach dem Frage/Antwort-Prinzip beherrscht. Dementsprechend werden zwei Kommunikationstechnologien ausgewählt, welche sich sowohl in die Serverumgebung als auch in die HoloLens integrieren lassen.

Für die Broadcasting-Kommunikation kommt die sogenannte Websockets-Technologie zum Einsatz [88]. Diese ermöglicht einem Server, sich direkt mit mehreren Clients zu verbinden. Der initiale Verbindungsvorgang wird dabei als Handshake bezeichnet. Nach dem Handshake sind sowohl Clients als auch Server dauerhaft bereit Nachrichten voneinander zu empfangen. Der Server wird für diesen Anwendungsfall so programmiert, dass er alle eintreffenden Nachrichten an alle verbundenen Clients außer den Sender weiterleitet, was dann als Broadcast bezeichnet wird. Abb. 4.7 zeigt schematisch die Websocket-Kommunikation.

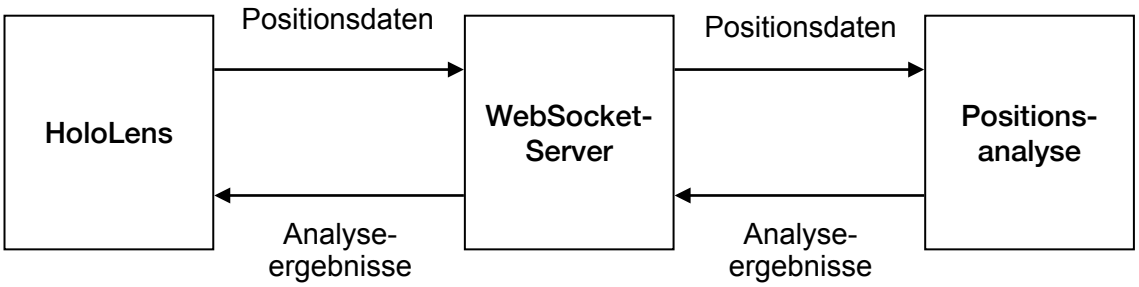


Abbildung 4.7.: Schematische Darstellung der Websocket-Kommunikation für die Übertragung von Positionsdaten und der Analyseergebnisse

Über die 1-1 Kommunikation soll die Kalibrierungskonfiguration als statische Datei vom Server abgefragt werden können. Für Anwendungsfälle dieser Art eignet sich eine serverseitig implementierte API (Application Programming Interface) nach dem REST-Paradigma (Representational State Transfer) [89]. Es wird ein API-Endpunkt definiert, welcher auf Anfragen mit der HTTP-Methode GET reagiert. Der Client sendet als Parameter die ID des Markers, dessen Konfigurationsdaten er empfangen möchte. Der Server antwortet mit den Transformationswerten im JSON-Format, dessen Schema in Listing 4.1 veranschaulicht wird. Die Client-Server-Kommunikation wird in Abb. 4.8 schematisch dargestellt.

4.3.5. Datenspeicherung

Neben der Echtzeit-Verarbeitung der Positionsdaten sollen diese und die ermittelten Prozessparameter in einer Datenbank gespeichert werden. Da es sich hierbei um verhältnismäßig kleine Datenmengen handelt und es auch keine besonderen Anforderungen an die Zugriffsgeschwindigkeit gibt, kommt eine Vielzahl von Datenbanktechnologien in Frage. Um die Entwicklung zu vereinfachen, kommt die Datenbank MongoDB der gleichnamigen Firma zum Einsatz [90]. Hierbei handelt es sich um eine Dokumentendatenbank, eine Datenbanktechnologie die zu den NoSQL bzw. nicht-relationalen Datenbanken gehört. Charakteristisch für NoSQL-Datenbanken ist, das diese nicht auf starren Datenstrukturen basieren und das Datenbank-Schema variabel ist. Dies ermöglicht es z. B. neue Datenfelder dynamisch hinzuzufügen und so heterogene Daten an einem Ort zu speichern, was zu schnelleren

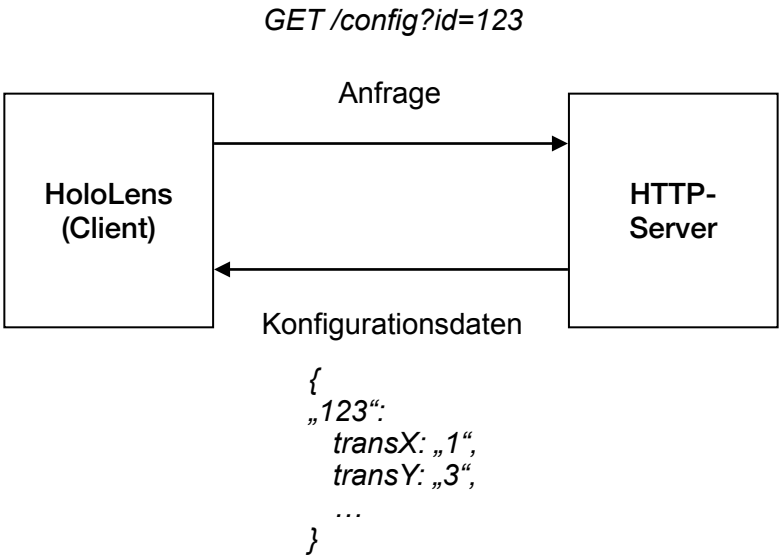


Abbildung 4.8.: Schematische Darstellung der Client-Server-Kommunikation für die Abfrage der Kalibrierungsdaten

Entwicklungszeiten führt und für den Entwickler meist intuitiver und komfortabler ist [91]. Dokumentendatenbanken wie MongoDB speichern Daten, wie der Name bereits andeutet, in Dokumenten und nicht in Form von Tabellen wie dies z. B. in relationalen Datenbanken geschieht. Die Dokumente verwenden strukturierte Datenformate, häufig das bereits erwähnte JSON-Format [92]. MongoDB bietet Schnittstellen zu allen gängigen Programmiersprachen an [93].

Eine MongoDB-Datenbank lässt sich mit sogenannten Collections strukturieren. Diese dienen dazu, Dokumente mit ähnlicher Datenstruktur oder -quelle zu gruppieren. Für die entwickelte Anwendung werden zwei Collections erstellt, eine für die erfassten Positions-Rohdaten und eine für die ermittelten Prozesszeiten. Da in MongoDB Daten im JSON-Format direkt in der Datenbank gespeichert werden können ist für die Sicherung der empfangenen Positionsdaten keine Umformung notwendig. Das Format mit dem die Daten gespeichert werden entspricht demnach dem zuvor festgelegten Format, welches in Listing 4.2 dargestellt ist.

Für die Speicherung der Prozesszeiten wird ein zusätzliches Format definiert, mit dem die errechneten Zeiten in einer separaten Collection gespeichert werden können. Benötigt werden insgesamt vier Attribute: Die ID der zugehörigen Aufgabe, den Zeitpunkt an dem die Aufgabe gestartet und beendet wurde sowie die Dauer, welche nach Beendigung der Aufgabe berechnet werden kann. Das Schema des JSON-Objekts ist in Listing 4.4 dargestellt.

```
{
  ID: "A",
  timestampStart: "2021-12-15T14:54:43.431",
  timestampEnd: "2021-12-15T14:59:18.844",
  duration: "275.413"
}
```

Listing 4.4: JSON-Objekt für die Speicherung der berechneten Prozesszeiten mit Beispielwerten

4.3.6. Algorithmus zur Bestimmung von Prozesszeiten

Für die Bestimmung von Prozesszeiten aus den erfassten Positionen wird der in Abschnitt 3.3 beschriebene Ansatz verwendet. Grundlage für den Algorithmus ist eine geometrische Beschreibung des Prozesses, welche sowohl die Reihenfolge der Prozessschritte als auch die Kontrollvolumen in denen diese stattfinden beinhaltet. Zusätzlich ist definiert, ob der Teilprozess mit dem Betreten oder dem Verlassen der geometrischen Begrenzung gestartet oder beendet wird (siehe 4.3.3).

Um die Prozesszeiten zu bestimmen kommen zwei unterschiedliche Verfahren in Frage:

- **Echtzeit-Verarbeitung:** Die empfangenen Positionsdaten werden in der Reihenfolge der Erfassung einzeln verarbeitet und auf Grundlage eines einzelnen Messwerts bestimmt, ob eine Aufgabe begonnen oder abgeschlossen wurde oder diese derzeit noch ausgeführt wird. Vorteil dieser Variante ist die unmittelbare Verfügbarkeit der Zeitdaten und die permanente Dokumentierung des Produktionsfortschritts.
- **Stapel-Verarbeitung:** Die empfangenen Positionsdaten werden zunächst gesammelt und verarbeitet nachdem z. B. ein Produktionszyklus abgeschlossen ist. Vorteil dieser Variante ist die Möglichkeit, die Daten vor der Verarbeitung zu Filtern um beispielsweise Messwertausreißer oder Randfälle aussortieren zu können. Dies ermöglicht eine zuverlässigere Bestimmung der Prozesszeiten.

Für diese Arbeit wird lediglich das Verfahren der Echtzeit-Verarbeitung angewendet. Der im Folgenden beschriebene Algorithmus lässt sich jedoch auch für die Stapel-Verarbeitung verwenden. Abb. 4.9 zeigt das Ablaufdiagramm des Algorithmus zur Bestimmung von Prozesszeiten.

Ausgangspunkt für den Analysealgorithmus ist ein Positionsdatensatz bestehend aus einer X-, Y- und Z-Koordinate. Durch die vorangegangene Transformation der Koordinaten besitzt der Datenpunkt mit dem KS_{MZ} das selbe Bezugskoordinatensystem wie die zuvor festgelegten Kontrollvolumen. Der Algorithmus beginnt mit dem Einlesen der geometrischen Daten

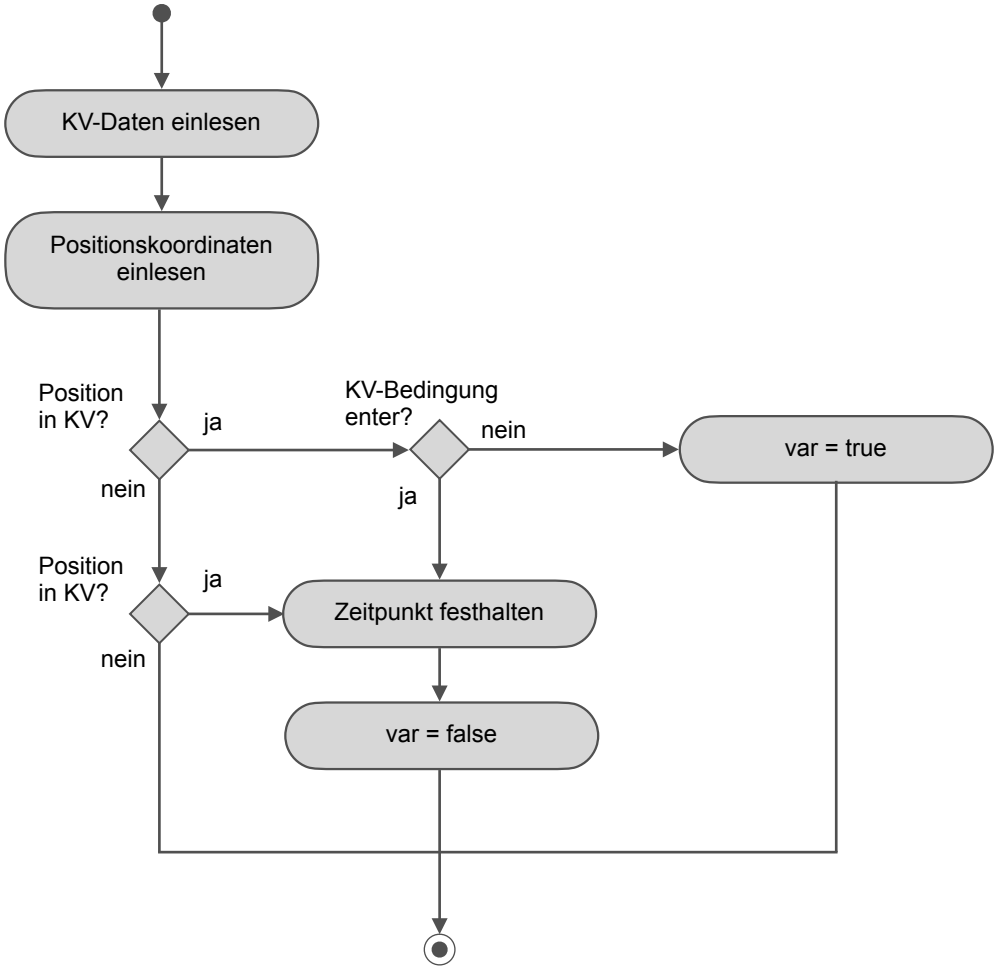


Abbildung 4.9.: Ablaufdiagramm des Algorithmus für die Prozesszeitbestimmung

des Start-Kontrollvolumens des ersten Prozessschrittes. Nun werden die ankommenden Positionsdaten nacheinander dahingehend geprüft, ob sie sich innerhalb des Kontrollvolumens befinden. Ist dies der Fall, gilt der Prozessschritt als begonnen und der Zeitpunkt wird in der Datenbank festgehalten. Nun wird das End-Kontrollvolumen eingelesen und entsprechend mit den empfangenen Datenpunkten verglichen. Wird eine Kollision festgestellt, wird der entsprechende Endzeitpunkt festgehalten. Einen Sonderfall bilden Kontrollvolumen, bei denen das Verlassen als Auslöser definiert wurde. Hier wird die detektierte Kollision zunächst nur in einer Hilfsvariable festgehalten und die Überprüfungslogik für die nachfolgenden Datenpunkte umgekehrt. Es wird also überprüft, ob die empfangene Position außerhalb des Kontrollvolumens liegt. Wird auf diese Weise ein Verlassen des Kontrollvolumens durch den Arbeiter detektiert, wird der entsprechende Zeitpunkt ebenfalls festgehalten. Nachdem Start- und Endzeitpunkt für einen Prozessschritt bekannt sind, kann die Prozessdauer berechnet werden und das Ergebnis referenziert durch die Prozess-ID in der Datenbank gespeichert werden. Hierzu kommt das in Listing 4.4 definierte Datenformat zum Einsatz.

4.3.7. Algorithmus zur Optimierung der Prozessreihenfolge

Die Optimierung der Prozessreihenfolge dient im Rahmen dieser Arbeit als Anwendungsbeispiel für die Nutzung der automatisiert bestimmten Prozesszeiten. In der Fachliteratur wird zur Verallgemeinerung von Optimierungsaufgaben in der industriellen Ablaufplanung häufig das sogenannte „Job Shop Problem“ herangezogen. Das Szenario des Job Shops lässt sich wie folgt beschreiben: In einer Produktionsanlage werden mehrere Aufträge (Jobs) gleichzeitig bearbeitet. Diese Aufträge bestehen wiederum aus verschiedenen Aufgaben (Tasks), welche von unterschiedlichen Maschinen (bzw. Robotern oder Arbeitern) durchgeführt werden. Hierbei gelten folgende Voraussetzungen [94]:

- Eine Aufgabe kann nur durchgeführt werden, wenn die vorherige Aufgabe des gleichen Auftrags abgeschlossen ist.
- Eine Maschine kann jeweils nur eine Aufgabe ausführen.
- Aufgaben können nicht unterbrochen werden.

Bei diesem Problem handelt es sich um eine Spezialform des „Traveling Salesman Problems“, wobei es sich um ein NP-schweres Optimierungsproblem handelt, was deshalb auch für das Job Shop Problem gilt [95]. NP ist eine Abkürzung für „nichtdeterministisch polynomielle Zeit“. Vereinfacht bedeutet dies, dass für NP-schwere Probleme die exakte Lösung nicht in einem realistischen Zeitrahmen bestimmt werden kann, weshalb Näherungsalgorithmen eingesetzt werden müssen [96]. Varianten des Problems, z. B. mit mehreren Maschineninstanzen für die gleiche Aufgabe oder die Berücksichtigung von Rüst- und Übergangszeiten erhöhen die Komplexität nochmals zusätzlich [97]. Da die Auseinandersetzung mit unterschiedlichen Algorithmen zur Lösung von Job Shop Problemen über den wissenschaftlichen Rahmen dieser Arbeit hinausgeht, wird für diese Arbeit der Open-Source-Optimierungsalgorithmus CP-SAT Solver der Firma Google [98, 99] eingesetzt und auf die Validierung der Optimierungsergebnisse verzichtet. Aus diesem Grund wird an dieser Stelle zudem auf eine detaillierte Beschreibung der Implementierung des Algorithmus verzichtet. Interessierte Leser finden technische Details zur Implementierung und Anwendung des Optimierungsalgorithmus in Anhang D.

5 Implementierung

Im nachfolgenden Kapitel wird die Implementierung der zuvor entworfenen Anwendung beschrieben. Es werden ausgewählte, für das Verständnis wichtige Quellcode-Ausschnitte gezeigt und die für die Umsetzung verwendete Software genannt. Der Quellcode in seiner Gesamtheit ist zusätzlich auf dem beigefügten Datenträger verfügbar. Das Kapitel untergliedert sich wie auch bereits der Systementwurf in zwei Hauptteile, die HoloLens-Anwendung zur Positionsbestimmung und die Serveranwendung zur Analyse der Positionsdaten.

5.1. HoloLens-Anwendung

Für die Entwicklung der HoloLens-AR-Anwendung wird die Entwicklungsumgebung Unity der Firma Unity Technologies in der Version 2020.3.4f1 verwendet [56]. Die Anwendung gliedert sich in zwei Teile, das Positionsbestimmungssystem und das Modul zur Übertragung der Positionsdaten. Die Implementierung beider Teile wird im folgenden Abschnitt beschrieben.

5.1.1. Positionsbestimmungssystem

Kalibrierung

Um die Kopfposition des Arbeiters in den raumfesten Koordinaten der Montagezelle zu bestimmen, wird zunächst das Kalibrierungsverfahren implementiert. Es wurde festgelegt, dass die Kalibrierung mit einem QR-Marker mit bekannter Raumposition erfolgen soll. Damit die HoloLens mit den verbauten Sensoren QR-Codes und deren relative Position erkennen kann, wird das von Microsoft bereitgestellte QR SDK verwendet [100]. Die Entwicklung von Unity-Anwendungen ist stark fokussiert auf die Darstellung von 3D-Objekten. Damit die Bewegungen dieser Objekte flüssig erscheinen, muss das dem Anwender angezeigte Bild ständig neu berechnet werden. Die Häufigkeit dieser Berechnung wird in der Einheit Bilder pro Sekunde (engl.: frames per second, FPS) angegeben. Die HoloLens erreicht eine Bildrate von bis zu 60 FPS [101]. In Unity können Funktionen programmiert werden, die bei jeder Neuberechnung des Bildes ausgeführt werden. Auch das verwendete QR SDK nutzt dies und prüft bei jeder Frame-Neuberechnung, ob sich ein QR-Code im Sensor-Sichtfeld befindet und bestimmt gegebenenfalls dessen Position und Orientierung im HoloLens-Koordinatensystem (KS_{HL}) und liest die im QR-Code gespeicherten Daten aus.

Um dem Anwender die Kontrolle über den Kalibrierungsvorgang zu geben, wird eine virtuelle Schaltfläche über dem erkannten QR-Code platziert. Berührt der Anwender diese, startet der Kalibrierungsprozess, indem zunächst die Kalibrierungskonfiguration mit der ausgelesenen QR-Code-ID vom Server abgefragt wird. Zu diesem Zweck wird ein HTTP-GET-Request

gesendet, welcher als Parameter die ID enthält. Der Aufbau eines solchen Requests ist in Listing 5.1 dargestellt.

```
1 GET "http:/10.17.213.164:3000/configdata?&id=" + id;
```

Listing 5.1: GET-Request zur Abfrage der Konfigurationsdaten

Nachdem der Server mit den Konfigurationsdaten geantwortet hat, werden diese genutzt, um ein zuvor erstelltes Unity-Objekt, welches den Koordinatensystem-Ursprung des raumfesten Koordinatensystems veranschaulichen soll, an der entsprechenden Stelle in der 3D-Szene zu platzieren. Wie dieses Objekt genutzt werden kann, um die notwendigen Koordinatentransformationen auszuführen, wird im nächsten Abschnitt beschrieben. Abb. 5.1 zeigt die über dem QR-Code platzierte virtuelle Schaltfläche und die AR-Visualisierung des raumfesten Koordinatensystems KS_{MZ}.

Abbildung 5.1.: Virtuelle Schaltfläche auf dem QR-Marker und Visualisierung des raumfesten Koordinatensystems KS_{MZ}

Koordinatentransformation

Nach dem Kalibrierungsvorgang befindet sich an der Stelle des Raum-Koordinatenursprungs ein Unity-Objekt, welches in Listing 5.2 als „roomOrigin“ bezeichnet ist. Die Position der HoloLens kann abgefragt werden, indem die Position der Unity-Kamera abgefragt wird. Die Kamera befindet sich im Koordinatensystem der Anwendung dort, wo auch der Kopf des Anwenders befindet, spiegelt also die Ergebnisse des vom System dauerhaft durchgeführten SLAM-Trackings wieder. Durch die Anwendung der Funktion „InverseTransformPoint()“ können die HoloLens-Koordinaten in Raum-Koordinaten transformiert werden. Nach der Transformierung wird der Zeitpunkt festgehalten und ein JSON-Objekt entsprechend Listing

4.2 generiert und in Form einer Zeichenkette gespeichert, welche im nächsten Schritt an den Server gesendet wird.

```
1 Vector3 position = roomOrigin.InverseTransformPoint(Camera.main.↙
    ↳ transform.position);
```

Listing 5.2: Transformation der Kopfposition von HoloLens- in Raumkoordinaten

5.1.2. Datenübertragung

Die Übertragung der Positionsdaten soll mithilfe der Websockets-Technologie erfolgen. Zur Implementierung dieser in die HoloLens-Unity-Anwendung kommt das Open-Source-Paket NativeWebSocket in der Version 1.1.1 zum Einsatz [102]. Es wird davon ausgegangen, dass der Anwendung die IP-Adresse des Servers und der zugehörige Port bekannt sind. Mit dem Befehl „new WebSocket“ wird die Verbindung zum Server hergestellt. Nach dem Verbindungsaufbau kann die Funktion „SendText“ genutzt werden, um die Positionsdaten zu senden (siehe Listing 5.3).

```
1 websocket = new WebSocket("ws://10.17.213.164:3000");
2 websocket.SendText(positionData);
```

Listing 5.3: Websocket-Kommunikation in der HoloLens-Anwendung

5.1.3. Verwendete Software für die AR-Anwendung

Die für die Entwicklung der AR-Anwendung verwendete Software ist in Tabelle 5.1 aufgelistet. Die verwendeten Unity-Pakete sind ebenfalls in der Programmiersprache C# implementiert.

Tabelle 5.1.: Übersicht der für die AR-Anwendung verwendeten Softwareversionen

Name	Entwickler	Version
C#	Microsoft	6.0
Unity	Unity Technologies	2020.3.4f1
MRTK	Microsoft	2.7.2.0
MixedReality.QR SDK	Microsoft	0.5.3013
NativeWebSocket	GitHub/endel	1.1.1

5.2. Serveranwendung

5.2.1. Kommunikationsschnittstelle

Die Serveranwendung wird in der Programmiersprache JavaScript implementiert und in der Laufzeitumgebung Node.js ausgeführt [103]. Die Verwendung von Paketen (engl.: packages) ermöglicht es, sowohl die Websocket- als auch die API-Integration ohne großen Aufwand zu realisieren. Für die Implementierung der Websocket-Schnittstelle wird das Paket WS verwendet [104]. In Listing 5.4 ist ein Ausschnitt des JavaScript Codes für die Definition des Websocket-Servers dargestellt. Dieser wird so programmiert, dass er alle eingehenden Nachrichten an alle verbundenen Clients außer den Sender selbst weiterleitet.

```
1 wss.on('connection', function(ws) {
2   ws.on('message', function(data) {
3     wss.clients.forEach(function (client) {
4       if (client !== ws) {
5         client.send(data)
6       }
7     })
8   })
9 })
```

Listing 5.4: Websocket-Implementierung des Servers

Die Implementierung der HTTP-Endpunkte erfolgt mit dem Paket Express [105]. Listing 5.5 zeigt, wie der GET-Endpunkt „getconfig“ definiert wird. Die zugehörige Funktion liest den ID-Parameter der Anfrage aus und gibt die für die ID gespeicherten Konfigurationsdaten im JSON-Format zurück (siehe Listing 4.1).

```
1 app.get('/getconfig', function(req, res) {
2   id = req.query.id
3   res.send(qrConfigData[id])
4 })
```

Listing 5.5: Express-Implementierung des Servers

5.2.2. Datenbankanbindung

Für die Anbindung der MongoDB-Datenbank wird das Node-Paket mongoose verwendet [106]. Dieses ermöglicht es in wenigen Schritten, Daten aus der MongoDB-Datenbank abzurufen bzw. zu speichern. Listing 5.6 zeigt den dafür notwendigen Code. Zunächst wird ein sogenanntes Schema definiert, in dem festgehalten wird, welche Felder und Datentypen ein Dokument in der Datenbank-Collection hat. Anschließend wird das Schema zu einem

sogenannten „Model“ umgewandelt, also mit einer Collection verknüpft. In diesem Fall ist dies die Collection „Workerpositions“. Zuletzt kann mit dem Model ein neues Dokument instanziiert werden, an welches die zu speichernden Daten übergeben werden. Der Befehl „document.save()“ speichert das neue Dokument schließlich in der Datenbank.

```
1  const positionSchema = new mongoose.Schema({
2    timestamp: { type: Date, required: true },
3    headPosition: {
4      x: Number,
5      y: Number,
6      z: Number
7    }
8  })
9
10 Workerposition = mongoose.model('Workerposition', positionSchema)
11
12 wss.on('connection', function(ws) {
13   ws.on('message', function(data) {
14     let document = new Workerposition
15     document.timestamp = data[timestamp]
16     document.headPosition = {
17       x: data[X],
18       y: data[Y],
19       z: data[Z]
20     }
21     document.save()
22   })
23 })
```

Listing 5.6: Datenbankanbindung mit mongoose.js

5.2.3. Algorithmus zur Bestimmung von Prozessschrittzeiten aus Positionsdaten

Der Algorithmus für die Bestimmung der Prozessschrittzeiten wird gemäß dem in Abb. 4.9 dargestellten Prozessschema implementiert. Zentraler Bestandteil des Algorithmus ist die Prüfung, ob sich die verarbeiteten Positionsdaten innerhalb eines Kontrollvolumens befinden oder nicht. Zu diesem Zweck wird die Funktion „checkCollision“ definiert, welche als Eingangsparameter die zu prüfende Arbeiterposition und die geometrischen Grenzen des Kontrollvolumens erwartet. Da es sich bei den Kontrollvolumen um einfache Quader handelt, lässt sich die Prüfung durchführen, indem jede Positionskoordinate daraufhin überprüft wird, ob sie zwischen den Grenzen des Kontrollvolumens in der entsprechenden Koordinatenrichtung liegt. Befindet sich die Position innerhalb des Volumens, wird *true* zurückgegeben,

ansonsten *false*. Der verwendete JavaScript-Code ist in Listing 5.7 dargestellt.

```
1  function checkCollision(workerPos, geomBounds) {
2
3    if (workerPos.x > geomBounds.xMin &&
4      workerPos.x < geomBounds.xMax &&
5      workerPos.y > geomBounds.yMin &&
6      workerPos.y < geomBounds.yMax &&
7      workerPos.z > geomBounds.zMin &&
8      workerPos.z < geomBounds.zMax) {
9
10     return true
11
12   } else {
13
14     return false
15
16   }
17
18 }
```

Listing 5.7: Implementierung der Funktion zur Überprüfung, ob sich die empfangene Position innerhalb eines Kontrollvolumens befindet

5.2.4. Verwendete Software für die Serveranwendung

Die für die Entwicklung der Serveranwendung verwendete Software ist in Tabelle 5.2 aufgelistet. Python wird für die Implementierung des Optimierungsalgorithmus verwendet (siehe Anhang D).

Tabelle 5.2.: Übersicht der für die Serveranwendung verwendeten Softwareversionen

Name	Entwickler	Version
MongoDB	MongoDB, Inc.	5.0.4 Community
Python	Python Software Foundation	3.7
Node.js	OpenJS Foundation	16.13.0

6 Validierung

Die Validierung des entwickelten Systems erfolgt in zwei Schritten. Zunächst wird durch ein photogrammetrisches Messverfahren die Präzision der HoloLens-Anwendung zur Positionsbestimmung ermittelt. Anschließend wird das Gesamtsystem bestehend aus Positionsbestimmung und Positionsanalyse anhand eines beispielhaften Montageprozesses praktisch erprobt.

6.1. Messreihe: Präzision der Positionsbestimmung

Ziel der Entwicklung der Positionsbestimmung-Anwendung für die HoloLens war es, auf ortsfeste Lokalisierungshardware verzichten zu können, indem Daten des kontinuierlichen Kartierungs- und Lokalisierungsprozesses des AR-Systems genutzt werden. Ein maßgeblicher Faktor für die Bewertung des entwickelten Systems ist die Präzision der Positionsbestimmung und die Frage, ob diese vergleichbar mit anderen IPS ist.

In Abschnitt 3.2 wurde anhand von Literaturwerten eine erwartbare Lokalisierungsgenauigkeit von ca. 100 mm ermittelt. Da jedoch keine Werte verfügbar sind, die sich exakt auf den im Rahmen dieser Arbeit untersuchten Fall, eines automatisierten Kalibrierungsprozess anhand eines optischen Markers beziehen, wird eine Messreihe durchgeführt, um diesen Faktor in die Bewertung einfließen lassen zu können. Die Messreihe wendet das Verfahren der Photogrammetrie an, welches es ermöglicht, durch die Aufnahme von Bildern eine dreidimensionale Vermessung von Objekten vornehmen zu können.

Zunächst werden die technischen Grundlagen des Photogrammetrie-Verfahrens erläutert und die Eignung für die durchzuführende Messreihe überprüft. Anschließend wird die Vorgehensweise beschrieben und schließlich die Ergebnisse dokumentiert. Die Bewertung und Einordnung der Messergebnisse folgt in Abschnitt 7.1.1.

6.1.1. Technische Grundlagen

Grundprinzip der Photogrammetrie ist es, ein Objekt aus verschiedenen Perspektiven aufzunehmen, um anschließend dreidimensionale Merkmale aus den Aufnahmen rekonstruieren zu können. Photogrammetrische Aufnahmen können mit einer Vielzahl an Methoden gemacht werden, bei denen z. B. Laser- oder Radarwellen ausgesendet werden und deren Reflexion detektiert wird. Technisch einfacher sind Verfahren, bei denen auf aktive Strahlenquellen verzichtet wird und rein passive Aufnahmen des Objektes in Form von Fotos gemacht werden [107]. Ein solches bildbasiertes Verfahren wird auch für die Validierung der Positionsbestimmungsanwendung verwendet, nämlich das Photogrammetrie-System

TRITOP der Firma GOM. Dieses besteht aus einer Spiegelreflexkamera mit speziellem Objektiv zur Aufnahme der benötigten Bilder sowie einer Software, in der die Aufnahmen automatisiert ausgewertet und die gewünschten Maße bestimmt werden können [108]. Für das TRITOP-System gibt der Hersteller die in einem genormten Test bestimmte Präzision der Vermessung mit 0,028 mm an, was in diesem Fall mehr als ausreichend für die Validierung der HoloLens-Positionsbestimmung ist [109].

6.1.2. Messaufbau

Am Roboter-Arbeitstisch wird ein gedruckter QR-Code in DIN-A3-Größe angebracht. Damit die Position des QR-Codes später in der Photogrammetrie-Software ausgewertet werden kann, werden drei Referenzpunkte jeweils an den Ecken der quadratischen Finder Patterns aufgeklebt. An der HoloLens werden ebenfalls vier Referenzpunkte aufgeklebt, sodass die Position und Orientierung der HoloLens später bestimmt werden kann. Die Platzierung der Referenzpunkte ist in Abb. 6.1 dargestellt.

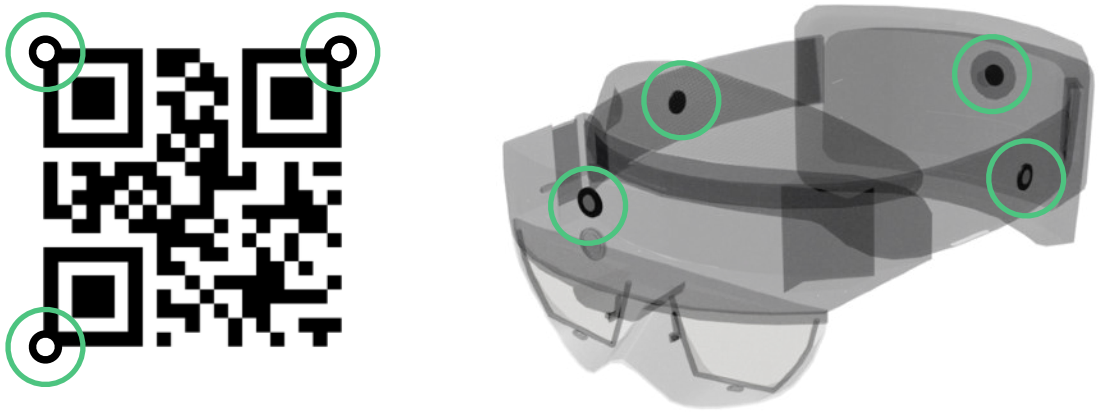


Abbildung 6.1.: Platzierung der Referenzpunkte auf QR-Marker und HoloLens

Für die Verwendung des TRITOP-Systems sind zusätzlich Maßstäbe und identifizierbare Zusatzreferenzpunkte notwendig, welche auf dem Boden verteilt werden bzw. an Arbeitstisch und Stativ befestigt werden. Diese ermöglichen es der Photogrammetrie-Software, einen geometrischen Zusammenhang zwischen den Einzelbildern herzustellen und die exakte Vermessung der aufgeklebten Referenzpunkte vorzunehmen. Abb. 6.2 zeigt die Anordnung der für die Messreihe benötigten Systemkomponenten.

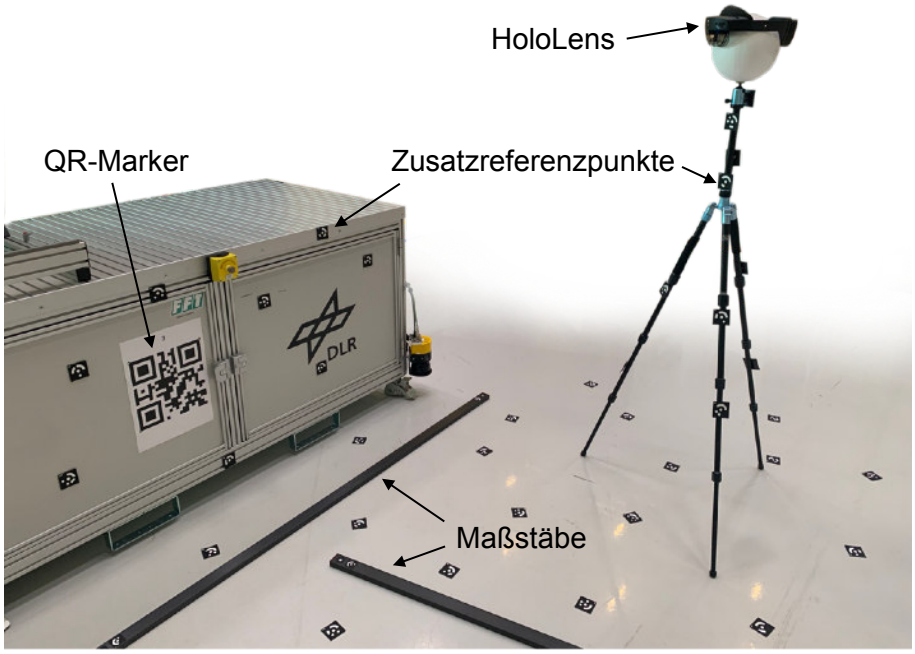


Abbildung 6.2.: Messaufbau für die Präzisionsuntersuchung der Positionsbestimmung

6.1.3. Durchführung

Insgesamt werden 15 Messungen in drei unterschiedlichen Positionen vorgenommen. Zunächst wird die HoloLens-Anwendung gestartet, die Kalibrierung durch die Erkennung des QR-Codes von einer einheitlichen Position in etwa 1,5 m Entfernung zum Marker vorgenommen. Nach der Kalibrierung wird das Headset auf einem Stativ in der jeweiligen Messposition abgelegt. Anschließend wird eine Minute lang die von der HoloLens bestimmte und an den Server gesendete Position aufgezeichnet und letztlich die Realposition durch das Photogrammetrie-System bestimmt. Zu diesem Zweck werden mit der TRITOP-Kamera etwa 50 Bilder aus verschiedenen Perspektiven aufgenommen. Nach jedem Messzyklus wird die HoloLens-Anwendung neu gestartet und neu kalibriert, sodass voneinander unabhängige Messergebnisse erzeugt werden. Die Position der Messpunkte ist in Abb. 6.3 schematisch dargestellt.
Nach erfolgter Durchführung der Einzelmessungen können die vom AR-System an den Server gesendeten Positionsdaten heruntergeladen werden und mit den durch das TRITOP-System gemessenen Referenzdaten verglichen werden.

Mithilfe der TRITOP-Software lässt sich die exakte Position der HoloLens für alle 15 Messpunkte bestimmen. Hierzu werden die vier an der HoloLens angebrachten Marker verwendet, welche automatisch von der Software erkannt werden. Da sich der Koordinatenursprung des KS_{HL} zwischen den Augen des Headset-Trägers befindet, muss dieser nach der Positionsbestimmung der Marker noch exakt berechnet werden. Die hierfür verwendeten Abstände in X- und Z-Koordinaten sind in Abb. 6.4 dargestellt.

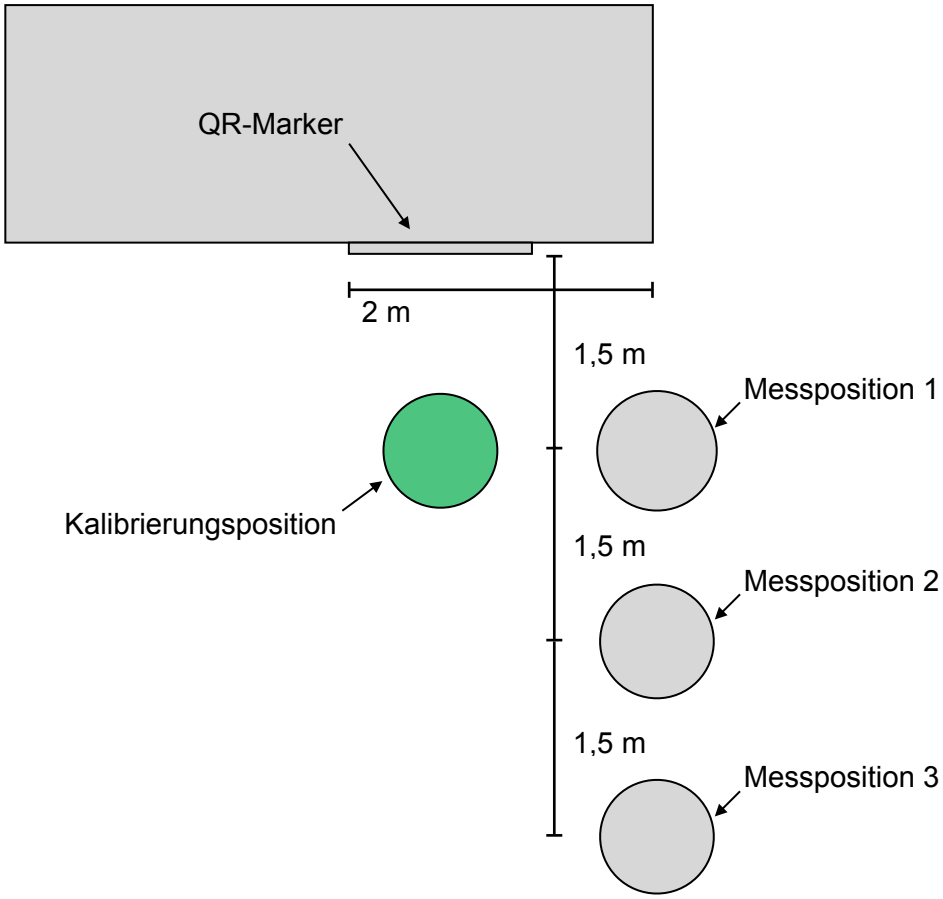


Abbildung 6.3.: Schematische Draufsicht der Messpunkte

Abbildung 6.4.: Lage des Ursprungs des KS_{HL} in Bezug zu den verwendeten Photogrammetriemarkern

6.1.4. Ergebnisse

Mit der beschriebenen Vorgehensweise zur Auswertung der Messergebnisse können die minimalen und maximalen Abweichungen sowie die Medianabweichungen bestimmt werden. Diese werden für jede Messposition und Koordinatenachse getrennt ausgewertet. Die Ergebnisse werden in Tabelle 6.1 aufgelistet und in Abb. 6.5 als Boxplot dargestellt.

Tabelle 6.1.: Ergebnisse der Abweichungsmessung (Werte in mm)

Position	Achse	Min.	Max.	Median
Pos. 1	X	–24,6	7,5	1,4
	Y	–10,8	19,0	12,8
	Z	–7,3	47,0	28,1
Pos. 2	X	–32,9	24,0	10,3
	Y	–64,3	14,4	–7,9
	Z	2,9	50,1	12,3
Pos. 3	X	–39,1	59,4	–20,0
	Y	–78,0	12,1	–33,3
	Z	34,9	70,5	48,5

Es fällt auf, dass mit zunehmender Entfernung vom für die Kalibrierung verwendeten QR-Marker auch die Schwankungen der Positionsabweichungen zunehmen. Dennoch liegt kein Messwert mehr als 78 mm entfernt von dem von der AR-Anwendung bestimmten Position. Vergleicht man die Median-Abweichungen, so bewegen sich diese allesamt im Bereich von ±50 mm.

Die durchgeführte Messreihe gibt keinen Aufschluss darüber, ob die gemessenen Abweichungen aus dem Kalibrierungsvorgang oder durch Fehler bei der SLAM-Positionsverfolgung entstehen. Um dies zu untersuchen, bedarf es einem Messverfahren, welches die HoloLens-Position bestimmen kann, während der Anwender diese auf dem Kopf trägt. Im Rahmen dieser Arbeit war es jedoch nicht möglich, dynamische Messungen der Realposition vorzunehmen, weshalb eine Vergleichsmessung nicht direkt nach der erfolgten Kalibrierung durchgeführt werden konnte. Aus diesem Grund war es ebenfalls nicht möglich, dynamische Zustände zu evaluieren und beispielsweise die Positionsdifferenz zu messen, während der Anwender sich im Raum bewegt. Die im Stand der Technik beschriebenen Messreihen zur Positionsgenauigkeit der SLAM-Lokalisierung der HoloLens berücksichtigen die Dynamik. Da deren Ergebnisse nicht signifikant von den ermittelten Werten abweichen lässt sich ableiten, dass auch bei einer dynamischen Messreihe keine siginikanten Vergrößerungen der Positionsabweichung zu erwarten wären.

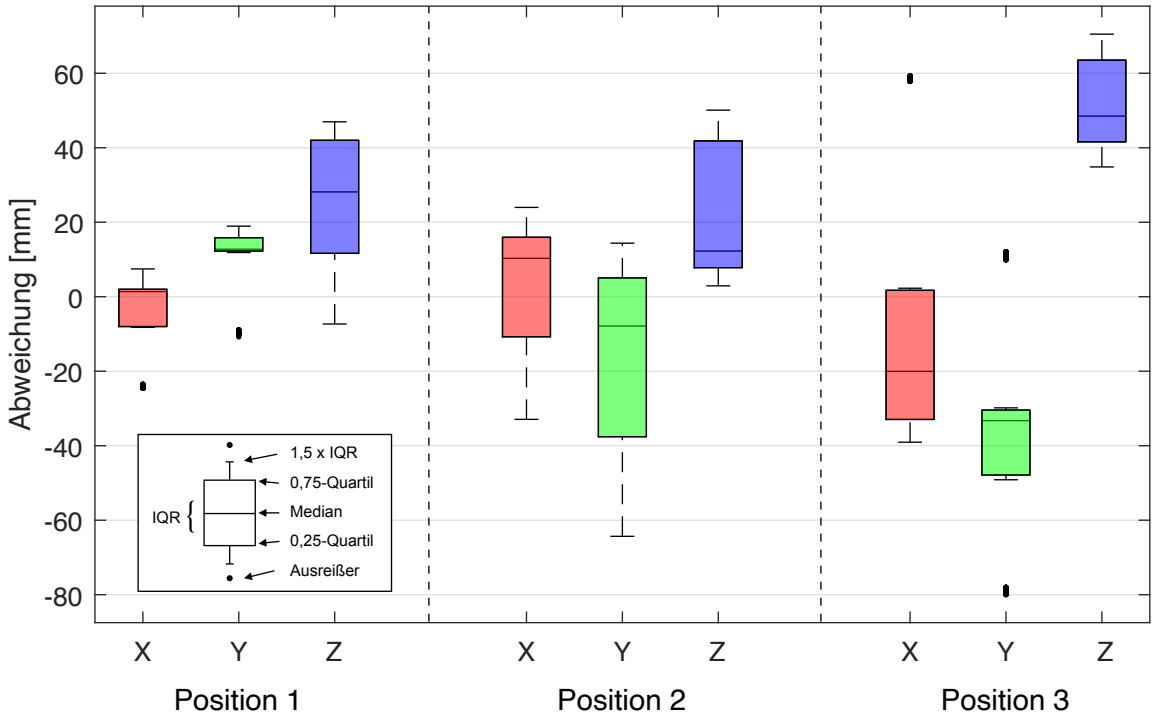


Abbildung 6.5.: Boxplot der Abweichung zwischen HoloLens-Positionierung und Referenzsystem

6.2. Anwendungsfall: Kollaborative Getriebemontage

Die entwickelte Anwendung zur Positionsbestimmung und zur Ermittlung von Prozesszeiten aus den Positionsdaten wird im Folgenden praktisch validiert. Als Beispiel dient die Montage eines Getriebes, welche kollaborativ von Mensch und Roboter durchgeführt wird. Der Mensch trägt dabei die HoloLens mit der entwickelten Anwendung. Durch die Analyse der Positionsdaten werden Prozesszeiten für die einzelnen menschlichen Arbeitsschritte bestimmt und diese gemeinsam mit vorab bestimmten Roboterprozesszeiten dazu verwendet, die Reihenfolge des Prozesses zu optimieren. Die einzelnen Getriebekomponenten sind in Abb. 6.6 dargestellt und benannt.

6.2.1. Prozessbeschreibung

Für die geometrische Prozessbeschreibung entsprechend der in Abschnitt 4.3.3 festgelegten Methodik werden zunächst Kontrollvolumen definiert. Für den gezeigten Prozess lassen sich drei markante Orte identifizieren, welche jeweils mit einem Kontrollvolumen überlagert werden:

- **KV-AB:** Der Arbeitsbereich an dem die Montage stattfindet.
- **KV-L1:** Das Lager für Getriebekomponenten.
- **KV-L2:** Das Lager für Schrauben und Werkzeuge.

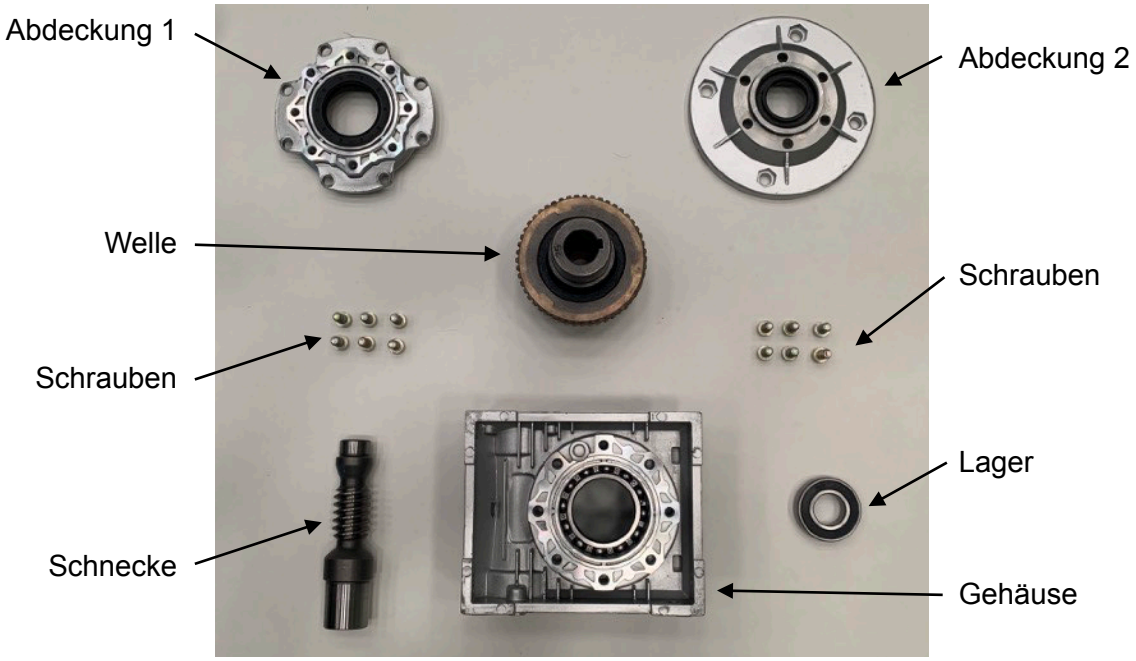


Abbildung 6.6.: Übersicht der zu montierenden Getriebe-Einzelteile

Abb. 6.7 zeigt die Position der Kontrollvolumen in der Draufsicht der Arbeitsumgebung. Ausgehend von den definierten Kontrollvolumen können die menschlichen Montageaufgaben definiert werden. Eine Montageaufgabe besteht dabei aus einem oder mehreren Prozessschritten, für die jeweils festgelegt wird, welche Kontrollvolumen-Interaktion des Arbeiters für das Starten oder Beenden des jeweiligen Schrittes entscheidend ist. Tabelle 6.2 listet die einzelnen Montageaufgaben, die zugehörigen Prozessschritte sowie die für die Prozesszeitbestimmung verwendeten Kontrollvolumen auf. Der Begriff „enter“ bezeichnet ein Kontrollvolumen, welches beim Betreten aktiviert wird und der Begriff „leave“ eines, dass beim Verlassen aktiviert wird. Im weiteren Verlauf des Textes werden die einzelnen Montageaufgaben mit den jeweiligen IDs von A bis H referenziert.

Zusätzlich erfolgt die Festlegung der vom Roboter durchgeführten Prozessschritte, für die die IDs 1 bis 9 vergeben werden. Bei den Prozessschritten 1 bis 4 handelt es sich um sogenannte Pick-and-Place-Aufgaben, also das Greifen und Platzieren von Komponenten, welche dann vom Arbeiter montiert werden. Um die Prozessoptimierung besser veranschaulichen zu können, werden zudem drei Zusatzaufgaben (5 bis 7) definiert, welche unabhängig von den menschlichen Aufgaben vom Roboter durchgeführt werden können. Im realen Produktionsumfeld könnten dies automatisierte Prüfprozesse oder Vormontage-Schritte sein. Die Schritte 8 und 9 sind jeweils Robotertätigkeiten, die das Verschrauben der zuvor montierten Getriebekomponenten veranschaulichen. Eine Übersicht der Roboterprozessschritte inklusive der gemessenen Prozesszeiten für diese finden sich in Tabelle 6.3. Da die einzelnen Roboterprozessschritte in unterschiedlichen Roboter-Posen starten und enden, müssen für die zeitliche Optimierung der Reihenfolge auch die Verfahrwege von einer Aufgabe zur nächsten berücksichtigt werden. Eine ausführliche Auflistung der ebenfalls gemessenen Verfahrzeiten findet sich in Anhang C.

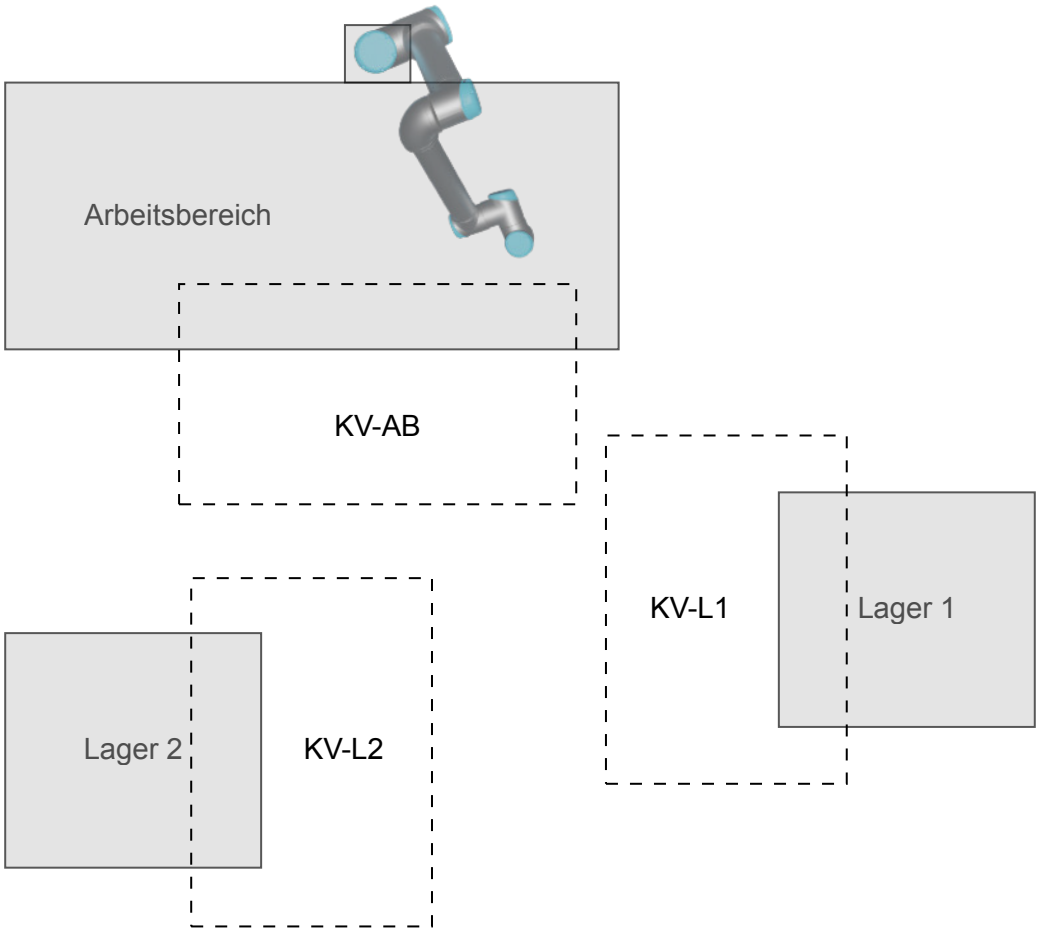


Abbildung 6.7.: Schematische Darstellung der definierten Kontrollvolumen (nicht maßstabsgetreu)

Tabelle 6.2.: Übersicht der Prozessschritte und der zugehörigen Kontrollvolumen inkl. Aktivierungsbedingung

ID	Aufgabe	Schritt	Start-KV	End-KV
A	Gehäuse holen	Laufweg Lager 1	AB - leave	L1 - enter
		Bauteil bringen	L1 - leave	AB - enter
B	Schrauben holen 1	Laufweg Lager 2	AB - leave	L2 - enter
		Schrauben bringen	L2 - leave	AB - enter
C	Montage Welle	Montage	AB - enter	AB - leave
D	Welle holen	Laufweg Lager 1	AB - leave	L1 - enter
		Welle bringen	L1 - leave	AB - enter
E	Schrauben holen 2	Laufweg Lager 2	AB - leave	L2 - enter
		Schrauben bringen	L2 - leave	AB - enter
F	Montage Schnecke	Montage	AB - enter	AB - leave
G	Verpackung holen	Laufweg Lager 2	AB - leave	L2 - enter
		Verpackung bringen	L2 - leave	AB - enter
H	Verpacken & Ablegen	Verpacken	AB - enter	AB - leave
		Ablegen	AB - leave	L2 - enter

Zuletzt werden Abhängigkeiten zwischen den einzelnen Roboter- und Arbeiteraufgaben definiert. Gemeint sind Prozessschritte, welche lediglich ausgeführt werden können, nachdem ein bestimmter anderer Schritt ausgeführt wurde. So kann beispielsweise die Verschraubung der Komponenten durch den Roboter (ID: 8 und 9) erst durchgeführt werden, nachdem der Arbeiter die entsprechenden Teile vormontiert hat. Eine vollständige Abhängigkeitsmatrix ist in Tabelle 6.4 abgebildet.

Die Prozessbeschreibung wird entsprechend dem in Abschnitt 4.3.3 festgelegten JSON-Schema notiert und auf dem Server gespeichert. Die vollständige JSON-Prozessbeschreibung findet sich in Anhang B.

Tabelle 6.3.: Roboterprozessschritte

ID	Bezeichnung	Dauer
1	Schnecke greifen	18,7 s
2	Lager greifen	20,4 s
3	Abdeckung 1 greifen	26,3 s
4	Abdeckung 2 greifen	22,4 s
5	Zusatzaufgabe 1	13,3 s
6	Zusatzaufgabe 2	11,1 s
7	Zusatzaufgabe 3	8,9 s
8	Verschrauben 1	38,1 s
9	Verschrauben 2	27,8 s

Tabelle 6.4.: Abhängigkeitsmatrix der Arbeiter- und Roboterprozessschritte

	ist abhängig von ...																
	A	B	C	D	E	F	G	H	1	2	3	4	5	6	7	8	9
A																	
B																	
C	■	■							■	■	■						
D																	
E																	
F				■	■							■					
G																	
H							■									■	■
1																	
2																	
3																	
4																	
5																	
6																	
7																	
8			■														
9						■											

6.2.2. Validierungsszenarien

Um den Prozess der Optimierung für verschiedene Prozesszeiten zu zeigen, werden insgesamt drei Szenarien vorgegeben:

- **Szenario 1:** Die Prozesszeiten werden mit der entwickelten Anwendung aufgezeichnet und ausgewertet. Hierbei wird der Durchschnittswert von mehreren Prozessdurchführungen verwendet. Die ermittelten Zeiten dienen als Referenzzeiten für die nachfolgenden Szenarien.
- **Szenario 2:** Die ermittelten Referenzzeiten werden jeweils um 30 % reduziert, was die Prozessdurchführung durch einen geübten bzw. schnellen Montagearbeiter veranschaulichen soll.
- **Szenario 3:** Die ermittelten Referenzzeiten werden jeweils um 30 % erhöht, was die Prozessdurchführung durch einen ungeübten bzw. langsamen Montagearbeiter veranschaulichen soll.

Anhand der gewählten Szenarien soll veranschaulicht werden, wie durch eine gezielte Erfassung der individuellen Prozesszeiten eines Montagearbeiters der Prozess so optimiert werden kann, dass die Gesamtdurchführungszeit minimal ist. Die Roboterprozesszeiten, welche ebenfalls in den Optimierungsprozess einfließen, sind für alle drei Szenarien gleich.

6.2.3. Datenerfassung und -auswertung

Die Prozessschrittzeiten für Szenario 1 werden durch die reale Durchführung der Getriebemontage mit der entwickelten Anwendung bestimmt. Abb. 6.8 zeigt Impressionen dieses Prozesses.

Die einzelnen manuellen Arbeitsschritte werden dabei in ungeordneter Reihenfolge unter Berücksichtigung von eventuellen Abhängigkeiten ausgeführt. Die HoloLens-Anwendung speichert die Roh-Positionsdaten und die ermittelten Prozesszeiten. Abb. 6.9 zeigt ein Spaghetti-Diagramm eines Montagedurchlaufs in dem der Laufweg des Arbeiters visualisiert wird. Die Ausführung wird insgesamt zehn Mal von einer Person wiederholt. Die Durchschnittswerte der auf diese Weise erfassten Prozesszeiten sind in Tabelle 6.5 aufgelistet.

Anschließend werden die Werte für die Szenarien 2 und 3 rechnerisch bestimmt. Die Ergebnisse sind ebenfalls in Tabelle 6.5 dokumentiert.

6.2.4. Prozessoptimierung

Nachdem die Prozesszeiten für die einzelnen menschlichen Arbeitsschritte mit der AR-Anwendung bestimmt wurden, können diese zusammen mit den gemessenen Roboterzeiten

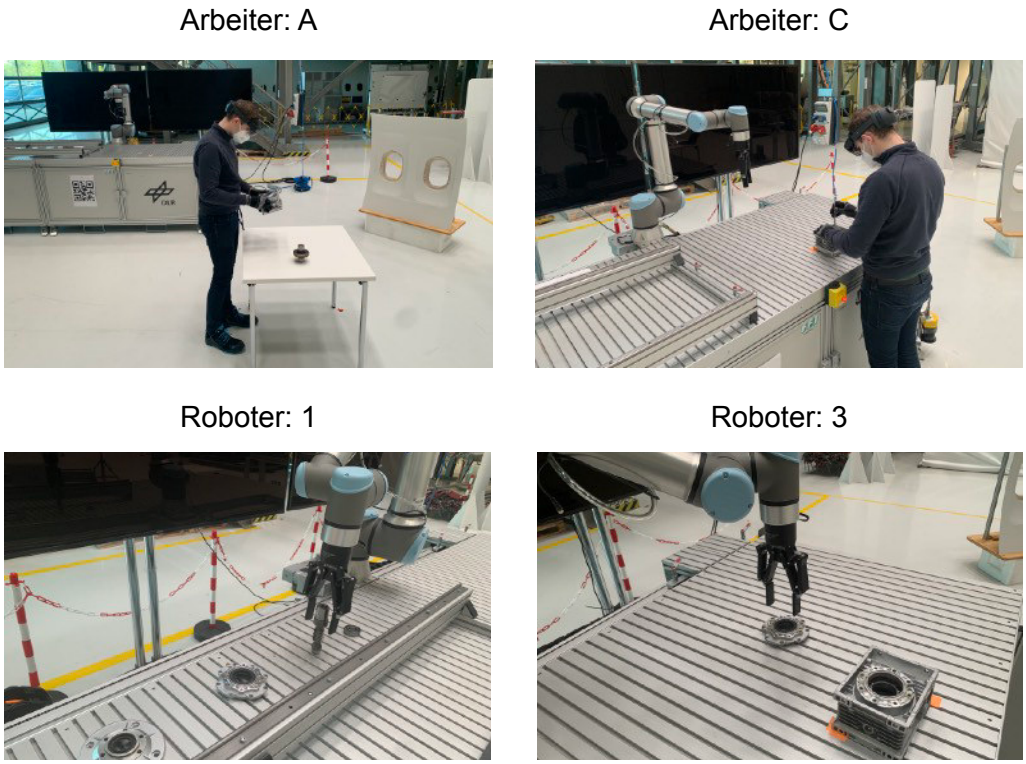


Abbildung 6.8.: Impressionen des Montageprozesses

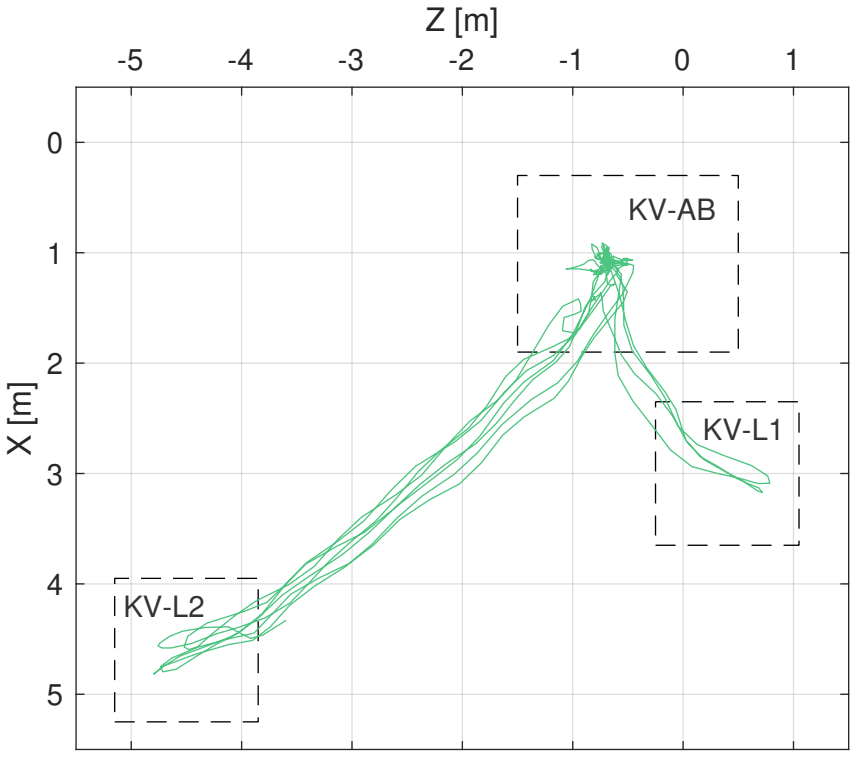


Abbildung 6.9.: Spaghetti-Diagramm der Arbeiterposition während eines Montagevorgangs

Tabelle 6.5.: Prozesszeiten für die drei Montageszenarien

Aufgabe	Szenario 1	Szenario 2	Szenario 3
A	23,0 s	17,7 s	28,3 s
B	13,7 s	10,5 s	16,8 s
C	99,6 s	76,6 s	122,6 s
D	6,2 s	4,8 s	7,7 s
E	14,0 s	10,8 s	17,3 s
F	74,4 s	57,2 s	91,5 s
G	12,5 s	9,6 s	15,4 s
H	24,8 s	19,1 s	30,6 s
Summe	268,2 s	206,3 s	330,2 s

als Grundlage für die Anwendung des Prozessoptimierungsalgorithmus verwendet werden (Details siehe Anhang D). Für die Optimierung kollaborativer Mensch-Roboter-Prozesse kommt eine Vielzahl an Optimierungsdimensionen in Frage. Für den gezeigten Anwendungsfall wurde die Minimierung der Gesamtprozesszeit gewählt. Durch eine Umstellung des Optimierungsalgorithmus wäre es jedoch auch möglich, Betriebszeiten und damit den Energieverbrauch des Roboters zu minimieren oder Varianten zu bevorzugen, die einen größtmöglichen Zeitpuffer für Aufgaben mit Fehlerpotenzial beinhalten.

Abb. 6.10 zeigt die für die drei Szenarien optimierte Prozessreihenfolge in schematischer Darstellung. Ein erwartbares Ergebnis hierbei ist die Tatsache, dass der Algorithmus insbesondere die Startzeit der unabhängig von den menschlichen Aufgaben durchführbaren Roboterprozessschritte 5,6 und 7 variiert. Wenn der Arbeiter länger für die Ausführung seiner Aufgaben braucht, dann werden diese Roboterprozessschritte parallel zu den menschlichen Aufgaben eingeplant, was den Gesamtprozess beschleunigt, ohne gleichzeitig Wartezeiten für den Arbeiter zu erzeugen.

An dieser Stelle ist anzumerken, dass es sich bei dem vorliegenden Optimierungsproblem um ein vergleichsweise einfach zu lösendes Problem handelt, für das es mehrere optimale Lösungen gibt. Aus diesem Grund wird beispielsweise die Reihenfolge bestimmter Prozessschritte quasi zufällig gewählt. Durch das Hinzufügen weiterer Optimierungsziele wie beispielsweise einer Minimierung der Roboterbewegungen oder das Einbeziehen einer größeren Anzahl an Prozessschritten erhöht sich die Optimierungskomplexität deutlich, sodass die optimale Lösung nicht mehr in ausreichend kurzer Zeit bestimmt werden kann und auf Näherungslösungen zurückgegriffen werden muss.

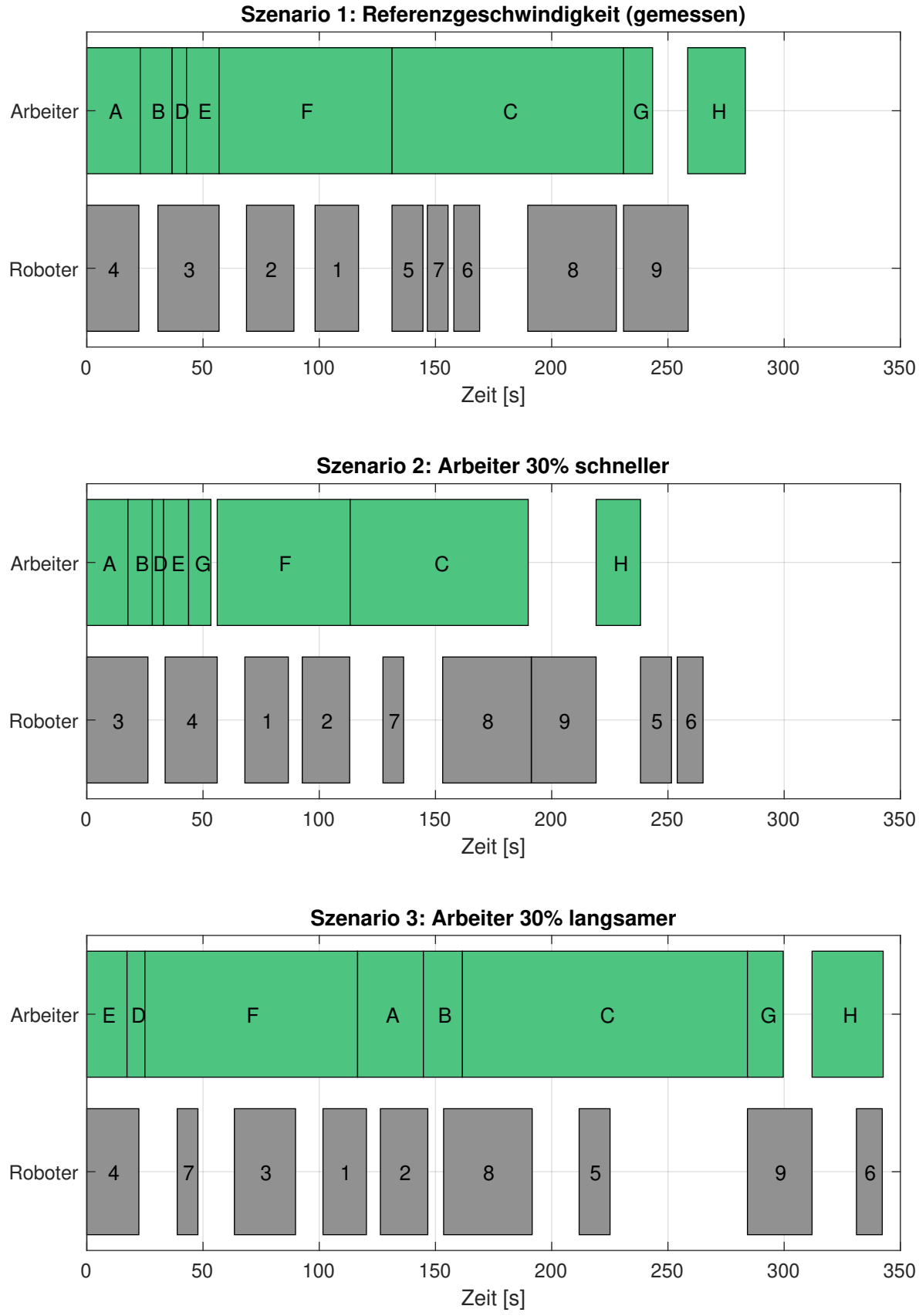


Abbildung 6.10.: Schematische Darstellung der optimierten Prozessreihenfolgen

6.2.5. Erfahrungen und Beobachtungen

Es lässt sich festhalten, dass das Tragen des AR-Headset den Ausführenden nicht in seiner Bewegungsfreiheit einschränkt. Alle Laufwege und Handbewegungen lassen sich in gleicher Art und Weise wie ohne das Headset ausführen. Auch der Tragekomfort lässt sich als angenehm beschreiben und ist vergleichbar mit dem Tragen eines Schutzhelms. Wie der empfundene Komfort sich aber über lange Zeiträume z. B. über eine Schicht von acht Stunden hinweg entwickelt, lässt sich aus dem untersuchten Anwendungsfall nicht realistisch ableiten.

Weitaus größer ist die Einschränkung des Blickfeldes des Headset-Trägers. Diese ergibt sich vor allem durch die verdunkelnde Scheibe vor der Displayeinheit, die die HoloLens zur Verstärkung des Hologramm-Kontrastes bei der Darstellung von AR-Inhalten besitzt. Auffällig ist, dass der Träger der HoloLens im dargestellten Montage-Szenario die Tendenz entwickelt, unten an der Display-Einheit des Geräts vorbeizuschauen. Dies könnte daran liegen, dass bei der entwickelten Anwendung keine für die Montage notwendigen Informationen auf dem AR-Display dargestellt werden und somit unterbewusst die ungehinderte Blicklinie auf das Montageobjekt gesucht wird.

7 Diskussion

Nach der erfolgten Validierung werden im folgenden Abschnitt die Ergebnisse bewertet und anschließend diskutiert, ob eine Anwendung des entwickelten Positionsbestimmungssystems im industriellen Kontext möglich wäre.

7.1. Bewertung der Ergebnisse

7.1.1. Aufwand und Präzision der Positionsbestimmung

Der Aufwand für die Implementierung des entwickelten HoloLens-basierten IPS lässt sich als eher gering bezeichnen. Um die Position eines Menschen in einem Raum zu bestimmen reicht es aus, einen gedruckten QR-Marker zu platzieren und dessen Abstand und Orientierung relativ zu einem selbst gewählten raumfesten Koordinatensystem zu messen. Weitere ortsgebundene Infrastruktur ist nicht notwendig. Da der Kalibrierungsprozess automatisiert stattfindet, ist bereits wenige Sekunden nach dem Start des AR-Headsets eine Positionsbestimmung möglich. Indem mehrere QR-Marker als Koordinatenanker definiert werden, ist es zudem möglich, mit einem einzelnen Gerät eine individuelle Positionsbestimmung in unterschiedlichen Räumen oder Teilen einer Produktionsanlage durchzuführen.

Mit Hilfe der durchgeführten photogrammetrischen Messreihe wurden mittlere Abweichungen bei der Positionsbestimmung im Bereich von etwa –35 mm bis 50 mm mit Ausreißern von ca. –80 mm bis 70 mm festgestellt. Berücksichtigt wurde dabei eine Entfernung von bis zu 5 m vom für die Kalibrierung verwendeten Marker. Das Ergebnis liegt damit innerhalb der in Abschnitt 3.2.4 aus Literaturwerten abgeleiteten Erwartungen. Für den in dieser Arbeit gezeigten Anwendungsfall der Prozesszeitbestimmung stellte sich heraus, dass die erreichte Genauigkeit ausreichend ist um verlässlich zu detektieren, ob der Arbeiter sich innerhalb eines Kontrollvolumens befindet.

7.1.2. Vergleich mit anderen Systemen zur Positionsbestimmung

In Kapiten 3 wurden unterschiedliche am Markt verfügbare Systeme zur Positionsbestimmung in Innenräumen vorgestellt. Bezogen auf die Anschaffungskosten und die erreichbare Präzision bewegen sich vor allem UWB-basierte und markerlose Motion Capture-Systeme in einem Bereich mit der HoloLens-Positionsbestimmung. Mit der durchgeführten Messreihe konnte bestätigt werden, dass die Präzision in etwa gleich oder sogar höher als die der genannten Alternativen ist. Der Aufwand für die Implementierung der AR-Positionsbestimmung

ist als deutlich geringer zu bewerten, was vor allem daran liegt, dass keine ortsfeste Infrastruktur notwendig ist.

7.1.3. Berechnung von Prozesszeiten aus Positionsdaten

Zuletzt wird der gezeigte Anwendungsfall, die Berechnung von Prozesszeiten aus den von der HoloLens erfassten Positionsdaten bewertet. Im Rahmen der praktischen Erprobung wurde festgestellt, dass die Bestimmung von Prozesszeiten mit der entwickelten Anwendung zuverlässig und genau ist, vorausgesetzt, der Arbeiter folgt exakt der vorgegebenen Prozessreihenfolge und achtet darauf, Arbeitsschritte in den vorgegebenen Kollisionsvolumen auszuführen. Dies stellt die größte Limitierung des implementierten Ansatzes dar, denn es ist wahrscheinlich, dass in einem realen industriellen Kontext auftretende Abweichungen von diesen Vorgaben, die Berechnung der Prozesszeiten negativ beeinflussen. Abhilfe könnte hierbei die Einbeziehung weiterer Daten wie beispielsweise die von der in der HoloLens verbauten Kamera kontinuierlich erfassten Bilddaten sein. Diese könnten dazu verwendet werden, um Tätigkeiten zu identifizieren und so eine korrekte Zuordnung der Zeiten zu ermöglichen. Ein weiterer vielversprechender Ansatz ist die Anwendung von Machine Learning Algorithmen um automatisiert wiederkehrende Muster in den aufgezeichneten Daten zu erkennen wodurch einzelne Prozessschritte identifiziert werden können.

7.2. Bewertung der industriellen Anwendbarkeit

Abschließend wird die industrielle Anwendbarkeit eines AR-basierten Systems zur Bestimmung von Arbeiterpositionen, wie es in der vorliegenden Arbeit entwickelt wurde, bewertet. Einbezogen in die Bewertung werden sowohl die Ergebnisse der Validierung als auch Literaturquellen.

7.2.1. Technische Bewertung

Im Rahmen dieser Arbeit wurde gezeigt, dass die Microsoft HoloLens 2 es ermöglicht, präzise die Kopfposition eines Arbeiters während einer Montageaufgabe zu erfassen. Gegenüber anderen Systemen zur Positionsbestimmung in Innenräumen besteht der Vorteil, dass auf ortsfeste Infrasturktur weitgehend verzichtet und das Gerät flexibel eingesetzt werden kann. Durch die Netzwerkkonektivität der HoloLens war es ohne Probleme möglich, erfasste Daten an eine zentrale Servereinheit zu übertragen, um diese dort auswerten zu können.

Fraglich jedoch ist, ob die HoloLens 2 zum jetzigen Entwicklungsstand dauerhaft im Produktionsumfeld eingesetzt werden kann. Dagegen spricht zum einen die recht geringe Akkulaufzeit, welche vom Hersteller mit 2 bis 3 Stunden angegeben wird [52], im realen Einsatz jedoch noch einmal niedriger ausfallen dürfte. Zum anderen ist ungeklärt, ob auf der

HoloLens ausgeführte Software robust genug für den dauerhaften Einsatz ist oder mit Ausfallzeiten durch Systemabstürze gerechnet werden muss. Insbesondere für Anwendungsfälle bei denen die zuverlässige Positionsbestimmung kritisch für die Sicherheit des Arbeiters ist, würde ein Softwareabsturz zu einem Produktionsstillstand führen.

7.2.2. Skalierbarkeit und Flexibilität

Wichtige Aspekte für industriell eingesetzte Systeme ist die Skalierbarkeit und Flexibilität dieser, insbesondere bei sich ändernden Produktionsbedingungen. Insbesondere die Flexibilität spricht für den Einsatz der entwickelten HoloLens-Anwendung zur Positionsbestimmung. So können neue Produktionsbereiche innerhalb weniger Minuten für die Verfolgung von Arbeiterpositionen vorbereitet werden. Weniger vorteilhaft im Vergleich zu anderen IPS ist die Skalierbarkeit der AR-Anwendung. Erhöht sich die Anzahl der Personen für die die Position dauerhaft bestimmt werden soll, so muss für jede ein zusätzliches AR-Headset angeschafft werden. Deutlich einfacher und preisgünstiger verhält es sich hierbei z. B. bei UWB-Positionierungssystemen. Die ortsfeste Infrastruktur erweist sich hier als Vorteil, da durch Sie die Anzahl der verfolgten Personen in einem gewissen Maße beliebig erweitert werden kann. Notwendig ist hierfür lediglich die Anschaffung und Einrichtung weiterer UWB-Empfänger. Diese Tatsache bietet insbesondere dann einen großen Vorteil, wenn auch Objekte verfolgt werden sollen.

7.2.3. Ergonomie

Eine wichtige Frage zur Bewertung der industriellen Anwendbarkeit des HoloLens-Positionierungssystems ist, ob der Arbeiter zusätzliche Belastungen durch das Tragen des Headsets erfährt oder in seiner Arbeit eingeschränkt wird. In der praktischen Erprobung hat sich gezeigt, dass die Bewegungsfreiheit des HoloLens-Trägers nicht beeinflusst wird, die Verdunklung des Sichtfeldes jedoch als störend empfunden wird. Zu klären ist die Frage, ob diese Einschränkungen der Sicht auch Auswirkungen auf die Arbeitssicherheit haben können, was einen großflächigen Einsatz deutlich erschweren würde.

7.2.4. Fazit und weitere Aspekte

Zusammenfassend lässt sich sagen, dass sich die HoloLens durchaus als präzises und leistungsfähiges IPS erwiesen hat, dessen Kernparameter für eine industrielle Anwendbarkeit geeignet wären. Der mögliche Einsatzhorizont wird jedoch durch die Limitierung bei der Skalierbarkeit und die technischen Rahmenbedingungen eingeschränkt. Vorerst scheint demnach der Einsatz der HoloLens zur Erfassung menschlicher Positionsdaten vor allem

dort vielversprechend, wo die HoloLens bereits andere Anwendungsfälle wie beispielsweise die Darstellung von Montagehinweisen übernimmt. In solchen Szenarien könnte die Anwendung zur Positionsbestimmung zusätzlich im Hintergrund eingesetzt werden und so ohne Mitwirkung des Arbeiters für die Optimierung von Prozessen nutzbare Daten sammeln.

8 Zusammenfassung und Ausblick

Um sich wandelnden Marktanforderungen zu begegnen, werden Produktionsanlagen immer stärker digitalisiert und automatisiert. Unter dem Begriff Industrie 4.0 wurde die Vision eines sich selbst steuernden und optimierenden Produktionssystems zusammengefasst, welches aus cyber-physischen Systemen besteht. Ein zentraler Punkt für den Erfolg dieses Ansatzes ist die Verfügbarkeit von Daten. Da auch zukünftig menschliche Arbeiter ein wichtiger Bestandteil von Produktionsprozessen sein werden, besteht die Herausforderung darin, auch den Menschen und mit ihm verknüpfte Daten für die Optimierung von Prozessen nutzbar zu machen. Da der Mensch naturgemäß über keine digitalen Schnittstellen verfügt, ist die Schaffung solcher Schnittstellen mit großem Aufwand verbunden und erfordert in der Regel eine komplexe technische Infrastruktur. Im Rahmen dieser Arbeit wurde untersucht, ob sich der Aufwand für die Erfassung menschlicher Daten durch den Einsatz eines AR-Headsets minimieren lässt. Grundlage hierfür ist die Eigenschaft von AR-Systemen, für die virtuelle Darstellung von Inhalten mit verbauten Sensoren dauerhaft eine Vielzahl an Daten des Anwenders und seiner Umgebung zu erfassen. Ziel der Arbeit war, am Beispiel der kontinuierlichen Erfassung der Arbeiterposition im Raum zu zeigen, wie die vom AR-Headset erfassten Sensordaten systematisch genutzt werden können, um menschliche Prozessparameter wie Prozess- und Transferzeiten zu bestimmen.

Im Rahmen der Arbeit wurde eine AR-Anwendung für die Microsoft HoloLens 2 entwickelt, mit der es möglich ist, die Position eines Arbeiters im Raum dauerhaft während eines manuellen Arbeitsprozesses zu bestimmen. Kern der Anwendung ist die Implementierung eines Kalibrierungsprozesses, welcher es ermöglicht, Positionsdaten in raumfesten Koordinaten zu erfassen. Grundlage der Kalibrierung bildet ein im Raum angebrachter QR-Marker dessen vermessene Position und Lage Ausgangspunkt für die nötige Koordinatentransformation ist. Der erwartete Vorteil, auf ortsfeste Infrastruktur weitgehend verzichten zu können, hat sich im Rahmen der praktischen Validierung bestätigt. Zusätzlich wurde im Rahmen einer Messreihe die Präzision der Positionsbestimmung untersucht. Die ermittelten Positionsabweichungen bewegten sich in einem Bereich von weniger als 100 mm und liegt damit in der Größenordnung anderer Positionierungssysteme mit raumgebundener Infrastruktur. Während die breite industrielle Anwendbarkeit der HoloLens als reines Lokalisierungssystem aus technischer Sicht eher unwahrscheinlich ist, kommt ein Einsatz der entwickelten Anwendung insbesondere dort infrage, wo AR-Anwendungen im Produktionsprozess für andere Anwendungsfälle eingesetzt werden. In solchen Szenarien könnten ohne großen Aufwand zusätzliche Daten generiert werden, welche dann zu Optimierungszwecken ausgewertet werden können.

Ebenfalls entwickelt wurde eine serverbasierte Anwendung zur automatisierten Analyse der erfassten Positionsdaten. Es wurde beispielhaft gezeigt, wie aus den Arbeiter-Positionsdaten

Durchführungszeiten einzelner Prozessschritte bestimmt werden können, welche dann wiederum für die Optimierung der Prozessreihenfolge in kollaborativen Mensch-Roboter-Prozessen verwendet werden können. Zur Berechnung der Prozesszeiten wurde ein Analyseprozess, basierend auf einer vorher festgelegten geometrischen Prozessbeschreibung entwickelt. Diese Prozessbeschreibung basiert auf Kontrollvolumen, welche es ermöglichen, Anfang und Ende eines Prozessschrittes durch das Betreten oder Verlassen eines Kontrollvolumens zu definieren. Die Analyse der Daten wurde praktisch anhand eines beispielhaften, kollaborativ von Mensch und Roboter ausgeführten, Montageprozesses eines Getriebes validiert. Es konnte gezeigt werden, wie mit wenigen vorbereitenden Schritten und ohne aufwendige ortsfeste Infrastruktur, aussagekräftige Daten zur menschlichen Arbeit erfasst werden können. Mit den so bestimmten Prozesszeiten konnte gezeigt werden, wie die Gesamtprozessdauer durch die Anpassung der Prozessreihenfolge optimiert werden kann. Eine Einschränkung des gewählten Ansatzes ist die Notwendigkeit, dass für die Bestimmung der Prozesszeiten eine präzise geometrische Beschreibung des Prozesses und der Reihenfolge mit Kontrollvolumen notwendig ist. Zwar ermöglicht dies eine einfache und genaue Berechnung der Prozesszeiten, erfordert aber auch, dass dieser vorgegebene Prozess genau eingehalten wird.

Das entwickelte System lässt sich in vielfältiger Art und Weise erweitern, so wäre es beispielsweise denkbar, den für die Positionsbestimmung notwendigen Kalibrierungsprozess noch stärker zu automatisieren, sodass diese unabhängig vom Nutzer kontinuierlich stattfindet. Aufseiten der Datenauswertung ist es sinnvoll, auch die zeitversetzte Analyse von ganzen Positionssequenzen zu erforschen, um die Bestimmung von Prozesszeiten weniger anfällig für Fehler zu machen oder über die reine Prozesszeitbestimmung hinausgehende Anwendungsfälle zu untersuchen. Die genannte Einschränkung des in dieser Arbeit entwickelten Systems, vorher eine geometrische Prozessbeschreibung erstellen zu müssen, könnte abgeschwächt werden, indem diese automatisiert generiert wird. Ein Ansatz hierfür könnte beispielsweise durch die Nutzung des AutomationML-Formats [110] sein, um eine Prozessbeschreibung aus digitalen Simulationsdaten abzuleiten. Interessant wäre zudem die wissenschaftliche Auseinandersetzung mit der Frage, ob durch den Einsatz von Machine Learning Algorithmen menschliche Tätigkeiten auch ohne die vorherige Festlegung von Prozessreihenfolgen und Arbeitsbereichen aus den Positionsdaten eines Arbeiters erkannt werden können.

Ein weiterer Anknüpfungspunkt ist die Erschließung weiterer passiver menschlicher Datenquellen mit den in der HoloLens verbauten Sensoren. So ist es denkbar, auch Daten zur Körperhaltung durch die systematische Auswertung von Handpositionen und Kopfhaltung mit der HoloLens zu erfassen. Zudem ließen sich durch die verfügbaren Daten zur Blickrichtung des Anwenders Daten zur menschlichen Intention und Aufmerksamkeit ableiten. Zuletzt könnte durch eine gezielte Analyse der durch die HoloLens aufgenommenen Kamerabilder auch Objekte erkannt werden und so beispielsweise Materialflüsse nachvollzogen werden.

V Literatur

[1] Thomas Bauernhansl. „Die Vierte Industrielle Revolution – Der Weg in ein wertschaffendes Produktionsparadigma“. In: *Industrie 4.0 in Produktion, Automatisierung und Logistik*. Springer Fachmedien Wiesbaden, 2014, S. 5–35. DOI: 10.1007/978-3-658-04682-8_1.

[2] Holger Kreitling. „Legende und Wahrheit: Henry Ford und sein Fließband“. de. In: *DIE WELT* (Juni 2011). URL: https://www.welt.de/print/wams/vermischtes/article13412755/Henry-Ford-und-sein-Fliessband.html (besucht am 07. 03. 2022).

[3] Bundesverband der Deutschen Industrie e.V. *Gestern war Industrie 4.0 noch Zukunft, heute ist es Realität - Einblick in die vierte Revolution*. de. URL: http://www.bdi.eu/leben-4.0/innovation (besucht am 07. 03. 2022).

[4] WFB. *Was ist Industrie 4.0? Die Definition von Digitalisierung und Industrie 4.0*. de. URL: https://www.wfb-bremen.de/de/page/stories/digitalisierung-industrie40/was-ist-industrie-40-eine-kurze-erklaerung (besucht am 07. 03. 2022).

[5] SAP. *Was ist das Internet der Dinge (Internet of Things; IoT)?* German. URL: https://www.sap.com/germany/insights/what-is-iot-internet-of-things.html (besucht am 07. 03. 2022).

[6] Dieter Spath, Hrsg. *Produktionsarbeit der Zukunft - Industrie 4.0*. Stuttgart: Fraunhofer Verlag, 2013. ISBN: 978-3-83960-570-7.

[7] Susanne Oberer-Treitz und Alexander Verl. „Einführung in die industrielle Robotik mit Mensch-Roboter-Kooperation“. In: *Handbuch Mensch-Roboter-Kollaboration*. Carl Hanser Verlag GmbH & Co. KG, Jan. 2019, S. 1–35. DOI: 10.3139/9783446453760.001.

[8] Universal Robots. *UR10e – der flexible Industrieroboter von Universal Robots*. URL: https://www.universal-robots.com/de/produkte/ur10-roboter/ (besucht am 08. 03. 2022).

[9] Francesco De Pace u. a. „A systematic review of Augmented Reality interfaces for collaborative industrial robots“. In: *Computers & Industrial Engineering* 149 (Nov. 2020), S. 106806. DOI: 10.1016/j.cie.2020.106806.

[10] Pedro Neto u. a. „Gesture-based human-robot interaction for human assistance in manufacturing“. In: *The International Journal of Advanced Manufacturing Technology* 101.1-4 (Okt. 2018), S. 119–135. DOI: 10.1007/s00170-018-2788-x.

[11] Alberto Poncela und Leticia Gallardo-Estrella. „Command-based voice teleoperation of a mobile robot via a human-robot interface“. In: *Robotica* 33.1 (Jan. 2014), S. 1–18. DOI: 10.1017/s0263574714000010.

[12] Juraj Slovák u. a. „Vision and RTLS Safety Implementation in an Experimental Human—Robot Collaboration Scenario“. In: *Sensors* 21.7 (Apr. 2021), S. 2419. DOI: 10.3390/s21072419.

[13] Jiwoong Lim u. a. „Designing Path of Collision Avoidance for Mobile Manipulator in Worker Safety Monitoring System Using Reinforcement Learning“. In: *2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*. IEEE, März 2021. DOI: 10.1109/isr50024.2021.9419504.

[14] Wei Fang und Zewu An. „A scalable wearable AR system for manual order picking based on warehouse floor-related navigation“. In: *The International Journal of Advanced Manufacturing Technology* 109.7-8 (Juli 2020), S. 2023–2037. DOI: 10.1007/s00170-020-05771-3.

[15] Bo-Chen Huang u. a. „ARBIN: Augmented Reality Based Indoor Navigation System“. In: *Sensors* 20.20 (Okt. 2020), S. 5890. DOI: 10.3390/s20205890.

[16] Terry L. Jones und Cara Schlegel. „Can Real Time Location System Technology (RTLS) Provide Useful Estimates of Time Use by Nursing Personnel?“ In: *Research in Nursing & Health* 37.1 (Dez. 2013), S. 75–84. DOI: 10.1002/nur.21578.

[17] Abdelmoumen Norrdine u. a. „MQTT-Based Surveillance System of IoT Using UWB Real Time Location System“. In: *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. IEEE, Nov. 2020. DOI: 10.1109/ithings-greencom-cpscom-smartdata-cybermatics50389.2020.00050.

[18] Tao Cheng u. a. „Automated task-level activity analysis through fusion of real time location sensors and worker’s thoracic posture data“. In: *Automation in Construction* 29 (Jan. 2013), S. 24–39. DOI: 10.1016/j.autcon.2012.08.003.

[19] Maurizio Faccio u. a. „Human Factor Analyser for work measurement of manual manufacturing and assembly processes“. In: *The International Journal of Advanced Manufacturing Technology* 103.1-4 (März 2019), S. 861–877. DOI: 10.1007/s00170-019-03570-z.

[20] Francesco Caputo u. a. „A Human Postures Inertial Tracking System for Ergonomic Assessments“. In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, Aug. 2018, S. 173–184. DOI: 10.1007/978-3-319-96068-5_19.

[21] Jörg Krüger und The Duy Nguyen. „Automated vision-based live ergonomics analysis in assembly operations“. In: *CIRP Annals* 64.1 (2015), S. 9–12. DOI: 10.1016/j.cirp.2015.04.046.

[22] Christoph Petzoldt u. a. „Functionalities and Implementation of Future Informational Assistance Systems for Manual Assembly“. In: *Communications in Computer and Information Science*. Springer International Publishing, 2020, S. 88–109. DOI: 10.1007/978-3-030-64351-5_7.

[23] Irio De Feudis u. a. „Evaluation of Vision-Based Hand Tool Tracking Methods for Quality Assessment and Training in Human-Centered Industry 4.0“. In: *Applied Sciences* 12.4 (Feb. 2022), S. 1796. DOI: 10.3390/app12041796.

[24] Hugo Nascimento, Martin Mujica und Mourad Benoussaad. „Collision Avoidance in Human-Robot Interaction Using Kinect Vision System Combined With Robot's Model and Data“. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Okt. 2020. DOI: 10.1109/iros45743.2020.9341248.

[25] Photchara Ratasamee u. a. „Human–Robot Collision Avoidance Using A Modified Social Force Model With Body Pose And Face Orientation“. In: *International Journal of Humanoid Robotics* 10.01 (März 2013), S. 1350008. DOI: 10.1142/s0219843613500084.

[26] Maurizio Faccio u. a. „Real-time assistance to manual assembly through depth camera and visual feedback“. In: *Procedia CIRP* 81 (2019), S. 1254–1259. DOI: 10.1016/j.procir.2019.03.303.

[27] AJung Moon u. a. „Meet Me where I'm Gazing: How Shared Attention Gaze Affects Human-Robot Handover Timing“. In: *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*. ACM, März 2014. DOI: 10.1145/2559636.2559656.

[28] Ravi Teja Chadalavada u. a. „Bi-directional navigation intent communication using spatial augmented reality and eye-tracking glasses for improved safety in human–robot interaction“. In: *Robotics and Computer-Integrated Manufacturing* 61 (Feb. 2020), S. 101830. DOI: 10.1016/j.rcim.2019.101830.

[29] Sander Mathijn Spook u. a. „Implementing sensor technology applications for workplace health promotion: a needs assessment among workers with physically demanding work“. In: *BMC Public Health* 19.1 (Aug. 2019). DOI: 10.1186/s12889-019-7364-2.

[30] Fan Wu, Taiyang Wu und Mehmet Rasit Yuce. „Design and Implementation of a Wearable Sensor Network System for IoT-Connected Safety and Health Applications“. In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE, Apr. 2019. DOI: 10.1109/wf-iot.2019.8767280.

[31] Hanne Austad u. a. „Towards a wearable sensor system for continuous occupational cold stress assessment“. In: *Industrial Health* 56.3 (2018), S. 228–240. DOI: 10.2486/indhealth.2017-0162.

[32] Geb Thomas u. a. „Low-Cost, Distributed Environmental Monitors for Factory Worker Health“. In: *Sensors* 18.5 (Mai 2018), S. 1411. DOI: 10.3390/s18051411.

[33] Ralf Dörner u. a. „Einführung in Virtual und Augmented Reality“. In: *Virtual und Augmented Reality (VR/AR)*. Springer Berlin Heidelberg, 2019, S. 1–42. DOI: 10.1007/978-3-662-58861-1_1.

[34] Paul Milgram u. a. „Augmented reality: a class of displays on the reality-virtuality continuum“. In: *SPIE Proceedings*. Hrsg. von Hari Das. SPIE, Dez. 1995. DOI: 10.1117/12.197321.

[35] Paul Grimm u. a. „VR/AR-Ausgabegeräte“. In: *Virtual und Augmented Reality (VR/AR)*. Springer Berlin Heidelberg, 2019, S. 163–217. DOI: 10.1007/978-3-662-58861-1_5.

[36] Wolfgang Broll. „Augmentierte Realität“. In: *Virtual und Augmented Reality (VR/AR)*. Springer Berlin Heidelberg, 2019, S. 315–356. DOI: 10.1007/978-3-662-58861-1_8.

[37] Paul Grimm u. a. „VR/AR-Eingabegeräte und Tracking“. In: *Virtual und Augmented Reality (VR/AR)*. Springer Berlin Heidelberg, 2019, S. 117–162. DOI: 10.1007/978-3-662-58861-1_4.

[38] Apple. *Augmented Reality*. de-DE. URL: https://www.apple.com/de/augmented-reality/ (besucht am 13. 03. 2022).

[39] Microsoft. *Microsoft Mixed Reality-/AR-Guides | Microsoft Dynamics 365*. de. URL: https://dynamics.microsoft.com/de-de/mixed-reality/guides/ (besucht am 28. 04. 2022).

[40] Heinrich Gotzig und Georg Otto Geduld. „LIDAR-Sensorik“. In: *Handbuch Fahrerassistenzsysteme*. Springer Fachmedien Wiesbaden, 2015, S. 317–334. DOI: 10.1007/978-3-658-05734-3_18.

[41] Ralf Dörner und Frank Steinicke. „Wahrnehmungsaspekte von VR“. In: *Virtual und Augmented Reality (VR/AR)*. Springer Berlin Heidelberg, 2019, S. 43–78. DOI: 10.1007/978-3-662-58861-1_2.

[42] Microsoft. *HoloLens 2 – Übersicht, Funktionen und Spezifikationen*. de-de. URL: https://www.microsoft.com/de-de/hololens/hardware (besucht am 05. 02. 2022).

[43] Magic Leap. *Magic Leap 1*. en-us. URL: https://www.magicleap.com/magic-leap-1 (besucht am 13. 03. 2022).

[44] Eleonora Bottani und Giuseppe Vignali. „Augmented reality technology in the manufacturing industry: A review of the last decade“. In: *IISE Transactions* 51.3 (Feb. 2019), S. 284–310. DOI: 10.1080/24725854.2018.1493244.

[45] X. Wang, S. K. Ong und A. Y. C. Nee. „A comprehensive survey of augmented reality assembly research“. In: *Advances in Manufacturing* 4.1 (Jan. 2016), S. 1–22. DOI: 10.1007/s40436-015-0131-4.

[46] Riccardo Palmarini u. a. „A systematic review of augmented reality applications in maintenance“. In: *Robotics and Computer-Integrated Manufacturing* 49 (Feb. 2018), S. 215–228. DOI: 10.1016/j.rcim.2017.06.002.

[47] Dimitris Mourtzis, Vasilios Zogopoulos und Ekaterini Vlachou. „Augmented Reality supported Product Design towards Industry 4.0: a Teaching Factory paradigm“. In: *Procedia Manufacturing* 23 (2018), S. 207–212. DOI: 10.1016/j.promfg.2018.04.018.

[48] Kognitiv Spark. *Enabling Expert Guidance With Mixed Reality*. en-CA. URL: https://www.kognitivspark.com (besucht am 16. 03. 2022).

[49] Lorne Fade. *Forbes Business Council: The Benefits Of Augmented Reality For Employee Training*. en. URL: https://www.forbes.com/sites/forbesbusinesscouncil/2021/02/12/the-benefits-of-augmented-reality-for-employee-training/ (besucht am 16. 03. 2022).

[50] Heather Kelly. *Microsoft's new $3,500 HoloLens 2 headset means business*. Feb. 2019. URL: https://www.cnn.com/2019/02/24/tech/microsoft-hololens-2/index.html (besucht am 05. 02. 2022).

[51] Microsoft. *HoloLens 2 – Preise und Optionen*. de-de. URL: https://www.microsoft.com/de-de/hololens/buy (besucht am 05. 02. 2022).

[52] Microsoft Docs. *HoloLens 2 - Hardware*. de-de. URL: https://www.microsoft.com/de-de/hololens/hardware (besucht am 05. 02. 2022).

[53] Microsoft Docs. *Funktionsweise von Inside-Out-Tracking - Enthusiast Guide*. de-de. URL: https://docs.microsoft.com/de-de/windows/mixed-reality/enthusiast-guide/tracking-system (besucht am 15. 03. 2022).

[54] Microsoft Docs. *Auswählen Ihrer Engine - Mixed Reality*. de-de. URL: https://docs.microsoft.com/de-de/windows/mixed-reality/develop/choosing-an-engine (besucht am 15. 03. 2022).

[55] Wolfgang Broll u. a. „Authoring von VR/AR-Anwendungen“. In: *Virtual und Augmented Reality (VR/AR)*. Springer Berlin Heidelberg, 2019, S. 393–423. DOI: 10.1007/978-3-662-58861-1_10.

[56] Unity Technologies. *Augmented-Reality-Entwicklungssoftware | AR-Engine für Apps | Unity*. de. URL: https://unity.com/de/unity/features/ar (besucht am 15. 03. 2022).

[57] Microsoft. *What is the Mixed Reality Toolkit*. original-date: 2016-01-28T18:54:58Z. Feb. 2022. URL: https://github.com/microsoft/MixedRealityToolkit-Unity (besucht am 08. 02. 2022).

[58] Nel Samama. *Indoor Positioning*. Wiley, Juli 2019. DOI: 10.1002/9781119421887.

[59] Alberto Menache. „Motion Capture Primer“. In: *Understanding Motion Capture for Computer Animation*. Elsevier, 2011, S. 1–46. DOI: 10.1016/b978-0-12-381496-8.00001-9.

[60] Eline van der Kruk und Marco M. Reijne. „Accuracy of human motion capture systems for sport applications; state-of-the-art review“. In: *European Journal of Sport Science* 18.6 (Mai 2018), S. 806–819. DOI: 10.1080/17461391.2018.1463397.

[61] Robert M. Kanko u. a. „Concurrent assessment of gait kinematics using marker-based and markerless motion capture“. In: *Journal of Biomechanics* 127 (2021), S. 110665. DOI: 10.1016/j.jbiomech.2021.110665.

[62] OptiTrack. *Prime^X 41 - Specs*. en. URL: http://optitrack.com/cameras/primex-41/specs.html (besucht am 22. 02. 2022).

[63] OptiTrack. *Build Your Own Motion Capture System*. en. URL: http://optitrack.com/systems/index.html (besucht am 22. 02. 2022).

[64] Nobuyasu Nakano u. a. „Evaluation of 3D Markerless Motion Capture Accuracy Using OpenPose With Multiple Video Cameras“. In: *Frontiers in Sports and Active Living* 2 (Mai 2020). DOI: 10.3389/fspor.2020.00050.

[65] Captury. *Markerless motion capture technology*. en-US. Apr. 2020. URL: https://captury.com/ (besucht am 09. 04. 2022).

[66] Hexagon Manufacturing Intelligence. *Leica Absolute Tracker AT403*. en. URL: https://www.hexagonmi.com/products/laser-tracker-systems/leica-absolute-tracker-at403 (besucht am 21. 02. 2022).

[67] Hexagon Manufacturing Intelligence. *Leica Absolute Tracker AT960*. de-DE. URL: https://www.hexagonmi.com/de-DE/products/laser-tracker-systems/leica-absolute-tracker-at960 (besucht am 10. 04. 2022).

[68] Hexagon Manufacturing Intelligence. *Zubehör für Laser Tracker*. de-DE. URL: https://www.hexagonmi.com/de-DE/products/laser-tracker-systems/accessories-for-laser-tracker-systems (besucht am 10. 04. 2022).

[69] Peter Pirc. *Mythen und Wahrheiten über UWB*. de. URL: https://www.industr.com/de/mythen-und-wahrheiten-ueber-uwb-2598766 (besucht am 09. 04. 2022).

[70] Cemin Zhang u. a. „Accurate UWB indoor localization system utilizing time difference of arrival approach“. In: *2006 IEEE Radio and Wireless Symposium*. IEEE. DOI: 10.1109/rws.2006.1615207.

[71] Sewio. *UWB Anchors for Indoor Location Tracking*. en-US. URL: https://www.sewio.net/uwb-anchors/ (besucht am 06. 04. 2022).

[72] Profibus e.V. *omlox - the standard for locating technologies*. en-US. URL: https://www.profibus.com/technology/omlox (besucht am 06. 04. 2022).

[73] Leo Becker. *Ultrabreitband-Chip U1: Apple will offene Standards unterstützen*. de. URL: https://www.heise.de/news/Ultrabreitband-Chip-U1-Apple-will-offene-Standards-unterstuetzen-6052277.html (besucht am 06. 04. 2022).

[74] Andrii Kudriashov u. a. „Introduction to Mobile Robots Navigation, Localization and Mapping“. In: *Mechanisms and Machine Science*. Springer International Publishing, 2020, S. 7–38. DOI: 10.1007/978-3-030-48981-6_2.

[75] Microsoft Docs. *Spatial mapping - Mixed Reality*. en-us. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping (besucht am 10. 04. 2022).

[76] A. L. Guinet u. a. „Reliability of the head tracking measured by Microsoft Hololens during different walking conditions“. In: *Computer Methods in Biomechanics and Biomedical Engineering* 22.sup1 (Mai 2019), S169–S171. DOI: 10.1080/10255842.2020.1714228.

[77] Patrick Hübner u. a. „Evaluation of HoloLens Tracking and Depth Sensing for Indoor Mapping Applications“. In: *Sensors* 20.4 (Feb. 2020), S. 1021. DOI: 10.3390/s20041021.

[78] Inês Soares u. a. „Accuracy and Repeatability Tests on HoloLens 2 and HTC Vive“. In: *Multimodal Technologies and Interaction* 5.8 (Aug. 2021), S. 47. DOI: 10.3390/mti5080047.

[79] Jindrich Cyrus u. a. „Hololens Used for Precise Position Tracking of the Third Party Devices - Autonomous Vehicles“. In: *Communications - Scientific letters of the University of Zilina* 21.2 (Mai 2019), S. 18–23. DOI: 10.26552/com.c.2019.2.18-23.

[80] Na Guo u. a. „An Online Calibration Method for Microsoft HoloLens“. In: *IEEE Access* 7 (2019), S. 101795–101803. DOI: 10.1109/access.2019.2930701.

[81] Christopher M. Andrews u. a. „Registration Techniques for Clinical Applications of Three-Dimensional Augmented Reality Devices“. In: *IEEE Journal of Translational Engineering in Health and Medicine* 9 (2021), S. 1–14. DOI: 10.1109/jtehm.2020.3045642.

[82] Vuforia Developer Portal. *Vuforia Engine 10.5 - Model Targets*. en. URL: https://library.vuforia.com/objects/model-targets (besucht am 03. 04. 2022).

[83] P. Hübner, M. Weinmann und S. Wursthorn. „Marker-based localization of the Microsoft HoloLens in building models“. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-1 (Sep. 2018), S. 195–202. DOI: 10.5194/isprs-archives-xlii-1-195-2018.

[84] ISO Central Secretary. *Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification*. en. Standard ISO/IEC 18004:2015. Geneva, CH: International Organization for Standardization, 2015. URL: https://www.iso.org/standard/62021.html.

[85] Microsoft Docs. *QR code tracking overview - Mixed Reality*. en-us. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/qr-code-tracking-overview (besucht am 03. 04. 2022).

[86] W3Schools. *JSON Introduction*. en-US. URL: https://www.w3schools.com/js/js_json_intro.asp (besucht am 12. 04. 2022).

[87] Jesper Larsson Träff und Robert A. van de Geijn. „Broadcast“. In: *Encyclopedia of Parallel Computing*. Hrsg. von David Padua. Boston, MA: Springer US, 2011, S. 186–192. DOI: 10.1007/978-0-387-09766-4_29.

[88] Alexey Melnikov und Ian Fette. *The WebSocket Protocol*. Dez. 2011. DOI: 10.17487/RFC6455. URL: https://datatracker.ietf.org/doc/rfc6455 (besucht am 06. 04. 2022).

[89] Jaime Navon und Federico Fernandez. „The Essence of REST Architectural Style“. In: *REST: From Research to Practice*. Springer New York, 2011, S. 21–33. DOI: 10.1007/978-1-4419-8303-9_1.

[90] MongoDB. *The Application Data Platform*. en-us. URL: https://www.mongodb.com (besucht am 25. 03. 2022).

[91] MongoDB. *NoSQL Databankenerklärung*. de-de. URL: https://www.mongodb.com/de-de/nosql-explained (besucht am 25. 03. 2022).

[92] MongoDB. *Document Database - NoSQL*. en-us. URL: https://www.mongodb.com/de/document-databases (besucht am 25. 03. 2022).

[93] MongoDB. *Start Developing with MongoDB — MongoDB Drivers*. en. URL: https://www.mongodb.com/docs/drivers/ (besucht am 04. 04. 2022).

[94] Alan S. Manne. „On the Job-Shop Scheduling Problem“. In: *Operations Research* 8.2 (Apr. 1960), S. 219–223. DOI: 10.1287/opre.8.2.219.

[95] David Applegate und William Cook. „A Computational Study of the Job-Shop Scheduling Problem“. In: *ORSA Journal on Computing* 3.2 (Mai 1991), S. 149–156. DOI: 10.1287/ijoc.3.2.149.

[96] Bernhard Korte und Jens Vygen. „Einführung“. In: *Kombinatorische Optimierung*. Springer Berlin Heidelberg, 2012, S. 1–13. DOI: 10.1007/978-3-642-25401-7_1.

[97] Isabel Jasmin Acker. „Mehrkriterielles Job-Shop-Scheduling mit alternativen Maschinenfolgen“. In: *Business Excellence in Produktion und Logistik*. Hrsg. von Wolf Wenger, Martin Josef Geiger und Andreas Kleine. Wiesbaden: Gabler, 2011, S. 65–86. DOI: 10.1007/978-3-8349-6688-9_4.

[98] Google Developers. *CP-SAT Solver | OR-Tools*. en. URL: https://developers.google.com/optimization/cp/cp_solver (besucht am 24. 03. 2022).

[99] Google Developers. *The Job Shop Problem | OR-Tools*. en. URL: https://developers.google.com/optimization/scheduling/job_shop (besucht am 24. 03. 2022).

[100] NuGet. *Microsoft.MixedReality.QR 0.5.3013*. en. URL: https://www.nuget.org/packages/Microsoft.MixedReality.QR/ (besucht am 03. 04. 2022).

[101] Microsoft Docs. *Grundlegendes zur Leistung für Mixed Reality - Mixed Reality*. de-de. URL: https://docs.microsoft.com/de-de/windows/mixed-reality/develop/advanced-concepts/understanding-performance-for-mixed-reality (besucht am 09. 04. 2022).

[102] GitHub. *endel/NativeWebSocket*. URL: https://github.com/endel/NativeWebSocket (besucht am 06. 04. 2022).

[103] OpenJS. *Node.js*. en. URL: https://nodejs.org/en/ (besucht am 09. 04. 2022).

[104] GitHub/websockets. *ws: a Node.js WebSocket library*. URL: https://github.com/websockets/ws (besucht am 09. 04. 2022).

[105] OpenJS. *Express - Node.js-Framework von Webanwendungen*. de. URL: https://expressjs.com/de/ (besucht am 09. 04. 2022).

[106] *mongoose.js*. en. URL: https://mongoosejs.com/ (besucht am 09. 04. 2022).

[107] Christian Heipke. „Photogrammetrie und Fernerkundung – eine Einführung“. In: *Photogrammetrie und Fernerkundung*. Springer Berlin Heidelberg, 2017, S. 1–27. DOI: 10.1007/978-3-662-47094-7_37.

[108] GOM. *TRITOP: Optisches Photogrammetriesystem*. de. URL: https://www.gom.com/de-de/produkte/3d-scanning/tritop (besucht am 12. 04. 2022).

[109] GOM GmbH. *Acceptance/Reverification of the Camera According to VDI/VDE 2634, Part 1 (internes Dokument)*.

[110] Arndt Lüder und Nicole Schmidt. „AutomationML in a Nutshell“. In: *Springer Reference Technik*. Springer Berlin Heidelberg, 2020, S. 1–48. DOI: 10.1007/978-3-662-45537-1_61-2.

VI Anhang

A. Inhalte des Datenträgers

Auf der beigelegten CD befinden sich:

- die digitale Version der Arbeit
- die für die Ausarbeitung erstellten LaTeX-Dokumente
- die Literaturquellen
- der Programmcode der Serveranwendung und der AR-Anwendung

B. Vollständige geometrische Prozessbeschreibung des Beispielprozesses

```
{
 "tasks": [
   {
     "id": "A",
     "steps": [
       {
         "id": "A1",
         "startGeometry": "AB",
         "endGeometry": "L1",
         "startConfType": "leave",
         "endConfType": "enter"
       },
       {
         "id": "A2",
         "startGeometry": "L1",
         "endGeometry": "AB",
         "startConfType": "enter",
         "endConfType": "enter"
       }
     ],
     "dependencies": []
   },
   {
     "id": "B",
```

```
25      "steps": [
26        {
27          "id": "B1",
28          "startGeometry": "AB",
29          "endGeometry": "L2",
30          "startConfType": "leave",
31          "endConfType": "enter"
32        },
33        {
34          "id": "B2",
35          "startGeometry": "L2",
36          "endGeometry": "AB",
37          "startConfType": "enter",
38          "endConfType": "enter"
39        }
40      ],
41      "dependencies": []
42    },
43    {
44      "id": "C",
45      "steps": [
46        {
47          "id": "C1",
48          "startGeometry": "AB",
49          "endGeometry": "AB",
50          "startConfType": "enter",
51          "endConfType": "leave"
52        }
53      ],
54      "dependencies": ["A", "B", "1", "2", "3"]
55    },
56    {
57      "id": "D",
58      "steps": [
59        {
60          "id": "D1",
61          "startGeometry": "AB",
62          "endGeometry": "L1",
63          "startConfType": "leave",
64          "endConfType": "enter"
65        },
66        {
67          "id": "D2",
```

```
68            "startGeometry": "L1",
69            "endGeometry": "AB",
70            "startConfType": "enter",
71            "endConfType": "enter"
72          }
73        ],
74        "dependencies": []
75      },
76      {
77        "id": "E",
78        "steps": [
79          {
80            "id": "E1",
81            "startGeometry": "AB,
82            "endGeometry": "L2",
83            "startConfType": "leave",
84            "endConfType": "enter"
85          },
86          {
87            "id": "E2",
88            "startGeometry": "L2",
89            "endGeometry": "AB",
90            "startConfType": "enter",
91            "endConfType": "enter"
92          }
93        ],
94        "dependencies": []
95      },
96      {
97        "id": "F",
98        "steps": [
99          {
100           "id": "F1",
101           "startGeometry": "AB",
102           "endGeometry": "AB",
103           "startConfType": "enter",
104           "endConfType": "leave"
105         }
106       ],
107       "dependencies": ["D", "E", "4"]
108     },
109     {
110       "id": "G",
```

```
      "steps": [
        {
          "id": "G1",
          "startGeometry": "AB",
          "endGeometry": "L2",
          "startConfType": "enter",
          "endConfType": "enter"
        },
        {
          "id": "G2",
          "startGeometry": "L2",
          "endGeometry": "AB",
          "startConfType": "enter",
          "endConfType": "enter"
        }
      ],
      "dependencies": []
    },
    {
      "id": "H",
      "steps": [
        {
          "id": "H1",
          "startGeometry": "AB",
          "endGeometry": "AB",
          "startConfType": "enter",
          "endConfType": "leave"
        },
        {
          "id": "H2",
          "startGeometry": "L2",
          "endGeometry": "L2",
          "startConfType": "enter",
          "endConfType": "leave"
        }
      ],
      "dependencies": ["G", "8", "9"]
    }
  ],
  "geometries": [
    {
      "id": "L1",
      "posX": "3",
```

```
154       "posY": "0.5",
155       "posZ": "0.4",
156       "length": "1.3",
157       "width": "3",
158       "height": "1.3",
159     },
160     {
161       "id": "L2",
162       "posX": "4.6",
163       "posY": "0.5",
164       "posZ": "-4.5",
165       "length": "1.3",
166       "width": "3",
167       "height": "1.3",
168     },
169     {
170       "id": "AB",
171       "posX": "1.1",
172       "posY": "0.5",
173       "posZ": "-0.5",
174       "length": "1.6",
175       "width": "3",
176       "height": "2",
177     }
178   ],
179 }
```

Listing VI.1: Geometrische Prozessbeschreibung im JSON-Format

C. Roboter-Zeiten des validierten Anwendungsfalls

Zusätzlich zu den gemessenen Roboter-Prozesszeiten wurden für die Optimierung des beispielhaften Montageprozesses auch die Verfahrzeiten zwischen einzelnen Prozessschritten berücksichtigt. In diesem Fall wurden die Zeiten während der realen Ausführung mit dem Roboter softwaregestützt gemessen. Für komplexere Anwendungsfälle mit einer größeren Anzahl an Varianten wäre es jedoch sinnvoll, die Werte durch Simulationen zu ermitteln. Die Messergebnisse sind in Tabelle VI.1 aufgelistet.

Tabelle VI.1.: Roboter-Übergangszeiten

		zu								
von		1	2	3	4	5	6	7	8	9
	1	-	6,0 s	8,4 s	11,4 s	14,4 s	14,4 s	13,0 s	7,1 s	10,8 s
	2	9,1 s	-	10,3 s	12,9 s	16,0 s	16,0 s	14,3 s	6,8 s	9,7 s
	3	11,8 s	11,8 s	-	7,4 s	17,8 s	17,8 s	16,4 s	5,7 s	8,4 s
	4	11,9 s	12,5 s	8,2 s	-	18,3 s	18,3 s	16,5 s	4,7 s	8,5 s
	5	15,2 s	15,2 s	14,6 s	14,5 s	-	2,5 s	1,9 s	18,4 s	23,9 s
	6	13,4 s	13,9 s	14,6 s	15,2 s	5,0 s	-	3,4 s	20,7 s	26,5 s
	7	16,2 s	16,2 s	15,7 s	15,6 s	2,5 s	2,5 s	-	16,8 s	22,9 s
	8	16,6 s	16,5 s	13,5 s	15,8 s	20,2 s	20,2 s	22,7 s	-	0,1 s
	9	11,5 s	12,0 s	9,2 s	11,0 s	19,1 s	19,1 s	17,3 s	2,8 s	-

D. Optimierung der Prozessreihenfolge mit CP-SAT-Solver

Anhand eines Beispiels wird an dieser Stelle die Funktionsweise und Implementierung des CP-SAT-Solvers zur Optimierung der Prozessreihenfolge erläutert. Im vorgestellten Beispiel sollen zwei Maschinen jeweils drei Aufgaben so erledigen, dass die Gesamtzeit für die Bearbeitung möglichst gering ist. Alle Ausgangsparameter werden im JSON-Format definiert und sind in Listing VI.2. Unter dem Schlüssel „tasks“ werden zunächst die Aufgaben als Arrays definiert. Zusätzlich werden Übergangszeiten zwischen den einzelnen Aufgaben vorgegeben sowie Abhängigkeiten. Die Abhängigkeiten definieren, welche Aufgabe abgeschlossen sein muss, bevor eine andere Aufgabe begonnen wird.

```
1  {
2    "tasks": [
3      [177, 0, "1"],
4      [105, 0, "2"],
5      [766, 0, "3"],
6      [48, 1, "A"],
7      [108, 1, "B"],
8      [572, 1, "C"]
9    ],
10   "transTimes": {
11     "1":{"2":60, "3":84},
12     "2":{"1":91, "3":103},
13     "3":{"1":118, "2":118},
14     "A":{"B":54, "C":52},
15     "B":{"A":12, "C":76},
16     "C":{"A":22, "B":42}
17   },
18   "dependencies": [
19     ["B", "1"],
```

```
20      ["2", "C"]
21    ]
22  }
```

Listing VI.2: Eingangswerte für die Optimierung

Auf Grundlage der Eingangsparameter werden für jede Aufgabe Variablen für den Startzeitpunkt, den Endzeitpunkt, die Prozessposition und das Durchführungsintervall definiert. Es wird zudem festgelegt, dass sich einzelne Intervalle bei einer Maschine nicht überschneiden dürfen.

```
1  for task_id, task in enumerate(tasks):
2    machine = task[1]
3    duration = task[0]
4    suffix = '_%i' % (task_id)
5    start_var = model.NewIntVar(0, horizon, 'start' + suffix)
6    end_var = model.NewIntVar(0, horizon, 'end' + suffix)
7    rank = model.NewIntVar(0,len(machine_tasks[machine])-1, 'rank' ↙
      ↳ + suffix)
8    task_ranks[task_id] = rank
9    interval_var = model.NewIntervalVar(start_var, duration, ↙
      ↳ end_var,'interval' + suffix)
10   all_tasks[task_id] = task_type(start=start_var, end=end_var, ↙
      ↳ interval=interval_var)
11   machine_to_intervals[machine].append(interval_var)
12
13 # Create and add disjunctive constraints.
14 for machine in all_machines:
15   model.AddNoOverlap(machine_to_intervals[machine])
```

Listing VI.3: Initialisierung der Variablen und Festlegung der No-Overlap-Bedingung

Zusätzlich sollen die Übergangszeiten zwischen einzelnen Aufgaben berücksichtigt werden. Dazu werden Bedingungen eingeführt, welche die auf eine Aufgabe folgende nächste Aufgabe mindestens um die vorher definierte Übergangszeit verzögert einplant.

```
1
2  for machine in all_machines:
3    for i in machine_tasks[machine]:
4      for j in machine_tasks[machine]:
5        if i==j:
6          continue
7
8        followVar = model.NewBoolVar('%i␣follows␣%i' % (j, i))
```

```
9         model.Add(all_tasks[j].start >= all_tasks[i].start).↙
    ↳ OnlyEnforceIf(followVar)
10        model.Add(all_tasks[j].start < all_tasks[i].start).↙
    ↳ OnlyEnforceIf(followVar.Not())
11        model.Add(task_ranks[j] >= task_ranks[i] + 1).OnlyEnforceIf↙
    ↳ (followVar)
12
13  for machine in all_machines:
14    for i in machine_tasks[machine]:
15      for j in machine_tasks[machine]:
16        if i==j:
17          continue
18
19        followVar = model.NewBoolVar('%i␣follows␣%i' % (j, i))
20        model.Add(task_ranks[j] == task_ranks[i] + 1).OnlyEnforceIf↙
    ↳ (followVar)
21        model.Add(task_ranks[j] != task_ranks[i] + 1).OnlyEnforceIf↙
    ↳ (followVar.Not())
22        model.Add(all_tasks[j].start >= all_tasks[i].end + ↙
    ↳ getTransTime(i,j,tasks,trans_times)).OnlyEnforceIf(↙
    ↳ followVar)
23
24  for i, dep in enumerate(dependencies):
25    model.Add(all_tasks[task_indicies[dep[0]]].start >= all_tasks[↙
    ↳ task_indicies[dep[1]]].end)
```

Listing VI.4: Festlegung der Bedingungen für die Reihenfolge unter Berücksichtigung der Übergangszeiten

Zuletzt wird festgelegt, dass der Solver die Gesamtprozessdauer minimieren soll. Mit der Funktion „solver.Solve()“ wird der Lösungsprozess gestartet.

```
1  obj_var = model.NewIntVar(0, horizon, 'makespan')
2  model.AddMaxEquality(obj_var, [
3    all_tasks[task_id].end
4    for task_id in enumerate(tasks)
5  ])
6  model.Minimize(obj_var)
7
8  solver = cp_model.CpSolver()
9  solver.parameters.max_time_in_seconds = 60 * 0.5 #60 * 2
10 solution_printer = SolutionPrinter(obj_var)
11 status = solver.Solve(model, solution_printer)
```

Listing VI.5: Definition des Optimierungsziels und Ausführung des Solvers

E. Verarbeitung von Roboterdaten und Robotersteuerung

Die zuvor ermittelten Roboter-Prozesszeiten sowie Übergangszeiten werden in einem Datenformat ähnlich dem der Prozessbeschreibung gespeichert. Der Aufbau ist in Listing VI.6 beispielhaft für einen Roboter-Prozessschritt dargestellt. Neben der Reihenfolge ist für jeden Schritt die gemessene Dauer sowie die Übergangszeit zu allem anderen Prozessschritten gespeichert.

```
1  {
2   "externalTasks": {
3    "ur10e": [
4      {
5        "id": "1",
6        "dependencies": [],
7        "duration": 18.7,
8        "transitions": {
9          "2": 6.0,
10         "3": 8.4,
11         "4": 11.4,
12         "5": 14.4,
13         "6": 14.4,
14         "7": 13.0,
15         "8": 7.1,
16         "9": 10.8
17       }
18     },
19     ...
20   ]
21  }
22 }
```

Listing VI.6: JSON-Schema für Speicherung von Roboter-Prozesszeiten

Ist die Reihenfolge der Roboter-Aufgaben festgelegt, so lässt sich die Anwendung zur Analyse von Arbeiterpositionsdaten so erweitern, dass auch ein automatisches Starten von Roboterprozessen durch den Server erfolgen kann. Hierzu wird auf dem UR10e ein Programm definiert, in dem alle Einzelprozesse und deren Wegpunkte vorgegeben werden. Ein beispielhaftes Programm ist in Abb. VI.1 dargestellt. Das Programm wartet auf eine Veränderung der Variable „task_id“ und führt dann den entsprechenden Prozessschritt aus. Die Variable kann durch den Server verändert werden, indem auf den sogenannten Interpreter-Mode zurückgegriffen wird, welcher im Beispiel in Programmzeile drei aufgerufen wird.

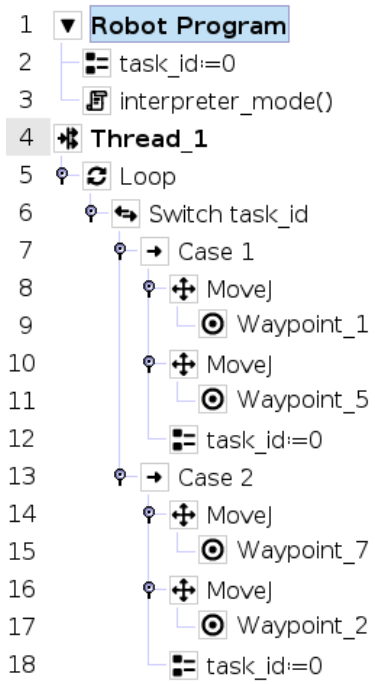


Abbildung VI.1.: Ausschnitt aus dem Roboter-Programm

Als Gegenstück zum auf dem Roboter ausgeführten Interpreter-Mode fungiert eine in Python implementierte Server-Anwendung, welche in Listing VI.7 dargestellt ist. Mit der IP-Adresse des Roboters wird eine Verbindung hergestellt (Zeile 3) und der Befehl zur Änderung der Variable „task_id“ an diesen gesendet (Zeile 4).

```
1 def send_cmd_interpreter_mode(ip, taskId):
2   interpreter = InterpreterHelper(ip)
3   interpreter.connect()
4   interpreter.execute_command("task_id="+ taskId)
5   interpreter.end_interpreter()
```

Listing VI.7: Python-Code zum Senden von Interpreter-Nachrichten an Roboter