



Konzeption und prototypische Umsetzung  
einer Steuerzentrale eines smarten Büros mit  
dem Fokus einer einfachen Handhabung der  
formalisierten Interaktionen für  
Softwareentwickler

## MASTER-THESIS

für die Prüfung zum  
Master of Science  
des Studienganges Professional Software Engineering  
an der  
Knowledge Foundation @ Reutlingen University

von

**Mikka Jenne**

Abgabedatum 31. August 2022

Bearbeitungszeitraum  
Teilnehmernummer  
Kurs  
Standort der Universität  
Standort der Firma  
Betreuer der Ausbildungsfirma  
Gutachter der Studienakademie

24 Wochen  
800864  
PSEJG20  
Reutlingen  
Karlsruhe  
Dr. Robin Braun  
Prof. Dr. Natividad Martinez Madrid

## **Zusammenfassung**

Augmented Reality ist eine Technologie, die dem Nutzer ein visuelles Erlebnis mit einer angereicherten Welt voller virtueller Objekte ermöglicht. Das Resultat, eine Kombination aus Realität und Virtualität, bietet dem Benutzer eine neue Art der Wahrnehmung der Gegenwart.

Diese Bachelorarbeit befasst sich mit der Konzeption und Umsetzung eines industriellen Assistenzsystems unter Verwendung der Augmented Reality Technologie. Dabei soll die Umgebung mit Hilfe des SLAM Verfahrens analysiert werden, um auf dieser Basis dreidimensionale Objekte als Referenz zu realen Objekten im Raum virtuell platzieren zu können. Durch die entstehende Visualisierung können Informationen zu den jeweiligen Objekten in eine Datenbank eingetragen und angezeigt werden, dadurch kann das Überwachen von Industriemaschinen vereinfacht werden.

Zu dem Konzept gehört sowohl die Ausarbeitung der grundlegenden Softwarearchitektur, als auch ein allgemein-gültiges Datenmodell zur Persistierung der generierten Daten. Für die bestmögliche Umsetzung der Augmented Reality Experience werden hierzu bereits schon bestehende Frameworks und Software Development Kits, beispielsweise Google ARCore, verwendet.

Der entstandene Prototyp ist ein eigenständiges System. Die Architektur ist modular aufgebaut, um eine stetige Weiterentwicklung zu gewährleisten.

## **Abstract**

Augmented Reality is a technology that enables the user to have a visual experience with an enriched world full of virtual objects. The result offers the user a new way of perceiving surrounding as an Combination of reality and virtuality.

This bachelor thesis deals with the conception and implementation of an industrial assistance system using augmented reality technology. The environment can be analyzed with the help of the SLAM method in order to be able to place three-dimensional objects virtually as a reference to real objects in space. The resulting visualization enables information on the respective objects to be entered in a database and displayed, which enables the simplified monitoring of Industrial machines.

The concept includes the development of the basic software architecture as well as a generally applicable data model for saving the generated data. Already existing frameworks and software development kits where used for the best possible implementation of the augmented reality experience as Google ARCore.

The created prototype is an standalone system. The architecture is modular in order to ensure continuous further development.

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>2</b>
1.1 Motivation . . . . .	2
1.2 cjt Systemsoftware AG . . . . .	6
1.3 Aufgabenstellung . . . . .	7
1.4 Aufbau der Arbeit . . . . .	8
1.5 Stand der Technik . . . . .	10
<b>2 Grundlagen</b>	<b>11</b>
2.1 Augmented Reality . . . . .	11
2.1.1 Virtual Reality, Augmented Reality und Mixed Reality . . . . .	13
2.1.2 Varianten der Augmented Reality . . . . .	16
2.1.3 Positionsbestimmung . . . . .	18
2.1.4 Augmented Reality in der Industrie . . . . .	21
2.2 SLAM - Simultanious Localization And Mapping . . . . .	22
2.2.1 Definition des Problems . . . . .	22
2.2.2 Localization . . . . .	23
2.2.3 Mapping . . . . .	24
2.2.4 Verfahren zur Lösung des SLAM Problems . . . . .	24
2.3 Quaternionen . . . . .	26
2.3.1 Rotation . . . . .	27
2.4 Technologien . . . . .	27
2.4.1 Google ARCore . . . . .	27
2.4.2 Android Jetpack . . . . .	31
2.4.3 Sceneform SDK . . . . .	35
2.4.4 SQLite . . . . .	35
2.5 Softwarearchitektur . . . . .	36
2.5.1 Modulare Software-Architektur . . . . .	37
2.5.2 MVVM . . . . .	38
2.5.3 Android Architecture Components . . . . .	39
2.6 Datenmodellierung . . . . .	42

<i>INHALTSVERZEICHNIS</i>	1
<b>3 Konzeption</b>	<b>46</b>
3.1 Anwendungsumfeld . . . . .	46
3.2 Scan-Phase . . . . .	47
3.3 Visualisierungs-Phase . . . . .	48
3.4 Architekturkonzept . . . . .	48
3.5 Softwarekonzept . . . . .	50
3.6 Auswahl des AR Frameworks . . . . .	53
3.6.1 ARToolKit . . . . .	54
3.6.2 Google ARCore . . . . .	54
3.7 Datenmodell . . . . .	55
<b>4 Umsetzung</b>	<b>58</b>
4.1 Implementierung . . . . .	58
4.1.1 Startmenü . . . . .	59
4.1.2 Scan-Phase . . . . .	63
4.1.3 Visualisierungs-Phase . . . . .	77
<b>5 Evaluation</b>	<b>83</b>
<b>6 Fazit</b>	<b>85</b>
<b>7 Ausblick</b>	<b>87</b>
<b>Anhang</b>	<b>I</b>
<b>Index</b>	<b>I</b>
<b>Literaturverzeichnis</b>	<b>VII</b>

# Kapitel 1

## Einleitung

In diesem Teil der Arbeit wird auf die Motivation des Themas eingegangen. Darüber hinaus wird sowohl die Aufgabenstellung als auch der Aufbau der Arbeit genauestens dargelegt. Eine nähere Betrachtung des Standes der Technik untermauert die Beweggründe dieser Ausarbeitung.

### 1.1 Motivation

Jede neu entwickelte Technologie durchlebt im Laufe der Entstehung ein enormes Aufsehen. Es wird viel darüber debattiert, fantasiiert und geplant ohne jedoch genau die Resultate abwägen zu können. Durch fehlende Erfahrung und nicht ausgereifte Konzepte werden Highlights erwartet, die zu diesem Zeitpunkt technisch nicht umsetzbar sind. Jede neue technologische Idee durchläuft bestimmte Phasen der Entwicklung. [B12 2020]

Ein sogenannter Hype Cycle, dt. Hype-Zyklus, ist ein visualisiertes Modell, das die Entwicklung einer neuen Technologie von der Innovation über die Umsetzung bis hin zur ausgereiften Marktfähigkeit repräsentiert und so diese Phasen der Entwicklung verdeutlicht.

Der Hype Cycle, entwickelt von der Forschungsgruppe Gartner Inc. und die durch deren Mitarbeiterin Jackie Fenn geprägten Definitionen, ist in fünf Entwicklungsphasen untergliedert:

1. *Technologische Auslösung*: Bekanntwerden der Technologie und erste Projekte, die auf beachtliches Interesse in der Öffentlichkeit stoßen.
2. *Gipfel der überzogenen Erwartungen*: Die Technologie ist auf dem Gipfel der Aufmerksamkeit. Es werden unrealistische und enthusiastische Erwartungen veröffentlicht und diskutiert.
3. *Tal der Enttäuschung*: Da die Technologie nicht die zuvor aufgebauten Erwartungen erfüllen kann, sinkt die Aufmerksamkeit der enttäuschten Enthusiasten und der Medienvertreter auf den Tiefpunkt.
4. *Pfad der Erleuchtung*: Die neue Technologie wird in dieser Phase realistisch mit ihren Stärken und Schwächen betrachtet.

5. *Plateau der Produktivität:* Die neuen Möglichkeiten durch die Technologie werden als Vorteile akzeptiert, und Geschäftsprozesse werden entwickelt. [PINKER 2016]

Nachdem eine Innovation den Gipfel der überzogenen Erwartungen, z.B. die vollständige Revolutionierung der Industrie oder Szenarien, wie z.B. Hologramme, die man nur aus Science-Fiction Filmen kennt, passiert hat, folgt das Tal der Enttäuschung, wobei festgestellt wird, dass die Erwartungen nicht übertragbar sind, bzw. nur zum Teil in die Realität umgesetzt werden können und die Technologie an Interesse verliert. Nach der erneuten Sammlung, der "*Kurs-Korrektur*" [BITFORGE 2019], wird die präsente Innovation realistischer beurteilt. Durch die objektive und realitätsnähre Betrachtungsweise entsteht ein neues und realistisches Bild der Möglichkeiten, als auch der Grenzen der Technologie. Zum Ende hin geht die ehemals neue Innovation in eine routinierte Technologie über, wobei diese an Anerkennung gewinnt und sich bewährt. Bei den Nutzern findet sie immer mehr Zuspruch und zu Beginn utopisch gedachte Konzepte sind längst einer realen Praxis gewichen. Durch die steigende Zuwendung zu der Technologie erfährt diese eine stetige Weiterentwicklung und die ursprünglich kleine Nutzergruppe kann zu einer großen Community wachsen. Bezogen auf die Phasen des Hype Cycles befindet sich diese Technologie dann an der Stelle des Plateaus der Produktivität und bestätigt die Marktreife. Ab diesem Zeitpunkt handelt es sich nicht mehr um eine Zukunftsvision, Hype oder Highlight, sondern um eine am Markt etablierte Technologie.

Momentan befindet sich Augmented Reality auf dem Pfad der Erleuchtung und ist auf sehr gutem Wege zu einer ausgereiften Technologie, da das Wachstum der Verwendung von Augmented Reality stetig steigt und mittlerweile ein weites Portfolio an möglichen Einsatzgebieten vorweist. Die jetzige Erfahrung und der technologische Fortschritt bringt Augmented Reality den ursprünglich angedachten Visionen und Ideen einen Schritt näher, sodass in naher Zukunft die letzte Phase, das Erreichen des Plateaus der Produktivität erzielt werden kann. Den finalen Schritt der endgültigen Marktreife zu erlangen ist ein faszinierender und wichtiger Grund für meine Motivation mich dieser Technologie und der dahinterstehenden Theorie zu widmen. Die Technologie der Augmented Reality weist, wie bereits erwähnt, ein immer größer werdendes Portfolio an Einsatzgebieten vor, unter anderem:

- Industrie
- Produktion und Lagerlogistik
- Wartung und Reparatur
- Spiele, bzw. Gaming
- Medizin
- Marketing und Werbung
- Navigation
- Unterhaltung und Fernsehen
- Schulung und Training

- Militär

Neben der Affinität der Technologie bringt AR Vorteile mit sich, z.B. Arbeitsprozesse und Heran gehensweisen an Projekte zu modernisieren. Ebenso wird die Fehleranfälligkeit durch menschliche Unachtsamkeit mit einer Kombination aus vielen Informationen und deren visueller Darstellung in Echtzeit reduziert. Hinzu kommt die Steigerung der Produktivität und Verbesserung des Wissenstransfers durch die dargebotenen Angaben die visualisiert werden.

Schon im Jahr 2016 sprach Apple CEO Tim Cook die Innovation *Augmented Reality* bei zahlreichen Events, in Keynotes und Interviews an und war zu diesem Zeitpunkt schon enorm begeistert davon. Er beteuerte: „...*using the tech would become as normal as eating three meals a day.*“ [LESWING 2016] Bei jeder sich ihm bietenden Möglichkeit ging der Apple CEO auf seine Überzeugung gegenüber Augmented Reality ein. Erst vor Kurzem bestätigte er, dass AR zahlreiche und innovative Einsatzgebiete erlangen und immer mehr an Wichtigkeit zunehmen würde, als er sagte: „*pervade your life, it will play a big role...*“ [EADICICCO 2020]

Ein sehr großer Anwendungsbereich mit viel Potential, das bei weitem noch nicht ausgeschöpft sei, ist der Industriezweig. Die Marktstudie zu Augmented Reality der IDG Research Services und PTC untermauern das immer weiter steigende Wachstum der Anwendungen in Unternehmen. „Fast 75% der deutschen Unternehmen setzen bereits *Virtual oder Augmented Reality* ein oder planen dies.“ [MUNDT 2020] Eine Prognose berechnet einen Umsatz in zweistelliger Milliarden-Höhe unter Einsatz und Anwendung von AR der bis 2021 erreicht werden soll. So wird umso deutlicher, dass es sich bei Augmented Reality um ein sehr zukunftsorientiertes Marktsegment mit viel Potential handelt. Die Technologie rückt immer weiter in den Vordergrund und immer mehr Unternehmen setzen sich damit auseinander.

Durch den Trend zur Vollautomatisierung von Produktions- und Industrieprozessen werden Mengen an Daten erzeugt. Die moderne Industrie 4.0 steigert das Wachstum der Daten, indem Maschinen immer mehr digitalisiert und Protokolle parallel zur Laufzeit der Anlage erzeugt werden. Die entstehenden Daten werden über eine Cloud zur Verfügung gestellt. Dieser Vorgang des Datentransfers ist Teil einer globalen Infrastruktur, die es ermöglicht physische und virtuelle Gegenstände miteinander zu vernetzen, auch bekannt als Internet of Things, dt. Internet der Dinge (IoT). Entstehende Daten sind z.B. Maschinendaten aus Maschinen-Protokollen, Prozessinformationen, Produktionsdaten oder Informationen der Endverbraucher. [KRAUSS 2019] Augmented Reality ermöglicht es, einen Großteil der Daten sinnvoll zu nutzen, d.h. dem Nutzer Auskünfte über Maschinen besonders hervorzuheben. Darüber hinaus können unter anderem durch AR veraltete Papierprozesse abgelöst oder digitale Lösungen in bestehende IT-Infrastrukturen eingebunden werden. [BOUVERET und HUMAN 2019] Dabei sind Rückschlüsse auf die allgemeine Arbeitsoptimierung zu ziehen. Notwendige Informationen sind ohne großen Aufwand, schnell und zentral an einem Ort durch AR zur Verfügung gestellt und abrufbar. Damit können Wartungen schneller durchgeführt, Defekte besser behoben, Leerläufe oder Fehler am Endprodukt vermieden werden. Mit solch einer Unterstützung kann z.B. auch die Orientierung in riesigen Produktionshallen aufrecht erhalten werden, um einen besseren Überblick über alle Maschinen zu erhalten. Auch wird die Effizienz von Prozessen erhöht.

Ein Mechaniker kann alle notwendigen Informationen, die er benötigt, der Augmented Reality Applikation entnehmen, um z.B. eine Wartung einer Anlage schneller durchzuführen oder über Vorfälle an der Maschinerie in Echtzeit informiert zu sein.

## 1.2 cjt Systemsoftware AG

Die Arbeit wurde bei der Firma cjt Systemsoftware AG durchgeführt. Diese wurde 1999 von Christian J. Tauber und Ulrich Beck gegründet. Das Unternehmen beschäftigt mittlerweile mehr als 60 Mitarbeiterinnen und Mitarbeiter. Mit ihrem Sitz in Karlsruhe ist sie in einer der größten Technologiestädten Deutschlands angesiedelt.

Durch das stetige Wachstum der cjt Systemsoftware AG vergrößert sich auch deren Portfolio kontinuierlich. Dabei setzt das Consulting-Unternehmen hauptsächlich auf maßgeschneiderte Software- und Netzwerklösungen. Großkunden wie Siemens AG, Lufthansa Cargo, Forschungszentrum Karlsruhe (KIT) und Fraunhofer IOSB zeugen von der hohen Qualität der geleisteten Arbeit. Dabei agiert das Unternehmen nicht nur in Deutschland sondern auch international, darunter in Ländern wie China und den USA.

## 1.3 Aufgabenstellung

Im Rahmen dieser Ausarbeitung soll ein System konzipiert, entwickelt und umgesetzt werden, welches basierend auf Augmented Reality ein Informations- und Unterstützungssystem im industriellen Bereich grundlegend realisiert.

Die entstehende prototypische Applikation soll es ermöglichen, einen Überblick über eine Produktions- oder Industriehalle zu verschaffen, indem die Umgebung und alle in der Halle stehenden Maschinen über die Kamera eines Smart-Devices erkannt und als Overlay über das Live-Bild gelegt und angezeigt werden. Die erkannten Objekte können vom Nutzer eingetragen werden, um die Position des Gegenstandes festhalten zu können. Nach dem Positionieren des Objektes kann der Nutzer die benötigten Informationen eintragen, um so grundlegende Informationen über diesen Gegenstand griffbereit zu haben. Damit soll der Grundbaustein für ein Unterstützungssystem gelegt werden, das beliebig erweiterbar ist, um ein nützliches *Gadget* mit vielen *Features* in der Industrie zur Verfügung zu stellen.

Grundlegend ist das Projekt in drei Aufgabenbereiche unterteilt:

- Planung der Architektur
- Implementierung der Grundfunktionen der Applikation
- Grundlage der Objektinformation als Datenmodell

Um eine Applikation übersichtlich zu gestalten und für die Zukunft Erweiterungen möglichst einfach integrieren zu können, ist es zu Anfang die Aufgabe, eine solide und übersichtliche Grundstruktur, bzw. Software- Architektur zu erstellen. Dadurch werden Funktionen und Klassen einer klaren Struktur zugeordnet und eine einheitliche Linie vorgegeben.

Ein weiterer Aspekt, der bei der Erstellung der Architektur berücksichtigt werden soll, ist der Ansatz der modularen Softwarearchitektur. Damit können einzelne Funktionen unabhängig voneinander getestet und Abhängigkeiten oder Frameworks leichter ersetzt, hinzugefügt oder entfernt werden. Darüber hinaus begünstigt eine modulare Konzeption bessere Kontrollierbarkeit und Übersichtlichkeit in großen Softwareprojekten.

Die Hauptaufgabe ist die Realisierung der Augmented Reality-Funktion, sozusagen der Kern der Anwendung. Dabei wird die Applikation in zwei Phasen unterteilt, welche es separat zu planen und implementieren gilt.

Die erste Phase, genannt Scan-Phase, beschäftigt sich mit dem scannen der Umgebung, bzw. des Raumes. Die Aufgabe dabei besteht darin, mittels dem Simultanious Localization And Mapping (SLAM) - Verfahren eine Karte der Umgebung zu erstellen und die räumliche Lage innerhalb dieser Karte zu schätzen, um auf Basis dieser erstellten Karte virtuell Objekte auf der Karte platzieren zu können. Mit den gewonnenen Informationen der räumlichen Darstellung durch das SLAM - Verfahren kann der Nutzer virtuelle Objekte im virtuellen Raum an Ort und Stelle platzieren, als Referenz zu dem existierenden Objekt in der Realität, welches erkannt wurde. Bei der Erstellung eines Objekts soll der Nutzer die Möglichkeit haben, Informationen über das Objekt in das System einzupflegen, damit er diese immer abrufen kann.

In der zweiten Phase, genannt Visualisierungs-Phase, sollte der Nutzer die Möglichkeit haben, sich im Raum frei bewegen zu können. Mit der Lokalisierung des Nutzer-Geräts und den bekannten Informationen der in Phase eins gesetzten Objekte, sollten dem Nutzer die virtuellen Objekte in seiner unmittelbaren Umgebung angezeigt werden. Mit dem Wissen, dass sich im Blickfeld der AR-Applikation ein Objekt befindet, können für dieses in der Datenbank weitere Informationen abgefragt und dem Nutzer zur Verfügung gestellt werden.

Ein weiterer wichtiger Punkt dieses Konzeptes wird die Modellierung eines geeigneten, grundlegenden und prototypischen Datenmodells sein. Dieses Datenmodell gibt vor, welche Informationen in Phase eins beim Erstellen eines Objektes vom Nutzer eingegeben und damit erfasst werden sollten, um die wichtigsten Daten zur Verfügung zu haben.

## 1.4 Aufbau der Arbeit

Nach den soeben genannten einleitenden Informationen widmet sich das Kapitel (2) den essentiellen und wichtigsten Grundlagen dieser Arbeit. Zu Anfang wird dem Leser der Terminus der Augmented Reality (2.1) offenbart, um allgemein Kontexte im Bezug zu dieser Arbeit zu begreifen, gefolgt von einer Einführung in die Thematik des Verfahrens SLAM-Simultaneous Localization And Mapping (2.2) der überbegrifflichen Materie der Robotik. Eine weitere notwendige Grundlage ist das Verständnis von Quaternonen (2.3) und die damit zusammenhängende Rotation und Translation von Objekten in einem dreidimensionalen Raum. Nach den erworbenen Grundkenntnissen der Basis-Thematiken, wird das Wissen über die Voraussetzungen und verwendeten Technologien (2.4) geschaffen. Darauf folgend wird im Allgemeinen auf Softwarearchitektur (2.5) und Modulare Software Architektur (2.5.1) eingegangen. Abschließend zu Kapitel (2) wird zu guter Letzt der Bereich der Datenmodellierung (2.6) thematisiert.

An Kapitel (2) anschließend wird in Kapitel (3) die Konzeption dargelegt. Anfänglich werden in diesem Abschnitt der Arbeit Gedanken, Überlegungen und vorläufige Konzeptionen der Arbeit aufgefasst und erläutert. Unter anderem welche Bedingungen die Arbeitsumgebung (3.1), in der die Applikation ihren Nutzen erweist, mit sich bringt. Erweiternd dazu wird darauf eingegangen, wie die beiden Phasen Scan-Phase (3.2) und Visualisierungs-Phase (3.3) konzipiert wurden.

Ebenso wird das Architekturkonzept (3.4), welches für das System vorgesehen war, genauesten dargelegt. Im Anschluss wird auf das ebenso tragende Softwarekonzept (3.5) eingegangen. Darauf folgend eine kurze Evaluierung wieso sich für das angewandte AR-Framework (3.6) entschieden wurde und abschließend zu Kapitel (3) die Intension des konzipierten Datenmodell's (3.7).

Kapitel (4) befasst sich mit der Umsetzung des Konzepts, dem Ablauf der Entwicklungsphase, den besonders erwähnenswerten Lösungen und den dabei aufgetretenen Problemen.

Nach der Umsetzung (4) gibt es eine Evaluierung (5), um das Projekt sach- und fachgerecht zu beurteilen und einen eigenen Fazit zu ziehen.

Die letzten zwei Kapitel, Fazit (6) und Ausblick (7), runden die Dokumentation ab und schließen die Arbeit. Die vorzuweisenden Ergebnisse werden analysiert und Verbesserungsvorschläge angemerkt. Der Ausblick gibt Aufschluss darüber, welche Erweiterungsmöglichkeiten es für diese Arbeit gibt und wie innovativ sich dieser Grundbaustein in Zukunft erweist.

## 1.5 Stand der Technik

Die moderne Technologie Augmented Reality findet in aller Munde Zuspruch und offenbart ein weites Spektrum an Einsatzmöglichkeiten. Speziell im Hinblick auf Augmented Reality in kombinierter Nutzung mit Smartphones ist dies eine *State of the Art*-Methode, um Interaktionen mit virtuellen Objekten zu fördern. Vor allem im Bereich der mobilen AR (siehe Abschnitt 2.1.2) gibt es nützliche Apps, die sich am Markt etabliert haben. Die nach aktuellem Stand am meisten aufsehenerregendsten Marktsektoren sind unter anderem Gaming, beispielsweise Pokémon Go, E-Commerce, unter anderem IKEA Place und die DHL Packset App, Bildung und Wissen, wie etwa GeoGebra 3D Grafikrechner und SketchAR und der Sektor Reisen, als weiteres Beispiel der AR gestützte Google Übersetzer. Auch die Umsetzung von Augmented Reality Navigation ist in kleineren Ausführungen durch beispielgebend Google entwickelte Anwendungen vertreten.

Im Bereich der Industrie gibt es hinsichtlich mobiler AR reichlich wenige Anwendungen die allseits bekannt sind, bzw. ein hohes Aufsehen erregt haben. Im Einsatz befinden sich hier verstärkt Smart-Glasses (siehe Abschnitt 2.1.2) und weniger der Einsatz von mobilen Endgeräten. Aufschluss über den allgemeinen Einsatz von AR in der Industrie wird in Kapitel 2 Abschnitt 2.1.4 gegeben.

Eine Anwendung die eine Räumlichkeit scannt, ausgewählte Objekte virtuell reproduziert und diese entstehenden Informationen dauerhaft zur Verfügung stellt, gibt es derzeit keine vergleichbaren. Die meisten Anwendungen, am Beispiel der IKEA Place Anwendung, beschränken sich auf bestimmte Positionen, bzw. darzustellende Objekte an den vorgesehenen Positionen und einer kurzen Gültigkeit und Lebensdauer. Damit ist die Langlebigkeit der Informationen adressiert. Die Objekte werden bei erneutem Aufruf neu generiert und platziert. Dies bedeutet, dass es keinerlei Anhaltspunkte für schon bereits verwendete Objekte gibt. Sie dienen damit lediglich der Ansicht, um die Vorstellungskraft des Menschen zu unterstützen.

Die hinter AR verborgene Technologie ist in den letzten Jahren durch intensivste Entwicklung und auch Forschung weit gekommen. Durch die anfängliche Integration von einfachen Bildern und Texten ist es heutzutage üblich, adaptive Anpassungen der gegebenen Informationen vorzunehmen. [QUERVEL 2020] Damit können sämtliche Objekte und Bewegungen erstellt werden. Auch ist es möglich, komplette Bewegungsmuster der Realität nachzuahmen und diese täuschend echt darzustellen.

Im Fokus der Augmented Reality gibt es mittlerweile unzählige SDKs und Frameworks, die alle unterschiedlichste Präferenzen haben. Darunter gibt es auch Frameworks der großen Firmen, die vielen ein Begriff sind, hierunter Google ARCore, ARKit von Apple, vuforia, Wikitude und ARToolKit. Diese sind ebenso die meist benutzten Frameworks zur Umsetzung von AR-Anwendungen. Google ARCore bietet eine tiefgreifende und ausführliche Application Programming Interface (API), welche in Kombination mit vielen weiteren SDKs verwendet werden kann, um eine beeindruckende User- als auch AR-Experience zu verschaffen. Demnach hat ARCore, neben Apple ARKit, zur Entwicklung einer Anwendung eine führende Position inne.

# Kapitel 2

## Grundlagen

In diesem Kapitel werden die für diese Bachelorarbeit notwendigen Grundlagen geschaffen, um ein fundiertes Wissen und Verständnis über verwendete Technologien zu bilden. Auf alle diese Informationen und Voraussetzungen wird im Folgenden eingegangen, um nachfolgende Konzeption und Umsetzung besser zu verstehen.

### 2.1 Augmented Reality

Eine der wichtigsten Grundlagen dieser Arbeit ist das Verständnis des Begriffs der Augmented Reality.

Augmented Reality, im Deutschen „*erweiterte Realität*“, ist eine durch den Computer gestützte Erweiterung der Realität, bzw. der menschlichen Wahrnehmung. Es ermöglicht dem Nutzer, die reale Welt mit Überlagerung oder Zusammensetzung virtueller Objekte und visueller Informationen zu sehen. Mittels einer Art Overlay werden diese Objekte und Informationen über die reale Welt gelegt und dem Nutzer zur Verfügung gestellt. Allgemein soll damit dem Nutzer ein weit gefächerter Überblick verschafft und Hilfestellung geleistet werden, alhn aber in keinerlei Interaktion mit der Umgebung einschränken. Die Definition, welche sich in der Wissenschaft weitestgehend durchgesetzt und etabliert hat, ist die Definition nach Azuma aus dem Jahre 1997.

„Augmented Reality (AR) is a variation of Virtual Environments (VE), or Virtual Reality as it is more commonly called. VE technologies completely immerse a user inside a synthetic environment. While immersed, the user cannot see the real world around him. In contrast, AR allows the user to see the real world, with virtual objects superimposed upon or composited with the real world. Therefore, AR supplements reality, rather than completely replacing it.“ [AZUMA 1997]

Ein Augmented Reality System verfügt nach [AZUMA 1997] über folgende drei charakteristische Merkmale:

1. Es kombiniert Realität und Virtualität.
2. Es ist interaktiv in Echtzeit.

### 3. Die virtuellen Inhalte sind im 3D registriert.

Das erste genannte Merkmal kombiniert die reale Welt mit dem oben genannten Overlay, der Überlagerung der Realität, um künstliche virtuelle Objekte und visuelle Informationen darstellen zu können. Dies bedeutet, der Nutzer nimmt die reale Umgebung gleichzeitig mit den darin liegenden virtuellen Objekten als ein Ganzes wahr. Daraus resultiert die Interaktion von virtuellen Objekten und Informationen mit der realen Welt in Echtzeit, damit sie als Teil der Realität registriert werden können. Das dritte Merkmal umfasst die Darstellung von Objekten als scheinbar reales Objekt. Mit dem letzt genannten Merkmal wird das Ziel verfolgt die projizierten, bzw. nicht realen Teile täuschend echt in die Umgebung zu integrieren.

Eine etwas allgemein formuliertere Definition ist die nach [DÖRNER 2019], welche die drei charakteristischen Merkmale besonders aufgreift:

„Augmentierte Realität (AR) ist eine (unmittelbare und interaktive) um virtuelle Inhalte (für beliebige Sinne) angereicherte Wahrnehmung der realen Umgebung in Echtzeit, welche sich in ihrer Ausprägung und Anmutung soweit wie möglich an der Realität orientiert, sodass im Extremfall (so dies gewünscht ist) eine Unterscheidung zwischen realen und virtuellen (Sinnes-) Eindrücken nicht mehr möglich ist.,“ [DÖRNER 2019]

Die oben aufgeführte Definition von Azuma dient Dörner als Grundlage.

Der Autor L. Frank Baum [GAEVERT 1856] verkündete die ersten Ideen und Gedanken einer Augmented Reality Anwendung in „*The Master Key*“ [BAUM 1996]. Eine erste tatsächliche Realisierung eines Augmented Reality Systems erfolgte erst über 60 Jahre später. Ivan Edward Sutherland [SUTHERLAND 1938] stellte sein Projekt 1968 an der University of Utah vor. Dabei handelte es sich um ein sogenanntes *Head-Mounted Display (HMD)*. Ziel dieser Entwicklung war weniger das Erweitern der Realität, sondern viel mehr das Erzeugen dreidimensionaler Illusionen, die reale Objekte mit einer einfachen Grafik in Echtzeit überlagern. Er gilt nichtsdestotrotz als erste Person mit der Vision, einen Nutzer in realer Umgebung mit virtuellen Objekten interagieren zu lassen. Anfang der 90er Jahre prägten zwei Forscher, Thomas P. Caudell und David W. Mizell, den Begriff der Augmented Reality durch ein Pilotprojekt bei Boeing. Das Projekt diente dazu, Informationen in das Gesichtsfeld über eine Brille einzusetzen, um Arbeitern das Verlegen von Kabeln im und um das Flugzeug zu erleichtern. Nach dieser bahnbrechenden Erfindung begann eine stetige Weiterentwicklung der Technologie. Im Jahre 1999 wurde von Hirokazu Kato und Mark Billinghurst *ARToolKit*, ein Computer-Vision-basiertes Tracking für AR, veröffentlicht und „löste eine große Welle an Forschungsarbeiten auf der ganzen Welt aus.“ [DÖRNER 2019]

We describe an augmented reality conferencing system which uses the overlay of virtual images on the real world. Remote collaborators are represented on Virtual Monitors which can be freely positioned about a user in space. Users can collaboratively view and interact with virtual objects using a shared virtual whiteboard. This is possible through precise virtual image registration using fast and accurate computer vision techniques and HMD calibration. We propose a method for tracking fiducial markers

and a calibration method for optical see-through HMD based on the marker tracking.  
[KATO und BILLINGHURST 1999]

Dieser Ausschnitt war der grundlegende Baustein des Durchbruchs dieser Technologie und den vorangestellten Forschungen und eine fundierte Grundlage für alle weiteren Forschungen und Entwicklungen, die darauf folgten.

Heutzutage dreht sich die Entwicklung vielmehr um mobile AR, welche durch die anfängliche Revolution von *ARToolkit* und die darauf entstehenden Entwicklungen und Produktionen von großen Firmen, wie z.B. Google, Microsoft, Apple und Facebook entstand. Die zuletzt große Bewegung in dem Bereich der AR waren die Vorstellungen großer Software-Plattformen für mobile AR-Applikationen, z.B. Windows Mixed Reality. Durch *Apple's ARKit* und *Google's ARCore* kamen im Jahr 2017 zwei moderne und innovative Frameworks auf den Markt, die die Entwicklung von Augmented Reality-Applikationen stark beeinflussten. Die Frameworks wurden bei den ersten Produktionen für Entertainment-Anwendungen genutzt, um z.B. mobile Spiele zur Unterhaltung oder Funktionen bei Sport-Fernsehübertragungen zur Anzeige der Entfernung des Freistoßes zu realisieren.

Diese Arbeit widmet sich ausschließlich dem industriellen Aspekt und stellt andere Bereiche der Augmented Reality in den Hintergrund. Die schon in Kapitel (1.1) aufgeführte Markstudie bestätigt das enorme Potential hinter Augmented Reality und deren Einsetzbarkeit in der Industrie. Hauptsächlich in der Produktion, der Wartung oder der Reparatur von Maschinen kann Augmentierte Realität eingesetzt werden und zeigt einen positiv erzeugten Mehrwert. Dabei können bei Maschinen das Anzeigen von protokollierten Fehlern oder eine visuelle Hilfestellung bei Defekts, sowohl bei der Reparatur, als auch bei der Ersetzung einzelner Komponenten eine deutliche Reduzierung des zeitlichen Aufwands oder eine effektivere Arbeitsweise vorweisen.

*Harvard Business Review* legte einen Vergleich offen, indem ein Techniker ein Steuergerät einer Windkraftanlage mithilfe eines AR-Headsets verkabelte und in Betrieb nahm. Alle benötigten Informationen wurden Schritt für Schritt über das Headset zur Verfügung gestellt. Das aufwändige Nachschlagen in einer Dokumentation entfiel. Zum Vergleich führte der Techniker den gleichen Prozess ohne die Hilfe der AR-Anwendung, lediglich unter Verwendung des vorliegenden, neben ihm befindlichen Handbuchs durch. Dieser Test bestätigte eine Leistungsverbesserung des Arbeiters beim ersten Gebrauch um 34%. [ABRAHAM und ANNUNZIATA 2017] Diese Erkenntnis des Tests stützte den Gedanken der Leistungsverbesserung und der Reduzierung des zeitlichen Aufwands, somit ließ dieses Resultat die Annahme zu, das vorliegende Ergebnis auf die Gesamtheit zu projizieren und für alle Anwendungsfälle allgemeingültig zu machen. Auf Basis der vorab getätigten Studie von Boeing, die sogenannte „*wing assambly study*“ in Kooperation mit der Iowa State University, wurden bei einer Vielzahl von Tests und Analysen eine Leistungsverbesserung von 30 % ermittelt. [AREA 2015]

### 2.1.1 Virtual Reality, Augmented Reality und Mixed Reality

Während immer mehr Leute mit dem Begriff Virtual Reality etwas anfangen können, gibt es doch noch viele Unsicherheiten bei den dazukommenden Begriffen der Augmented und der Mixed Reality. Diese drei Begriffe lassen sich meist nicht immer voneinander unterscheiden, da es viele Überschneidungen aber auch gravierende Unterschiede gibt.

Folgender Abschnitt beleuchtet die Unterschiede und lässt die drei Formen der erweiterten Realität voneinander unterscheidbar machen.

## Virtual Reality

Virtual Reality, dt. Virtuelle Realität (VR), ist eine in Echtzeit computergenerierte, interaktive und virtuelle Umgebung. Eine Darstellung und gleichzeitige Wahrnehmung der Wirklichkeit in all ihren Facetten und Eigenschaften. Das Ziel dieser Technologie ist, den Nutzer von der Außenwelt abzuschirmen und diese durch eine computergenerierte und detaillierte Welt zu ersetzen. [MÜHLROTH 2018] Auch bekannt als Immersion.

Die konventionelle Computergraphik ist für den Menschen spürbar nicht von belangen und weckt keine physischen Emotionen, wogegen VR diese etwas beeinflussen kann. Wie diese Unterschiede spürbar sind, wird in folgendem erläutert.

Die Tabelle 2.1 fasst die Unterscheidungsmerkmale von Virtueller Realität zur konventionellen Computergraphik zusammen. [DÖRNER 2019]

<b>3D- Computergraphik</b>	<b>Virtuelle Realität</b>
Rein visuelle Präsentation	Multimodale Präsentation
Präsentation nicht notwendigerweise zeitkritisch	Echtzeitdarstellung
Exozentrische Perspektive	Egozentrische Perspektive
Statische Szene oder vorberechnete Animation	Echtzeitinteraktion und -simulation
2D-Interaktion (Maus, Tastatur)	3D-Interaktion (Körperbewegung, -gestik)
Nicht-immersive Präsentation	Immersive Präsentation

Tabelle 2.1: Merkmale der Computergraphik gegenüber der VR [DÖRNER 2019]

Virtuelle Realität ist immer in Verbindung mit Head-Mounted Displays zu betrachten, da ein Gerät benötigt wird, welches den Nutzer von der realen Welt abschottet und in die virtuelle Welt begleitet. Diese werden auf Hochtouren von großen Firmen, wie Microsoft, Sony, Facebook etc. entwickelt. Die erste entwickelte und auf dem Markt veröffentlichte Brille war die HoloLens von Microsoft, gefolgt von der Brille Namens Oculus Rift von Facebook usw.

Mit der stetigen Weiterentwicklung dieser Brillen und der Technologie wird versucht nach und nach mehr Sinne des Menschen manipulieren zu können, bzw. das Spiel- und Gefühlerlebnis bei Konsolen-Spielen immer realistischer zu gestalten. Allerdings sind die Möglichkeiten im Massenmarkt stark beschränkt auf die folgenden aufgelisteten Sinne:

- Sehen: Durch Head-Mounted Displays (Oculus Rift), die die reale Welt abschirmen und vom Nutzer nicht mehr wahrgenommen werden kann
- Hören: Durch Kopfhörer, somit werden Geräusche der Realität übertönt
- Fühlen: Durch Controller mit haptischem Feedback, um Ereignisse der virtuellen Welt physisch spürbar zu gestalten.

Die Entwicklung dieser Technologie wird uns in Zukunft weiter begleiten. Vielleicht gibt es irgendwann die Möglichkeit, weitere Sinne virtuell zu steuern. An den Universitäten von Singapur und Tokio, gibt es beispielsweise Forscher-Teams, die ein großes Budget zur Verfügung haben, um dieser Sinnesträubung auf den Grund zu gehen. Sie fanden heraus, dass thermische und elektrische Stimulationen bestimmte Geschmackseindrücke vermitteln können. Auch Wissenschaftler in China und an der Oxford University haben herausgefunden, dass bestimmte Audiosignale mit einem süßen Geschmack assoziiert werden. [BREHM 2017] Forscher aus aller Welt gehen mit einer bestimmten Ernsthaftigkeit die Möglichkeiten der Sinneserweiterung in der virtuellen Realität an.

### **Augmented Reality**

Augmented Reality (AR) setzt im Gegenzug zu Virtual Reality (VR) auf das tatsächliche Erweitern der Realität durch das Einblenden von Informationen, Vorgängen, Hilfestellungen oder die Wegbeschreibung bei Head-Up Displays in Personen-, Kraft- und Nutzfahrzeugen, während Virtual Reality den Nutzer in eine völlig eigene Welt entlockt und von der Realität abkapselt. Bei AR soll dem Nutzer die zu bewältigenden Aufgaben vereinfacht und werden, ohne die Realität außer Acht zu lassen. Darüberhinaus bietet Augmented Reality deutlich mehr Ansatzmöglichkeiten diese Technologie umzusetzen, wie die z.B. schon erwähnten Head-Up Displays bei Autos und Flugzeugen, oder bei Smartphones und Tablets, die durch die Kamera die Realität erweitern, Brillen mit eingeblendeten Projektionen, wie die Oculus Rift und anderweitige große Projektionen.

In Kapitel (2.1.2) wird auf die Varianten genauer eingegangen.

### **Mixed Reality**

Dem Namen entsprechend ist Mixed Reality (MR) eine Mischung aus Augmented Reality und Virtual Reality, jedoch die Art und Weise, wie AR und VR vereint werden, ist dabei entscheidend. Es gibt MR-Brillen, die die reale Welt als Basis der Interaktion nehmen und Brillen die alleinig auf digitalen Bildern aufbauen. [SCHANZE 2018]

#### **Mixed Reality bezüglich der realen Welt**

Diese Art der erweiterten Realität basiert auf den gegebenen Grundlagen der AR. Ähnlich zu Augmented Reality werden Objekte der realen Welt hinzugefügt, indem sie an bestimmte Stellen projiziert werden. Mixed Reality baut darauf auf und verankert die digitalen Objekte mit dem realen dreidimensionalen Raum, sodass eine realitätsnahe Interaktion stattfinden kann. Virtuelle und echte Welt verschmelzen dadurch endgültig zu einer einzigen Welt.

#### **Mixed Reality bezüglich der virtuellen Welt**

Hinsichtlich der VR gibt es zu Anfang keine grundlegenden Änderungen gegenüber der MR. Hierbei kann der Nutzer durch MR ebenso die Außenwelt ausblenden und sich lediglich auf die virtuelle Wahrnehmung fixieren. Erst in der Benutzung werden die gravierenden Unterschiede zu VR sichtbar. Der Nutzer kann sich frei bewegen, d.h. die Bewegungsfreiheit wird nur durch die reale Umgebung eingeschränkt, während Virtual Reality sich auf einen sensorgestützten Raum begrenzt. [MÜHLROTH

2018]

Bei MR wird jeder Schritt und jede Bewegung in die computergenerierte Welt übertragen und schafft so deutlich mehr Bewegungsfreiheiten.

Ein großer Investor dieser Technologie ist Microsoft mit Windows Mixed Reality (WMR), wo bei bereits schon Erfolge erzielt werden konnten, die Entwicklung einer standhaften Plattform die nur darauf wartet, ausgebaut und intensiver genutzt zu werden. Für die Plattform vorgesehene Brillen gibt es schon viele Produzenten, wie z.B. die *HMD Odyssey* von Samsung oder der *Explorer* von Lenovo.

### 2.1.2 Varianten der Augmented Reality

Augmented Reality Anwendungen funktionieren alle nach dem gleichen Prinzip und verfolgen das gleiche Ziel, die Realität durch digitale Informationen oder Objekte zu erweitern. Die einzige deutliche Abweichung ist das jeweilige Endgerät und die technische Umsetzung in Zusammenhang mit der Hardware. Auf zwei dieser Varianten, nämlich dem mobilen und dem Smart-Brillen- oder auch sogenannten Headset-AR, wird im Folgenden eingegangen.

#### Mobiles AR

Smartphones sind in unserer Gesellschaft nicht mehr wegzudenken und haben sich fest etabliert. Durch die vielseitige und alltägliche Nutzung von Tablets und Smartphones eröffnete diese Sparte eine gute Möglichkeit, Augmented Reality in den Alltag zu integrieren. Somit belebt der ständige Gebrauch dieser Technologie nicht aus und bereichert die Art und Weise Spiele und Anwendungen zu entwickeln. Sowohl die software-, als auch hardwaretechnischen Fortschritte eines Smartphones zeigen eine deutlich Steigerung, um solche mobilen AR-Anwendungen problemlos entwickeln zu können. Durch die im Smartphone integrierte Kamera werden Live-Aufnahmen analysiert und dienen als Ausgangssituation für die AR-Anwendung. Die gegebenen Möglichkeiten der vorhandenen Kamera, kann einen Echtzeithintergrund erzeugen und zusätzlich mit Informationen per Overlay darstellen. Je nach Anwendung kann der Nutzer auf verschiedene Weise mit den eingeblendeten und virtuellen Objekten interagieren, z.B. durch das Bewegen des Geräts, um das Objekt aus verschiedenen Blickwinkeln anschauen zu können, oder der direkten Interaktion mit dem Objekt durch Drehen, Skalieren oder Verschieben am Bildschirm. Ein Vorreiter der mobilen AR ist das 2016 auf dem Markt erschienenen *Pokémon Go* von Niantic, das den Ansatz der AR prägt. [FISCHBACH 2016]

Die weit verbreitete social Media Applikation *Snapchat* baut seit geraumer Zeit ebenso auf Augmented Reality, um Bilder lebhafter zu gestalten, wie der folgenden Abbildung (2.1) zu entnehmen ist.

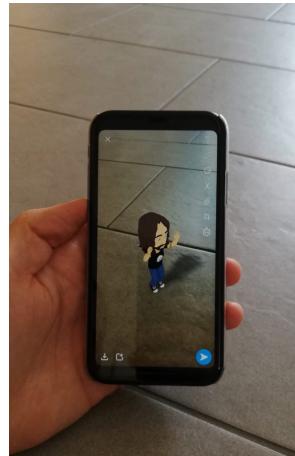


Abbildung 2.1: Mobile-AR in Snapchat

### Smart-Brillen AR

Unter Smart- oder Datenbrillen und Smart Glasses wird ein Konstrukt verstanden, das eine Brille mit einem tragbaren Minicomputer verwirklicht. Dabei werden Informationen über kleinste Monitore oder Prismen ausgegeben und dem Nutzer zusätzlich in das Sichtfeld projiziert. Im Gegensatz zur mobilen AR hat der Nutzer kein Gerät in der Hand und ist somit in seiner Bewegung weniger eingeschränkt und flexibler.

Die Technologie der Smart-Brillen ist ein ähnliches Konzept zu der VR-Brille, allerdings befindet sich die Smart-Brille für den öffentlichen Gebrauch noch in der Entwicklungsphase, da nur bedingte Funktionen möglich sind und die Kosten für den Normalverbraucher nur bedingt tragbar erscheinen. Mit der überarbeiteten Version der *HoloLens 2* von Microsoft ist eine deutlich komfortablere und für den alltäglichen Gebrauch geeigneteren Ausführung entwickelt worden, die im Vergleich zum Vorgängermodell deutlich praktischer, leichter und handlicher ist. Das Design der Datenbrille ist von einer normalen Brille noch weit entfernt, ermöglicht mittlerweile aber das uneingeschränkte Sehen und Wahrnehmen der Umgebung. Die Bedienung des Geräts basiert auf schon vorhandenen Möglichkeiten die bereits in anderen Geräten Anwendung finden. Darunter gibt es am Gerät angebrachte Touchsensoren, Handbewegungen und -gesten und Eye-Tracking.

Speziell im industriellen Bezug bieten die neu entwickelten Brillen ein akzeptables Gewicht, so dass diese ohne große Probleme dauerhaft tragbar sind und den Mitarbeiter bei seiner Arbeit nicht übermäßig einschränken.

Die Programmierung solcher AR-Brillen laufen häufig über eigene SDKs und plattformunabhängige Laufzeit- und Entwicklungsumgebungen, z.B. Unity oder Unreal Engine. Diese gelten als führende Produkte im Bereich der 3D-Echtzeitdarstellung. Ein marktführendes Produkt ist unter anderem die Microsoft HoloLens 2, die der folgenden Abbildung zu entnehmen ist.



Abbildung 2.2: Test der HoloLens 2



(a) Google Glass 2

(b) HoloLens 2

Abbildung 2.3: Datenbrillen (HMD)

Eine große Herausforderung der Augmented Reality ist die Bestimmung der Position, in der Daten und Objekte projiziert werden. Auf die Ansätze, diese Herausforderung zu bewältigen, wird in folgendem Abschnitt 2.1.3 eingegangen.

### 2.1.3 Positionsbestimmung

Um ein digitales Objekt als Overlay dem Kamera-Live-Bild hinzuzufügen, werden genauestens definierte Positionen benötigt. Diese Positionen können durch unterschiedliche Ansätze ermittelt werden. Je nach Anwendungsfall, z.B. als Navigation, Routenplaner oder Google Maps „Live

*View*“ [BERGER 2019], reicht eine etwas ungenauere Positionsbestimmung per GPS, da in Relation zur realen Welt eine Abweichung um Zentimeter oder wenige Meter nicht von Belang ist. Bei Positionsbestimmungen auf kleinstem Raum ist eine genaue Lokalisierung wichtig und basiert auf einem deutlich präziseren Ansatz.

Welche oben genannten Ansätze es gibt und welche Unterschiede zu beachten sind, wird in Folgendem näher erläutert.

### Marker-basierte Positionsbestimmung

Speziell bei der Marker-basierten Positionsbestimmung gibt es verschiedene Möglichkeiten den Marker zu gestalten. Es können Binär- oder QR-Codes als Markierung verwendet werden, ein Beispiel eines solchen Codes ist der Abbildung 2.4 zu entnehmen. Diese Codes sind meistens quadratisch und haben ein eindeutiges Zeichen in der Mitte. Um die Rechenzeit gering zu halten gibt es einfache Muster, wie die Abbildung 2.4 zeigt.

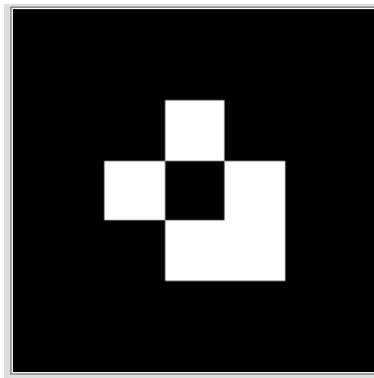


Abbildung 2.4: Marker-basierte Augmented Reality Positionsbestimmung

Neben der einfachen Markierungserkennung gibt es eine weiterentwickelte Möglichkeit der Bild- sowie der Objekterkennung. Diese sind Ansätze die grundlegend auf dem Ausgangspunkt des Binär-Code-Verfahrens aufbauen. Eine detailliertere Erläuterung der soeben genannten Erkennungsmöglichkeiten findet im Rahmen dieser Arbeit nicht statt, allerdings wird allgemein kurz auf die Funktionsweise einer Marker-basierten Positionsbestimmung eingegangen.

Durch ein Kamerabild wird nach einem vordefinierten Marker, bzw. nach einer festgelegten Markierung gesucht. Ist diese mit der Kamera erfasst, wird die Markierung durch Bildverarbeitungsalgorithmen und bestimmter Filterung eindeutig identifiziert. Mit den gewonnenen Informationen und der Übereinstimmung des angegebenen Codes wird die Position, sowie die Orientierung des Markers berechnet. Mit den Angaben der Lage und Orientierung wird auf der Markierung das anzuseigende digitale Objekt generiert und als Overlay über dem Kamerabild angezeigt. Die Markierung dient sozusagen als Grundlage um den digitalen Gegenstand überhaupt anzeigen zu können.

Da der Marker immer im Blickfeld der Kamera sein muss, um das virtuelle Objekt anzuzeigen, bringt diese Art der Positionsbestimmung eine enorme Einschränkung mit sich, die in diesem Bezug unumgänglich ist.

Ein weiterer Ansatz der Positionsbestimmung ist als Überbegriff das Gegenstück zur oben aufgeführten Lokalisierung von Markern, die sogenannte Marker-unabhängige Positionsbestimmung, welche ebenso verschiedene Ausführungen vorweist.

### GPS-basierte Positionsbestimmung

Die Methode des GPS-basierten Positionsbestimmungsverfahren verwendet hauptsächlich die Koordinaten der realen Welt. In Zusammenarbeit mit zusätzlichen im Anwendungsgerät verbauten internen Sensoren, bspw. Positions-, Geschwindigkeits- und Beschleunigungssensoren und Teile des Inertial Navigation System (INS), z.B. dem Gyroskop, kann diese Art der Positionsbestimmung optimal Anwendung finden. Allerdings werden dabei deutlich mehr Komponenten benötigt und ist deutlich komplexer umzusetzen, als ein markerbasiertes System.

Bei einem Anwendungsfall von GPS-basierter Positionsbestimmung geht es meist um Routenplaner, Navigation oder Szenarien die sich auf offenen Flächen abspielen, da im Gegensatz zu Marker-basierten Anwendungen das Größenverhältnis deutliche Unterschiede vorweist. Der Benutzer ist nicht auf einen bestimmten Bezirk beschränkt und ist nicht auf die millimetergenauer Darstellung angewiesen.

Eine Alternative zu GPS ist die Anwendung des SLAM-Verfahrens. Dabei wird eine virtuelle Karte, bzw. ein geometrisches Modell der Umgebung erstellt. Wichtige Grundlagen zum Erzeugen eines solchen Modells sind eigenständig gefundene Landmarken die gleichzeitig lokalisiert werden. Darauf folgt ein Vergleich der Pose, der Position des Geräts und der geschätzten Karteninformationen aus dem Scan der Umgebung. Damit sind im erweiterten Sinne Marker geschaffen, die Anhaltspunkte für Augmented Reality-Interaktionen schaffen.

In Kapitel 2.2 wird die Thematik des SLAM-Verfahrens erläutert.

### 2.1.4 Augmented Reality in der Industrie

Das weit gefächerte Portfolio der Augmented Reality umfasst viele Anwendungsbereiche und Fachgebiete. Selbst dem übergeordneten Bereich der Industrie gibt es viele verschiedene Einsatzgebiete. Aus den vielen Möglichkeiten der Anwendung haben sich über die Jahre der Entwicklung der Technologie einige Gebiete in der Industrie herauskristallisiert, die besonders großen Nutzen davon haben. [SCHART 2017] In den Bereichen Instandhaltung und Wartung, Betrieb und Training ist Augmented Reality auf bestem Wege, fester Bestandteil des Alltags zu werden. Bei dieser Betrachtung ist es sinnvoll, sich auf die damit einhergehenden Lösungen, die Reduktion der Kosten und dem Zeitaufwand, sowie die Verbesserung der Sicherheit, zu fokussieren. [MUNDT 2020]

Bei einer Wartung oder Reparatur einer Maschine sind notwendige Informationen direkt greifbar und werden Schritt für Schritt angezeigt, sodass in bestimmten Situationen selbst ein Lehne die Anweisungen befolgen könnte. So werden zusätzliche Recherchearbeiten oder Unklarheiten über Vorgänge aus dem Weg geräumt. Ein Mitarbeiter kann so mit einem Tablet oder einer Smart-Glas Anweisungen visuell auf die realen Maschinen projizieren und Arbeitsschritte im Sichtfeld anzeigen lassen. Ebenso geht dieser Vorgang auch bei der Produktion von Bauteilen o.ä., indem eine AR-Anwendung Anweisungen und Prozessschritte, z.B. auf das Werkstück oder Produkt, projiziert. Die Inbetriebnahme oder Bedienung einer komplexen Anlage oder Maschine ist nach herkömmlichen Standard enorm zeitintensiv und dadurch können zusätzlich viele Fragen auftreten, die meist den Prozess noch länger gestalten als vorgesehen. AR-Anwendungen können Bedienungsanleitung oder -hilfen digitalisiert, indem Informationen, Inhalte oder Bedienelemente durch Augmented Reality auf der Anlage platziert werden.

In Fortbildungen, Schulungen oder Einarbeitungen in neue Geräte kann AR durchaus von großem Vorteil sein. Mit wenig Aufwand, einem effektiveren Training können Schulungen interaktiver und vor allem sicherer abgehalten werden. Durch die Visualisierung der Trainings- und Schulungsinhalten verbessert sich der Lernprozess und damit wird auch die Nutzung der Maschinen verständlicher. [SCHART 2017]

## 2.2 SLAM - Simultaneous Localization And Mapping

Als Simultaneous Localization and Mapping, auch SLAM-Problem genannt, bezeichnet man die Aufgabe, die Trajektorie<sup>1</sup> samt Orientierungsinformation einer sich bewegenden Plattform, z.B. ein Smartphone, Tablet oder jegliche Art von Roboter, aus Beobachtungen zu schätzen und gleichzeitig aus den gewonnenen Informationen eine Karte der Umgebung zu erstellen. Diese Aufgabe ist für den weiteren Prozess bedeutend. Zum einen sollen die generierten Karten sehr präzise sein, um einen hohen Wert für den Nutzer oder für spezielle auf der Karte aufbauende Anwendungen darzustellen. Zum anderen benötigen autonome Roboter, beispielsweise Saug- oder Mähroboter, ein solch erzeugtes geometrisches Modell der Umgebung, um zielgerichtet selbstständig navigieren zu können.

Das Simultaneous Localization and Mapping oder kurz SLAM Problem behandelt das gleichzeitige Schätzen der Position und Ausrichtung einer mobilen Plattform im Raum anhand der sich an Bord befindlichen Sensoren sowie den Aufbau eines Modells der Umgebung. Dieses Problem ist von großer praktischer Relevanz und ist Kernbestandteil der meisten mobilen Sensorsysteme. [FREEDEN und RUMMEL 2016]

1986 wurden auf der *IEEE Robotics and Automation Conference* erste mathematische Definitionen vorgenommen, die mittels statischer Theorien ermittelt und mit ersten Studien belegt wurden. Einige Jahre später, im Jahr 1995, wurde das SLAM Problem erstmals auf dem internationalen Symposium für Robotikforschung (*ISRR'95*) vorgestellt. Die Forschungen hielten an, bis auf der *ISRR'99* die erste SLAM Sitzung stattfand.

### 2.2.1 Definition des Problems

Angenommen ein Roboter startet in einer Position, auch Pose genannt, und einer Konfiguration  $p_0$  und bewegt sich durch eine ihm unbekannte Umgebung, stellen die nicht vorhandenen Kenntnisse seiner Position und der damit verbundenen Orientierung das Hauptproblem dar. Die Einstellung beinhaltet Position und Ausrichtung des Roboters. Je nach Bewegung in Raum oder Ebene ist die Pose meist als 3- oder 6-dimensionaler Vektor abgebildet. Die Bewegung des Roboters wird durch bekannte Kontrollkommandos  $u$  angewiesen, allerdings mit einer gewissen Unsicherheit versehen. Dabei wird zwischen den Zeitpunkten  $t-1$  und  $t$  die Bewegung des Roboters mit  $ut$  beschrieben und somit auch die unterschiedlichen Posen  $pt-1$  nach  $pt$ . Die Umgebung wird parallel dazu über diverse Sensoren, z.B. interne Sensoren, bspw. Geschwindigkeits- oder Positionssensoren, und externer Sensoren, unter anderem Abstands- oder taktile Sensoren. Neben der Protokollierung der Position, bei der es zu Störungen oder Berechnungsfehlern kommen kann, gibt es Beobachtungen durch Sensoren die verrauscht, bzw. fehlerhaft sind und durch  $zt$  angegeben werden.

Mit diesen vorhandenen Werten ist die Schätzung der Trajektorie  $p0:T=[p0,p1,\dots,pT]/T$  des Roboters von Beginn der Fortbewegung bis zum Zeitpunkt  $T$  das Ziel. Gleichzeitig zur Berechnung der Trajektorie wird eine Karte  $m$  des Umfelds geschätzt, deren Darstellung den Anforderungen entsprechend angepasst werden kann. Mit den Anforderungen sind verschiedene Repräsentationen der Karten gemeint, darunter beispielsweise eine Veranschaulichung von Punktansammlungen an Gegenständen,

---

<sup>1</sup>Lösungskurve oder Bewegungspfad eines Objekts

gerenderte Oberflächenmodelle oder 2D-Rasterkarten und verstärkt visualisierte 3D-Voxelkarten<sup>2</sup>. Basierend auf den Sensormessintervallen  $z1:T$  und den dabei stattfindenden Kontrollkommandos  $u1:T$  wird die Karte des Umfeldes und alle Positionen bestimmt. Die Wahrscheinlichkeitsverteilung  $p(p0:T, m/z1:T, u1:T)$  wird durch die geschätzten Werte  $p0:T$  und  $m$  berechnet.

Die Berechnung der Wahrscheinlichkeitsverteilung ist auch unter dem Namen *Offline-SLAM* bekannt. In der Praxis ist allerdings die Schätzung der Position  $xt$  und der Karte der Umgebung durch  $p(pt, m/z1:t, u1:t)$  interessanter, da Roboter Entscheidungen basierend auf aktuellen Informationen, z.B. der Posenschätzung und dem Umgebungsmodell, treffen sollen. Die Variante der Echtzeitschätzung ist auch als *Online-SLAM* bekannt. [FREEDEN und RUMMEL 2016]

### 2.2.2 Localization

Damit das Endgerät, Smartphone oder der Roboter seine Position in Erfahrung bringen und schätzen kann, werden Informationen und Möglichkeiten benötigt, die Bewegung in irgendeiner Form zu messen. Da das Nutzergerät eine eigene virtuelle Karte, unabhängig von der GPS-basierten Position, generiert, ist das *Tracking* über die Weltkoordinaten nicht Bestandteil des eigentlichen Verfahrens. Für die Erfassung der internen Systemzustände gibt es sogenannte interne Sensoren die in dem Roboter zur Verfügung stehen. Bestandteile dieser internen Sensorik sind unter anderem Positions-, Geschwindigkeits-, Beschleunigungssensoren und das Inertial Navigation System. Für die Positions- und Geschwindigkeitserkennung gibt es z.B. einen optischen Codierer, welcher durch Lichtimpulse die Geschwindigkeit als auch die zurückgelegte Strecke schätzen kann. Das INS besteht aus Lagesensoren und einem Kreiselkompass (Gyroskop). Diese sind essentiell für die Bestimmung der Orientierung und Neigung des Geräts. Ausgehend von der Erdkugel beziehen sich diese Sensoren auf das gegebene inertiale Koordinatensystem. Mittels *Odometrie* können die von den Sensoren bereitgestellten Informationen über Position und Orientierung berechnet werden. Radgetriebene Systeme führen die Berechnung durch, da diese basierend auf dem Durchmesser des Rades ermittelt wird. In Kombination mit *Koppelnavigation*<sup>3</sup> gilt dieses Verfahren für Roboter, bzw. Fahrzeuge auf Land, als Grundlage der Navigation. Durch die theoretische Berechnung werden Fehlerbetrachtungen, z.B. Verschleiß, Schlupf oder Unrundheit von Rädern, vernachlässigt und ist somit nicht als alleiniges Verfahren einzusetzen.

$$\Delta U = \frac{\pi D}{nC} N \quad (2.1)$$

Mit  $D$  als Raddurchmesser,  $n$  als Getriebeübersetzung,  $C$  als Enkoderauflösung und  $N$  als Anzahl der Enkoderimpulse wird die Positions- und Orientierung berechnet.

Smartphones berechnen die Fortbewegung über gegebene Sensoren, darunter Beschleunigungs- und Neigungssensoren und Gyroskop. Dieses Zusammenspiel an Sensoren ermöglicht die Wahrnehmung der Positionsveränderung und kann somit diese bestimmen.

---

<sup>2</sup>(Zusammensetzung aus dem englischen volume *vox* und elements *el*), bez. einen Gitterpunkt in einem dreidimensionalen Gitter.

<sup>3</sup>Engl. dead reckoning, ist die laufende Positionsbestimmung eines bewegten Objekts infolge der Bewegungsrichtung und Geschwindigkeit [GEOFFREY 2019]

### 2.2.3 Mapping

Um neben der Berechnung der Position durch die interne Sensorik die Umgebung registrieren und schätzen zu können, gibt es externe Sensoren, die sich mit der Erfassung der Umwelt beschäftigen. Auch hier gibt es viele Arten von Sensoren, mit denen es ermöglicht wird, die Umgebung wahrzunehmen. Darunter sind taktile Sensoren, Näherungs-, Abstands-, Positionssensoren und visuelle Sensoren. Hinsichtlich Abstandssensoren, die Messungen zwischen Gegenstand und Sensor durchführen, gibt es im Allgemeinen erhebliche Vorteile gegenüber den Näherungssensoren. Unter Anderem besitzen sie zum Beispiel eine größere Reichweite und ein größeres Blickfeld als Näherungssensoren. Somit können sie die Entfernung zu Gegenständen genauer ermitteln und geometrische Umweltinformationen besser erfassen. Für die Messung des Abstandes eignet sich die Time of flight (TOF)-Kamera, die mit dem Laufzeitverfahren Distanzen messen und berechnen kann.

Das Laufzeitverfahren funktioniert wie folgt:

$$d = \frac{1}{2}ct \quad (2.2)$$

Abstand zwischen der Zielfläche und dem Sensor  $d$  ist das Produkt aus der Signalgeschwindigkeit  $c$  und der messbaren Laufzeit  $t$ . Beispiele für solche TOF-Kameras und deren Repräsentation sind folgender Abbildung (2.5) zu entnehmen.

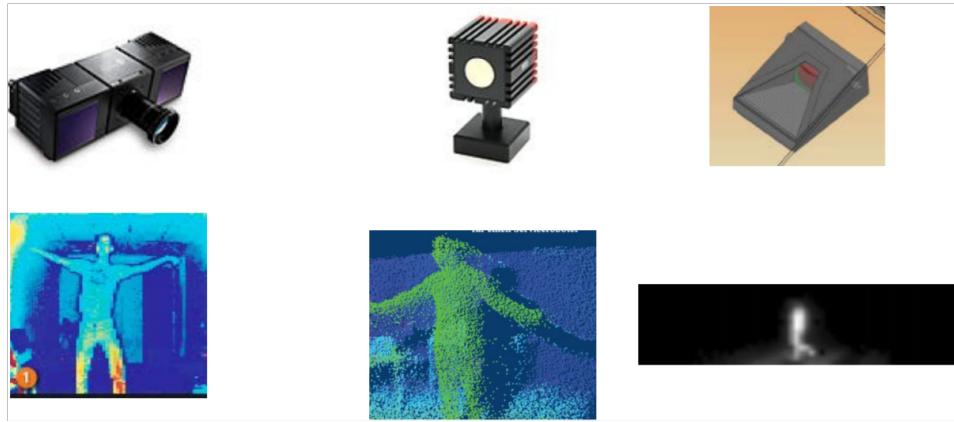


Abbildung 2.5: Time-of-Flight Kamera und deren Repräsentation [STRAND 2020a]

Auf der linken Seite ist eine PMD-Kamera zu sehen, gefolgt von einer SwissRanger-Kamera und auf der rechten Seite eine IFM-Kamera.

### 2.2.4 Verfahren zur Lösung des SLAM Problems

Das Lösen des SLAM Problems wird in der Praxis mit den folgenden Verfahren, die sich bei der Anwendung bewährt haben, durchgeführt. In den folgenden Abschnitten werden zwei dieser Verfahren erläutert, sodass ein Grundverständnis der Funktionsweisen der Verfahren entsteht. Als

Erstes erfolgt eine nähere Erläuterung des graph-basierten SLAM Verfahrens, gefolgt von der Lösung des Problems mit einem rekursiven Ansatz, dem erweiterten Kalman Filter (EKF).

### Graph-basiertes SLAM Verfahren

Grundlegend wird bei dem Algorithmus des graph-basierten Verfahrens, während der Bewegungsaufzeichnung des Roboters, ein Graph modelliert, dessen Positionen zu verschiedenen Zeitpunkten durch Knoten dargestellt werden. Alle nebeneinander liegenden Positionen, bzw. Punkte, die lediglich minimale Unterschiede in der Umgebung aufweisen, werden zu einem Punkt gebündelt. Diese korrespondierenden Positionen, bzw. Knoten im Graphen werden über eine Kante verknüpft. Gebunden an die Bewegung des Roboters werden die Kanten modelliert. Der folgende Graph (2.6) veranschaulicht ein solches Modell.

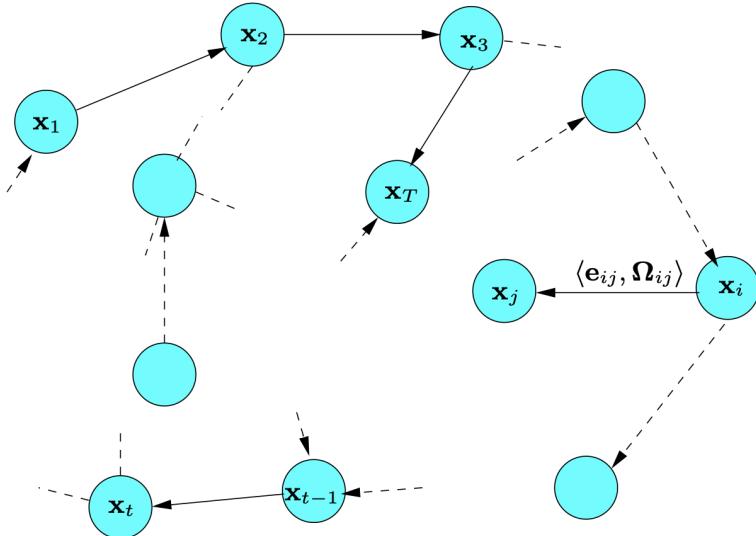


Abbildung 2.6: Graphische Darstellung des SLAM Verfahrens [GRISSETTI u. a. 2010]

Der Abbildung ist zu entnehmen, dass der Graph auch Kanten darstellt, die nicht den folgernden Knoten ansprechen, sondern eine Position vertreten, die in ihrer Beobachtung auf korrespondierende Knoten zurückgeht. Durch wiederholte Scans können genauere Merkmale und Bedingungen festgelegt werden, um die daraus resultierenden Positionen besser differenzieren zu können. Bei Verwendung von Kameras können genauer identifizierten Merkmale die geschätzten relativen Orientierungen sein, die durch eine zusätzliche Funktion berechnet werden können. Bei der Nutzung eines Lasersensors wird meist der Iterativ Closest Point (ICP) Algorithmus angewendet, um die Veränderungen zwischen den Aufnahmepositionen wahrzunehmen. ICP ist ein iteratives, ständig wiederholendes Verfahren, das die korrespondierenden Punkte schätzt und solange transformiert, bis diese unter einem gewissen Schwellwert liegen. Korrespondierende Punkte sind die, die den kleinsten Abstand zueinander haben (Closest Point). [STRAND 2020b]

Der inkrementelle Ansatz des graph-basierten SLAM Verfahrens ist ursprünglich ein *offline-Verfahren*,

welches über die Jahre schrittweise zu einem *online-Verfahren* führte. Durch diese ständige Beobachtung und Neuberechnung der aktuellen Position zu jedem Zeitpunkt wurde die Bedeutung der inkrementellen Verfahren gesteigert.

### Rekursives Schätzproblem

Der erweiterte Kalman-Filter (EKF) ist eine wahrscheinlichkeitsbasierte rekursive Schätzung der Position des Objekts und der Position der Landmarken unter Nutzung linearer Bewegungsmuster basierend auf dem Bayes-Filter. Der Bayes-Filter-Algorithmus ist das generelle Verfahren der rekursiven Zustandsschätzung und wird über den Satz von Bayes<sup>4</sup> hergeleitet.

Unter der Voraussetzung einer Normalverteilung und eines linearen Bewegungsmodells, worauf der Kalman-Filter setzt, kann diese durch das SLAM Problem nicht erfüllt werden. An dieser Stelle wird der erweiterte Kalman-Filter eingesetzt, da dieser nicht-lineare Bewegungsmodelle, bzw. Funktionen mittels Taylorreihe approximiert. Somit kann eine Messung in etwas vorhergesagt werden. Diese Schätzung dient somit als Grundlage zur eigentlichen Messung. Nach der Berechnung wird der Zustand aktualisiert und ausgegeben. Mit diesem Ergebnis werden die darauffolgenden Punkte rekursiv berechnet, bis die Scan-Phase zu Ende ist.

## 2.3 Quaternionen

Quaternionen, lat. „*Menge der Vier*“, wurden erstmals 1843 von Sir William Rowan Hamilton beschrieben. Sie bezeichnen einen Zahlenbereich, der die reellen Zahlen erweitert. Die mathematische Formel wird häufig zur Darstellung einer Drehung im dreidimensionalen Raum verwendet. Die Theorie der Quaternionen basiert auf der mathematischen Grundlage der komplexen Zahlen, die 1833 von Hamilton als Bildung einer Algebra galten und werden daher als hyperkomplexe Zahlen aufgefasst.

Mit  $a, b, c, d \in q$  werden Quaternionen wie folgt beschrieben:

$$q = a + b \cdot i + c \cdot j + d \cdot k \quad (2.3)$$

In oben genannter Formel ist die Variable  $a$  der Realteil und  $(b, c, d)^T$  der Imaginärteil  $u$  der Gleichung. Auch geschrieben als  $q = (a, u)^T$ , um die Teile differenzieren zu können.

Mit der sogenannten *Brougham Bridge*-Formel ist es möglich Vektoren zu dividieren.

$$i^2 = j^2 = k^2 = ijk = -1 \quad (2.4a)$$

$$ij = -ji = k \quad (2.4b)$$

$$jk = -kj = i \quad (2.4c)$$

$$i^2 = j^2 = k^2 = j \quad (2.4d)$$

---

<sup>4</sup>Mathematischer Satz der Wahrscheinlichkeitstheorie. Darunter wird die Berechnung einer bedingten Wahrscheinlichkeit beschrieben

### 2.3.1 Rotation

Quaternionen dienen als Möglichkeit, Drehungen, bzw. Rotationen darzustellen, unter anderem wegen der reduzierten Anzahl an benötigten Parametern im Vergleich zur Berechnung von Rotationsmatrizen und deren Translation. Die Quaternionen bieten eine deutlich kompaktere Darstellung der Rechenwege. Somit können Rotationen intuitiver veranschaulicht werden, d.h. es sind direkte Angaben von Drehwinkel und -achse möglich. Die kompaktere Darbietung beinhaltet lediglich vier Werte im Vergleich zu den neun Werten der Berechnung der Rotationsmatrix und die Rotation kann um eine bestimmte Achse ermittelt werden. Zudem ist eine sehr hohe numerische Stabilität gewährleistet.

Der Drehwinkel wird wie folgt berechnet:

$$\theta = 2 \cdot \arccos a \quad (2.5)$$

Der Realteil  $a$  des Quaternions wird mittels Arkuskosinus berechnet, so erhält man die Drehung des Objekts. Die Drehachse lässt sich bestimmen, indem alle Werte des Imaginärteils  $(b, c, d)^T$  mit  $u$  addiert werden. Die Variable  $u$  setzt sich wie folgt zusammen:

$$u = \sin w, \rightarrow w = \frac{t}{2}, \rightarrow t = \theta \quad (2.6)$$

Mit diesen zwei Berechnungen werden Drehwinkel und Drehachse bestimmt und können für weitere Berechnungen verwendet werden.

## 2.4 Technologien

Dieser Abschnitt befasst sich mit den wichtigsten Programmier-Technologien, die in diesem Projekt eingesetzt werden.

Der Augenmerk liegt auf dem Framework *Google ARCore*, da dieses Hauptbestandteil des Projekts ist.

### 2.4.1 Google ARCore

ARCore ist die von Google veröffentlichte Plattform zum Erstellen von Augmented Reality Erlebnissen, primär entwickelt für Android Phones. Das Software Development Kit (SDK) ist in den Programmiersprachen Java und Kotlin geschrieben. Mit bestimmten APIs ermöglicht ARCore die Erfassung der Umgebung durch ein Tablet oder Smartphone, um so die Welt zu verstehen. Mitte des Jahres 2017 wurde die *Google ARCore API* zum ersten Mal den Entwicklern und der AR-Community präsentiert. Im ersten Quartal 2018 wurde diese dann veröffentlicht und löste das Vorreiterprojekt „*Project Tango*“ ab.

*Project Tango* war ebenso eine Google-Plattform, mit dem Smartphones und Tablets ein „Raumgefühl“ erhalten. [ELGAN 2016] Die Plattform kombinierte die Eingabe verschiedener Sensoren, z.B. radarähnliche Infrarotstrahler, einer Infrarotkamera und hochpräzisen Beschleunigungsmessern, Barometern und Gyroskopen, um schnell nutzbare Informationen der Umgebung zu generieren.

Mit der Überarbeitung des Konzepts wurde das neue unabhängige Projekt *ARCore* geschaffen und somit die Anzahl der Hardwarekomponenten reduziert, bspw. wird keine spezielle Infrarotkamera mehr benötigt. Die ARCore API verwendet bei Anwendungen ausschließlich das Kamerabild des Smartphones und dessen Sensoren.

Zum Integrieren von virtuellen Objekten in die reale Welt verwendet ARCore drei Schlüsselfunktionen unter Benutzung von Java und OpenGL, Unity und Unreal:

- Motion tracking, dt. Bewegungsverfolgung, ermöglicht das Verstehen und Verfolgen der Position des Geräts relativ zur Realität.
- Environmental understanding, dt. Verständnis der Umgebung, erkennt die Position und Größe aller Arten von Oberflächen: horizontale, vertikale und abgewinkelte Oberflächen, wie Wände, Böden, Tische oder Stühle.
- Light estimation, dt. Lichtschätzung, schätzt die durch das Smartphone gegebenen Lichtverhältnisse der Umgebung. [LANHAM 2018]

### Motion tracking

Für die Funktion der Bewegungsverfolgung verwendet ARCore das Verfahren von Simultanious Localization And Mapping (SLAM) 2.2, um so zu verstehen wo sich das Smartphone in Relation zur realen Welt befindet. Zusätzlich erkennt ARCore visuell unterschiedliche Merkmale im aktuell aufgenommenen Kamerabild, sogenannte Merkmalspunkte „*feature points*“. Anhand dieser Punkte wird die Änderung des Standortes berechnet. Durch Trägheitsmessungen der Inertial Measurement Unit (IMU), dt. Inertiale Messeinheit des Geräts, kombiniert mit den visuellen Informationen des Kamerabildes wird die Position und Ausrichtung der Kamera in Relation zur Realität abgeschätzt.

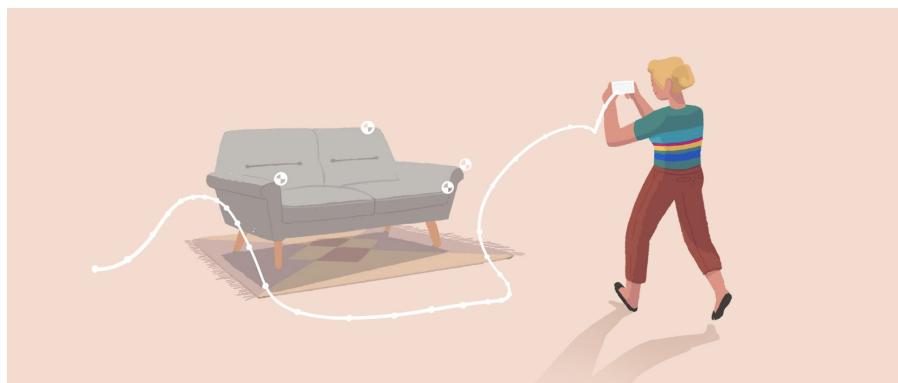


Abbildung 2.7: Skizze zur Bewegungsverfolgung des Smartphones [ARCORE 2020]

Die Inertial Measurement Unit in Smart-Devices besitzt drei Beschleunigungssensoren, z.B. piezoelektrische Beschleunigungssensoren oder Micro-Electro-Mechanical Systems (MEMS), für die drei Raumachsen:

- Die Abszissenachse (X-Achse), die horizontale (waagerechte) Koordinatenachse,
- Die Ordinatenachse (Y-Achse), die darauf vertikale (senkrechte) Koordinatenachse und
- Die Applikatenachse (Z-Achse), die auf beiden anderen Achsen senkrechte Achse [DIN-461 1973]

Ebenso besitzt die IMU Drehratensensoren, z.B. ein Gyroskop zur Messung der Rotationsgeschwindigkeit, welche die Drehraten um diese Raumachsen messen. Durch die kontinuierliche Auslesung der Sensordaten, wird die Position des Geräts berechnet und in Form von drei Koordinaten ausgegeben. Ausgehend von dem Referenzkoordinatensystem wird die Position neu bestimmt. Die Abbildung 2.8 veranschaulicht solch ein Beispiel.

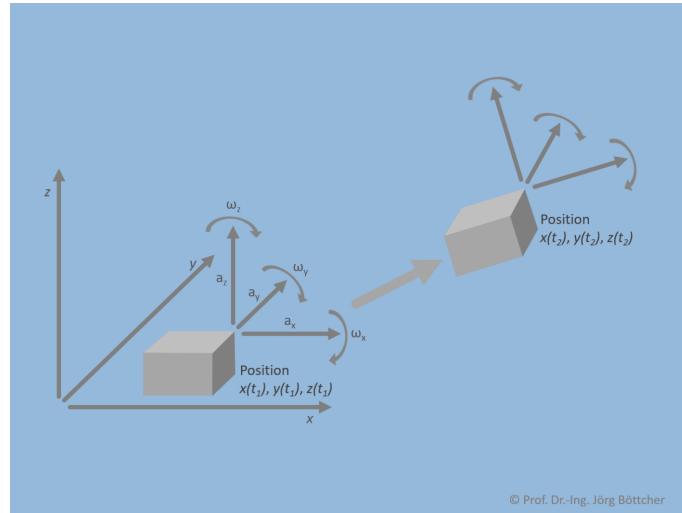


Abbildung 2.8: Grundprinzip einer IMU [BÖTTCHER 2020]

### Environmental understanding

Die zweite Schlüsselfunktion des Frameworks ist das Verständnis der Umgebung. Dieses wird erlangt, indem „*feature points*“, Ebenen und Flächen beim scannen des Umfelds erkannt werden. Beim Entstehen mehrerer solcher *feature points* erstellt ARCore ein Cluster aller naheliegenden Punkte. Dieses erzeugte Cluster repräsentiert eine horizontale oder vertikale Fläche. Mit diesen gewonnenen Informationen ist das Platzieren von virtuellen Objekten auf den erfassten Flächen möglich.



Abbildung 2.9: Skizze zum Umgebungsverständnis des Smartphones [ARCORE 2020]

### Light estimation

Mit der Funktion der Lichtschätzung kann ARCore Lichtverhältnisse und Informationen über die Beleuchtung der Umgebung erkennen. Dadurch kann das SDK durchschnittliche Intensität und eine

angepasste Farbkorrektur eines bestimmten Kamerabildes liefern, um so die Aufarbeitung des Bildes der Umgebung anzupassen. Diese Funktion verstärkt die Anpassung des virtuellen Objektes an die Realität und lässt dieses noch glaubwürdiger erscheinen.



Abbildung 2.10: Skizze zu den Lichtverhältnissen des Smartphones [ARCORE 2020]

### 2.4.2 Android Jetpack

Android Jetpack ist eine Ansammlung an Bibliotheken, die es Entwicklern ermöglicht *Best Practices* zu befolgen, Boilerplate<sup>5</sup>-Code zu reduzieren und Quell-/ Source-Code zu schreiben. Die durch Android Jetpack zur Verfügung gestellten Bibliotheken fördern die konsistente Funktionsweise der Android-Versionen und -Geräten. Darüber hinaus wird mit Android Jetpack ein Überblick verschafft, um Best Practices und empfohlene Architekturen zu berücksichtigen.

Außerdem wurde Android Jetpack für moderne Design-Praktiken entwickelt, darunter fällt „*separation of concerns*“<sup>6</sup>, *testability*, *loose coupling*, *Observer Pattern*<sup>7</sup>, *Inversion of Control*<sup>8</sup> und *productivity features*, z.B. Plugin-Integrationen. Dies sind Techniken, um eine robuste und qualitativ hochwertige App zu erstellen.

Die Bibliothekensammlung ist die Grundlage für die verwendete Architektur, Android Architecture Components in Abschnitt 2.5.3.

Die Abbildung 2.11 zeigt einen Überblick über die in Jetpack enthaltenen Bibliotheken. Eine nähere Erläuterung aller Bibliotheken findet im Rahmen dieser Ausarbeitung nicht statt, wobei die zur Verwendung geplanten „Architecture Components“ beschrieben werden.

---

<sup>5</sup>Sind Codeabschnitte, die an mehreren Stellen mit wenigen oder gar keinen Änderungen aufgeführt sein müssen. [DEVELOPER 2019]

<sup>6</sup>(SoC), Entwurfsprinzip, um verschiedene Verantwortlichkeiten auf verschiedenen Abschnitte zu trennen. [NATESAN 2019]

<sup>7</sup>Beobachter-Muster, Verhaltensmuster der GoF-Muster

<sup>8</sup>(IoC), Umsetzungsparadigma, das ein spezifisches Objekt über ein mehrfach verwendetes Objekt aufruft. [NIEMANN 2020]

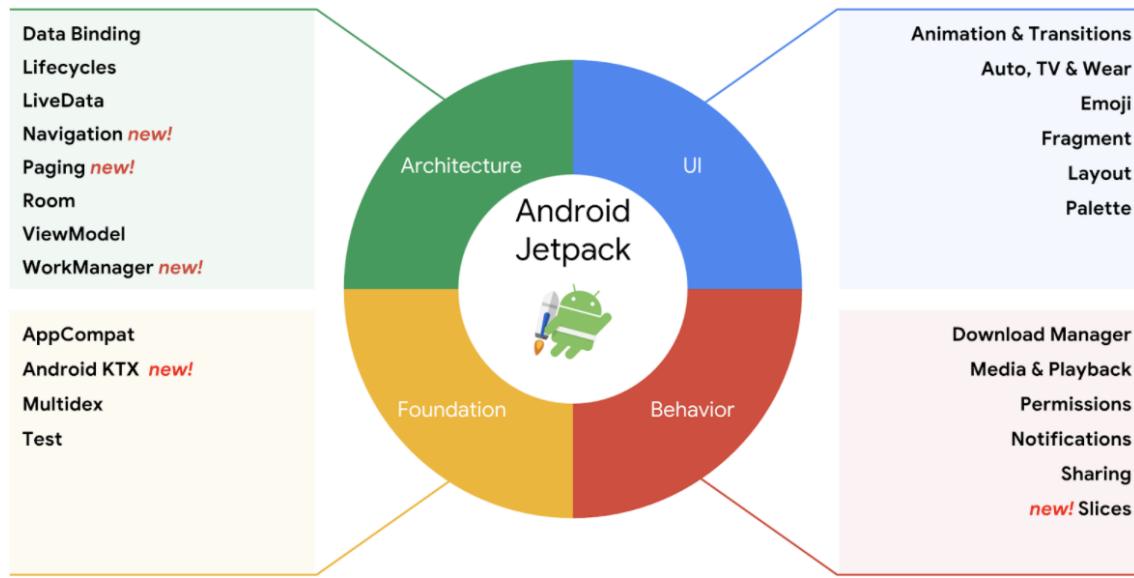


Abbildung 2.11: Komponenten von Android Jetpack [SELLS, POIESZ und NG 2018]

LiveData, Room und ViewModel sind Bestandteil der *Android Architecture Components*-Konstellation und bilden die Basis für die Architektur, welche an das *MVVM-Pattern* angelehnt ist.

## LiveData

*LiveData* gehört zu den *Observer-Pattern* und ist eine Datenhalterklasse die eine Beobachtungsfunktion, bzw. -charakter besitzt. Der Unterschied zu einem regulären Observable ist, die lebenszyklus-unabhängige LiveData-Komponente. Dies bedeutet, der Lebenszyklus anderer Komponenten z.B. *Activities*, *Fragments* oder *Services*, wird berücksichtigt. Mit dieser Kenntnis wird sichergestellt, dass LiveData nur App-Komponenten beobachtet, welche sich in einem aktiven Lebenszyklus befinden.

Vorteile die die *LiveData*-Komponente mit sich bringt sind unter anderem die Sicherstellung der Übereinstimmung der Benutzeroberfläche und deren Datenstatus, vorbeugend gegenüber Speicher-verlust, immer auf dem aktuellsten Stand der gespeicherten Informationen und der Wegfall der manuellen Lebenszykluskoordination. [DEVELOPERS 2020]

## Room

*Room* ist eine Persistenzbibliothek und bietet ein Abstraktionsschicht über der eigentlichen *SQLite*-Datenbank, um einen robusten Datenbankzugriff zu managen und das volle Potential von SQLite gleichzeitig zu nutzen. [DEVELOPERS 2017a]

Die robuste SQL-Objektzuordnungsbibliothek *Room* kann einen Cache mit den Daten der Applikation auf einem Gerät erstellen. Dieser Cache dient als Wahrheitsquelle der Applikation und

ermöglicht den Benutzern eine konsistente Kopie der Informationen der App anzuzeigen und immer zugänglich zu machen. *Room* kann in drei Kategorien, bzw. in drei Hauptkomponenten unterteilt werden:

- Room database: Dient als Zugriffspunkt für die zugrundeliegende SQLite-Datenbank
- DAO (Data Access Object): Beinhaltet Methoden um Datenbankzugriffe zu gewährleisten.
- Entity: Repräsentiert ein Objekttabelle der Datenbank.

Das Diagramm 2.12 veranschaulicht die Interaktion, bzw. Funktionsweisen der Hauptkomponenten.

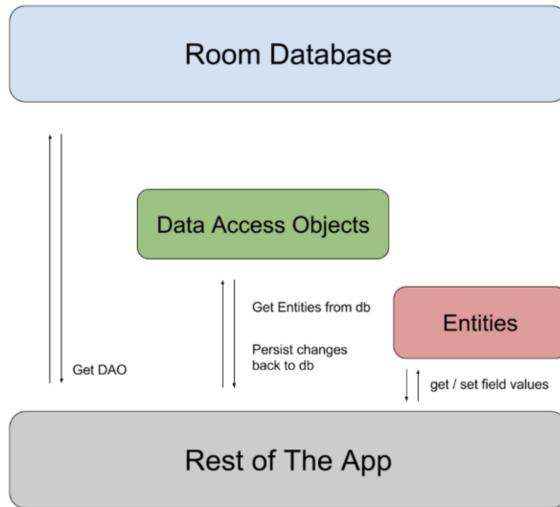


Abbildung 2.12: Room Architektur Diagramm [DEVELOPERS 2017b]

Allgemein ermöglicht *Room* einen vereinfachten Zugriff auf die Datenbank, hat einen hohen Grad an „*testability*“, zur Kompilierungszeit wird eine SQL-Abfrageüberprüfung durchgeführt und interagiert einwandfrei mit anderen Android Architecture Components, z.B. LiveData und ViewModel.

## ViewModel

Grundsätzlich dient das *ViewModel* als lebenszyklusbewusste Speicherung und Verwaltung von benutzeroberflächenbezogenen Daten. Durch die *ViewModel*-Klasse können Informationen Konfigurationsänderungen, z.B. Bildschirmsdrehungen, die als Änderung zählen, überstehen. Eine Konfiguration kann sog. Activities verwerfen und neu laden, so können nicht persistierte Daten verloren gehen.

Ein *ViewModel* fungiert ebenso als Kommunikationsschnittstelle zwischen den darüber- und darunterliegenden Komponenten der Architektur (siehe Abschnitt 2.5.3). Des Weiteren kann die

*ViewModel*-Komponente Informationen zwischen Benutzeroberflächen und verschiedenen Fragmenten, engl. „fragments“<sup>9</sup>, teilen.

Die Abbildung 2.13 zeigt eine sich ändernde Activity, die zu Beginn und nach Neuerstellung oder Aktualisierung Zugriff auf die unveränderten Daten hat.

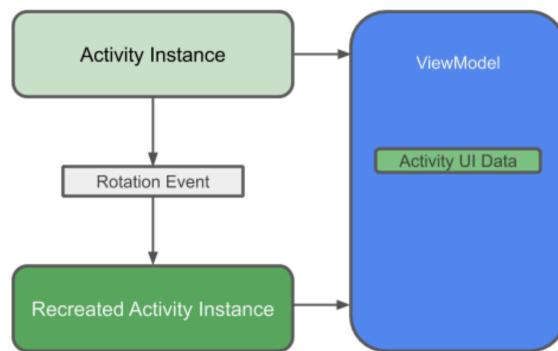


Abbildung 2.13: ViewModel Struktur Diagramm [CODELABS 2020]

---

<sup>9</sup>Teil einer User Interface (UI) in einer Android Activity

### 2.4.3 Sceneform SDK

Das Sceneform SDK ist ein Open-Source Projekt der Google LLC., welches für das Rendern von 3D-Szenen und -Animationen zuständig ist. Basierend auf der physikalischen Echtzeit-Rendering-Engine, *physically based renderer* für Android, iOS, macOS, Linux und Windows von Google Filament, wurde Sceneform entwickelt.

Bei dem Ansatz des physikalisch basierten Renders in der Computergrafik wird versucht, alle Modalitäten der realen Welt genauer zu modellieren. Mit dieser Methode werden virtuelle 3D-Objekte in der Augmented Reality noch realer dargestellt, um einen Echtheitseffekt zu erzielen und zu simulieren. Ein Beispiel ist das Simulieren des Lichtflusses, um einen Schatten des Objekts zu erzeugen, wodurch die Wirksamkeit immer mehr der Realität entspricht.

Dreidimensionale Modelle werden als *(.obj)*-Dateien in das Projekt importiert. Diese Dateien beinhalten aufgelistete Punkte, Vektoren und Linien, welche das Modell repräsentieren. Das *Sceneform SDK* verwendet die *.obj*-Datei und konvertiert, bzw. rendert diese in eine, *Sceneform binary assets (.sfb)*-Datei. Diese generierte Datei ermöglicht das Generieren eines dreidimensionalen Objekts, welches anschließend in AR-Anwendungen basierend auf Google ARCore verwendet werden kann. Für die finale Renderung und Anzeige ist die *ModelRenderable*-Klasse des SDKs zuständig.

### 2.4.4 SQLite

SQLite ist eine Open-Source In-Process-Bibliothek, die ein in sich geschlossenes, serverloses, transaktionsfreies SQL-Datenbankmodul ohne Konfiguration implementiert. SQLite ist eine eingebettete SQL-Datenbank-Engine die im Gegensatz zu anderen SQL-Datenbanken über keinen separaten Serverprozess verfügt, d.h. Transaktionen, Lese- und Schreibzugriffe werden direkt auf eine normale Festplattendatei getätigkt. [HIPP 2018]

Darüber hinaus ist SQLite plattformübergreifend und kann beliebig Datenbank- und Objekttabellen, Indizes und Ansichten erstellen und verwalten.

## 2.5 Softwarearchitektur

Komponenten in Form von Klassen, Objekten oder Bibliotheken und deren Verbindungen zwischen einzelnen Komponenten beschreibt die Architektur eines Softwaresystems. Vielmehr geht es bei der Software-Architektur darum, Anforderungen und deren Zusammenhänge untereinander von dem zu konstruierenden System zu beschreiben und nicht einen detaillierten Entwurf vorzulegen. Jedoch hat die Architektur einen enormen Einfluss auf die qualitativen und nicht-funktionalen Eigenschaften des daraus resultierenden Systems.

Die Terminologie nach dem *IEEE-Standard 1471-2000* zur Software Architekturbeschreibung, deren Aufgaben und Zweck [JECKLE u. a. 2005] sind wie folgt definiert:

Die grundlegende Organisation eines Systems, dargestellt durch dessen Komponenten, deren Beziehungen zueinander und zur Umgebung sowie den Prinzipien, die den Entwurf und die Evolution des Systems bestimmen. [HASSELBRING 2006]

Softwarearchitektur bietet viele Möglichkeiten, ein System zu entwerfen und Anforderungen und Eigenschaften umzusetzen. Daher gibt es auch hier viele Ansätze, Lösungen und Abwandlungen, um den Standards und Richtlinien gerecht zu werden. Beispiele dafür sind unter anderem die Modulare Software Architektur (2.5.1) und das Architektur-Entwurfsmuster MVVM (2.5.2).

Das Buch *Design Patterns - Elements of Reusable Object-Oriented Software* von E. Gamma, R. Helm, R. Johnson und J. Vlissides befasst sich mit den verschiedensten Möglichkeiten und Ausprägungen der Softwarearchitektur.

Im Rahmen dieser Ausarbeitung findet keine Aufzählung und Beschreibung verschiedener Arten der Architekturmuster und -stile statt, lediglich die für das Projekt verwendeten Muster werden in folgenden Kapiteln aufgegriffen.

Die Abbildung (2.14) veranschaulicht eine vereinfachte Struktur eines Architekturmusters und deren Komponenten und Zusammenhänge zueinander. Die Zeichnung dient zur Veranschaulichung, um darzulegen, wie ein Architekturdiagramm aussehen kann, bzw. wie die allgemeine Struktur repräsentiert wird.

Im Falle dieser Abbildung wurde das Strukturmuster Model View Controller (MVC) ausgewählt, welches die zu sehenden Komponenten in drei in sich geschlossene und unabhängige miteinander agierende Fragmente unterteilt.

Neben der Strukturierung von Systemen und Applikationen kann die Modellierung einer Softwarearchitektur auch dabei helfen, eine Architektur genauer zu beschreiben und zu dokumentieren. Ebenso können Diagramme auch das Management sowie die Planung beeinflussen und verstärken. Die Modellierung der Softwarearchitektur stellt somit keinen Selbstzweck dar, sondern bietet einen Mehrwert, weil es zur Verständigung, Dokumentation und Kommunikation zwischen Entwicklern und Kunden zusätzlich beiträgt.

Durch die Modellierung der Architektur kann frühzeitig eine sinnvolle Evaluierung und Bewertung des Entwurfs durchgeführt werden. Mit diesen entstehenden Bewertungen können folgende Schritte besser geplant und umgesetzt werden.

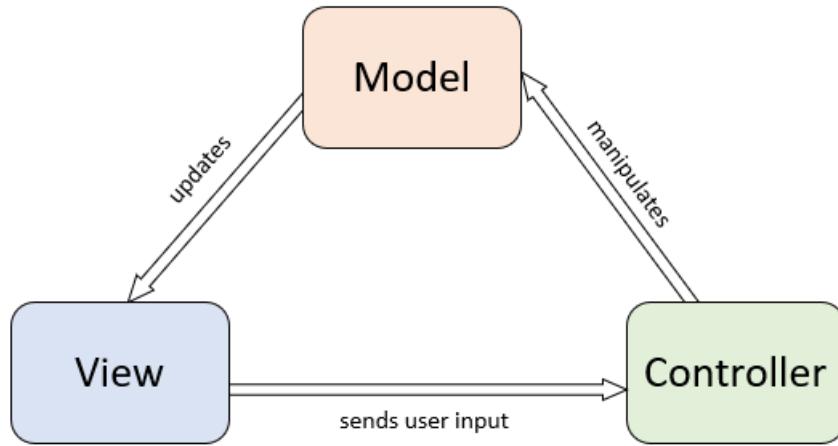


Abbildung 2.14: MVC Architektur Diagramm [ModelView Controller 2020]

Eine weitere Variante, bzw. Ausprägung der Software-Architektur ist die Modulare Software-Architektur (siehe Abschnitt 2.5.1). Viele Softwarearchitekturen lehnen sich an das Prinzip der Modularen Architektur an und nehmen diese als Grundlage.

### 2.5.1 Modulare Software-Architektur

Der Begriff der Modularität beschreibt den Hauptteil der Architektur. Modularität bezeichnet die Aufteilung eines großen Ganzen in mehrere kleinere Teile, die als Module, Komponenten oder Bausteine beschrieben werden. Verschiedene Teile können bei geeigneter Funktion und Kompatibilität zusammengeführt werden oder über entsprechend definierte Schnittstellen interagieren.

Unter der allgemeinen modularen Software-Architektur, bzw. Programmierung, versteht man ein Programmierparadigma. Dabei sollte die Software aus systematisch und logisch aufgeteilten Teilblöcken bestehen, diese werden Module oder Bausteine genannt. Der Aufbau der modularen Software kann praktisch in allen imperativen Programmiersprachen<sup>10</sup> Anwendung finden. Durch Modularität soll die Software bessere Kontrollierbarkeit, Übersichtlichkeit und Testbarkeit innerhalb großer Softwareprojekte gewährleisten. So sind die einzelnen Bausteine an sich unabhängig und für sich selbst zuständig, trotzdem können diese mit weiteren Bausteinen kombiniert und verschachtelt werden.

In der Entwicklung sind die einzelnen Module eigenständig und separat zu planen, programmieren und testen. Dadurch ist der Rahmen des einzelnen Moduls überschaubar. Nach erfolgreicher Testung der Module können die Einzelteile als eine Anwendung zusammengeführt werden, indem sie logisch

<sup>10</sup>Programmierparadigma, das aus einer Folge von Anweisungen besteht.

über Schnittstellen verknüpft werden. Erst nach diesem Schritt ist die entstehende Applikation vollständig einsatzbereit.

Die modulare Programmierung gilt als Erweiterung des prozeduralen Ansatzes, dabei sind die Module in kleineren Ansätzen auf die Klassen der objektorientierten Programmierung zurückzuführen. [SCHREYER 2018]

Um eine Software-Architektur übersichtlich und strukturiert umzusetzen, gibt es demzufolge Entwurfsmuster, engl. Patterns, die zusätzlich für Ordnung innerhalb der Architektur sorgen. Ein für die Ausarbeitung relevantes Muster ist das MVVM-Pattern. (siehe 2.5.2)

### 2.5.2 MVVM

Das Model-View-ViewModel-Pattern ist, wie bereits erwähnt, ein Architekturmuster das den Entwicklern als Vorlage dient, um ordnungsgemäße, standardisierte und strukturierte grafische Oberflächen und das dahinterstehende logische System zu entwickeln. Es wird besonders für interaktive Systeme eingesetzt. Das MVVM-Muster wird allgemein als Weiterentwicklung des bekannten MVC-Patterns (der Abbildung 2.14 zu entnehmen) angesehen.

Grundlegend ist das Model-View-Controller-Muster, MVC, für die klare Abgrenzung zwischen dem Datenmodell, der Darstellung von Daten und der Steuerung von Nutzerinteraktionen zuständig. Der Controller koordiniert die Aktionen der View und dem Model. Dieser fungiert als Verbindungsstück zwischen den Bestandteilen und informiert die View im Bedarfsfall bei Änderungen am Model.

Demnach ist auch das Model-View-ViewModel-Muster daran angelehnt, die Benutzeroberfläche von deren Logik zu trennen. Durch diese Trennung ist eine erhöhte Wart- und Testbarkeit gewährleistet. Die ermöglichte parallele Arbeitsweise von UI-Designern und Entwicklern unterstützt deren Arbeitsfluss und beschleunigt den Entwicklungsprozess. [MVVM-Pattern 2010]

Die Logik der Anwendung wird in den ViewModel-Klassen (2.4.2) dargestellt und dienen als Kommunikationsschnittstelle zwischen dem Model, dem Datenhalter und der View, der Datenrepräsentation. Durch die Unabhängigkeit der einzelnen Module kann ein Unit Test vereinfacht durchgeführt werden, mitunter ein großer Vorteil des Patterns.

Die Ebenen, die das MVVM-Pattern enthält, werden in Abbildung (2.15) grafisch aufgeführt.

Die View-Ebene stellt die Informationen der Benutzeroberfläche dar, fängt die Benutzereingaben ab und gibt diese über Datenbindungen dann an das darunterliegende ViewModel weiter. Die View-Klasse enthält lediglich die Komponenten, die die Oberfläche optisch ausmachen und die dazugehörigen Datenbindungsschnittstellen bereitstellt, um die Informationen übergeben zu können. Ebenso werden über diese Schnittstellen Informationen geladen, um diese auf der Oberfläche repräsentieren zu können. Diese Vorgehensweise erleichtert das Austauschen der View, ohne den Code generell zu ändern.

Die ViewModel, bereits in Kapitel (2.4.2) beschrieben, ist dafür zuständig, die Daten zwischen der persistierten Speicherung und der Repräsentation zu transferieren. Es stellt das Model für die View dar und gibt das eigentliche Model nach außen.

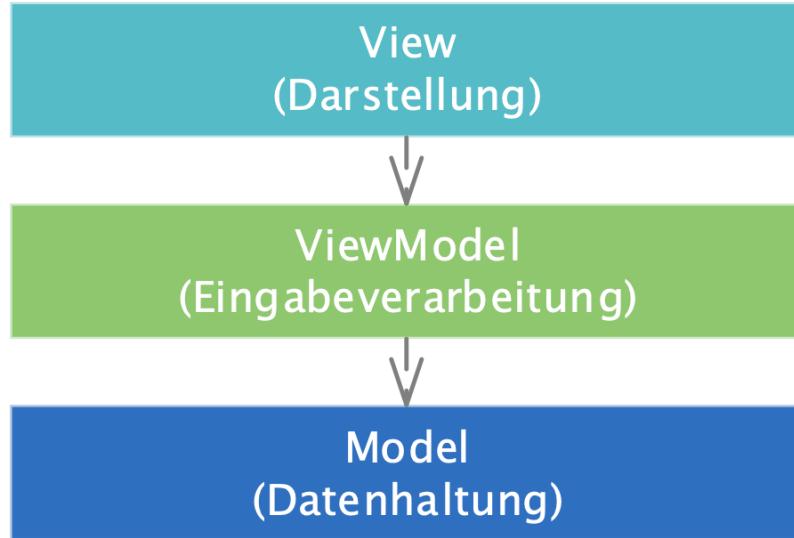


Abbildung 2.15: MVVM Architektur Diagramm [JAECKLE, GOLL und DAUSMANN 2015]

Die unterste Ebene, das Model, bzw. die Datenhaltung, stellt die Abbildung der Daten dar und gilt nicht als reine Abbildung der Datenquelle. Diese Daten werden für die visuelle Anwendung auf der View-Ebene benötigt und durch die ViewModel-Ebene bereitgestellt. Darüber hinaus können die Daten von dem Nutzer über die View manipuliert und an das Model zurückgegeben werden. Durch diese Möglichkeiten ist vorausgesetzt, dass das Model folgende Funktionalitäten bereitstellen sollte [EDER 2016]:

- Validierung der Daten
- Benachrichtigung bei Eigenschafts-Änderungen
- Verarbeitung von Business-Rules

Sozusagen dient das Model als Schnittstelle zwischen der anhängenden Datenbank und dem View-Model, welches die Daten hält. Dazu kommt, dass das Model festlegt, wie die Objekte aufgebaut sind, d.h. dort wird festgelegt, welche Informationen das Objekt beinhaltet.

Die nachfolgende Abbildung (2.16) veranschaulicht den Informationsfluss des MVVM-Patterns in einzelnen Schritten.

Basierend auf der Grundlage des Model-View-ViewModel-Patterns wurde die für das Projekt anvisierte Architektur, Android Architecture Components (2.5.3), entwickelt.

### 2.5.3 Android Architecture Components

Die Android Architecture Components bestehen aus einer Sammlung von Tools und Bibliotheken, engl. Libraries, die entwickelt wurden, um die Konstruktion von testbaren und stabilen Android-Apps

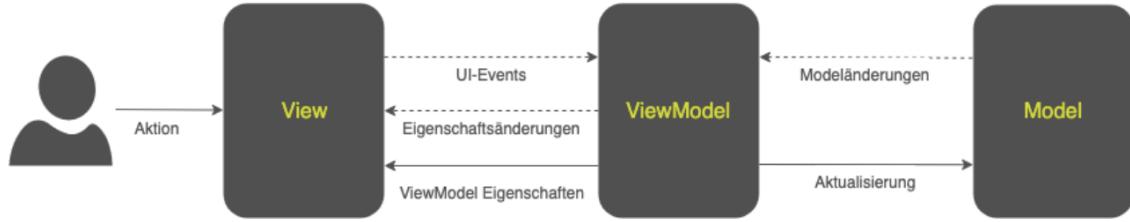


Abbildung 2.16: MVVM Informationsfluss [RICARDO 2019]

zu vereinfachen. Mit der Einführung von Android Jetpack (2.4.2) wurden viele Komponenten, z.B. Bibliotheken und Frameworks, für die Entwicklung von Apps vereint. In den vielen Bibliotheken, die von Android Jetpack zur Verfügung gestellt werden, befinden sich auch die sogenannten *Architecture components*. Diese *Components* unterstützen und fördern die Umsetzung der *MVVM Pattern* basierten Architektur (2.5.2). Die zuvor genannten Bibliotheken ViewModel, LiveData und Room sind maßgebliche Bestandteile für die Gestaltung des angestrebten Musters Model-View-ViewModel. Google selbst empfiehlt im Rahmen der veröffentlichten Dokumentation „*Guide to app architecture*“ die Verwendung der *Architecture components*, um eine stabile und leistungsfähige Applikation zu designen. Diese basiert auf bewährten Prinzipien, unter anderem dem *separation of concerns* Prinzip.

Folgender Abschnitt befasst sich im Einzelnen mit den Komponenten, die für die *Android architecture* essentiell sind:



Abbildung 2.17: Android Grundkomponente [Architecture Components 2020]

Grundsätzlich sind Activities und Fragments nicht Teil der eigentlichen „*Architecture Components*“, sondern sind lediglich Grundkomponenten der Android App Entwicklung die die Benutzeroberfläche und die dazu korrespondierenden Layout-Dateien verwaltet.

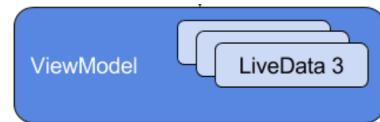


Abbildung 2.18: ViewModel Komponente [Architecture Components 2020]

Das ViewModel (siehe Abschnitt 2.4.2) verwaltet die relevanten Daten der UI. Diese Komponente hat dem Activity-Objekt gegenüber einen unabhängigen Lebenszyklus. Somit sind Daten bei unvorhergesehenen Unterbrechungen oder Fehlern der UI nicht betroffen und können bei erneutem Aufbau der Oberfläche ohne Probleme nachgeladen werden. Die im ViewModel enthaltenen LiveData

Objekte (2.4.2) sind beobachtbare Daten-Container, die eine Activity ohne expliziten Aufruf über Datenänderungen informieren. Durch diese Methode wird der Lebenszyklus der aktiven Komponente respektiert und berücksichtigt.



Abbildung 2.19: Repository Komponente [Architecture Components 2020]

Mit dem Repository wird ein „best practice“ Fall umgesetzt, der eigentlich kein Bestandteil der Android Jetpack Library ist. Diese Komponente ist dafür zuständig das ViewModel von der eigentlichen Datenquelle zu trennen, um Informationen aus verschiedenen Quellen beziehen und synchronisieren zu können. Das Repository dient als Schnittstelle zu mehreren Datenbeziehungspunkten und verwaltet ein Verzeichnis zur Speicherung und Beschreibung von Objekten und Informationen. Zudem kapselt das Repository die Logik für das Persistieren und Erzeugen von Entitäten und Aggregaten von der Ansicht.

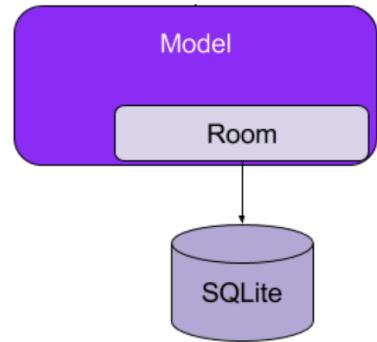


Abbildung 2.20: Model / Room Komponente [Architecture Components 2020]

Die *Persistence Library Room* (2.4.2) ist ein Datenbank *Layer* auf der eigentlichen SQLite Datenbank. Zugriffe auf die Datenbank werden durch Room gekapselt und vereinfacht, indem viel mit Annotationen, z.B. Entities (*Entity*) und Data Objects (*Dao*), gearbeitet wird. Der Abbildung (2.20) ist demnach zu entnehmen, dass die Bibliothek „Room“ einen Großteil der Funktionen eines Models abdeckt und so die Model-Komponente repräsentiert. Unter diesem Datenbank Layer befindet sich die eigentliche Persistenzschicht, die SQLite Datenbank.

In einem fortlaufenden Architektur-Diagramm sind die Komponenten wie folgt angeordnet, um die Abhängigkeiten zwischen den einzelnen Teilen zu verdeutlichen.

In der Abbildung (2.21) wird deutlich, welches Modul welche Abhängigkeiten besitzt, bzw. von welchen Modulen Informationen bezogen werden können. Einzelne Komponenten können ihren Vorder oder Nachgänger niemals umgehen und sind so einer festen Hierarchie eingeordnet. Die Abbildung zeigt auch eine beispielhafte Ausführung mehrerer Datenbezugsquellen.

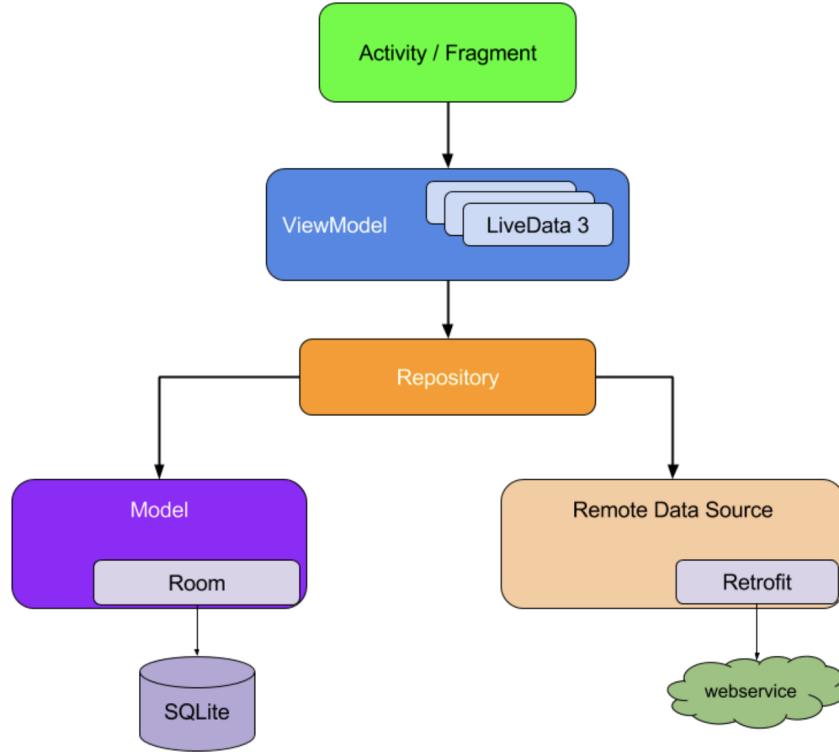


Abbildung 2.21: Android Architecture Components Diagram [*Architecture Components 2020*]

Bei genauer Betrachtung des Diagramms sind alle Komponenten wiederzufinden, die ebenso in dem MVVM-Muster gegeben sind und das Muster prägen. Durch die vielen Komponenten und Schnittstellen, die dazu beitragen, dass weitere Module angehängt oder ausgetauscht werden können, ist Modularität gewährleistet.

## 2.6 Datenmodellierung

Datenmodellierung ist ein Verfahren zur formalen Abbildung von Objekten, deren Attribute und Beziehungen in einem definierten Kontext stehen. Die Modellierung verfolgt das Ziel, den Überblick über die Daten oder die Applikation zu erhalten. Hauptaufgabe ist die eindeutige Definition von Objekten und deren Spezifikationen und Ausprägungen. Ebenso hilft die Datenmodellierung bei der visuellen Darstellung von Informationen und bei der Einhaltung von Voraussetzungen und Richtlinien.

Die aus der Modellierung entstehenden Datenmodelle stellen die Objekte und Beziehungen untereinander dar und repräsentieren die zu persistierende Struktur der Objekte. Datenmodelle sind ein wichtiger Bestandteil einer Software und haben meist eine längere Lebensdauer als die Software an sich, da Änderungen an den Modellen meist viele Risiken, dazu zählt überwiegend der Datenverlust, mit sich bringen.

Durch die Übersichtlichkeit der Datenmodelle können Standardwerte, Semantik, Sicherheit und die Datenqualität gleichzeitig sichergestellt werden. Die Konsistenz von Namenskonventionen wird durch die Überschaubarkeit auch leichter überprüfbar.

In der Softwareentwicklung wird die Datenmodellierung als wesentliche Teilaufgabe gesehen, um ein System aufzubauen. Innerhalb des Datendesigns gibt es drei wichtige Varianten, die aufeinander aufbauen:

- Konzeptionelles Datenbankschema: Betrachtung eines für die Applikation wichtigen Szenarios der realen Welt unter Berücksichtigung aller relevanten Objekte, Eigenschaften und Beziehungen zwischen ihnen. Diese Informationen werden aus dem Kontext heraus analysiert und grafisch als auch textuell dargestellt. Die Darstellung basiert auf einem Entity-Relationship-Modell (ERM), das die Objekte (Entities) mit ihren jeweiligen Eigenschaften (Attributes) und die Beziehungen (Relationships) darstellt. Das ist das sogenannte Semantische Datenmodell.
- Logisches Datenbankschema: Hierbei wird das konzeptionelle Datenbankschema um datentechnische Angaben, z.B. Feldtypen /-formate und Identifikationseigenschaften (Primary Keys), erweitert. Das logische Datenbankschemata unterliegt den Regeln der von dem Datenbank-Management-System (DBMS) gegebenen Struktur, z.B. dem relationalen Datenmodell, das alle Daten und Objekte in Tabellen ablegt. Dies wird auch relationales oder objektorientiertes Datenmodell genannt. [ROEING 2019]
- Physisches Datenbankschema: Dabei werden die Datensätze auf den physischen Komponenten organisiert und verwaltet. Eine Festlegung der Zugriffsoperationen finden in dieser Phase statt. Zugriffsoperationen sind Einfügen, Löschen, Aktualisieren und Suchen von Objekten und Informationen. Zudem wird hierbei versucht, die Performance und effiziente Bearbeitung von Datenbankzugriffen durch Modellierung hochzuhalten.

Die drei oben aufgeführten Phasen sind Bestandteile der Methodik des Datenbankdesigns, der sogenannten „*top down-Methodik*“. Wobei die erste Phase die allgemeine Anforderungsanalyse, die in obiger Auflistung nicht aufgeführt ist, beinhaltet. Bei dieser Phase finden grundlegende Anforderungsprozesse statt, zum einen die Informationsstrukturanforderungen und zum anderen die Datenverarbeitungsanforderungen.

Ein Schema, nachdem relationale Datenbanken aufgeteilt werden, ist die sogenannte Normalisierung von Datenbanken. Darunter versteht man die Aufteilung von Attributen in mehrere Relationen (Tabellen) mithilfe von Normalisierungsregeln und Normalformen. [BEGEROW 2020] Ziel dieser Zerlegung ist die Erstellung einer redundanzfreien Speicherung der Daten, um Daten löschen zu können, ohne einen Informationsverlust zu erleiden. Die Normalisierung wird in mehrere Stufen unterteilt:

- Nullte Normalform (0NF)
- Erste Normalform (1NF)
- Zweite Normalform (2NF)
- Dritte Normalform (3NF)
- Boyce Codd Normalform (BCNF)
- Vierte & Fünfte Normalform (4NF & 5NF)

Die meist verwendete Normalform ist die dritte Normalform, da diese für die Balance zwischen Redundanz, Performance und Flexibilität entscheidend ist und oft ausreicht.

Im Folgenden wird nicht weiter auf die Normalformen eingegangen, sie dienen lediglich der vollständigen Nennung.

# Kapitel 3

## Konzeption

In diesem Kapitel wird das erarbeitete Konzept dieser Ausarbeitung dargelegt. Dazu gehören die Überlegungen zu einzelnen Arbeitsschritten und dem grundsätzlich angedachten Aufbau des Projekts, sowie Entscheidungs- und Beweggründe, weshalb bestimmte Technologien gewählt wurden. Zu Beginn wird auf die Einsatzmöglichkeiten (3.1), in der die Applikation Anwendung finden könnte, eingegangen. Anschließend werden die Grundgedanken zu den einzelnen Phasen, Scan-Phase (3.2) und Visualisierungs-Phase (3.3) des Systems erläutert, ebenso was sie beinhalten und wie sie funktionstechnisch angedacht sind. Mit den zugrundeliegenden Informationen wird auf das Architekturkonzept (3.4), sowie auf das Softwarekonzept (3.4) eingegangen. Des Weiteren werden die Hintergründe der Wahl des AR-Frameworks (3.6) aufgezeigt. Abschließend wird noch das konzipierte und prototypische Datenmodell (3.7) dargelegt.

### 3.1 Anwendungsumfeld

Bei der Konzeption wurde darauf Rücksicht genommen, die Anwendung möglichst so zu planen, dass der Einsatzort zu Anfang nicht zwingend maßgeblich ist. Dies bedeutet, dass je nach Anwendungsfall die Applikation angepasst werden kann. Allerdings sind für das Unterstützungssystem ausschließlich Produktionshallen, eventuell auch Logistik- oder Lagerhallen, vorgesehen. Durch die individuelle Möglichkeit des Umgebungscans ist die Applikation anfangs nicht an festgelegte Orte gebunden. Damit wird zum Ausdruck gebracht, dass der Umgebungscan flexibel durchgeführt werden kann, da keine Anforderungen an die Umgebung gestellt sind. Die darauf aufbauende Visualisierungs-Phase ist demnach abhängig von dem Ort, indem der Scan durchgeführt wurde und kann nicht anderweitig, in anderen Hallen, eingesetzt werden.

Ein mögliches Anwendungsumfeld dieser Applikation wären die Räumlichkeiten des nahe bei cjt Systemsoftware AG angesiedelten Unternehmens Siemens AG. Neben der Vielzahl an gemeinsamen Projekten, wäre dies eine weitere Kooperation mit der Siemens AG und würde eine unterstützende Maßnahme einleiten.

## 3.2 Scan-Phase

Die Definition der Scan-Phase ist entscheidend für die Konzeption und Umsetzung der Architektur- und Softwarekonzeption. Anhand der dabei entstehenden Anforderungen und Zielsestellungen konnte die Konzeption ausgelegt und erarbeitet werden. Dieser Abschnitt widmet sich der expliziten theoretischen Erläuterung dieser Phase und somit auch einer der beiden Kernfunktionen dieses Softwaresystems.

Der Nutzer hält ein Smartphone oder Tablet in der Hand und steht am Eingang der Räumlichkeit. An diesem Punkt wird der Scan-Vorgang gestartet. Mittels Smartphone-Kamera hat der Nutzer die Möglichkeit, das Umfeld zu scannen. Dabei wird am Anfang der Startpunkt als Nullpunkt, bzw. als Ursprungskoordinaten des erstellten Koordinatensystems der Umgebung, festgelegt. Der Nutzer kann sich frei im Raum bewegen und ist lediglich durch das Halten des Smartphones eingeschränkt. Während der Aufnahme wird anhand der im Smartphone integrierten Kamera und Sensoren unter Verwendung der Sensordaten ein Modell erstellt, welches die Umgebung virtuell repräsentiert. Dabei wird das Modell auf einem dreidimensionalen Koordinatensystem wiedergespiegelt, um Position, Blickrichtung und Höhe des zu erstellenden Objektes festzuhalten. So kommt das Simultaneous Localization And Mapping Verfahren (siehe Abschnitt 2.2) zum Einsatz, das das Smartphone genauestens räumlich lokalisiert und gleichzeitig mapped. Dies bedeutet, dass anhand der generierten Sensordaten eine Analyse der Informationen stattfindet und daraus eine virtuelle Karte des Raums erstellt wird. Dadurch kann das Gerät die Realität wiedergeben. Anhand der Informationen, die durch Gegenstände, Wände und dem Boden grundlegend gegeben sind, resultieren die Anhaltspunkte der virtuellen Umgebung.

Dem Nutzer soll die Möglichkeit geboten sein, Objekte an denen vom Anwender vorgesehenen Positionen platzieren zu können. Die Objekte werden dann, anhand der durch das SLAM Verfahren zugrundeliegenden Informationen, an der virtuellen Position erstellt und positioniert. Somit stehen Daten über die Position des Objekts zur Verfügung, welche anschließend in der Datenbank persistiert werden. Währenddessen das Objekt erstellt wird, kann der Anwender zusätzliche Informationen zu dem Objekt, welches ebenso in der Realität vorhanden ist, einpflegen. Die einzutragenden Informationen belaufen sich zu Anfang der prototypischen Konzeption auf den Namen des Geräts, die Position sowie die Blickrichtung als auch den aktuellen Zustandsstatus des realen Objekts. Eine eindeutige Identifikationsnummer wird zudem automatisch generiert, um eine einheitliche Richtlinie vorzugeben. Damit muss, bzw. soll der Nutzer die Entscheidung nicht selbst treffen. Die genauere Konzeption dieses Falls wird in dem Abschnitt Datenmodell (3.7) aufgegriffen.

Ziel bei der oben aufgeführten Scan-Funktion ist das Erstellen und Platzieren von virtuellen Objekten in der durch Scans erstellte und realitätsnahe Inszenierung des Umfeldes. In überschaubarem Rahmen wurden Anforderungen für die Scan-Phase erarbeitet, unter anderem die Einfachheit der Objekterstellung, die simpel und überschaubar zu erstellende Benutzeroberfläche, die offensichtliche Objektdarstellung in kurzer Zeit und einen nicht allzu hohen Zeitaufwand.

Nachdem die Scan-Phase, eine der beiden Kernfunktionen, konzeptionell präzisiert wurde, kommt nun die Erläuterung der zweiten Kernfunktion, die sogenannte Visualisierungs-Phase.

### 3.3 Visualisierungs-Phase

Dieser Abschnitt definiert die Aufgabe der Visualisierungs-Phase. Dabei werden auch hier Anforderungen und Zielsetzungen dieser Funktion theoretisch ausgeführt, um die anfänglichen Ideen, womit die allgemeine Konzeption der Software erarbeitet wurde, aufzuzeigen. Dafür ist vorauszusetzen, dass die Scan-Phase alle erforderlichen Daten und Informationen liefert.

Diese Funktion ist hauptsächlich zur reinen Darstellung der vorab in der Scan-Phase erstellten Objekte gedacht. Mit diesem Teil des Systems kann der Nutzer sich frei im Raum bewegen und bekommt die Objekte in seinem Umfeld auf dem Bildschirm des Smartphones angezeigt. Befindet sich ein Objekt in der Nähe, hat der Nutzer die Möglichkeit, durch Anklicken des Objekts genauere Informationen über dieses zu erhalten.

Startet der Anwender die Methode, werden alle Objekte aus der Datenbank abgefragt und mit dem dazugehörigen Zustandsstatus in den Raum projiziert, sodass Auffälligkeiten sofort zu sehen sind. Dies bedeutet, bei normalem, aktivem Status ist die Farbe der Projektion grün, bei fehlerhaftem, passivem Status rot. Dadurch wird schnell klar, ob es an bestimmten Objekten in der Realität Probleme gibt. Mithilfe dieser Funktion wird im Handumdrehen ein Überblick über die Situation und Lage der Objekte verschafft. Falls es zu einem ungewollten Verhalten kommt, kann der Anwender unverzüglich eingreifen und dementsprechend handeln. Wird ein Objekt durch die Kamera fokussiert, kann der Anwender das Objekt anklicken, um weitere Informationen über dieses zu erhalten. Die Visualisierungs-Phase sieht es allerdings nicht vor, während die Applikation läuft, neue Objekte hinzuzufügen. Dafür müsste der Anwender wiederum zur Funktion der Scan-Phase wechseln. Eine Änderung an den Objekten selbst kann nur bedingt vorgenommen werden, dies betrifft ausschließlich Informationsänderungen, die das Objekt selbst betreffen.

### 3.4 Architekturkonzept

Um das Augmented Reality basierte Assistenzsystem grundlegend zu definieren, ist es von Vorteil ein geeignetes Konzept sowie einen ersten Entwurf zu erstellen. Dieser sollte als Fundament für eine Applikation mit stetig steigender Anzahl an Funktionen dienen und für künftige Weiterentwicklungen verwendet werden. Anfänglich soll der Prototyp dafür sorgen, dem Nutzer einen Überblick über beispielsweise Maschinen in Produktionshallen zu verschaffen. Dies bedeutet, dass Informationen zu bestimmten Objekten schnell zur Verfügung stehen und eventuell defekte Maschinen zügig aufzufinden sind. Grundsätzlich muss die Anwendung in der Lage sein, auf Benutzerbedienungen zu reagieren, die Umgebung zu scannen, 3D-Objekte an der vom Nutzer vorgesehenen Stelle zu platzieren, Informationen zu den jeweiligen Objekten hinzuzufügen und ändern zu können. Wird die Applikation gestartet, die Umgebung gescannt und Objekte auf die vom Nutzer vorgesehenen Positionen platziert, sollen die erstellten Objekte gespeichert und bei erneutem Start der Anwendung aufgerufen und platziert werden. Neben der Augmented Reality Funktion soll die entstehende Software für eine fortlaufende Entwicklung geeignet sein. Für diese Eigenschaft ist ein modularer Architekturansatz vorgesehen, um das schnelle und einfache Wechseln von Komponenten und Bestandteilen zu ermöglichen. Darunter sind beispielsweise Änderungen an der Benutzeroberfläche

oder die Verwendung einer anderen Datenbank, bzw. hinzukommende Funktionen, die unabhängig von bestehenden Klassen hinzugefügt werden können. Durch das MVVM-Pattern, welches durch die *Android Architecture Components* gefördert wird, ist das Testen von *GUI*- und *BackEnd*-Elementen unabhängig voneinander begünstigt.

Wie bereits erwähnt, sind die *Android Architectur Components* an das Entwurfsmuster *MVVM* angelehnt und sorgen im Allgemeinen dafür, die Software im gesamten überschaubar und beherrschbar zu gestalten. Dabei werden die folgenden Prinzipien beachtet und auch gewährleistet. Durch gezielte Abstraktion von Informationen wird die Komplexität einer Anwendung steuerbar. Mit der großen Anzahl an verschiedenen Segmenten, die der Abbildung (3.1) zu entnehmen sind, wird die Trennung der Zuständigkeit (engl. *seperation of concerns*) deutlich. Dabei hat jede einzelne Komponente eine Zuständigkeit, bzw. Aufgabe, die sie zu bewerkstelligen hat. Sei es die *View* mit der der Nutzer interagiert, dem *ViewModel* als Kommunikationsschnittstelle, dem *Repository* als Verzeichnis zur Speicherung und Beschreibung von Objekten oder als Schnittstelle zu mehreren Datenbeziehungspunkten oder das *Model*, das die Zugriffe auf die Datenbank kapselt und die Struktur der Objekte vorgibt. Ein weiterer Aspekt ist das daraus resultierende und idealerweise in sich geschlossene System, das durch lose Kopplung und hohe Kohäsion in eine Menge an Komponenten zerlegt ist, die dadurch gewonnene Modularität.

Die Software wurde zunächst, den Phasen entsprechend (siehe Abschnitt 3.2 & 3.3), in zwei große Rubriken, den sogenannten Scan- und Visualisierungs-Phasen, unterteilt. Diese Phasen finden sich in der Abbildung (3.1) der konzeptionellen Architektur des Unterstützungssystems wieder.

*Views*, bzw. *Activities* werden mithilfe der von Android zur Verfügung gestellten *AppCompat-Library* erstellt. Die von Android Jetpack vorhandene Bibliothek *ViewModel* dient als Kommunikations-schnittstelle zwischen der *GUI* und dem *Repository*. Die ebenso von Android Jetpack publizierte *LiveData*-Bibliothek gilt als Observer zur *GUI* und benachrichtigt diese bei Änderungen. Mit der *Repository*-Komponente zwischen *ViewModel* und Datenbank, bzw. *Model* wird eine reine und neu generierte API-Schnittstelle erstellt, welche der Benutzeroberfläche Aufrufe für Datenbankzugriffe zur Verfügung stellt. Die *Room*-Bibliothek stellt ein Zugriffs-Layer da, welches die Datenbankzugriffe kapselt und steht gleichzeitig als *Model* bereit. Dort werden Entities, in der für die Datenbank vorgesehene Struktur, festgelegt. Data Objects (DAOs) regeln die Datenbank-Zugriffe über definierte SQL-Queries, die vorab, den Funktionen entsprechend, modelliert werden. Um die entstehenden Daten der dreidimensionalen AR-Objekte zu speichern wird eine *SQLite*-Datenbank verwendet.

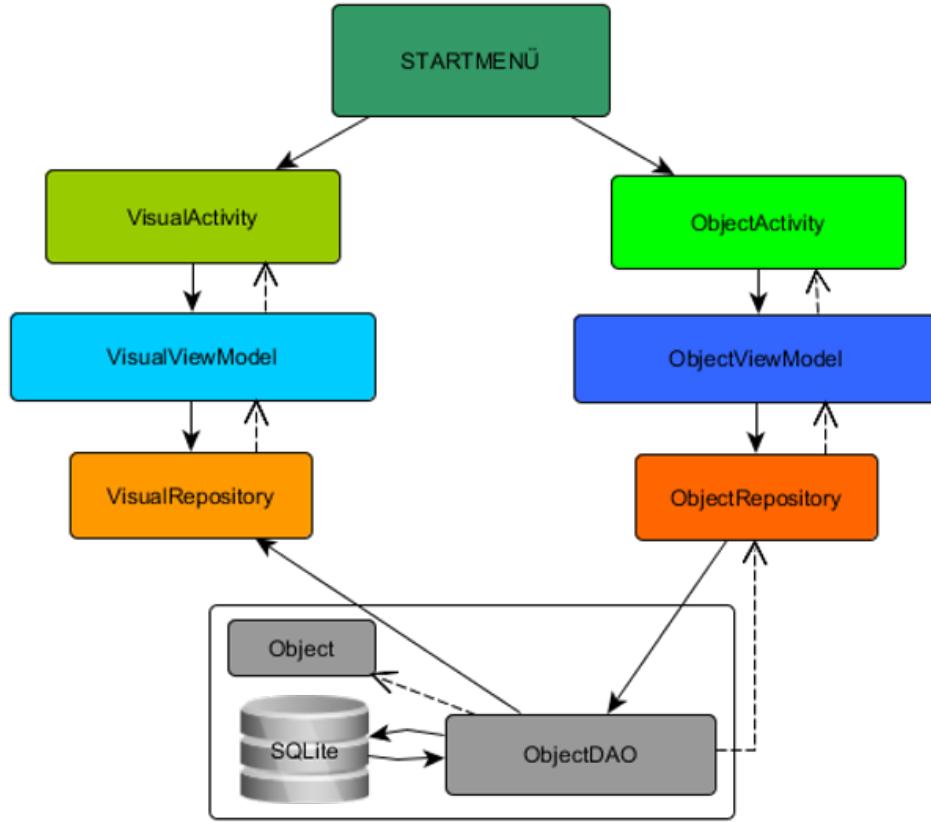


Abbildung 3.1: Konzeptionelle Software-Architektur

### 3.5 Softwarekonzept

Nachdem die Architektur konzipiert und ein grobes Bild des entstehenden Systems geschaffen wurde, konnte die Konzeption des Softwaresystems beginnen. In der Konzeption werden die Kernfunktionen, sowie die Anforderungen und Ziele veranschaulicht.

Beim Starten der Applikation öffnet sich das Startmenü der Anwendung. Hier sind auf einen Blick zentral alle Funktionen, die das System unterstützt, übersichtlich angezeigt. Dort wird die Möglichkeit offeriert, die Scan-Phase (3.2) zu starten oder fortzuführen oder die Visualisierungs-Phase (3.3) nach erfolgtem Scannen zu starten. Dabei wird auch das Ziel verfolgt, den Nutzer nicht mit Informationen zu erschlagen, bzw. den Nutzer nicht unnötig zu verwirren und ohne große Umschweife zu der eigentlichen Tätigkeit zu führen. Auch dient das Startmenü lediglich zur Navigation und leitet den Benutzer auf die von ihm gewählte Tätigkeit weiter.

### Scan-Phase

Die oben genauestens beschriebene Scan-Phase (3.2) wird nochmals aufgegriffen. Mithilfe der Scan-Phase wird die Konzeption dieser Kernfunktion auf Basis des Layouts und der darin geplanten Komponenten dargelegt.

Grundsätzlich soll die Scan-Funktion eine Oberfläche enthalten, die die Wiedergabe des Kamerabildes gewährleistet. Dies soll mit einem speziellen *ARCore-Fragment* umgesetzt werden, welches die Benutzeroberfläche überlagert und einen Großteil des Bildschirms einnimmt. Die Oberfläche soll zu folgenden Interaktionen zwischen Nutzer und Applikation, bzw. den darin enthaltenen dreidimensionalen Objekten, dienen. Die vorgesehenen Interaktionen belaufen sich auf das Scannen der Umgebung durch die vom Smartphone gegebene Kamera, das Setzen von virtuellen Objekten auf die vom Nutzer vorgesehene Position und das Hinzufügen von Informationen zu den jeweiligen Objekten. Unterhalb des AR-Fragments soll eine *Gallery* angezeigt werden, welche eine Auswahl der zu setzenden AR-Objekte bietet und mit einem *Button* pro Objekt versehen ist. So kann beim Klick auf ein Objekt-Button das dementsprechende *3D asset* gerendert und auf dem AR-Fragment angezeigt werden. Die Gallery kann im Laufe der Entwicklung des Systems zu jeder Zeit um weitere *3D assets* expandieren. Während des Renderings eines Objekts wird der Nutzer auf die Seite der Dateneingabe weitergeleitet, um speziell zu dem erzeugten Objekt Informationen eingeben zu können. Diese Informationen werden bei erneutem Wechsel zur Hauptseite samt der Daten der Position des Objekts in die SQLite Datenbank gespeichert, um den Voraussetzungen des Datenmodells (siehe Abschnitt 3.7) zu entsprechen.

Ziel dieser Funktion ist, dem Nutzer eine simple Möglichkeit zu bieten, das Umfeld zu scannen und Objekte an den vom Nutzer vorgesehenen Positionen zu platzieren. Dabei gilt es, die Anwendung so einfach wie möglich zu halten und das System auf die Kernaufgabe zu fokussieren.

Abbildung (3.2) verdeutlicht den konzipierten Inhalt der Scan-Phasen-Kernfunktion in einem Anwendungsfall-Diagramm, sodass die oben aufgeführte Beschreibung zur optischen Darstellung mit den dahinterliegenden Klassen die Vorstellung und das Konzept präzisiert.

Auf die zweite Kernfunktion wird in folgendem Abschnitt eingegangen, sowohl der Aufbau der Kernfunktion, als auch die damit einhergehenden Anforderungen und Zielsetzungen.

### Visualisierungs-Phase

Die zuvor erläuterte Visualisierungs-Phase wird in diesem Abschnitt seitens der Softwarekonzeption auf die Ziele, Anforderungen und Funktionen hin beleuchtet.

Ähnlich zur Scan-Phase gibt es bei dieser Funktion ein AR-Fragment, welches das Kamerabild wiedergibt. Dieses Fragment soll sich über den gesamten Bildschirm erstrecken und ist zusätzlich mit zwei weiteren Buttons zu versehen. Diese werden im weiteren Verlauf präzisiert. Das Ziel dieser Kernfunktion ist die Darstellung aller vorhandenen, in der Datenbank gespeicherten, Gegenstände. Hierbei werden lediglich die Erzeugnisse, die in der Scan-Phase erstellt wurden, abgerufen und als neue Objekte gerendert. Dafür wird die Position in der SQLite Datenbank abgefragt und als Basis für das neue Objekt geladen. So können die vorhandenen Marker visuell dargestellt werden. Die Funktion an sich sieht keine Änderungen der bestehenden Objekte vor, dafür muss der Nutzer

erneut in die Scan-Phase wechseln.

Bevor der Benutzer von dem Startmenü zur Visualisierungs-Phase übergehen kann, soll ein Hinweis erscheinen, welcher sicherstellen soll, dass die Scan-Phase schon bereit mindestens einmal durchgeführt wurde. Damit wird versichert, dass bereits Objekte existieren und angezeigt werden können. Denn ohne vorherigen Scan ist die Kernfunktion der Visualisierungs-Phase sinnlos.

Neben dem AR-Fragment gibt es zwei Buttons, einen davon zum Navigieren zur Benutzeroberfläche der Scan-Phase, um bei Bedarf weitere Objekte hinzuzufügen, und den zweiten zum Starten, bzw. erneuten Laden der Funktion. Damit wird die Abfrage der Daten, die in der Datenbank enthalten sind, gestartet und der Prozess des Objekterstellens, bzw. - renderns begonnen. Dadurch erscheinen alle bereits gespeicherten Objekte auf der Karte. Auf die Funktionen, Methoden und deren Umsetzung wird in Kapitel (4) genauestens eingegangen.

Dem Diagramm ist der geplante Ablauf zum jetzigen Zeitpunkt und Funktionsweisen des Unterstützungssystems zu entnehmen. Dabei ist grob veranschaulicht, welche Möglichkeiten sich dem Nutzer bieten, um einen Überblick zu verschaffen, anhand welcher Informationen die Applikation entwickelt wird.

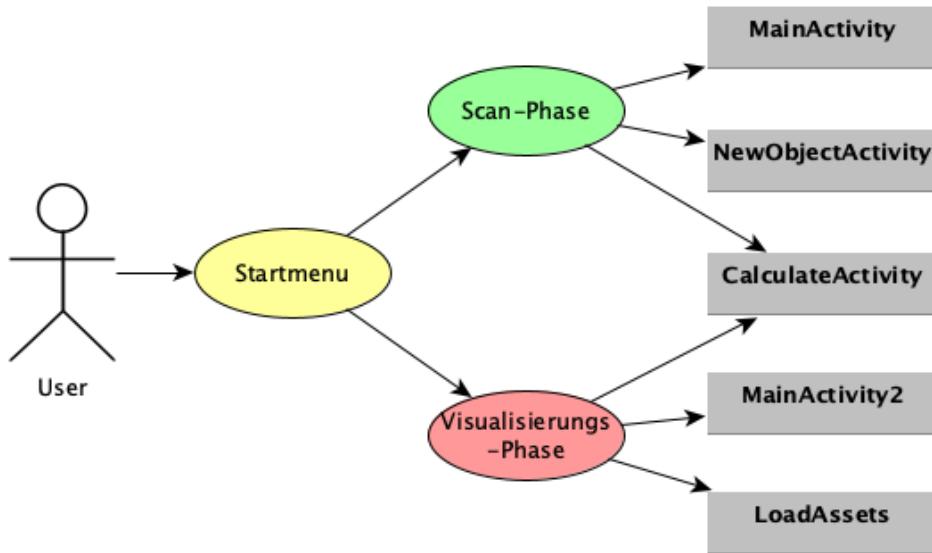


Abbildung 3.2: Anwendungsfall-Diagramm

Nachdem auch das Softwarekonzept weitestgehend erarbeitet wurde, ging es um die Entscheidung, welches Augmented Reality-Framework für dieses Projekt ausgewählt wird. Nun wird die Evaluierung der Frameworks aufgegriffen, um einen Einblick in die Entscheidungsfindung zu gewähren.

### 3.6 Auswahl des AR Frameworks

Mittlerweile gibt es eine enorme Auswahl an AR-Frameworks, die alle unterschiedliche Präferenzen und Einsatzmöglichkeiten haben. Somit sind, auf den Bereich bezogen, Vor- und Nachteile im Vergleich von mehreren Frameworks nicht ausgeschlossen. Einige Alternativen wurden getestet und auf deren Brauchbarkeit evaluiert und analysiert. Dazu wurden Kriterien ausgearbeitet, die die Auswahl an Frameworks einschränken und nach Möglichkeit das passendste ergeben soll:

1. Eine performante Darstellung von Objekten.
2. Möglichkeiten zur Positionsbestimmung.
3. Eine aktive Community und stetige Weiterentwicklung des Systems.
4. Möglichkeit zur Integration weiterer Technologien.
5. Open Source-Projekt, um Flexibilität und weitestgehende Unabhängigkeit zu gewährleisten.

Aufgrund der großen Anzahl an AR-Frameworks war es nicht möglich alle in Betracht gezogenen Frameworks detailliert aufzuführen. Lediglich die engere Auswahl der Tools wird aufgegriffen.

Nach ausführlicher Recherche wurden letzten Endes drei Frameworks in die nähere Auswahl aufgenommen, darunter *vuforia*, *ARToolKit* und *Google ARCore*. Beweggründe zu dieser Entscheidung waren die überwiegende Übereinstimmung zu zuvor aufgestellten Kriterien und Anforderungen. Alle Technologien konnten eine aktive und große Community, sowie eine aktive Entwicklung, bzw. Weiterentwicklung vorweisen. Zusätzlich bieten alle Frameworks viele Möglichkeiten zur Integration weiterer Technologien, um nach Belieben immer mehr Funktionen in das zu entwickelnde System übernehmen zu können. Die Performance der Technologien konnte in allen Aspekten überzeugen und ist unter einfachen Bedingungen mehr als ausreichend. Ein Kritikpunkt gegen die Verwendung von *vuforia* war die fehlende Verfügbarkeit des Quellcodes, da dieser unter kostenpflichtiger Lizenz steht und nicht als Open Source-Projekt gilt. Das Framework *vuforia* wurde daher nicht in Betracht gezogen.

### 3.6.1 ARToolKit

*ARToolKit* ist ein SDK zur plattformunabhängigen Entwicklung von Augmented Reality Anwendungen. Der Quellcode dieses Frameworks ist seit 2001 frei verfügbar und ist vielseitig einsetzbar. Das Verfahren zur Positionsbestimmung durch Marker läuft schnell, problemlos und robust. Marker wie in Abbildung (2.4) werden schnell und ohne Probleme erkannt. Jedoch gibt es diesbezüglich Einschränkungen die beachtet werden müssen, um eine zuverlässige und stabile Markererkennung zu gewährleisten. Zu verhindernde Defizite sind Verdeckung des Markers, zu große Distanz von Kamera zu Marker, Blickwinkel auf den Marker und die Lichtverhältnisse des Umfelds.

Nachteilig dieses Frameworks für dieses Projekt ist die fehlende Unterstützung bei der Realisierung und Erkennung der Umgebung, bzw. des Umfelds. So ist das Tracking nur über Marker, bzw. erweiterte markerbasierte Tracking-Methoden möglich.

Nach dieser Erkenntnis wurde auch dieses Framework als eher ungeeignet für die Verwendung in dem Projekt eingestuft. Möglicherweise gäbe es einen Weg, einen SLAM-Algorithmus in Kombination mit *ARToolKit* unter erweitertem Aufwand zu verwenden, allerdings erschien dies zu umständlich und ineffektiv.

### 3.6.2 Google ARCore

*Google ARCore* (siehe Abschnitt 2.4.1) konnte bei der Analyse von sich überzeugen. Lediglich die Beschränkung der Betriebssysteme auf *Android* und *iOS* ist ein kaum bedeutender Nachteil, da das

Assistenzsystem in der Gesamtbetrachtung für ein Smart-Device konzipiert und entwickelt wird. Der entscheidende Vorteil bei ARCore ist der unterstützende Algorithmus zur Umgebungserkennung. Des Weiteren ist Google ARCore ein Open-Source Projekt und befindet sich in kontinuierlicher Weiterentwicklung. Neben der stetigen Entwicklung hat sich eine große und aktive Community gebildet, die bei auftauchenden Problemen zur Stelle ist. Ein weiterer Vorteil sind die verschiedensten Möglichkeiten der Interaktion mit AR-Objekten. Es gibt die klassische Markererkennung, die nicht nur mit Markern (siehe Abbildung 2.4) sondern auch mit Bildern funktioniert, dem sogenannten *AugmentedImage*. Darüber hinaus ist ARCore in der Lage Gesichter zu erkennen und diese als abgewandelte Marker zu verwenden, um eine konkrete Position ansprechen zu können. Diese Methode ist unter dem Namen *AugmentedFace* bekannt. Eine weitere Methode ist die Erkennung von realen Punkten, Oberflächen und Gegenständen. So können virtuelle Objekte möglichst realitätsnah in den Raum visualisiert werden.

Durch diese ganzen positiven Aspekte ist die Wahl des Frameworks auf das von Google entwickelte ARCore gefallen und wurde wegen den überzeugenden Grundlagen als Framework für dieses Projekt eingesetzt.

### 3.7 Datenmodell

Eine gute Basisstruktur der Datenspeicherung ist bei heutigem Datenverkehr sehr wichtig und hat hohe Priorität. Deshalb ist es sinnvoll, schon von Anfang an eine klare Richtlinie und Struktur zu erstellen, um bei nachträglichen Ergänzungen oder Änderungen zusätzlich so wenig Zeit wie möglich investieren zu müssen. Parallel zur Konzeption der Softwarearchitektur wurden Attribute zusammengetragen, die eine Basis von Informationen über verschiedenste Objekte aufgreifen soll. Mit den Objekten sind Gegenstände der realen Welt adressiert, die auch als virtuelle AR-Objekte repräsentiert werden.

Bei der Datenmodellierung wurde entschieden, dass in der Scan-Phase (3.2), beim Erstellen eines Objekts, nur die notwendigsten Daten erhoben und gespeichert werden. Bei diesen Informationen handelt es sich um sogenannte Stammdaten. Bestandteile dieser Stammdaten sind Name, Größe und Baujahr des Objekts. Dazu kommen noch weitere relevante Informationen, darunter eine eindeutige ID zur genauen Erkennung eines Objekts und die virtuelle Position des AR-Objekts. Die virtuelle Position ist durch die Scan-Phase gegeben und wird benötigt, um die Gegenstände in der Visualisierungs-Phase an der richtigen Stelle erneut einblenden zu können.

Durch ein Entity-Relationship-Modell (ERM) wurden die benötigten Daten zusammengetragen, aufgeführt und zu einem Modell konzipiert, um den Strukturen und Vorgaben der Datenmodellierung zu entsprechen und eine gute Basisstruktur zu erschaffen. Mit dieser Grundlage kann in Zukunft weitergearbeitet werden. Das Modell wurde mit der zuvor angesprochenen dritten Normalform umgesetzt, da diese die perfekte Balance in dem Datenmodell herstellt, um auftretende Probleme möglichst einfach zu lösen und diese nahe der Realität in das relationale Datenbankmodell einzupflegen. Dadurch kann eine Erweiterung des Datenmodells gewährleistet werden. Die Abbildung (3.3) zeigt das erstellte ER-Modell auf und veranschaulicht die Modellierung.



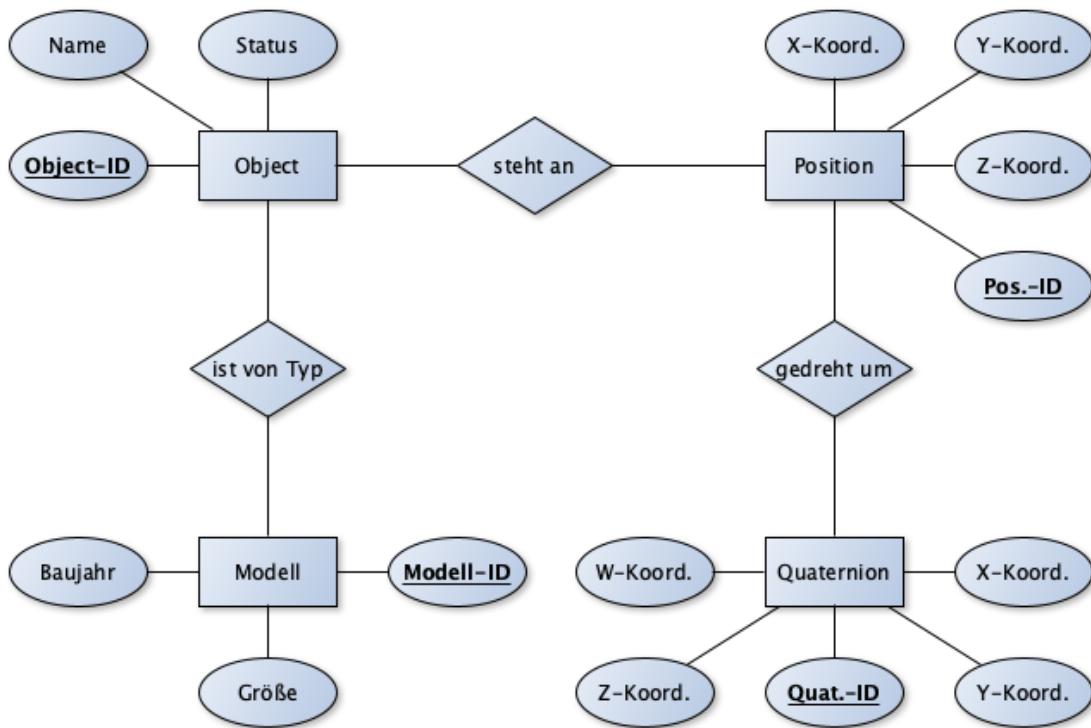


Abbildung 3.3: Entity Relationship Datenmodell

Nach Beendigung des Datenkonzepts waren alle notwendigen Konzeptionen abgeschlossen. Diese wurden in Summe zusammengetragen, im Gesamten nochmals betrachtet und anhand neu gewonnener Kenntnisse überarbeitet und evaluiert. Da zu diesem Zeitpunkt alle Konzeptionen abgeschlossen waren, konnten diese zusammen analysiert und überdacht werden.

Nachdem die Phase der Konzeption endgültig abgeschlossen war, konnte die Umsetzung des Konzepts und die Entwicklung des Systems starten.

# Kapitel 4

## Umsetzung

Dieses Kapitel greift die in der Konzeption (3) aufgeführten Planungen und Ideen auf und legt die Implementierung, bzw. Umsetzung detaillierte dar. Darunter sind Lösungsansätze aufgezeigt und aufgetretene Probleme und deren Behebung dokumentiert. Allgemein zählt dazu die Umsetzung des Startmenüs und der beiden Kernfunktionen. Sowohl die Frontend- als auch die Backend-Aspekte werden in der Implementierung (4.1) aufgezeigt.

### 4.1 Implementierung

Nachdem die Konzeption endgültig abgeschlossen war, ging es an die Umsetzung des Konzepts und an die Implementierung der Funktionen, die das System ausmachen.

Die Use Cases und deren Implementierung wurden nach logischer und chronologischer Reihenfolge entwickelt und dokumentiert. Diese festgelegte Reihenfolge ist auch der Abbildung (3.2) zu entnehmen. Angefangen mit dem Startmenü, das dem Nutzer die Möglichkeit offenbart, zwischen den Funktionen zu wählen, folgte die Implementierung der Scan-Phase, in der die realen Objekte virtualisiert und im Raum platziert werden. Zu guter Letzt kam die Visualisierungs-Phase, welche aufbauend auf die Scan-Phase funktioniert und die zuvor gescannten Objekte erneut virtuell im Raum einsetzt.

Beim ersten Gebrauch der Anwendung ist eine vorzeitige Nutzung der Visualisierungs-Phase ohne weiteres nicht möglich. Zuvor muss der Nutzer einen Scan durchführen, sodass das System Daten generiert, Informationen liefert und diese zur Verfügung stellt. Dies hat zur Folge, dass die zweite Kernfunktion mit den zuvor erzeugten Informationen durch die Scan-Phase operiert und ausgeführt werden kann.

Aufgeteilt wurden die Use Cases der Anwendung jeweils immer nach Frontend und Backend, demnach wird auch die Dokumentation in diesem Stil geschildert.

Bevor die Umsetzung verwirklicht werden konnte, wurden vorab noch die letzten Vorbereitungen durchgeführt.

Zum Start der eigentlichen Implementierung war es die Aufgabe, die Entwicklungsumgebung zu

wählen und einzurichten, um bestmöglich arbeiten zu können. Als Integrated Development Environment (IDE) wurde die speziell für Android-Applikationen entwickelte Software *Android Studio* ausgesucht. Diese ist besonders für die Entwicklung von Android-Applikation geeignet, ausschließlich für Hardware, dessen Software das Android Betriebssystem nutzt.

Android Studio ist von Google LLC. und JetBrains entwickelt und basiert auf der IntelliJ IDEA Community Edition IDE. Als Build-Management-Automatisierungs-Tool stellt Android Studio das Tool Gradle zur Verfügung, welches die zu bauenden Projekte durch die verwendeten Dependencies, Frameworks und Tools beschreibt. Um die notwendigen Libraries in der Applikation verwenden zu können, werden in dieser Datei unter anderem die Bibliotheken und deren verwendete Version eingetragen. Im Fall des zu entwickelnden Unterstützungssystems wurden dort die Bibliotheken zur Nutzung der Android Architecture Components eingetragen, darunter Room und LiveData. Darüber hinaus wird dort auch die Version des ARCore Frameworks und des Sceneform SDKs verwaltet, welches ebenso essentielle Bestandteile der Applikation sind.

Nachdem alle Abhängigkeiten erfolgreich eingebunden waren, wurden zur übersichtlichen Gestaltung der Klassen, Objekte, ViewModel, Repositories und Activities eine Ordnerstruktur angelegt, die alle Klassen des gleichen Typs im Laufe der Entwicklung beinhalten sollten. Dadurch kann bei der Programmierung besser und übersichtlicher durch das Projekt navigiert werden und es befinden sich alle Klassen ähnlicher Eigenschaften im selben Zielordner.

Damit im Laufe der Entwicklung die Applikation bestmöglich getestet werden konnte, war es notwendig, ein Testgerät zu verwenden. Mit dem verfügbaren Emulator<sup>1</sup> von Android Studio konnte keine realitätsnahe Testung durchgeführt werden. Deshalb wurde zum Testen der Anwendung ein Smartphone benutzt. Dieses musste zu Anfang unter den vorbereitenden Maßnahmen des Projekts organisiert werden.

Nachdem nun alle Schritte abgeschlossen waren, ging es an die Entwicklung des ersten Use Cases, dem Startmenü, um die Grundlage des Systems zu festigen.

#### 4.1.1 Startmenü

Das Startmenü ist die zentrale Anlaufstelle des prototypischen Unterstützungssystems und bildet den Einstiegspunkt in die Interaktionen zwischen Anwender und der Applikation, dem Assistenzsystem. Nun folgt die Beschreibung der Implementierung des Startmenüs, sowohl des Frontends als auch des Backends.

##### Frontend

Wird das Unterstützungssystem heruntergeladen, genauer gesagt auf die verwendete Hardware geladen, folgt das Speichern der Anwendung auf dem Gerät. Die Applikation erscheint auf dem Applikationshauptmenü des Smart-Devices und wird dort zur Nutzung zur Verfügung gestellt. Beim Start öffnet sich das Hauptfenster der Applikation und darin das Startmenü, auf das im Verlauf noch genauer eingegangen wird.

---

<sup>1</sup>Ein System, das ein anderes in speziellen Teilbereichen nachbildet

Zuallererst werden bei initialer Instandsetzung bestimmte Genehmigungen eingeholt, um die Anwendung überhaupt starten zu können. Darunter soll der Nutzer dem System die Erlaubnis erteilen, auf die Kamera zugreifen zu dürfen. Um die Umgebung in der Hauptfunktion scannen zu können, muss diese in weiterer Abfolge des Assistenzsystems genutzt werden können. Diese Abfrage erfolgt über ein kleines PopUp-Fenster, welches von Android fest vorgegeben ist. Dieses Fenster ist der Abbildung (4.1) zu entnehmen. Mit dem Zulassen des Zugriffs auf die Kamera kann der Nutzer und die Applikation wie gewünscht fortfahren. Wird allerdings der Zugriff abgelehnt, kann die Applikation nicht starten, bzw. wird der Zugriff auf die Kamera verweigert, kann die Applikation nicht einwandfrei benutzt werden.

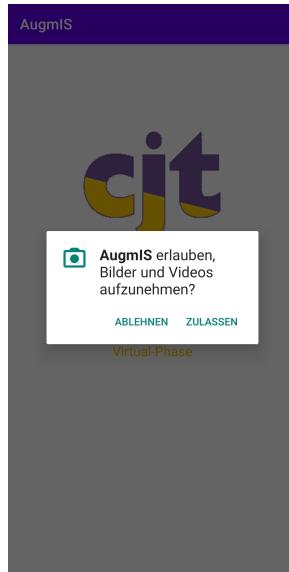


Abbildung 4.1: Start der Applikation

Da speziell das Assistenzsystem auf die gegebenen Komponenten der Hardware zugreift, z. B. die im Smart-Device integrierte Kamera, zählt diese zu den nativen Applikationen. Darunter sind Anwendungen zu verstehen, die ausdrücklich für das Betriebssystem des Endgeräts konzipiert sind oder die zum Beispiel auf die vom Endgerät vorhandenen Komponenten zugreifen, wie in etwa auf die Bild-Mediathek, der Datei-Strukturen oder die oben bereits erwähnte Kamera-Komponente.

Angenommen der Nutzer lässt den Zugriff auf die Kamera für das Assistenzsystem zu, dann startet die Applikation wie erwünscht. Darauffolgend ist dann das eigentliche Startmenü zu sehen. Diese Ansicht ist der nachfolgenden Abbildung (4.2) zu entnehmen. Die Benutzeroberfläche weist ein sehr einfaches und schlicht gehaltenes Design auf. Darauf ist das Firmen-Logo der cjt Systemsoftware AG zentriert zu sehen, außerdem zwei farblich voneinander getrennten Buttons zur Navigation innerhalb der Applikation. Die Buttons adressieren jeweils die nachfolgende, zugehörige Funktion. Dies sind zum einen der lilafarbenen Button, der zur Scan-Phase weiterleitet und zum anderen der gelbfarbene Button, der den Nutzer zur Visualisierungs-Phase navigiert. In der Gesamtheit ist das

Layout an den Farben des Firmen-Logos angelehnt und bildet einen ruhigen und harmonischen Einstieg in das Programm.

Grundlegend wurden für die Designs der Benutzeroberflächen Prinzipien der User-Experience, dt. Nutzererfahrung und Nutzererlebnis (UX), berücksichtigt. Darunter folgende Punkte:

- Übersichtlichkeit (Digestibility): Auf einen Blick verstehen, worum es geht. Der User versteht intuitiv was zu tun ist.
- Klarheit (Clarity): Deutliche und verständliche Ausdrucksweise und klar herauszufindenden Nutzen der Funktion, bzw. Anwendung.
- Vertrauen (Trust): Gutes Design erlangt Vertrauen. Offenlegung der Aktionen und Hintergründe.
- Begeisterung (Delight): Komplexe Sachverhalte einfach zu lösen.

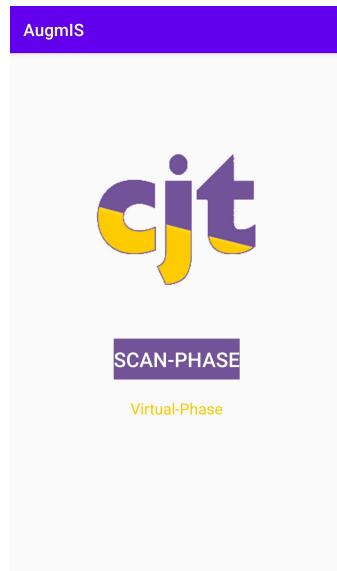


Abbildung 4.2: Startmenü der Applikation

Nachdem das Design des Startmenüs und dem PopUp-Fenster abgeschlossen war, ging es darum, die Funktionalität der UI-Komponenten zu implementieren.

### Backend

Durch die schlichte und einfach gehaltene Modellierung des Startmenüs, zählt dieses als Ansicht zum Einstieg in die Nutzung des Assistenzsystems, zur Abfrage der „*Camera-Permission*“ und zur weiteren Navigation innerhalb der Anwendung.

Mit der „*Permission-Request*“ wird abgefragt, ob die Genehmigung zur Nutzung der Kamera in der „*AndroidManifest.xml*“-Datei erteilt wurde, bzw. dadurch wird das PopUp-Fenster zur Nachfrage der Genehmigung geöffnet und die daraus resultierende Antwort in die Manifest-Datei eingetragen.

Die Manifest-Datei beinhaltet alle optionalen Metadaten, die für das Assistenzsystem benötigt werden. Dazu gehören die Google ARCore-Metadaten, sowie grundlegende Informationen über das

Projekt, die das Android-System erfordert, um die Applikation installieren und ausführen zu können. Dazu wird unter anderem die Berechtigung zur Kameranutzung benötigt, ebenso werden Angaben zu Activities, bzw. Benutzeroberflächen gemacht und Stammdaten, z.B. Titel, -bild und App-Icon, die das Projekt eindeutig spezifizieren, definiert.

Demnach beinhaltet die Backend-Funktion des Startmenüs die Abfrage der „*Camera-Permission*“ und die Button-Klick-Funktionen, die den Nutzer jeweils auf die von ihm gewählte Funktion weiterleiten. Sozusagen handelt es sich um eine simple Navigation.

Als das Startmenü implementiert war, ging es an die Entwicklung der Scan-Phase, der ersten Kernfunktion des Assistenzsystems.

### 4.1.2 Scan-Phase

Bei der Implementierung der Scan-Phase kam es zu den ersten praktischen Berührpunkten mit der Google ARCore API. Diese wurde zu Beginn studiert, um einen Überblick, wie die Struktur dieser Schnittstelle aufgebaut ist, zu erlangen. Dadurch konnten Methoden und Funktionen analysiert werden, um im weiteren Verlauf der Entwicklung damit arbeiten zu können.

Die Idee hinter der Scan-Phase ist die Implementierung einer Methode, die es ermöglicht, das Umfeld über die Kamera wahrzunehmen und mit den gewonnenen Erkenntnissen über den Raum weiterarbeiten zu können. Die nachstehende Funktion, virtuelle Objekte auf Basis der vom Scan erfahrenen Kenntnissen in der Räumlichkeit zu platzieren, arbeitet mit diesen in Erfahrung gebrachten Kenntnissen.

Zur Schaffung einer Grundlage, um die wichtigsten Funktionen zu implementieren, wurden zuallererst die Benutzeroberflächen erstellt. Diese dienen unter anderem dazu, den Fokus für die Funktionsweise der eigentlichen Methode nicht zu verlieren. Demnach werden die entwickelten GUIs nun aufgeführt und näher beschrieben, um Methoden des Backends im Anschluss besser nachvollziehen zu können.

#### Frontend

Das Assistenzsystem verfügt über eine Vollbildkamera-Ansicht, die mit einem weißen Rahmen überlagert ist. Damit können über die Kamera Bilder zentriert eingefangen werden. Somit weiß der Nutzer, wie er das Smart-Device in Position bringen muss, damit er das zu scannende Bild über die Kamera bestmöglich erfassen kann.

Ursprünglich war die Applikation nicht mit diesem Layout (4.3) konzipiert, da während der Implementierung der Scan-Phase eine schwerwiegende Änderung vorgenommen werden musste. Diese wird in dem Abschnitt der Scan-Phasen-Backend-Entwicklung zu gegebenem Zeitpunkt aufgegriffen und genauestens dargelegt.



Abbildung 4.3: Markererkennung der Applikation zum Start der Scan-Phase

Des Weiteren beinhaltet das Assistenzsystem die eigentliche Scan-Phasen UI, mit der die Umgebung aufgenommen wird und anhand dieser Daten der Nutzer die Objekte an von ihm definierte Stellen platzieren kann. Die Benutzeroberfläche hat ebenso ein schlichtes und überschaubares Layout, welches dem Anwender zugutekommt.

Hauptsächlich beinhaltet die GUI ein Fragment, welches das Kamerabild in Echtzeit wiedergibt. Am oberen Ende des Bildschirms, bzw. der Abbildung (4.4) befindet sich eine lilafarbene Fläche, die zur Anzeige von Informationen vorhanden ist und gleichzeitig als erweiterte Abgrenzung zur Topdown-Leiste dient. Am unteren Ende des Bildes befindet sich eine „*Android-Gallery*“, die alle zu platzierenden Objekte beinhaltet. In der zugrundeliegenden Abbildung sind es zwei Platzhalter-Objekte, die bei Berührungen des Bildes erstellt werden. Prinzipiell ist diese Gallery um beliebig viele Objekte erweiterbar. Diese Erweiterung erfolgt allerdings nicht dynamisch, sondern muss über manuelles Einfügen im Code stattfinden.

Wird während der laufenden Anwendung ein Objekt über dessen Button erstellt, wird der Nutzer unverzüglich auf eine weitere Seite (4.5) geleitet, um parallel zur Renderung des Objekts Informationen diesbezüglich der Datenbank hinzufügen zu können. Auf dieses Layout wird im weiteren Verlauf der Ausarbeitung noch eingegangen. Vorwegzunehmen ist, dass die Informationen mit den Daten der virtuellen Position in der Datenbank persistiert werden.

Bevor der Scan-Vorgang startet, wird inmitten des Fragments ein Overlay angezeigt, welches eine Geste, bzw. Bewegung vorgibt. Imitiert der Nutzer diese Bewegung, wird der Scan-Prozess anlaufen.

Um weiterhin auf der Seite navigieren zu können, gibt es im rechten unteren Eck einen weite-

ren Button, welcher ebenso der Abbildung (4.4) zu entnehmen ist. Diese Schaltfläche leitet den Nutzer weiter, bzw. wieder zurück zum Startmenü und beendet die Scan-Phase.

Damit der Nutzer über die in der Datenbank gespeicherten Informationen mehr Möglichkeiten besitzt, gibt es im linken oberen Eck der UI eine zusätzliche Schaltfläche versehen mit einem roten Kreuz versehen. Dadurch hat der Nutzer die Möglichkeit, alle vorhandenen, in der Datenbank persistierten Informationen zu löschen und den Scan-Vorgang erneut von der ursprünglichen Ausgangsposition zu beginnen. Nachdem die Daten gelöscht werden, wird der Nutzer aufgefordert, den Ursprungspunkt, den zu „trackenden“ Marker, abzuscannen und die Phase erneut zu beginnen.

Mit dieser Funktion verfügt der Nutzer über mehr Entscheidungsmöglichkeiten, bevor die Applikation komplett gelöscht und neu aufgespielt werden muss.



Abbildung 4.4: Scan-Phase der Applikation

Zu dem Zeitpunkt, wenn der Anwender ein Objekt erstellen und zusätzliche Informationen hinzufügen möchte, wird darauffolgend eine weitere UI angezeigt. Bei dieser Ansicht ist vorgesehen, dass der Nutzer die zu dem erstellen Objekt dazugehörigen Informationen einträgt, sodass diese anschließend in die Datenbank aufgenommen werden können. Die Benutzeroberfläche verfügt über mehrere Text-Eingabefelder, die ein Großteil der Daten des Datenmodells abdecken. In Abbildung (4.5) sind diese Felder zusehen. Dabei werden Informationen über den Objektnamen, -baujahr, -größe, sowie -status von der Person, die das Assistenzsystem nutzt, gefordert.

Im Hintergrund werden ebenso die Daten der Position und Drehung des Objekts bereitgestellt. Die wird im Backend näher erläutert. Über die Betätigung des „*save*“-Buttons werden die Informationen lokal gespeichert und für die Eintragung in die Datenbank vorgemerkt. Nachdem der Button bedient wurde, wird der Anwender auf die Oberfläche der Scan-Phase weitergeleitet, um bei Bedarf weitere Objekte zu erstellen.

The screenshot shows a web-based form titled "cjt-AugmIS". It contains four input fields: "Object name...", "Objects creation date...", "Objects size...", and "Set state of object '1' or '-1'...". Below these fields is a large, empty rectangular area. At the bottom right of this area is a blue button labeled "SAVE".

Abbildung 4.5: Erstellen eines neuen Objekts

Nachdem der Abschnitt der Frontend-Entwicklung vollends dargelegt wurde, geht es mit den Funktionen im Backend weiter. Darüber hinaus werden in diesem Abschnitt auch die Lösungsansätze, sowie aufgetretene Probleme und deren Behebung genauestens aufgeführt und geschildert, damit der Nutzer mit der aufgetretenen Problematik vertraut wird und die Entscheidungen und Lösungsfindungen nachvollziehen kann.

## Backend

Mit Beginn der Backend-Entwicklung wurden in erster Linie die Grundstrukturen implementiert, um der *MVVM*-Architektur gerecht zu werden. Dabei lag der Fokus bei der Instanziierung der notwendigen Komponenten der *Android Architecture Components*. Nach chronologischer Reihenfolge wurden die Klassen erstellt und mit den dazugehörigen Funktionen und Methoden versehen. Angefangen mit der Entity-Klasse zur Beschreibung des Objekts und deren allgemeinem Aufbau, der bereits in der Konzeption (3) unter dem Datenmodell (3.7) festgehalten wurde. Darauffolgend wurde das „Data Object“ erstellt, welches die Zugriffe der Datenbankobjekte verwaltet. Abschließend wird im Bereich der Datenbank ein Room-Layer eingebaut, um die eigentliche Datenbank zu instanziieren und eine Zugriffsschicht auf diese zu implementieren.

Zur Modularisierung und Generierung einer Schnittstelle zur Kommunikation zwischen einzelnen Datenbeziehungspunkten und der Datenbank wurde ein Repository erstellt, das aus verschiedenen Quellen Daten zusammenbringen kann und eine saubere API für den Datenzugriff auf den Rest der Anwendung bietet. Um die vorhandenen Klassen zum Datentransfer und zum Persistieren der Daten mit der eigentlichen Benutzeroberfläche zu verbinden, wurde ein ViewModel implementiert, welches die Daten zwischen den einzelnen Komponenten teilt und bereitstellt.

Nach Aufzählung der einzelnen Bestandteile wird auf diese nun genauer eingegangen.

In einer Java-Klasse wird mithilfe der gegebenen Bibliothek „Room“ ein Entity-Objekt erstellt. Hierbei gibt es eine eindeutige Annotation, die die Klasse und deren beinhalteten Variablen deklariert. In dem folgend aufgeführten Code-Beispiel (4.1) ist diese Annotation zu entnehmen, in dem eine Klasse als Entity deklariert und somit in eine Datenbank-Tabelle konvertiert wird. In dieser Tabelle, definiert als „*object\_table*“, gibt es weitere Variablen, die zuerst mit Annotationen versehen und dann gemäß der Anforderungen des Konzepts sind. Diese Variablen repräsentieren die Attribute des Datenbankschemas und stellen die einzelnen Informations-, bzw. Datenbankspalten dar. Zur Veranschaulichung dient die Initialisierung der ID-Vergabe eines Objekts. Diese wird als „*id*“-Spalte und ebenso als Primärschlüssel<sup>2</sup>-Variable deklariert.

---

```

1  @Entity(tableName = "object_table")
2  public class Object {
3      @ColumnInfo(name = "id")
4      @NonNull
5      @PrimaryKey(autoGenerate = true)
6      private int id;
7
8      public int getId() { return this.id; }
9      public void setId(int id) { this.id = id; }
10     ...
11 }
```

---

<sup>2</sup>Einmaliger und eindeutiger Wert einer Tabelle, bzw. eines Attributs, um dieses eindeutig zu kennzeichnen.

---

Code-Beispiel 4.1: Entity Code zur Initialisierung der Objekte

Das mit der Entity-Klasse kommunizierende Modul ist das „*Data Object*“, welches als Interface angelegt wurde. Das „*Object Dao*“ verwendet ebenso die Bibliothek „Room“ und wandelt die Java-Klasse per Annotation in ein „*Dao*“ um. Dieses beinhaltet hauptsächlich die SQL-Queries zur Datenabfrage der vorhandenen Informationen.

---

```

1 @Query("SELECT * FROM object_table ORDER BY name")
2 LiveData<List<Object>> getObjectName();
```

---

Code-Beispiel 4.2: SQL-Query zur Abfrage der Objekt-Namen

Nachdem die beiden Klassen erstellt worden sind, wurde das Datenbank-Layer auf der eigentlichen Datenbank implementiert. Über einen „Builder“ wird die Datenbank-Instanz erzeugt und mit einem Namen versehen. Im Falle des Assistenzsystems als „*object\_database*“ deklariert.

In zukünftigen Entwicklungen, falls notwendig, gäbe es die Möglichkeit, in diesem Schema weitere Datenbanken zu erstellen und diese über weitere „*Dao*“s zu referenzieren. Darüber hinaus ist die Möglichkeit gegeben, weitere Datenbank-Tabellen zur Speicherung von Objekten und deren Informationen zu erstellen. Ebenso ist durch ein Repository die Option geboten, die derzeit auf dem Smartphone gespeicherte Datenbank auf einen externen Server auszulagern, um so die Daten weitläufiger zur Verfügung zu stellen. Eine Datenbeziehung von generierten Daten der Maschinen und Geräten selbst wäre auch vorstellbar, allerdings müssten diese vorab noch aufbereitet und zur Nutzung bereitgestellt werden. Diese Methode wird allerdings nicht näher betrachtet, da sie nicht Teil dieser Arbeit ist.

Im Ausblick (7) wird darauf nochmals eingegangen.

---

```

1 @Database(entities = {Object.class}, version = 1, exportSchema = false)
2 public abstract class ObjectRoomDatabase extends RoomDatabase {
3     ...
4     INSTANCE = Room.databaseBuilder(context.getApplicationContext(),
5         ObjectRoomDatabase.class, "object_database")
6         .addCallback(sRoomDatabaseCallback)
7         .build();
8     ...
9 }
```

---

Code-Beispiel 4.3: Erzeugung des Datenbank-Layers „Room“

Letzter wichtiger zu implementierender Aspekt war das Kommunikations-Modul zwischen den Datenbank-Transferen und der Benutzeroberfläche, das *ViewModel*. In dieser Klasse wird beim Start der Anwendung eine Liste anhand der Objekte in der Datenbank erstellt, welche die Informationen des einzelnen Objekts besitzt. Diese werden über eine „*get*“-Funktion aus dem zuvor instanzierten Repository geladen und in der erzeugten Liste lokal abgelegt. Diese Liste wird dann für die Präsentation der Informationen auf der Nutzeroberfläche verwendet. Bei Änderungen der Informationen wird über die UI der „Oberserver“ benachrichtigt. Dieser registriert die Änderungen und überträgt

sie nach vollständigem Abschluss der Transaktion in die Datenbank. Der Transfer erfolgt wiederum über das Repository, da dort die Datenbankzugriffe verwaltet und aufgerufen werden.

Die ursprünglich anders angedachte Implementierung der eigentlichen Scan-Phase unter Beachtung der ARCore-API stellte sich im Laufe der Entwicklung als nicht praktikabel heraus. Auf die darauf folgende Änderungen und die aufgetretene Problemstellung wird weiter unten näher eingegangen. Das Fragment auf dem User Interface der Scan-Phase (4.4) wird als AR-Fragment, unter der Benutzung des Sceneform SDKs, deklariert. Basierend auf dieser Initialisierung können die Interaktionen mit den ARCore-, bzw. Sceneform- Elementen gewährleistet werden. Dadurch ist ebenso die Nutzung der Kamera gegeben. Diese Funktionen sind Bestandteile der ARCore- und Sceneform-API. Über einen „*FragmentManager*“ wird die „.xml“-Datei mit dem darin enthaltenen Fragment der ID „*sceneform\_fragment*“ initialisiert.

---

```

1 private ArFragment fragment;
2 ...
3 fragment = (ArFragment)
4 getSupportFragmentManager().findFragmentById(R.id.sceneform_fragment);
5 ...

```

---

Code-Beispiel 4.4: Initialisierung des Fragments

Um anschließend Objekte auf diesem Fragment einblenden zu können, ist es vorab notwendig, eine „*Session*“ zu generieren, die unter anderem für die Konfigurationen der Kamera zuständig ist. Dadurch wird bei Start der Scan-Phase ein dreidimensionales Koordinatensystem erstellt, welches den Ursprungspunkt der Anwendung darstellt. Mit Verwendung der internen Sensoren des Smartphones werden darauffolgende Bewegungen registriert und mittels SLAM Verfahren berechnet. So ist die Kalkulation der Lokalisierung möglich und kann anhand der Kamera die Umgebung mit Hilfe des SLAM Verfahrens abgebildet werden. Durch diese Gegebenheiten wird die virtuelle Karte des Umfelds erzeugt und dient so zur Veranschaulichung der zu platzierenden Objekte. In der Abbildung (4.6) ist ein Beispiel zu sehen, indem ein Objekt erstellt wird, welches die Koordinaten von dem Ursprungspunkt berechnet. So wird deutlich, wie die Position eines Objekts unter der Verwendung des von ARCore gegebenen SLAM Verfahrens errechnet wird. Voraussetzung dafür ist, dass der Anwender an der Position (x,y,z = 0) startet, sich frei im Raum bewegt und ein Objekt an Postion (x = 5, y = 1, z = 3.5) setzt.

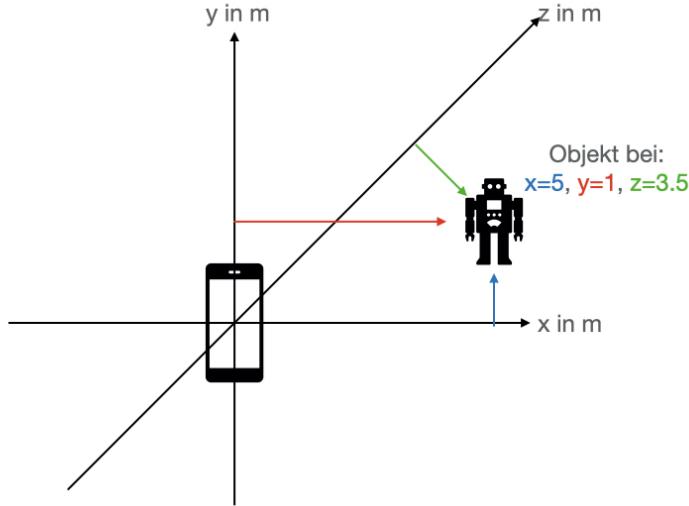


Abbildung 4.6: Aufbau der Positionsberechnung von Objekten

Beim Setzen eines Objekts wird als Anhaltspunkt der Mittelpunkt des Bildschirms berechnet, um das Objekt auf dem Bildschirm zentriert platziieren zu können. Ist diese Position berechnet, folgt durch Hilfe des SLAM Verfahrens die genaue Ermittlung der virtuellen Position. Anhand der Information dieser Position ist das Objekt zu platzieren.

Das Objekte wird gesetzt, indem der Nutzer in der UI (4.4) im Bereich der „Gallery“ ein Objekt anklickt. Über die der Sceneform-API zur Verfügung gestellten Methode („ModelRenderable“) wird das Objekt erschaffen und als „Anchor“ auf die berechnete Position gesetzt. Ein Anchor ist die fixe Position des Objekts, an dem dies platziert wird und so lange dort vorhanden bleibt, bis die Applikation beendet wird. Die Datei, die als dreidimensionales „asset“ generiert werden soll, wird über die „uri“-Variable lokalisiert und als Parameter übergeben.

---

```

1 ModelRenderable.builder()
2 .setSource(owner.get(), uri)
3 .build()
4 .handle((renderable, throwable) -> {
5     ...
6 })

```

---

Code-Beispiel 4.5: ModelRenderable Builder

Während der Objekt-Renderung öffnet sich die GUI (4.5) zur Eingabe der Informationen, die sich auf das Objekt beziehen. Diese werden dann zusammen mit der virtuellen Position des Objekts abgegriffen und in die Datenbank geschrieben. Drückt der Anwender den „save“-Button der Oberfläche (4.5), beendet er diesen Vorgang und kehrt auf die UI der Scan-Phase zurück, um weitere Objekte platzieren zu können.

Zusätzlich zu der Position des Objekts wird dessen Rotation durch Quaternionen berechnet und in der Datenbank gespeichert, um anhand der Berechnung die Darstellung so real wie möglich wiederzugeben. Da die Rotation ausgehend von der Kamerahaltung berechnet wird, wird das Objekt bei der Ankerung immer in Blickrichtung der Kamera positioniert.

Ursprünglich war geplant, die generierte Session sowie die darin erstellten Objekte in der Datenbank abzuspeichern, um diese bei erneutem Aufruf der Scan-Phase, bzw. bei Anwendung der Visualisierungs-Phase zu laden und innerhalb dieser Session wieder anzeigen zu lassen. Da die Datenbank nur gewisse Datentypen zulässt, war die erste auftretende Schwierigkeit, die Speicherung der Session als Objekt, die zunächst in eine „BLOB“-Datei hätte konvertiert werden müssen. Eine „BLOB“-Datei ist ein Binary Large Object, welches anhand der Binärcodierung abgespeichert wird. Dabei kann es sich unter anderem um große Bild- oder Audio-Dateien handeln, ebenso können auch mittels dieser Objekt-Konvertierung anderweitig große Dateien gespeichert werden. Daraus entstand die Konsequenz, die die Visualisierungs-Phase betraf, nämlich dass eine erzeugte Session nur eine gewissen Zeit verwendet werden kann. Dies bedeutet, dass nach erneutem Starten des Assistenzsystems alle zuvor erzeugten Objekte einer Session auf der ihnen zugewiesenen Position nicht mehr erreichbar sind. Durch die Unterschiedlichen Startpunkte, die bei wiederholtem Starten der Anwendung erzeugt werden würden, würde sich die ursprüngliche Ausgangsposition der Applikation verschieben und so auch die Objekte an eine fälschliche Position projizieren. Genauer gesagt, müssten die Objekte dann ausgehend von dem neu erzeugten Startpunkt referenziert werden. Somit wäre das Ergebnis nicht exakt und könnte keine Anwendung finden, da die Applikation Informationen anzeigen würde, die der Realität nicht entsprechen würden.

Basierend auf dieser Erkenntnis musste umdisponiert und ein neuer Lösungsansatz konzipiert werden. Mit dem Wissen über den aktuellen Stand der ARCore API galt es eine Lösung zu entwickeln, mit der Positionsinformationen exakt wiedergegeben werden können.

Dieser Ansatz wird nun erläutert.

Die Idee war es, einen Fixpunkt zu erstellen, welcher dazu dienen würde, einen immer gleichbleibenden Startpunkt vorzugeben. Dadurch gäbe es keine Unterschiede des Ausgangspunktes mehr und der Nutzer könnte trotz seiner aktuellen Position die Anwendung starten. Dazu müsste er beim Start der Applikation an den Ursprungspunkt kehren, um daraufhin die Scanfunktion zu beginnen. Für diesen Ansatz würde ein Marker zur Verfügung gestellt werden (siehe Abbildung 4.7), der vor dem ersten Gebrauch der Software vom Nutzer an einer Position angebracht wird und an dieser dauerhaft bestehen bleibt. Somit ist ein immer gleichbleibender Ausgangspunkt geschaffen, von dem aus die Objekte referenziert werden können. Der durch die Anwendung bestimmte Marker müsste entweder als physikalisches Bild an einer bestimmten Stelle in der Realität angebracht werden oder als Bild-Datei auf einem Computer immer vorhanden und an der richtigen Position wiederzufinden sein, um die Genauigkeit zu gewährleisten.

Zur Umsetzung war es notwendig, eine weitere Funktion zur Applikation hinzuzufügen (siehe Abbildung 4.3 in Scan-Phase 4.1.2 Frontend), damit ein Marker verfolgt werden kann. Wird dieser Marker erkannt, folgt die eigentliche Scan-Phase, die nach der Umkonzeptionierung sowohl bei der Objektplatzierung als auch bei der Datenspeicherung ebenso überarbeitet wurde.



Abbildung 4.7: Marker zur Erkennung der Ausgangsposition

Angenommen der Nutzer startet die Applikation an einer beliebigen Position im Raum und platziert Objekte an von ihm vorgesehenen Stellen, dann würden die Objekte erstellt, aber keinerlei Anhaltspunkte geschaffen werden, um bei nachzutragenden Objekten die gleichen Voraussetzungen des zu referenzierenden Standpunktes zu erfüllen. Daher wird bei der Scan-Phase die Methode zur Markererkennung eingebaut, um einen initialen Fixpunkt zu erstellen. Dadurch ist es gewährleistet, ausgehend von diesem Punkt erneut Objekte zu platzieren, egal wo im Raum die Applikation gestartet wird. Demnach werden die Objekte immer abhängig zur Position des Fixpunktes gespeichert, erstellt und angezeigt. Dies bedeutet, dass die Objekte immer eine Referenz zu dem initialen Marker sind. Auch wenn ein Objekt nachträglich hinzugefügt werden soll, ist es lediglich erforderlich, den Marker einzuscannen. Danach begibt sich der Nutzer an den gewünschten Ort im realen Raum, an dem das Objekt in der Anwendung platziert werden soll. Jetzt muss er die Umgebung mit der Kamera aufnehmen und, indem er auf den Button für das Erstellen des Objekts drückt, dieses virtuell erschaffen. Bei der Erzeugung wird die Position ermittelt und anhand dieser die Differenz, bzw. der Abstand zu dem initialen Marker in der Datenbank über einen Schreibbefehl gespeichert. Die Berechnung der Distanz zwischen Objekt und Ursprungsmarker erfolgt durch eine präzise Subtraktion, bei der lediglich die Ursprungskoordinaten von den aktuellen Positionskoordinaten subtrahiert werden. Dieser Vorgang ist dem folgenden Code-Beispiel (4.6) zu entnehmen. Dabei wird das aktuelle Objekt „object“ und das initiale Objekt „initialObject“ als Parameter übergeben, die sogenannte „call by Reference“, um die Distanz berechnen zu können. Das Ergebnis wird in mehreren lokalen Variablen zwischengespeichert und zum Schluss in eine neue Position „Pose“ mit Translation und Rotation, von der Berechnung durch Quaternionen, ermittelt. Der Rückgabewerte dieser Funktion ist demnach eine „Pose“, die nach Aufruf der Methode übergeben und so in der Datenbank mittels „insert“-Befehl hinterlegt wird. Die Position und Rotation des einzelnen Objekts ist mit dem Resultat gegeben und kann in der Visualisierungs-Phase verwendet werden.

---

```

1 public Pose returnValueFromPosition(Object object, Object initialObject){
2     float tX = object.getTx() - initialObject.getTx();
3     float tY = object.getTy() - initialObject.getTy();
4     float tZ = object.getTz() - initialObject.getTz();
5
6     float qX = object.getQx() - initialObject.getQx();
7     float qY = object.getQy() - initialObject.getQy();
8     float qZ = object.getQz() - initialObject.getQz();
9     float qW = object.getQw() - initialObject.getQw();
10
11    Pose pose = new Pose();
12    pose.setTx(tX);
13    pose.setTy(tY);
14    pose.setTz(tZ);
15    pose.setQx(qX);
16    pose.setQy(qY);
17    pose.setQz(qZ);
18    pose.setQw(qW);
19
20    return pose;
21 }
```

```

7   float qY = object.getQy() - initialObject.getQy();
8   float qZ = object.getQz() - initialObject.getQz();
9   float qW = object.getQw() - initialObject.getQw();

11  float[] rotation = {qX,qY,qZ,qW};
12  float[] translation = {tX,tY,tZ};

14  return new Pose(translation, rotation);
15 }

```

Code-Beispiel 4.6: Berechnung der Distanz zwischen Marker und Ursprungspunkt

Zur Veranschaulichung der zuvor beschriebenen Methodik dient die Abbildung (4.8). Durch diese Skizze wird nochmals verdeutlicht, dass sich alles in einem erstellten Koordinatensystem abspielt und lediglich die Distanz der einzelnen Objekte zum Marker berechnet und in der Datenbank persistiert werden.

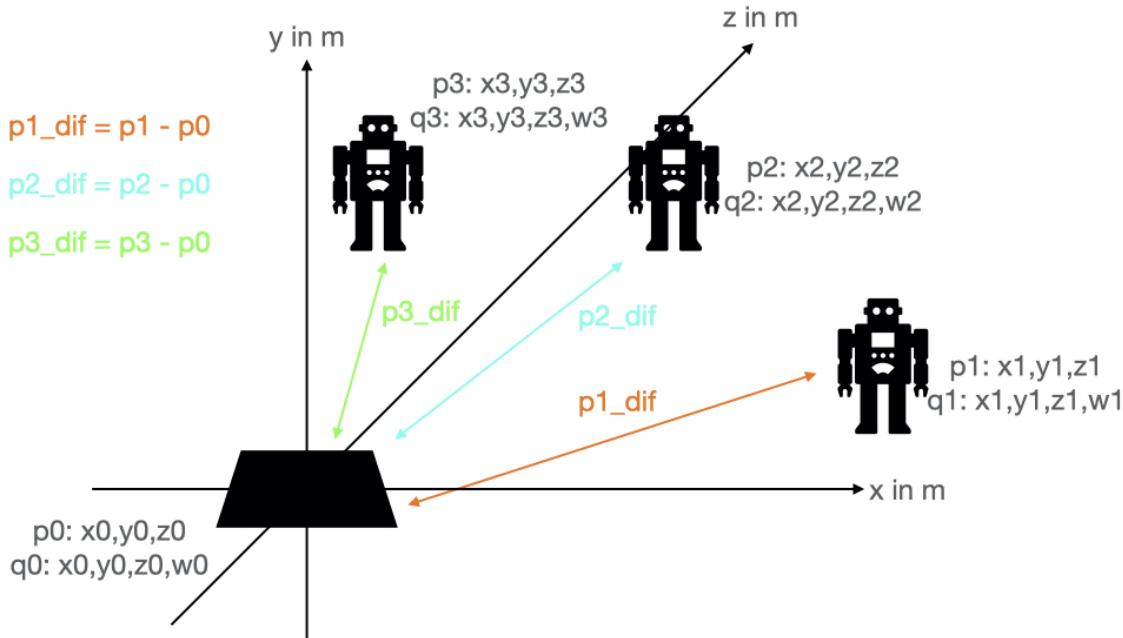


Abbildung 4.8: Positions berechnung zum Ursprungsmarker

Um die Backend-Implementierung zu demonstrieren, wird nun während der Laufzeit anhand erstellter Screenshots eine Beschreibung des Ablaufs der Applikation stattfinden.

In Abbildung (4.9 (a)) ist zu sehen, dass der vorgegebene Marker verfolgt werden muss, um die Applikation starten zu können. Dabei begrenzt die hervorgehobene Umrandung die Fläche, in der der Marker im Bildschirm zur Bildregistrierung platziert werden muss. Nach der Erkennung des Bildes, leitet die Benutzeroberfläche automatische auf die eigentliche Ansicht der Scan-Phase um.

Ist der Scan aktiviert, folgt die Wahrnehmung der Umgebung durch das SLAM Verfahren und die Berechnung der vorliegenden Oberflächen. Damit die zu setzenden Objekte exakt auf der Oberfläche des Gegenstandes, der Wand oder dem Boden platziert werden können, wird anhand der Sensoren im Smart-Device und dessen Kamera dieser dort berechnet. Dazu wird die berechnete Fläche mit einem Feld voller Punkte auf dem Bildschirm überlagert, um dem Nutzer ersichtlich zu machen, dass an dieser Position die Schätzung der Oberfläche durchgeführt wurde und die Anwendung darauf ein Objekt platzieren kann. Um dies zu verdeutlichen, wird inmitten des Bildschirmes (siehe Abbildung 4.9 (b)) ein grüner Punkt angezeigt, der die Platzierung des Objektes vorgibt und dem Nutzer verständlich macht, dass ein Objekt erzeugt und an dieser Stelle angesiedelt werden kann. Ist eine Oberfläche der Umgebung noch nicht vollständig berechnet, bzw. erkannt, fehlen die Punkte auf dem Bildschirm und es ist lediglich ein grau hinterlegtes Kreuz zu erkennen. Der Nutzer wird dadurch daran gehindert, ein Objekt an einer nicht bekannten und nicht berechnet Stelle zu platzieren.

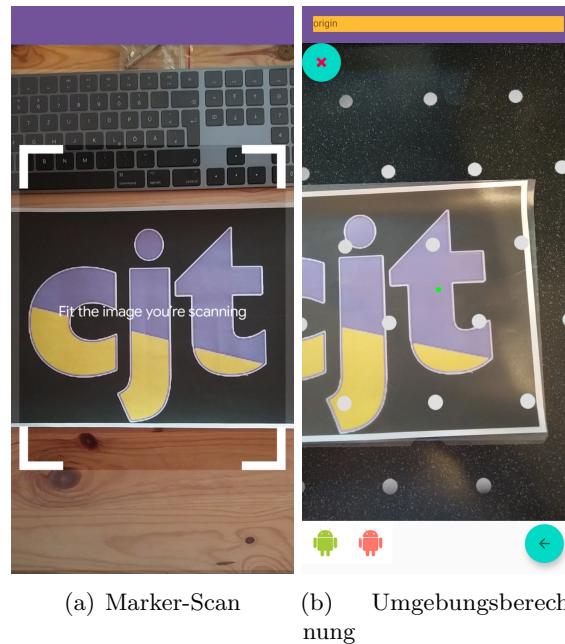


Abbildung 4.9: Funktion der Scan-Phase Teil 1

Wird nun ein Objekt vom Benutzer erstellt und an eine von ihm gewählte Position platziert, trägt er die dazugehörigen Informationen in der UI ein, bestätigt diese und übernimmt die Garantie dafür, das Objekt richtig positioniert und betitelt zu haben. Danach wird die ursprüngliche UI der Scan-Phase geöffnet und das zuvor erstellte Objekt an der vorgesehenen Stelle platziert. Demnach können weitere Objekte erzeugt werden. Die folgende Abbildung demonstriert die Darstellung solcher Objekte. Dabei wurden willkürlich „Assets“ gewählt, die bei tatsächlicher Anwendung in realem Szenario angepasst werden können. Die aktuell verwendeten Objekte fungieren lediglich als Veranschaulichung der Aktionen und der Repräsentation des Status eines Objekts. Demzufolge ist der Abbildung (4.10) die Erstellung und Darstellung solcher Objekte zu entnehmen.

Hat der Nutzer alle gewünschten Objekte platziert, kann dieser über den Button, rechts unten, die Funktion der Scan-Phase verlassen und die Visualisierungs-Phase starten oder die Anwendung schließen. Da alle eingegebenen Informationen persistiert sind, gehen diese nicht verloren und können jederzeit über die Visualisierungs-Phase aufgerufen werden. Ist dem Nutzer allerdings ein Fehler bei der Erstellung der Objekte unterlaufen, hat er nach aktuellem Stand die Möglichkeit, alle vorhandenen Objekte und Daten zu löschen und die Scan-Phase von Anfang an erneut zu starten.

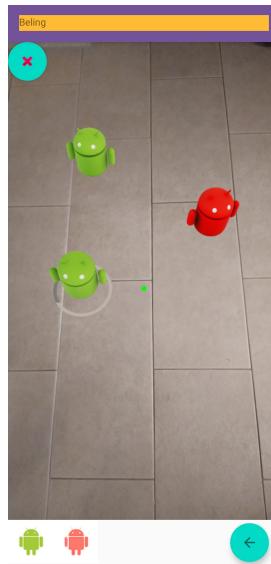


Abbildung 4.10: Funktion der Scan-Phase Teil 2

Nachdem die Scan-Phase nun vollends beschrieben wurde, geht es in folgendem Kapitel um die Erläuterung des dritten und letzten Use Cases. Dabei werden sowohl die Frontend als auch die Backend Aspekte aufgezeigt. Ebenso wird die Funktionsweise der Objekt-Repräsentation anhand eines kleinen Beispiels, ähnlich wie bereits in der Scan-Phase erläutert, veranschaulicht.

#### 4.1.3 Visualisierungs-Phase

Nach vollständiger Beendigung der Scan-Phase ging es in der Entwicklung weiter mit der Visualisierungs-Phase. Die zuvor gespeicherten Objekte und deren Informationen werden dabei abgerufen und können jederzeit erneut visualisiert und im Raum platziert werden. Die folgende Darlegung der Frontend-Entwicklung der Visualisierungs-Phase gibt Aufschluss über die Benutzeroberfläche dieser Phase und deren Aufbau. Die Ausführung des Frontends und deren bildliche Darstellung dient dazu, das anschließende Backend nachzuvollziehen zu können. Abschließend werden die Aspekte des Frontends aufgegriffen und anhand der Backend-Implementierung mit deren Funktionen verknüpft.

##### FrontEnd

Bei der Visualisierungs-Phase gibt es prinzipiell eine Benutzeroberfläche, die diesen Use Case ausmacht. Unter eigentlicher Anwendung dieser Phase und deren GUI befinden sich darüber hinaus weitere Benutzeroberflächen, die bereits in der Scan-Phase schon erläutert wurden. Darunter zum Beispiel die in Abbildung (4.3) zu entnehmende Markererkennungs-UI, die ebenso Bestandteil der Visualisierungs-Phase ist.

Die primäre GUI ist ähnlich zu der Benutzeroberfläche der Scan-Phase aufgebaut. Ein Fragment das sich über den ganzen Bildschirm erstreckt, repräsentiert das Livebild der Kamera. Dadurch können die bereits in der Scan-Phase erstellten Objekte erneut angezeigt werden. Des Weiteren befinden sich auf dieser Oberfläche zwei Buttons, zum einen zur Navigation, um auf das Startmenü zu gelangen, und zum anderen, um die Objekte von der Datenbank abzugreifen, rendern und auf dem Bildschirm anzeigen zu lassen. Diese befinden sich jeweils in der linken und rechten unteren Ecke des Bildschirms, wie der Abbildung 4.11 zu entnehmen ist.

Da diese UI nur als Ansicht der zu visualisierenden Objekte gedacht ist, stehen dem Nutzer für die Interaktionen nur die Objekte, die als Overlay über das Kamerabild gelegt werden, zur Verfügung. Die Nutzeraktionen beschränken sich lediglich auf den Gebrauch der zuvor genannten Buttons und die dynamisch erzeugten Schaltflächen der Objekte, um über diese zusätzliche Informationen zu erhalten.

Nun erfolgt die Beschreibung der Backend-Implementierung und Einblicke in die Programmierung der Visualisierungs-Phase.



Abbildung 4.11: Visualisierungs-Phase der Applikation

## BackEnd

Nachdem das Problem der variablen Startposition in der Implementierung der Scan-Phase umgangen wurde, konnte die Visualisierungs-Phase entwickelt werden.

Wählt der Anwender auf der Startmenü-UI (4.2) den Button „Virtual-Phase“, wird eine Anmerkung getätigigt, indem der Nutzer darüber in Kenntnis gesetzt wird, dass er zu Beginn die Scan-Phase durchgeführt haben muss, bevor die Visualisierungs-Phase gestartet werden kann. Diesen Vermerk kann der Benutzer ignorieren, wenn dieser weiß, dass ein Scan durchgeführt wurde, oder er akzeptiert den Hinweis und wird auf die dementsprechende Funktion weitergeleitet. Der Nutzer wird dadurch davor bewahrt, eine Funktion starten zu wollen, die ohne Inhalt keine Änderungen aufzeigt.

Danach wird der Nutzer aufgefordert, den initialen Marker (4.7) einzuscannen, um die ursprüngliche Position, bzw. die Markierung festzuhalten. Damit kann der weitere Programmablauf folgen, indem die GUI der Visualisierungs-Phase (4.11) angezeigt und eine neue Session zur Objektplatzierung gestartet wird.

Über eine ViewModel-Komponente werden die Objekte aus der Datenbank abgefragt und in einer Liste gespeichert, um diese zu einem späteren Zeitpunkt zur Platzierung der Objekte verwenden zu können.

Ist der Marker erkannt worden, wird mithilfe des Buttons, der sich im rechten unteren Eck der Benutzeroberfläche (4.11) befindet, die Methode zur Datenabfrage, Erzeugung und Platzierung der Objekte gestartet. Dabei wird über den Button-Klick eine Funktion aufgerufen, die die Liste der Objekte anspricht und die jeweils darin enthaltenen Objekte nacheinander generiert. Nach Betätigen des Buttons ist dieser auf der UI nicht mehr aufzufinden, um einer weiteren Platzierung aller Objekte vorzubeugen. Die von dem ViewModel gegebene Liste wird, abhängig von deren Länge, iterativ abgearbeitet und die Objekte nacheinander angelegt. Dies erfolgt über eine „for“-Schleife, welche abhängig von dem Ursprungspunkt, dem Marker „objectMarkerCenter“, die Position des aktuellen Objekts „obj“ an der Stelle „i“ in der Liste neu errechnet und darstellt.

---

```

1 ...
2 for (int i = 1; i < listOfObj.size(); i++) {
3     obj = listOfObj.get(i);
4     if (!obj.getName().equals("origin")) {
5         calculateObjectPosition(obj, i, objectMarkerCenter);
6     }
7 }
```

---

Code-Beispiel 4.7: Abfolge der Objekte in der Liste

Mit der gegebenen „calculateObjectPosition“ wird dann die Distanz vom Ursprungsmarker zu dem Objekt addiert, um die Positionen und Distanzen der einzelnen Objekte der Scan-Phase exakt wiederzugeben. Ist die Addition zu einem Objekt durchgeführt, repräsentiert diese Berechnung unabhängig von der Startposition des Anwenders die neue Position, an der das Objekt platziert werden muss. Somit ist der Nutzer bei Verwendung der Applikation in der Lage, aus einem vorab beliebigen Punkt zu starten und alle Objekte exakt auf die referenzierte, ursprüngliche Position

zu setzen, die in der Scan-Phase festgelegt wurde. Nachdem die Position des aktuellen Objekts berechnet wurde, wird diese als neue „Pose“ initialisiert und übergeben, um das zu erstellende Objekt zu rendern und auf der Oberfläche anzuzeigen. Dabei findet eine Abfrage des Status statt, um daraufhin entscheiden zu können, in welcher Farbe das Objekt zu präsentieren ist.

---

```

1 public void calculateObjectPosition(Object object, int i, Object objOrigin){
2     ...
3     qx = object.getQx() + objOrigin.getQx();
4     qy = object.getQy() + objOrigin.getQy();
5     qw = object.getQw() + objOrigin.getQw();
6     qz = object.getQz() + objOrigin.getQz();
7
8     tx = object.getTx() + objOrigin.getTx();
9     ty = object.getTy() + objOrigin.getTy();
10    tz = object.getTz() + objOrigin.getTz();
11
12    float[] rotation = {qx, qy, qz, qw};
13    float[] translation = {tx, ty, tz};
14
15    Pose pose = new Pose(translation, rotation);
16
17    if (objectViewModel.getAllObjectsName().getValue().get(i).getState() == 1) {
18        addObject(Uri.parse("object.sfb"), pose, i);
19    }
20
21    if (objectViewModel.getAllObjectsName().getValue().get(i).getState() == -1) {
22        addObject(Uri.parse("object_red.sfb"), pose, i);
23    }
24}

```

---

Code-Beispiel 4.8: Berechnung der Markerplatzierung

Sind alle Listenelemente abgearbeitet, ist die Erstellung und Renderung der Objekte fertig. Danach werden dem Nutzer alle vorhandenen Objekte angezeigt und visualisiert. Mit diesen Erkenntnissen kann der Nutzer durch den Raum laufen und alle Objekte überwachen. Wird ein Objekt anvisiert, kann über eine Berührung des Objekts auf dem Display alle vorhandenen Informationen eingesehen werden. Somit sind die Informationen des einzelnen Objekts schnell sichtbar. Die Applikation kann beliebig oft geschlossen und neu gestartet werden. Der Nutzer kann sich zu jeder Zeit die Objekte visualisieren lassen, vorausgesetzt, der Ursprungsmarker wird zu Anfang eingescannt. Daraufhin können alle Objekte präsentiert und die dazugehörigen Informationen angezeigt werden.

Um die soeben beschriebene Backend-Implementierung der Visualisierungs-Phase zu demonstrieren, wird diese anhand eines kleinen Szenarios veranschaulicht.

Startet der Nutzer die Visualisierungs-Phase, muss er den initialen Marker scannen, um fortfahren zu können. Darauffolgend öffnet sich die GUI dieser Phase und mit einem Klick auf den Button rechts unten auf der Oberfläche synchronisiert er die Funktion und alle Objekte, die in der Scan-Phase

gesetzt wurden, werden erneut generiert. Daraufhin kann der Nutzer die Objekte an den Stellen einsehen, an denen sie sich ursprünglich befanden. So werden diese samt Status angezeigt und der Nutzer hat den gesamten Überblick über alle Objekte und kann diese beobachten. Der Schritt ist der Abbildung (4.12) zu entnehmen.

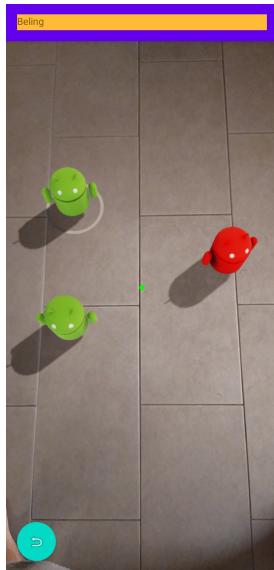
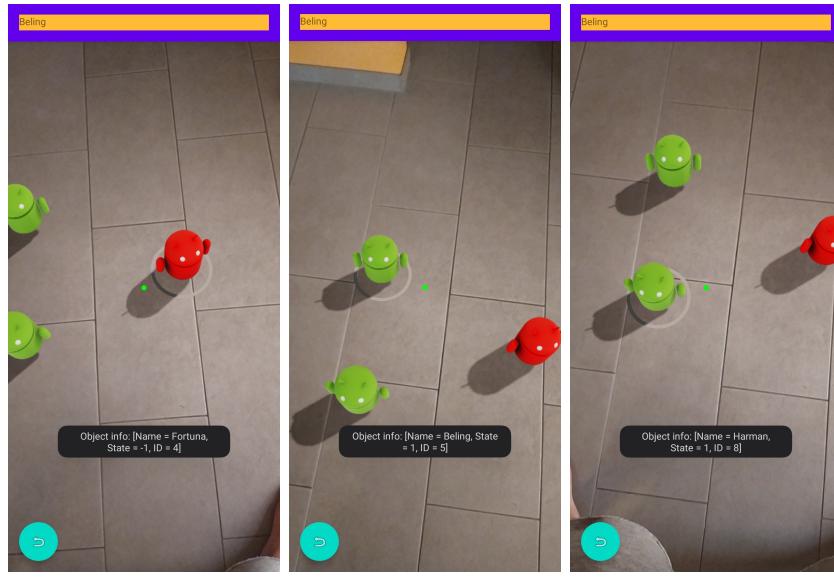


Abbildung 4.12: Funktion Visualisierungs-Phase Teil 1

Eine weiter Funktion dieser Phase ist die Verwaltung der Objekte und deren Informationen. Wird ein Objekt anvisiert, werden mit einem Klick auf das Objekt die Informationen über ein PopUp angezeigt. Darunter ist der Name des Objekts, der Status und die ID, die zu Anfang automatisch generiert und vergeben wurde. So kann zu jedem einzelnen Objekt dessen Informationen zur Verfügung gestellt werden.



(a) Anzeige der Objektin-  
formationen 1      (b) Anzeige der Objektin-  
formationen 2      (c) Anzeige der Objektin-  
formationen 3

Abbildung 4.13: Funktion Visualisierungs-Phase Teil 2

Nachdem die Aspekte der Front- und Backend-Entwicklung der drei Use Cases geschildert und Einblicke in die Entwicklung und die damit verbundenen Fragestellungen und Schwierigkeiten gewährt wurden, geht es nun mit der Evaluierung des Systems, dem Fazit und dem Ausblick weiter.

## Kapitel 5

# Evaluation

Nach Beendigung der Entwicklungsphase wurde eine Evaluation des Softwareprodukts durchgeführt. Diese dient zur Kategorisierung und Einstufung der Applikation in die Gebrauchstauglichkeit. Dabei wurde auch ein Vergleich zu schon bestehenden, bzw. ähnlichen Produkten aufgestellt. Darunter befinden sich ebenso Aspekte der Benutzerfreundlichkeit, der Weiterentwicklungsmöglichkeiten und eine Überprüfung auf gewünschte, bzw. vorab definierte Anforderungen und Ziele.

Im Bereich der Industrie gibt es derzeit einige Anwendungen, die auf Augmented Reality aufbauen, darunter sind allerdings wenige Apps, die sich mit vorliegenden Prozessen auseinandersetzen. Es gibt vereinzelt Anwendungen, die mit Interaktionen in Prozessabläufen arbeiten. Diese basieren allerdings weniger auf einer Smartphone-Applikation. Es wird vermehrt auf VR- und AR-Brillen gesetzt. Diese sind meist für die Begleitung der Prozesse zuständig und projizieren zusätzliche Informationen oder Visualisierungen, um dem Nutzer erweiterte Einblicke zu verschaffen. Es gibt auch Anwendungen die visualisierte Arbeitsprozesse vorgeben, und das Potenzial, den Effekt und die Effizienz des „learning by doing“-Ansatzes ausschöpfen oder auch Vorstellungen und Planungen zum besseren Vorstellen visualisieren. Beispiele dazu sind zum einen der „augmented presenter“ [TEPCON 2020] und der „augmented instructor“ [TEPCON 2020] der tepcon GmbH. Der „augmented presenter“ bietet die Möglichkeit, eine Inneneinrichtung einer Produktionshalle zu visualisieren und damit Maschinen an verschiedenen Stellen zu platziert. Anhand dieser Projektion wird überprüft, ob sich die vorgesehene Position für die Platzierung der zu planende Maschine eignet. Diese Anwendung dient zur optimierten Fabrikplanung, bevor die Produkte bestellt oder in Betrieb genommen werden. Auch kann diese Anwendung für Vertrieb und Marketing, Schulungen und Messen zur Demonstration der Möglichkeiten der AR verwendet werden. Der „augmented instructor“ basiert auf dem Gebrauch der AR-Brille und visualisiert und projiziert Arbeitsschritte zur Veranschaulichung.

Das entwickelte, prototypische Assistenzsystem dient zur übersichtlichen Verdeutlichung der vorhandenen Maschinen in einer Produktionshalle. Dadurch kann die Umgebung eingescannt und Maschinen als Objekte virtuell dargestellt werden. Mit den nötigen Daten ermöglicht dieses System die schnelle Abrufung von Informationen zu einzelnen Maschinen. Auch die Option der Echtzeit-Statusüberprüfung kann in weiterer Entwicklung realisiert werden.

So kann das Assistenzsystem schnell Informationen zur Verfügung stellen, die immer verfügbar sind. Durch die einfache Erweiterbarkeit, die durch die Architektur gewährleistet ist, können weitere nützliche Funktionen hinzugefügt werden und bestätigt dessen stetig erweiterbare Alltagstauglichkeit. Auch im Hinblick auf die Benutzerfreundlichkeit ist eine einfache UI gegeben, die überschaubar und schnell zu verstehen ist. Die Norm ISO 9241-10 schreibt eine gute Steuerbarkeit, Selbstbeschreibungsfähigkeit, Individualisierbarkeit und Aufgabenangemessenheit vor. In der Entwicklung der Anwendung wurde versucht, sich daran zu orientieren. In Usability-Tests wurden die Anforderungen der Norm in kleinem Rahmen, aber noch nicht abschließend, überprüft. Auf diese Tests wird im Folgenden aus Relevanzgründen nicht näher eingegangen. Im Rahmen der Testungen wurden Verbesserungsvorschläge geäußert, die sich in diesem Bezug nur auf Erweiterungen bezogen haben. Ebenso wurde die Genauigkeit bemängelt, da diese manchmal fehlerhaft Objekte erstellt und wiedergibt. Die Ursache ist der Handhabung durch den Nutzer zuzuschreiben, der je nach Lage und Ausrichtung des Smart-Device die Berechnung beeinflusst.

In der Gesamtheit wurde das System als überaus hilfreich und Entwicklungsfähig eingestuft. In dieser Arbeit stehen lediglich die Konzeption, Grundlagenschaffung und prototypische Entwicklung im Fokus. Die Grundfunktion wurde implementiert und zur Verfügung gestellt.

Nun folgt abschließend, um die Dokumentation abzurunden, das allgemeine Fazit des Projekts und der Ausblick, der die Zukunftsaussichten des Projekts darlegt.

# Kapitel 6

## Fazit

Ziel dieser Arbeit war die Konzeption und prototypische Umsetzung eines Assistenzsystems zur Unterstützung industrieller Prozesse. Dabei wurde der Fokus auf die übersichtliche und virtuelle Darstellung von Maschinen und Geräten gelegt. Zu der Übersicht kam auch dazu, dass zu einzelnen Objekten schnell Informationen eingesehen werden können, um diese bei Bedarf zur Hand zu haben. Dafür wurden in den ersten Schritten Anforderungen zur Umsetzung des Systems festgelegt. Unter Berücksichtigung der zu gewährleistenden Modularisierung musste ein geeignetes Entwurfsmuster gefunden werden, auf dem die Architektur aufbaut, um künftige Weiterentwicklungen an dem Projekt zu ermöglichen und diese keine großen Herausforderungen darstellen. Durch die Android Architecture Components wurde sich an dem MVVM-Muster angelehnt, welches unter anderem die Anforderungen der Modularisierung erfüllt. Bevor die konkrete Umsetzung stattfinden konnte, wurden anhand der Anforderungen und geplanten Funktionen Use Cases für die Implementierung definiert. Nachdem diese modelliert waren, konnte das eigentliche System aufgestellt und entwickelt werden.

Die Positions berechnung des Smart-Devices basiert auf Kalkulationen der hardwareinternen Sensoren und unterliegt somit einer sehr sensiblen Basis. Je nach Ausrichtung und Lage des Geräts kann sich sowohl dessen Orientierung, als auch dessen Blickrichtung unterscheiden und so das eigentliche Ergebnis verfälschen, obwohl sich das Gerät an der physikalisch identischen Position befindet. Dadurch ist die Benutzung der Applikation stark von den Aktionen und der garantiierten Handhabung des Nutzers abhängig.

Das verwendete Framework Google ARCore erweist einen effektiven Nutzen und ist eine gute Grundlage für die Implementierung einer Augmented Reality Applikation. Das Framework stellt eine fundamentierte API zur Verfügung, die eine gute Basis für die darauf aufbauenden Entwicklungen schafft. Bei der praktischen Anwendung des Frameworks gibt es jedoch vereinzelt Probleme mit der Anzeige der virtuellen Objekte, da aus den Positions berechnungen phasenweise Übereinstimmungen mit diesen nicht erkannt und dadurch die Objekte nicht angezeigt werden. Daher ist es in diesem Fall notwendig, an den Ausgangspunkt zurückzukehren, damit die Anzeige der Objekte aktualisiert werden kann und so die nicht angezeigten Visualisierungen wieder auftauchen.

Werden Objekte über die Scan-Phase an eine bestimmte Position gesetzt, wird die daraus resultierende Differenz zum Ursprungsmarker gespeichert. Bei der Ausführung der Visualisierungs-Phase werden die gespeicherten Objekte und deren Positions differenz abhängig von der Ursprungsmarkierung wieder virtuell im Raum platziert. Dabei ist die Ausgangssituation der Markierungsverfolgung entscheidend. Dies bedeutet, dass das Smart-Device vom Nutzer richtig angewendet werden muss und es beispielsweise nicht über Kopf gehalten werden darf. Dadurch würde die Berechnung und Darstellung der Objekte nicht exakt mit der vorgesehenen Position übereinstimmen. Dies hat eine ungenauer Repräsentation der Objekte im Raum zufolge.

Es ist durch diese Sensibilität der Anwendung vorauszusetzen, dass der Nutzer das System verantwortungsvoll und gewissenhaft verwendet. Da dies von dem Assistenzsystem nicht überprüft und auch nicht korrigiert werden kann, muss die fachgerechte Handhabung vorausgesetzt werden.

Im Großen und Ganzen sind die Benutzeroberflächen überschaubar und bieten eine gute Grundlage für die Anwendung der Augmented Reality Applikation.

In der Gesamtheit bietet das Ergebnis dieser Arbeit einen klaren und leicht verständlichen Aufbau des Systems, sowie eine Grundlage für Weiterentwicklungen. Das Konzept wurde eigenständig erarbeitet und wie geplant umgesetzt. Somit ist der Grundbaustein für weitere, darauf aufbauende Funktionen gelegt.

# Kapitel 7

## Ausblick

Der in dieser Arbeit entstandene Prototyp ist ein eigenständiges System zur visualisierten Übersicht von Maschinen und Geräten, beispielsweise in Produktionshallen. Bevor die Anwendung allerdings produktiv eingesetzt werden kann, sind noch einige Optimierungen vorzunehmen. Darunter die Verwendung von weiteren Objekten zur visuellen Darstellung der Maschinen. Eventuell können auch detailgetreue Abbildungen in einem bestimmten Anwendungsumfeld oder die Verwendung universeller ortsunabhängiger Objekte eingesetzt werden. Des Weiteren muss die Überarbeitung der Informationsanzeige der Objekte, die aktuell über ein vereinfachtes Informationsfenster eingeblendet werden, erfolgen. Eine Augmented Reality basierte Anzeige der Informationen wäre dabei denkbar. In dieser Arbeit wurden lediglich die Grundsteine für ein Projekt gelegt, welches in Zukunft stetig weiterentwickelt werden kann und viel Potential für weitere Funktionen, auf Basis der AR-Anwendung, bietet. Die Ergebnisse des Prototypen sind für erste Tests geeignet, allerdings für den derzeitigen Einsatz noch nicht vorgesehen. Da zum jetzigen Zeitpunkt noch kein Zusatznutzen für Anwender daraus resultiert und die maschinenbezogenen Daten im Hinblick auf die statusabhängigen Informationen keine Echtzeit-Auskünfte bietet, ist das Assistenzsystem für Firmen noch uninteressant und unrentabel. Der Prototyp muss der Prototyp weiterentwickelt und verbessert werden.

Um Informationen über einzelne Maschinen in Echtzeit aufrufen und visualisieren zu können, wäre ein weiterer Entwicklungsschritt die Benutzung von Echtzeitdaten der Maschinen. Dafür wäre eine Datenaufbereitung und eine globale Verfügbarkeit der Informationen notwendig. Diese Idee könnte über eine Internet of Things, dt. Internet der Dinge-Lösung, umgesetzt werden. Dabei könnten die Maschinendaten, die über eine Cloud ausgelagert sind, in Echtzeit abgegriffen und stets den aktuellen Zustand und Status der einzelnen Geräte abgerufen werden. Auch könnten die Daten direkt von den jeweiligen Geräten verwendet, aufbereitet und in der Anwendung genutzt werden. Dadurch wäre die Möglichkeit gegeben, genauere Informationen zu den Objekten zu erhalten.

Eine zusätzliche Erweiterung der Anwendung wäre das automatische Lokalisieren von Anomalien sämtlicher Maschinen. Dabei würden auftretende Fehler registriert werden und über eine Benachrichtigung den Nutzer darüber in Kenntnis setzen. Somit könnte bei Ausfällen der Maschinen schnell

reagiert und agiert werden, um diese auftretenden Fehler schnellstmöglich zu beheben. Basierend auf dieser Erweiterung wäre es ebenso möglich, bei großen Umgebungen, bzw. Produktionshallen eine räumliche Navigation einzubauen. Der Mechaniker wird auf direktem Wege zu der wartungsbedürftigen Maschine geleitet und muss sich nicht zusätzlich in der Räumlichkeit orientieren.

Im Moment steht nur eine Laufzeitumgebung für ein natives Android System zur Verfügung. Ein weiterer Schritt zur Systemoptimierung wäre die Einbindung von AR-Brillen, da der Nutzer im Vergleich zum Smartphone in seinen Bewegungen nicht eingeschränkt wäre und sich mit den Händen frei bewegen könnte.

Aufbauend darauf könnten auch Schritte der Reparaturarbeiten optimiert werden, indem über die AR-Brille Anweisungen und Anleitungen zur Wartung der Maschine über die AR-Brille zusätzlich bereitgestellt werden.

Dies sind einige Ansätze für Einsatzmöglichkeiten des Projekts und wie sie in Zukunft erweitert werden könnten, um ein umfängliches Assistenzsystem zu erschaffen und zu einer echten Unterstützung industrieller Prozesse zu werden.

# Abbildungsverzeichnis

2.1	Mobile-AR in Snapchat . . . . .	17
2.2	Test der HoloLens 2 . . . . .	18
2.3	Datenbrillen (HMD) . . . . .	18
2.4	Marker-basierte Augmented Reality Positionsbestimmung . . . . .	19
2.5	Time-of-Flight Kamera und deren Repräsentation [STRAND 2020a] . . . . .	24
2.6	Graphische Darstellung des SLAM Verfahrens [GRISSETTI u. a. 2010] . . . . .	25
2.7	Skizze zur Bewegungsverfolgung des Smartphones [ARCORE 2020] . . . . .	29
2.8	Grundprinzip einer IMU [BÖTTCHER 2020] . . . . .	30
2.9	Skizze zum Umgebungsverständnis des Smartphones [ARCORE 2020] . . . . .	30
2.10	Skizze zu den Lichtverhältnissen des Smartphones [ARCORE 2020] . . . . .	31
2.11	Komponenten von Android Jetpack [SELLS, POIESZ und NG 2018] . . . . .	32
2.12	Room Architektur Diagramm [DEVELOPERS 2017b] . . . . .	33
2.13	ViewModel Struktur Diagramm [CODELABS 2020] . . . . .	34
2.14	MVC Architektur Diagramm [ <i>ModelView Controller</i> 2020] . . . . .	37
2.15	MVVM Architektur Diagramm [JAECKLE, GOLL und DAUSMANN 2015] . . . . .	39
2.16	MVVM Informationsfluss [RICARDO 2019] . . . . .	40
2.17	Android Grundkomponente [Architecture Components 2020] . . . . .	40
2.18	ViewModel Komponente [Architecture Components 2020] . . . . .	40
2.19	Repository Komponente [Architecture Components 2020] . . . . .	41
2.20	Model / Room Komponente [Architecture Components 2020] . . . . .	41
2.21	Android Architecture Components Diagram[Architecture Components 2020] . . . . .	42
3.1	Konzeptionelle Software-Architektur . . . . .	50
3.2	Anwendungsfall-Diagramm . . . . .	52
3.3	Entity Relationship Datenmodell . . . . .	57
4.1	Start der Applikation . . . . .	60
4.2	Startmenü der Applikation . . . . .	62
4.3	Markererkennung der Applikation zum Start der Scan-Phase . . . . .	64
4.4	Scan-Phase der Applikation . . . . .	65
4.5	Erstellen eines neuen Objekts . . . . .	66
4.6	Aufbau der Positionsberechnung von Objekten . . . . .	71
4.7	Marker zur Erkennung der Ausgangsposition . . . . .	73

4.8 Positions berechnung zum Ursprungsmarker . . . . .	74
4.9 Funktion der Scan-Phase Teil 1 . . . . .	75
4.10 Funktion der Scan-Phase Teil 2 . . . . .	76
4.11 Visualisierungs-Phase der Applikation . . . . .	78
4.12 Funktion Visualisierungs-Phase Teil 1 . . . . .	81
4.13 Funktion Visualisierungs-Phase Teil 2 . . . . .	82

# **Tabellenverzeichnis**

2.1 Merkmale der Computergraphik gegenüber der VR [DÖRNER 2019] . . . . .	14
---	----

# Liste der Code-Beispiele

4.1	Entity Code zur Initialisierung der Objekte . . . . .	67
4.2	SQL-Query zur Abfrage der Objekt-Namen . . . . .	69
4.3	Erzeugung des Datenbank-Layers „Room“ . . . . .	69
4.4	Initialisierung des Fragments . . . . .	70
4.5	ModelRenderable Builder . . . . .	71
4.6	Berechnung der Distanz zwischen Marker und Ursprungspunkt . . . . .	73
4.7	Abfolge der Objekte in der Liste . . . . .	79
4.8	Berechnung der Markerplatzierung . . . . .	80

# Abkürzungsverzeichnis

<b>API</b>	Application Programming Interface . . . . .	10
<b>AR</b>	Augmented Reality . . . . .	15
<b>DBMS</b>	Datenbank-Management-System . . . . .	44
<b>EKF</b>	Erweiterter Kalman Filter	
<b>ERM</b>	Entity-Relationship-Modell . . . . .	44
<b>IOSB</b>	Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung IOSB	
<b>GPS</b>	Global Positioning System	
<b>GUI</b>	Graphical User Interface, dt. Benutzeroberfläche	
<b>HMD</b>	Head-Mounted Display . . . . .	12
<b>ICP</b>	Iterativ Closest Point . . . . .	25
<b>IDE</b>	Integrated Development Environment . . . . .	59
<b>ID</b>	Identifikation	
<b>IMU</b>	Inertial Measurement Unit . . . . .	29
<b>INS</b>	Inertial Navigation System . . . . .	20
<b>IoT</b>	Internet of Things, dt. Internet der Dinge . . . . .	4
<b>KIT</b>	Forschungszentrum Karlsruhe . . . . .	6
<b>MEMS</b>	Micro-Electro-Mechanical Systems . . . . .	29
<b>MR</b>	Mixed Reality . . . . .	15
<b>MVC</b>	Model View Controller . . . . .	36
<b>MVVM</b>	Model View ViewModel	
<b>SDK</b>	Software Development Kit . . . . .	27
<b>SLAM</b>	Simultaneous Localization And Mapping . . . . .	7
<b>SQL</b>	Structured Query Language	
<b>TOF</b>	Time of flight . . . . .	24
<b>UX</b>	User-Experience, dt. Nutzererfahrung und Nutzererlebnis . . . . .	61

<b>UI</b>	User Interface . . . . .	34
<b>VR</b>	Virtual Reality . . . . .	15
<b>WMR</b>	Windows Mixed Reality . . . . .	16

# Literatur

- ABRAHAM, Magid und Marco ANNUNZIATA [März 2017]. *Augmented Reality Is Already Improving Worker Performance*. <https://hbr.org/2017/03/augmented-reality-is-already-improving-worker-performance> [siehe S. 13].
- Architecture Components [2020]. <https://developer.android.com/jetpack/guide> [siehe S. 40–42].
- ARCORE, Google [2020]. *Fundamental Concept*. <https://developers.google.com/ar/discover/concepts> [siehe S. 29–31].
- AREA [2015]. *Augmented Reality Can Increase Productivity*. <https://thearea.org/augmented-reality-can-increase-productivity/> [siehe S. 13].
- AZUMA, Ronald T. [Aug. 1997]. *A Survey of Augmented Reality*. Massachusetts Institute of Technology [siehe S. 11].
- B12, Studio [Juni 2020]. *VR und AR- Nur ein Hype oder echter Mehrwert?* [https://studio-b12.de/content/projects/vrar/virtual-reality/pdf/2-Nur\\_ein\\_Hype\\_oder\\_echter\\_Mehrwehrt.pdf](https://studio-b12.de/content/projects/vrar/virtual-reality/pdf/2-Nur_ein_Hype_oder_echter_Mehrwehrt.pdf) [siehe S. 2].
- BAUM, L. Frank [Feb. 1996]. *The Master Key - An Electrical Fairy Tale*. Hrsg. von Dennis AMUNDSON. Project Gutenberg-tm [siehe S. 12].
- BEGEROW [2020]. *Normalisierung von Datenbanken*. <https://www.datenbanken-verstehen.de/datenbankmodellierung/normalisierung/> [siehe S. 44].
- BERGER, Daniel [2019]. *Live View: Google Maps schaltet AR-Navigation für alle frei*. <https://www.heise.de/newsticker/meldung/Live-View-Google-Maps-schaltet-AR-Navigation-fuer-alle-frei-4491407.html> [siehe S. 19].
- BITFORGE [Okt. 2019]. *Augmented Reality offiziell kein Hype mehr*. <https://bitforge.ch/augmented-reality/augmented-reality-offiziell-kein-hype-mehr/> [siehe S. 3].
- BOUVERET, Christopher und Sebastian HUMAN [Nov. 2019]. *Augmented Reality in der Industrie: Herausforderungen, Potenziale, Chancen*. <https://www.industry-of-things.de/augmented-reality-in-der-industrie-herausforderungen-potenziale-chancen-a-882695/> [siehe S. 4].
- BREHM, Frank [März 2017]. *VR-Angriff auf alle Sinne - Wie schmeckt die virtuelle Realität?* <https://www.mice-club.com/magazin/artikel/vr-angriff-auf-alle-sinne> [siehe S. 15].

- BÖTTCHER, Prof. Dr.-Ing Jörg [2020]. *Inertiale Messeinheit (IMU)*. <https://messtechnik-und-sensorik.org/8-sensoren-fuer-drehzahl-geschwindigkeit-beschleunigung-und-position-im-raum/> [siehe S. 30].
- CODELABS, Google [2020]. *Android Room with a View - Java*. <https://codelabs.developers.google.com/codelabs/android-room-with-a-view/> [siehe S. 34].
- DEVELOPER, Moody [2019]. *What is boilerplate code? Why and when to use the boilerplate code?* <https://fullstackworld.com/post/what-is-boilerplate-code-why-when-to-use-the-boilerplate-code> [siehe S. 31].
- DEVELOPERS, Google [2020]. *LiveData Overview*. <https://developer.android.com/topic/libraries/architecture/livedata> [siehe S. 32].
- [2017a]. *Room Persistence Library*. <https://developer.android.com/topic/libraries/architecture/room> [siehe S. 32].
  - [2017b]. *Save data in a local database using Room*. <https://developer.android.com/training/data-storage/room> [siehe S. 33].
- DIN-461 [1973]. *Graphische Darstellung in Koordinatensystemen*. ISO - Internationale Organisation für Normen [siehe S. 29].
- DÖRNER, Ralf [Sep. 2019]. *Virtual und Augmented Reality (VR/AR) - Grundlagen und Methoden der Virtuellen und Augmentierten Realität*. Springer Vieweg, Berlin, Heidelberg [siehe S. 12, 14].
- EADICICCO, Lisa [Jan. 2020]. <https://www.businessinsider.de/international/apple-q1-earnings-call-tim-cook-augmented-reality-ar-2020-1/?r=US&IR=T> [siehe S. 4].
- EDER, Norbert [2016]. *MVVM – Model-View-ViewModel: Die Serie*. <https://www.norberteder.com/model-view-viewmodel-die-serie/> [siehe S. 39].
- ELGAN, Mike [2016]. *How Google's Project Tango will change your life*. <https://www.computerworld.com/article/3018733/how-googles-project-tango-will-change-your-life.html> [siehe S. 27].
- FISCHBACH, Pascal [Aug. 2016]. *First Mover: Pokémon Go präsentiert der Welt "Augmented Reality"*. <https://conrenfonds.com/2016/08/24/first-mover-pokemon-go-praesentiert-der-welt-augmented-reality-die-abloesung-des-smartphones-oder-nur-spielzeug> [siehe S. 16].
- FREEDEN, Willi und Reiner RUMMEL [2016]. *Handbuch der Geodäsie*. Springer Spektrum, Berlin, Heidelberg [siehe S. 22, 23].
- GAEVERT, Thomas [Mai 1856]. *Bibliographie Lyman Frank Baum*. <http://www.isfdb.org/cgi-bin/ea.cgi?162> [siehe S. 12].
- GEOFFREY [2019]. *Koppelnavigation*. <https://de.wikipedia.org/wiki/Koppelnavigation> [siehe S. 23].
- GRISSETTI, Giorgio u. a. [2010]. *A Tutorial on Graph-Based SLAM*. Department of Computer Science, University of Freiburg [siehe S. 25].

- HASSELBRING, Wilhelm [2006]. *Software-Architektur*. Springer-Verlag 2006 [siehe S. 36].
- HIPP, D. Richard [2018]. *About SQLite*. <https://www.sqlite.org/about.html> [siehe S. 35].
- JAECKLE, Lukas, Joachim GOLL und Manfred DAUSMANN [2015]. *Das Architekturmuster Model-View-ViewModel*. [https://www.it-designers-gruppe.de/fileadmin/Inhalte/Studentenportal/Das\\_Architekturmuster\\_MVVM\\_\\_Text\\_\\_\\_1\\_.pdf](https://www.it-designers-gruppe.de/fileadmin/Inhalte/Studentenportal/Das_Architekturmuster_MVVM__Text___1_.pdf) [siehe S. 39].
- JECKLE, M. u. a. [2005]. *UML 2 glasklar*. Hanser Fachbuchverlag [siehe S. 36].
- KATO, Hirokazu und Mark BILLINGHURST [Okt. 1999]. *Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System*. hitl.washington.edu [siehe S. 13].
- KRAUSS, Melanie [Feb. 2019]. *Europa baut Führung bei Industrie 4.0 weiter aus*. <https://www.maschinenmarkt.vogel.de/europa-baut-fuehrung-bei-industrie-40-weiter-aus-a-801681/> [siehe S. 4].
- LANHAM, Micheal [2018]. *Learn ARCore - Fundamentals of Google ARCore*. Packt Publishing Ltd. [siehe S. 28].
- LESWING, Kif [Okt. 2016]. *Apple CEO Tim Cook thinks augmented reality will be as important as 'eating three meals a day'*. <https://www.businessinsider.com/apple-ceo-tim-cook-explains-augmented-reality-2016-10?r=DE&IR=T> [siehe S. 4].
- ModelView Controller* [2020]. <https://bmu-verlag.de/model-view-controller-mvc-softwareentwicklungsma> [siehe S. 37].
- MUNDT, Elisa [Jan. 2020]. *Marktstudie zu Augmented und Virtual Reality - Einblicke in die Marktreife immersiver Technologien*. <https://www.industry-of-things.de/vr-und-ar-werden-realitaet-a-898199/> [siehe S. 4, 21].
- MVVM-Pattern* [2010]. <https://entwickler.de/online/tipps-und-tricks-zum-mvvm-pattern-153374.html> [siehe S. 38].
- MÜHLROTH, Adrian [Nov. 2018]. *Was ist der Unterschied zwischen VR, AR und MR?* <https://www.techbook.de/entertainment/virtual-reality/unterschied-vr-ar-mr> [siehe S. 14, 15].
- NATESAN, N. [2019]. *How to implement Design Pattern – Separation of concerns*. <https://www.castsoftware.com/blog/how-to-implement-design-pattern-separation-of-concerns> [siehe S. 31].
- NIEMANN, Alexander [2020]. *IoC (inversion of control)*. <https://www.itwissen.info/IoC-inversion-of-control-Umkehrung-des-Kontrollflusses.html> [siehe S. 31].
- PINKER, Alexander [Sep. 2016]. *Kurve der Innovationen- der Gartner Hype Cycle erklärt*. [https://medialist.info/2016/09/16/innovationen-im-wandel-der-zeit-der-gartner-hype\\_cycle/](https://medialist.info/2016/09/16/innovationen-im-wandel-der-zeit-der-gartner-hype_cycle/) [siehe S. 3].
- QUERVEL, Céline [2020]. *Augmented Reality (AR): Definition, Technik & Potenzial*. <https://worldofvr.net/augmented-reality/> [siehe S. 10].

- RICARDO [2019]. *Warum Android MVVM eine gute Sache ist.* <https://blog.cosee.biz/2019/04/18/warum-android-mvvm-eine-gute-sache-ist> [siehe S. 40].
- ROEING, Frank [2019]. *Datenmodellierung.* <https://de.wikipedia.org/wiki/Datenmodellierung> [siehe S. 44].
- SCHANZE, Robert [Okt. 2018]. *Was ist Mixed Reality? – Unterschied zu Virtual und Augmented Reality erklärt.* <https://www.giga.de/artikel/was-ist-mixed-reality-unterschied-zu-virtual-augmented-reality-erklaert> [siehe S. 15].
- SCHART, Dirk [2017]. *Augmented Reality für die digitale Zukunft der Industrie.* <https://www.vdi-wissensforum.de/news/augmented-reality-fuer-die-digitale-zukunft-der-industrie/> [siehe S. 21].
- SCHREYER, Eric [2018]. *Modulare Programmierung.* [https://de.wikipedia.org/wiki/Modulare\\_Programmierung](https://de.wikipedia.org/wiki/Modulare_Programmierung) [siehe S. 38].
- SELLS, Chris, Benjamin POIESZ und Karen NG [2018]. *Use Android Jetpack to Accelerate Your App Development.* <https://android-developers.googleblog.com/2018/05/use-android-jetpack-to-accelerate-your.html> [siehe S. 32].
- STRAND, Prof. Dr. Marcus [2020a]. *Einführung in die Robotik - Vorlesung 3 Externe Sensoren* [siehe S. 24].
- [2020b]. *Einführung in die Robotik - Vorlesung 5 ICP* [siehe S. 25].
- SUTHERLAND [Mai 1938]. *Biographie Ivan E. Sutherland.* <https://dl.acm.org/profile/81100265287> [siehe S. 12].
- TEPCON [2020]. *augmented presenter and augmented instructor.* [=https://www.tepcon.de/](https://www.tepcon.de/) [siehe S. 83].

## **Erklärung**

Ich versichere, dass ich diese Master-Thesis mit dem Thema: *“Konzeption und prototypische Umsetzung einer Steuerzentrale eines smarten Büros mit dem Fokus einer einfachen Handhabung der formalisierten Interaktionen für Softwareentwickler”* selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlichen oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet habe. Die Arbeit wurde noch keiner Kommission zur Prüfung vorgelegt und verletzt in keiner Weise Rechte Dritter.

---

Ort, Datum

Unterschrift