Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Henrik Wortmann

Objekterkennung unter Nutzung von Machine Learning für Augmented Reality Anwendungen

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Henrik Wortmann

Objekterkennung unter Nutzung von Machine Learning für Augmented Reality Anwendungen

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Masterstudiengang Automatisierung
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Jörg Dahlkemper
Zweitgutachter : Prof. Dr. Kai von Luck

Abgegeben am 17. Juni 2020

Henrik Wortmann

Thema der Masterarbeit

Objekterkennung unter Nutzung von Machine Learning für Augmented Reality Anwendungen

Stichworte

Objekterkennung, Machine Learning, Anwendung, App, iOS, Datenbank, Smartphone, Tablet

Kurzzusammenfassung

Diese Arbeit liefert ein Konzept, wie eine Augmented-Reality-Anwendung für iOS Smartphones und Tablets entwickelt werden kann, die Techniker bei Arbeiten an Schaltschränken unterstützt. Hierfür wird zunächst eine Objekterkennung auf Machine-Learning-Basis entworfen, die Bauteile in einem Schaltschrank erkennt. In der Anwendung wird jedes Bauteil mit Schaltflächen versehen, die durch Antippen zusätzliche Informationen liefern. Die Informationen sind in einer Datenbank hinterlegt, um größtmögliche Flexibilität im Hinblick auf die Implementierung weiterer Schaltschränke zu bieten.

Henrik Wortmann

Title of the paper

Object recognition using machine learning for augmented reality applications

Keywords

object detection, machine learning, app, iOS, database, smartphone, tablet

Abstract

This thesis provides a concept how to develop an augmented reality application for iOS smartphones and tablets that supports technicians working on control cabinets. For this purpose, an object recognition system based on machine learning is designed, which recognizes components in a switch cabinet. In the application each component is provided with buttons that provide additional information by tapping. The information is stored in a database in order to offer the greatest possible flexibility with regard to the implementation of further control cabinets.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ABI Application Binary Interface

AP Average Precision

AR Augmented Reality

CNN Convolutional-Neural-Network

CSTI Creative Space for Technical Innovations

HE Höheneinheit

HMD Head-Mounted Display

IMU Inertial Measurement Unit

mAP mean Average Precision

MR Mixed Reality

NFC Near Field Communication

NoSQL Not only SQL

R-CNN Region-based Convolutional Neural Networks

RFID Radio-Frequency Identification

SSD Single Shot Detector

SQL Structured Query Language

TOF Time of Flight

VR Virtual Reality

WLAN Wireless Local Area Network

YOLO You Only Look Once

1. Einführung

Smartphones und Tablets sind aus dem heutigen Alltag kaum noch wegzudenken. Diese Geräte sind durch technische Entwicklungen, erweiterte Konnektivität und umfangreiche Software mehr als reine Kommunikationsmittel, sie sind zu Helfern in vielen Alltagssituationen geworden. Durch die fortschreitenden Leistungssteigerungen mit Einführungen neuer Geräte bieten sich auch auf der Softwareseite breitere und anspruchsvollere Entwicklungsmöglichkeiten.
Erste Smartphones kamen bereits Ende der 90er-Jahre auf den Markt, ein Durchbruch in den Massenmarkt begann jedoch erst mit der Einführung des ersten iPhones 2007. Während bei PCs damals 3D-Grafiken bereits Alltag waren, war bei Smartphones an vergleichbare Anwendungen nicht zu denken. Mittlerweile hat sich die Rechenleistung von Smartphones deutlich weiterentwickelt und es können hochauflösende Grafiken dargestellt, Bilder verarbeitet und neuronale Netze verwendet und auch trainiert werden.
Zusätzlich zur Leistungsentwicklung wurde auch die Kamera- sowie Sensorausstattung über die Jahre verbessert bzw. erweitert. Diese Sensorik ermöglicht eine umfangreiche Erfassung der Umgebung, was sowohl spielerische Ansätze als auch professionelle Einsatzmöglichkeiten erlaubt.
Mit der Leistungs- und Sensorentwicklung haben auch „Augmented Reality“-(AR-) Anwendungen auf Smartphones Einzug gehalten. Die Kamera, in Verbindung mit der Sensorik, kann genutzt werden, um einem Anwender Informationen im Kamerabild live einzublenden. In der Spieleindustrie wird dies genutzt, um die virtuelle Welt mit der realen zu verschmelzen. Ein besonders erfolgreiches AR-Spiel war „Pokémon Go“ aus dem Jahre 2016 (vgl. Niantic 2016).
Große Unternehmen wie „Apple“, „Google“ u. a. haben die Entwicklung durch eigene Frameworks vereinfacht und dadurch die Verbreitung von AR unterstützt. Die bekanntesten Vertreter sind „ARKit“ von Apple (vgl. Apple 2019a) und „ARCore“ von Google (vgl. Google 2019a). Diese Frameworks sind frei verfügbar, sodass jeder damit Apps entwickeln kann. Auch die Frameworks werden von den Herstellern kontinuierlich weiterentwickelt. ARKit ist bereits in der dritten Version erschienen und ermöglicht sogar AR-Anwendungen mit mehreren Benutzern.
Ein weiteres Einsatzszenario wurde beispielsweise vom Autohersteller Daimler gezeigt: Statt einer Bedienungsanleitung in Papierform ist es mit einer App auf dem Smartphone möglich, die Bedienelemente des Autos mit der Smartphonekamera zu erfassen und als Annotation

die Bezeichnung und auf Wunsch eine genaue Erklärung zu Bedienelementen zu erhalten (vgl. Daimler 2017).
Aber nicht nur für Kunden können AR-Anwendungen interessant sein. Auch für Techniker unterschiedlichster Bereiche wäre es eine Entlastung, wenn eine direkte Einblendung Informationen liefert, sodass keine lange Suche nach den entsprechenden Informationen im Schaltplan erforderlich ist.
In der Industrie erfreut sich AR wachsender Beliebtheit. Immer mehr Unternehmen beginnen, das Potential zu erkennen und untersuchen den individuellen Mehrwert. Diese Entwicklung ist nicht nur auf einzelne Industriebereiche beschränkt, sondern zieht sich durch unterschiedlichste Branchen. Einige mögliche Anwendungen hat das Magazin „Forbes“ 2018 in einem Onlineartikel zusammengefasst (vgl. Marr 2018). Ein interessantes Beispiel hat der „Gatwick Airport“ mit der „Gatwick Passenger App“ geliefert und ist dafür mit mehreren Preisen ausgezeichnet worden (vgl. Koolonavich 2018). Mit dieser App werden Besucher mit Hilfe von Navigationslinien im Kamerabild des Smartphones zu ihren Zielen, z.B. Terminals oder Schalter, gelotst. Der britische Farbhersteller „Dulux“ bietet eine App an, mit der der Kunde durch das Smartphone ausprobieren kann, wie die Farben an den eigenen Raumwänden aussehen (vgl. Dulux 2017).
Ein weiteres Beispiel ist eine App des schwedischen Möbelhauses „Ikea“. Diese ermöglicht es, auf dem Smartphone Möbel im Raum zu platzieren und mit Hilfe der Smartphonekamera im Raum anzuschauen (vgl. Ikea 2017). Dem Nutzer wird so räumlich dargestellt, wie das Möbelstück an dem dafür vorgesehenen Ort aussehen würde.
Unter den Begriffen „Industrie 4.0“, „Internet of Things“ (IoT) oder auch „Smart Factory“ versteht man im Allgemeinen intelligente und vernetzte Systeme (vgl. Kagermann, Wahlster und Helbig 2012). Besonders die Vernetzung ist für die AR interessant. Mit Hilfe von AR-Anwendungen ist es möglich, Daten aus der Umwelt oder von Sensoren im System in das Kamerabild einzublenden und so einen besseren Überblick zu schaffen. Vernetzte Sensoren können von der AR-Anwendung abgefragt und deren Daten angezeigt werden. Dabei kann es sich beispielsweise um die Anzeige der aktuellen Motordrehzahl handeln, die eingeblendet wird, sobald die Anwendung den Motor im Bild erkannt hat.

1.1. Zielsetzung

Ziel dieser Arbeit ist es, die AR-Technologie zu nutzen, um eine Smartphoneanwendung zu entwickeln, die Technikern bei Arbeiten an Schaltschränken hilft, ihre Aufgaben zu erledigen. Schaltschränke bestehen häufig aus mehreren hundert Bauteilen und noch mehr Anschlüssen. Damit sich ein Techniker darin zurechtfindet, ist ein umfangreicher Schaltplan nötig. Um den richtigen Punkt zu finden, muss der Techniker in einer Tabelle nach der Bauteilbezeichnung suchen, dann die entsprechende Seite ablesen und dorthin blättern. Dieses Vorgehen ist zwar übersichtlich, kostet aber auch viel Zeit. Mit Hilfe einer AR-Anwendung

könnte der Techniker einen Anschluss anklicken und würde sofort auf die entsprechende Seite im Schaltplan geleitet werden. Der Techniker wird so einerseits zeitlich entlastet, andererseits muss er aber auch keinen kompletten Ordner mitbringen. Schaltpläne werden heute zwar auch in einzelnen Fällen als PDF mitgenommen, aber auch hier sind die beiden anfangs beschriebenen Schritte, diesmal in digitaler Form, notwendig, um die korrekte Seite aufzurufen.
Zunächst wird untersucht, wie eine derartige Anwendung aussehen muss, um einen Mehrwert zu bieten. Weiterhin wird geprüft, welche Informationen für den Anwendungsfall angezeigt und hinterlegt werden müssen. Neben dem Inhalt wird außerdem eruiert, wie die Informationen aufbereitet und dargestellt werden sollten, damit sich für den Techniker der maximale Mehrwert ergibt.
Es wird angestrebt, dass die Anwendung universell angepasst werden kann, unabhängig von Art und Aufbau des Schaltschranks. Ein einfacher Sicherungskasten kann dabei bereits als Schaltschrank angesehen werden.
Diese Arbeit konzentriert sich auf elektrische Arbeiten. Mit Informationen zum mechanischen Ein- und Ausbau von Bauteilen wird der Techniker nicht versorgt. In Form von Annotationen oder hinter weiteren Schaltflächen können allerdings Zusatzinformationen bereitgestellt werden.

1.2. Motivation

Schaltschränke gibt es in nahezu jedem Unternehmen. Je nach Art und Bauteilen im Schrank müssen unterschiedliche Arbeiten ausgeführt werden. Für viele Bauteile gibt es vorgeschriebene Regelwartungen (z. B. Schmierung von Lagern) und Inspektionen (z. B. an Verschleißteilen), die regelmäßig durch Techniker durchgeführt werden müssen, und außerplanmäßige Wartungen, die beispielsweise durch einen Defekt oder Rückruf des Herstellers verursacht werden, und eine Instandsetzung erfordern. Kosten für Wartungsarbeiten finden sich als Wartungsaufwand in der Bilanz von Unternehmen wieder. Dabei ist es unerheblich, ob es sich um Wartungspersonal des Unternehmens oder um eine Fremdfirma handelt. Häufig muss außerdem für Wartungsarbeiten der entsprechende Produktionsteil abgeschaltet werden, wodurch zusätzliche Kosten entstehen.
Becker & Brinkmann (vgl. Becker und Brinkmann 2000) haben sich im Jahr 2000 mit der Kostenrechnung für die Instandhaltung in Unternehmen befasst. Dabei haben sie auch die Wartungskosten ermittelt und festgestellt, das sich diese je nach Branche in einer Größenordnung von durchschnittlich 3% bis 7% der Anschaffungskosten bewegen. Von diesem Anteil wiederum entfallen durchschnittlich 41,8% auf die Personalkosten. Hinzu kommen nochmals Fremdleistungskosten von durchschnittlich 29,6%, die sich aus weiteren Personal- und Materialkosten zusammensetzen. Personalkosten verursachen somit einen großen Teil der Instandhaltungskosten in Unternehmen. Wenn die Einsatzzeiten von Technikern durch eine

AR-Anwendung reduziert werden können, bedeutet dies eine deutliche Einsparung für die Unternehmen. Sind die Anlagen für Mitarbeiter nur schwer zugänglich, z. B. bei Offshore-windkraftanlagen, können die Kosten sehr viel höher sein. Hier wird mit bis zu einem Viertel der Anschaffungskosten gerechnet (vgl. Bundesverband WindEnergie 2018). Kürzere Arbeitszeiten können die Kosten daher besonders stark reduzieren.
Der Kostenpunkt ist aber nicht der einzige Grund, der für die Unterstützung mit einer App spricht. Mit Hilfe der App können wichtige Informationen, die Auskunft darüber geben, was beim entsprechenden Arbeitsvorgang besonders zu beachten ist, weitergegeben werden. Zusatzinformationen können insbesondere der Fehlervermeidung dienen. Außerdem kann dies eine Entlastung des Technikers, gerade bei selten ausgeführten Aufgaben, bedeuten. Auch die Fehlerdiagnose lässt sich stark verkürzen und vereinfachen, wodurch weitere Kostenersparnisse entstehen. Defekte Teile können direkt im Kamerabild markiert oder Livedaten eingeblendet werden, die zur schnellen Fehlerdiagnose verhelfen können.

2. Stand der Technik

Im folgenden Kapitel werden grundlegende Informationen zum Stand der Technik und Forschung im Bereich Augmented Reality sowie zur damit verbundenen Software dargestellt. Darüber hinaus werden hier mögliche Aufgaben aufgezeigt, die Techniker an Schaltschränken vornehmen und die durch AR unterstützt werden können.

2.1. Augmented Reality im Mixed-Reality-Kontinuum

Augmented Reality bezeichnet die computergestützte Erweiterung der Realität. Dieses Konzept ist grundsätzlich nicht neu, sondern geht auf die Idee eines Head-Mounted-Displays (HMD) von Ivan Sutherland im Jahre 1968 zurück (vgl. Sutherland 1968). Seine Idee besteht darin, das Bild, welches man sieht, mit einem passenden zweidimensionalen Bild zu überlagern, um die Illusion zu erzeugen, man würde ein dreidimensionales Objekt vor sich haben. Dabei muss sich die perspektivische Ansicht der Überlagerung exakt mit der Kopfbewegung des Betrachters mitbewegen. Bewegte, perspektivische Bilder erscheinen dabei immer dreidimensional, auch wenn es sich um keine Stereodarstellung handelt. In der Psychologie ist dieses Phänomen unter dem Begriff „kinetischer Tiefeneffekt“ bekannnt (vgl. Green 1961). Aufgrund dieses Effekts ist es möglich, auf einem einfachen Smartphonedisplay die Illusion eines augmentierten, dreidimensionalen Objekts zu erzeugen, ohne dass ein Stereobild notwendig ist.
Im Jahre 1994 hat Paul Milgram mit dem „Realitäts-Virtualitäts-Kontinuum“ eine Einordnung der AR zwischen Realität und Virtualität geliefert. Dabei wird der Bereich, der Virtualität und Realität mischt, als „Mixed Reality“ (MR) bezeichnet. In diesem Bereich werden die Terminologien „Augmented Reality“ und „Augmented Virtuality“ (AV) genannt. Grundsätzlich bezeichnet AR, wie bereits eingangs genannt, die Erweiterung der Realität mit virtuellen Daten. Umgekehrt wird von einer augmentierten Virtualität gesprochen, wenn eine virtuelle Darstellung mit Daten aus der realen Welt angereichert wird. Dies hat Milgram in seiner Veröffentlichung grafisch als kontinuierlichen Strahl dargestellt (Abbildung 2.1). Heute wird der Begriff der augmentierten Realität vorwiegend durch Anwendungen auf Smartphones und Tablets geprägt, die im Kamerabild zusätzliche Informationen oder Objekte darstellen. Die Ergänzung der Realität ist oft auf optische Eindrücke beschränkt.

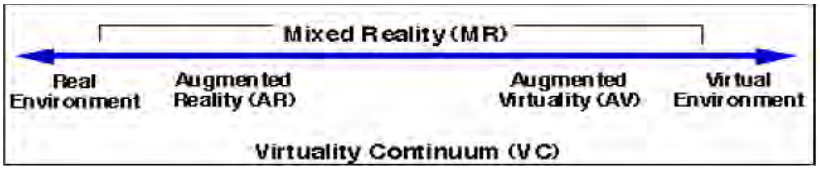


Abbildung 2.1.: Realitäts-Virtualitäts-Kontinuum nach Milgram (Milgram und Kishino 1994)

2.1.1. Mögliche Ausprägungen der Augmented Reality

Der Begriff AR beschreibt die Erweiterung der Realität sehr allgemein. Dabei können die Ausprägungen unterschiedlich sein, indem verschiedene Wahrnehmungsbereiche angesprochen werden, nämlich die menschlichen Sinne Sehen, Hören, Fühlen, Riechen und Schmecken (vgl. Nowitzki 2003).

2.1.1.1. Sehen

Die am weitesten verbreitete Form der AR ist diejenige, die mit Bildern die Realität erweitert. Häufig geschieht dies mit Hilfe einer Kamera und einem Display, welches das Kamerabild wiedergibt und mit zusätzlichen Informationen oder Grafiken ergänzt. Diese Variante wird auf Smartphones oder Head-Mounted-Devices (HMDs) wie z. B. der Hololens (vgl. Microsoft 2019b) oder Google Glass (vgl. Google 2019b) genutzt.
AR wird bereits seit Mitte der neunziger Jahre bei Sportübertragungen eingesetzt. Einblendungen wie Yard-Linien beim Football, Abseitslinien beim Fußball oder Weitenlinien beim Skispringen entsprechen genau der Idee von AR. Ein reales Bild wird hier durch überlagerte Einblendungen mit zusätzlichen Informationen angereichert.
Eine andere Möglichkeit ist auch die Projektion eines Beamerbildes. Hier werden die Informationen mit Hilfe eines Beamers direkt auf die reale Welt projiziert. Ein Beispiel hierfür stammt aus der Medizin: Vorschlag: "... aus der Medizin: Hier kann der Einsatz eines Videobildes, das die inneren Organe auf die Bauchdecke des Patienten projiziert, operative Eingriffe erleichtern (vgl. Volonté u. a. 2011). Weitergehende Beispiele zeigen den Einsatz eines Endoskops, welches durch Einblendungen den Arzt anleitet (vgl. Sielhorst, Feuerstein und Nassir Navab 2008).
Ein weiteres Beispiel für die Beameranwendung ist „StereoBlocks“ (vgl. Jota und Benko 2011). Hier können reale Bauklötze gesetzt werden, während mit einem Beamerbild das zugehörige 3D-Modell gezeigt wird. An dieser Stelle kommt eine weitere Variante ins Spiel: das Fühlen.

2.1.1.2. Fühlen

Neben optischen Reizen können auch haptische Reize als Erweiterungen der Realität verwendet werden. Beispielsweise kann repräsentativ für einen virtuellen Gegenstand ein reales Objekt bewegt werden, wobei der Anwender trotzdem den virtuellen Gegenstand sieht. Die Augmentierung wird so um eine Tastkomponente erweitert. Ein Beispiel hierfür liefert, wie im Abschnitt „Sehen“ erwähnt, „StereoBlocks“. Auch Oberflächen können mittels Vibrationsmotoren haptisch erfühlt werden. Dies ermöglicht unter anderem Interaktionen mit Museumsgegenständen, die eigentlich hinter Glas und für Besucher unzugänglich sind (vgl. Dima, Hurcombe und Wright 2014).

2.1.1.3. Hören

Eine weitere gängige Form der Augmentierung ist das Hören. Mit Kopfhörern können Informationen oder Geräusche eingebracht werden, die die optische Wahrnehmung der Situation ergänzen. In einem Artikel, erschienen in der „Technology Review“ im Jahr 2017, berichtet Kölling über Ansätze zur akustischen Augmentierung (vgl. Kölling 2017). Durch akustische AR könnten Ablenkungen durch Einblendungen vermieden werden und z.B. Statuen zu sprechen beginnen, was für die Tourismusindustrie vielfältige Möglichkeiten eröffnet.

2.1.2. Darstellungsmöglichkeiten und Formen in der AR

Aus den erwähnten unterschiedlichen Wahrnehmungsarten ergeben sich vielfältige Möglichkeiten für die Darstellung in der AR. Bei der grafischen Darstellung ist nicht nur relevant mit welcher Technologie diese gezeigt wird, sondern auch die Art der Darstellung. Handelt es sich um eine 2D- oder 3D-Grafik? Handelt es sich um ein Overlay oder sollen lediglich einige Punkte annotiert werden? Je nachdem welche Variante gewählt wird, hat das auch Einfluss auf die Entwicklungsschritte. Auch ob es sich um eine Liveeinblendung handeln soll, muss vorher anhand der Anforderungen überlegt werden.
Zusätzlich ist es wichtig zu ermitteln, wo Einblendungen positioniert werden können. Hierfür ist ein „View Management“ notwendig, welches Annotationen skaliert und in freien Bildbereichen darstellt (vgl. Bell, Feiner und Höllerer 2001). Nachfolgend werden unterschiedliche Augmentierungsmöglichkeiten kurz vorgestellt:

2.1.2.1. Zweidimensionale Einblendungen

Zweidimensionale Einblendungen können in unterschiedlichen Formen präsentiert werden. Neben Überblendungen (Overlays), die Bereiche des Bildes überdecken und so hervorhe-

ben bzw. abdecken, stehen auch Annotationen zur Auswahl. Dabei werden Bereiche umrandet oder hervorgehobene Bereiche gewählt, an denen zusätzliche Informationen angehängt sind. Eine zweidimensionale Einblendung muss dabei nicht statisch sein, sondern kann auch dynamisch, z. B. mit Animationen hervorgehoben werden.

2.1.2.2. Dreidimensionale Einblendungen

Dreidimensionale Einblendungen sind häufig Objekte, die in einem definierten Bereich im Sichtfeld hinzugefügt werden. Ein häufig genutzter Anwendungsfall ist die Einblendung eines Produkts in dreidimensionaler Ansicht, sobald ein Foto des Objekts abgescannt wurde. Oft wird dies mit Animationen des Produkts verbunden, bei dem der Nutzer auch die Funktionen ausprobieren kann. Ein weiterer Anwendungsfall ist das Anprobieren von Uhren. Der Anwender hält das Smartphone über sein Handgelenk und bekommt ein dreidimensionales Modell der gewünschten Uhr auf seinem Handgelenk angezeigt (vgl. AR-Watches 2019).

2.1.2.3. Audio und Vibration

Wie in Abschnitt 2.1.1 bereits erwähnt, gehören nicht nur optische Eindrücke zur AR, sondern auch akustische und haptische. Mit Audioeinspielungen kann der Eindruck von Objekten verfeinert werden, indem der Nutzer auch die zugehörigen Geräusche wahrnehmen kann. Vibrationsmotoren können dafür sorgen, dass die Interaktion mit Objekten realistischer wird, wenn beispielsweise bei Berührung eine kurze Vibration ausgelöst wird. Bei dieser Anwendung spricht man von einem „haptischen Feedback“.

2.1.3. Interaktionsmöglichkeiten

Damit eine AR-Anwendung nicht nur in Ausgaberichtung funktioniert, sondern durch Eingaben vom Benutzer beeinflusst werden kann, sind Interaktionen notwendig. Je nach verwendetem Gerät können unterschiedliche Arten der Interaktion genutzt werden. Dabei ist nicht zwingend eine bestimmte Interaktionsart nötig, sondern es kann auch mehrere Bedienungsoptionen geben. Da sich die Rechenleistung von kleinen, mobilen Geräten laufend verbessert, entstehen auch immer weitere Möglichkeiten.
Die wohl häufigste Interaktionsart in der AR ist primär im Smartphone- und Tabletbereich angesiedelt und erfolgt über einen Touchscreen, welcher im Regelfall auch dem Ausgabegerät entspricht. Durch „Berührung“ können Interaktionen direkt in der augmentierten Umgebung ausgeführt werden. Ein möglicher Anwendungsfall ist das Aufrufen von Zusatzinformationen durch Antippen eines Objekts.

Im Falle von Head-Mounted-Displays (HMDs) (siehe Abschnitt 2.2.1.2) gibt es keinen Touchscreen, über den eine Interaktion möglich wäre. Hier wird die Bedienung über die eingebauten Kameras vorgenommen. Dazu gibt es vordefinierte Gesten, die mit den Händen bzw. Fingern im Sichtfeld an den virtuellen Objekten ausgeführt werden. Beispielsweise bewirkt eine Klickbewegung mit dem Zeigefinger die Auswahl des virtuellen Objekts.
Im Mixed-Reality-Bereich bieten sich weitere haptische Interaktionsmöglichkeiten. Die Manipulation von realen Objekten kann eine analoge Reaktion eines virtuellen Objekts auslösen. Ein Beispiel wäre ein Rad, an dem gedreht werden kann, was anschließend die Drehung eines virtuellen Objekts auslöst.
Neben den bereits beschriebenen, haptischen Interaktionsmöglichkeiten gibt es auch die Möglichkeit, eine Sprachsteuerung zu integrieren. Auf vielen Geräten sind heute bereits Sprachassistenten wie Siri (vgl. Apple 2019b), Google (vgl. Google 2019c), Alexa (vgl. Amazon 2019) oder Cortana (vgl. Microsoft 2019a) vorhanden, die auch aus dem Smart-Home-Bereich bekannt sind. Analog dazu kann eine Spracherkennung verwendet werden, um auf Zuruf Objekte in der AR zu manipulieren oder auszuwählen.

2.2. Geräte

Mit dem technologischen Fortschritt kommen immer mehr AR-taugliche Geräte auf den Markt. Wie eingangs beschrieben, bestand die Idee der AR im Jahre 1968 aus einem HMD. Diese Art der AR-Geräte wurde stetig weiterentwickelt. Bekannte Vertreter dieser Gattung sind die Microsoft Hololens und Google Glass. Während anfangs nur einfache Drahtgitter eingeblendet werden konnten, sind heute animierte, dreidimensionale Objekte samt Live- und Audiodaten zeigbar. HMDs haben einen praktischen Vorteil: Da sie auf dem Kopf getragen werden, bleiben die Hände frei. Ein Arbeiter kann durch Einblendungen geleitet werden, während er mit beiden Händen Arbeiten ausführt. Bei der Verwendung eines Smartphones muss der Anwender das Gerät in der Hand halten und kann nur einhändig Arbeiten durchführen. Ein Nachteil der HMDs ist der hohe Preis, weshalb diese Geräte fast ausschließlich in der Forschung und im professionellen Bereich zu finden sind.
Eine weitere Variante der Geräte, die Augmentierungen erzeugen können, sind Computer in Verbindung z. B. mit Beamern. Durch Beamer können Einblendungen direkt auf die realen Objekte projiziert werden. Diese Art der AR erfordert allerdings Aufbau und Ausrichtung der Beamer vor Ort und ist daher nur beschränkt mobil einsetzbar.
Die beiden bereits genannten Gerätetypen sind jedoch häufig teuer bzw. nicht mobil. Am weitesten verbreitete AR-taugliche Geräte sind Smartphones und Tablets (hier zusammengefasst). Abbildung 2.2 zeigt, dass die Smartphoneverbreitung in den vergangenen Jahren stark gestiegen ist und ein weiterer Anstieg zu erwarten ist. Zwar ist nicht jedes Smartphone AR-tauglich, AR Anwendungen geraten jedoch zunehmend in den Fokus der Hersteller. Es wird explizit mit AR-fähiger Hardware geworben. Durch die Verbreitung der Smartphones

wird die Nutzerbasis deutlich breiter und es wird für Entwickler attraktiver, die AR-Plattform für Anwendungen zu nutzen.

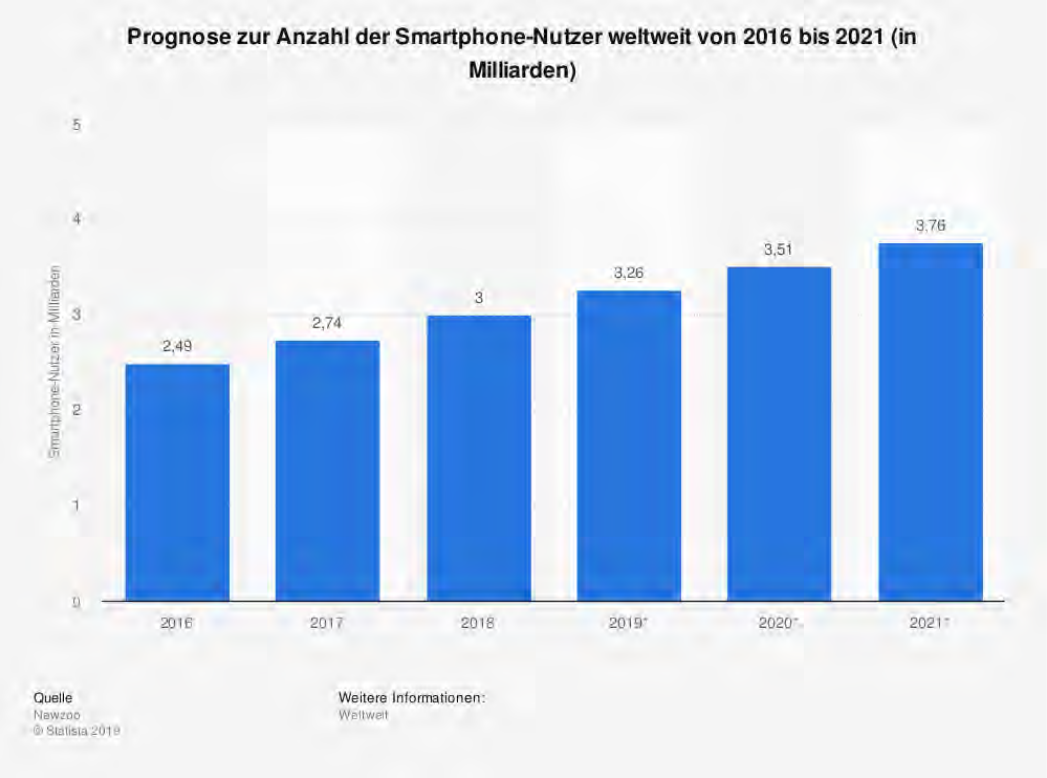


Abbildung 2.2.: Prognose zur Anzahl der Smartphone-Nutzer weltweit von 2016 bis 2021 in Milliarden (VentureBeat 2019)

2.2.1. Sensoren in HMDs und Smartphones

Heute sind in Smartphones, Tablets, HMDs etc. viele unterschiedliche Sensoren verbaut, um die Umgebung präzise zu erfassen und auch Interaktionen zu ermöglichen. Zusätzlich besteht oft die Möglichkeit, diese Geräte mit externen Sensoren zu erweitern, um noch mehr Einsatzszenarien abzudecken. Je nach Smartphone oder Tablet gibt es Abweichungen bei den Sensorausstattungen, wobei einzelne Sensoren wegfallen oder andere, spezielle Sensoren hinzukommen können. Aufgrund der Vielzahl von unterschiedlichen Smartphones am Markt werden hier lediglich die häufigsten Sensoren erfasst. Die HMDs werden gesondert betrachtet, da diese einer anderen Gerätegruppe zuzuordnen sind und entsprechend andere Sensoren aufweisen.

2.2.1.1. Smartphones/Tablets

Nachfolgend werden Sensoren aufgelistet, welche in den meisten Smartphones enthalten sind, und deren Funktion erklärt (vgl. Androidmag 2014):

1. **Beschleunigungssensor:** Dieser Sensor erfasst die auf das Gerät einwirkende Beschleunigung in X-, Y- und Z-Richtung. Die Sensordaten werden beispielsweise genutzt, um Spiele zu steuern, um die Ausrichtung des Geräts zu erfassen und Inhalte daran anzupassen oder als Ergänzung zur Navigation mit GPS. So kann der Weg weiterverfolgt werden, auch wenn kein GPS-Signal vorhanden ist.

2. **Drehratensensor:** Dieser Sensor erfasst die Rotationen des Smartphones. Die Daten können ebenfalls für die Steuerung von Spielen genutzt werden. Weiterhin ist dieser Sensor besonders wichtig für die Bildstabilisierung bei Foto- und Videoaufnahmen.

3. **Ortungssensorik:** Die Ortung erfolgt durch mehrere Sensoren. Die Verwendung mehrerer unterschiedlicher Systeme zur Positionsbestimmung erhöht die Genauigkeit und bietet Redundanz.

   GPS, Galileo, Glonass: Hierbei handelt es sich um satellitengestützte Systeme. Anhand von Satellitensignalen berechnet das Smartphone seine Position auf wenige Meter genau. Häufig sind die Geräte in der Lage, neben dem bekannten amerikanischen GPS auch das europäische System Galileo und das russische Glonass System zu nutzen.

   Mobilfunk: Zusätzlich zum Satellitensignal werden auch Mobilfunksender im Empfangsbereich genutzt.

   WLAN: Auch anhand des WLAN-Empfangsbereichs kann die Position weiter eingegrenzt werden.

   Bluetooth: Im Nahbereich kann Bluetooth zusätzlich zur Lokalisierung weiterer Bluetoothgeräte eingesetzt werden.

4. **Magnetometer:** Mit dem Magnetometer kann das Smartphone den magnetischen Nordpol erfassen und erhält so eine Kompassfunktion, mit der auch die Ausrichtung in der Navigation möglich ist. Ein weiterer Einsatzzweck ist die Displayabschaltung mit magnetischen Hüllen. Sobald man die Hülle zuklappt, wird das Display deaktiviert.

5. **Näherungssensor:** Dieser Sensor ist für die Telefonfunktion besonders wichtig. Er sorgt dafür, dass das Display deaktiviert wird, sobald das Handy an das Ohr gehalten wird. So wird vermieden, dass beim Telefonieren unerwünschte Eingaben getätigt werden.

6. **Thermometer und Feuchtigkeitssensor:** Umgebungstemperatur und Umgebungsfeuchtigkeit können von einigen Geräten erfasst werden. Die Daten werden zum Beispiel für Gesundheitsapps verwendet.
7. **Helligkeitssensor:** Mit diesem Sensor wird die Umgebungshelligkeit erfasst und die Displayhelligkeit kann automatisch in Abhängigkeit von der Umgebungsbeleuchtung angepasst werden.
8. **Barometer:** Mit dem Barometer wird anhand des Luftdrucks eine Höhenmessung vorgenommen. Dieser Wert dient als Ergänzung zur Höhenmessung durch GPS. Eine Indoornavigation über mehrere Stockwerke wird so auch möglich.

2.2.1.2. Head-Mounted-Displays

Head-Mounted-Displays weisen teilweise andere Sensoren auf als Smartphones. Hier werden exemplarisch die Sensoren der Microsoft Hololens (vgl. Microsoft 2019b) vorgestellt, da diese am Markt verfügbar sind. Die Sensoren decken sich mit denen der Google Glass. Sowohl von Google als auch von Microsoft sind Nachfolgemodelle angekündigt, aber noch keine genauen technischen Daten verfügbar.

1. **Inertial Measurement Unit (IMU):** Die IMU vereint Beschleunigungssensor, Gyroskop und Magnetometer in einem Chip und ist für die Erfassung der Nutzerbewegungen zuständig.
2. **4 Umgebungskameras:** In Verbindung mit der nachfolgend beschriebenen Tiefenkamera wird mit diesen speziellen Grauwertkameras ein dreidimensionales Bild der Umgebung erfasst.
3. **Tiefenkamera:** Eine Tiefenkamera wird genutzt, um ein dreidimensionales Bild der Umgebung zu erfassen und Entfernungen zu ermitteln, um Augmentierungen zu positionieren oder mit ihnen zu interagieren. Die Entfernungen werden mittels „time of flight“ (TOF) gemessen.
4. **Foto/HD-Kamera:** Diese Kamera kann für hochauflösende Aufnahmen oder für eine zusätzliche farbliche Umgebungserfassung genutzt werden.
5. **4 Mikrofone:** Die Mikrofone sorgen dafür, dass die Akustik der Umgebung wahrgenommen und entsprechend verarbeitet werden kann. So können Einblendungen in Abhängigkeit von akustischen Signalen entstehen.
6. **Umgebungslichtsensor:** Ähnlich wie beim Smartphone kann dieser Sensor zur weiteren Erfassung der Umgebung dienen und Anpassungen an die Umgebungsbeleuchtung ermöglichen.

Die Displays der HMDs arbeiten häufig nach dem See-Through-Prinzip. Im Falle der Hololens handelt es sich um ein sogenanntes „Waveguide Display“. Bei dieser Displayart werden die Signale über Glasfasern am Rahmen eingespeist und durch ein spezielles Glas zum eigentlichen Projektionspunkt geleitet (vgl. Kress und Cummings 2017).
Da HMDs nur wenige Schalter und kein Touchdisplay besitzen, gibt es andere Wege der Interaktion. Häufig wird mit Fingergesten gearbeitet, die von den Kameras erfasst werden. Ein „Klicken“ mit dem Finger in der Luft kann eine Aktion auslösen. Als weitere Interaktionsmöglichkeit steht mit Hilfe der Mikrofone die Sprachsteuerung zur Verfügung.
Neben den HMDs, die nach dem See-Through-Prinzip arbeiten, setzen auch einige VR-Brillen auf AR-Funktionen. Beispielsweise bietet die HTC Vive Pro (vgl. HTC 2019) durch integrierte Kameras die Möglichkeit, durch Einblendung der Umgebung eine AR zu schaffen. Diese Geräte sind aufgrund von Größe, Gewicht und der Tatsache, dass sie zusätzlich externe Hardware erfordern, nur begrenzt für einen mobilen Einsatz geeignet und werden daher in dieser Arbeit nicht weiter betrachtet.

2.2.2. Sensorerweiterungen

Bei HMDs gibt es in der Regel nur wenige Schnittstellen, mit denen die Geräte erweitert werden können. Dies sorgt dafür, dass diese Geräte meist auf die interne, vom Hersteller verbaute Sensorik beschränkt sind. Bei Smartphones hingegen sind die internen Sensoren fest verbaut. Es gibt jedoch zusätzliche Module, die aufgesteckt werden können.
Ein bekanntes Zusatzmodul stammt von der amerikanischen Firma „FLIR“. Mit dem Modell „FLIR ONE Gen 3“ (vgl. FLIR 2019) ist es möglich, Smartphones um eine Wärmebildkamera zu erweitern. So kann das Smartphone in begrenztem Umfang dedizierte Wärmebildkameras ersetzen und der Nutzer spart sich ein zusätzliches Gerät. Gleichzeitig können weitere Sensoren in die Funktionalität von Apps eingebunden werden. Im Falle einer Wärmebildkamera kann ein Elektriker diese zur Fehlererkennung nutzen oder es kann eine Überprüfung der Isolation von Gebäuden erfolgen.
Von unterschiedlichen Herstellern gibt es weitere Zusatzmodule mit diversen Funktionen. Meist handelt es sich um Kameramodule, aber auch Sensoren wie Geigerzähler sind verfügbar.

2.2.3. Konnektivität

Wichtig für den Einsatz von Geräten, die Anwender mit Augmentierungen unterstützen sollen, ist die Konnektivität der jeweiligen Geräte. Durch Kommunikationsmöglichkeiten bieten

sich Einsatzszenarien, in denen der Anwender aktuelle Daten einblenden kann. Für die Kommunikation sind verschiedene Technologien nutzbar, welche sich auf unterschiedlichen Strecken einsetzen lassen.

- **Mobilfunk (Smartphones/Tablets):** Ist immer bei Smartphones verbaut und häufig optional bei Tables erhältlich. Das Mobilfunknetz besitzt eine große Reichweite und hohe Netzabdeckung. Je nach Gerät und Empfang können hohe Übertragungsraten (bis zu 10 Gbit/s) erreicht werden. Im internationalen Vergleich ist die Netzabdeckung mit dem schnellen 4G-Netz in Deutschland (Abbildung 2.3) vergleichsweise gering.
- **WLAN (Smartphones/Tablets/HMD):** Arbeitet meist mit Reichweiten bis ca. 100 m, je nach Bebauung. Ein Lokaler Zugriffspunkt muss in Reichweite sein. Die Übertragung findet im 2,4 Ghz- und 5 Ghz-Bereich statt. Je nach Standard können die Übertragungsraten bis zu sieben Gigabit betragen.
- **Bluetooth (Smartphones/Tablets/HMD):** Arbeitet mit einer geringen Datenrate von bis zu 1 Mbit/s und einer Reichweite von maximal 100 m im Freifeld. Bluetooth zeichnet sich durch einen schnellen Verbindungsaufbau und geringem Energieverbrauch (Bluetooth Low Energy ab Version 4.0) aus.
- **NFC & RFID (Smartphones/Tablets):** Wird für sehr kleine Datenmengen, häufig zur Identifikation von Bauteilen, verwendet. Die Reichweite beträgt bei RFID in der Regel maximal 50 cm und bei NFC maximal 10 cm.
- **Infrarot:** Früher häufig bei Smartphones für Dateiübertragungen mit geringer Datenrate verwendet. Funktioniert nur über kurze Sichtverbindungen.

2.3. Tracking

Damit AR funktioniert, muss permanent verfolgt werden, wo sich das Gerät befindet und wie es relativ zum augmentierten Objekt positioniert ist. Je stabiler die Verfolgung des Gegenstands ist, egal ob zwei- oder dreidimensional, desto präziser lassen sich virtuelle Objekte oder Annotationen ausrichten. Ein instabiles Tracking kann sich dabei durch unterschiedliche Effekte bemerkbar machen, die das Erlebnis des virtuellen Objekts bzw. der Annotation stören:

- Das Tracking kann komplett verloren gehen, das heißt, die Einblendung verschwindet oder taucht an anderen Orten wieder auf.
- Durch Messfehler springt die Position der Einblendung auf dem Bildschirm.

Abbildung 2.3.: 4G-Netzabdeckung in Europa (Wagner 2018)

Die Stabilität des Trackings hängt einerseits mit der verwendeten Gerätesensorik zusammen, andererseits aber auch mit dem getrackten Objekt. Jedes Objekt liefert unterschiedlich deutliche Referenzpunkte.
Für das Tracking gibt es unterschiedliche Verfahren. Die wichtigsten werden nachfolgend kurz vorgestellt.

2.3.1. Optisches Tracking

Das optische Tracking ist eine häufig eingesetzte Technologie. Dabei handelt es sich um einen Oberbegriff, der alle Trackingtechnologien umfasst, die mit Licht arbeiten. Zu den optischen Verfahren wird insbesondere zwischen Tracking mit oder ohne spezielle Marker unterschieden.

2.3.1.1. Tracking mit speziellen Markern

Beim Markertracking werden spezielle Grafiken oder dreidimensionale Marker verwendet, um die Erfassung z. B. mit einer Kamera zu erleichtern. Häufig handelt es sich hierbei um reflektierende Kugeln, wie beim „Advanced-Realtime-Tracking-System“ (vgl. Tracking 2019). An diesen Markern wird projiziertes Licht im sichtbaren oder infraroten Bereich reflektiert und durch mehrere Kameras erfasst. Aus diesen Daten wird die Position des Objekts im Raum errechnet und kann für Bewegungsaufzeichnungen oder Einblendungen in AR oder VR genutzt werden.
Entscheidend ist sowohl bei zwei- als auch bei dreidimensionalen Markern, dass diese keine Rotationssymmetrien enthalten, da die Ausrichtung sonst nicht eindeutig bestimmt werden kann. Bei den zweidimensionalen Markern handelt es sich um spezielle Aufdrucke, die in der Regel quadratisch und in Schwarz und Weiß gehalten sind. Bekannte Vertreter sind QR-Codes und ARUCO-Marker. QR-Codes sind einfach und günstig zu generieren, können gedruckt und von Handykameras häufig ohne Zusatzapps erfasst werden. Meist sind in diesen Markern Links versteckt, die den Nutzer zu zusätzlichen Informationen leiten oder auch Apps aufrufen. Diese Codes kommen in unterschiedlichen Komplexitäten vor und unterscheiden sich hinsichtlich ihrer Redundanz und der speicherbaren Zeichen. Mit weiterentwickelten Varianten können auch mehrere Tausend Zeichen in einem Code gespeichert werden (vgl. Wave 2020). Aufgrund der damit einhergehenden Komplexität ist die eindeutige Erkennung von QR-Codes aus größeren Distanzen schwierig, weshalb diese nur begrenzt für Tracking geeignet sind (vgl. Elgendy, Guzsvinecz und Sik-Lanyi 2019).
An dieser Stelle kommen ARUCO-Marker ins Spiel: Diese können durch ihre geringe Komplexität schnell erfasst und verfolgt werden. Dabei ist es nicht nur möglich, die Position, sondern auch die Ausrichtung zu erfassen. Die Komplexität der Marker wird durch die Anzahl der nötigen unterschiedlichen Marker gesteuert, das heißt, die Anzahl der zu unterscheidenden Objekte gibt vor, wie komplex der Marker sein muss. Diese Minimalisierung sorgt dafür, dass mit der geringen Komplexität auch der Rechenaufwand gering gehalten wird und so auch in einem Bild mehrere Marker verfolgt werden können. Wie ARUCO-Marker generiert und erfasst werden, beschreibt Garrido-Jurado (vgl. Garrido-Jurado u. a. 2014). Die Erkennung verläuft in vier Schritten:

1. **Segmentierung:** Die deutlichsten Konturen werden erfasst, indem eine Kantenerkennung mittels lokaler Schwellwertbildung durchgeführt wird.
2. **Konturenextraktion und Filter:** Hier werden Konturen erfasst und dabei alle nicht rechteckigen Konturen entfernt. Dadurch bleiben nur die äußeren Ränder erhalten.
3. **Extraktion des Markercodes:** Im dritten Schritt wird der innere Bereich der Marker analysiert. Hierfür wird zunächst die perspektivische Projektion aufgelöst und ein Raster über das Rechteck gelegt. Anschließend wird ein optimaler Schwellwert ermittelt, um die Unterscheidung der Farben bei unterschiedlichen Beleuchtungsverhältnissen

zu ermöglichen. Je nachdem, ob ein Feld schwarz oder weiß ist, wird dieses im Raster als Null oder Eins markiert.

4. **Markeridentifikation und Fehlerkorrektur:** Die bekannten Marker werden in einem „Dictionary“ abgelegt. Mit diesem wird der erfasste Marker verglichen und die zugehörigen Daten abgerufen. Bekannte Marker sind z. B. ARUCO-Marker und QR-Codes (siehe Abbildung 2.4a und 2.4b).

(a) Mit OpenCV generierter ARUCO-Marker

(b) QR-Code mit Inhalt „Augmented Reality“

Abbildung 2.4.: ARUCO-Marker und QR-Code im Vergleich

2.3.1.2. Markerloses Tracking

Spezielle Marker, wie vorangegangen beschrieben, erfordern immer einen speziellen Aufdruck. Dieser sorgt zwar für eine gute Erkennung und zuverlässiges Tracking, stört aber gerade in den Printmedien den Gesamteindruck des Artikels. Für ein markerloses Tracking kann theoretisch jede beliebige Grafik als Referenz hinterlegt werden, wobei auch hier gilt, dass ausreichend Referenzpunkte für die Erkennung vorhanden sein müssen. In der Regel werden hierbei nicht nur Kanten, sondern auch Farben und Farbübergänge als Featurepunkte genutzt. Eine Erkennung funktioniert teilweise zwar auch mit schwachen Kontrasten, ist dann jedoch stark winkel- und beleuchtungsabhängig.

2.3.2. Laufzeitbasierte Verfahren

Laufzeitbasierte Verfahren messen die Signallaufzeiten zwischen Sender und Empfänger und errechnen daraus die aktuelle Position. Dabei kann es sich sowohl um optische als auch um akustische Signale handeln.
Ein Beispiel für diese Verfahren stammt aus dem VR-Bereich mit der HTC-Vive. Hier senden im Raum installierte „Lighthouses“ gepulste Infrarotlaser aus, die von der VR-Brille erfasst werden (vgl. HTC 2019). Der Computer ermittelt anschließend durch die Zeitdifferenz die

Position der Brille. Es handelt sich hierbei um ein laufzeitbasiertes Verfahren und es ermöglicht einen Trackingbereich von ca. 4.5 m x 4.5 m.
Ein besonders bekanntes Beispiel ist das „Global Positioning System“ (GPS). Ein GPS-Empfänger ist in fast jedem aktuellen Smartphone bereits verbaut. Aus den Signalen von mindestens drei Satelliten kann dann die Position errechnet werden. Da die Signale im Indoorbereich in der Regel zu schwach sind und die Position nur auf wenige Meter genau ist, eignet sich dieses Verfahren lediglich für grobes Tracking im Outdoorbereich.

2.3.3. Unterschiedliche Trackingprinzipien

Beim Tracking wird zwischen zwei verschiedenen Prinzipien unterschieden: Dem Inside-Out-Tracking und dem Outside-In-Tracking. *Inside-Out* bedeutet, dass das bewegte Objekt (z. B. Smartphone) die Positionsdaten selbst ermittelt, ohne Zuhilfenahme externer Sender, die die Position mitteilen. Das *Outside-In*-Tracking nutzt hingegen die Positionsdaten, die von externen Sendern bereitgestellt werden. Das zu trackende Objekt verfügt dabei über keine interne Sensorik.

2.3.4. Genauigkeit des Trackings

Die Genauigkeit des Trackings hängt von unterschiedlichen Einflussfaktoren ab. Bei Sensoren gilt grundsätzlich, dass jeder Messwert fehlerbehaftet ist. Werden die Daten mehrerer Sensoren verwendet, spricht man von einer Sensorfusion. Für das Tracking bietet sich ein solches Verfahren an, um die Messfehler einzelner Sensoren mit den Daten anderer zu korrigieren oder zu glätten und Abweichungen zu minimieren. Für die Sensorfusion gibt es unterschiedliche Techniken. Eine der bekanntesten Fusionstechniken ist das „Kalman-Filter“ (vgl. Kalman 1960), welches von Rudolf Kálmán im Jahre 1960 vorgestellt wurde. Das Filter nimmt normalverteilte Messfehler an und berechnet eine Prädiktion des nächsten Messwerts. Hierbei gilt: Je älter die Werte sind, desto weniger stark werden sie gewichtet. Sobald ein neuer Wert gemessen wird, erfolgt eine Korrekturphase. Dies funktioniert jedoch nur im linearen Zustandsraum. Das erweiterte Kalman-Filter (EKF) oder auch das neuere Unscented-Kalman-Filter (UKF) liefern Möglichkeiten, Werte auch im nichtlinearen Zustandsraum zu korrigieren. Eine Übersicht über die Verwendung der unterschiedlichen Kalman-Filter liefern Welch und Bishop (vgl. Welch und Bishop 2006).
Die Genauigkeit kann nicht nur durch die Korrektur mittels mehrerer interner Sensoren, sondern auch durch Hinzuziehen der Kameraaufnahmen oder externer Trackingsysteme verbessert werden.

2.3.5. Kalibrierung und Registrierung

Beim Tracking müssen unterschiedliche Koordinatensysteme betrachtet werden, damit virtuelle Objekte an der korrekten Position in passender Ausrichtung darstellbar sind. Der Tracker erfasst die Pose-Daten (Position und Orientierung) relativ zum Ursprung des Trackingsystems. Die Abbildung des Koordinatensystems des Trackers auf das des Renderers wird als Registrierung bezeichnet. Damit dies präzise ausgeführt werden kann, ist die genaue Kenntnis vieler Parameter notwendig. Die dazu notwendigen Schritte werden unter dem Begriff Kalibrierung zusammengefasst. Es handelt sich z. B. um die Kamerakalibrierung, wo die Linsenverzerrung korrigiert werden muss.
Soll ein Kamerabild annotiert werden, steht die Kamera nur selten exakt senkrecht über dem Objekt. Meist ist der Blickwinkel leicht geneigt, wodurch auch die Augmentierung an diesen Winkel angepasst werden muss. Durch den Blickwinkeleffekt wird eine Annotation, die bei senkrechter Betrachtung rechteckig ist, bei anderem Winkel trapezförmig verzerrt. Wie eine Kalibrierung mit Hinblick auf die Augmentierung aussehen kann, hat Stricker gezeigt (vgl. Stricker und N. Navab 1999).

Kalibrierung und Registrierung sind in modernen Toolkits integriert, wodurch programmtechnisch in der Regel keine größeren Anpassungen mehr vorgenommen werden müssen.

2.4. Aktuelle Einsatzszenarien der AR

AR-Anwendungen werden bereits in Unternehmen, aber auch bei Privatnutzern eingesetzt. Dabei gibt es viele unterschiedliche Einsatzmöglichkeiten. Beispielhaft werden hier die AR-Funktionen einiger Anwendungen erläutert.

2.4.1. Privatnutzer

Bei Endnutzern ist die AR bereits in unterschiedlichen Bereichen, meist in Form von Smartphoneanwendungen, verbreitet. Häufige Anwendungen sind im Gamingsektor zu finden, die Spielfelder mit Hilfe von AR auf Tischen positionieren oder virtuelle Objekte in die reale Umgebung einblenden. Ein bekanntes Beispiel ist Pokémon Go. In diesem Spiel sind virtuelle Wesen in der realen Umgebung versteckt und der Spieler kann diese einfangen (vgl. Niantic 2016).
Mit einem AR-Sternatlas (vgl. LIMITED 2019) lassen sich Objekte am Nachthimmel identifizieren, indem man das Smartphone in die gewünschte Richtung ausrichtet. Für einige Gebirgsregionen gibt es Anwendungen, die Berggipfel annotieren. Man richtet die Smartphonekamera auf eine Bergregion und die Bezeichnungen und Höhen der einzelnen Berge

werden eingeblendet (vgl. Peakfinder 2019).
Weitere Anwendungsbereiche existieren in Form von Bedienungsanleitungen, wo der Benutzer beispielsweise die Funktionen einzelner Tasten erklärt bekommt, ohne die entsprechende Passage in der Anleitung zu suchen. Beispiele hierfür gibt es von Automobilherstellern, die auf diese Weise die Fahrzeugbedienung erklären (vgl. Daimler 2017).
Primär aus dem Heimbereich gibt es Anwendungen von Möbelherstellern, die es einem erlauben, im realen Raum Möbel zu betrachten (vgl. Ikea 2017). Dadurch kann sich der Kunde eine Vorstellung davon machen, wie Möbel in der eigenen Wohnung aussehen, und so seine Einrichtung planen. In diesem Zusammenhang sind auch AR-Maßwerkzeuge hilfreich, mit denen man im Kamerabild ganze Räume vermessen kann. Für diesen Anwendungsfall gibt es unterschiedliche Apps, die vorgestellt werden (vgl. Martin 2019).

2.4.2. Industrie

Auch in der Industrie werden AR-Anwendungen bereits in verschiedenen Bereichen eingesetzt, um Arbeiten zu beschleunigen oder zu vereinfachen. Je nach Bereich gibt es diverse Unterstützungsmöglichkeiten.
Bei der Montage können Arbeiten annotiert werden, indem eine Einblendung der nächsten Montageschritte erfolgt. Dabei kann es sich z. B. um Montagepunkte oder benötigte Materialien handeln. Zusätzlich kann eine direkte Kontrolle von Arbeitsschritten erfolgen und so übergangenen oder fehlerhaften Montageschritten vorgebeugt werden. Anwendungsbeispiele liefert PTC (vgl. PTC 2019).
Durch die Verwendung von Livedaten in der AR lässt sich der Zustand einzelner Anlagenkomponenten darstellen, sofern eine Liveüberwachung erfolgt. Dies kann der Fehler- und Anomalieerkennung dienen und auf diesem Weg Wartungs- und Reparaturarbeiten beschleunigen.
Auch in der Ausbildung wird AR bereits eingesetzt. Zur Schweißausbildung gibt es von der Firma Soldamatic (vgl. Soldamatic 2019) Schweißsimulatoren, die mit Hilfe von AR unterschiedliche Schweißverfahren simulieren können und anschließend eine genaue Analyse der Schweißnaht ermöglichen. Fehler werden so schnell erkannt und können gezielt behoben werden. Durch den Simulator verringern sich Ausbildungszeiten und Ausbildungskosten, da kompaktere Aufbauten möglich sind und geringere Materialkosten entstehen.

2.5. Softwareentwicklung

Je nach Entwicklungsziel bieten sich unterschiedliche Entwicklungsumgebungen, Programmiersprachen und Frameworks an, die im Folgenden dargestellt werden.

2.5.1. Betriebssysteme

Da die Software häufig betriebsystemspezifisch entwickelt werden muss, lohnt es sich, zunächst einen Blick auf den Markt der mobilen Betriebssysteme zu werfen. Abbildung 2.5 zeigt die Entwicklung der Marktanteile an der mobilen Internetnutzung weltweit über einen Zeitraum von zehn Jahren. Die Statistik verdeutlicht, dass sich auf dem Markt zwei Betriebssysteme behaupten konnten. Diese sind Android von Google und iOS von Apple. Der Marktanteil anderer Betriebssysteme ist bis 2019 auf unter ein Prozent gesunken. Aus diesem Grund werden hier nur die beiden marktführenden Systeme genauer betrachtet.

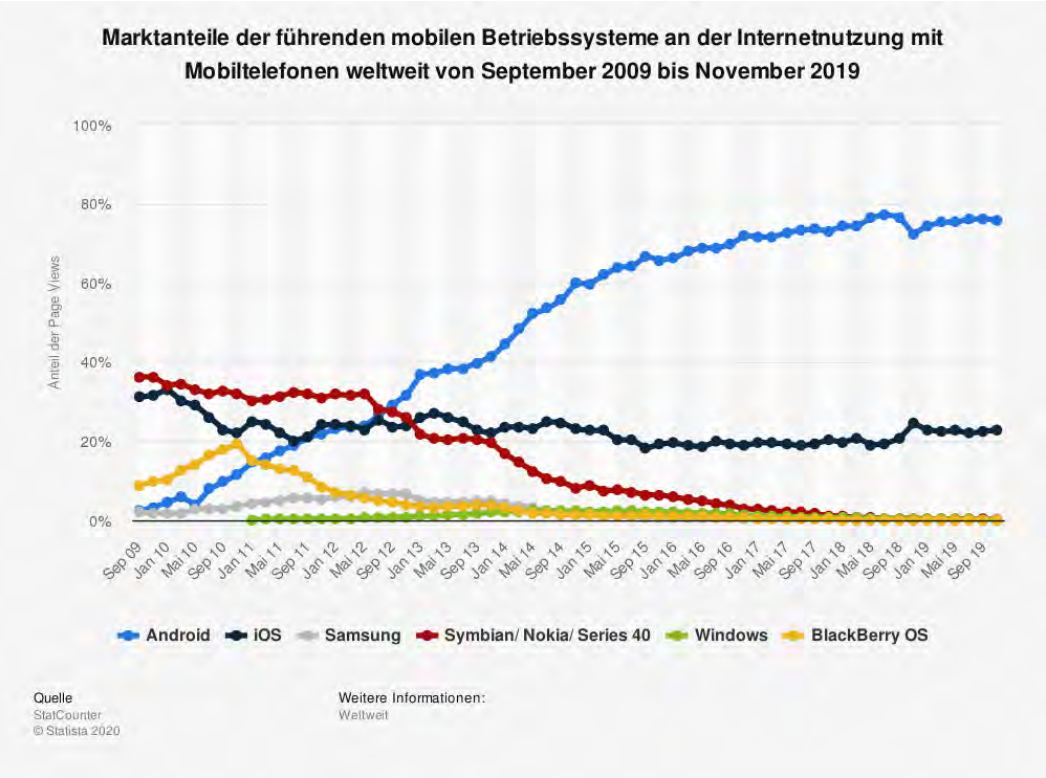


Abbildung 2.5.: Marktanteile der führenden mobilen Betriebssysteme an der Internetnutzung mit Mobiltelefonen weltweit von September 2009 bis November 2019 (Statcounter 2019)

Android wird von Google entwickelt und ist inzwischen in der zehnten Version erschienen (Stand Januar 2020). Hersteller von Android-Geräten versehen das System häufig mit eigenen Softwarepaketen, die das Design des Userinterfaces beeinflussen, während der grundsätzliche Aufbau in der Regel gleich bleibt. Das System ist eine freie Software unter der Apache-Lizenz (vgl. Android 2020) und basiert auf einem Linux-Kernel. Software für Android wird in der Regel mit der Programmiersprache Java entwickelt. Geschwindigkeitskritische

Bereiche greifen dabei häufig auf native C oder C++-Bibliotheken zurück.
Bei iOS handelt es sich um das mobile Betriebssystem von Apple, welches inzwischen in der 13. Version erschienen ist. Ein wesentlicher Unterschied ist, dass Apple andere Hardwarehersteller nicht lizenziert, sondern das System nur auf eigener Hardware einsetzt. Es basiert auf einem BSD-Linux-Kernel und Software für das System kann in den Sprachen Objective-C oder Swift entwickelt werden.

2.5.2. Programmiersprachen

Je nach Zielsystem werden unterschiedliche Programmiersprachen zur Anwendungsentwicklung verwendet. Es gibt eine große Zahl von Sprachen, nachfolgend werden daher nur jene vier vorgestellt, welche für die Entwicklung mobiler Anwendungen unter iOS und Android relevant sind.

- **Java** (vgl. Oracle 2019) ist eine objektorientierte Programmiersprache der Firma Sun Microsystems. Java Software läuft in einer virtualisierten Umgebung mit dem Ziel, diese plattformunabhängig auszuführen.
- **Kotlin** (vgl. Kotlin 2020) ist eine junge objektorientierte Programmiersprache für die Java Virtual Machine (JVM). Seit 2019 ist es die von Google präferierte Sprache für die Entwicklung von Android Apps.
- **C/C++/Objective-C** (vgl. Ritchie 1993) ist eine performante Programmiersprache aus den siebziger Jahren und wird heute noch viel (vor allem im Embedded-Systems-Bereich) genutzt.
- **Swift** (vgl. Apple 2019c) ist eine 2014 von Apple vorgestellte eigenentwickelte Programmiersprache zur Entwicklung von Anwendungen für Apple-Systeme. Swift ist eine multiparadigmatische Sprache, das heißt, sie greift die Ideen unterschiedlicher Sprachen auf.

2.5.3. Bibliotheken für die AR-Entwicklung

Für die unterschiedlichen Plattformen und Ausprägungen der AR gibt es eine Vielzahl unterschiedlicher Softwarebibliotheken für die Anwendungsentwicklung, die je nach Konzept verwendet werden. Nachfolgend werden die wichtigsten Bibliotheken kurz mit deren Funktionalitäten beschrieben.

- **ARKit** (vgl. Apple 2019a) und **ARCore** (vgl. Google 2019a) bilden die zentralen AR-Programmierschnittstellen von Apple bzw. Google und liefern die Basis für AR-Anwendungen. Sie liefern Funktionalitäten wie z. B. Tracking und Bilderkennung. Für das Tracking werden sowohl Sensorhardware als auch die Kamera verwendet.
- **OpenCV** (Open Source Computer Vision Library) (vgl. OpenCV 2019b) ist eine Bibliothek, die aus dem Bereich der Bildverarbeitung kommt und auf eine große Community inklusive großer Firmen zurückgreifen kann. Es sind viele Bildverarbeitungsalgorithmen implementiert, die z. B. für Tracking, Bilderkennung und Annotation verwendet werden können.
- **Vuforia** ist ein kommerzielles AR-Framework der Firma PTC und wird in einer Vielzahl von AR-Anwendungen in der Industrie verwendet. Es zeichnet sich durch gutes Tracking und umfassende Plattformunterstützung aus (vgl. PTC 2019).

Neben den genannten großen Vertretern gibt es weitere AR-Bibliotheken wie z. B. EasyAR (vgl. EasyAR 2019).

2.5.4. Bild- und Objekterkennung

In kamerabasierten AR-Anwendungen kann Bild- und Objekterkennung zum Einsatz kommen, um Lokalisierung und Registrierung durchzuführen. Dies kann softwareseitig auf unterschiedliche Arten durchgeführt werden. Neben Algorithmen aus der Bildverarbeitung stehen oft Möglichkeiten des maschinellen Lernens (siehe Kapitel 2.6) zur Verfügung.
Bilderkennung lässt sich in die drei Bereiche Bildklassifikation, Objektlokalisierung und Objekterkennung gliedern (vgl. Joshi 2015). Die Bildklassifikation liefert ein Label für das eingegebene Bild. Im Schritt der Objektlokalisierung werden Bounding Boxes um die gefundenen Objekte gezeichnet. Von einer Objekterkennung wird gesprochen, wenn eine Kombination aus beiden Schritten vorliegt (vgl. Brownlee 2019). Bei Objekterkennung handelt es sich um ein Klassifikationsproblem mehrerer Objekte in einem Bild.

2.5.4.1. Referenzbild

In Abschnitt 2.3.1.2 wurde bereits erwähnt, dass die Bilderkennung markerlos, anhand eines Referenzbildes erfolgen kann. Zusätzlich zu bereits genannten Histogramm- und Kontrasteigenschaften sollte das Referenzbild unverzerrt fotografiert sein, um Schwierigkeiten durch Trapezverzerrung (Abschnitt 2.3.5) vorzubeugen. Bei Verwendung eines Referenzbildes werden über Algorithmen die besonderen Merkmale des Bildes identifiziert und mit dem

zu untersuchenden Bild (häufig Kamerastream) abgeglichen. Anschließend erfolgt eine Klassifizierung, um zu entscheiden, ob es sich um das gesuchte Muster aus dem Referenzbild handelt.

2.5.4.2. Algorithmen zur Merkmalsextraktion

Ein einfacher Abgleich eines Referenzbildausschnitts ist stark abhängig von Größe, Ausrichtung und Beleuchtung, wodurch die Identifikation nur schwer möglich ist. Durch Algorithmen der Bildverarbeitung können Merkmale aus Referenzbildern extrahiert werden, die zur Identifikation von Objekten und Bildern dienen und die genannten Probleme beheben. Drei mögliche Algorithmen, die diese Funktionen bieten, werden nachfolgend kurz vorgestellt (vgl. OpenCV 2019a).

- **SIFT** (Scale-Invariant Feature Transform) Der Algorithmus durchläuft fünf Schritte:
  1. *Scale-space Extrema Detection:* Lokale Extrema erkennen.
  2. *Keypoint Localization:* Keypoints werden untersucht und mit einem Grenzwert verglichen. Keypoints können z. B. Ecken oder Kanten sein.
  3. *Orientation Assignment:* Dem Keypoint einen Winkel zuweisen, um für Rotationsinvarianz zu sorgen.
  4. *Keypoint Descriptor:* Betrachtet die Umgebung des Merkmals und speichert mehrere Werte in einem Vektor, um den Punkt zu beschreiben.
  5. *Keypoint Matching:* Vergleicht mit Hilfe des Nearest-Neighbour-Verfahrens die gefundenen Keypoints mit den Referenzen.
- **SURF** (Speeded-Up Robust Features) ist eine schnellere Version des SIFT-Algorithmus.
- **ORB** (Oriented FAST and Rotated BRIEF) besteht aus Keypoint Detector und Deskriptor und ist schneller als SIFT und SURF.

2.5.5. Datenbanken

In der Softwareentwicklung werden hauptsächlich sechs verschiedene Datenbankmodelle verwendet (vgl. Jarosch 2016):

- **Hierarchisches Datenbankmodell:** Anfragen gehen von einem einzigen Objekttyp aus. Es ist das älteste Modell und in einer Baumstruktur aufgebaut. Bei komplizierten Zusammenhängen stößt dieses Modell allerdings schnell an seine Grenzen und wird daher heute nur noch selten verwendet.
- **Netzwerkdatenbankmodell:** Die Daten in diesem Modell können durch ein beliebiges Netzwerk miteinander verbunden sein. Dadurch kann die Suche von unterschiedlichen Objekten ausgehen und m:n-Beziehungen werden möglich.
- **Relationales Datenbankmodell:** Dieses Datenbankmodell basiert auf der relationalen Algebra. Daten werden in Tabellen gespeichert und der Datenzugriff erfolgt meistens mit der Datenbanksprache SQL (Structured Query Language). Relationale Datenbanken sind weit verbreitet und Implementationen sind beispielsweise: „SQLite“, „PostgreSQL“, „Microsoft SQL“ oder „MySQL“.
- **Objektrelationales Datenbankmodell:** Das Modell ist hier um das Konzept der Vererbung erweitert und entspricht sonst dem relationalen Datenbankmodell.
- **Objektorientiertes Datenbankmodell:** Daten werden als Objekte verwaltet und durch Attribute beschrieben. Zugriffsfunktionen werden in den Objekten abgelegt.
- **Dokumentenorientiertes Datenbankmodell:** Hierbei handelt es sich um eine Ausprägung von „NoSQL“-Datenbanken (Not only SQL). Das Datenmodell ist nicht relational und unterstützt verteilte Webanwendungen, dabei gibt es kein bestimmtes Datenbankschema und die Daten können je nach Typ der Datenbank unterschiedlich abgelegt werden. Möglich sind hier Schlüssel-Wertpaare (Key/Value Store), in Spalten oder Spaltenfamilien (Column Store), in Dokumentspeichern (Document Store) oder in Graphen (Graph Database) (vgl. Meier 2018).

2.5.5.1. Replikation

Die Replikation einer Datenbank bezeichnet den Prozess der mehrfachen Speicherung. Redundante Speicherung von Daten hat je nach Implementierung mehrere Vorteile. Die Verfügbarkeit der Daten wird erhöht. Selbst bei einem Totalausfall eines Servers wird die Wiederherstellung (Recovery) durch den oder die Spiegelserver ermöglicht. Gleichzeitig kann ein anderer Server die Funktion übernehmen und Datenbankabfragen bearbeiten. Eine ähnliche Funktionalität gibt es bei Datenbanken, die auf Smartphones synchronisiert werden. Da die Netzwerkverbindung zur zentralen Datenbank nicht permanent garantiert werden kann, wird auf dem Gerät mit einer lokalen Datenbank gearbeitet, die bei erneuter Verbindung mit der zentral gespeicherten Datenbank synchronisiert wird.
Auch aus Performancegründen kann die Replikation auf mehreren Servern sinnvoll sein. Bei Lesezugriffen kann die Last auf mehrere Server gleichmäßig verteilt und die Antwortzeit

reduziert werden. Auch bei einer lokalen Kopie der Datenbank handelt es sich um eine Möglichkeit der Lastverteilung.
Es gibt zwei unterschiedliche Verfahren der Replikation:

- **Synchrone Replikation:** Die Replikation ist erst erfolgreich, wenn diese auf allen Replikaten erfolgreich war.
- **Asynchrone Replikation:** Es ist erlaubt, dass sich Datenbankobjekte unterscheiden.

Je nach Architektur kann es bei der Replikation zu Konflikten kommen, die gelöst werden müssen. Für die Konfliktlösung gibt es unterschiedliche Optionen, die auch je nach Datenbanktyp unterschiedlich sind. Bekannte Optionen sind das Master-Slave-Verfahren, welches sich für mobile Anwendungen anbietet, oder die Identifikation des aktuellen Standes anhand eines Zeitstempels (vgl. Meier und Kaufmann 2019).

2.5.5.2. Dateiablage

Datenbanken sind primär auf die Speicherung von Werten ausgelegt und weniger für die Ablage ganzer Dateien. Diese werden in der Regel auf Fileservern abgelegt. Dennoch bieten viele Datenbanken die Möglichkeit, Dateien abzulegen. Der große Unterschied zwischen diesen Ablageoptionen ist die Ablageart. Auf Fileservern werden die Dateien in einem Dateisystem abgelegt, während bei der Ablage in einer Datenbank die Dateien z. B. in „Binary Large Objects“ (BLOB) oder Dokumenten gespeichert werden. Theoretisch lassen sich Dateien also auf beiden Wegen speichern. Unterschiede ergeben sich hier in Performance, Administrationsaufwand und Infrastrukturanforderungen. Welche Art sinnvoll ist, hängt somit von der Dateigröße und dem Anwendungsfall ab.

2.5.5.3. Content Management

Da es bei größeren Datenbanken unübersichtlich und daher auch fehleranfällig ist, direkt z. B. mit SQL-Befehlen die Datenbank zu bearbeiten, werden Content-Management-Systeme (CMS) verwendet. Ein bekannter Vertreter ist Wordpress[1]. CMS bieten eine grafische Oberfläche und sorgen dafür, dass Einträge bzw. Änderungen korrekt in der Datenbank abgelegt werden. Die Oberfläche kann dabei so implementiert werden, dass in einem Formular Daten eingegeben und automatisch in Datenbankbefehle umgewandelt werden, die die Eingaben in die Datenbank übertragen. Je nach Art und Umfang der Datenbank können unterschiedliche CMS geeignet oder durch Anbindung an andere Unternehmenssysteme vorgegeben sein.

[1] Wordpress, https://de.wordpress.com/, abgerufen am 10.06.2020

2.6. Machine Learning

Machine Learning (ML) ist ein Oberbegriff für Verfahren, in denen künstliche Systeme aus Daten und Erfahrungen Wissen generieren.

2.6.1. Grundlagen des Machine Learning

In diesem Abschnitt werden Grundlagen geliefert, auf dem verschieden ML-Verfahren aufbauen.

2.6.1.1. Künstliche Neuronen

Mit künstlichen Neuronen wird versucht, die Funktionsweise eines Gehirns nachzubilden, um eine künstliche Intelligenz zu schaffen. Bereits 1943 wurde ein erstes Konzept über eine vereinfachte Hirnzelle, dem sogenannten MCP-Neuron, veröffentlicht (vgl. McCulloch und Pitts 1943). Im Jahre 1958 wurde ein Algorithmus vorgestellt, der automatisch passende Gewichtungskoeffizienten ermittelt (vgl. Rosenblatt 1958). Anhand dieser Koeffizienten entscheidet sich, ob ein Neuron feuert oder nicht. Es handelt sich dabei um eine binäre Klassifizierung.
Als Aktivierungsfunktion $\phi(z)$ wird beim Perzeptron-Algorithmus eine Sprungfunktion (Heaviside-Funktion) verwendet:

$$\phi(z) = \begin{cases} 1 & \text{wenn } z \geq 0 \\ -1 & \text{sonst} \end{cases}$$

Wie ein Perzeptron nach Rosenblatt lernt, gliedert sich in zwei Schritte:

1. Gewichtungen werden mit 0 oder kleinen Zufallswerten initialisiert
2. Bei jedem Trainingsobjekt x werden zwei Schritte ausgeführt:

   Berechnung des Ausgabewertes $\hat{y}$

   Aktualisierung der Gewichtungen

Durch die Aktivierungsfunktion wird eine binäre Ausgabe erzeugt, die als Vorhersage der Klasse des Objekts dient (Schritt eins des Trainings). Beim Training wird der Wert dieser Ausgabe genutzt, um Fehler festzustellen und die Gewichtungen zu aktualisieren (Schritt zwei des Trainings).
Das Perzeptron ist nicht die einzige Art künstlicher Neuronen. Ein weiteres bekanntes Modell ist beispielsweise das adaptive lineare Neuron (Adaline) (vgl. Widrow u. a. 1960). Im

Gegensatz zu Rosenblatts Perzeptron wird eine lineare Aktivierungsfunktion anstatt einer Sprungfunktion verwendet. Dieser Algorithmus dient auch als Grundlage für fortgeschrittenere Lernalgorithmen, wie z. B. Support-Vector-Machines (SVM) oder Regressionsmodelle.

2.6.1.2. Konvolutionale Neuronale Netze

Konvolutionale neuronale Netze (Convolutional Neural Networks, CNNs) wurden zu Beginn der 1990er Jahre entwickelt (vgl. LeCun u. a. 1990) und orientieren sich an der Funktionsweise des visuellen Kortex des menschlichen Gehirns. Dieser Bereich ist unter anderem für die Objekterkennung zuständig. Angelehnt an diese menschliche Fähigkeit werden CNNs für maschinelles Sehen und zur Bildklassifizierung eingesetzt.
CNNs sind über mehrere Schichten (Layer) aufgebaut und in der Lage, aus Rohdaten Merkmale zu extrahieren. Bei der Verarbeitung von zweidimensionalen Bildern werden in den ersten Schichten Low-level-Merkmale (z. B. Kanten) extrahiert. In den tieferen Schichten werden diese zu High-level-Merkmalen (z. B. Objektformen) kombiniert. Bei den Schichten wird zwischen Convolutional Layer und Pooling Layer unterschieden. Die Aktivität jedes Neurons im Convolutional Layer wird durch eine diskrete Faltung berechnet, wobei jeder Bildausschnitt mit einem Filterkernel betrachtet wird, sodass auch benachbarte Neuronen beeinflusst werden. Jedes Neuron reagiert auf die, entsprechend des Filterkernels, lokale Umgebung des vorangegangenen Layers. Anschließend wird über eine Aktivierungsfunktion $f(x) = max(0, x)$ (Rectifier) der Output berechnet.
Nach den Berechnungen des Convolutional Layers folgt ein Pooling Layer, in dem überflüssige Informationen verworfen werden. Häufig wird hierfür das Max-Pooling-Verfahren verwendet. Dabei bleibt nur das aktivste Neuron im Filterbereich erhalten, wodurch sich im Falle eines 2*2-Filters eine Datenreduktion von 75% ergibt. Dies bietet mehrere Vorteile:

- lokale Invarianz
- Reduktion der Merkmalzahl $\rightarrow$ schnellere Berechnungen und geringerer Platzbedarf $\rightarrow$ tiefere Netze möglich

Das Netzwerk schließt letztendlich mit mindestens einem vollständig vernetzten Layer ab, welches der Architektur eines mehrschichtigen Perzeptrons entspricht. In der Regel hat der letzte Layer so viele Neuronen, wie es Klassen gibt.

2.6.1.3. Diskrete Faltung

Wie vorangegangen beschrieben, basieren CNNs mathematisch auf der diskreten Faltung. Bei zweidimensionalen Daten handelt es sich um die Faltung zweier Matrizen. Aus einer

Matrix **X** und einer Filtermatrix **W** ergibt sich die Matrix $\mathbf{Y} = \mathbf{X} * \mathbf{W}$ nach folgender Definition:

$$\mathbf{Y} = \mathbf{X} * \mathbf{W} \rightarrow \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} \mathbf{X}[i - k_1, j - k_2]\mathbf{W}[k_1, k_2]$$

Im Machine Learning gibt es stets endliche Merkmalsmatrizen. Damit das Filter die Merkmale am Rand einer Matrix gleich bewertet, werden rundherum Nullen angehängt. Dieses Verfahren wird „Zero-Padding“ genannt. Die Grenzen sind hier in der Theorie unendlich, in der Praxis werden für Berechnungen jedoch endliche Grenzen benötigt. Hierfür gibt es drei Varianten des Paddings:

- Full-Padding: Erhöht Anzahl der Ausgabedimensionen
- Same-Padding: Ein- und Ausgabegröße stimmen überein
- Valid-Padding: kein Padding

Bei CNNs wird meistens das Same-Padding angewandt und die Dimension in den Poolingschichten verringert (vgl. Raschka und Mirjalili 2018). Die Größe der Faltungsausgabe lässt sich im eindimensionalen Bereich aus Größe des Eingabevektors n, Filtergröße m, Padding p und der Schrittweite s berechnen:

$$o = \lfloor \frac{n + 2p - m}{s} \rfloor + 1$$

2.6.2. Lernverfahren des Machine Learning

Es gibt drei Arten von ML, wie Systeme lernen können: überwachtes, unüberwachtes und verstärkendes Lernen. Jede dieser Arten ist für unterschiedliche Anwendungen geeignet und wird nachfolgend kurz vorgestellt (vgl. Otte 2019).

2.6.2.1. Überwachtes Lernen

Ziel des überwachten Lernens ist es, mit Hilfe von gekennzeichneten Trainingsdaten ein Modell zu erlernen, welches anschließend auf unbekannten Daten Vorhersagen treffen kann. Die Trainingsdaten werden mit Labels versehen und nur diese können auch als Ausgabewerte auftreten. Ein Teilbereich des überwachten Lernens ist die Klassifizierung, wobei ein Label einer Klasse entspricht. Bei einer Unterscheidung zwischen zwei Klassen spricht man von einer binären Klassifizierung. Binäre Klassifikatoren werden unter anderem in Spamfiltern eingesetzt und unterscheiden zwischen Spam und nicht Spam. Gibt es mehr als zwei

Klassen, spricht man von einer Mehrfachklassifizierung, wozu z. B. die Erkennung unterschiedlicher Buchstaben zählt.
Für das Training einer Objekterkennung ist dieses Verfahren besonders relevant. Hier werden Objekte in Bilddaten gelabelt, wobei jedes markierte Objekt einer Klasse entspricht. Anhand dieser Daten kann das Modell trainiert werden, um diese Klassen in neuen Bildern zu erkennen.

Trainingsdaten

Wie gut das System lernen kann, hängt von den Trainingsdaten ab. Diese Daten müssen zunächst generiert und ggf. aufbereitet werden. Eine Datenvorverarbeitung kann die Qualität der Trainingsdaten signifikant verbessern.
Durch eine Vorverarbeitung können unvollständige Werte aus der Datensammlung entfernt werden, die den Lernprozess verlangsamen oder gar behindern würden. Mit Hilfe von Schätzern können außerdem fehlende Werte ergänzt werden. Neben numerischen Werten können in einem Datensatz auch nominale oder ordinale Merkmale auftauchen. In der Vorverarbeitung können Klassen, die bisher als Strings vorhanden waren, durch einen ganzzahligen Wert kodiert werden. Dies ist erforderlich, da ML-Bibliotheken häufig nur Ganzzahlen als Klassenbezeichnungen erlauben.

Aufteilung in Trainings- und Testdaten

Der gesamte Datensatz kann in zwei Teilmengen unterteilt werden: Trainings- und Testdaten. Dabei werden die Trainingsdaten zum Lernen des Systems genutzt. Anschließend wird das trainierte System mit den Testdaten geprüft, um herauszufinden, wie gut das System arbeitet. Bei der Aufteilung des Datensatzes ist auf das Verhältnis der Teilmengen Test- und Trainingsdaten zu achten. In den Testdaten können Informationen enthalten sein, die dann beim Training fehlen und so die Performance des Systems verschlechtern. Häufig verwendete Aufteilungen liegen im Bereich von 70% - 80% Trainingsdaten und 20% - 30% Testdaten (vgl. Bronshtein 2020).
Um ein neuronales Netz für die Objekterkennung zu trainieren, benötigt man als Trainingsdaten Bilder, die das zu erkennende Objekt aus unterschiedlichen Blickwinkeln zeigen. Durch Augmentierungen wie Bildrauschen oder Beleuchtungsänderungen können die Datensätze künstlich erweitert werden (vgl. Taylor und Nitschke 2017). Auch verschiedene Rotationen oder Ausschnitte können die Trainingsdaten ergänzen. Wenn in den Bilddaten konstante Hintergründe enthalten sind, besteht die Gefahr, dass sich das Netz nicht nur am gewünschten Objekt trainiert, sondern auch den Hintergrund betrachtet. Daher kann es passieren,

dass die Objekterkennung nicht durch das Objekt selbst, sondern durch den Hintergrund ausgelöst wird. NVIDIA hat dieses Problem mit synthetischen Daten gelöst und bessere Ergebnisse erzielt, als wenn nur reale Daten verwendet wurden (vgl. Tremblay u. a. 2018).

Struktur von Trainingsdaten für die Objekterkennung

Die Trainingsdaten für eine Objekterkennung müssen folgende Informationen enthalten: Welches Objekt befindet sich an welcher Position in welchem Bild. Diese Informationen werden durch „Bounding Boxes“ dargestellt. Hierbei handelt es sich um Rechtecke, die Objekte im Bild umgeben und so deren Position markieren. Die Position wird durch eine X-/Y-Ursprungskoordinate sowie Höhe und Breite eindeutig definiert. Jedem dieser Rechtecke wird zusätzlich eine Klasse zugewiesen, die aussagt, um welches Objekt es sich handelt (siehe Abbildung 2.6).

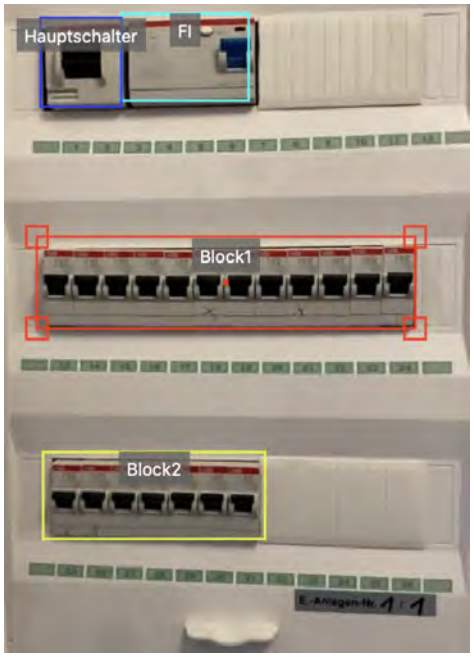


Abbildung 2.6.: Beispiel: Bild mit markierten Objekten

2.6.2.2. Unüberwachtes Lernen

Während beim überwachten oder verstärkenden Lernen das richtige Ergebnis bekannt ist, versucht das System beim unüberwachten Lernen, Strukturen in Datensätzen zu erkennen,

um Informationen zu extrahieren. Dabei gibt es keine Zielvariable oder Belohnungsfunktion. Ein wichtiger Einsatzbereich des unüberwachten Lernens ist das Clustering. In unstrukturierten Daten werden Muster gesucht, um eine Klassifizierung vorzunehmen. Beispielsweise kann es sich um Gruppen von X-/Y-Koordinaten handeln, die sich an Positionen in einem Koordinatensystem konzentrieren.
Ein weiterer Teilbereich ist die Dimensionsreduktion. Diese Variante wird häufig eingesetzt, um an Messdaten, die oftmals hochdimensional sind, eine Vorverarbeitung vorzunehmen. Auf diesem Weg können die Daten auch von Rauschen befreit werden.

2.6.2.3. Verstärkendes Lernen

Beim verstärkenden Lernen ist das Ziel, ein System zu entwickeln, welches durch Interaktion mit der Umgebung seine Leistung verbessert. Damit ein System durch Interaktion lernen kann, muss eine Belohnungsfunktion existieren, um zu erkennen, wie erfolgreich es gehandelt hat. Im Allgemeinen versucht das System seine Belohnung zu maximieren, indem jedem Zustand positive oder negative Bewertungen zugeordnet werden (vgl. Raschka und Mirjalili 2018).

2.6.3. Machine-Learning-Verfahren für die Objekterkennung

Im Bereich der Objekterkennung mit Machine Learning gibt es verschiedene Ansätze die Netze zu gestalten. Nachfolgend werden die drei wichtigsten Vertreter vorgestellt. Die Position gefundener Objekte wird durch Bounding Boxes dargestellt.

2.6.3.1. Parameter zur eindeutigen Objektidentifikation

Damit ein Objekt eindeutig auf einem Bild markiert werden kann, sind fünf Parameter nötig:

- Name der Klasse (Wie heißt das Objekt?)
- X-Koordinate der oberen, linken Ecke
- Y-Koordinate der oberen, linken Ecke
- Breite der Bounding Box
- Höhe der Bounding Box

2.6.3.2. Region-based Convolutional Neural Networks (R-CNN)

Die Objekterkennung durch R-CNNs arbeitet in drei Schritten (siehe Abbildung 2.7). Im ersten Schritt wird eine selektive Suche auf dem Eingangsbild ausgeführt, die Bereiche identifiziert, die Objekte enthalten können. Diese Bereiche werden als „region proposals“ bezeichnet. In einem zweiten Schritt wird die Größe der region proposals an ein CNN angepasst. Dieses CNN extrahiert Merkmale und klassifiziert die Objekte anhand der gefundenen Merkmale. Im dritten Schritt werden mit den region proposals Bounding Boxes durch eine SVM berechnet, die mit Merkmalen aus dem CNN trainiert wird (vgl. Girshick u. a. 2014).

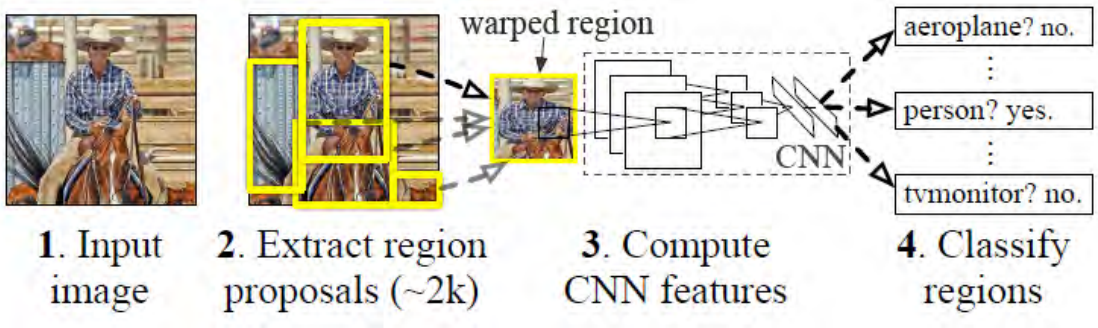


Abbildung 2.7.: Funktionsweise von R-CNNs (Girshick u. a. 2014)

Vom R-CNN gibt es Varianten, die Optimierungen enthalten. Diese betreffen u. a. die Geschwindigkeit einer oder mehrerer Schritte.
Beim „Fast R-CNN“ wird das ganze Bild im CNN betrachtet anstatt der einzelnen region proposals. Dies verbessert die Effizienz, da überschneidende Bereiche nicht mehrfach berechnet werden (vgl. Girshick 2015).
Eine weitere Optimierung ist der „Faster R-CNN“. Bei diesem Detektor ist die Suche nach den region proposals nicht mit einem externen Algorithmus implementiert, sondern erfolgt direkt im Netz durch das Region Proposal Network (vgl. Ren u. a. 2016). Das weitere Design entspricht dem des Fast R-CNN.

2.6.3.3. You Only Look Once (YOLO)

Wie der Name bereits sagt, durchläuft das Eingangsbild nur einmal das neuronale Netz. YOLO unterteilt das Bild in ein Raster der Größe S x S und berechnet für jedes Element Bounding Boxes in Verbindung mit einer zugehörigen Wahrscheinlichkeit. Den meisten dieser Boxen ist nur eine niedrige Wahrscheinlichkeit zugeordnet, die unter einer definierten Schwelle liegt. Diese Boxen werden verworfen und übrig bleiben diejenigen, die mit großer Wahrscheinlichkeit die Objekte umranden. Durch diese einmalige Berechnung ist YOLO sehr

schnell und kann in Echtzeit ausgeführt werden. Da YOLO nur eine Klasse je Rasterelement vorhersagt, kann es Probleme bei der Erkennung von sehr kleinen Objekten geben (vgl. Redmon, Divvala u. a. 2016).

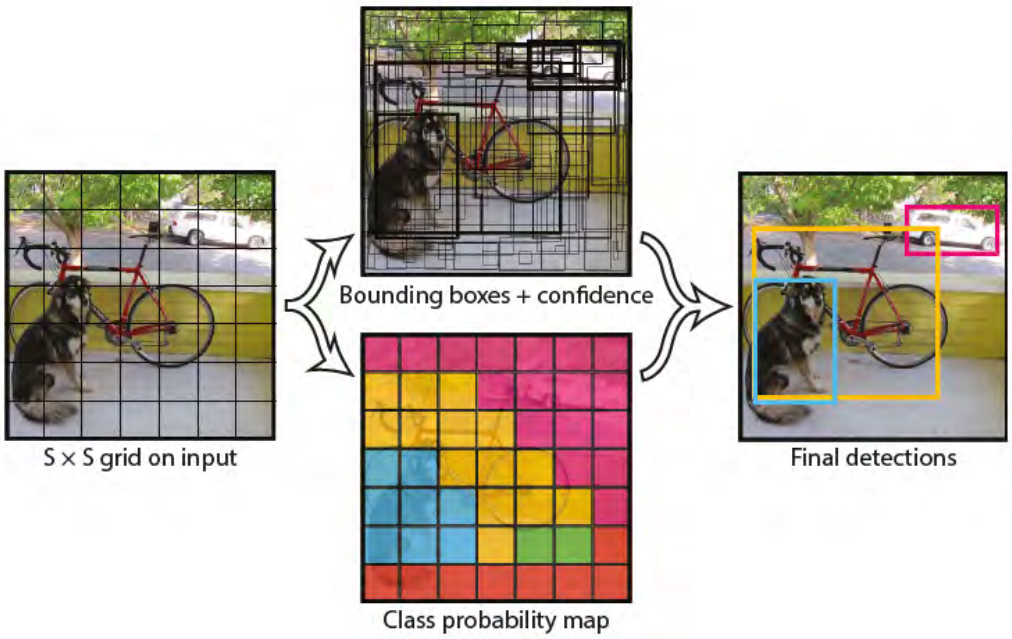


Abbildung 2.8.: YOLO Object Detector (Redmon, Divvala u. a. 2016)

2.6.3.4. Single Shot Detector (SSD)

Während R-CNNs aus Proposal Network und einem Klassifizierer bestehen, arbeiten SSDs einschichtig ohne Proposal Network. Zunächst werden Merkmale aus den Bild extrahiert. An diese „Feature Map“ sind mehrere „Convolution Layer“ angeschlossen, die für die Objekterkennung zuständig sind. In diesen Schichten werden kleinere „Convolution Filter“ auf die Feature Map angewandt und für jeden Bereich die gefundenen Klassen und Wahrscheinlichkeiten ermittelt. Das Netz arbeitet mit Standardboxen von verschiedenen Größen und Seitenverhältnissen. Die ermittelten Bounding Boxes werden relativ zu jeder Standardbox angegeben, die wiederum relativ zu der jeweiligen Feature-Map-Position angegeben werden. Zuletzt wird eine Non-Max-Funktion angewendet, die geringe Wahrscheinlichkeiten unterdrückt (siehe Abbildung 2.10).

Der SSD stellt eine Balance zwischen Genauigkeit und Geschwindigkeit dar. Jedes Bild wird einmal durch ein Faltungsnetz betrachtet und eine Feature Map wird generiert. Anhand eines

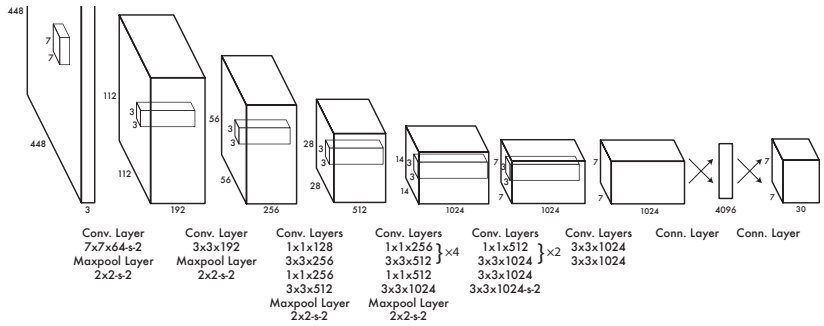


Abbildung 2.9.: Aufbau des neuronalen Netzes beim YOLO Object Detector (Redmon, Divvala u. a. 2016)

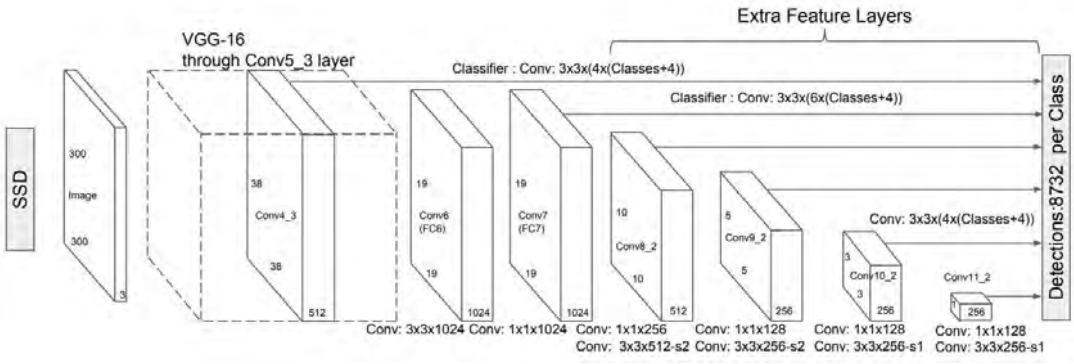


Abbildung 2.10.: Aufbau eines Single Shot Detectors (Liu u. a. 2016)

3x3 Faltungskernels, der über die Feature Map bewegt wird, werden Klassen und Bounding Boxes berechnet (vgl. Liu u. a. 2016).

2.6.4. Bewertung von trainierten Modellen

Nach dem Training eines ML-Modells muss festgestellt werden, wie gut dieses arbeitet, um die Funktionalität nachzuweisen. Im Bereich der Objekterkennung handelt es sich um zwei Parameter, die passend erkannt werden müssen:

1. Klassifikation: Welches Objekt wurde erkannt?
2. Lokalisation: Wo wurde das Objekt gefunden?

Um eine Bewertung vornehmen zu können, müssen neben Trainingsdaten auch Testdaten zur Verfügung stehen. Diese Testdaten entsprechen der „Ground Truth“ und werden mit den erkannten Daten verglichen. Häufig handelt es sich hierbei um einen kleinen Prozentsatz der Trainingsdaten. Gängige Bewertungsverfahren werden nachfolgend vorgestellt:

2.6.4.1. Confusion Matrix

Wie gut ein trainiertes System arbeitet, spiegelt sich in den Werten „Precision“ und „Recall“ wider, aus denen sich eine Confusion-Matrix (Abbildung 2.11) erstellen lässt. In dieser Matrix werden vier Werte abgebildet:

- True Positives (TP): Klasse korrekt erkannt
- False Positives (FP): Klasse fälschlicherweise erkannt
- True Negatives (TN): Klasse korrekterweise nicht gefunden
- False Negatives (FN): Klasse fälschlicherweise nicht gefunden

Mit diesen Daten können die Werte für Precision und Recall nach folgenden Formeln berechnet werden:

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

Precision bezeichnet den Anteil der korrekten Erkennungen einer Klasse bezogen auf die Gesamtzahl der Erkennungen dieser Klasse.

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

Recall bezeichnet den Anteil der korrekten Erkennungen einer Klasse bezogen auf das gesamte Vorkommen einer Klasse, das heißt, inklusive Ground Truth.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Abbildung 2.11.: Confusion Matrix

Diese Werte bilden die Grundlage für weitere Metriken wie die Average Precision (AP).

2.6.4.2. Pixel Accuracy

Bei diesem einfachen Verfahren werden einzelne Pixel der Bilder betrachtet und deren Klassen verglichen. Als Ergebnis erhält man einen Prozentwert, wie viele Pixel korrekt klassifiziert wurden. Dieser Wert ist allerdings nur bedingt aussagekräftig, wenn eine Klasse das Bild dominiert. Hierbei kann es sich auch um den Hintergrund handeln. Wird eine Genauigkeit von 90% berechnet, kann dies bedeuten, dass 90% der Pixel korrekt dem Hintergrund zugeordnet wurden. Über die zu erkennenden Objekte trifft dieses Verfahren daher kaum eine Aussage (vgl. Tiu 2019).

2.6.4.3. Intersection Over Union (IoU, Jaccard-Index)

IoU, oder auch Jaccard-Index genannt, ist ein gängiges Verfahren zur Bewertung, wie gut ein Objekt in Bildern erkannt wurde. Die IoU berechnet sich bilderweise für jede Klasse nach folgender Formel:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{2.3}$$

Als Area of Overlap ist dabei die überlappende Fläche zwischen der vom Modell erkannten Bounding Box und der Ground Truth definiert. Bei der Area of Union handelt es sich um die Vereinigung der Flächen von Bounding Box und Ground Truth. Der IoU wird je Klasse berechnet. Weiterhin kann der durchschnittliche IoU für jedes Bild angegeben werden, indem der Durchschnittswert der IoU aller Klassen im Bild berechnet wird. Ab einem Wert von $0,5$ wird von einer guten Erkennung gesprochen (vgl. ebd.).

2.6.4.4. Dice Coefficient (F1-Score)

Der Dice-Coefficient korreliert positiv mit der IoU. Dies bedeutet, dass bei der Betrachtung eines Bildes beide Metriken auch die gleiche Objekterkennung als „besser“ einstufen werden. Ein Unterschied ergibt sich, wenn man einen Datensatz betrachtet. Dabei vergrößert sich der Unterschied beider Metriken, da bei der Berechnung der IoU über einen Datensatz einzelne Erkennungsfehler stärker ins Gewicht fallen. Der IoU trifft daher eher eine Aussage

über den schlechtesten Performancefall, während der F1-Score als gewichteter Durchschnitt von Precision und Recall eher die durchschnittliche Performance misst (vgl. Tiu 2019).

$$F1 = \frac{2 * Area\ of\ Overlap \quad \square}{\sum Pixel\ of\ Areas(\square + \square)} \tag{2.4}$$

2.6.4.5. Average Precision (AP)

Im Bereich der Objekterkennung werden als Metrik Werte der Average Precision angegeben. Die Berechnung der AP bezieht sich dabei auf den IoU, der als Schwellwert dient. Dabei gibt es verschiedene Vorgehen, den Grenzwert für die Berechnung der AP festzulegen. Da ab einer IoU von $0,5$ von einer guten Erkennung gesprochen wird, werden Erkennungen mit einer $IoU >= 0,5$ also korrekt angenommen. AP-Werte mit dieser Grenze werden auch als AP50% angegeben. Mit dieser Schwelle wird eine Precision-Recall-Kurve $p(r)$ generiert. Die AP entspricht dann der Fläche unter dieser Kurve.

$$AP = \int_0^1 p(r)dr \tag{2.5}$$

Diese Berechnungsvariante ist gängig für Forschungen auf dem „PASCAL VOC“-Datensatz (vgl. Everingham u. a. 2015) und wird vor allem verwendet, wenn eine präzise Lokalisierung nicht notwendig ist. Durch die Betrachtung mit einer IoU kann diese Metrik nicht zwischen einem präzisen Modell und einem ungenauen unterscheiden.
Auf einen ganzen Datensatz angewendet, erfolgt die Berechnung klassenweise. Anschließend wird ein Durchschnittswert gebildet, weshalb diese Metrik auch als „mean average precision“ (mAP) bezeichnet wird.
Eine weitere Berechnungsmöglichkeit wird in Forschungen basierend auf dem „COCO“-Datensatz (vgl. Lin u. a. 2015) verbreitet angewendet. Dort wird die mAP als Durchschnittswert über einen in 5%-Schritten von 50% bis 95% inkrementierenden IoU-Grenzwert berechnet.

2.6.5. Mögliche Probleme des trainierten Netzes

Von einem Machine-Learning-Modell wird erwartet, dass es generalisiert funktioniert. Im Falle einer Objekterkennung bedeutet dies, dass das Modell Objekte in Daten findet, die es

zuvor noch nicht gesehen hat. Die Performance kann in zwei Ausprägungen von der gewünschten Generalisierung abweichen. Diese Fälle werden als „Overfitting“ und „Underfitting“ bezeichnet.

2.6.5.1. Overfitting (high variance)

Overfitting tritt auf, wenn das Modell von zu vielen Features in den Trainingsdaten lernt. Dadurch arbeitet das Modell nicht generalisiert genug, um in neuen Daten die Objekte zu erkennen. Häufig tritt dieser Fall ein, wenn ein Modell zu lange trainiert wurde und es sich dadurch zu exakt an die Trainingsdaten anpasst. Dies kann dazu führen, dass bei vielen verrauschten Bildern das Rauschen als Merkmal erkannt wird und das Modell daher keine Objekte erkennt, die nicht verrauscht sind.

2.6.5.2. Underfitting (high bias)

Underfitting tritt auf, wenn das Modell nicht ausreichend trainiert wurde. Ursächlich dafür können z. B. ein zu kurzes Training oder zu wenige Trainingsdaten sein.

3. Anforderungsanalyse

In diesem Kapitel werden Anforderungen an die Anwendung definiert. Dafür wird zunächst das Szenario festgelegt, welches als Grundlage für die Anforderungsermittlung dient. Anschließend werden die einzelnen Teilbereiche der AR-Anwendung Aspekte betrachtet.

3.1. Definition des Szenarios

In dieser Arbeit wird ein Konzept erarbeitet, das verdeutlicht, wie mit AR die Arbeiten an elektrischen Anlagen unterstützt werden können. Da es beliebig viele unterschiedliche Arbeitsszenarien gibt, wird das Szenario auf Wartungs- und Reparaturarbeiten an Schaltschränken festgelegt, welche nachfolgend präziser beschrieben werden.

3.1.1. Wartungsarbeiten

An Schaltschränken gibt es geplante und ungeplante Wartungsarbeiten. Geplante Arbeiten wiederholen sich meist nach bestimmten Zeitintervallen, während ungeplante Wartungsarbeiten durch besondere Ereignisse hervorgerufen werden. Dabei muss es sich nicht zwingend um Defekte handeln, sondern es können auch unerwartet hoher Verschleiß oder Rückrufaktionen der Hersteller einzelner Komponenten ungeplante Wartungsarbeiten verursachen.

3.1.2. Defekte

Ungeplante Arbeiten sind häufig auf plötzliche Defekte zurückzuführen. Tritt ein solcher Fall ein, gibt es mehrere Handlungsmöglichkeiten, die abhängig vom defekten Bauteil durchgeführt werden. Aufgrund der Vielzahl von Herstellern mit eigenen Bedingungen ist der Ablauf eines Servicefalls unterschiedlich. Drei Grundkonzepte von Servicefällen werden nachfolgend als Beispiele kurz beschrieben:

- **Reparatur vor Ort:** Je nach Ursache des Defekts können einzelne Komponenten vor Ort repariert bzw. wieder in Betrieb gesetzt werden. Ist die Ursache lediglich ein defektes oder loses Kabel, kann dieses einfach getauscht oder wieder gesichert werden. Nach kurzem Stillstand kann die Anlage wieder in Betrieb gehen. Wenn es sich bei der Ursache jedoch beispielsweise um einen defekten Kondensator auf einer Platine handelt, ist eine Reparatur am Installationsort häufig nicht möglich.
- **Bring-In-Service:** Diese Art des Services wird in der Regel nur für nicht betriebswichtige Anlagen verwendet. Die defekte Komponente oder die ganze Anlage muss zum Hersteller gebracht und dort repariert werden. Während der Reparaturdauer fällt die Anlage aus.
- **Vorabaustausch:** Für betriebswichtige Anlagen wird meistens ein Servicevertrag mit Vorabaustausch gewählt, welcher die Servicefirma verpflichtet, innerhalb eines definierten Zeitraums nach Fehlermeldung ein Austauschteil bereitzustellen. Die defekte Komponente wird sofort gegen ein neues Produkt ausgetauscht. Erst anschließend, wenn die Anlage wieder läuft, wird das defekte Bauteil an den Hersteller zur Reparatur geschickt. Durch dieses Vorgehen wird die Stillstandszeit minimiert.

3.1.3. Erkennung von Defekten

Um zu wissen, wann ein Austausch welcher Komponente erforderlich ist, muss zunächst der Defekt erkannt werden. Die Meldung kann durch ein Überwachungssystem erfolgen, welches Unregelmäßigkeiten im Betrieb, beispielsweise Stromschwankungen, entdeckt. Außerdem kann die Meldung durch den Nutzer der Anlage erfolgen, wenn diese nicht mehr korrekt arbeitet oder ein unerwartetes Verhalten aufweist. Welche Komponente diese Fehler verursacht, steht damit noch nicht zwangsläufig fest. Wie schnell ein Fehler entdeckt wird, hängt dabei häufig mit der Erfahrung und Anlagenkenntnis der Techniker zusammen. Ist es eine unbekannte Anlage, hilft nur ein Blick in den Schaltplan, um mögliche Ursachen auszumachen. Eine augmentierte Einblendung des Schaltplans, idealerweise mit Livedaten der Komponenten, kann daher die Erkennung vereinfachen.
Ein Defekt kann sich außerdem akustisch äußern, wenn plötzlich unerwartete Geräusche auftreten, die es bei einer fehlerfrei arbeitenden Anlage nicht gibt. Die ursächliche Komponente ist durch Rückverfolgung des Geräusches oftmals leicht zu finden, sofern diese zugänglich ist.

Thermografie

Da der Stromfluss in elektrischen Bauteilen mit Wärmeemissionen einhergeht, äußern sich elektrische Defekte manchmal durch erhöhte Temperaturentwicklung. Dies ist fühlbar, was aber nicht immer gefahrlos für den Techniker ist. Daher ist hier eine Thermografiekamera hilfreich, um auch aus der Entfernung Unregelmäßigkeiten zu erkennen. Diese Kameras gibt es als Aufsätze für Smartphones (bei wenigen Sondermodellen auch eingebaut) oder aber als separate Geräte. Hier ist zwischen Kosten und Auflösungsvermögen abzuwägen, welche Art sinnvoll ist. In Abschnitt 2.2.2 wurde bereits ein Thermografieaufsatz gezeigt, der als Aufstecksystem die Möglichkeiten von Smartphones ergänzt. Ein solcher Kameraaufsatz ist deutlich kompakter als separate Thermografiekameras und kann daher einfach, z. B. im Werkzeugkoffer, mitgenommen werden.

3.2. Festlegung des Szenarios

Als Versuchsszenario wird repräsentativ für einen Schaltschrank ein Serverschrank gewählt. Dieser enthält sowohl Server- als auch Netzwerkhardware, wobei sich für das AR-Szenario auf die vier verbauten Server konzentriert wird. Die Server haben eine schwarze Front, wobei jeweils zwei dieser Server identisch sind. Die Anwendung soll es einem Techniker ermöglichen, schnell die Bauteile im Schrank zu identifizieren bzw. zu lokalisieren. Auch in Schaltschränken kommt es vor, dass ein Bauteil mehrfach verbaut ist, jedoch verschiedene Funktionen erfüllt.

3.2.1. Aufbau des Schaltschranks

Tabelle 3.1 zeigt den Aufbau des für das Untersuchungsszenario relevanten Teils des Schaltschranks. Dort befinden sich in den Höheneinheiten (HE) 32-47 die vier zu erkennenden Server. Wie dieses Szenario in der Realität aussieht, zeigt Abbildung 3.1.

3.2.2. Zweites Versuchsszenario

Aufgrund aktueller Umgebungsbedingungen durch die Coronapandemie ist das Versuchsszenario des Serverschranks im Laufe der Arbeit nicht mehr frei zugänglich, weshalb ein neues Szenario gewählt werden muss. Dieses soll sich weiterhin im Rahmen Schaltschrank bzw. Serverschrank bewegen und ähnlich kontrollierbare Bedingungen aufweisen. Das Szenario des Serverschranks ist dadurch charakterisiert, dass eine kontrastarme, flache, schwarze

Abbildung 3.1.: Die vier Server, die erkannt werden sollen

Tabelle 3.1.: Aufbau des CSTI-Racks

Höheneinheiten	Gerät
48	frei
44-47	Main01
40-43	Main02
36-39	CSTI01
32-35	CSTI02
1-31	nicht relevant

Oberfläche betrachtet wird.
Als neues Szenario wird der Sicherungsschrank (Abbildung 3.2) in der eigenen Wohnung gewählt. Dieser ist frei für Versuche zugänglich und bietet durch eine größtenteils flache, weiße Oberfläche vergleichbare Kontrastverhältnisse wie der Serverschrank. Die Beleuchtungsverhältnisse sind kontrollierbar. Weiterhin handelt es sich bei einem Sicherungsschrank um einen Schaltschrank.

Die Funktionen der verschiedenen Bauteile sind in Tabelle 3.2 aufgelistet.

3.3. Hardwareanforderungen

Nachfolgend werden die Anforderungen an die Hardware dargestellt, um eine Objekterkennung in einer AR-Anwendung zu ermöglichen. Zu den Hardwareanforderungen zählen neben dem AR-Gerät auch die Anforderungen an die Versuchsumgebung.

Abbildung 3.2.: Das zweite Versuchsszenario mit den zu erkennenden Objekten

Tabelle 3.2.: Liste der Elemente des Sicherungskastens

Position	Funktion
1./2.	Hauptschalter
3.-6.	FI
13.-15.	Herd
16.	Waschmaschine
17.	Flur
18.	kl. Zimmer 1
19.	kl. Zimmer 2
20.	Steckdosen Flur
21.	—
22.	Heizung
23.	—
24.	Bad
25./26.	—
27.	Geschirrspüler
28.	Trockner
29.	Zimmer 1
30.	Zimmer 2
31.	Zimmer 3

3.3.1. Geräte

Als Geräte für AR-Anwendungen kommen viele unterschiedliche Modelle infrage. Eingeschränkt werden die Möglichkeiten durch die Vorgabe, dass sich die Verwendung der Geräte in der Nähe eines Schaltschranks so einfach wie möglich gestalten sollte. Daher ist die Auswahl auf kompakte und mobile Geräte beschränkt. Allerdings muss auch die Darstellung eines Schaltplans in einer Größe möglich sein, die möglichst nah an die Originalgröße heranreicht. Durch Zoomen auf dem Bildschirm können Details vergrößert werden. Dies reduziert jedoch gleichzeitig den Überblick über umgebende Elemente. Aus diesem Grund sollten die Geräte eine Bildschirmdiagonale zwischen fünf und zwölf Zoll haben. Größere Geräte werden unhandlich, während kleinere Geräte die Übersichtlichkeit stark einschränken, selbst wenn die Funktionalität der Anwendung gegeben ist. Bei solchen Geräten besteht die Gefahr, dass diese aus praktischen Gründen nicht genutzt werden.
Für ein mobiles Einsatzszenario ist auch eine angemessene Akkulaufzeit notwendig. Ein Arbeitstag mit gelegentlicher Nutzung der Anwendung muss möglich sein. Gelegentliche

Benutzung bedeutet in diesem Zusammenhang, dass bei Arbeitsbeginn der Schaltschrank gescannt und eine Interaktion wie z. B. das Aufrufen eines Schaltplans für einen Anschluss durchgeführt wird. Danach wird das Gerät beiseitegelegt und die Arbeit ausgeführt. Dabei muss es möglich sein, dass der Arbeiter zwischendurch einzelne Schritte überprüfen kann.
Um das System aktuell zu halten, ist eine Netzwerkverbindung notwendig. Damit das Gerät nicht ständig über ein Kabel verbunden werden muss, sollte die Verbindung drahtlos möglich sein. Dies kann z. B. durch WLAN, Bluetooth und das Mobilfunknetz geschehen. Eine Kombination aus allen drei Varianten ist heute bereits in vielen Geräten gängig. Bluetooth und WLAN haben nur eine kurze Reichweite und setzen eine Infrastruktur vor Ort voraus, während Kommunikation über das Mobilfunknetz von spezieller Infrastruktur unabhängig ist.
Weitere Kriterien betreffen die Geräteleistung und den Speicherplatz. Die Speicheranforderungen sind heutzutage leicht zu erfüllen, da die meisten Smartphones bereits 16GB oder mehr Speicher besitzen. Ein Schaltplan ist nur wenige Megabyte groß, daher kann bereits bei kleinem Speicher eine große Zahl an Plänen lokal vorgehalten werden. Wichtiger ist die Rechenleistung, da diese für die Erkennung und das Tracking von Objekten benötigt wird. Darüber hinaus sind Kamera und Sensoren nötig. Aktuelle AR-Frameworks prüfen daher vor Anwendungsstart, ob die Anwendung im AR-Modus gestartet werden kann.
Bei Geräten von Apple ist die Hardwareanforderung hinsichtlich der Kompatibilität mit dem AR-SDK von Apple, dass es sich bei dem Prozessor um einen A9-Chip oder neuer handelt. Dies schränkt die Auswahl bereits auf Geräte aus dem Jahr 2015 oder neuer ein.
Bedingt durch die offene Implementierung des Androidsystems und der Verfügbarkeit für andere Hersteller ist die Liste möglicher Geräte deutlich komplexer. Eine Liste findet sich auf der Entwicklerseite von ARCore[1]. Einige Geräte haben auch abweichende Androidversions- oder auch Geräterevisionsanforderungen, daher ist bei älteren Androidgeräten ein Blick auf die Tabelle der Entwicklerseite unerlässlich.
Tabelle 3.3 liefert abschließend eine kurze Übersicht über die Anforderungen an das Entwicklungsgerät.

Tabelle 3.3.: Auflistung der Anforderungen an das Entwicklungsgerät

Anforderung	Wert
Displaydiagonale	5-12 Zoll
Akkulaufzeit	> 8 h
Kommunikation	WLAN, Mobilfunk
Speicherplatz	> 8 GB
Leistung	AR-tauglich

[1] Google, https://developers.google.com/ar/discover/supported-devices, abgerufen am 10.06.2020

3.3.2. Versuchsumgebung

Eine fest definierte Versuchsumgebung wird im Bereich der Objekterkennung mit Machine Learning für zwei Schritte benötigt: die Erstellung von Trainingsdaten und den späteren Test der Erkennung mit Hilfe des trainierten Systems.
Um zuverlässige Versuche durchzuführen, muss diese kontrollierte Testumgebung bereitstehen, in der vom Training bis zum Versuch keine Veränderungen vorgenommen werden. Nur durch die feste Definition der Umgebung können reproduzierbare Ergebnisse geschaffen werden. Sowohl für die Erstellung der Trainingsdaten als auch beim Test der Erkennung ist es wichtig, Kontrolle über die Lichtverhältnisse zu haben. Hierdurch kann getestet werden, wie groß der Einfluss der Umgebungsbeleuchtung auf die Funktionalität der Objekterkennung ist.

3.3.3. Tracking

Ein stabiles Tracking der erkannten Objekte ist unerlässlich für die Nutzbarkeit der Anwendung. Nur bei einer stabilen Anzeige der Objekte können diese zuverlässig ausgewählt werden, was eine erfolgreiche Interaktion ermöglicht. Die Stabilität des Trackings ist in der Versuchsumgebung zu evaluieren. Stabilität ist auch eine Anforderung an die AR-Bibliothek, die in der Anwendung benutzt wird. Diese Bibliotheken beziehen in der Regel bereits unterschiedliche Sensoren der Geräte ein, um das Tracking zu stabilisieren. Je nach Hersteller, Gerät und Umsetzung kann dies unterschiedlich stabil ausfallen, weshalb ggf. weitere Maßnahmen wie z. B. Glättung der Erkennungsfrequenz oder der zusätzliche Einsatz von Funktionen aus der Bildverarbeitung notwendig werden. Stabilität bedeutet in diesem Zusammenhang, dass die Position des erkannten Objekts nur um wenige Pixel schwanken darf und die Erkennung eines Objekts konstant bleibt, also nicht zwischen unterschiedlichen Objekten schwankt.

3.4. Softwareanforderungen

Im folgenden Abschnitt werden die Anforderungen an die Software herausgestellt. Dabei werden zunächst die Anforderungen an die Gerätesoftware und anschließend an die zur Entwicklung nötigen Bausteine definiert.

3.4.1. Betriebssysteme

Damit die Betriebssysteme auch AR- und ML-Funktionen bieten, müssen diese mit einer bestimmten Version auf den Smartphones oder Tablets installiert sein. Die nötigen Versionen sind in Tabelle 3.4 aufgelistet.

Tabelle 3.4.: Minimal benötigte Betriebssystemversionen

Betriebssystem	minimale Version
Android	minimum 7.0 (wenn nicht anders angegeben)
Apple iOS	11.0

3.4.2. Programmiersprachen

Aus der Wahl von Gerät und Betriebssystem resultiert die Forderung der Programmiersprache. Grundsätzlich gilt, wie in Kapitel 2.5.1 beschrieben, dass Android Apps in Java oder Kotlin und iOS Apps in Swift oder Objective C entwickelt werden. Sowohl unter Android als auch unter iOS sollte die Sprache gewählt werden, die den Code minimiert und die bessere Kompatibilität zu erforderlichen Paketen aufweist. Hierfür soll eine Verwaltung per Paketmanager möglich sein. Gleichzeitig muss die Sprache eine stabile Binärschnittstelle (application binary interface (ABI)) enthalten, damit keine Kompatibilitätsprobleme mit anderen Frameworks auftreten. Hier spricht man auch von ABI-Stabilität.
Ein besonders wichtiger Faktor ist neben Sicherheit und Stabilität die Zukunftssicherheit. Die Anwendung soll nicht in wenigen Jahren komplett portiert werden müssen, was in der Regel mit einem großen Aufwand verbunden wäre.

3.4.3. Softwarebibliotheken

In der Softwareentwicklung kann in vielen Fällen auf bestehende Bibliotheken zurückgegriffen werden. In Kapitel 2.5.3 wurden bereits einige Softwarebibliotheken für AR-Anwendungen kurz beschrieben. Grundlage für die Auswahl passender Bibliotheken sind Anforderungen an die Anwendung, welche hier aufgelistet werden.
Vor der Einbindung einer Bibliothek in die Anwendung muss zunächst die Lizenz geprüft werden. Damit die Software kostenlos bleibt, beschränkt sich die Auswahl auf kostenfreie Pakete.
Damit eine AR-Anwendung nutzbar ist, wird ein stabiles Tracking der Umgebung vorausgesetzt. Das bedeutet, dass die Bibliothek Funktionen bereitstellen muss, die eine Position in

der Umgebung erkennen und konstant verfolgen, auch wenn das Gerät bewegt wird. Minimale Bewegungen der Position können dabei in Kauf genommen werden, jedoch darf die Position nicht „springen“. Als minimal werden hier Bewegungen bezeichnet, die sich im Bild als leichtes Wackeln um den Koordinatenpunkt im Raum zeigen. Sprünge der Positionskoordinaten sollten nicht entstehen.
Mit den AR-Bibliotheken ist es möglich, Positionen im Raum zu markieren und zu verfolgen, selbst wenn diese Koordinaten zeitweise aus dem Bild verschwinden. Die relevanten Positionen können entweder manuell oder durch eine automatische Erkennung festgelegt werden. Um diese Objekterkennung zuverlässig zu gestalten, sollen mit Machine Learning trainierte Modelle in die Anwendung eingebunden werden. Daher muss eine Bibliothek ausgewählt werden, welche die Nutzung dieser Modelle ermöglicht.
Neben AR und ML wird auch die Anbindung an eine Datenbank durch Bibliotheken bereitgestellt. Je nach Programmiersprache stehen hierfür mehrere unterschiedliche Pakete zur Verfügung, die nach den folgenden Anforderungen ausgewählt werden müssen.

- Lese- und Schreibzugriffe müssen in wenigen Schritten implementierbar sein
- Netzwerkverbindungen zu redundanten Servern müssen möglich sein
- Fallback auf lokale Datenbank ist notwendig

3.4.4. Objekterkennung

Neben dem Tracking besteht eine weitere zentrale Anforderung in der Objekterkennung. Damit der Anwender nicht in einem bestimmten Winkel vor den Objekten stehen muss, soll die Erkennung bis zu einem Winkel von 45° in jede Richtung um das Objekt arbeiten. Gleichzeitig ist es erforderlich, dass die Bauteile zuverlässig erkannt werden. Neben dieser Klassifikation ist auch eine präzise Lokalisation der Objekte notwendig. Damit von einer guten Erkennung gesprochen werden kann, muss der IoU einen Wert von mindestens $0,5$ aufweisen. Als weitere Bewertungskriterien sollen mAP50% und „Varied AP“ herangezogen werden. Die mAP50% soll dabei möglichst 100% erreichen. Eine zusätzliche Herausforderung besteht darin, dass im Testszenario, und auch bei elektrischen Anlagen, häufig mehrfach dasselbe Bauteil für unterschiedliche Funktionen genutzt wird, was die Unterscheidung erschwert. Konkret handelt es sich im Anwendungsbeispiel um identische Servergehäuse, die sich optisch lediglich durch ein Label und die Einbauposition unterscheiden.
Im Szenario des Sicherungskastens in der Wohnung sind zwei identische Reihen von Sicherungen vorhanden, die durch die Objekterkennung zuverlässig voneinander unterschieden werden müssen. Eine Unterscheidung jeder einzelnen Sicherung durch das ML-Modell ist hingegen nicht notwendig, da es sich um identische Bauteile handelt und bei bekannter Anzahl in einer Reihe die Unterteilung durch ein Raster erfolgen kann.
Eine weitere Anforderung an die Objekterkennung resultiert daraus, dass die Anwendung auf

einem mobilen Gerät laufen soll. Daher kann nicht auf leistungsstarke Hardware zurückgegriffen werden. Damit keine wahrnehmbaren Verzögerungen entstehen, soll die Erkennung in Echtzeit arbeiten. Aufgrund von Verzögerungen durch Netzwerkverbindungen ist eine Auslagerung der Berechnung auf externe Systeme nur unter bestimmten Voraussetzungen möglich, weshalb die Objekterkennung lokal auf dem Gerät erfolgen soll.

3.4.5. Datenbank

Bei der Softwareentwicklung können Daten entweder codiert im Quelltext gespeichert oder von externen Quellen bezogen werden. Codierung eignet sich lediglich für persistente Daten, die höchstens bei einer neuen Programmversion geändert werden sollen. Im Quelltext codierte Daten werden daher möglichst vermieden und in der Regel nur temporär zum Testen während der Anwendungsentwicklung verwendet. Für die Programmversionen, die an Endnutzer verteilt werden, sollten Daten von einer externen Datenbank abgerufen werden. Auf diese Weise wird es möglich, zusätzliche Inhalte hinzuzufügen, ohne neue Programmversionen liefern zu müssen. Das Programm muss lediglich einen Zugang zur Datenbank, den Umgang mit den Daten und deren Darstellung auf dem Endgerät bereitstellen.
Für das Szenario des Schaltschranks ist eine Datenbank unerlässlich, damit die Anwendung nicht nur für einen Schaltschrank bestimmte Informationen liefern kann, sondern für weitere Anlagen nutzbar wird. Da es viele unterschiedliche Datenbankarten (Kapitel 2.5.5) und verschiedene Implementierungen gibt, müssen die genauen Anforderungen an die Datenbank analysiert werden, um eine Auswahl treffen zu können. Zusätzlich muss bei den Datenbanken beachtet werden, welchen Lizenzen diese unterliegen oder ob sie frei verfügbar sind.

3.4.5.1. Datenbankabfrage

Die Datensätze müssen eindeutig abfragbar sein – das heißt, Schlüssel, anhand derer Datensätze identifiziert werden, müssen einzigartig sein. Um größtmögliche Kompatibilität zu gewährleisten, soll die Datenbank mit der standardisierten Sprache abfragbar sein, wobei es sich sowohl um SQL-, als auch um NoSQL-Datenbanken handeln kann.
Die Komplexität der Anfragen soll klein gehalten werden, was sich auf das Design der Datenbank auswirkt. Durch eine Abfrage sollen nur die notwendigen Daten zurückgeliefert werden und keine weiteren, nicht benötigten Daten. Mit dieser Vorüberlegung soll die übertragene Datenmenge reduziert und deren Verarbeitung vereinfacht werden.

3.4.5.2. Online- oder Offlinedatenbank

Die Anwendung wird auf mobilen Geräten eingesetzt. Einerseits muss die Datenbank an einem zentralen Punkt verfügbar sein, damit der Inhalt durch ein Content-Management-System (CMS) verwaltet werden kann und nicht auf jedem Gerät manuell eine neue Version eingespielt werden muss. Andererseits wäre für eine reine Onlinedatenbank eine permanente Netzwerkverbindung erforderlich, um die Funktionalität zu gewährleisten. Ohne Netzwerkverbindung wäre die Anwendung somit unbrauchbar. Schaltschränke finden sich häufig in Kellern oder anderen abgelegenen Orten eines Gebäudes, wo die Mobilfunkabdeckung häufig schwächer und das Signal zusätzlich durch den Schaltschrank gestört ist. GSM-Modems für Schaltschränke haben aus diesem Grund einen SMA-Anschluss (Sub-Miniature-A), damit eine Antenne aus dem Schaltschrank herausgeführt werden kann. Daher muss eine Offlinedatenbank lokal auf dem Gerät vorhanden sein, um die Funktionalität im Falle des Signalabbruchs zu erhalten. Die Datenbank muss eine Replikationsfunktion besitzen, um die lokale Version mit der Onlineversion abgleichen zu können.

3.4.5.3. Datenbankinhalt

Für Arbeiten an einem Schaltschrank sind folgende Daten notwendig:

- Bezeichnung des Anschlusses
- Zielpunkt der Verbindung
- Funktion
- Referenz zur Seite im Schaltplan
- Referenz zu entsprechendem Schaltplandokument

Diese Daten können als einfache Werte in der Datenbank eingetragen sein. Ein Schaltplan hingegen muss als Dokument hinterlegt werden, um dem Arbeiter weiterführende Informationen zu liefern. Die Anforderungen für die Dokumentenablage werden nachfolgend behandelt.

3.4.5.4. Dateiablage

In der Anwendung sollen nicht nur Texte oder Zahlen angezeigt werden, sondern auch Grafiken oder ganze Dokumente. Damit diese strukturiert abgefragt werden können, müssen diese Daten auch in der Datenbank systematisch abgelegt sein. Schaltpläne enthalten in der Regel Bild- und Textdaten und liegen als PDF vor. Diese Daten sind Binärdaten und

Tabelle 3.5.: Anforderungen an die Dateiablage

Kriterium	Anforderung
Infrastruktur	Datenbank- und Fileserver oder Dateiablagemöglichkeit in Datenbank
Performance	geringe Anforderungen, da Daten auch lokal vorgehalten
Dateigröße	< 10 MB
Dateimenge	gering (eine Datei je implementierten Schaltschrank)
Nutzerzahl	< 30

Tabelle 3.6.: Zusammenfassung der Datenbankanforderungen

Anforderung	Erläuterung
Art	Tabellen- oder Dokumentenstruktur
Abfrage	Standard SQL- oder No-SQL-Sprache
Redundanz	Hochverfügbar, Servercluster
Dateiablage	In Datenbank oder auf Fileserver
Replikation	lokale Kopie auf Gerät
Preis	kostenfrei

können nicht in einem Schema gespeichert werden. Solche Daten werden auch als „Binary Large Objects“ (BLOBs) bezeichnet und können auf drei grundlegende Arten abgelegt werden. Neben der Ablage in einem Dateisystem bieten auch SQL- und No-SQL-Datenbanken Möglichkeiten, BLOBs zu speichern. Ob diese Daten in einer Datenbank oder besser im Dateisystem hinterlegt werden, hängt von der Infrastruktur und von Performanceanforderungen ab. Auch Größe, Menge, Sicherheit und Verfügbarkeit der zu speichernden Dateien sind Kriterien, die für die Entscheidung herangezogen werden müssen.
Entsprechend des geplanten Nutzungsszenarios ergeben sich folgende Anforderungen, die in der Auswahl berücksichtigt werden müssen:

3.4.5.5. Zusammenfassung der Datenbankanforderungen

Zusammengefasst muss die Datenbank Replikationsmöglichkeiten bieten, damit eine Kopie lokal vorgehalten werden kann. Ein Dokumentenspeicher innerhalb der Datenbank ist nicht zwingend erforderlich, da in der Datenbank auch ein Verweis auf den Speicherort der Schaltpläne hinterlegt werden kann. Diese Referenz sorgt dann dafür, dass die betreffende Datei aus dem Gerätespeicher oder einer Onlinequelle aufgerufen werden kann. Um die Anwendungskosten zu minimieren, sollte die Datenbank entweder im Unternehmen bereits eingesetzt oder frei verfügbar sein.

3.5. Machine Learning

Das Machine Learning ist wesentlich für die Qualität der Objekterkennung. In diesem Bereich gibt es viele unterschiedliche Verfahren und Bibliotheken, die auf ihre Eignung geprüft und dementsprechend ausgewählt werden müssen. Zusätzlich sind anforderungsspezifische Trainingsdaten notwendig, um das System zu trainieren.

3.5.1. Anforderungen an das Toolkit

Die Softwarebibliothek muss ermöglichen, dass trainierte ML-Modelle importiert werden können. Darüber hinaus ist es erforderlich, dass die Objekterkennung durch das Netz in Echtzeit ohne wahrnehmbare Verzögerung abläuft. Die Erkennung und Aktualisierung der Objektposition muss hierfür in weniger als 40 ms erfolgt.
Da Objekterkennungen in der Regel auf CNNs basieren, muss speziell diese Art der Netze mit der gewählten Bibliothek implementierbar sein. Als Eingangsdaten müssen die Frames eines Videostreams von der eingebauten Kamera genutzt werden.

3.5.2. Trainingsdaten

Voraussetzung für das Training eines Machine-Learning-Models sind passende Daten, auf deren Basis das Training erfolgen kann. In diesem Fall wird eine Objekterkennung entwickelt, das heißt, man benötigt als Trainingsdaten Informationen über die zu erkennenden Objekte. Diese müssen als gelabelte Bilddaten vorhanden bzw. erstellbar sein. Dabei geht es nicht ausschließlich um die Datenmenge, sondern darum, Daten zu verwenden, die das Objekt möglichst variabel zeigen. Dadurch lernt das Modell, die Objekte unter verschiedenen Bedingungen zu erkennen. Hierfür müssen Bilddaten zur Verfügung stehen, die unter unterschiedlichen Beleuchtungsverhältnissen und Blickwinkeln erstellt wurden. Wie ein Datensatz aussehen kann, wurde bereits in Kapitel 2.6.2.1 beschrieben.
Zusätzliche Variabilität der Daten muss durch Augmentierungen künstlich erzeugt werden. Dabei sind Bilder des Objekts zu nutzen, die es bei unterschiedlichen Belichtungen zeigen.
Für das Training soll ein Datensatz erzeugt werden, der groß genug ist, um eine Aufteilung in Trainings- und Validierungsdaten zu ermöglichen. Weitere Bilddaten sollen mit Verfahren der Dataaugmentation erzeugt werden. Ob die Trainingsdaten ausreichen, soll durch Auswertung mit Metriken wie mAP bestimmt werden.

3.5.3. Labeling

Da das Labeling ohne ein spezialisiertes Tool sehr zeitraubend wäre, soll es nicht durch manuelles Erstellen einer .json- oder .xml-Datei durchgeführt werden, sondern iterativ. Das verwendete Tool soll Bilder importieren, durch Ziehen mit der Maus Labels erstellen und diese automatisch zu jedem Bild speichern können. Webanwendungen, die den Upload der Bilder auf einen Cloudspeicher erfordern, sollen aus Datenschutzgründen nicht genutzt werden.
Da unterschiedliche ML-Toolkits auch verschiedene Formate beim Import der Labels erwarten, sollen durch Exportfunktionen mindestens die Formate für CreateML, YOLO und Tensorflow bereitgestellt werden. Optionale Funktionen sind das Bearbeiten der Bilder und Hinzufügen von Augmentierungen.
Um schnell größere Datenmengen zu labeln, soll es außerdem möglich sein, die Daten mit bereits trainierten Netzen vorzulabeln, damit nur noch manuelle Korrekturen durchgeführt werden müssen. Auf diese Weise kann der Arbeits- und Zeitaufwand beim Generieren großer Datensätze minimiert werden. So kann beispielsweise ein erstes Training mit nur wenigen Bildern erfolgen und das Ergebnis genutzt werden, um weitere Bilder vorzulabeln und diese wiederum zum Training zu verwenden.

3.5.4. Machine-Learning-Modell

Das fertige, trainierte Machine-Learning-Modell soll in einem Format vorliegen, welches mit dem gewählten Framework kompatibel ist. Als Eingabewerte soll nicht mehr notwendig sein, als das Einspeisen eines Bildes. Die Ausgabewerte müssen die Parameter einer Bounding Box für jede Klasse enthalten sowie einen zugehörigen Confidence-Wert.
Da es sich um eine mobile Anwendung handelt, soll das Modell möglichst wenig Speicherplatz belegen. Als Richtwert werden hier max. 100 MB für ein Modell, speziell für das Szenario des Sicherungskastens, festgelegt. Diese Größe ist auch als Update über ein mobiles Datennetz noch handhabbar.

3.5.5. Trainingsplattform

Da gängige Objektdetektoren auf CNNs basieren, lässt sich die Eigenschaft nutzen, dass diese Netze sich gut parallelisieren lassen. Während eine moderne CPU zwar bereits bis zu 64 Kerne mit jeweils zwei Threads besitzt, können Grafikkarten durch ihre spezielle Architektur mit mehreren tausend, teils auf Tensorberechnungen spezialisierten Kernen einen erheblichen Leistungsschub beim Training von CNNs bieten. NVIDIA gibt die ML-Performance

einer Tesla V100 Grafikkarte als vergleichbar zu 32 CPUs an, abhängig von CPU und Trainingsverfahren (vgl. NVIDIA 2020).
Für das Training soll aus den genannten Gründen die Parallelisierung auf einer Grafikkarte genutzt werden, um hohen Rechenzeiten vorzubeugen.

3.6. Userinterface

Im folgenden Abschnitt werden die Anforderungen ermittelt, die den Aufbau des Userinterfaces (UI) betreffen. Diese Anforderungen lassen sich in Inhalt, Interaktion und Darstellung gliedern.

3.6.1. Inhaltliche Anforderungen an die Anwendung

Die inhaltlichen Anforderungen an die Anwendung sind stark durch den Anwendungsfall getrieben, für den die Applikation verwendet werden soll. Dabei gilt es abzuwägen, welche und wie viele Informationen sinnvoll darzustellen sind. Wenige Informationen bergen die Gefahr, dass Details fehlen, während viele Informationen die Übersichtlichkeit einschränken und dabei wichtige Punkte übersehen werden können. Daher darf das Display nicht direkt mit Inhalten geflutet werden, sondern muss zunächst die Bezeichnungen der erkannten Bauteile im Bild darstellen. Erst nach einer Nutzerinteraktion – z. B. durch Antippen – sollen weitere Inhalte dargestellt werden.
Diese Zusatzinformationen lassen sich in zwei Kategorien unterteilen: primäre Inhalte, die notwendig sind, um die Funktion zu sichern, und sekundäre Inhalte für weiterführende Informationen.

- Primäre Inhalte:

  Funktionsbeschreibung des Bauteils (maximal zehn Wörter)

  Schaltplanausschnitt (falls vorhanden)

- Sekundäre Inhalte:

  Detaillierte Beschreibung der Funktion

  Bilder

  Monitoringinformationen

Damit das System möglichst variabel bleibt, sollen so viele Inhalte wie möglich in die Datenbank geschrieben werden. Hierdurch kann die Anwendung nachträglich leicht um weitere Inhalte ergänzt werden, ohne das Programm zu editieren.

3.6.2. Interaktion und Darstellung

Die Interaktion mit der Anwendung kann unterschiedlich realisiert werden. Dabei wird zwischen grundlegenden Anforderungen, die nötig sind, um die Funktionalität zu gewährleisten, und optionalen Anforderungen, wie Komfortfunktionen oder alternative Interaktionsmöglichkeiten, unterschieden.
Die Anwendung soll auf einem Smartphone oder Tablet laufen, daher ist die grundlegende Bedienung durch Eingaben auf dem Touchdisplay unbedingt notwendig. Um Probleme bei der Eingabe zu vermeiden, müssen diese außerdem in geeigneter Größe eingeblendet sein. Dies beugt zugleich Fehlbedienungen vor. Weiterhin soll die Bedienoberfläche intuitiv gestaltet werden, wobei gängige Schaltflächen, wie ein „X“ zum Schließen von Einblendungen, bevorzugt zu nutzen sind. Gleiches gilt für Gesten, die auf dem Display ausgeführt werden.
Das für eine AR-Anwendung wichtige Kamerabild soll das Display vollständig ausfüllen. Eingeblendete Schaltflächen müssen groß genug sein, um diese mit dem Finger auszuwählen, gleichzeitig aber nicht so groß, dass die Orientierung im Kamerabild beeinträchtigt wird. Damit ein Techniker bei der Arbeit nicht ständig das Tablet vor den Schaltschrank halten muss, ist sicherzustellen, dass aufgerufene Annotationen erst dann verschwinden, wenn die Einblendung manuell geschlossen wird. Dadurch kann der Techniker das Gerät neben sich ablegen, um mit beiden Händen Arbeiten auszuführen und gleichzeitig die Informationen der Einblendung zu erhalten.

3.7. Zusammenfassung der Anforderungen

Dieser Abschnitt fasst noch einmal die Anforderungen an die Anwendung zusammen und dient als Grundlage für die Konzeption.
Es wird ein Entwicklungsgerät benötigt, welches einen Touchscreen mit einer Diagonale zwischen fünf und zwölf Zoll besitzt und Kommunikationsschnittstellen enthält. Auf diesem Gerät soll die Anwendung in einer kontrollierten Umgebung getestet werden, damit reproduzierbare Resultate entstehen. Für die Anwendung wird eine Objekterkennung benötigt, die zuverlässig Objekte im Server- bzw. Sicherungsschrank erkennt. Dabei soll die Erkennung auch unter verschiedenen Lichtverhältnissen und Neigungswinkeln bis zu 45^{circ} funktionieren.
Für Funktionalität und Flexibilität der Anwendung muss eine Datenbank gewählt werden, in der sowohl Werte als auch Dateien abgelegt werden können.
Die Darstellung der Anwendung soll die zur Verfügung stehende Bildschirmfläche vollständig nutzen und einen Techniker mit Informationen über die erkannten Objekte versorgen. Dabei dürfen Einblendungen die Verwendbarkeit der Anwendung nicht beeinträchtigen und sollen daher den Techniker mit kurzen Informationen und einem Verweis auf präzisere Informationen im Schaltplan unterstützen.

4. Konzeption

Durch die Vielzahl der Plattformen und Frameworks entstehen viele mögliche Kombinationen für den Aufbau einer AR-Anwendung. Im folgenden Kapitel werden Varianten für die einzelnen Komponenten der Anwendung aufgezeigt und ausgewählt.

4.1. Hardware

Nachfolgend wird auf Basis der Anforderungen aus Kapitel 3.3 ein Gerät ausgewählt, für welches die Anwendung entwickelt wird.

4.1.1. Geräte

Zu Beginn dieser Arbeit wurden in Kapitel 2.2 bereits mögliche AR-fähige Geräte aufgezeigt. In Verbindung mit den technischen Anforderungen der Anwendung (Kapitel 3.3) ergibt sich eine Übersicht von möglichen Geräten für die Entwicklung. Diese dient als Grundlage für die Auswahl eines Entwicklungsgeräts. Die Tabelle wurde dabei unter der Voraussetzung erstellt, dass die Geräte im Forschungslabor „Creative Space for Technical Innovations“ (CSTI[1]) der HAW Hamburg verfügbar sein müssen. Im Androidbereich kommen fortlaufend neue Geräte unterschiedlichster Hersteller auf den Markt oder sind bereits am Markt, weshalb hier eine Abgrenzung erforderlich wird.
Die Auswahl des Entwicklungsgeräts erfolgt als erster Schritt, da durch diese Entscheidung weitere Folgeentscheidungen beeinflusst und eingegrenzt werden. Aus Tabelle 4.1 geht hervor, dass, bis auf das Google Pixel C, alle Geräte für AR-Anwendungen geeignet sind. Aufgrund des Preises und der geringen Verbreitung kommt die Hololens für diesen Einsatz nicht in Frage. Außerdem müsste ein HMD immer separat transportiert werden, was nur explizit geplante Einsätze ermöglicht, nicht jedoch eine spontane Nutzung. Der Preis des iPhone 11 Pro Max ist zu relativieren, da iOS Anwendungen grundsätzlich abwärtskompatibel sind. In diesem Fall bedeutet das, dass alle iPhones bis zurück zum iPhone SE und 6s (aufgrund der AR-Anforderung nicht älter) ebenfalls kompatibel sind. Gleiches gilt für das iPad Air 3.

[1] CSTI, https://csti.haw-hamburg.de/

Tabelle 4.1.: Vergleich der verfügbaren Geräte

Gerät	Geräteart	Konnektivität	Speicher	Display	AR-Tauglichkeit	OS	Preis[a] ca.
iPhone 11 Pro Max	Smartphone	LTE, WLAN, Bluetooth, NFC	64 GB	6,5 Zoll Touch	ARKit, ARCore	iOS	€1249
iPad Air 3	Tablet	LTE, WLAN, Bluetooth	64 GB	10,5 Zoll Touch	ARKit, ARCore	iPadOS	€549
Google Pixel C	Tablet	LTE, WLAN, Bluetooth	32 GB	10,2 Zoll Touch	Nein	Android	€549
Google Pixel 3XL	Smartphone	LTE, WLAN, Bluetooth, NFC	64 GB	6,3 Zoll Touch	ARCore, 60fps	Android	€629
Samsung S10	Smartphone	LTE, WLAN, Bluetooth, NFC	128 GB	5,8 Zoll Touch	ARCore	Android	€559
Microsoft Hololens 1	HMD	WLAN, Bluetooth	64 GB	Holographisch	Ja	Nativ	€3299-€5489
Huawei Nexus 6P	Smartphone	LTE, WLAN, Bluetooth	64 GB	5,7 Zoll Touch	ab Android 8.0	Android	€679

[a]Bei Markteinführung

Auch hier können viele ältere Geräte für die Anwendung genutzt werden. Zudem sind Anwendungen zwischen iPad und iPhone in der Regel kompatibel, wodurch eine Anwendung mit Anpassungen an die unterschiedlichen Displaygrößen auf beide Systeme ausgerollt werden kann. Bei Androidgeräten ist die Frage nach der Kompatibilität deutlich komplexer. Hier müssen einzelne Geräte genau geprüft werden, da je nach Hersteller und Gerät die nötige OS-Version abweicht und oftmals schon nach wenigen Jahren der Support aufgegeben und die Geräte nicht mehr mit aktueller Software versorgt werden.
Aus den genannten Gründen wird ein Apple-Gerät als Zielplattform gewählt, wodurch die Wahl auf iPhone oder iPad eingeschränkt wird. Ein größeres Display macht die Arbeit komfortabler, solange kein mobiler Einsatz erwartet wird. Da dies bei der Entwicklung noch nicht gefordert ist, fällt die Wahl des Entwicklungsgeräts auf das iPad. Anwendungen, die für ein iPhone oder iPad entwickelt wurden, sind auch auf dem jeweils anderen Gerät lauffähig. Gegebenenfalls sollten dann jedoch Anpassungen der Darstellung vorgenommen werden.
Das iPad läuft mit dem Betriebssystem iPadOS (ersetzt seit neuestem iOS auf Tablets). Diese Entscheidung beschränkt die Auswahl der Softwarebausteine in den nachfolgenden Kapiteln, wodurch die Komplexität des weiteren Konzeptionsverfahrens verringert wird.

4.1.2. Tracking

Für ein stabiles Tracking, entsprechend der Anforderungen aus Kapitel 3.3.3, können zwei verschiedene Ansätze in einer AR-Anwendung verfolgt werden:

- Tracking durch Sensoren, nachdem ein Objekt einmalig erfasst wurde
- Fortlaufende Aktualisierung der Positionsdaten durch Objekterkennung

Das iPad bietet für beide Verfahren ausreichend Leistung und die passende Sensorik. Präferiert wird hier die Aktualisierung der Positionsdaten durch das ML-Modell. Die fortlaufende Anwendung dieses Modells auf das Kamerabild ist notwendig, damit Objekte, die neu im Bild auftauchen, auch direkt erkannt werden. Die Ausgabewerte des Modells enthalten die Koordinaten aller im Bild befindlichen Objekte. Dies kann auch für das Tracking nutzbar gemacht

werden, indem die Koordinaten der Bounding Boxes aller Objekte aktualisiert werden. Daher kann hier auf eine explizite Einbeziehung weiterer Sensoren verzichtet und ein ML-basiertes Tracking implementiert werden.

4.2. Softwareentwicklung

In diesem Kapitel werden die einzelnen Softwarebausteine ausgewählt, welche in die Anwendung implementiert werden.

4.2.1. Programmiersprachen

Auf Basis der Geräteauswahl beschränkt sich die Wahl der Programmiersprache auf Objective-C oder Swift, da nur diese für iOS geeignet sind. Diese beiden Sprachen werden mit den jeweiligen Vor- und Nachteilen ihrer Eigenschaften und Funktionen gegenübergestellt. Auf dieser Grundlage findet die Auswahl derjenigen Sprache statt, welche für die Anwendung am besten geeignet ist.

Tabelle 4.2.: Vergleich der möglichen Programmiersprachen

Objective-C	Swift
• nah an klassischer C-Syntax	• Syntax einfacher und schlanker
• sehr stabil	• Sicherheit durch statische Typisierung
• kompatibler zu C, C++ Libraries	• keine Dereferenzierung bei Pointern notwendig
• Cocoapods oder Carthage als Paketmanager	• Weniger Dateien, Definition und Implementierung vereint
• Gefahr, dass Apple den Support einstellt	• Bessere Performance
	• Stabilitätsvorteil durch Optionals (Variablen ohne Wert)
	• Datentypen als Konstante oder Variable
	• Swift Package Manager
	• junge Sprache, ggf. noch Bugs

Mit Swift hat Apple eine neue Programmiersprache eingeführt, welche die Anwendungsentwicklung vereinfachen soll und gleichzeitig, durch strikte Typisierung, sehr sicher und stabil

ist. Zwar können in der relativ jungen Sprache noch Bugs auftreten, mit der neuesten Version hat die Sprache aber ABI-Stabilität erreicht, wodurch die Kompatibilität zu Betriebssystemen sichergestellt ist. Objective-C gilt zwar grundsätzlich als stabiler, jedoch nimmt die Verbreitung ab, während Swift bei iOS Anwendungen immer mehr verwendet wird und auch Apple immer mehr Teile des Betriebssystems auf Swift umstellt. Aus Gründen der Zukunftssicherheit, Performance und Einfachheit wird für diese Anwendung auch Swift verwendet.

4.2.2. Objekterkennung

Die Anwendung muss mehrere Objekte im Kamerabild erkennen und entsprechend mit Bounding Boxes und Labeln versehen. Entsprechend der Anforderungen muss es möglich sein, dass dies mit einem großen Blickwinkel und einem breiten Spektrum an Lichtverhältnissen geschieht. Durch eine Image-Detection, das heißt, bei einem Vergleich mit einem Referenzbild, wäre dies nur an einem zweidimensionalen Objekt, also einem Bild, möglich. An einem realen, dreidimensionalen Objekt funktioniert dieser Ansatz hingegen nur aus dem Blickwinkel, aus dem auch das Referenzbild aufgenommen wurde. Durch klassische Algorithmen mit Featureextraktion ist die Erkennung unterschiedlicher 3D-Objekte sehr aufwendig, weshalb hier ein Machine-Learning-Ansatz verfolgt wird. Die Konzeption der Objekterkennung wird in Kapitel 4.3 behandelt.

4.2.3. Datenbank

Auf Basis der Anforderungen an die Datenbank, welche in Kapitel 3.4.5 festgelegt wurden, werden hier unterschiedliche Datenbanken gegenübergestellt und auf Erfüllungsgrad der Anforderungen geprüft. Da es sehr viele Datenbanken auf dem Markt gibt, wird hier nur eine Auswahl häufiger Vertreter betrachtet.

Tabelle 4.3.: Auswahl der Datenbank

Datenbank	Art	Wertspeicher	Dateiablage	Redundanz	Lizenz
SQLite	relational	Tabelle	BLOB	kein interner Support	frei
PostgreSQL	relational	Tabelle	BLOB	Failover, Load Balancing	frei
Microsoft SQL	relational	Tabelle	BLOB/Filestream	Failover	kommerziell
MongoDB	nichtrelational (Dokumentenstruktur)	JSON	in Chunks oder GridFS (>16MB) (vgl. MongoDB 2020)	Cluster (3 Server, Master/Slave)	frei
Firebase	nichtrelational (Dokumentenstruktur)	JSON	Firebase Cloudstorage	Master/Slave	kommerziell
MariaDB	relational	Tabelle	BLOB	Cluster oder Master/Slave	frei

4.2.3.1. Tabellen- oder Dokumentenstruktur

Aus den Anforderungen, ermittelt in Kapitel 3.4.5, geht hervor, dass sowohl einzelne Werte und kurze Beschreibungen als auch kleine Bilddaten in der Datenbank abgelegt werden

müssen. Hinzu kommt eine Datei je Schaltschrank, die den kompletten Schaltplan enthält. Auf dieser Grundlage wird eine dokumentenorientierte Datenbank ausgewählt. Dateien können direkt in Dokumenten als Chunks abgelegt werden, weshalb ein zusätzlicher Fileserver nicht benötigt wird. Ein Dateiabruf ist also mit einem Dokumentenaufruf möglich.

4.2.3.2. Festlegung der Datenbank

Als zu Grunde liegende Datenbank für die Anwendung wird die kostenfreie MongoDB gewählt. Für die Anwendung bietet diese Datenbank mehrere Vorteile.
Die MongoDB erlaubt einen redundanten Aufbau in einem Cluster von mindestens drei Servern, bei denen einer ausfallen darf, ohne die Verfügbarkeit zu beeinträchtigen. Diese Redundanz kann durch den Betrieb von drei Instanzen in einem Kubernetes Cluster weiter gesteigert werden. Die Konsistenz der Daten wird durch ein Master/Slave System gesichert. Schreibvorgänge finden nur auf dem Masterknoten statt, während für Lesevorgänge alle Knoten zur Verfügung stehen.
Die Dokumentenstruktur der Datenbank bietet den Vorteil, dass diese flexibel erweiterbar ist. Durch diese Flexibilität können weitere Schaltschränke mit ggf. erforderlichen, zusätzlichen Parametern implementiert werden, ohne die Struktur der bestehenden Daten anzupassen. Dies ist wichtig, da sich die Parameter mit der Erweiterung der Anwendung ändern können.
Ein Dokument kann eine maximale Größe von 16MB haben. Dies stellt gleichzeitig auch die maximale Dateigröße dar, die mit dem GridFS-Dateisystem in einem Dokument gespeichert werden kann. Für größere Dateien wäre der Abruf mehrerer Dokumente nötig.

4.2.3.3. Datenbankdesign

Ein mögliches Design resultiert aus der Datenbankauswahl. In diesem Fall werden JSON-Dokumente für die MongoDB benötigt und entsprechend der Anforderungen strukturiert. Durch die sinnvolle Strukturierung wird verhindert, dass bei einer Abfrage unnötige Daten mitgeliefert werden. Gerade bei nichtrelationalen Datenbanken kann es vorkommen, dass bei einer Abfrage nicht benötigte Daten mitgeliefert werden, wenn die Dokumentenstruktur ungünstig gewählt ist.
Als Grundlage für den Inhalt der Datenbank dienen die Informationen aus Kapitel 3.6.1 und Tabelle 3.2. Mit diesen Daten wird ein JSON-Dokument erstellt, welches in die Datenbank importiert werden kann.

4.2.3.4. Dateiablage

Um die passende Ablageform für Dateien wie z. B. PDF oder Bilddateien zu finden, müssen zunächst die Dateieigenschaften betrachtet werden. Für den vorliegenden Anwendungsfall gibt es zwei Typen von Dateien:

- PDF-Datei mit komplettem Schaltplan. Größe: 1-10 MB
- JPG-Datei mit kleinen Schaltplanausschnitten zur Einblendung. Größe: ca. 100 KB

Da zunächst nur ein Schaltschrank als Demonstrationsobjekt implementiert wird, ist die Speicherung eines Schaltplans mit maximal 10 MB Größe und vier kleiner Ausschnitte mit Größen von je maximal 100 KB notwendig. Der maximale Gesamtspeicherbedarf für diesen Fall beträgt daher 10,4 MB. Für Versuchszwecke können diese Dateien lokal auf dem Gerät vorgehalten werden. Der Speicherbedarf von 10,4 MB fällt bei einem Speicherplatz von 32 GB bei den AR-Geräten kaum ins Gewicht.
Für einen produktiven und auf mehrere Szenarien ausgedehnten Einsatz mit vielen Dateien empfiehlt sich die Dateiablage in der MongoDB. Dies minimiert den Verwaltungsaufwand eines zusätzlichen Fileservers und erlaubt es gleichzeitig, die Redundanz-, Verwaltungs- und Sicherheitsoptionen der existierenden Datenbank zu nutzen. Darüber hinaus reduziert sich der Aufwand für die Erstellung von Backups, während hier nur die Datenbank gesichert werden muss und nicht zusätzlich noch ein Fileserver.

4.2.4. Softwarebibliotheken

Für die Entwicklung müssen für die Datenbank, die AR-Umgebung und für das Machine Learning Softwarepakete ausgewählt werden, die die geforderten Funktionen zuverlässig bereitstellen. Die Wahl der Datenbank in Kapitel 4.2.3 schränkt die Auswahl der Softwarepakete zur Anbindung der Datenbank ein. Da die MongoDB als zugrunde liegende Datenbank ausgewählt wurde, werden nachfolgend nur Bibliotheken betrachtet, die eine Anbindung dieser Datenbank unterstützen.

4.2.4.1. AR-Bibliothek

Für die AR-Funktionalität einer mobilen Anwendung sind am Markt unterschiedliche Bibliotheken verfügbar, wovon die wichtigsten in Kapitel 2.5.3 kurz vorgestellt wurden. Eine wesentliche Aufgabe dieser Module besteht darin, das Tracking zu realisieren. In der nachfolgenden Tabelle 4.4 werden diese Bibliotheken gegenübergestellt und entsprechend der Anforderungen an das Tracking aus Kapitel 3.3.3 ausgewählt.

Tabelle 4.4.: Softwarepaket zur AR-Implementierung

Softwarepaket	Stärken	Funktionalität	OS	Lizenz
ARKit	Trackingstabilität	Sensorfusion, sehr umfangreich	iOS/iPadOS	kostenlos
ARCore	Mappingbereich	sehr umfangreich	iOS & Android	kostenlos
Vuforia	benötigt nur Kamera, plattformunabhängig	benötigt Marker für Tracking	iOS & Android	kostenpflichtig

Als Softwarebibliothek zur Implementierung der AR-Funktionen wird Apples hauseigenes ARKit ausgewählt. Dies ist zwar nur auf Apple-Geräten lauffähig, jedoch kommt dem Paket die Eigenschaft zu Gute, dass es für diese begrenzte Gerätegruppe optimiert ist. Das Tracking kann sehr gut stabilisiert werden, da die genauen Sensoreigenschaften der einzelnen Geräte bekannt und konsistent sind. Die anderen Pakete müssen eine große Bandbreite von Geräten und damit auch Sensoren unterstützen. Weitere Vorteile sind, dass ARKit nativ unter iOS läuft und in der Programmiersprache Swift zur Verfügung steht. Zudem entstehen, im Gegensatz zu Vuforia, keine Lizenzkosten.
ARCore hat einen leichten Vorteil, wenn es um das Mapping größerer Umgebungen geht. Hier werden mehr Punkte gespeichert, bevor alte Umgebungspunkte wieder verworfen werden. Für die Arbeit an einem Schaltschrank, bei der es um das Tracking einzelner Objekte geht, ist die Verwendung von ARKit vorteilhaft.

4.2.4.2. Anbindung der Datenbank

Für die Anbindung der MongoDB werden nachfolgend nur Bibliotheken betrachtet, die eine Anbindung dieser Datenbank unterstützen. Ein weiteres Kriterium zur Aufnahme in die Auswahl ist die Unterstützung der Programmiersprache Swift, wobei hier zwischen nativen Swift-Bibliotheken und C-Wrappern zu unterscheiden ist, die eine Verwendung mit Swift ermöglichen. Tabelle 4.5 zeigt die Funktionen der unterschiedlichen Softwarepakete, die es

Tabelle 4.5.: Softwarepaket zur Datenbankanbindung (Stand Dezember 2019)

Softwarepaket	Verbindung zu redundanten Clustern	Verbindungsart	Einschränkungen	Lizenz
Stitch	Ja	Atlas API Schnittstelle	nur Verbindung zu MongoDB Atlas Cloud[a]	Apache License 2.0
Mongoswift	Ja	direkte Netzwerkverbindung	C-Wrapper fehlerhaft	Apache License 2.0
MongoKitten	Ja	direkte Netzwerkverbindung	keine	MIT License

[a] MongoDB Atlas Cloud, www.mongodb.com, abgerufen am 04.06.2020

ermöglichen, die MongoDB anzubinden. Versuche haben gezeigt, dass eine Entscheidung für ein Paket nicht nur auf den Funktionsdaten basieren kann, sondern auch darauf geachtet werden muss, dass es fortlaufend von der Community gepflegt wird. Einen Hinweis auf die Softwarepflege liefert ein Blick auf die letzten Commitdaten im Repository der Entwickler. Aus diesem Grund wird hier lediglich der Stand von Dezember 2019 betrachtet und das Paket auf dieser Basis gewählt.

Die aufgelisteten Softwarepakete bieten ähnliche Funktionalitäten, weshalb die Wahl auf Basis der aktuell besten Unterstützung fällt. Die Verbindung konnte hingegen mit MongoSwift problemlos aufgebaut werden, allerdings zeigte sich beim Lesen der Datenbank ein Fehler im C-Wrapper, wodurch dieses Paket nicht nutzbar ist. Mongokitten hingegen bietet, wie Stitch die volle Funktionalität. Stitch benötigt einen Account für die MongoDB Atlas Cloud, da dort die Datenbank gehostet wird, wobei nur eine Datenbank kostenlos ist. Für diese Arbeit bietet die Wahl von Stitch zwei Vorteile: Setup und Betrieb einer Datenbank entfallen und die DB ist überall erreichbar.

4.2.4.3. ML-Bibliothek

Neben den Bibliotheken zur AR-Funktionalität und Datenbankanbindung gibt es auch für die Einbindung von ML-Modellen unterschiedliche Möglichkeiten. Eine Auswahl der wichtigsten Vertreter wird in Tabelle 4.6 verglichen und auf dieser Basis ein geeignetes Paket zur Verwendung ausgewählt. Für den vorhandenen Anwendungsfall wird CoreML gewählt. Dieses

Tabelle 4.6.: Softwarepakete zur Einbindung von ML-Modellen

Softwarepaket	Unterstützung für Objekterkennung	Unterstützte Frameworks	Lizenz	OS
CoreML	Ja	CreateML, TensorFlow	kostenfrei	iOS
MLKit	Betastatus	API & TensorFlow Lite	kostenfrei [a]	iOS & Android
Fritz AI	Ja	CoreML & TensorFlow	kommerziell	iOS & Android

[a] Je nach Anwendungsfall Cloudabo notwendig

Softwarepaket ist kostenfrei und liefert alle Funktionen, die für das Training und die Einbindung in die Anwendung notwendig sind. CoreML ist für iOS und iPadOS in Kombination mit der zugehörigen Hardware optimiert. Beim Kompilieren der Anwendung werden ML-Modelle automatisch auf die Hardware angepasst.

4.3. Machine Learning

In diesem Abschnitt werden unterschiedliche ML-Frameworks vorgestellt und es soll dabei entsprechend der Anforderungen verglichen werden, wie gut die jeweiligen Umgebungen für das Training einer Objekterkennung zur späteren Verwendung auf einem iPad geeignet sind. Dabei ist eventuell eine Konvertierung in das „mlmodel“ -Format notwendig, bevor das trainierte System verwendet werden kann.

4.3.1. Auswahl des Object Detectors

In Kapitel 2.6.3 wurden bereits verschiedene Object Detectors vorgestellt. Diese unterscheiden sich nicht nur in ihrem Aufbau, sondern auch in der Performance. Für die Auswahl eines geeigneten Detectors ist es notwendig, eine Abwägung zwischen Präzision und Geschwindigkeit vorzunehmen. Verschiedene Forschungen haben Detektoren verglichen. Grundsätzlich lässt sich festhalten, dass die zweischichtigen R-CNNs präziser arbeiten als die einschichtigen SSDs oder YOLO-Detektoren (vgl. Hui 2019). Die R-CNNs sind jedoch rechenintensiver und schaffen es daher auch auf leistungsstarker Hardware nur wenige Frames pro Minute zu verarbeiten, weshalb sie nicht echtzeitfähig sind.
Sowohl YOLO-Detektoren als auch SSDs können in Echtzeitanwendungen eingesetzt werden. Ob ein YOLO- oder SSD-Detektor schneller ist, hängt von der jeweiligen Version ab. YOLO ist aktuell in einer dritten Version vorhanden (YOLOv3), die bei vergleichbarer mAP schneller arbeitet als SSDs (vgl. Redmon und Farhadi 2018). Bereits in der zweiten Version (YOLOv2) werden höhere Frameraten erreicht als bei SSDs (vgl. Redmon und Farhadi 2016). Da am Sicherungskasten keine kleinen Objekte detektiert werden müssen und ein Geschwindigkeitsvorteil besteht, wird hier ein YOLO-Detektor gewählt. Ein YOLO-Object-Detector kommt in Apple's CreateML bei Objekterkennungen zum Einsatz. Hierbei handelt es sich um einen TinyYOLO Object Detector, eine speziell für mobile Anwendungen entwickelte Variante mit 9 Schichten. Die Entwicklung mit CreateML bietet den Vorteil, dass keine Konvertierung in das Format „.mlmodel“ notwendig wird.

Tabelle 4.7.: Auswahl der Machine Learning Frameworks

Framework	Objekterkennung	Training	Mögliche Detektoren	Ausgabe
Keras	Ja	lokal & extern	YOLO, SSD, R-CNN	.h5
Tensorflow	Ja	lokal & extern	YOLO, SSD, R-CNN	.tf
CreateML	Ja	nur lokal	YOLO	.mlmodel
Pytorch	Ja	lokal & extern	YOLO, SSD, R-CNN	.pt, .ptb

Für den vorhandenen Anwendungsfall wird im ersten Schritt das Tool CreateML gewählt. Da dieses automatisch mit xCode mitgeliefert wird, ist keine weitere Software notwendig. Das Tool bietet nur grundlegende Funktionen und wenige Einstellmöglichkeiten. Daher eignet es sich besonders für einen Test, da zudem kein Programmieraufwand erforderlich ist und das Format des erstellten Netzes direkt in CoreML verwendet werden kann. Neben den Trainingsbildern in einem Ordner werden zusätzlich die Labels in einer JSON-Datei benötigt. Bei ausreichend großen Datensätzen wählt das Tool selbstständig Trainings- und Validierungsdaten aus. Nachteilig ist allerdings die Berechnungsdauer. Da nur interne oder direkt angeschlossene Hardware genutzt werden kann, sind die Beschleunigungsmöglichkeiten

Tabelle 4.8.: Auswahl der Machine Learning Frameworks

Framework[a]	Stärken	Nachteile	Lizenz
Keras	• Schnelles Erstellen von Prototypen • kleine Datensätze • Support für unterschiedliche Backends	• Vergleichsweise langsam	• MIT-Lizenz
Tensorflow	• Große Datensätze • Hohe Performance • Funktionalität • Objekterkennung		• Apache 2.0
CreateML	• Einfach für einzelne Einsätze • Objekterkennung • kein Programmieraufwand	• geringe Flexibilität	• proprietär
Pytorch	• Flexibilität • Kurze Trainingsdauer • Debugging	• Komplexe Architektur	• Open Source

[a]Quelle: Eigene Darstellung basierend auf Edureka 2018

der Berechnung begrenzt. Bei großen Datensätzen sollte daher auf ein anderes Framework ausgewichen werden, um die Rechenzeiten zu minimieren.
Theoretisch eignen sich alle in Tabelle 4.8 genannten Frameworks zum Training einer Objekterkennung. Diese haben jedoch unterschiedliche Stärken und Schwächen, die in Form der wichtigsten Unterschiede in genannter Tabelle gezeigt sind.

Entscheidend für die Verwendung unter iOS ist, dass das trainierte Netz sich in ein „.mlmodel“ konvertieren lässt, damit es sich ohne große Anpassungen in mobile iOS-Anwendungen integrieren lässt. Durch Skripte ist die Anpassung von verschiedenen Modellen möglich.

4.3.2. Trainingsdaten

Für das Szenario liegen keine vorgefertigten Trainingsdaten vor, daher müssen diese generiert werden. Hierzu sind Bilder zu erstellen und anschließend zu labeln. Die Labels müssen mit einer Verknüpfung zum jeweiligen Bild in einer .xml- oder .json-Datei hinterlegt werden. Da die manuelle Eingabe der X- und Y-Positionen der Labels sehr mühsam wäre, gibt es eine Vielzahl kostenloser und kostenpflichtiger Tools. Entsprechend der Anforderungen an das Labeling aus Kapitel 3.5.3 ergibt sich folgende Übersichtstabelle (Tabelle 4.9) gängiger Labelingtools mit deren jeweiligen Funktionen. Nicht nur das gewählte Tool für das Labeling ist wichtig, sondern auch die Auswahl der Bilder, die gelabelt und als Trainingsdaten verwendet

Tabelle 4.9.: Auswahl des Labeltools

Tool	unterstützte Exportformate	Augmentierungsfunktionen	Kosten	OS
RectLabel[a]	YOLO (names, txt) COCO (JSON) CreateML (JSON)	Nein	2,99$/Monat	MacOS
Labelbox[b]	JSON, Konvertierung notwendig	Nein	limitiert kostenfrei	Webapp
MakeML[c]	Turi Create (CSV) Pascal VOC (XML) COCO (JSON) CreateML (JSON)	Ja	ab 4,53$/Monat	MacOS
Hasty.ai[d]	JSON, Konvertierung notwendig	Nein	limitiert kostenfrei	Webapp
LabelIMG[e]	YOLO, PASCAL VOC	Nein	kostenfrei	unabhängig

[a] RectLabel, www.rectlabel.com, abgerufen am 04.06.2020
[b] Labelbox, www.labelbox.com, abgerufen am 04.06.2020
[c] MakeML, https://makeml.app/, abgerufen am 04.06.2020
[d] Hasty.ai, www.hasty.ai, abgerufen am 04.06.2020
[e] LabelIMG, https://github.com/tzutalin/labelImg, abgerufen am 06.04.2020

werden. Aus Kapitel 3.5.2 geht hervor, dass eine große Datenmenge sowie eine Variabilität in der Menge vorhanden sein müssen. Insgesamt sollen mit einer Kamera zunächst 100 Bilder aufgenommen werden. Dabei ist zu beachten, dass die Aufnahmen die Objekte bei starker und schwacher Beleuchtung aus unterschiedlichen Blickwinkeln zeigen. Die Labels lassen sich mit dem gewählten Labelingtool erstellen und iterative Funktionen können genutzt werden. Bereits bei 100 Bildern ist es aufwendig, diese manuell zu labeln. Daher sollen zunächst nur ca. 15 Bilder gelabelt und mit diesen anschließend ein Netz trainiert werden. Das trainierte Netz soll automatisch die verbleibenden Bilder mit Labels versehen. Aufgrund der geringen, verwendeten Trainingsdatenmenge werden die Labels teilweise noch fehlerhaft sein, sodass diese manuell korrigiert werden müssen. Jedoch erspart ein solches iteratives Vorgehen idealerweise viele Labelschritte. Durch Wiederholung dieses Vorgehens können Datensätze schnell erweitert werden.
Zusätzlich zu den 100 gelabelten Fotos sind diese mit Augmentierungen zu versehen. Dies kann mit Bildverarbeitungsbibliotheken erfolgen oder alternativ mit einem Labelingtool, wenn dieses Bildverarbeitungsfunktionen bietet. Labels müssen hierfür nicht neu erstellt werden, da sie entweder an den gleichen Positionen bleiben, z. B. beim Verrauschen oder bei Beleuchtungsänderungen von Bildern, oder aber bei Zoom oder Rotation durch mathematische Operationen angepasst werden können.

Für das Labeling wird die Software RectLabel verwendet. Hierfür entstehen geringe monatliche Lizenzkosten in Höhe von 2,99 $. Dafür bietet diese Software wichtige Möglichkeiten beim Export und dem iterativen Vorgehen.

Nach dem Öffnen eines Ordners wird noch ein Pfad für Annotationen benötigt. Dies kann der Ordner sein, in dem auch die Bilder abgelegt sind. Der Labelvorgang besteht standardmäßig aus folgendem Ablauf:

1. Bereich markieren
2. Label vergeben
3. bestätigen
4. weiter zum nächsten Bild

Durch zusätzliche Einstellungen kann der Vorgang beschleunigt werden: Die Bestätigung kann ausgeschaltet und das zuletzt verwendete Label übernommen werden. Diese reduzierten Labelschritte müssen dann für jedes Label über den gesamten Datensatz ausgeführt werden. Dadurch, dass die Labelbenennung nur einmalig am Anfang des Datensatzes erfolgt, sind je Bild nur eine Markierung und zwei Tastendrücke erforderlich. In Verbindung mit einem iterativen Vorgehen können mit diesem Tool schnell große Datensätze gelabelt werden. Für das iterative Vorgehen muss ausreichend schnelle Hardware vorhanden sein, damit das Trainieren kleiner Netze nicht mehr Zeit in Anspruch nimmt als das manuelle Labeln dieser Daten.
Neben dem Labelvorgang zeichnet sich RectLabel auch durch seine Exportfunktionen aus. Zunächst liegen die Labels in einer XML-Datei vor. Die Software kann daraus automatisch die Labeldaten für CreateML, YOLO, Tensorflow und COCO exportieren. Eine manuelle Konvertierung ist daher nicht erforderlich.

4.3.3. Augmentierungen

Die Augmentierungen werden durch ein Skript vorgenommen, welches sowohl die Bilder verändert als auch die zugehörigen XML-Dateien anpasst, damit diese Bilder nicht einen zusätzlichen Labelingaufwand darstellen. ML-Bibliotheken bieten zum Teil Augmentierungsfunktionen. Diese haben den Vorteil, dass nicht manuell eine Bildbearbeitung programmiert werden muss. Allerdings hat man keine Kontrolle über die Ausgabewerte der Funktionen, das heißt, eine automatische Anpassung der Labels kann problematisch sein. Es soll möglichst auf Augmentierungsfunktionen zurückgegriffen werden, die keine Anpassung der Labels erfordern oder Rückgabewerte liefern, die eine Umrechnung der Labelpositionen ermöglichen. Falls weitere Augmentierungen notwendig sind, ist eine Implementierung mit der Bildverarbeitungsbibliothek OpenCV notwendig.

4.3.4. Darstellung der Bounding Boxes

Durch die Labelingsoftware wird pro Bild eine XML-Datei angelegt, die mindestens folgende Informationen enthält:

- Dateiname
- Labelname
- X- und Y-Position des Labels

Je nach verwendeter Software können standardmäßig weitere Daten enthalten sein. Bei Daten, die mit RectLabel gelabelt wurden, sieht das Format folgendermaßen aus:

```
<annotation>
        <folder>sicherung1_frames</folder>
        <filename>sicherung1-000005.jpeg</filename>
        <size>
                <width>450</width>
                <height>800</height>
                <depth>3</depth>
        </size>
        <object>
                <name>FI</name>
                <bndbox>
                        <xmin>96</xmin>
                        <ymin>172</ymin>
                        <xmax>156</xmax>
                        <ymax>236</ymax>
                </bndbox>
        </object>
</annotation>
```

Listing 4.1: Bounding Box im XML-Format

Je nach genutztem ML-Framework werden die Labels in verschiedenen Formaten erwartet. Im Falle von CreateML ist ein definiertes JSON-Format nötig, welches die Label aller Bilddateien enthält. Dies bedeutet, dass die Daten aller betreffenden XML-Dateien in eine neue JSON-Datei integriert werden müssen. Das Zielformat muss folgender Struktur entsprechen:

```
[
        {
        "imagefilename": "sicherung1-000005.jpeg",
```

```
4         "annotation":
5         [
6         {
7                 "label": "FI",
8                 "coordinates":
9                 {       "y": 204.0,
10                        "x": 126.0,
11                        "height": 64.0,
12                        "width": 60.0
13                }
14        }
15        ]},...]
```

Listing 4.2: Bounding Box im CreateML JSON-Format

In der JSON-Struktur ist ein Array enthalten, welches die Namen aller genutzten Dateien („imagefilename“) enthält. An jeden Dateinamen ist wiederum ein Array angehängt, welches die zugehörigen Annotationen („annotation“) beinhaltet. In diesem untergeordneten Array sind sowohl Labelnamen als auch deren Koordinaten auf dem Bild enthalten.
In der Software RectLabel ist bereits eine Funktion integriert, die die XML-Dateien in das JSON-Format konvertiert. Alternativ kann dies durch ein Skript geleistet werden, welches die Formatkonvertierung vornimmt. Bei der Konvertierung ist zu beachten, dass die Koordinaten verschieden formatiert sind. In der XML-Datei ist das Label durch ein Rechteck zwischen zwei X-/Y-Koordinaten definiert. CreateML benötigt im JSON-Format hingegen eine Ursprungskoordinate sowie Höhe und Breite. Der Koordinatenursprung ist definiert als der Mittelpunkt des rechteckigen Labels. Um den Ursprung sowie Höhe und Breite zu ermitteln, gilt daher:

$$\begin{aligned} x &= x_{min} + \frac{x_{max} - x_{min}}{2} \\ y &= y_{min} + \frac{y_{max} - y_{min}}{2} \\ height &= y_{max} - y_{min} \\ width &= x_{max} - x_{min} \end{aligned} \tag{4.1}$$

Von der X-/Y-Koordinate wird das Rechteck in positive Richtung um die Breite und Höhe aufgespannt. Abbildung 4.1 zeigt ein rundes Objekt mit einer Bounding Box. Da das manuelle Labeling mit Bounding Boxes schon bei wenigen Bildern einen großen Zeitaufwand erfordert, sind automatische Labelschritte erforderlich. Hierfür können entweder Bildverarbeitungsalgorithmen oder zusätzliche ML-Modelle helfen. Nachfolgend wird untersucht, wie ein iteratives Vorgehen beim Labeling das Erstellen eines Datensatzes für die Objekterkennung beschleunigen kann.

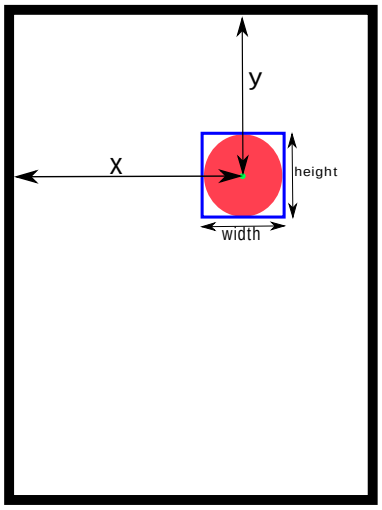


Abbildung 4.1.: Koordinaten der Bounding Box um ein Objekt

4.3.5. Voruntersuchungen

Im Rahmen von Voruntersuchungen soll ermittelt werden, wie sich die Verwendung unterschiedlich großer Trainingsdatensätze auf die Objekterkennung auswirkt. Dabei wird bei jedem erstellten Modell die Qualität der Erkennung über die Werte der AP bei IoU-Thresholds von 50% und 50%-95% gemessen. Da für diese Untersuchungen die Erzeugung verschieden großer Datensätze erforderlich ist, wird parallel untersucht, ob mit einem kleinen Datensatz erstellte Modelle bei der Erzeugung größerer Datensätze helfen können. In einem letzten Schritt werden mit Methoden der ImageDataAugmentation (speziell Farbverschiebung und Beleuchtungsänderungen) künstlich weitere Daten dem Datensatz hinzugefügt und es erfolgt ein erneuter Test der Erkennung.

4.3.6. Beschreibung des Vorgehens

Die Untersuchung am Szenario des Sicherungskastens soll zeigen, wie viele und welche Trainingsdaten benötigt werden, um eine Objekterkennung mit CreateML zu erzeugen, die in einem kontrastarmen Szenario stabil arbeitet. Hierfür wird ein Zeitlupenvideo vom Sicherungskasten aufgenommen. Anschließend werden die Frames aus dem Video extrahiert. Durch die Zeitlupe können mit einem Video besonders viele Frames generiert werden. Das Video soll einen Schwenk von links nach rechts sowie von unten nach oben enthalten, um die Objekte aus unterschiedlichen Winkeln zu zeigen.
Aus diesem Video werden alle Frames exportiert und in einem Ordner „all_frames“ gespeichert. Aus diesem Ordner werden in die Ordner „100th_frames“ und „10th_frames“ jeweils

jedes 100. bzw. jedes 10. Frame aus dem Ordner mit allen Frames exportiert. Diese Ordner werden dann schrittweise genutzt, um die Trainingsdaten zu generieren.
Zunächst werden die Bilder aus dem Ordner mit jedem 100. Bild manuell gelabelt. Dieser Datensatz wird anschließend mit CreateML zum Training eines ersten Modells verwendet. Das erstellte Modell wird in Rectlabel importiert und benutzt, um die Bilder in dem Ordner mit jedem 10. Bild zu labeln. Anschließend werden die Label der Bilder manuell geprüft und ggf. korrigiert. Aus diesen Bildern ergibt sich ein Datensatz, welcher erneut für ein Training genutzt wird. Anschließend wird dieser Vorgang mit dem neuen Modell wiederholt und alle Frames werden gelabelt. So entsteht schrittweise ein großer Datensatz, ohne jedes Bild manuell labeln zu müssen.
Erwartet wird, dass sich sowohl die Iterationszahl als auch die Zahl der verwendeten Bilder auf die Güte der Erkennung auswirken.

4.3.7. Erzeugung der Datensätze

Die Videoaufnahme erfolgt mit einem iPhone 11 Pro im Zeitlupenmodus. Die Aufzeichnung erfolgt in diesem Modus mit 120 Bildern pro Sekunde. Bei einer Aufnahmedauer von 20s ergeben sich bereits 2400 Bilder. Das fertige Video steht im „.mov“-Format in Full-HD-Qualität und „h264“-Codierung zur Verfügung. Aus dieser Videodatei werden mit der Software „ffmpeg“ die einzelnen Bilder extrahiert und als „.jpg“-Dateien gespeichert. Aus Performancegründen wurde dieser Export auf einem High-Performance-Server ausgeführt. Diese Bilddaten werden als Trainingsdaten in RectLabel importiert und iterativ gelabelt.
Das fertige Video enthält 3689 Bilder, die durch den Export nummeriert in einem Ordner abgelegt sind. Es werden weitere Ordner erstellt, in die jedes jeweils 10. bzw. 100. Bild kopiert wird. Mit dem folgenden Befehl kann dieser Prozess automatisiert werden:

```
for file in `find /SOURCE/DIR -type f | awk 'NR %10 == 0'`;
do echo mv $file /DEST/DIR ; done
```

Zunächst werden die Daten aus dem Ordner mit jedem 100. Bild in RectLabel importiert und diese 36 Bilder mit Labels versehen. Anschließend werden die Labels für CreateML in eine .json-Datei exportiert, wonach das Training mit CreateML starten kann. Mit dem trainierten Modell wird anschließend der Ordner mit jedem 10. Bild, insgesamt 368 Bildern, gelabelt. Nach diesem Vorgang wird erneut ein Training durchgeführt und das Modell auf alle Bilder angewendet. Nachdem ein Modell mit allen Bildern trainiert ist, wird das Trainingsergebnis analysiert.

4.3.8. Ergänzung des Datensatzes mit Augmentierungen

Für das Erstellen weiterer Trainingsdaten mit Hilfe von Augmentierungen gibt es, neben den Möglichkeiten der Bildverarbeitungsbibliothek OpenCV, Funktionen in Machine Learning Frameworks (vgl. Albon und Safari 2019), die diese Aufgabe übernehmen. Diese haben den Vorteil, dass kein zusätzlicher Programmieraufwand entsteht, sondern lediglich Parameter entsprechend der Anforderungen einzustellen sind, damit die geforderten Daten generiert werden. Je nach Trainingsart besteht außerdem die Möglichkeit, Daten live während des Trainings zu augmentieren, wodurch kein Vorhalten augmentierter Daten mehr notwendig ist. Im Falle des Trainings mit CreateML ist dies leider nicht möglich, weshalb Daten generiert und gespeichert werden müssen.

4.3.9. Dataaugmentation durch Keras ImageDataGenerator

Der ImageDataGenerator der Keras-Preprocessing-Bibliothek (vgl. Keras 2020) bietet sowohl die Möglichkeit der Liveaugmentierung als auch der Speicherung von generierten, augmentierten Bildern. Letztere Funktionalität ist nötig, um Daten für das Training mit CreateML zu generieren. Mit der „.flow“-Funktion des ImageDataGenerator können aus einem Eingabebild t zufällig augmentierte, weitere Bilder generiert werden. Aus n Eingabebildern entsteht ein Datensatz mit $n + n * m$ Bildern.
Für die Augmentierungen stehen unterschiedliche Parameter zur Verfügung. Dabei ist zu beachten, dass für die augmentierten Bilder die Labels angepasst werden müssen. Bei einfachen Helligkeitsänderungen ist lediglich der neue Dateiname in die XML-Datei einzufügen, während bei Rotationen auch die Koordinaten umgerechnet werden müssen.

4.3.10. Ablauf der Dataaugmentation

Zur Erzeugung von augmentierten Bildern wird ein Pythonskript entwickelt, welches neue Bilddaten generiert und die Labels in der dazugehören XML-Datei anpasst. Um dieses Skript flexibel zu gestalten, werden bei der Ausführung vier optionale Parameter an die Funktion „augment_data(source, dest, copy_xml, total)“ übergeben:

- -i Pfad der Inputdateien
- -o Pfad der Outputdateien
- -x XML-Dateien verarbeiten
- -t Zahl der je Eingabebild generierten, augmentierten Bilder

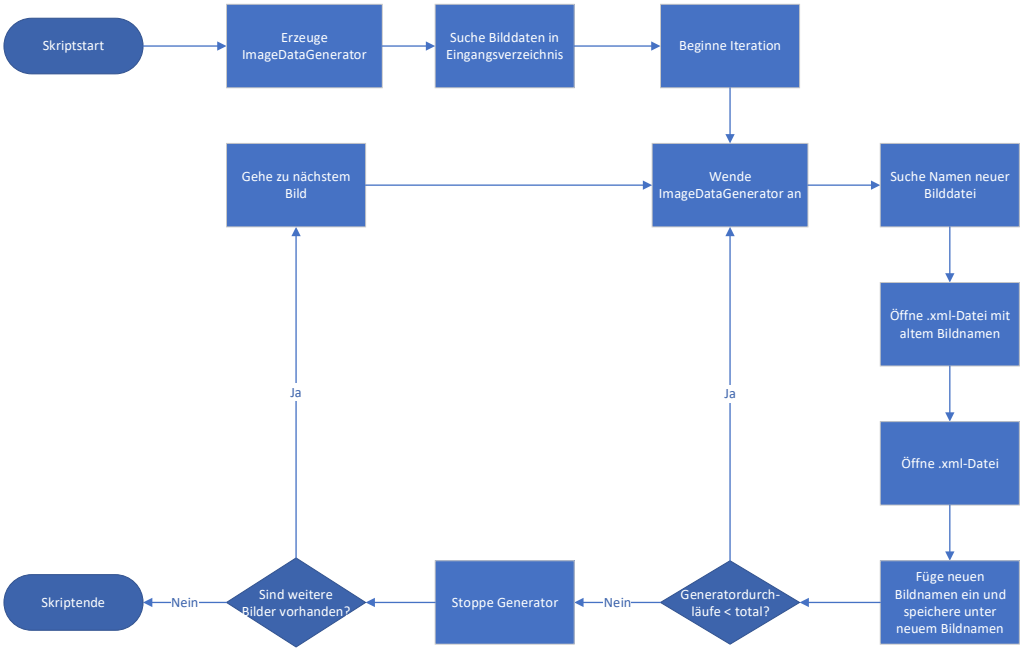


Abbildung 4.2.: Ablaufdiagramm des Augmentierungsskripts

In Abbildung 4.2 ist der Ablauf des Skriptes „augment_data.py“ dargestellt. Zunächst wird ein Objekt des ImageDataGenerator mit den gewünschten Augmentierungsparametern erzeugt, welcher anschließend, mit Hilfe der Flow-Funktion und einer Schleife, so oft vom Eingangsbild neue Zufallsbilder erzeugt, bis der Übergabewert „-t“ erreicht ist. Wird der Parameter „-x“ aktiviert, wird außerdem die zum Eingangsbild zugehörige XML-Datei geöffnet, im Tag „filename“ der Dateiname des neuen Bildes eingefügt und die Datei unter diesem Namen abgespeichert. Eine Anpassung weiterer Tags ist nur notwendig, wenn sich die Position der Labels im Bild verschiebt. Da die Flow-Funktion den Namen des neuen Bildes zufällig generiert und diesen nicht ausgibt, wird nach der Erzeugung eines Bildes durch die Funktion „get_latest_file(path, *paths)“ die zuletzt erstellte Datei im Ausgabeordner gesucht. Dies geschieht anhand des Zeitstempels der Datei im Dateisystem.
Durch die automatische Anpassung der Pfadstruktur kann dieses Skript betriebssystemunabhängig ausgeführt werden, wodurch eine Auslagerung der Datenerzeugung auf leistungsstarke Systeme möglich ist. Um anschließend eine Berechnung in CreateML zu ermöglichen, sind die neuen Daten noch in das JSON-Format der Annotationen (siehe Kapitel 4.3.4) aufzunehmen.

4.3.11. Generierung weiterer Trainingsdaten

Für die Erzeugung weiterer Trainingsdaten werden das Skript sowie die gelabelten Bilder auf einen Server kopiert, damit Daten unabhängig von Arbeitsplätzen generiert werden können. Durch das Skript wird eine hohe CPU-Last bei der Berechnung der Daten erzeugt. Zudem steigt die Berechnungsdauer mit der Datenmenge und Zahl geforderter, augmentierter Bilder.
Durch den ersten Test ohne Augmentierungen steht ein Datensatz von 3689 Bildern zur Verfügung. Für jedes dieser Bilder werden zehn neue Bilder generiert, deren Helligkeit zufällig geändert wurde. Daraus entsteht ein Gesamtdatensatz von $3689 + (3689 * 10) = 40579$ Bildern, der für das Training der Objekterkennung genutzt wird. Durch erneutes Training mit diesem ergänzten Datensatz und eine anschließende Ermittlung der mAP durch die Testdaten wird geprüft, wie sich die Nutzung von Augmentierungen in den Trainingsdaten auswirkt.

4.3.12. Ergebnisse

Das trainierte Modell wird gegen einen Testdatensatz mit unterschiedlichen Beleuchtungen getestet. Die Testdaten waren nicht in den Trainingsdaten enthalten, d. h. das Modell wird gegen ihm unbekannte Daten getestet. Als Vergleichswert werden mAP50% als auch VariedmAP mit einem Durchschnitt über alle enthaltenen Klassen berechnet.

Tabelle 4.10.: Ergebnis der Voruntersuchungen

Bilddaten	Augmentierungen	Validierung mAP 50%	Validierung mAP	Testdaten mAP 50%	Testdaten mAP
36	nein	zu wenig Daten	zu wenig Daten	89%	60,25%
368	nein	100%	88,25%	96%	59,25%
3689	nein	100%	90,25%	98%	55,5%
40.560	ja, Farbverschiebungen	100%	89,5%	100%	62%

Der Vergleich der mAP-Werte (siehe Tabelle 4.10 und Abb. 4.3) zeigt, dass durch das Hinzufügen von Augmentierungen die mAP verbessert werden kann. Dies soll ausgenutzt werden, um eine stabile Erkennung in der Anwendung zu ermöglichen. Der Unterschied der mAP bei Validierungs- und Testdaten deutet darauf hin, dass Overfitting vorliegt, welches mit augmentierten Daten verringert werden konnte.

4.3.13. Zusammenfassung Machine-Learning-Konzept

Das Konzept für das Training der Objekterkennung besteht aus mehreren Schritten:

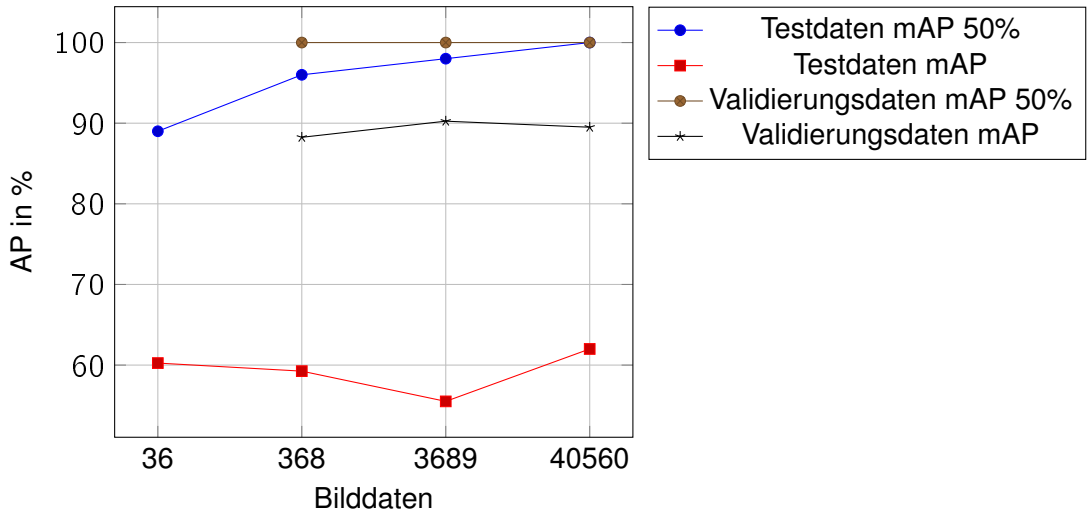


Abbildung 4.3.: Diagramm der AP der Validierungs- und Testdaten

1. Erstellen von Fotos aus unterschiedlichen Blickwinkeln und unter verschiedenen Belichtungsverhältnissen
2. Erzeugung zusätzlicher Variabilität durch Augmentierungen
3. Iteratives Labeln der Bilder mit RectLabel
    a) Kleinen Datensatz erstellen
    b) Training mit CreateML
    c) Vorlabeln weiterer Daten mit trainiertem Netz in RectLabel
4. Praxistest des mit CreateML trainierten Netzes auf dem iPad

Das ML-Modell soll anschließend in einer Anwendung implementiert werden, die Zusatzinformationen zu Objekten in einem Schaltschrank liefern kann.
Mit CreateML trainierte Modelle bieten den Vorteil, dass diese automatisch optimiert werden und wenig Speicherplatz belegen. Die Größe trainierter Modelle im Rahmen der Voruntersuchungen bewegt sich zwischen 60 und 70 MB.

4.4. Userinterface

Nachfolgend wird ein Konzept erarbeitet, wie das UI der Anwendung aussehen soll. Dieses Konzept basiert auf den Anforderungen aus Kapitel 3.6.

4.4.1. Interaktion

Die wesentliche Bedienung von Tablets und Smartphones beschränkt sich auf den Touchscreen. Mit dieser Bedienungsweise ist nahezu jeder Anwender vertraut. Daher wird diese Art auch für die Anwendung gewählt. Ein weiterer Vorteil besteht darin, dass der Nutzer direkt mit den Einblendungen interagieren, sich zusätzliche Informationen anzeigen lassen und die Einblendungen wieder schließen kann.
Eine Implementierung von Sprachbefehlen und Gesten ist nicht vorgesehen, da die Nutzung von Sprachbefehlen Kenntnis über implementierte Befehle voraussetzen würde. Insbesondere die Erkennung von Gesten würde eine Einstiegshürde beim Benutzer schaffen, da diese Interaktionsart die Kenntnis der möglichen Gesten erfordern würde.

4.4.2. Darstellung

Die Darstellung der Anwendung soll nur die nötigsten Einblendungen enthalten, damit das Display nicht überfrachtet wird und die Anwendung handhabbar bleibt. Im Falle des Sicherungskastens soll durch Antippen der einzelnen Elemente die Funktion der jeweiligen Sicherung bzw. des Schalters eingeblendet werden. Die Einblendung soll dabei auf einer freien Fläche in der Nähe der Schaltflächen erfolgen. Welche Inhalte dargestellt werden, wird in Kapitel 4.4.3 beschrieben.
Die Markierung der Schaltflächen soll sich farblich absetzen und eine Umrandung aufweisen. Dazu soll eine blasse Farbe gewählt werden, die semitransparent ist, damit das Videobild sichtbar bleibt. Nach dem Antippen einer Schaltfläche soll eine kurze, farbliche Rückmeldung erfolgen, damit der Nutzer weiß, dass die Anwendung auf die Eingabe reagiert hat. Die zugehörige Einblendung der Annotation soll kontrastreich erfolgen, z. B. durch schwarze Schrift auf weißem Hintergrund.

4.4.3. Dargestellter Inhalt

In dieser Arbeit soll gezeigt werden, wie eine Objekterkennung für AR-Anwendungen arbeiten kann. Um die Erkennung testen zu können, werden zunächst die Objekte mit einem Rahmen umrandet und die Bezeichung der erkannten Objekte wird eingeblendet.
Nachdem die Erkennung stabil arbeitet, werden die gelabelten Bereiche als Schaltflächen verwendet, um weitere Informationen anzeigen zu können. Dazu zählt z. B. die Einblendung eines Schaltplans zum gewählten Objekt. Folgende Informationen sollen eingeblendet werden:

- Kurze Funktionsbeschreibung des Bauteils

- Schaltplan einblenden, falls vorhanden (nach Antippen des Bauteils)
- *Optional für zukünftige Versionen: Monitoringinformationen zum Bauteil*

4.5. Zusammenfassung

Die Anwendung wird für iOS entwickelt. Als Entwicklungsgerät steht ein iPad Air der dritten Generation zur Verfügung. Dieses besitzt einen A12-Prozessor und stellt somit ausreichend Leistung für AR-Anwendungen bereit. Gleichzeitig ist die Entwicklung durch die Displaygröße komfortabel und Probleme können einfacher identifiziert werden. Als Programmiersprache kommt die moderne Sprache Swift 5 in Verbindung mit der Entwicklungsumgebung xCode zum Einsatz.
Um das benötigte Machine-Learning-Modell zu erzeugen, werden Trainingsdaten benötigt. Die Aufnahme dieser Daten kann mit einer Handykamera erfolgen. Es können einzelne Fotos aus unterschiedlichen Blickwinkeln oder ein Schwenk in einem Video aufgenommen werden. Das Video hat den Vorteil, dass bei einem Export der Frames eine große Menge Bilddaten entsteht. Der Export kann z. B. mit der Software FFMPEG erfolgen. Diese Daten werden mit der Software „RectLabel“ mit den zugehörigen Labels versehen. Durch ein Pythonskript wird der Datensatz mit augmentierten Bildern vergrößert und anschließend erfolgt das Training des Modells mit CreateML. Daten der App, die variabel abrufbar sein sollen, werden in einer MongoDB hinterlegt.

5. Realisierung

In diesem Kapitel wird gezeigt, wie die einzelnen Komponenten der Anwendung entwickelt werden und wie deren Funktionalitäten ineinandergreifen.

5.1. Programmablauf

Abbildung 5.1 zeigt den Programmablauf der fertigen Anwendung. Beim Programmstart wird eine Sequenz von Befehlen abgearbeitet, die die Lauffähigkeit auf dem Gerät sicherstellt. Dazu gehört u. a. die Abfrage der Kameraberechtigungen. Nach dem Start der Anwendung wird durch die Klasse „videoCapture“ ein Livebild der Kamera im Vollbildmodus angezeigt. Die aufgenommenen Einzelbilder werden in das Format 416 x 416 Pixel gebracht und durch die Objekterkennung verarbeitet. Aus den Ausgabewerten des ML-Modells werden die zu den jeweilig erkannten Klassen gehörenden Bounding Boxes ermittelt und die Schaltflächen auf Basis dieser Daten generiert. Die Darstellung der Schaltflächen erfolgt in semitransparenter Überlagerung des Kamerabildes. Der Prozess vom Anpassen der Bildgröße über die Objekterkennung bis zur Erzeugung der Schaltflächen läuft kontinuierlich ab.
Das Drücken einer Schaltfläche löst weitere Programmaktionen aus. Zunächst wird abhängig davon, welche Schaltfläche gedrückt wurde, eine Datenbankabfrage generiert und an die MongoDB geschickt. Der Inhalt des Ergebnisses der Abfrage ist abhängig von den hinterlegten Informationen. Als Minimum wird ein Textoverlay generiert, welches eine kurze Beschreibung des ausgewählten Objekts enthält. Zusätzlich befindet sich im Overlay eine kleine Schaltfläche, mit der sich die Einblendung schließen lässt und es können andere Einblendungen abgerufen werden. Alternativ lassen sich Einblendungen direkt wechseln, indem eine andere Schaltfläche gedrückt wird.

5.2. Klassendiagramm

Bei der Entwicklung der Anwendung wurden fünf Klassen entwickelt, deren Funktionen hier kurz vorgestellt wird.

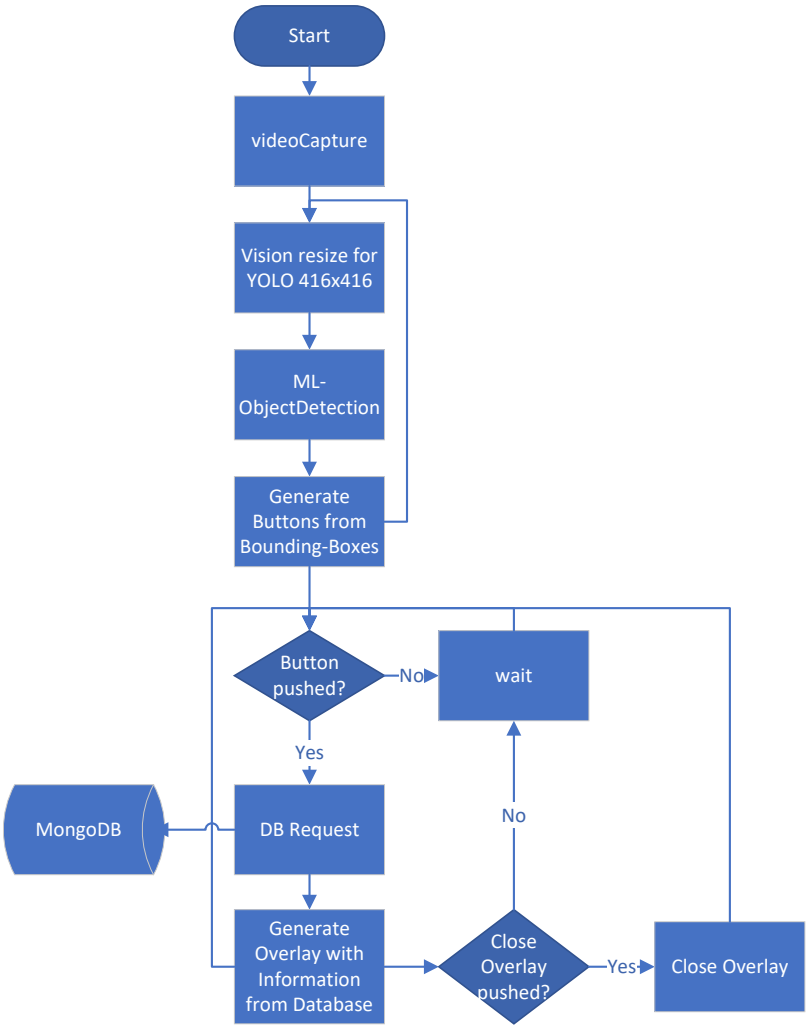


Abbildung 5.1.: Der Programmablauf der Anwendung

Die Klasse „AppDelegate“ (Abbildung 5.2) ist eine Standardklasse, die automatisch angelegt wird und die Funktion der Anwendung handhabt. Dazu gehören Startoptionen und die Funktionsweise der Anwendung im Vorder- oder Hintergrund.

Die Klasse „CoreML_Handler“ (Abbildung 5.3) enthält Funktionen, die für die Einbindung des ML-Modells notwendig sind.
In der Klasse „CoreML_DrawButtonView“ (Abbildung 5.4) sind alle Funktionen enthalten, die aus den Daten des ML-Modells das Userinterface erzeugen. Neben der Erzeugung und Darstellung von Schaltflächen und Annotationen werden hier auch die Eingaben verwaltet. Weiterhin ist diese Klasse dafür zuständig, dass bei Bedarf Informationen zum angefragten Schaltschrank mit der Funktion „getElements“ aus der Datenbank abgerufen und in Arrays gespeichert werden, die als lokaler Cache dienen.

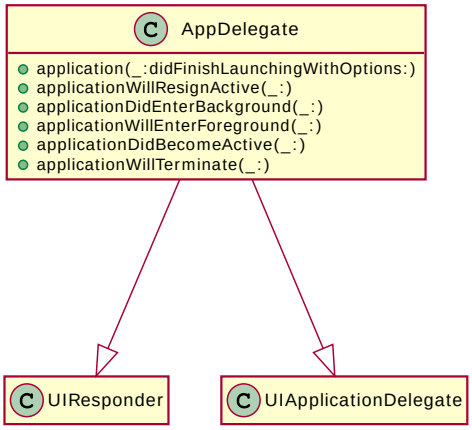


Abbildung 5.2.: Klasse AppDelegate

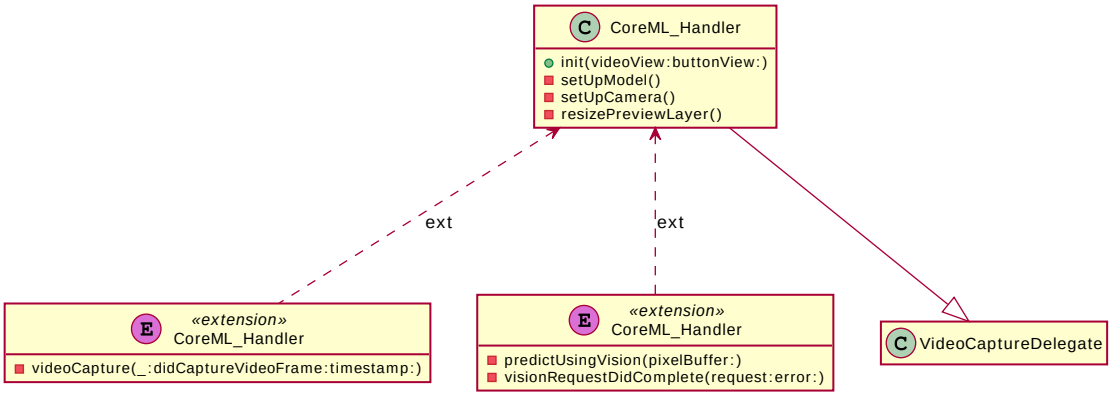


Abbildung 5.3.: Die Klasse CoreML_Handler

Die Klasse „VideoCapture“ (basierend auf Bokhan 2020) verwaltet die Steuerung der Kamera. Dabei werden Einstellungen der Kamera vorgenommen – hier z. B. die Einstellung der Kamera auf VGA-Auflösung mit 640x480 Pixeln und die Festlegung der Bildrate. VGA-Auflösung ist für diesen Anwendungsfall ausreichend, da für die Objekterkennung das Bild automatisch auf 416x416 Pixel skaliert wird.
Der „ViewController“ (Abbildung 5.5) verwaltet die Ansichten in der Anwendung, wobei es hier nur eine Hauptansicht gibt. In dieser Ansicht wird jeweils ein Objekt der Klassen VideoCapture und Coreml_Handler erzeugt.

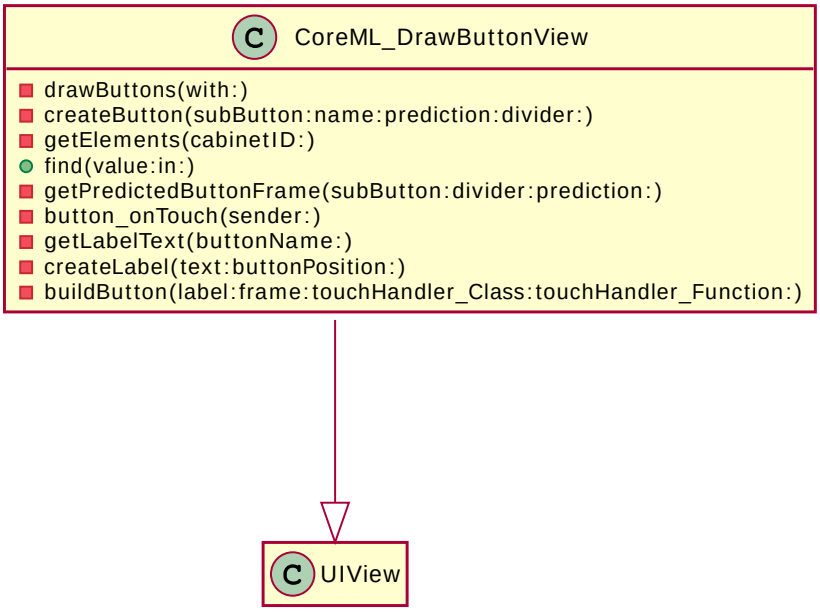


Abbildung 5.4.: Die Klasse CoreML_DrawButtonView

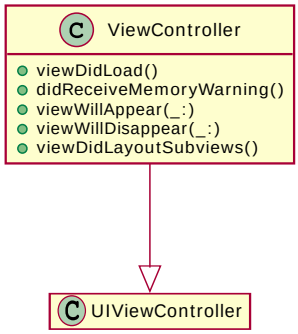


Abbildung 5.5.: Die Klasse ViewController

5.3. Entwicklung der Datenbankanbindung

Um die Anwendung flexibel einsetzbar zu gestalten, ist eine Datenbank in Form der dokumentenorientierten MongoDB vorgesehen (Kapitel 4.2.3). Hierfür ist eine Dokumentenstruktur zu entwickeln, die notwendige Daten erfasst. Um welche Daten es sich handelt, wurde in Kapitel 3.4.5 ermittelt.
Für die Datenbank muss zunächst ein Account in der Atlas Cloud angelegt werden, wodurch der Installations- und Administrationsaufwand entfällt.

5.3.1. Erstellung der Datenbank

Der Inhalt der Datenbank liegt in einer JSON-Datei vor. Durch den Befehl „mongoimport“ wird diese JSON-Datei in die MongoDB importiert. Durch den Import wird automatisch das Dokument angelegt und bereitgestellt. Damit die Daten auch abgerufen werden können, sind noch die Zugriffsrechte zu setzen. Zur Absicherung wird in der Datenbank ein Account angelegt, der ausschließlich Lesezugriffe ermöglicht. Schreibzugriffe werden für die Anwendung nicht benötigt.

5.3.2. Struktur der Datenbank

Als Datenbank wurde exemplarisch ein Schaltschrank mit dem Namen „Home“ implementiert, wobei dieser Name gleichzeitig der „SwitchCabinetID“ entspricht. Jeder implementierte Schaltschrank enthält ein Feld mit dem Namen „Elements“, welches ein Array enthält, wo alle in der Objekterkennung implementierten Klassen hinterlegt sind. Das Feld „ElementID“ entspricht dabei dem Namen der Klasse. Zu jeder Klasse wird ein Schaltplan oder ein Verweis auf diesen in einem Feld „WiringDiagram“ hinterlegt. Hinzu kommt ein Feld „Type“, in dem die Information zum Bauteiltyp hinterlegt sind und ein weiteres Feld „NumberOfSubelements“, welches eine Angabe über die Anzahl von Subelementen enthält. Dieser Wert wird verwendet, um erkannten Bounding Boxes in kleinere Schaltflächen zu unterteilen. Subelemente sind bei Klassen wie „Block1“ vorhanden, wo in der Objekterkennung mehrere Bauteile in einer Klasse zusammengefasst wurden (siehe Abbildung 5.7). Für diese Bauteile ist ein Array „Subelements“ definiert, welches ein Infofeld enthält sowie eine „SubelementID“, die der eindeutigen Position im Schaltschrank entspricht. Position und Info entsprechen in diesem Szenario den Daten aus Tabelle 3.2. Als Zusatzinformation wird der Typ der Sicherung hinterlegt, um einem Techniker die eindeutige Identifikation zu erleichtern. Bei einem Austausch gegen ein identisches Ersatzteil kann diese Information hilfreich sein. Abbildung

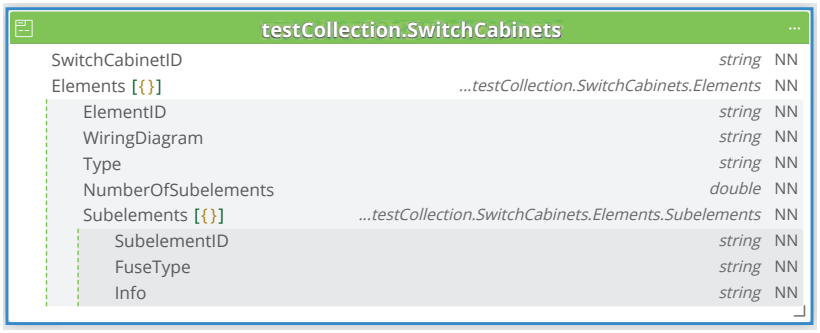


Abbildung 5.6.: Die Struktur der MongoDB

5.6 zeigt den Aufbau der fertigen Datenbank. Durch die Struktur können die Daten einzelner

Bauteile, ganzer Bauteilgruppen oder des ganzen Schaltschranks mit jeweils einer Abfrage abgerufen werden.

5.3.3. Anbindung der Datenbank

Die Datenbank wird über das Stitch Framework angesprochen. Durch den Mongoclient wird auf die Datenbank „ma“ mit der Collection „master“ zugegriffen. Für den Abruf ist eine Authentifizierung notwendig, z. B. durch Benutzername und Passwort oder einen API-Schlüssel. Anschließend wird auf der Collection eine Abfrage ausgeführt, die das zum Schaltschrank zugehörige Dokument zurückliefert. Erst nachdem die Informationen geladen wurden, werden Schaltflächen generiert.

5.3.4. Lokaler Cache der Datenbank

Die Anwendung soll weitestgehend auch unabhängig von der Qualität der Netzwerkverbindung arbeiten. Damit diese nicht mit jeder Interaktion belastet wird, wird bei Erkennung eines Schaltschranks eine lokale Kopie des Dokuments aus der MongoDB als Cache angelegt. Nachdem vorhergehend das Dokument aus der Datenbank abgerufen wurde, werden die Werte in lokale Arrays kopiert. Bei jeder Abfrage wird zunächst geprüft, ob im lokalen Cache die gesuchte Information zu finden ist. Erst wenn diese nicht lokal vorhanden ist, wird eine neue Anfrage an die Datenbank geschickt. Auf diesem Weg wird die Netzwerklast minimiert, die Abhängigkeit von der Verbindung reduziert und die I/O-Auslastung der Datenbank verringert.

5.4. Entwicklung und Einbindung der Objekterkennung

Als zentrales Element der Anwendung bildet die Objekterkennung die Grundlage für die Funktionalität. Entsprechend der Voruntersuchungen wird ein Modell mit CreateML trainiert. Dieses basiert auf Trainingsdaten, die sowohl reale Aufnahmen als auch augmentierte Bilder enthalten. Aufgrund der Ergebnisse in den Voruntersuchungen wird kein neues Modell trainiert, sondern das bestehende verwendet. Das Vorgehen hierzu wurde in Kapitel 4.3.5 beschrieben.

5.4.1. ML-Modell

Bei dem trainierten ML-Modell handelt es sich um einen YOLO-Object-Detector. Dieser ist im Format einer „mlmodel“-Datei vorhanden und kann direkt eingebunden werden. Für die Funktionalität sind folgende Eingabewerte erforderlich:

- Image (Color 416 x 416)
- iouThreshold (Standard: 0.45)
- confidenceThreshold (Standard: 0.25): Minimaler Confidencewert, damit eine Output Bounding Box erzeugt wird

Mit der Eingabe von Bildern und optionalen Thresholds erzeugt das Modell folgende Ausgabearrays:

- confidence MultiArray (Double 0 x 4): Boxes Class confidence
- coordinates MultiArray (Double 0 x 4): Boxes [x, y, width, height] (relativ zur Bildgröße)

Die Übergabe von Werten für iouThreshold und confidenceThreshold ist nur notwendig, wenn von den Standardwerten abweichende Werte für das Programm benötigt werden. Für die Objekterkennung ist für das ML-Modell ein Bild im quadratischen Format von 416x416 Pixeln erforderlich. Dieses Format entspricht in der Regel nicht dem Format des Videobildes der Kamera, weshalb die Eingabebilder auf die passende Größe zugeschnitten werden müssen. Im „Vision-“ Framework ist bereits eine Funktion vorhanden, die diese Aufgabe übernimmt.
Das Bild der Kamera wird durch eine Videocapture-Klasse aufgenommen. Hier lässt sich auch die Auflösung festlegen. Diese wird als VGA-Auflösung mit 640x480 Pixeln gewählt. Größere Auflösungen sind nicht sinnvoll, da eine Anpassung auf die Eingabegröße erfolgen muss und die Wahl einer kleinen Auflösung die zu verarbeitende Datenmenge verringert.
In der Anwendung werden für das Szenario des Sicherungskastens vier Klassen benötigt (FI, Hauptschalter, Block1, Block2). Als Ausgabewert liefert das Modell daher für jede dieser vier Klassen einen Confidence-Wert. Die zugehörige Bounding Box wird, relativ zur Bildgröße, durch eine X-/Y-Koordinate sowie Breite und Höhe definiert.

5.4.2. Einbindung des ML-Modells

Die vom ML-Modell gelieferten Informationen bilden die Grundlage für den Aufbau des UI (Kapitel 5.5). Durch das Vision Framework werden die Predictions des ML-Modells abgefragt. In der Standardeinstellung werden nur Bounding Boxes generiert, wenn der Confidencewert mindestens 0.25 erreicht. Dabei kann es vorkommen, dass eine Klasse doppelt,

mit verschiedenen Confidencewerten generiert wird. Um diesen Fall abzufangen, wird definiert, dass jede Klasse nur einmal im Bild auftauchen darf. Werden zwei Bounding Boxes derselben Klasse generiert, wird nur jene mit dem größeren Confidencewert verarbeitet und zur Erzeugung der UI herangezogen.

5.5. Entwicklung des Userinterfaces

Das Userinterface wird mit dem Framework UIKit erzeugt. Der Aufbau in der Anwendung ist zweischichtig, wobei eine Schicht aus dem Videobild (videoView) besteht und in einer überlagerten Schicht die Interaktionsschicht implementiert wird. Die Interaktionsschicht enthält dabei alle Schaltflächen und Einblendungen. Die Positionen der Schaltflächen leiten sich aus der Objekterkennung ab. Die Klassen „FI“ und „Hauptschalter“ stellen dabei Schaltflächen dar. Bei den Klassen „Block1“ und „Block2“ ist eine zusätzliche Unterteilung in kleinere, gleichgroße Schaltflächen entsprechend der Anzahl der Sicherungen je Block erforderlich. Dazu wird die erkannte Bounding Box in gleichgroße Teile unterteilt, wobei jeder Teil als einzelne Schaltfläche dient. Abbildung 5.7 zeigt die Unterteilung des erkannten Sicherungsblocks in einzelne Schaltflächen. Die Einblendung erfolgt in der gleichen Schicht neben der Schaltfläche.

Abbildung 5.7.: Unterteilung von Block1 in mehrere Schaltflächen

Die Unterteilung in zwei Schichten ist notwendig, damit das Kamerabild dauerhaft aktiv bleiben kann, ohne Interaktionen zu behindern.
Da bei der Eingabe auf Touchscreens nicht auf jedem Gerät eine haptische Rückmeldung z. B. durch einen Vibrationsmotor erfolgen kann, wird eine optische Rückmeldung implementiert. Im nicht gedrückten Modus wird eine Schaltfläche durch ein hellgraues, semitransparentes Feld markiert. Wird die Schaltfläche gedrückt, ändert sich die Farbe in ein dunkles Grau.
Eine weitere grafische Hervorhebung ist für eine Abgrenzung zwischen den Schaltflächen notwendig, damit auch bei nah beieinanderliegenden Flächen eine eindeutige Zuordnung des Interaktionsbereichs für den Anwender möglich ist. Hierfür kann eine beliebige Farbe

gewählt werden, welche einen guten Kontrast zu Hintergrund und Schaltfläche bietet. Aufgrund des weißen Hintergrunds und der schwarzen bzw. blauen Elemente an den Bauteilen wird hier ein roter Rahmen für jede Schaltfläche gewählt.
Die Einblendungen sollen in einem Bereich auftauchen, in dem die Bedienung nicht beeinträchtigt wird. Hierfür ist die Position dynamisch definiert. Abhängig von der Haltung des Tablets erscheint die Einblendung ober- oder unterhalb der gerade gedrückten Schaltfläche.

6. Validierung

Zum Nachweis der Funktionalität von Objekterkennung und Anwendung werden hier einige Fotos der Anwendung in Aktion dargestellt.

6.1. Erkennung der Objekte

Abbildung 6.1 zeigt, dass alle geforderten Elemente des Sicherungskastens erkannt werden. Die Schaltflächen werden über den einzelnen Bauteilen generiert und entsprechen der Größe der Bauteile auf dem Videobild. Jedes einzelne Bauteil wird dabei mit einer rot umrandeten Schaltfläche versehen.

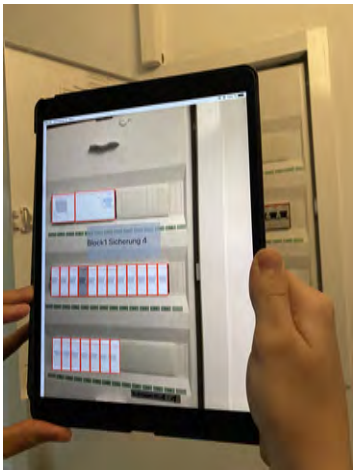
6.2. Interaktionen

Abbildung 6.2a zeigt die Anwendung, nachdem die vierte Sicherung des ersten Blocks ausgewählt wurde. In der Abbildung 6.2b wurde der FI ausgewählt und annotiert. Die Annotationen erscheinen ober- oder unterhalb der jeweiligen zugehörigen Schaltfläche, die nach dem Berühren dunkel gefärbt wird. Der Inhalt der Annotation zeigt an, welche Schaltfläche gedrückt wurde. Demnach konnte erfolgreich nachgewiesen werden, dass die Objekte korrekt erkannt und die Schaltflächen richtig zugeordnet wurden.

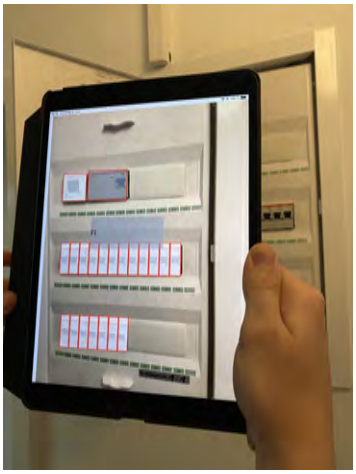
6.3. Test bei verschiedenen Beleuchtungsverhältnissen

In Kapitel 6.1 wurde bereits festgestellt, dass die Erkennung der Bauteile bei guter Ausleuchtung zuverlässig funktioniert. Die Abbildungen 6.3a und 6.3b zeigen die Funktion der Objekterkennung bei verschiedenen Beleuchtungsverhältnissen. Zu erkennen ist, dass die Schaltflächen nicht so präzise über den Bauteilen liegen wie bei guter Ausleuchtung. Bei schwacher Beleuchtung wackelt die Position der Schaltflächen zusätzlich sehr stark.

Abbildung 6.1.: Erkennung aller geforderten Objekte

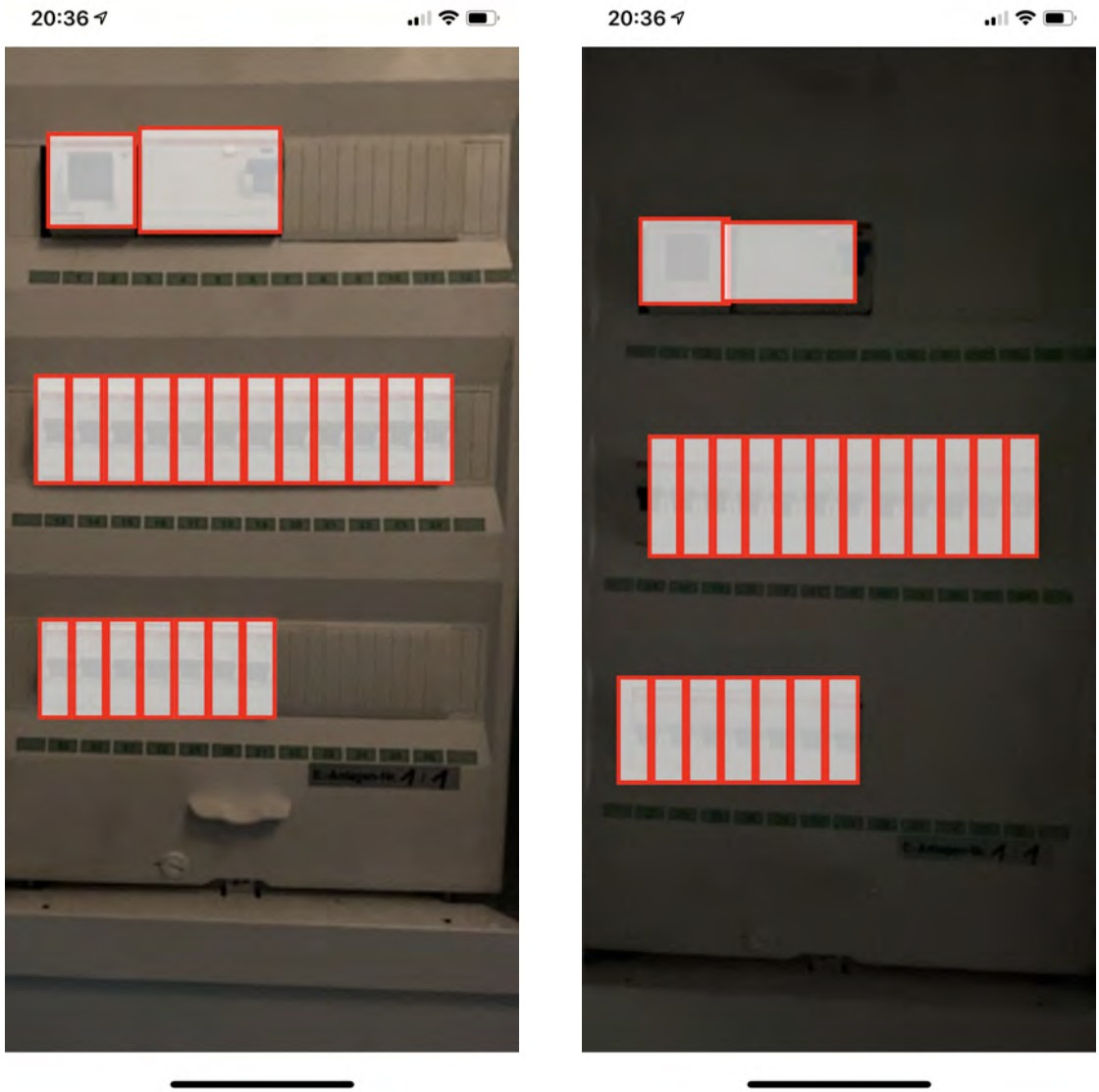


(a) Auswahl der Sicherung 4 von Block 1



(b) Auswahl des FI

Abbildung 6.2.: Interaktionen in der Anwendung



(a) Objekterkennung bei mittlerer Beleuchtung (b) Objekterkennung bei schwacher Beleuchtung

Abbildung 6.3.: Objekterkennung bei verschiedenen Lichtverhältnissen

7. Schluss

Im Folgenden wird der Inhalt dieser Arbeit kurz zusammengefasst und ein Ausblick über weiterführende Möglichkeiten der Anwendung gegeben.

7.1. Zusammenfassung

Zunächst erfolgte in Kapitel 2 ein Überblick über den aktuellen Stand der Technik. Neben einer Übersicht über Hardware, die AR-Funktionen beherrscht, und Softwarekomponenten wurden auch aktuelle Einsatzszenarien in der Industrie dargestellt. Weiterhin wurde aufgezeigt, welche Verfahren zur Objekterkennung mit Machine Learning verbreitet sind und wie hinsichtlich deren Funktionalität eine Vergleichbarkeit hergestellt sowie eine Bewertung vorgenommen werden kann.
Hierfür wurden in Kapitel 3 Anforderungen ermittelt, die für die Funktionalität der Anwendung in Kombination mit einer Objekterkennung erfüllt werden müssen, wobei neben den Anforderungen an die Software auch jene an die Hardware Betrachtung fanden. Dabei wurden die Anforderungen an die einzelnen Elemente der Anwendung so definiert, dass eine Flexibilität hinsichtlich eines Einsatzes in weiteren Szenarien gegeben ist. Für die Idee des Konzepts war es von Bedeutung, weitere Szenarien ergänzen zu können, ohne eine neue Anwendung erstellen zu müssen, und so eine Wiederverwendbarkeit zu gewährleisten. Hierbei wurde auch festgelegt, welche Informationen eine Anwendung als Minimalanforderung enthalten und darstellen soll. Der Informationsgehalt sollte dabei szenarienabhängig skalier- und erweiterbar sein.
Aus den Anforderungen wurde anschließend in Kapitel 4 ein Konzept erarbeitet, indem die einzelnen Elemente aus den zuvor ermittelten Anforderungen betrachtet und jeweils verschiedene Lösungen gegenübergestellt wurden. Besondere Aufmerksamkeit galt softwareseitig der Erstellung der Objekterkennung, die das zentrale Element der Anwendung darstellt. Da es sich um ein spezifisches Szenario handelt, für das noch keine Daten vorlagen, mussten zunächst ausreichende Trainingsdaten für das ML-Modell erzeugt werden. Um den Zeitaufwand hierfür zu minimieren, wurde ein iteratives Vorgehen gewählt. Damit das System unempfindlicher gegen Beleuchtungsänderungen wird, sind künstlich weitere Bilder aus den bereits gelabelten erstellt worden. Dieses Vorgehen führte dazu, dass ein Datensatz aus mehr als 40.000 Bildern generiert werden konnte. Im Rahmen von Voruntersuchungen

wurde ein Modell mit und eines ohne künstliche Daten trainiert. Durch einen Vergleich der mAP beider Modelle in Bezug auf Testdaten konnte bestätigt werden, dass künstliche Daten die Objekterkennung verbessern.
Allgemein wurden im Softwarebereich verschiedene Lösungswege aufgezeigt, die in unterschiedlichen Szenarien sinnvoll sein können. Wie bereits erwähnt, lag in dieser Arbeit ein wesentliches Augenmerk auf der Idee, ein Konzept zu schaffen, welches Wiederverwendbarkeit in verschiedenen Szenarien bietet. Durch den situationsbedingten Wechsel der Versuchsumgebung während der Arbeit wurde nachgewiesen, dass lediglich das ML-Modell und Einträge in der Datenbank angepasst werden müssen, damit die Anwendung auch im neuen Szenario funktioniert.
In Kapitel 5 wurde die Funktionalität der Anwendung hergestellt und der Programmaufbau mit fünf Klassen dokumentiert.
Kapitel 6 widmet sich der Funktionsweise der Anwendung im Betrieb am Schaltschrank. Die Anwendung arbeitet flüssig und die Objekte werden ohne wahrnehmbare Verzögerung erkannt. Bei Bewegung des iPads folgen die Schaltflächen fließend den Bauteilpositionen. Durch einen Versuch bei verschiedenen Beleuchtungsverhältnissen wurden Grenzen des Systems bei sehr schwacher Beleuchtung aufgezeigt.
Beim Antippen der Schaltflächen werden diese korrekt zugeordnet und die Informationen, welche zuvor aus der Datenbank geladen wurden, angezeigt. Optisch besteht noch Verbesserungsbedarf mittels einer dynamischeren Positionierung der Annotationen, da diese teilweise Schaltflächen überlagern können. Für eine intuitive Bedienung sollte außerdem das manuelle Schließen von Annotationen ermöglicht werden. Aktuell erscheinen diese, bis eine andere Schaltfläche gewählt wurde.
Insgesamt konnte gezeigt werden, dass das Konzept einer AR-Anwendung, die auf einer Objekterkennung mit Machine Learning basiert, in Verbindung mit dynamischen Informationen aus einer Datenbank funktioniert. Mit Hilfe der Anwendung kann sofort der Zweck des annotierten Objekts angezeigt werden, ohne in einer Tabelle oder einem Schaltplan nachzuschlagen. Informationen können in der Datenbank in einem entsprechenden Feld hinterlegt werden und stehen dem Techniker bei der nächsten Nutzung der Anwendung zur Verfügung, sodass keine Anpassung der App notwendig wird.

7.2. Ausblick

Damit die Anwendung weitere Verbreitung findet, muss die Plattformunabhängigkeit in den Fokus gerückt werden. Um möglichst viele Nutzer zu erreichen, sollte die Anwendung zusätzlich für Android zur Verfügung stehen. In der aktuellen Version der Anwendung sind ausschließlich grundlegende Informationen sowie ein Verweis auf einen Schaltplan enthalten. Für weitere Versionen kann ein automatischer Aufruf der betreffenden Schaltplanseite erfolgen.

Für die Erweiterung auf zusätzliche Szenarien muss geprüft werden, ob die Anwendung durch Erkennung einzelner Bauteilarten flexibler gestaltet werden kann. Auf diese Weise ist es nicht erforderlich, für jeden Schaltschrank ein neues Modell zu trainieren, sondern lediglich, wenn weitere Bauteilarten hinzugefügt werden. Sollten mehrere Bauteile einer Art in einem Schrank verbaut sein, ist es möglich, über die Anordnung den Schrank und die einzelnen Bauteile zu identifizieren. Durch dieses Vorgehen könnte der Implementierungsaufwand für weitere Schränke stark reduziert werden. Um eine Stabilitätsverbessung bei schlechten Lichtverhältnissen zu schaffen, kann davon profitiert werden, dass die meisten Smartphones und Tablets Kameralichter enthalten, deren Aktivierung auf kurze Entfernung die Ausleuchtung verbessern kann.
Durch die fortlaufende technische Weiterentwicklung ist zu erwarten, dass AR-Geräte auch zukünftig leistungsstärker und kompakter werden. Dies kann in Zukunft dazu führen, dass AR-Anwendungen auf Smartphones oder Tablets weitere Verbreitung finden, um Techniker mit Informationen zu versorgen. HMDs können Zusatzinformationen liefern, während der Anwender beide Hände für Arbeiten zur Verfügung hat. Eine größere Akzeptanz gegenüber heutigen Geräten kann durch eine kompaktere und leichtere Beschaffenheit sowie verbesserte Darstellung erreicht werden.
Neben der Hardware erfährt auch die Software eine fortlaufende Weiterentwicklung. Zum Ende der Arbeit wurde der YOLOv4 Object Detector (vgl. Bochkovskiy, Wang und Liao 2020) vorgestellt, der eine bessere mAP bei größeren Frameraten erreicht. Die Implementierung mit dieser aktualisierten Variante kann ggf. die Objekterkennung verbessern. Dies würde die Anwendung robuster gestalten, wodurch zukünftig ebenfalls eine höhere Akzeptanz sowie hierdurch ein Anstieg in der Verbreitung erreicht werden könnte.
Wie in dieser Ausarbeitung verdeutlicht, führt der Einsatz einer AR-Anwendung im Bereich von Arbeiten an Schaltschränken, wie sie hier konzipiert und beschrieben wurde, zu einer Entlastung von technischem Fachpersonal, verbunden mit einer bedeutenden Zeitersparnis sowie einem positiven Einfluss auf die Kostenreduktion. Weiterhin wäre hierbei von Interesse, das Potential dieser Technologie auf seine Anwendbarkeit innerhalb anderer Fachbereiche zu überprüfen und dahingehend zu erweitern.

Literatur

Albon, Chris und an O'Reilly Media Company Safari (2019). *Machine Learning Kochbuch*. German. ISBN: 978-3-96009-090-8.

Amazon (2019). *Amazon Alexa Offizielle Webseite: Was Ist Alexa?* https://developer.amazon.com/de-DE/alexa.

Android (2020). *Android License*. en. https://source.android.com/license?hl=de.

Androidmag (Apr. 2014). *Wahre Sensibelchen: ein Überblick über wirklich alle Handy-Sensoren – auch über kaum bekannte*. de-DE.

Apple (2019a). *ARKit Augmented Reality*. en. https://developer.apple.com/augmented-reality/.

Apple (2019b). *Siri Sprachassistent*. de-DE. https://www.apple.com/de/siri/.

Apple (2019c). *Swift – Apple (DE)*. de-DE. https://www.apple.com/de/swift/.

Becker, Wolfgang und Frank Brinkmann (2000). *Kostenrechnung für die Instandhaltung: Ergebnisse einer empirischen Untersuchung*. de. Bamberg: Otto-Friedrich-Univ. ISBN: 978-3-931810-19-1.

Bell, Blaine, Steven Feiner und Tobias Höllerer (2001). "View Management for Virtual and Augmented Reality". In: *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*. UIST '01. New York, NY, USA: ACM, S. 101–110. ISBN: 978-1-58113-438-4. DOI: 10.1145/502348.502363.

Bochkovskiy, Alexey, Chien-Yao Wang und Hong-Yuan Mark Liao (Apr. 2020). "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *arXiv:2004.10934 [cs, eess]*. arXiv: 2004.10934 [cs, eess].

Bokhan, Eugene (Mai 2020). *Videocapture, https://github.com/eugenebokhan/VideoCapture*.

Bronshtein, Adi (März 2020). *Train/Test Split and Cross Validation in Python*. en. https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6.

Brownlee, Jason (Mai 2019). *A Gentle Introduction to Object Recognition With Deep Learning*. en-US.

Bundesverband WindEnergie, e. V. (2018). *OffshoreBWE e.V.* de. https://www.wind-energie.de/themen/anlagentechnik/offshore/.

Daimler (Nov. 2017). *Intelligente Dialogtechnologie kombiniert mit Augmented Reality: Ask Mercedes: Der virtuelle Assistent hilft sofort weiter*. de. https://media.daimler.com/marsMediaSite/de/instance/ko/Intelligente-Dialogtechnologie-kombiniert-mit-Augmented-Reality-Ask-Mercedes-Der-virtuelle-Assistent-hilft-sofort-weiter.xhtml?oid=30345702.

Dima, Mariza, Linda Hurcombe und Mark Wright (2014). "Touching the Past: Haptic Augmented Reality for Museum Artefacts". en. In: *Virtual, Augmented and Mixed Reality. Applications of Virtual and Augmented Reality*. Hrsg. von Randall Shumaker und Stephanie Lackey. Lecture Notes in Computer Science. Springer International Publishing, S. 3–14. ISBN: 978-3-319-07464-1.

Dulux (Sep. 2017). *The Dulux Visualizer App*. en. https://www.dulux.co.uk/en/articles/dulux-visualizer-app.

EasyAR (2019). *Sdk Free Ar Sdk – Easy AR SDK*. https://www.easyar.com/view/sdk.html.

Edureka (Dez. 2018). *Keras vs TensorFlow vs PyTorch | Deep Learning Frameworks*. en-US.

Elgendy, Mostafa, Tibor Guzsvinecz und Cecilia Sik-Lanyi (Nov. 2019). "Identification of Markers in Challenging Conditions for People with Visual Impairment Using Convolutional Neural Network". en. In: *Applied Sciences* 9.23, S. 5110. ISSN: 2076-3417. DOI: 10.3390/app9235110.

Everingham, Mark u. a. (Jan. 2015). "The Pascal Visual Object Classes Challenge: A Retrospective". en. In: *International Journal of Computer Vision* 111.1, S. 98–136. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-014-0733-5.

FLIR (2019). *FLIR ONE Gen 3 | FLIR Systems*. de-DE. https://www.flir.com/products/flir-one-gen-3/.

Garrido-Jurado, S. u. a. (Juni 2014). "Automatic Generation and Detection of Highly Reliable Fiducial Markers under Occlusion". en. In: *Pattern Recognition* 47.6, S. 2280–2292. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.01.005.

Girshick, Ross (Sep. 2015). "Fast R-CNN". In: *arXiv:1504.08083 [cs]*. arXiv: 1504.08083 [cs].

Girshick, Ross u. a. (Okt. 2014). "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *arXiv:1311.2524 [cs]*. arXiv: 1311.2524 [cs].

Google (2019a). *ARCore - Google Developers*. en. https://developers.google.com/ar.

Google (2019b). *Google Glass*. en. https://www.google.com/glass/start/.

Google (2019c). *okGoogle Assistant – dein persönlicher Helfer*. de-DE. https://assistant.google.com/intl/de_de/.

Green, Bert F. (1961). *Kinetic Depth Effect (KDE). Wallach*.

HTC (2019). *VIVE | Discover Virtual Reality Beyond Imagination*. https://www.vive.com/de/.

Hui, Jonathan (März 2019). *Object Detection: Speed and Accuracy Comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet And...* en. https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359.

Ikea (2017). *IKEA Place*. de-de. https://apps.apple.com/de/app/ikea-place/id1279244498.

Jarosch, Helmut (2016). *Grundkurs Datenbankentwurf*. de. Wiesbaden: Springer Fachmedien Wiesbaden. ISBN: 978-3-8348-1682-5 978-3-8348-2161-4. DOI: 10.1007/978-3-8348-2161-4.

Joshi, Prateek (2015). *OpenCV with Python by Example: Build Real-World Computer Vision Applications and Develop Cool Demos Using OpenCV for Python*. eng. Community Experience Distilled. Birmingham: Packt Publishing. ISBN: 978-1-78528-393-2.

Jota, Ricardo und Hrvoje Benko (2011). "Constructing Virtual 3D Models with Physical Building Blocks". en. In: *Proceedings of the 2011 Annual Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '11*. Vancouver, BC, Canada: ACM Press, S. 2173. ISBN: 978-1-4503-0268-5. DOI: 10.1145/1979742.1979915.

Kagermann, Prof. Dr. Henning, Prof. Dr. Wolfgang Wahlster und Dr. Johannes Helbig (Okt. 2012). "Industrie 4.0 Umsetzungsempfehlungen". de. In: S. 72.

Kalman, R. E. (1960). "A New Approach to Linear Filtering and Prediction". In: DOI: 10.1115/1.3662552.

Keras (2020). *Image Preprocessing - Keras Documentation*. https://keras.io/preprocessing/.

Kölling Martin, Technology (Sep. 2017). *Post aus Japan: AR für die Ohren*. de. https://www.heise.de/tr/artikel/Post-aus-Japan-AR-fuer-die-Ohren-3844617.html.

Koolonavich, Nikholai (Mai 2018). *Gatwick's Augmented Reality Passenger App Wins Awards*. en-US.

Kotlin (Juni 2020). *Kotlin Programming Language*. https://kotlinlang.org/.

Kress, Bernard C. und William J. Cummings (Juni 2017). "Optical Architecture of HoloLens Mixed Reality Headset". In: *Digital Optical Technologies 2017*. Bd. 10335. International Society for Optics and Photonics, 103350K. DOI: 10.1117/12.2270017.

LeCun, Yann u. a. (1990). "Handwritten Digit Recognition with a Back-Propagation Network". In: *Advances in Neural Information Processing Systems 2*. Hrsg. von D. S. Touretzky. Morgan-Kaufmann, S. 396–404.

LIMITED, ESCAPE VELOCITY (2019). *Sternatlas*. de-de. https://apps.apple.com/de/app/sternatlas/id345542655.

Lin, Tsung-Yi u. a. (Feb. 2015). "Microsoft COCO: Common Objects in Context". In: *arXiv:1405.0312 [cs]*. arXiv: 1405.0312 [cs].

Liu, Wei u. a. (2016). "SSD: Single Shot MultiBox Detector". In: *arXiv:1512.02325 [cs]* 9905, S. 21–37. DOI: 10.1007/978-3-319-46448-0_2. arXiv: 1512.02325 [cs].

Marr, Bernard (Juli 2018). *9 Powerful Real-World Applications Of Augmented Reality (AR) Today*. en. https://www.forbes.com/sites/bernardmarr/2018/07/30/9-powerful-real-world-applications-of-augmented-reality-ar-today/.

Martin, Marinus (Juni 2019). *AR-Messung: So messt ihr Gegenstände mit dem Handy aus*. de. https://www.netzwelt.de/news/171499-ar-messung-so-messt-gegenstaende-handy.html.

McCulloch, Warren S. und Walter Pitts (Dez. 1943). “A Logical Calculus of the Ideas Immanent in Nervous Activity”. en. In: *The bulletin of mathematical biophysics* 5.4, S. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259.

Meier, Andreas (2018). *Werkzeuge Der Digitalen Wirtschaft: Big Data, NoSQL & Co: Eine Einführung in Relationale Und Nicht-Relationale Datenbanken*. Essentials. Wiesbaden: Springer Vieweg. ISBN: 978-3-658-20337-5.

Meier, Andreas und Michael Kaufmann (2019). “SQL&NoSQL Databases”. en. In: *SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management*. Hrsg. von Andreas Meier und Michael Kaufmann. Wiesbaden: Springer Fachmedien, S. 201–218. ISBN: 978-3-658-24549-8. DOI: 10.1007/978-3-658-24549-8_7.

Microsoft (2019a). *Cortana Was Ist Cortana?* https://support.microsoft.com/de-de/help/17214/cortana-what-is.

Microsoft (Sep. 2019b). *HoloLens (1st Gen) Hardware*. en-us. https://docs.microsoft.com/en-us/hololens/hololens1-hardware.

Milgram, Paul und Fumio Kishino (Dez. 1994). “A Taxonomy of Mixed Reality Visual Displays”. In: *IEICE Trans. Information Systems* vol. E77-D, no. 12, S. 1321–1329.

MongoDB (Juni 2020). *GridFS — MongoDB Manual*. en. https://docs.mongodb.com/manual/core/gridfs.

Niantic (2016). *Pokemon*. de. https://www.pokemongo.com/de-de/.

Nowitzki, Hans-Peter (2003). *Der wohltemperierte Mensch: Aufklärungsanthropologien im Widerstreit*. de. Walter de Gruyter. ISBN: 978-3-11-017725-1.

NVIDIA (Feb. 2020). *V100S Technical Data*. https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf.

OpenCV (2019a). *Feature Detection and Description - OpenCV-Python Tutorials 1 Documentation*. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/.

OpenCV (2019b). *OpenCV*. https://opencv.org/about/.

Oracle (2019). *Java | Oracle*. https://www.java.com/de/.

Otte, Ralf (2019). *Künstliche Intelligenz für Dummies*. German. ISBN: 978-3-527-71494-0.

Peakfinder (2019). *PeakFinder App*. https://www.peakfinder.org/de/mobile/.

PTC (2019). *Vuforia Industrial Augmented Reality | Vuforia | PTC*. de-DE. https://www.ptc.com /de/products/augmented-reality.

Raschka, Sebastian und Vahid Mirjalili (2018). *Machine Learning mit Python und Scikit-Learn und TensorFlow: das umfassende Praxis-Handbuch für Data Science, Deep Learning und Predictive Analytics*. ger. Übers. von Knut Lorenzen. 2., aktualisierte und erweiterte Auflage. Frechen: mitp. ISBN: 978-3-95845-733-1 978-3-95845-735-5 978-3-95845-734-8.

Redmon, Joseph, Santosh Divvala u. a. (Mai 2016). “You Only Look Once: Unified, Real-Time Object Detection”. In: *arXiv:1506.02640 [cs]*. arXiv: 1506.02640 [cs].

Redmon, Joseph und Ali Farhadi (Dez. 2016). “YOLO9000: Better, Faster, Stronger”. In: *arXiv:1612.08242 [cs]*. arXiv: 1612.08242 [cs].

Redmon, Joseph und Ali Farhadi (Apr. 2018). “YOLOv3: An Incremental Improvement”. In: *arXiv:1804.02767 [cs]*. arXiv: 1804.02767 [cs].

Ren, Shaoqing u. a. (Jan. 2016). “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *arXiv:1506.01497 [cs]*. arXiv: 1506.01497 [cs].

Ritchie, Dennis M. (1993). “The Development of the C Language”. In: *The Second ACM SIGPLAN Conference on History of Programming Languages*. HOPL-II. New York, NY, USA: ACM, S. 201–208. ISBN: 978-0-89791-570-0. DOI: 10.1145/154766.155580.

Rosenblatt, F. (1958). *The Perceptron: A Probabilistic Model for Information Storage and Organization*.

Sielhorst, Tobias, Marco Feuerstein und Nassir Navab (Dez. 2008). “Advanced Medical Displays: A Literature Review of Augmented Reality”. In: *Journal of Display Technology* 4.4, S. 451–467. ISSN: 1551-319X, 1558-9323. DOI: 10.1109/JDT.2008.2001575.

Soldamatic (2019). *Home*. es. https://www.soldamatic.com/.

Statcounter (2019). *Mobile Betriebssysteme - Marktanteile Internetnutzung weltweit bis November 2019*. de. https://de.statista.com/statistik/daten/studie/184335/umfrage/marktanteil-der-mobilen-betriebssysteme-weltweit-seit-2009/.

Stricker, D. und N. Navab (Okt. 1999). “Calibration Propagation for Image Augmentation”. In: *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR’99)*, S. 95–102. DOI: 10.1109/IWAR.1999.803810.

Sutherland, Ivan E. (1968). “A Head-Mounted Three Dimensional Display”. en. In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I on - AFIPS ’68 (Fall, Part I)*. San Francisco, California: ACM Press, S. 757. DOI: 10.1145/1476589.1476686.

Taylor, Luke und Geoff Nitschke (Aug. 2017). “Improving Deep Learning Using Generic Data Augmentation”. en. In: *arXiv:1708.06020 [cs, stat]*. arXiv: 1708.06020 [cs, stat].

Tiu, Ekin (Aug. 2019). *Metrics to Evaluate Your Semantic Segmentation Model*. en. https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2.

Tracking, Advanced Realtime (2019). *ART Advanced Realtime Tracking*. https://ar-tracking.com/.

Tremblay, Jonathan u. a. (Apr. 2018). “Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization”. In: *arXiv:1804.06516 [cs]*. arXiv: 1804.06516 [cs].

VentureBeat (2019). *Prognose zur Anzahl der Smartphone-Nutzer weltweit von 2016 bis 2021 (in Milliarden). Statista*. de. https://de.statista.com/statistik/daten/studie/309656/umfrage/prognose-zur-anzahl-der-smartphone-nutzer-weltweit/.

Volonté, Francesco u. a. (Juli 2011). “Augmented Reality and Image Overlay Navigation with OsiriX in Laparoscopic and Robotic Surgery: Not Only a Matter of Fashion”. en. In: *Jour-*

nal of Hepato-Biliary-Pancreatic Sciences 18.4, S. 506–509. ISSN: 18686974. DOI: 10.1007/s00534-011-0385-6.

Wagner, Patrick (Mai 2018). *Infografik: 4G ist in Deutschland immer noch Neuland*. de. https://de.statista.com/infografik/13887/4g-netzabdeckung-in-europa/.

AR-Watches (2019). *AR Watches - Augmented Reality Commerce Platform*. https://ar-watches.com/.

Wave, Denso (2020). *QRcode.ComDENSO WAVE*. https://www.qrcode.com/en/.

Welch, Greg und Gary Bishop (2006). “An Introduction to the Kalman Filter”. en. In: S. 16.

Widrow, B. u. a. (1960). *Adaptive Ädaline"Neuron Using Chemical "Memistors.".* en.

A. Digitaler Anhang

Inhalt der CD

/Datenbank/

- database.json

/Masterarbeit/

- Masterarbeit_HenrikWortmann.pdf

/Skripte/

- augment_data.py

/Software/

- Enthält den Quellcode des Programms.

/Trainingsdaten/

- Enthält die verwendeten Trainingsdaten ohne Augmentierungen. Vor der Ausführung müssen durch „pod install“ die Abhängigkeiten geladen werden.

/Videos/

- sicherung1.mov
- sicherung2_compressed.mp4

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 17. Juni 2020	
Ort, Datum	Unterschrift