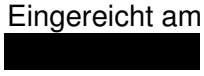
HAW HAMBURG

Masterarbeit

Edgar Wagner

Entwicklung und Evaluation eines praxistauglichen Wundmessungsverfahrens auf der HoloLens 2 mittels Convolutional Neural Networks

Betreuung durch: Prof. Dr. Jan Neuhöfer / Philipp Bresch
Eingereicht am: 31. Juli 2023



Fakultät Design, Medien und Information
Department Medientechnik

Faculty of Design, Media and Information
Department Media Technology

Abstract

Herkömmliche manuelle Methoden der Wundmessung können ungenau sein und erfordern ein hohes Maß an Fachwissen, was zu suboptimalen Behandlungsentscheidungen und längeren Heilungszeiten führen kann. Das Ziel dieser Masterarbeit ist die Entwicklung eines praxistauglichen Wundmessungsverfahrens auf der HoloLens 2 unter Verwendung eines Convolutional Neural Network. Konkret soll ein Prototyp entwickelt werden, der zum einen in der Lage ist, Patientendaten abzurufen und Vitaldaten per Spracheingabe aufzunehmen. Zum anderen soll die Anwendung Wunden fotografieren, analysieren und automatisch präzise Messungen durchführen können. Bei der zu entwickelnden Anwendung soll zudem die allgemeine Nutzbarkeit und die Genauigkeit des Wundmessungsverfahrens empirisch evaluiert werden. Die Ergebnisse der Untersuchung zeigen eine gute Nutzbarkeit der Anwendung, bei der Messgenauigkeit gibt es aber noch Verbesserungsmöglichkeiten.

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original

Inhaltsverzeichnis

Selbstständigkeitserklärung **III**

Abbildungsverzeichnis **VI**

Abkürzungsverzeichnis **VII**

Tabellenverzeichnis **VIII**

Listingverzeichnis **IX**

1 Einleitung **1**

1.1 Motivation 1
1.2 Zielsetzung der Arbeit 1
1.3 Aufbau der Arbeit 2

2 Grundlagen **3**

2.1 Augmented Reality 3
2.2 HoloLens 2 5
2.3 Unity 3D 7
2.4 Künstliche neuronale Netzwerke 11
2.4.1 Convolutional Neural Networks 13
2.4.2 Mask R-CNN 15
2.5 Klassische Wundmessungsverfahren 18

3 Stand der Technik **20**

4 Konzept **24**

4.1 Funktionale Anforderungen 24
4.2 Nicht-funktionale Anforderungen 25

5 Implementierung **27**

5.1 Patient erkennen 28
5.2 Aufgaben auswählen 30
5.3 Spracherkennung 30

5.4 Wundmessungsverfahren 32
5.4.1 Mask R-CNN Training 33
5.4.2 Berechnung der Wundgrößen 36

6 Evaluation **42**
6.1 Versuchsaufbau 42
6.2 Versuchsdurchführung 43

7 Ergebnisse **46**
7.1 Fragebogen 46
7.2 Messgenauigkeit 47

8 Diskussion **50**

9 Fazit **53**

A Fragebogen **54**

Literatur **57**

Anhang USB-Stick **61**

Abbildungsverzeichnis

2.1 Vereinfachte Darstellung des RV-Kontinuums 3
2.2 HoloLens 2 5
2.3 Spatial Mapping der HoloLens 2 6
2.4 Globales Koordinatensystem einer Szene, mit einem darin befindlichen Spielobjekt und dessen lokalen Koordinatensystems. 8
2.5 Button-Prefab des MRTK 10
2.6 Grundstruktur von ANNs 12
2.7 Vermehrte Bilder aus einer Datei 13
2.8 Aufbau eines CNN 13
2.9 Faltungsoperation im Convolutional Layer 14
2.10 Mask R-CNN 16
2.11 Von einem Mask R-CNN segmentierte Objekte 17
2.12 Körperachsen-Methode 19
2.13 Perpendikular-Methode 19
3.1 Von Pflegekräften erstellte Notizen 22
3.2 Manuelle Messung einer Wunde mit virtuellem Maßband 22
5.1 QR-Code 29
5.2 Patienten-Daten 29
5.3 Visitenaufgaben-Auswahl 31
5.4 Aufnahme der Vitaldaten per Sprache 31
5.5 Auswahl von Trainingsdatensätzen 33
5.6 Fotomodus 36
5.7 Scantips 38
5.8 Bild mit erkannter Wunde 41
6.1 Schaufensterpuppe mit 8 Wunden 43
6.2 Alle acht Wunden aus dem Versuch 44
7.1 Beispielbilder der acht Wunden mit segmentiertem Bereich 49
8.1 Wunde 5 51
8.2 Wunde 6 52

Abkürzungsverzeichnis

ANN Artificial Neural Network

AR Augmented Reality

AV Augmented Virtuality

CNN Convolutional Neural Network

HMD Head mounted display

MR Mixed Reality

MRTK Mixed Reality Toolkit

RPN Regional Proposal Network

R-CNN Region-based Convolutional Neural Network

RoI Region of Interest

VR Virtual Reality

Tabellenverzeichnis

6.1 Größe der einzelnen Wunden im Versuch in mm 42
7.1 SUS-Fragebogen Ergebnis 46
7.2 Messergebnisse: Breite 47
7.3 Messergebnisse: Länge 48

Listingverzeichnis

1 C#-Skript einer Beispiel-Komponente in Unity 9
2 AppState: Liste aller Fortschrittsstadien der Anwendung 28
3 Initialisierung des GrammarRecognizer 32
4 WoundConfig für Training 34
5 Datenerweiterungsoperationen von imgaug 34
6 Trainings-Funktion 35
7 Berechnung der Distanz von Kamera zur Wunde 37
8 Berechnung der Wundgrößen 40

1 Einleitung

1.1 Motivation

Die Behandlung von Wunden ist ein kritischer Aspekt in der medizinischen Versorgung und kann einen signifikanten Einfluss auf die Genesung von Patientinnen und Patienten haben. Die genaue Vermessung von Wunden, insbesondere von komplexen oder schwer zugänglichen Verletzungen, ist häufig eine zeitaufwändige und mühsame Aufgabe für medizinisches Fachpersonal. Herkömmliche manuelle Methoden der Wundvermessung können ungenau sein und erfordern ein hohes Maß an Fachwissen, was zu suboptimalen Behandlungsentscheidungen und längeren Heilungszeiten führen kann.

Hier setzt die Motivation für diese Masterarbeit an: Durch den Einsatz modernster Technologien wie der HoloLens 2, einer Augmented-Reality-Brille, in Kombination mit einem Convolutional Neural Network (kurz: CNN) als leistungsstarkes Objekt- und Klassifizierungswerkzeug, ist es möglich, ein praxistaugliches Wundmessungsverfahren zu entwickeln. Ein solches Verfahren könnte medizinischem Fachpersonal dabei helfen, Wunden schneller und präziser zu vermessen, Fortschritte in der Wundheilung effizienter zu überwachen und maßgeschneiderte Therapiepläne zu erstellen. Die Umsetzung dieser Arbeit wurde von der Tiplu GmbH in Auftrag gesetzt. Die Tiplu GmbH ist ein Softwareunternehmen aus Hamburg, das KI-gestützte Lösungen in den Bereichen Medizincontrolling, medizinische Entscheidungsunterstützung und Prozessautomation anbietet (Tiplu, 2023).

1.2 Zielsetzung der Arbeit

Das Hauptziel dieser Masterarbeit ist die Entwicklung eines praxistauglichen Wundmessungsverfahrens auf der HoloLens 2 unter Verwendung eines CNN. Konkret soll ein Prototyp entwickelt werden, der zum einen in der Lage ist, Patientendaten abzurufen und Vitaldaten mithilfe von Spracherkennung aufzunehmen und zum anderen

soll die Anwendung Wunden fotografieren, analysieren und automatisch präzise Messungen durchführen können. Es soll anhand einer empirischen Untersuchung geprüft werden, inwieweit eine solche Anwendung sinnvoll in der Praxis eingesetzt werden und den Aufwand für medizinisches Personal im Klinikalltag reduzieren kann.

1.3 Aufbau der Arbeit

Im folgenden zweiten Kapitel werden die technologischen Grundlagen erörtert, die für die Implementierung der Anwendung für die HoloLens 2 erforderlich sind. In Kapitel 3 wird der aktuelle Stand der Technik in Bezug auf Wundmessung mittels AR-Geräten und/oder CNN-Modellen näher erläutert. Im vierten Kapitel wird das Konzept der Anwendung anhand der funktionalen und nicht-funktionalen Anforderungen, die durch die Tiplu GmbH gegeben sind, vorgestellt. Kapitel 5 konzentriert sich auf die Implementierung der Anwendung, wobei der Prototyp und seine Funktionsweise detailliert präsentiert wird. Die Evaluation der Anwendung mittels einer empirischen Untersuchung wird im siebten Kapitel erklärt. Im sechsten Kapitel *Ergebnisse* werden die Ergebnisse der empirischen Untersuchung präsentiert. Dabei wird zum einen die berechnete Nutzbarkeit mittels System Usability Scale und zum anderen die Messergebnisse der Wundmessung dargestellt. In Kapitel 8 werden diese Ergebnisse interpretiert und eingeordnet. Das letzte Kapitel rundet die Arbeit schlussendlich mit einem Fazit ab.

2 Grundlagen

Dieses Kapitel bildet das theoretische Fundament für die vorliegende Arbeit. Es widmet sich den grundlegenden Konzepten und Technologien, die für das Verständnis und die Umsetzung des Projekts von zentraler Bedeutung sind, beginnend mit Augmented Reality, gefolgt von HoloLens 2, Unity, künstlichen neuralen Netzwerken und der klassischen Wundmessung.

2.1 Augmented Reality

Augmented Reality (dt. erweiterte Realität, kurz: AR) wird als eine Technologie bezeichnet, die es möglich macht, in Echtzeit die direkte oder indirekte Ansicht der realen, physischen Welt mit virtuellen, computergenerierten Informationen zu erweitern (vgl. Furht, 2011, S. 11). Um eine klare Abgrenzung zwischen AR und Begriffen wie *Virtual Reality* (kurz: VR) zu gewährleisten und um ein besseres Verständnis von AR zu ermöglichen, kann das von Paul Milgram und Fumio Kishino entwickelte Milgram's *Reality-Virtuality-Kontinuum* (kurz: RV-Kontinuum) herangezogen werden (Milgram et al., 1994, S. 283).

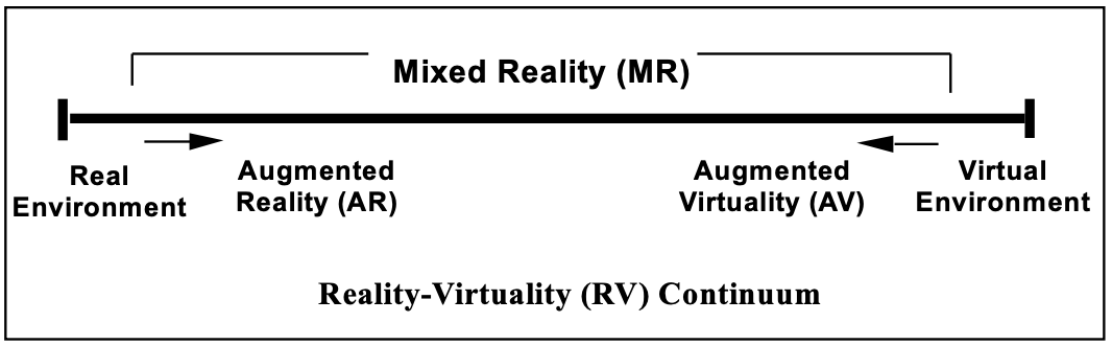


Abbildung 2.1: Vereinfachte Darstellung des RV-Kontinuums
(Quelle: Milgram et al., 1994, S. 283)

Wie in Abb. 2.1 zu sehen, beschreibt das RV-Kontinuum die Bandbreite der Realitätsstufen von der vollständig physischen Realität (*Real Environment*) bis zur vollständig virtuellen Realität (*Virtual Environment*). *Real Environment* repräsentiert die echte, physische Welt. Hier gibt es verständlicherweise keinerlei digitale Einblendungen

oder virtuelle Elemente. AR liegt auf dem Kontinuum nahe an der physischen Realität. Hier sind die virtuellen Inhalte nur als Erweiterung der realen Welt zu betrachten. *Augmented Virtuality* (dt.: *erweiterte Virtualität*, kurz: AV) ist eher eine überwiegend virtuelle Umgebung, in der echte Elemente oder Personen eingefügt werden. Ein Beispiel hierfür könnte ein Videospiel sein, in dem die Gesichter von echten Personen in einer ansonsten komplett computergenerierten Welt dargestellt werden. Die *Virtual Environment* (auch VR genannt) liegt auf dem Kontinuum entgegengesetzt zu der physischen Realität. Hier werden Nutzende vollständig in eine virtuelle Umgebung versetzt und sind von der realen Welt abgeschirmt. *Mixed Reality* (kurz: MR) wird als eine Umgebung definiert, in der Objekte der realen Welt und der virtuellen Welt zusammen auf einem einzigen Bildschirm dargestellt werden. Dies umfasst den gesamten Bereich zwischen den Extremen des RV-Kontinuums (Milgram et al., 1994, S. 283). AR- und AV-Systeme können daher auch als MR-Systeme bezeichnet werden.

Um AR-Inhalte in die reale Welt zu integrieren, benötigt man ein Verfahren zur Objekterkennung und -verfolgung sowie eine Methode zur Anzeige der digitalen Informationen. Die meisten AR-Anwendungen verwenden dazu die Kamera des Geräts, um die Umgebung zu erfassen und die digitalen Inhalte auf dem Bildschirm zu überlagern. Die Kamera erkennt dabei bestimmte Merkmale in der Umgebung, wie zum Beispiel Bildmarker, und platziert die Inhalte an entsprechender Stelle. Zu den gängigsten, AR-fähigen Geräten gehören zum einen Monitor-basierte Geräte, wie Smartphones und Tablets, die mittels der integrierten Kamera die Umgebung erfassen und die virtuellen Zusatzinhalte auf dem Bildschirm sichtbar machen. Mittels Interaktion mit dem Touch-Display können die User die Anwendungen bedienen. Zum anderen können auch „See-through"-AR-Geräte wie *Head-Mounted Displays* (kurz: HMD) für die Darstellung und Interaktion mit virtuellen Inhalten eingesetzt werden. Dabei werden die Inhalte dem Nutzenden direkt im Sichtfeld ausgegeben und können gegebenenfalls mittels Gesten- oder Sprachsteuerung bedient werden. Im folgenden Unterkapitel wird das für diese Arbeit ausgewählte AR-Gerät *HoloLens 2* vorgestellt.

2.2 HoloLens 2

Die HoloLens 2 (siehe Abbildung 2.2) ist eine AR-Brille, die von Microsoft entwickelt und im Jahr 2019 veröffentlicht wurde. Sie stellt den Nachfolger der HoloLens 1 dar. Bei der HoloLens 2 handelt es sich um ein HMD, welches nativ ohne zusätzlichen, externen Computer verwendet werden kann und somit *standalone* (dt.: eigenständig) ist. Die Hardware besitzt einen Qualcomm Snapdragon 850 Chip und nutzt das Windows 10 Holographic Betriebssystem (Cooley, 2023).

Abbildung 2.2: HoloLens 2
(Quelle: Bohn, 2019)

Pro Auge verfügt das Gerät über holografische Displays, welche jeweils die Inhalte in einem Sichtfeld von 43 Grad horizontal und 29 Grad vertikal ausgeben (diagonal 52 Grad). Mit einer Auflösung von 2K und einem Seitenverhältnis von 3:2 kann die HoloLens 2 47 Pixel pro Grad darstellen (Cooley, 2023). Da die Displays transparent sind, können die Nutzenden weiterhin die reale Welt sehen, während zusätzliche Inhalte stereoskopisch, also für den Betrachter dreidimensional auf den Displays projiziert werden. Im Gegensatz zur Darstellung von AR-Inhalten auf Monitoren von Smartphones oder Tablets müssen die Nutzenden ihre Aufmerksamkeit nicht zwischen einer Darstellung auf einem Display und der Realität aufteilen und während der gesamten Nutzung sind die Hände frei.

Damit die HoloLens 2 digitale Inhalte korrekt in der realen Umgebung platzieren kann, nutzt sie das sogenannte *Spatial Mapping*, welches eine virtuelle Nachbildung eines realen Raumes erstellt und diese Nachbildung als Dreiecks-Gitternetz darstellt (siehe Abb. 2.3). Dies wird mithilfe von vier Kameras und einer Tiefenkamera bewerkstelligt, die permanent die Umgebung abscannen und die sogenannte Spatial Map aktualisiert.

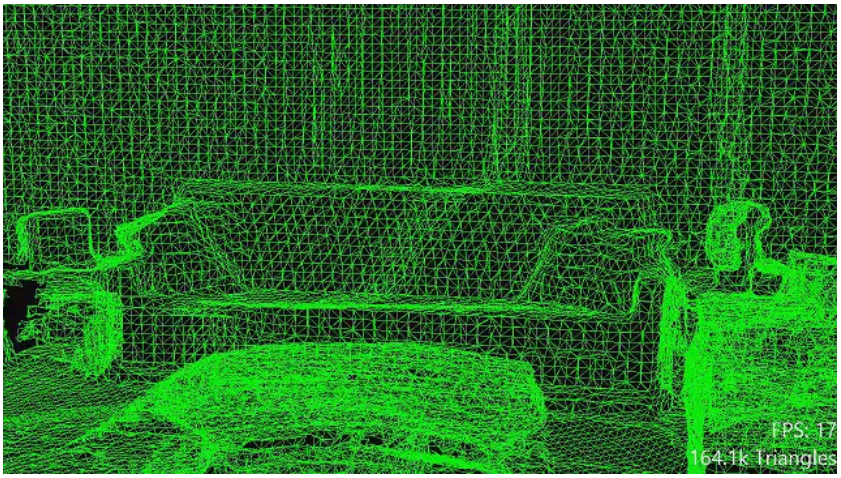


Abbildung 2.3: Spatial Mapping der HoloLens 2
(Quelle: Mattzmsft, 2023)

Zusätzlich verfügt die HoloLens über eine *Inertial Measurement Unit* (dt.: inertiale Messeinheit; kurz: IMU), die mittels Beschleunigungssensoren die Kopfbewegung und damit die Blickrichtung des Nutzenden erfassen kann (Cooley, 2023). Vier Mikrofone ermöglichen eine Sprachsteuerung und zwei Lautsprecher einen räumlichen Klang. Dadurch lässt sich die HoloLens 2 neben der angebotenen Gestensteuerung auch ohne den Einsatz der Hände, nur über Sprache und Kopfbewegungen, steuern. Zusätzlich gibt es die Eingabemöglichkeit über *Eye Tracking*. Durch Infrarotkameras, die direkt auf die Augen des Nutzenden gerichtet sind und deren Blickrichtung erfasst, können Nutzende ohne Gesten und Sprache mit Objekten interagieren.

Für die Entwicklung von Anwendungen für die HoloLens 2 können unterschiedliche Tools verwendet werden. Zum einen machen es Game Engines wie *Unity 3D* und *Unreal* möglich, solche Anwendungen mit relativ wenig Programmieraufwand in einem grafischen Editor zu entwickeln. Zum anderen lassen sich Anwendungen auch mittels

der OpenXR API erstellen, welches jedoch mit deutlich größerem Aufwand verbunden ist. Unity 3D kommt für die Entwicklung der Anwendung dieser Arbeit zum Einsatz, da bereits vorhandene Vorkenntnisse die notwendige Einarbeitungszeit minimieren. Im folgenden Unterkapitel werden die für diese Arbeit bedeutsamen Konzepte dieser Game Engine erläutert.

2.3 Unity 3D

Unity 3D ist eine weit verbreitete plattformübergreifende Game Engine, die von *Unity Technologies* entwickelt wurde. Mit Unity können Entwickelnde interaktive 2D- und 3D-Spiele sowie andere immersive Erfahrungen für verschiedene Plattformen erstellen, darunter Mobilgeräte, PCs, Konsolen und auch AR-Geräte wie die HoloLens 2. Die Nutzung dieser Game Engine ist während der Entwicklungsphase kostenlos. Zahlungspflichtig wird diese erst, wenn die erstellten Anwendungen kommerziell vertrieben werden (Unity-Technologies, 2023a). Der Inhalt einer von Unity erstellten Anwendung wird in sogenannten *Szenen* dargestellt. Diese enthalten die in der Anwendung befindlichen Objekte (z. B. 3D-Modelle, UI-Elemente) und können als abgeschlossenes Level eines Spiels betrachtet werden (Unity-Technologies, 2023f). Die Positionierung der Spielobjekte erfolgt durch ein linkshändiges globales kartesisches Koordinatensystem sowie durch beliebig viele lokale Koordinatensysteme. Ein lokales Koordinatensystem ist dabei an ein GameObject gebunden und beschreibt durch seine Koordinaten die Position und Orientierung relativ zum Objekt (Unity-Technologies, 2023g). Die Abbildung 2.4 zeigt ein Spielobjekt in Form eines roten Würfels und dessen lokales Koordinatensystem innerhalb des globalen Koordinatensystems.

In Unity werden Spielobjekte auch *GameObjects* genannt. GameObjects dienen als Container-Objekte für Komponenten (engl.: *Components*) und beliebig weitere Kindobjekte. Diese werden hierarchisch in der Szenenansicht im Editor organisiert. Komponenten dienen wiederum zur Implementierung des Verhaltens und der Eigenschaften eines GameObjects. Auf sie werden die in der Anwendung verfügbaren Funktionalitäten abgebildet (Unity-Technologies, 2023d). GameObjects samt ihren Kindobjekten und Komponenten lassen sich als sogenannte *Prefabs* speichern. Prefabs sind wieder-

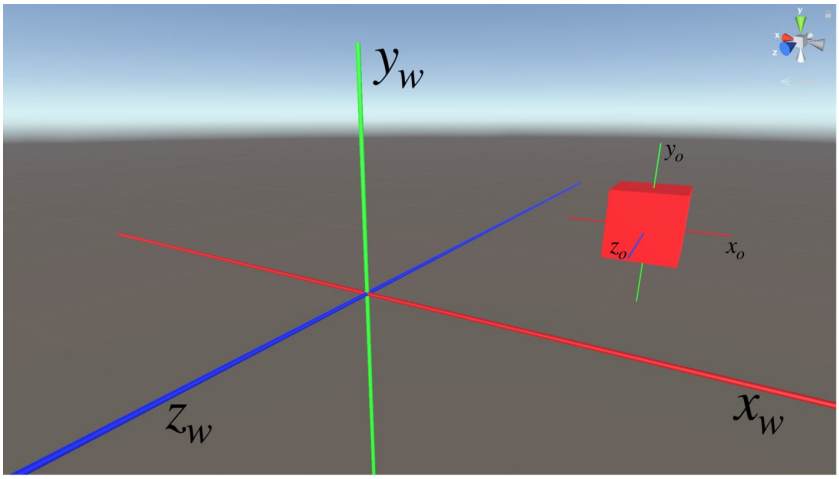


Abbildung 2.4: Globales Koordinatensystem einer Szene, mit einem darin befindlichen Spielobjekt und dessen lokalen Koordinatensystems. *(Quelle: Çamönü, 2019)*

verwendbare Vorlagen von GameObjects, die beliebig oft in der Szene instanziiert und verändert werden können (Unity-Technologies, 2023e).

Komponenten werden in Form von Skripten an das entsprechende Spielobjekt angefügt. Nativ erfolgt die Programmierung der Skripte durch die Erstellung von Klassen in der Programmiersprache C#. Alternativ kann u. a. auch die Sprache *JavaScript* verwendet werden. Die Einbindung einer Klasse in die Struktur der Engine erfolgt durch die Vererbung der Elternklasse *MonoBehaviour*. Erbt ein Skript von dieser Elternklasse, stellt die Engine eine Reihe von Methoden zur Verfügung, welche sie an definierten Zeitpunkten zur Laufzeit ausfuhrt. Listing 1 zeigt beispielhaft den grundlegenden Aufbau eines in C# verfassten Skriptes. Die Methode *Start* (siehe Listing 1, Zeile 3) wird vor dem Spielstart aufgerufen und dient zur Initialisierung einer Komponente bzw. der dazugehörigen Attribute. Die *Update*-Methode (siehe Listing 1, Zeile 7) wird periodisch vor jedem Frame aufgerufen und kann beispielsweise Funktionalitäten wie das Bewegen von GameObjects oder die Verarbeitung von Nutzereingaben enthalten (Unity-Technologies, 2023c).

```
...
public class SampleComponent : MonoBehaviour {
   private void Start () {

   }

   private void Update () {

   }
}
```

Listing 1: C#-Skript einer Beispiel-Komponente in Unity

Der Inhalt einer Szene wird mithilfe mindestens einer Kamera dargestellt, welche als GameObject in der Szene integriert ist und mittels Komponenten manipuliert werden kann, z. B. um eine Kamerafahrt umzusetzen. Die Kamera definiert den Beobachtungsstandpunkt sowie den zweidimensionalen Bildausschnitt der dreidimensionalen Szene. Der Bildausschnitt kann perspektivisch mit oder orthografisch ohne Tiefeneffekt dargestellt werden. Die orthografische Darstellung der Szene eignet sich beispielsweise zur Darstellung von 2D-Spielen (Unity-Technologies, 2023b).

Für bestimmte Anforderungen oder Plattformen existieren eine Vielzahl von kostenlosen Erweiterungen, die im Editor eingebettet werden können. Für die Entwicklung von z. B. AR-Anwendungen für die HoloLens 2 bietet Microsoft das *Mixed Reality Toolkit* (kurz: MRTK) an. Das MRTK für Unity stellt eine Reihe von Werkzeugen, Komponenten und vorgefertigten Funktionen bereit, die die Entwicklung von plattformübergreifenden MR-Anwendungen erleichtern. Zu den unterstützten Plattformen des MRTK gehören u. a. Meta Quest, Android, iOS, HoloLens 1 und HoloLens 2 (Semple, 2023). Eine der wichtigsten Komponenten des Toolkits ist das Input-System. Entwickelnde können damit auf einfache Weise auf Benutzereingaben zugreifen und diese verarbeiten. Ein weiterer entscheidender Bestandteil des MRTK ist das Spatial-Mapping-System, das die Umgebungserkennung und -darstellung in der virtuellen Welt ermöglicht. Dieses System erfasst die physischen Strukturen der Umgebung und erstellt ein digitales Modell, das in die AR-Anwendung integriert werden kann. Dadurch können virtuelle Objekte realistisch mit der realen Welt interagieren und sich entsprechend der Umgebung verhalten. Wie in der Abbildung 2.5 zu sehen, bietet das MRTK auch

vorgefertigte Prefabs für die Benutzeroberfläche an, die alle benötigten Funktionen beinhalten, die die Interaktion mit diesen Objekten, z. B. durch direkter Berührung, ermöglichen.

Abbildung 2.5: Button-Prefab des MRTK
(Quelle: Microsoft, 2022b)

Diese Prefabs lassen sich beliebig anpassen und ermöglichen ein schnelles Prototyping. Die Entwicklung von Anwendungen mit diesem Tool lässt sich auch durch den Einsatz einer Simulation in der Echtzeit-Vorschau des Editors beschleunigen. Die Bewegung im Raum und Interaktionen mit den Händen wird hierbei mit Maus und Tastatur simuliert. Damit können Entwickelnde ihre Änderungen an der Anwendung ohne Build-Prozess auf das Endgerät debuggen und testen.

Das MRTK bieten außerdem auch eine Spracherkennungsfunktion, die es ermöglicht, Sprachbefehle in die Anwendung zu integrieren. Die Spracherkennung des MRTK basiert auf maschinellen Lernalgorithmen, die in der Lage sind, menschliche Sprache präzise zu verstehen und zu interpretieren. Das Toolkit unterstützt dabei mehrere Sprachen. Entwickelnde können eigene Sprachbefehle definieren und diese mit bestimmten Aktionen oder Funktionen verknüpfen. Wenn der Nutzende einen Sprachbefehl ausspricht, erfasst das MRTK die Audioeingabe, wandelt sie in Text um und vergleicht den erkannten Text mit den definierten Sprachbefehlen. Bei Übereinstimmung wird die entsprechende Aktion oder Funktion in der Anwendung ausgelöst (Microsoft, 2022c). Nutzende können so beispielsweise Objekte durch Sprachbefehle auswählen oder mit virtuellen Charakteren über Sprache kommunizieren.

2.4 Künstliche neuronale Netzwerke

Künstliche neuronale Netzwerke (engl.: *Artificial Neural Networks*, kurz: ANN) sind von der Funktionsweise her von biologischen Nervensystemen wie dem menschlichen Gehirn inspiriert. Sie bestehen aus einer Vielzahl von miteinander verbundenen Rechenknoten (auch Neuronen genannt), die in verteilter Weise zusammenarbeiten, um aus den *Inputs* (dt.: Eingaben) zu lernen und so den endgültigen *Output* (dt.: Ausgabe) zu optimieren. Die Abbildung 2.6 zeigt die Grundstruktur dieser Netzwerke auf. ANNs bestehen aus einem mehrdimensionalen Vektor, dem sogenannten *Input Layer* (dt.: Eingabeschicht). Jeder der Neuronen in dieser Schicht repräsentiert einen Wert, der dem Netzwerk zugeführt wird. Diese Werte werden an einen oder mehreren *Hidden Layer* (dt.: versteckte Schichten) übertragen. Jede Verbindung zwischen den Eingabewerten und einer Hidden Layer hat ein *weight* (dt.: Gewicht), das anzeigt, wie wichtig der Input für die Hidden Layer ist. Das Lernen in einem ANN geschieht durch einen Prozess namens *Backpropagation*. Dabei werden die Unterschiede zwischen den tatsächlichen Ausgaben des Netzwerks und den erwarteten Ausgaben verwendet, um die Gewichte der Verbindungen zwischen den Neuronen anzupassen. Dieser Prozess wird als Lernprozess bezeichnet. Sind mehrere Hidden Layer übereinander angeordnet, wird auch von *Deep Learning* (dt.: tiefes Lernen) gesprochen (O'Shea und Nash, 2015, S. 1).

Damit ein ANN mit den gegebenen Eingaben ein gewünschtes Ergebnis liefern kann, muss es nach der Initialisierung auf die gewünschte Art trainiert werden. Hierfür wird ein Trainings-Datensatz verwendet. Beim Training wird der Fehler zwischen dem Output des ANN und dem korrekten Ergebnis ermittelt und minimiert. Je umfangreicher der Trainings-Datensatz ist, desto korrektere Ergebnisse kann das Netzwerk liefern. Es gibt hierbei zwei Arten von Trainingsprozessen, dem *Supervised* und *Unsupervised Learning*. Beim Supervised Learning ist für jedes Element im Trainings-Datensatz die richtige Klassifizierung mittels eines Labels hinterlegt. Hierbei spricht man von einem gelabelten Datensatz. Der Input wird durch das ANN gesendet, der Output mit dem bereits bekannten Label des Inputs verglichen und der Fehler berechnet. Beim Unsupervised Learning ist der Trainings-Datensatz nicht gelabelt. Das bedeutet, das neuro-

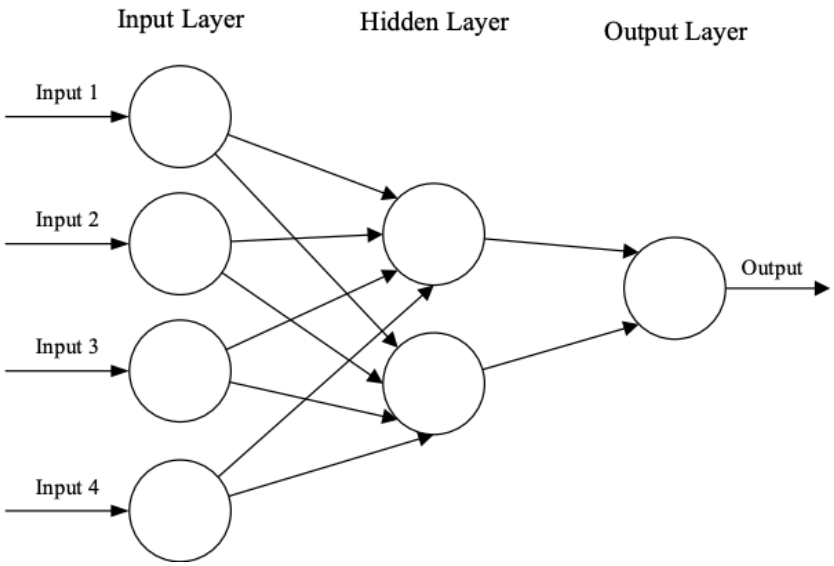


Abbildung 2.6: Grundstruktur von ANNs
(Quelle: O'Shea und Nash, 2015, S. 2)

nale Netz muss selbst eine gewisse Struktur durch das Analysieren gewisser Features in den Daten finden (O'Shea und Nash, 2015, S. 2).

Da umfangreiche Trainings-Datensätze besonders für sehr spezielle Probleme nicht immer verfügbar sein können, können Entwickler auf die sogenannte *Data Augmentation* (dt.: Datenerweiterung) zurückgreifen. Dies wird eingesetzt, um einen kleinen Trainings-Datensatz mit möglichst wenig Aufwand zu vergrößern. Dabei werden lediglich vorhandene Daten leicht verändert, um neue Daten zu erhalten. Ein Bild kann beispielsweise gespiegelt, gedreht oder verzerrt werden (siehe Abb. 2.7). Wird dies bei jeder Datei des Datensatzes erledigt, so kann dessen Größe vervielfacht werden, ohne neue Daten zu benötigen (Wong et al., 2016). ANN-Frameworks stellen diese Funktionalität meist zur Verfügung.

Zur Klassifizierung von Bildern wird eine spezielle Art von ANNs eingesetzt, dem Convolutional Neural Network. CNNs werden speziell für die Verarbeitung von Bildern und anderen 2D-Daten verwendet. Im folgenden Unterkapitel wird diese Art von Netzwerken und ihre Weiterentwicklung näher beleuchtet.

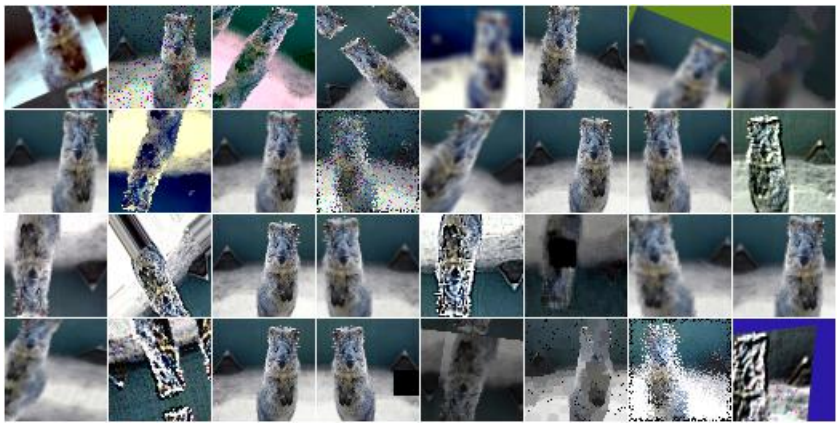


Abbildung 2.7: Vermehrte Bilder aus einer Datei
(Quelle: Jung, 2020)

2.4.1 Convolutional Neural Networks

Im Gegensatz zu traditionellen künstlichen neuronalen Netzen sind CNNs in der Lage, räumliche Informationen aus Bildern zu extrahieren und so komplexe Muster und Merkmale, wie zum Beispiel Kanten, Formen oder Texturen, zu erkennen. Diese Netzwerke unterscheiden sich im Wesentlichen von herkömmlichen ANNs durch ihre speziellen Layer. Wie in Abb. 2.8 zu sehen besteht ein CNN aus einem Input Layer, gefolgt von mehreren *Convolutional Layer* und *Pooling Layer*. In weiterer Folge besteht ein CNN aus *Fully Connected Layer* und einem Output Layer.

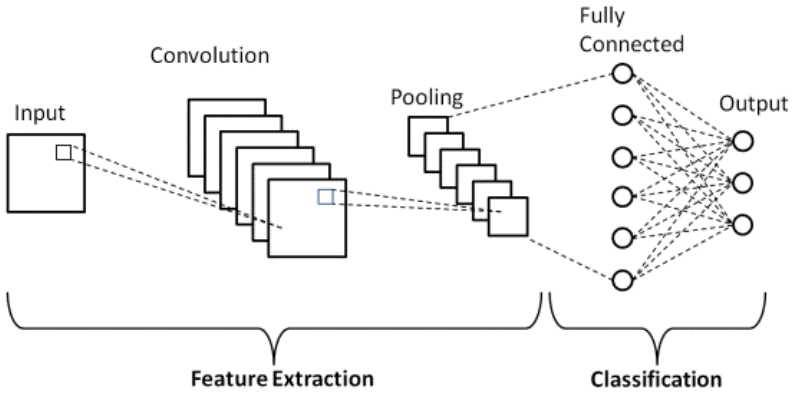


Abbildung 2.8: Aufbau eines CNN
(Quelle: Gurucharan, 2022)

Ein Convolutional Layer (siehe Abb. 2.9) besteht aus einer Vielzahl von Filtern, die auch als *Kernels* bezeichnet werden. Der Convolutional Layer extrahiert die unterschiedlichen Merkmale der Bilder, die an das Netzwerk gegeben werden (auch *Feature Extraction* genannt). Die Filter werden in der Regel als zweidimensionale Matrizen, bestehend aus Gewichtungen, repräsentiert. Wenn die 2D-Daten eines Bildes an diese Schicht übertragen werden, faltet die Schicht jeden Filter über die räumliche Dimensionalität der Eingabedaten. Dies erzeugt in seiner Ausgabe eine sogenannte *Activation Map* (auch *Feature Map* genannt), die zunächst alle grundlegenden Muster und Merkmale eines Bildes enthält (O'Shea und Nash, 2015, S. 8).

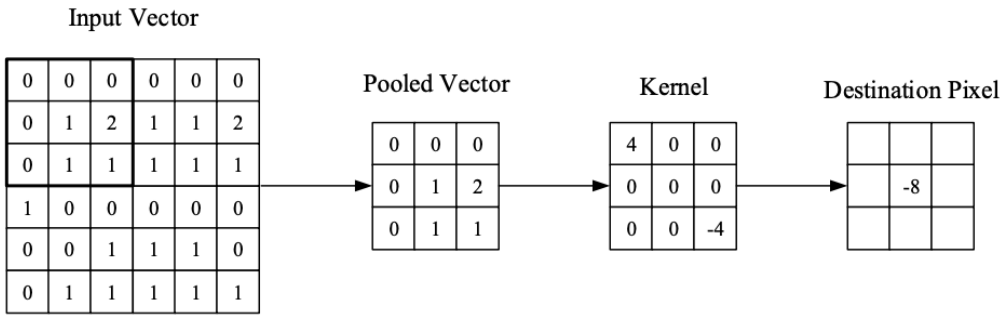


Abbildung 2.9: Faltungsoperation im Convolutional Layer
(Quelle: O'Shea und Nash, 2015, S. 6)

In den meisten Fällen folgt dem Convolutional Layer ein Pooling Layer. Ziel dieser Schicht ist es, die Größe der Activation Maps zu reduzieren, um Rechenaufwand zu sparen (O'Shea und Nash, 2015, S. 5). Der darauf folgende *Fully Connected* Layer ist eine Art von Schicht, der sich nicht aus Filtern, sondern aus Neuronen zusammensetzt, welche jeweils einen einzelnen Wert speichern. Er befindet sich am Ende eines CNN. Bei der Transformation von einer Activation Map zu einem Fully Connected Layer wird jeder Wert der Activation Map mit einem Neuron im folgenden Fully Connected Layer verknüpft. Diese Neuronen sind durch Gewichte mit Neuronen aus der nachfolgenden Schicht verbunden. Die Menge der Neuronen in der Ausgabeschicht entspricht der Anzahl der potenziellen Bildklassen (O'Shea und Nash, 2015, S. 8).

Dabei gibt es eine Reihe von Weiterentwicklungen des CNN, die auf das Prinzip aufbauen und Verbesserungen liefern. Dazu gehören zum einen das *Region-based Convo-*

lutional Neural Network (kurz: R-CNN) und das *Faster R-CNN*. Das R-CNN wurde erstmals von Girshick et al., 2013 vorgestellt und basiert auf der Idee, sogenannte *Regions of Interest* (dt.: Regionen von Interesse, kurz: RoIs) in einem Bild zu extrahieren und diese einzeln zu klassifizieren. Das ursprüngliche R-CNN-Modell hatte jedoch einige Einschränkungen, wie beispielsweise eine langsame Ausführungsgeschwindigkeit aufgrund der Verarbeitung von RoIs in separaten Vorwärtsdurchläufen des CNN. Um die Geschwindigkeit des R-CNN-Modells zu verbessern, wurde das Faster R-CNN eingeführt. Das Faster R-CNN integriert ein *Region Proposal Network* (kurz: RPN) in das CNN, um effizient RoIs zu generieren. Ein RPN ist verantwortlich für die Generierung von Vorschlägen potenzieller Regionen, die interessante Objekte im Bild enthalten könnten (Ren et al., 2015). Dieser Schritt basiert auf Ideen aus dem R-CNN-Ansatz.

Diese Netzwerke liefern zwar mit ihren RoIs etwa die Position eines Objektes auf einem Bild, für Pixel-genaue Ergebnisse reichen diese leider nicht aus. Für eine genaue Segmentierung von Objekten auf Bildern wurde das *Mask Region-based Convolutional Neural Network* (kurz: Mask R-CNN) entwickelt, welches im folgenden Unterkapitel näher beleuchtet wird.

2.4.2 Mask R-CNN

Mask R-CNN ist ein weiterer Fortschritt im Bereich der Objekterkennung. Vorgestellt wurde das Modell in einer Arbeit von He et al., 2017. Es baut auf den Vorarbeiten des R-CNN und seiner Weiterentwicklungen Faster R-CNN auf und erweitert diese um die Fähigkeit der Instanzsegmentierung. Instanzsegmentierung ist eine Kombination aus zwei Unterproblemen, der Objekterkennung und der *Semantic Segmentation*. Die Objekterkennung, also das Finden und Klassifizieren von Objekten auf Bildern, wurde bereits durch einfache CNN ermöglicht. Das zweite Unterproblem ist die *Semantic Segmentation* (dt.: semantische Segmentierung), welches dem Netzwerk ein genaueres Verständnis eines Bildes auf Pixelebene ermöglicht. Jedes einzelne Pixel im Bild erhält hierbei vom Netzwerk eine Objekt-Klasse.

Das Mask R-CNN-Modell besteht aus zwei Hauptteilen: einem Backbone-Netzwerk und einem Head-Netzwerk. Das Backbone-Netzwerk ist in der Regel ein tiefes CNN

zur Extraktion von Merkmalen. In der Originalarbeit von He et al., 2017 wurde *ResNet-101* in Kombination mit einem *Feature Pyramid Network* (kurz: FPN) als Backbone verwendet. Das Head-Netzwerk enthält die zur Durchführung der eigentlichen Aufgaben erforderlichen Teile. Dies umfasst ein RPN zur Generierung von RoIs, einen Klassifikator zur Kategorisierung der vorgeschlagenen Regionen, einen *Bounding-Box-Regressor* zur Anpassung der Bounding-Boxen und letztlich einen *Mask Predictor* zur Generierung einer binären Maske für jedes Objekt.

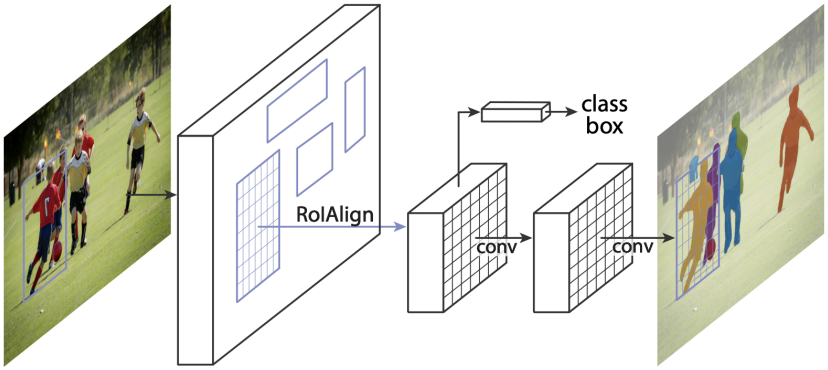


Abbildung 2.10: Mask R-CNN
(Quelle: He et al., 2017, S. 1)

Um diese pixelgenaue Segmentierung zu erreichen, führt Mask R-CNN eine Methode namens *RoIAlign* ein. Im Gegensatz zur *RoIPool*-Methode, die in Fast R-CNN und Faster R-CNN verwendet wird und Quantisierungsfehler verursacht, ermöglicht *RoIAlign* eine exakte Zuordnung zwischen den Eingabe-Features und der Ausgabe-Maske, indem es die Quantisierung des RoIs komplett vermeidet. Diese Verbesserung trägt wesentlich zur Erhöhung der Segmentierungsgenauigkeit von Mask R-CNN bei (He et al., 2017). Wie in Abb. 2.11 zu sehen, ist ein korrekt trainiertes Mask R-CNN in der Lage, wenn es z. B. auf Bilder mit Ballons trainiert wurde, die Fläche dieser Objekte nahezu pixelgenau auf dem Bild zu segmentieren.

Der Trainingsprozess des Mask R-CNN besteht aus zwei Hauptphasen: dem vortrainierten Backbone-Netzwerk und dem Finetuning der RPN und der Mask Prediction-Komponenten. Während des Trainings wird das Netzwerk mit annotierten Bilddaten trainiert, um die Klassifikation, Bounding-Box-Regression und Maskenvorhersage zu optimieren. Da dieser Prozess je nach Aufgabe viel Zeit und Rechenleistung benö-

Abbildung 2.11: Von einem Mask R-CNN segmentierte Objekte
(Quelle: Matterport, 2017)

tigt, kann *Transfer-Learning* angewendet werden. Transfer-Learning bezieht sich auf eine Technik des maschinellen Lernens, bei der ein Modell, das auf einer bestimmten Aufgabe trainiert wurde, auf eine andere, verwandte Aufgabe übertragen wird. Beim Transfer-Learning wird das bereits trainierte Modell als Ausgangspunkt verwendet, um die Trainingszeit und -ressourcen zu reduzieren und die Leistung auf der neuen Aufgabe zu verbessern. Transfer-Learning kann auf Mask R-CNN angewendet werden, um es auf spezifische Domänen oder Aufgaben anzupassen, ohne von Grund auf neu trainieren zu müssen. Um Transfer-Learning mit Mask R-CNN durchzuführen, sind normalerweise einige Schritte erforderlich. Zunächst wird ein vortrainiertes Modell geladen und die letzten Schichten, die für die Klassifizierung und Segmentierung verantwortlich sind, abgetrennt. Anschließend werden diese Schichten durch neue Schichten ersetzt, die der spezifischen Aufgabe angepasst sind. Schließlich wird das Modell auf den neuen Datensatz trainiert, um die weights anzupassen.

Mask R-CNN hat hervorragende Leistungen in verschiedenen Benchmarks für Objekterkennung und -segmentierung gezeigt, wie zum Beispiel dem COCO (Common Objects in Context)-Datensatz (He et al., 2017). Darüber hinaus hat sich Mask R-CNN als nützlich in einer Reihe von realen Anwendungen erwiesen, von der Erkennung von Personen und Objekten in Fotos und Videos bis hin zur medizinischen Bildanalyse, wo es z. B. zur Erkennung und Segmentierung von Wunden eingesetzt werden kann.

2.5 Klassische Wundmessungsverfahren

Die sorgfältige Dokumentation von Wunden ist ein entscheidender Aspekt in der täglichen medizinischen Praxis für Pflegepersonal und Ärzte. Dabei ist eine präzise Wundgrößenbestimmung und das Sammeln von Fotografien von entscheidender Bedeutung, um den Fortschritt von Schnittwunden oder anderen Verletzungen zu dokumentieren, Veränderungen in der Wundgröße und dem Wundzustand zu verfolgen und die Wirksamkeit der angewandten Behandlungen zu beurteilen. Es ist außerdem hilfreich, wenn die Pflegekraft z. B. eine Wundauflage in geeigneter Größe finden soll oder Prüfungen und Rückfragen durch Krankenkassen eine aktuelle Wunddokumentation erfordern. Die visuelle Wundvermessung ist eine der einfachsten und häufigsten Methoden, die von Pflegepersonal und Ärzten durchgeführt werden können. Hierbei erfolgt die manuelle Bestimmung der Wundgröße, Vermessung der Länge und Breite der Wunde mit einem Lineal oder einer Skala. Obwohl diese Methode einfach und kostengünstig ist, ist sie subjektiv und abhängig von der Herangehensweise der Pflegekraft bzw. des Arztes und somit fehleranfällig. Ergänzt mit einer regelmäßigen Wundfotografie ist es eine effektive Methode, um den Zustand der Wunde zu dokumentieren und Veränderungen im Laufe der Zeit zu verfolgen. Mit mobilen Geräten können Pflegekräfte und Ärzte Fotos von Wunden oder Verletzungen machen, um eine visuelle Referenz zu haben. Dabei sollten die Fotos möglichst immer unter gleichen Bedingungen erstellt werden (vgl. Protz, 2023).

Für die Messung der Länge und Breite einer Wunde gibt es zwei Herangehensweisen: die *Körperachsen*-Methode und die *Perpendikular*-Methode Hartmann, 2023. Bei der Körperachsen-Methode wird die Ausrichtung des Körpers zur Orientierung verwendet. Dabei wird die größte Länge der Wunde auf der Längsachse des Patienten

bestimmt und anschließend die größte Breite senkrecht zu dieser Achse (siehe Abb. 2.12). Bei der Perpendikular-Methode (Senkrecht-Methode) ist die Messachse unabhängig von der Körperachse. Hier wird zunächst die größte Länge einer Wunde bestimmt. Anschließend wird senkrecht zu der Achse der größten Länge die größte Breite bestimmt (siehe Abb. 2.13).

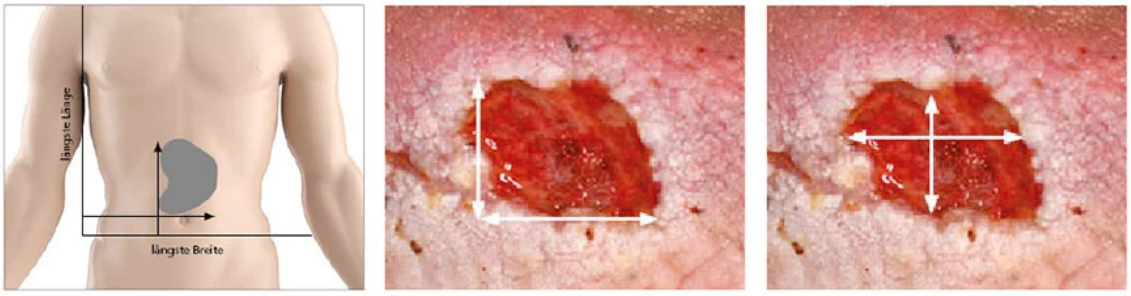


Abbildung 2.12: Körperachsen-Methode
(Quelle: Hartmann, 2023)

Abbildung 2.13: Perpendikular-Methode
(Quelle: Hartmann, 2023)

Da diese Verfahren allerdings viel Zeit und Aufwand benötigen und nicht immer genügend Kapazitäten des Personals vorhanden sind, gibt es viele Bestrebungen, diesen Prozess durch moderne Technik zu automatisieren. Im folgenden Kapitel werden verwandte Arbeiten vorgestellt, die unter anderem mittels Deep Learning Methoden eine halb- bzw. vollautomatisches Messungsverfahren für Wunden entwickelt haben.

3 Stand der Technik

Diese Kapitel befasst sich mit aktuellen verwandten Arbeiten, die sich mit der Dokumentation und Messung von Wunden mittels neuralen Netzwerken und/oder AR-Geräten beschäftigt haben. Es wird überprüft, aus welcher Motivation heraus die Forschenden die Konzeption und Implementierung eines digitalen Wundmessungsverfahrens in Angriff genommen haben, welche Ergebnisse diese Arbeiten hervorgebracht haben und welche Beschränkungen diese Ergebnisse vorweisen, die diese Arbeit möglicherweise überwinden kann.

Die Paper von Wang et al., 2020, Klinker et al., 2019 und von Carrión et al., 2022 sehen großes Potenzial in der Nutzung von CNN in der Wundmessung und -dokumentation. Wang et al., 2020 stellt zunächst die aktuelle Ausgangslage der Wundversorgung im US-amerikanischen Gesundheitssystem vor. Denn akute und chronische nicht heilende Wunden stellen eine schwere Belastung für das Gesundheitssystem dar, die medizinischen Kosten für die Behandlung sollen laut den Autoren bis 2024 auf über 22 Milliarden US-Dollar ansteigen. Eine genaue Dokumentation ist für die Beurteilung und Behandlung chronischer Wunden von entscheidender Bedeutung, um den Verlauf der Wundheilung zu überwachen und zukünftige Eingriffe festzulegen. Allerdings ist die manuelle Dokumentation zeitaufwendig und oft ungenau, was sich negativ auf den Behandlungsverlauf der Patienten auswirken kann. Die Segmentierung von Wunden anhand von Bildern wird als eine gute Lösung für diese Probleme betrachtet, die nicht nur die Vermessung des Wundbereichs ermöglicht, sondern auch eine effiziente Dateneingabe in die elektronische Krankenakte ermöglichen soll, um die Patientenversorgung zu verbessern. Das Ziel der Arbeit war es daher, ein vollautomatisches Wundmessungsverfahren auf Basis von CNN umzusetzen, um die Dokumentation effizienter und simpler für Pflegepersonal und Ärzte zu gestalten. Subjekt der Messungen waren chronische Fußgeschwüre. Es wurde ein Modell basierend auf *MobileNetV2* und *CCL* aufgesetzt und mit über 1100 Bildern von Fußgeschwüren trainiert. Dieses eigene Modell wurde dann mit den existierenden Modellen *SegNet*, *VGG16*, *U-Net* und auch Mask R-CNN mithilfe eines umfassenden Experiments analysiert und verglichen. Dabei zeigt das eigenständig erstellte Modell dieser Arbeit eine gute Performance, auch Mask R-CNN konnte bei der Erkennung von Wunden eine hohe Genauigkeit aufwei-

sen. In der Kategorie *Precision* (dt.: Genauigkeit) hat das Netzwerk mit einem Wert von 98,4 % am besten abgeschnitten. Die Genauigkeit eines künstlichen neuronalen Netzwerkes sagt aus, wie viele der positiven Klassifizierungen des Netzwerks tatsächlich korrekt sind. Die Arbeit von Wang et al., 2020 spezifiziert jedoch kein Endgerät für die Nutzung des CNN, wie die HoloLens 2 in dieser Arbeit, und konzentriert sich nur auf die Segmentierung von Wundbildern, ein Verfahren zur Ermittlung der realen Größe ist nicht gegeben.

In der Arbeit von Carrión et al., 2022 wurde ebenfalls ein auf Deep Learning basierendes System zur Größenschätzung von Wunden entwickelt. In dieser Arbeit wurde jedoch auch ein Verfahren zur Bestimmung von Wundgrößen entwickelt. Als Trainingsdatensatz wurden in dieser Arbeit 256 Bilder von verwundeten Labormäusen verwendet. Die Messung der Wundfläche wird mittels eines Rings in oranger Farbe, dass um die Wunden herum platziert wird, dass als Referenzobjekt auf dem Foto dient. Auch wenn das entwickelte System als sehr robust evaluiert wurde, kann dieses Verfahren nicht vollständig als *automatisch* bezeichnet werden, da dieser Schritt in der Vermessung notwendig ist.

Klinker et al., 2019 hat ein anderes Verfahren, ohne den Einsatz von Deep Learning, entwickelt und evaluiert. In der Arbeit wird ebenfalls die Notwendigkeit eines moderneren Verfahrens betont. Es wird beschrieben, dass der Vorgang der Wundfotografie und Vermessung oft eine Herausforderung darstellt. Manchmal sind für eine solche Dokumentation mehrere Pflegekräfte erforderlich, um beispielsweise an schwer erreichbaren Körperregionen des Patienten zu fotografieren. Das ist zeitaufwendig und verstärkt die ohnehin schon hohe Arbeitsbelastung zusätzlich. Die Dokumentation wird aus hygienischen Gründen in der Regel im Stationszimmer geschrieben. Die Dokumentation wird daher erst nach Abschluss der Wundbehandlung erstellt und erfolgt nicht zeitnah. Folglich müssen sich die Pflegekräfte spezifische Details über die Wunde merken oder, wie in Abbildung 3.1 zu sehen, aufschreiben, bis sie den Rechner erreichen. Die daraus resultierende Wunddokumentation wird häufig als ungenau und teilweise unvollständig beschrieben. Aus diesem Grund wurde ähnlich wie in dieser Arbeit auch ein digitales Verfahren für die Messung von Wunden auf der HoloLens (erste Generation) umgesetzt und untersucht, allerdings ohne Einsatz von Deep Learning, was dieses Wundmessungsverfahren nicht vollständig automatisiert. Wie in

Abbildung 3.2 zu sehen, misst der Nutzende die Wunde, in dem auf dem Spatial Mesh, welches hier den Körper des Messsubjekts repräsentiert, manuell zwei Ankerpunkte gesetzt werden. Die Distanz zwischen den Ankerpunkten wird anschließend als eine Größendimension der Wunde in Meter gespeichert.

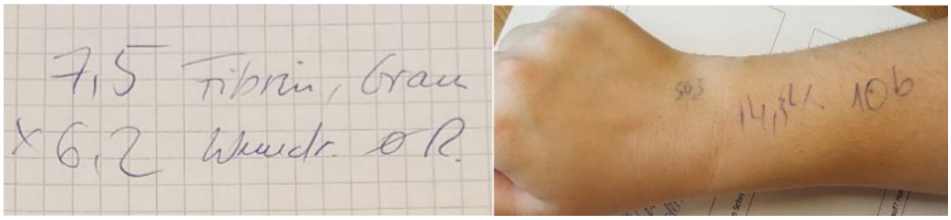


Abbildung 3.1: Von Pflegekräften erstellte Notizen
(Quelle: Klinker et al., 2019)

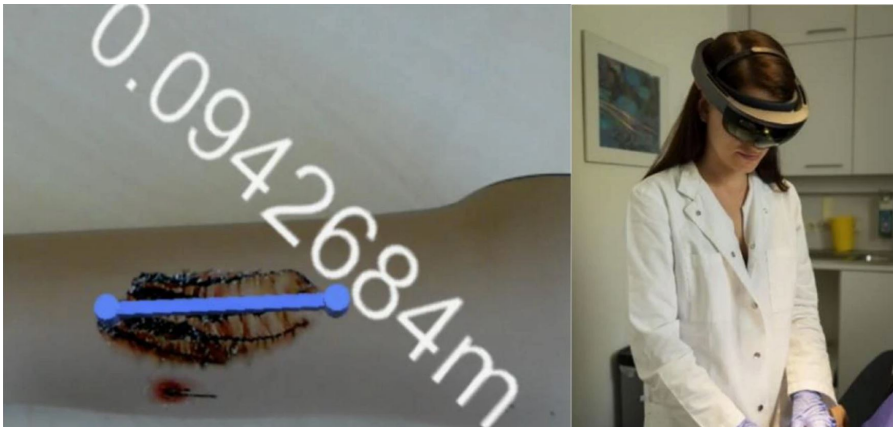


Abbildung 3.2: Manuelle Messung einer Wunde mit virtuellem Maßband
(Quelle: Klinker et al., 2019)

Offensichtlich ist diese Methode jedoch zeitlich aufwendig, da der Nutzende mit dem Maßband auf umständliche Weise hantieren muss. Außerdem ist die Methode dadurch fehleranfällig und uneinheitlich, da jeder Nutzende im Prinzip eine andere Wundmessungsmethode anwenden kann, also Körperachsen- oder Perpendikular-Methode, und somit zu unterschiedlichen Messergebnissen kommt.

Wie die Arbeiten, die in diesem Kapitel vorgestellt wurden, aufzeigen, ist, dass die Entwicklung eines praxistauglichen, automatisierten Wundmessungsverfahrens großes Potenzial für die Verbesserung der Behandlung von Patienten in den Krankenhäusern bietet. Aus dieser Motivation heraus wurde diese Arbeit konzipiert. Im folgenden

Kapitel wird das Konzept der zu entwickelten HoloLens-Anwendung und des Wundmessungsverfahrens anhand der von der Tiplu GmbH entwickelten funktionalen und nicht-funktionalen Anforderungen vorgestellt.

4 Konzept

Bevor das Projekt mit den behandelten Grundlagen aus dem vorigen Kapitel vollständig umgesetzt werden kann, müssen zunächst die Anforderungen betrachtet werden, damit die Anwendung nach diesen konzipiert werden kann. In diesem Kapitel werden die von der Tiplu GmbH gestellten funktionalen und nicht-funktionalen Anforderungen präsentiert. Nach Partsch, 2010, S. 25 sind Anforderungen „Aussagen über zu erfüllende Eigenschaften oder zu erbringende 'Leistungen' eines Systems [...]". Die übliche Klassifikation von Anforderungen ist die Abgrenzung von funktionalen und nicht-funktionalen Anforderungen (vgl. Partsch, 2010, S. 27). Die Anforderungen wurden gemeinsam mit der Tiplu GmbH im Rahmen der Konzeption dieser Arbeit aufgestellt.

4.1 Funktionale Anforderungen

Funktionale Anforderungen legen die bereitzustellenden Funktionen oder den Service eines Softwaresystems bzw. einer seiner Komponenten fest (Balzert, 2009, S. 456). Sie ergeben sich als Antwort auf die Fragen: „Was tut das System?" bzw. „Was soll es aufgrund der Aufgabenstellungen können?" (Partsch, 2010, S. 27). In der folgenden Auflistung werden die funktionalen Anforderungen an die AR-Anwendung vorgestellt:

1. **Patient erkennen**: Damit der Nutzende die korrekten Informationen des Patienten abrufen kann, muss eine Möglichkeit gefunden werden, den Patienten korrekt und möglichst einfach zu identifizieren. Als simple Methode eignet sich der Einsatz von QR-Codes, die das System auf die richtigen Daten zugreifen lassen. Dies sollte nach folgendem Szenario geschehen: Ein/e Arzt/Ärztin oder eine Pflegekraft betritt mit der HoloLens 2 das Patientenzimmer. Der Patient hat ein QR-Code am Handgelenk oder am Bett angebracht. Die HoloLens 2 soll den QR-Code scannen und die Daten des Patienten aufrufen und darstellen. Für den in dieser Arbeit zu entwickelnden Prototyp soll eine echte Anbindung an ein Patienten-Datensystem zunächst nicht umgesetzt werden. Es soll zunächst mit

lokalen Platzhalter-Daten für eine demonstrative Darstellung der Informationen gearbeitet werden.

2. **Visiten-Aufgabenauswahl**: Nachdem der korrekte Patient im System geladen wurde, sollen 2 bis 5 verschiedene Visiten-Aufgaben zu dem Patienten angezeigt werden, die der Nutzende in beliebiger Reihenfolge per Interaktion mit Checkboxen auswählen kann. Eine Visiten-Aufgabe könnte z. B. Blutdruckmessung oder Pulsmessung sein.

3. **Vitaldaten per Sprache aufnehmen**: Die Messwerte einer Vitalfunktionskontrolle sollen per Sprache aufgezeichnet werden. Dabei sollen aus dem erkanntem Text des gesprochenen Satzes strukturierte Werte erkannt werden. Beispielsweise wenn der Blutdruck gemessen wurde, sollen aus dem Satz „*Der systolische Wert beträgt 120, der diastolische Wert beträgt 80*“ die Variablen *systolisch = 120* und *diastolisch = 80* gefiltert werden. Dabei sollen mehrere Varianten der möglichen Begriffe und Sätze möglich sein, z. B.: „Der Blutdruck beträgt 120 zu 80“.

4. **Wundmessung und -dokumentation**: In einem anderen Aufgabentyp soll die Wunde fotografiert und die Größe automatisch mittels Einsatz eines Objektsegmentierungsverfahrens dokumentiert werden (z. B. Breite 30 mm, Länge 80 mm). Die Ermittlung der Wundtiefe soll zunächst außer Acht gelassen werden.

4.2 Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben die Vorgaben und Eigenschaften des zu erstellenden Zielsystems. Sie beinhalten spezifische Angaben und Einschränkungen, wie und unter welchen Umständen das System erstellt werden soll (vgl. Partsch, 2010, S. 27 ff.). Sie setzen also einen technischen Rahmen für die Umsetzung der Software. Die folgende Auflistung zeigt die nicht-funktionalen Anforderungen an die AR-Anwendung:

1. **Nutzbarkeit**: Die Benutzeroberfläche der Anwendung sollte intuitiv und benutzerfreundlich gestaltet sein, damit medizinisches Fachpersonal sie leicht er-

lernen und effizient nutzen kann. Es sollten klare Anweisungen, gut sichtbare Symbole und eine einfache Navigation vorhanden sein.

2. **Performance**: Die Anwendung sollte eine gute Performance bieten, um eine reibungslose und schnelle Ausführung zu gewährleisten.

3. **Kostengünstigkeit**: Die Anwendung sollte mit möglichst kostengünstigen bzw. kostenfreien Tools und Frameworks umgesetzt werden.

5 Implementierung

In diesem Kapitel wird die Umsetzung der in Kapitel 3 besprochenen Anforderungen mit den in Kapitel 2 erläuterten Technologien näher beleuchtet. Angefangen mit der Einrichtung des Unity-Projekts und der Umsetzung der einzelnen Features in der HoloLens-2-Anwendung in Unity 3D, bis hin zur Entwicklung des Wundmessungsverfahrens mittels Mask R-CNN.

Zunächst wurde ein Unity-3D-Projekt mit der Version 2021.3.17f1 und MRTK 2 aufgesetzt. Das MRTK bietet auch Vorlagen für Szenen, die alle grundlegenden GameObjects und Komponenten beinhaltet. In solch einer Szene befindet sich das GameObject *MixedRealityToolkit*, an welches die Komponente mit selbem Namen angefügt ist. Sie bietet eine Vielzahl an Konfigurationsoptionen, u. a. Kameraeinstellungen oder das Anlegen von Sprachkommandos. Auch die Spatial-Awareness-Funktion lässt sich hierüber aktivieren. Da das Wundmessungsverfahren, welches in einem späteren Unterkapitel erläutert wird, auf die Tiefendaten der HoloLens 2 zugreifen muss, wurde diese Funktion aktiviert. Das GameObject *MixedRealityPlayspace* repräsentiert den Beobachtungsstandpunkt des Nutzenden der HoloLens 2. Als Kindobjekt ist ein Kamera-Objekt angefügt, dessen Position und Orientierung durch die Kopfbewegung des Nutzenden in der Anwendung manipuliert wird.

Für den generellen Ablauf der Anwendung wurde die Komponente *AppManager* erzeugt, die als GameObject in der Szene platziert wurde. Sie regelt die Initialisierung aller benötigten Komponenten und schaltet dynamisch je nach Fortschritt in der Nutzung der Anwendung die benötigten UI-Elemente und Komponenten an bzw. aus. Die *enum*-Klasse *AppState* beinhaltet eine Liste aller Fortschrittsstadien der Anwendung (siehe Listing 2).

```
1 public enum AppState
2 { WAIT, STARTINFO, QRSCANSTART, QRSCANSTOP, QRSCANLOADING, WAITFORQRCODE,
      PATIENTINFO, STARTTASKS, STARTSPEECH, STARTWOUND, RESET }
```

Listing 2: AppState: Liste aller Fortschrittsstadien der Anwendung

Ist als Stadium *QRSCANSTART* ausgewählt, startet die Anwendung den QR-Code-Scanner, der für die Umsetzung der ersten funktionalen Anforderung zur Erkennung des Patienten mittels QR-Code benötigt wird. Dies wird im folgenden Unterkapitel behandelt.

5.1 Patient erkennen

Für die QR-Code-Verwaltung wurde nach Anleitung von Microsoft (Wen, 2022) das GitHub-Projekt *MixedRealty-QRCode-Sample* eingebunden (Microsoft, 2022a). Das Projekt enthält Prefabs, die in das eigene Unity-Projekt eingebunden werden können, samt den für das QR-Code-Tracking benötigte Komponenten. Die Komponenten *QrCodesManager* und *QRCodesVisualizer* verwalten das aktiveren der QR-Funktion und ermöglichen auch das Initialisieren von Prefabs zu neuen GameObject in der Szene, wenn ein QR-Code erkannt wurde. Diese GameObject verfolgen auch die Position und Rotation des QR-Codes im realen Raum. Sobald die Anwendung ein QR-Code eines im Patientensystem enthaltenden Patienten erfasst, wird ein Button mit der Aufschrift *Bestätigen* über jenen QR-Code dargestellt, samt Namen des Patienten als Label (siehe Abbildung 5.1).

Da diese Anwendung zunächst nur demonstrativ das Abrufen der Patientendaten aus einer echten Datenbank simulieren soll, wurde ein Dummy-Datensatz als json-Datei im Asset-Ordner des Unity-Projekts angelegt (*Assets/ExampleData/patients.json*), welche Informationen zu einigen Beispiel-Patienten enthalten, die dem Nutzenden dargestellt werden sollen. Für die Darstellung der Daten wurde ein Board aus den Prefabs des MRTK erstellt (siehe Abbildung 5.2), welches oben links zunächst Name, Geschlecht, Alter, Aufnahmedatum und Station ausgibt. Unten links werden die Vitalwerte aus der letzten Visite des Patienten ausgegeben. Wurde noch keine Visite gemacht, bleibt der Bereich leer. Oben rechts werden Informationen zu Diagnose, Be-

Abbildung 5.1: QR-Code

schwerden und Maßnahmen ausgegeben. Die Daten werden über die Komponente *ExampleDataManager* aus der Datei gelesen und sind somit für alle anderen Komponenten verfügbar. Das Eltern-GameOject des Boards enthält die Komponente *PatientDataBoard*, welches mit der öffentlichen Funktion *DisplayPatientDataForAccept* als Argument den Patienten-Datensatz entgegennimmt und in der Ansicht des Boards darstellt.

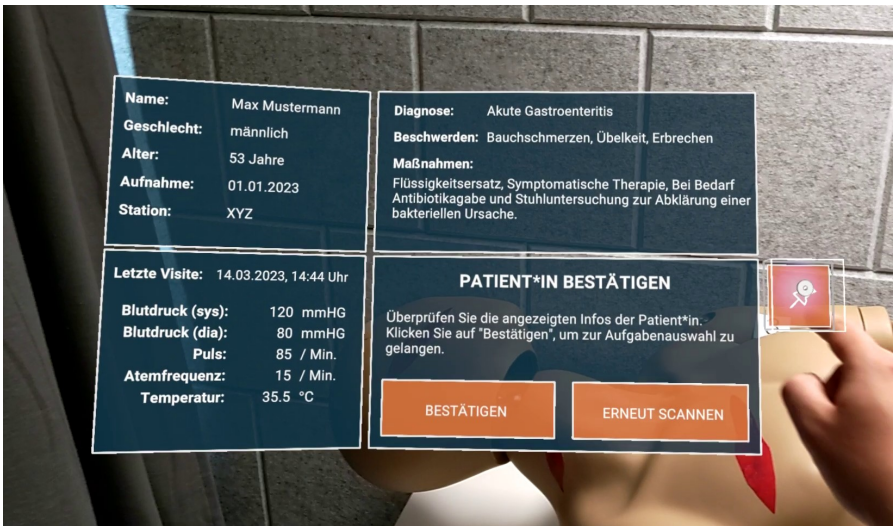


Abbildung 5.2: Patienten-Daten

Die Design-Eigenschaften wie Schriftart und Farbschema wurden aus dem Corporate Design der Tiplu GmbH übernommen. Hat der Nutzende die Daten bestätigt, gelangt dieser über den *Bestätigen*-Button zum nächsten Schritt gelangen.

5.2 Aufgaben auswählen

Die auswählbaren Visiten-Aufgaben werden ebenfalls mittels *ExampleDataManager* aus einer json-Datei gelesen, die im Unity-Projekt hinterlegt wurde (*Assets/ExampleData/tasks.json*). Bei realem Einsatz sollte die Anwendung in der Lage sein, individuelle Listen von Aufgaben je Patient laden zu können, für den Prototyp wurde eine beispielhafte Liste angelegt. Die Komponente *TaskSelector* liest diese Daten aus und stellt sie Auswahl in Form einer Reihe von Checkboxen, die der Nutzende beliebig anklicken kann. Es sollte nämlich möglich sein, bestimmte Aufgaben auszulassen, falls diese nicht täglich durchgeführt werden müssen. Die Aufgaben *Blutdruck*, *Puls*, *Temperatur* und *Atem* sind hierbei vom Typ Sprache, das bedeutet sie werden über die Spracherkennung der Anwendung aufgezeichnet und gespeichert. Die Aufgabe *Wunddoku* wird hierbei aufgrund der Komplexität der Datenaufzeichnung gesondert behandelt und erst nach der Spracheingabe durchgeführt. Hat der Nutzende die Aufgaben ausgewählt, gelangt dieser zur Vitaldaten-Spracheingabe, sollte eine Sprachaufgabe ausgewählt sein.

5.3 Spracherkennung

Hat der Nutzende eine oder mehrere Aufgaben vom Typ Sprache ausgewählt, erscheint nun in der Anwendung ein Board, welches die aufzuzeichnenden Vitaldaten auflistet. Wird nun beispielsweise der Satz „Der Blutdruck beträgt 101 zu 90.“ oder Sätze „Systolisch 101“ und „Diastolisch 90“ ins Mikrofon gesprochen und korrekt erkannt, werden diese Vitaldaten entsprechend in der Liste ausgegeben. Dabei kann der Nutzende auch die Werte nachträglich korrigieren, in dem der Satz nochmal gesprochen wird. Diese wird im folgenden Unterkapitel näher erklärt.

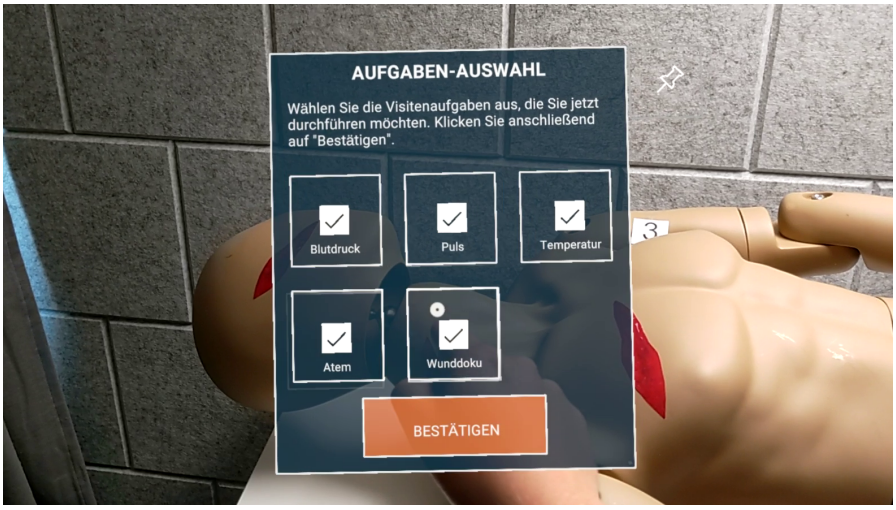


Abbildung 5.3: Visitenaufgaben-Auswahl

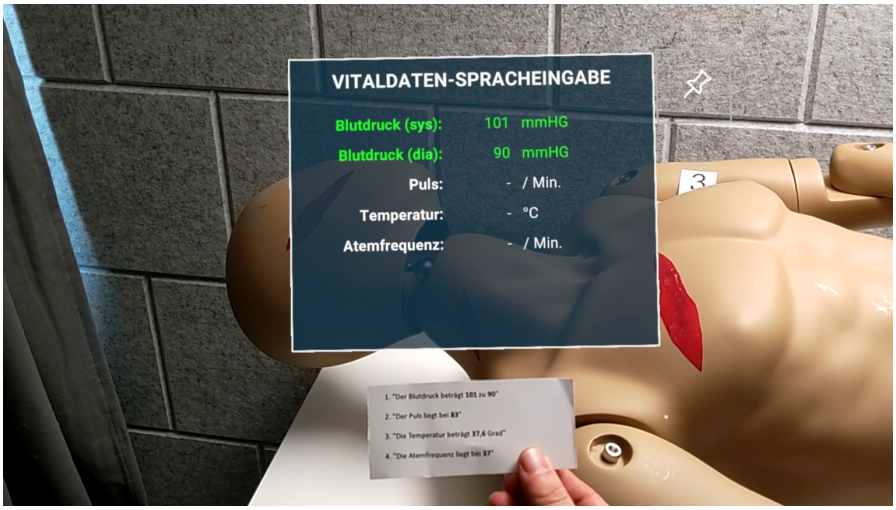


Abbildung 5.4: Aufnahme der Vitaldaten per Sprache

Für die Spracherkennung wurde die Komponente *SpeechRecognitionManager* erzeugt, die die Klasse *GrammarRecognizer* aus der *Speech*-Bibliothek von Microsoft inkludiert. Der *GrammarRecognizer* wird über die Funktion *InitGrammarRecognizer* (siehe Listing 3) initialisiert und erhält als Argument eine Referenz zu der XML-Datei */Assets/StreamingAssets/SRGS/myGrammar.xml*.

```
...
private void InitGrammarRecognizer()
{
    grammarRecognizer = new GrammarRecognizer(Application.streamingAssetsPath
        + "/SRGS/myGrammar.xml");
    grammarRecognizer.OnPhraseRecognized += Grammar_OnPhraseRecognized;
}
...
```

Listing 3: Initialisierung des GrammarRecognizer

SRGS steht für *Speech Recognition Grammar Specification* und handelt sich um eine XML-basierte Sprachgrammatik, die verwendet wird, um die sprachlichen Regeln und Strukturen festzulegen, die bei der Spracherkennung verwendet werden sollen (Hunt und McGlashan, 2004). SRGS wird hauptsächlich im Bereich der Spracherkennungstechnologien eingesetzt, beispielsweise bei der Entwicklung von Sprachassistenten oder interaktiven Sprachsystemen. SRGS ermöglicht es Entwicklern, die erwarteten Benutzereingaben in einer definierten Grammatik zu spezifizieren. Dies umfasst die Definition von Wortschatz, Grammatikregeln, Satzstrukturen und möglichen Antworten oder Aktionen des Systems. Durch die Verwendung von SRGS können Anwendungen und Geräte Benutzeranfragen effektiv analysieren und interpretieren, um entsprechende Aktionen auszuführen oder relevante Informationen zurückzugeben.

Spricht der Nutzende nun einen Satz zur Aufzeichnung von Vitaldaten, wird zunächst die Spracheingabe in Text umgewandelt und vom *GrammarRecognizer* mittels den Regeln der SRGS-Datei im Projekt interpretiert. Wenn z. B. „Der Puls beträgt 80“ gesprochen wurde, erkennt das System aus dem Satz das Schlagwort „Puls“ und die entsprechende Zahl und wandelt diese jeweils in die Datentypen *string* und *integer* um.

5.4 Wundmessungsverfahren

Die Umsetzung des praxistauglichen Wundmessungsverfahrens ist in zwei Phasen gegliedert: das Trainieren des Mask-R-CNN-Modells mittels gelabelten Wundbildern und die Entwicklung der eigentlichen Wundgrößenberechnung auf der HoloLens 2

mittels Webserver und Mask R-CNN. Zunächst wird die Trainingsphase näher betrachtet. Dafür wurde das von den Entwicklern von Mask R-CNN veröffentlichte GitHub-Repository, welches eine Python-Umgebung für das Trainieren und Nutzen des Modells mit entsprechenden Beispielen zur Verfügung stellt, genutzt (Matterport, 2017). Dieses Repository nutzt *tensorflow* und *keras* als Framework.

5.4.1 Mask R-CNN Training

Für den Trainingsdatensatz wurden Bilder von Wunden aus Adobe Stock verwendet. Aus den insgesamt 40 Bildern von Wunden wurden 32 für das Training und 8 für die Validierung eingeteilt. Die Abbildung 5.5 zeigt 3 Beispiele aus dem Trainingsdatensatz. Für die Labeldaten der Wundmasken auf den Bildern wurde das Tool *VGG Image Annotator* (kurz: VIA) eingesetzt. In diesem Programm lassen sich über einen grafischen Editor 2D-Polygone auf Bilder manuell platzieren. Die 2D-Koordinaten der Punktwolke der Polygone werden im json-Format zusammen mit einer Referenz zum Bild gespeichert (*/datasets/wound/train/via_region_data.json*). Diese Punktwolken dienen dem Modell als Wahrheitswert der Wundbilder im Training. Dasselbe wurde auch für den Validierungsdatensatz durchgeführt.

Abbildung 5.5: Auswahl von Trainingsdatensätzen

Bevor das Training gestartet werden konnte, musste zunächst eine Config-Klasse angelegt werden (siehe Listing 5), die von der Standard-Config-Klasse erbt. Hier können sämtliche Config-Variablen überschrieben werden. Als Backbone-Netzwerk wurde standardmäßig *Resnet101* ünernommen. *IMAGES_PER_GPU* gibt an, wie viele Bilder simultan pro Steps von der Grafikkarte des Rechners verarbeitet werden sollen.

Aufgrund der limitierten Rechenleistung der Grafikkarte, die in diesem Projekt verwendet wurde, konnte diese Zahl nicht erhöht werden. *NUM_CLASSES* legt fest, wie viele zu segmentierende Klassen trainiert werden sollen. Dies beinhaltet in diesem Falle die Klassen *wound* und *background*. *STEPS_PER_EPOCH* gibt an, wie viele Trainingsschritte pro Epoche durchgeführt werden sollen, *VALIDATION_STEPS* gibt die Anzahl der Validierungsschritte an. Um den Prozess zusätzlich zu beschleunigen, wurde die als maximale Bildgröße 512 × 512 mit der Variabel *IMAGE_MAX_DIM* festgelegt. Das Mask-R-CNN-Modell reduziert damit die Größe der Bilder, um ein schnelleres Training zu gewährleisten. Das Training soll insgesamt 14 Epochen (*NUM_EPOCHS* dauern, d. h. der gesamte Trainingsdatensatz soll 14-mal durchlaufen. Dies hat bei vorigen Tests den niedrigstmöglichen Verlust-Wert des Modells ergeben (0,39).

```
class WoundConfig(Config):
    BACKBONE = "resnet101"
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1
    NUM_CLASSES = 2
    NUM_EPOCHS = 14
    STEPS_PER_EPOCH = 500
    LEARNING_RATE = 0.001
    VALIDATION_STEPS = 50
    IMAGE_MIN_DIM = 512
    IMAGE_MAX_DIM = 512
    DETECTION_MIN_CONFIDENCE = 0.9
```

Listing 4: WoundConfig für Training

Aufgrund der geringen Anzahl an Trainingsdatensätzen und um z. B. das Netzwerk auf verschiedene Größen und Licht- und Farbverhältnissen zu trainieren, wurde das Python-Paket *imgaug* eingesetzt, um eine Reihe von Datenerweiterungsoperationen während des Trainingsprozesses durchzuführen. Wie in Listing 5 zu sehen, lassen sich die Operationen zufallsbasiert auf den gesamten Datensatz anwenden. Pro Epoche wird nämlich diese Datenerweiterung neu in das System geladen, was für eine größere Variation der Datensätze sorgt. Die Wundbilder werden unter anderem horizontal oder vertikal gespiegelt, Kontrastwerte werden erhöht oder gesenkt und die Bilder werden rotiert und / oder verzerrt.

```
augmentation = iaa.Sometimes(p, iaa.Sequential([
   iaa.Fliplr(0.5),
   iaa.Flipud(0.5),
   iaa.Sometimes(0.5, iaa.GaussianBlur(sigma=(0, 0.5))),
   iaa.ContrastNormalization((0.5, 1.5)),
   iaa.AdditiveGaussianNoise(loc=0, scale=(0.0, 0.05*255), per_channel=0.5),
   iaa.Multiply((0.5, 1.5)),
   iaa.Sometimes(0.5,
      iaa.Affine(
         scale={"x": (0.5, 1.5), "y": (0.5, 1.5)},
         translate_percent={"x": (-0.2, 0.2), "y": (-0.2, 0.2)},
         rotate=(-45, 45),
         shear=(-8, 8)))
], random_order=True))
```

Listing 5: Datenerweiterungsoperationen von imgaug

Wie in Listing 6 zu sehen, wird diese Datenerweiterungsoperationen mit an die Trainings-Funktion übergeben. Das Argument *layers = heads* gibt an, dass das Modell nur auf den oberen Schichten des Netzwerks trainiert werden soll. Das Transfer-Learning wurde hierbei auf den COCO-Datensatz von Microsoft angewandt, nach Empfehlung der Entwickler. Dieser Datensatz wurde auf über 300.000 Bildern trainiert und ist auf Objektsegmentierung spezialisiert.

```
...
model.train(dataset_train, dataset_val,
         learning_rate=config.LEARNING_RATE,
         epochs=config.NUM_EPOCHS,
         layers='heads',
         augmentation=augmentation)
...
```

Listing 6: Trainings-Funktion

Nach dem Trainings-Prozess wurde die Datei */logs/wound_weights_coco.h5* abgelegt. Diese enthält alle weights des Mask-R-CNN-Netzwerks, die für die Erkennung von Wunden angepasst wurden. Diese Datei wurde anschließend auf dem lokalen Webserver abgelegt, damit es möglich ist, extern über eine HTTP-Anfrage darauf zugreifen zu

können. Die Einrichtung des Servers wird im folgenden Unterkapitel näher beleuchtet.

5.4.2 Berechnung der Wundgrößen

Zunächst musste die Foto-Funktion der HoloLens 2 umgesetzt werden. Hierfür wurde die Komponente *WoundMeasurementManager* erzeugt, die auf die Klasse *PhotoCapture* zugreift, die aus der Windows-Bibliothek in Unity inkludiert wird. Diese ermöglicht das Erstellen und Verarbeiten von Fotos mittels der Foto-/Videokamera auf der Vorderseite der HoloLens 2. Bevor das Foto jedoch erzeugt werden kann, muss der Nutzende zunächst die Kamera auf die Wunde ausrichten. Als Indikator für die Mitte des zu erstellenden Fotos wurde ein grüner, kreisförmiger Cursor in die Szene angelegt (siehe Abbildung 5.6).

Abbildung 5.6: Fotomodus

Bevor das Foto geschossen wird, sendet das System einen Ray von der Position der Kamera im realen Raum aus und überprüft, ob es mit dem Spatial-Awareness-Mesh kollidiert (siehe Listing 7). Der Abstand zwischen der Kamera und der Kollision wird ermittelt und für die Berechnung der Wundgröße im späteren Verlauf verwendet.

```
...
public float GetHitDistance(Vector3 cameraPosition, Vector3 cameraForward)
{
   RaycastHit hit;
   Ray ray = new Ray(cameraPosition, cameraForward);

   if (Physics.Raycast(ray, out hit))
   {
      return hitDistance = Vector3.Distance(cameraPosition, hit.point);
   }
   return 0.0f;
}
...
```

Listing 7: Berechnung der Distanz von Kamera zur Wunde

Hat sich der Nutzer korrekt und senkrecht zur Wunde ausgerichtet, kann er mit dem Sprachbefehl „Foto“ das Bild erstellen. Dem Nutzenden wird danach darauf hingewiesen, dass das Bild verarbeitet wird. Nun wird das Bild über eine HTTP-Anfrage an den lokalen Webserver für die Erkennung der Wundmaske eingebunden und gesendet.

Der lokale Webserver wurde in einer Python-Umgebung lokal auf dem Rechner mithilfe des kostenlosen Frameworks *Flask* aufgesetzt und ist über die IP-Adresse des WLAN-Netzwerkes erreichbar. Flask ist ein leichtgewichtiges Web-Framework für die Programmiersprache Python, das entwickelt wurde, um die Erstellung von Webanwendungen und APIs einfach und schnell zu gestalten. Es zeichnet sich durch seine minimalistische Natur, Flexibilität und modulare Architektur aus und bietet Funktionen wie Routing, Vorlagen-Rendering und HTTP-Anfrageverarbeitung (Pallets, 2023). Auf dem Server wurde der Quellcode des Mask-R-CNN-Modells eingerichtet, der auch für das Training verwendet wurde.

Der Server gibt das Bild mit der erkannten Maske und den berechneten Breite- und Längenmaß in Pixel an die HoloLens-Anwendung zurück. Die Pixelmaße werden dann zusammen mit den Pixelmaßen des Bildes, der gemessenen Distanz zur Wunde und den hinterlegten Kamera-Parametern Brennweite (f) und Sensorgröße (Breite, Länge) berechnet. Grundlage der Berechnung ist ein von Fulton, 2023 vorgestelltes Verfahren.

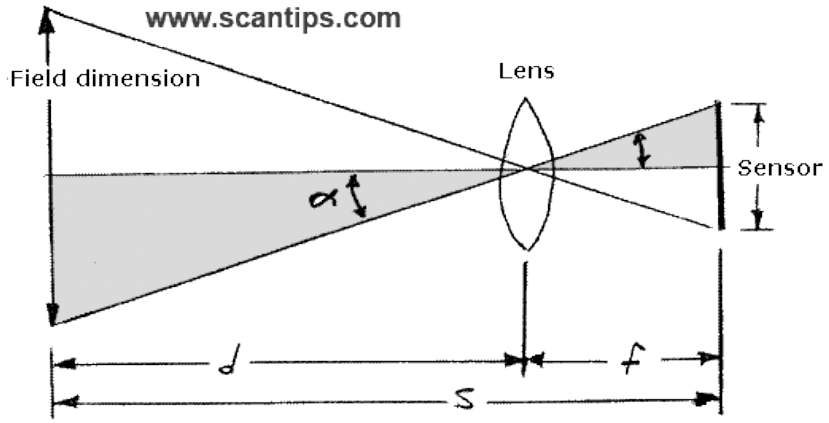


Abbildung 5.7: Scantips
(Quelle: Fulton, 2023)

Die Abbildung 5.7 zeigt ein einfaches Diagramm einer Linse mit den Größenverhältnissen zwischen der Sensorgröße der Kamera und des erfassten Objektes (*Field dimension*).

$$\frac{Sensor\ dimension\ (mm)}{Focal\ length\ f} = \frac{Field\ dimension\ (m)}{Distance\ to\ field\ d\ (m)}$$

Um z. B. die Länge bzw. die Höhe des Objektes auf dem Bild in Millimeter zu erhalten, wird *Field dimension* mit der Höhe des Objekts ersetzt und danach aufgelöst:

$$Object\ height\ on\ sensor\ (mm) = \frac{Sensor\ height\ (mm) * Object\ height\ (px)}{Sensor\ height\ (px)}$$

Dasselbe kann auch für die Breite des Objektes angewendet werden. Ist die Höhe des Objektes auf dem Sensor ermittelt, kann sie wie folgt in Verhältnis gesetzt werden:

$$\frac{Object\ height\ on\ sensor\ (mm)}{Focal\ length\ (mm)} = \frac{Real\ object\ size\ (m)}{Distance\ to\ object\ (m)}$$

Wird diese Gleichung nun nach *Real Object Height* umgestellt, erhält man die reale Höhe des Objektes auf dem Bild in Meter.

$$Real\ Object\ height\ (m) = \frac{Distance\ to\ object\ (m) * Object\ height\ on\ sensor\ (mm)}{Focal\ length\ (mm)}$$

Wichtig ist bei dieser Berechnung zu betrachten, dass der Blickwinkel der Kamera zum Objekt so senkrecht wie möglich eingestellt wird, damit keine Verzerrungen in die Berechnung einfließen. Die Korrektheit dieser Berechnung konnte durch Tests mit bekannter Distanz und Pixelmaßen bestätigt werden. Die PV-Kamera der HoloLens 2 hat eine durchschnittliche Brennweite von 4,7 mm und eine Sensorgröße von 6,4 × 3,6 mm. Mit diesen Parametern zusammen mit der ermittelten Distanz in Meter und der Größe der Wundmasken in Pixel auf den segmentierten Bildern werden die realen Wundgrößen berechnet. Die Pixelmaße des Wundobjekts auf den Bildern werden anhand eines rotierten Rechtecks ermittelt, das die Maske entlang der längsten Seite umschließt. Die Größe des Dreieckes wird anschließend auf dem Bild ausgemessen. Die Umrechnung in Millimeter-Maße erfolgt in der Unity-Klasse *WoundMeasurement* (siehe Listing 8).

```
1 public float[] GetWoundSizeInMillimeters(float widthPx, float lengthPx,
      float imgWidth, float imgHeight) {
2    float widthOnSensor = (sensorWidth * widthPx) / imgWidth;
3    float objectWidth = (distance * widthOnSensor) / focalLength;
4    float lengthOnSensor = (sensorHeight * lengthPx) / imgHeight;
5    float objectLength = (distance * lengthOnSensor) / focalLength;
6
7    float[] result = new float[2];
8    float width = objectWidth * 1000;
9    float length = objectLength * 1000;
10
11   result[0] = length;
12   result[1] = width;
13
14   if (width > length)
15   {
16      result[0] = width;
17      result[1] = length;
18   }
19   return result; }
```

Listing 8: Berechnung der Wundgrößen

Die Messergebnisse und das Bild mit der Maske werden nach der Berechnung dem Nutzenden zur Bestätigung auf dem GameObject *WoundDataBoard* ausgegeben (siehe Abb. 5.8). Nachdem der Nutzende das Bild zurückbekommt, erhält er die Möglichkeit, die Fotografie zu wiederholen. Bei Klick auf den Button mit dem Schriftzug *Weiter* wird das Bild samt den Maßen abgespeichert und kann gegebenenfalls an ein externes System über eine Webschnittstelle übertragen werden.

Damit die Nutzbarkeit der Anwendung und die Messgenauigkeit des entwickelten Verfahrens empirisch evaluiert werden kann, muss zunächst der Versuchsaufbau und -durchführung erläutert werden. Im folgenden Kapitel wird darauf eingegangen.

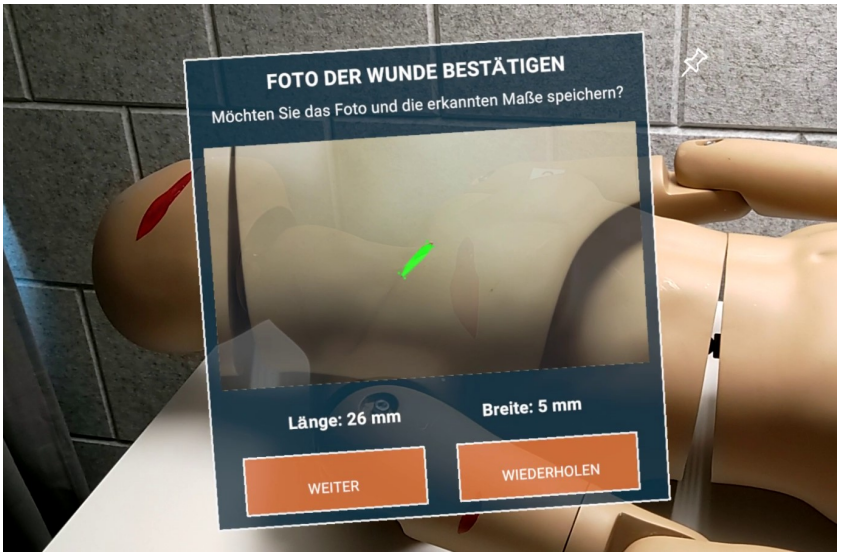


Abbildung 5.8: Bild mit erkannter Wunde

6 Evaluation

In diesem Kapitel wird der Versuchsaufbau zur Untersuchung der allgemeinen Nutzbarkeit der Anwendung und der Genauigkeit des Wundmessungsverfahrens beschrieben.

6.1 Versuchsaufbau

Für den Versuch wurde in einer Laborumgebung ein Krankenhausszenario mit Patienten simuliert. In einem Raum der Tiplu GmbH, in dem stets für die gleichen Lichtverhältnisse durch Verdunkeln der Fenster gesorgt wurde, wurde eine Schaufensterpuppe auf ein Schreibtisch gelegt. Ein QR-Code mit Referenz zu einem Beispielpatienten (Max Mustermann) wurde neben der Puppe angebracht. Für die Aufnahme von Vitaldaten wurden für jeden einzelnen Versuchsteilnehmer zufällige Blutdruck-, Puls-, Temperatur- und Atemfrequenzwerte auf einem Zettel ausgedruckt. Zur Untersuchung der Wundmessung wurden 8 künstliche, auf Papier ausgedruckte Wunden auf dem gesamten Körper der Schaufensterpuppe angebracht und von 1 bis 8 durchnummeriert (siehe Abbildung 6.1 und 7.1).

Die Wunden unterscheiden sich in Länge und Breite und decken Größen zwischen 5 und 145 Millimeter ab. In der folgenden Tabelle 6.1 werden alle Größen aufgelistet:

Nr.	Länge	Breite
1	98	24
2	68	16
3	145	29
4	40	10
5	28	5
6	140	14
7	96	17
8	55	29

Tabelle 6.1: Größe der einzelnen Wunden im Versuch in mm

In der Abbildung 7.1 werden in Leserichtung die Wunden 1 bis 8 dargestellt.

Abbildung 6.1: Schaufensterpuppe mit 8 Wunden

6.2 Versuchsdurchführung

Für die Rekrutierung wurde ein Rundschreiben an die Mitarbeitenden in der Tiplu GmbH in Harburg geschickt, in der um Teilnahme gebeten wurde. Interessierte konnten sich in Timeslots eintragen und pro Versuchsdurchlauf wurde eine Dauer von 20 Minuten geschätzt. Der Versuch wurde in einem abgeschlossenen Raum durchgeführt, um eine ungestörte Nutzung der Anwendung zu gewährleisten. Jeder Teilnehmende erhielt eine ausführliche Einweisung in die Verwendung der HoloLens 2 und der Anwendung. Es wurde darauf geachtet, dass alle Teilnehmende den gleichen Instruktionen und Bedingungen ausgesetzt waren. Die Teilnehmenden sollten den gesamten Ablauf der Anwendung testen. Zunächst sollten sie den QR-Code scannen, die Patientendaten abrufen und alle möglichen Aufgaben auswählen. Anschließend nehmen sie die vorbereiteten Vitaldaten auf und fotografieren alle 8 Wunde nach ihrer Nummerierung ab. Die Messergbnisse und Fotos wurden nach Abschluss der Nutzung an den lokalen Flask-Server übermittelt und in einer SQL-Datenbank gespeichert.

Nach der Nutzung der Anwendung wurden die Teilnehmenden aufgefordert, einen Fragebogen an einem Rechner anonymisiert auszufüllen. Dafür erhielten sie in der

Abbildung 6.2: Alle acht Wunden aus dem Versuch

Anwendung eine zufällig generierte ID, die sie im Fragebogen zu Beginn angeben sollten. Neben demografischen Fragen enthält der Fragebogen auch Fragen zur Bestimmung des *System Usability Scale* (kurz: SUS). Der SUS ist ein einfacher Fragebogen zur Bestimmung der allgemeinen Nutzbarkeit eines Systems oder Anwendung. Er besteht

aus 10 Fragen (siehe Anhang A unter *System Usability Score*), die jeweils mit einer 5-Punkt-Likert-Skala beantwortet werden (Brooke, 1995). Beispiel:

„Ich denke, ich würde die Anwendung regelmäßig nutzen." *Stimme überhaupt nicht zu* o o o o o *Stimme voll und ganz zu*

Dabei können die testenden Nutzenden ihre Zustimmung zwischen 1 für „Stimme überhaupt nicht zu" und 5 für „Stimme völlig zu" festlegen. Dabei haben die einzelnen Antworten zunächst keine Bedeutung. Um den SUS-Score zu berechnen, müssen die einzelnen Skalenwerte transformiert werden. Die Fragen 1, 3, 5, 7, 9 sind hierbei positiv gerichtet, die Fragen 2, 4, 6, 8, 10 sind negativ ausgerichtet. Bei den positiven Fragen muss die Skalaposition minus 1 gerechnet werden, bei den negativen Fragen wird 5 minus die Skalaposition gerechnet. Anschließend wird die Summe aller Werte berechnet und mit 2.5 multipliziert. Das Endergebnis ist dann ein Score-Wert, der von 0 bis 100 reicht. Am Ende des Fragebogens hatten die Teilnehmenden noch die Möglichkeit, über ein Freitextfeld optionales Feedback zur Anwendung zu geben.

Im folgenden Kapitel werden nun die Ergebisse dieser empirischen Untersuchung vorgestellt.

7 Ergebnisse

In diesem Kapitel werden die Ergebnisse der empirischen Untersuchung vorgestellt, welche im vorigen Kapitel beschrieben wurde. Zunächst werden die Ergebnisse des Fragebogens präsentiert und anschließend die Messergebnisse der Wundfotos, die von den Versuchsteilnehmern angefertigt wurden.

7.1 Fragebogen

An der empirischen Untersuchung haben sich 41 Mitarbeitende der Tiplu GmbH gemeldet. Davon waren 32 Personen männlich (78 %) und 9 weiblich (22 %). 14, also 34,1 % der Teilnehmenden, haben angegeben, bereits Erfahrung im Umgang mit der HoloLens 2 oder anderen AR-Systemen zu haben. 8 der Probandinnen haben bereits berufliche Erfahrung mit der Messung von Wunden. Lediglich 2 der Teilnehmenden haben während der Nutzung Beschwerden wie leichte Übelkeit und Schwindel erfahren und eine Person hat angegeben, dass die Nutzung für die Augen belastend war.

Die Ermittlung des SUS Score hat folgendes Ergebnis bei den 41 Teilnehmenden ergeben:

SUS Score	SD	Adjektiv
80.79	10.87	Exzellent

Tabelle 7.1: SUS-Fragebogen Ergebnis

Im letzten Teil des Fragebogens hatten die Versuchteilnehmenden optional noch die Möglichkeit, in einem Freitextfeld weitere Anmerkungen zu hinterlegen. Aus den 23 Antworten aus diesem Feld wurden folgende wesentliche Punkte extrahiert:

- Es war einigen Teilnehmenden nicht klar, welchen Bildausschnitt das Wundfoto letztendlich haben wird. Der kreisförmige Cursor zeigte zwar die Mitte des Bildes an, es gab jedoch kein Hinweis, wie das Foto letztendlich aussehen wird. Hier wäre beispielsweise ein grüner Rahmen, der den Bildausschnitt repräsentieren soll, hilfreich gewesen.

- Ein Sound beim Fotografieren wäre für die Nutzenden hilfreich gewesen, den Zeitpunkt des Schnappschusses zu erfassen.
- Die Performance, also die Geschwindigkeit des QR-Code-Scannens und der Wundmessung hätte besser sein sollen.
- Ein Teilnehmender, der auch bereits Erfahrung mit der Wundmessung angegeben hat, könnte sich vorstellen, dass die Nutzung der Anwendung den Klinikalltag erleichtern könnte, da man während der Nutzung die Hände frei hat und nicht unnötig viele Gegenstände wie Kamera und Maßband mit sich tragen muss. Auch das Einsehen der Patientendaten wurde als vorteilhaft betrachtet.

7.2 Messgenauigkeit

Im Rahmen der Untersuchungsphase dieser Arbeit wurden 41 Messungen der jeweiligen Wunden durchgeführt. Im Folgenden werden die ermittelten statistischen Werte der jeweiligen Wunden präsentiert. In der Tabelle 7.2 werden die ermittelten Messergebnisse der Wundbreiten und in Tabelle 7.3 die Ergebnisse der Wundlängen.

Nr.	Anzahl	GT (mm)	M (mm)	SD (mm)	Fehler (mm)	Fehler (%)
1	36	24.00	22.94	0.93	-1.06	-4.43
2	41	16.00	16.10	0.69	0.10	0.63
3	41	29.00	26.67	2.36	2.33	8.03
4	41	10.00	11.63	0.69	1.63	16.33
5	41	5.00	5.07	1.15	0.07	1.34
6	41	14.00	12.46	1.09	-1.54	-11.01
7	40	17.00	16.19	0.87	-0.81	-4.76
8	41	29.00	27.79	3.37	-1.21	-4.17

Tabelle 7.2: Messergebnisse: Breite

Die Spalte *Anzahl* zeigt die Anzahl der Mess-Datensätze. Bei Wunde 1 und Wunde 7 gib es einige Fotos, die nicht in die Statistik einfließen konnten, da auf diesen die Wunden nicht vollständig abgebildet waren. *GT* steht hier für den *Ground Truth*, also die manuell gemessene tatsächliche Größe der Wunden. Der Fehler in mm wird aus der Differenz des ermittelten Mittelwertes und dem Ground Truth berechnet. Der

Nr.	Anzahl	GT (mm)	M (mm)	SD (mm)	Fehler (mm)	Fehler (%)
1	36	98.00	86.98	5.80	-11.02	-11.24
2	41	68.00	58.91	5.79	-9.09	-13.37
3	41	145.00	122.31	10.29	22.69	15.65
4	41	40.00	37.68	1.99	-2.32	-5.79
5	41	28.00	26.58	1.07	-1.42	-5.06
6	41	140.00	127.52	13.9	-12.48	-8.91
7	40	96.00	87.26	4.82	-8.74	-9.10
8	41	55.00	52.83	10.95	-2.17	-3.95

Tabelle 7.3: Messergebnisse: Länge

Fehler in Prozent zeigt den prozentualen Anteil des Fehlers am Ground Truth. In der folgenden Abb. 7.1 wird je Wunde in Leserichtung ein Beispielfoto mit Segmentierungsmaske dargestellt:

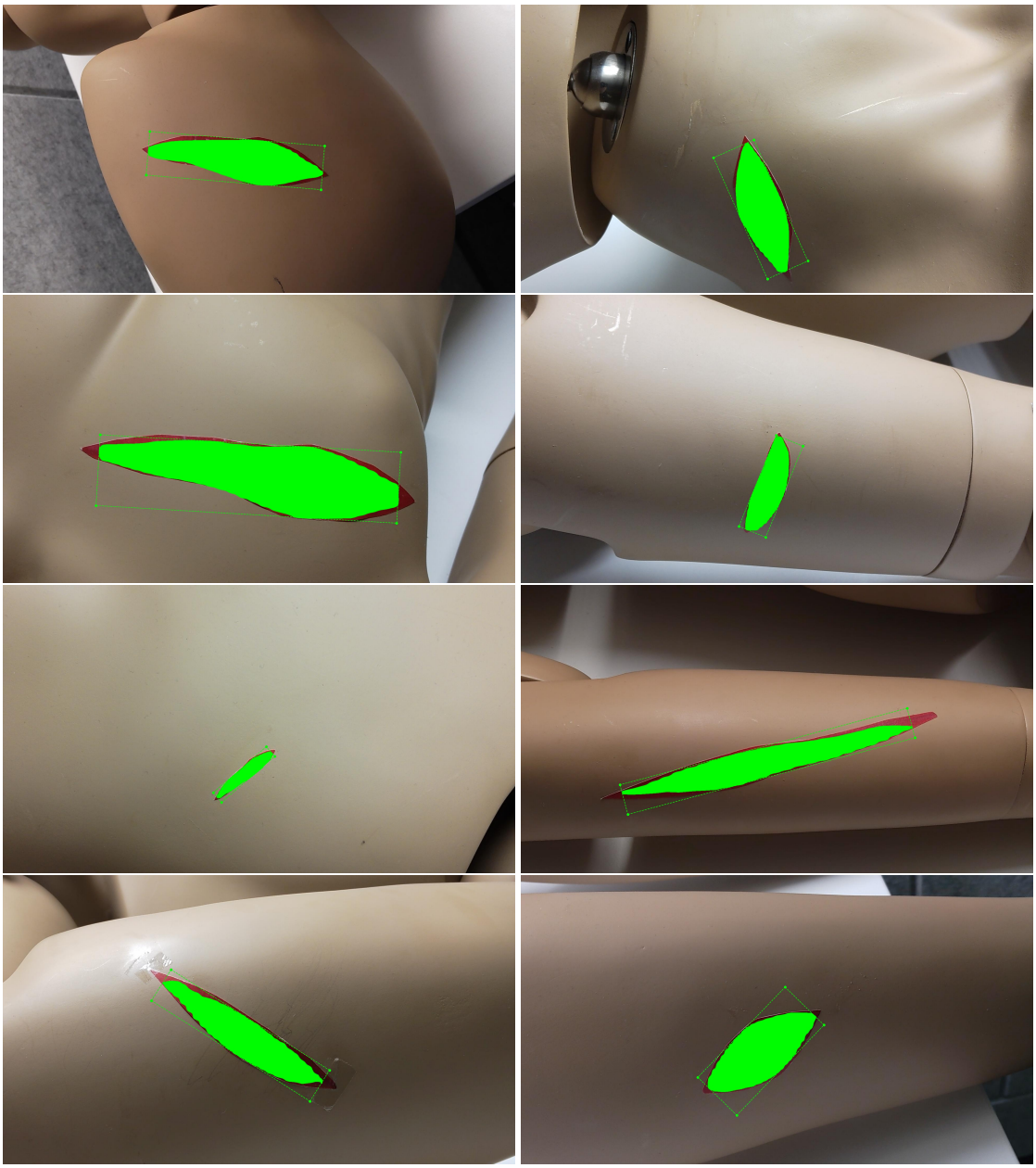


Abbildung 7.1: Beispielbilder der acht Wunden mit segmentiertem Bereich

8 Diskussion

In diesem Kapitel werden die im vorigen Kapitel erhobenen Ergebnisse analysiert und interpretiert. Es soll überprüft werden, ob die umgesetzte AR-Anwendung mit dem Wundmessungsverfahren zum einen als nutzbar eingestuft werden kann und ob die Messergebnisse der empirischen Untersuchung eine mögliche Praxistauglichkeit bestätigen können. Das Ziel dieser Diskussion ist es außerdem, Einschränkungen und Verbesserungsmöglichkeiten des Prototyps aufzuzeigen und mögliche Schlussfolgerungen für weitere Arbeiten in diesem Feld zu ziehen.

Zunächst werden die Ergebnisse des SUS näher betrachtet. 41 Teilnehmende haben im Fragebogen die Nutzbarkeit der Anwendung bewertet, welches einen gesamten SUS-Score von 80,79 von möglichen 100 ergeben hat. Dies kann mit dem Adjektiv *exzellent* bewertet werden, da der Score es im oberen vierten Quartil des SUS angesiedelt ist. Durch Analyse der Anmerkungen im gleichnamigen, optionalen Freitextfeldes im Fragebogen können weitere Schlüsse zur Nutzbarkeit gezogen werden. Auch wenn die Anwendung als allgemein sehr nutzbar eingestuft werden kann, wurden einige Schwächen der Anwendung aufgezeigt, die jedoch durch weitere Arbeit am Prototyp behoben werden können. Die Nutzung der Fotofunktion der Anwendung wurde als umständlich beschrieben, da es dem Nutzenden nicht klar ist, wie der Bildausschnitt des Fotos aussieht. Einige Teilnehmende der Untersuchung konnten ebenfalls Erfahrung mit der Wundmessung mitbringen und haben angegeben, dass die Nutzung eines solchen Wundmessungsverfahrens die Wunddokumentation erheblich erleichtern und beschleunigen kann. Nur einige wenige gaben an, dass die Anwendung zu Beschwerden wie Motion Sickness geführt hat. Die HoloLens 2 weist große Fortschritte in der Bequemlichkeit der Nutzung im Gegensatz zur vorigen Generation auf. Zukünfte AR-Hardware können in der Lage sein, die letzten für den Menschen als unangenehm empfundene Beschwerden, die aus der Nutzung solcher Systemen resultieren, zu beseitigen. Es ist beispielsweise abzuwarten, welche Performance die für 2024 angekündigte *Apple Vision Pro*, einem AR-System mit ähnlichen Features wie der HoloLens 2, liefern wird und wieweit sie den Markt verändern wird.

Was das Wundmessungsverfahren betrifft, sind eher schlechtere Ergebnisse für die Genauigkeit der Größenbestimmung entstanden. Bei der Wunde 5 (siehe Abbildung

8.1) wurde ein gemittelter Messfehler in der Breite von 0,07 mm und in der Länge von 1,07 mm mit einer relativ geringen Standardabweichung von 1,15 erzielt. Generell ist die Messfehler bei der Bestimmung der Wundbreite wesentlich geringer als bei der Wundlänge. Wie in den Ergebnissen zu sehen, weisen die absoluten Fehler in mm die geringsten Werte auf. Bei der Wundlänge allerdings ist der Fehler größer, da das trainierte Mask-R-CNN-Modell die Maskierung nicht immer vollständig über die gesamte Wundlänge legt (siehe Abbildung 8.2). Grund hierfür kann die Form der falschen Wunden in der Untersuchung sein. Die meisten der Wunden auf der Schaufensterpuppe hatten spitze Ecken an den Enden der Wunden und klare Abgrenzungen. Die Bilder, die im Training des Netzwerks eingesetzt wurden, zeigen eher Wunden mit unklaren Rändern und runden Enden. Zudem unterscheidet sich die Beschaffenheit der „Haut“ der Schaufensterpuppe erheblich von denen aus den realen Wundbildern.

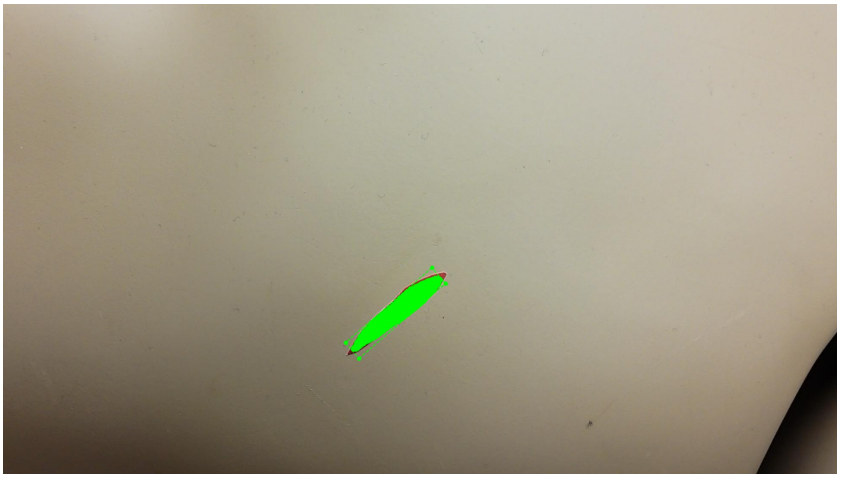


Abbildung 8.1: Wunde 5

Da es keine genauen Standards und Vorgaben über eine praxistaugliche Fehlertoleranz bei der Größenbestimmung existiert, ist es schwierig, die Messgenauigkeit in den Ergebnissen als eher gut oder schlecht einzuordnen. Logischerweise ist stets eine möglichst niedrige Fehlertoleranz anzustreben. Um die Ergebnisse dieser Arbeit auf quantisierte zu analysieren, kann als maximale Fehlertoleranz 5 % festgelegt werden. Demnach ist der Messfehler der Untersuchung noch zu hoch, um das Wundmessungsverfahren als praxistauglich einstufen zu können. Nach diesem Kriterium schneidet nur die Wunde 5 gut ab. Weil das umgesetzte Wundmessungsverfahren nicht mit

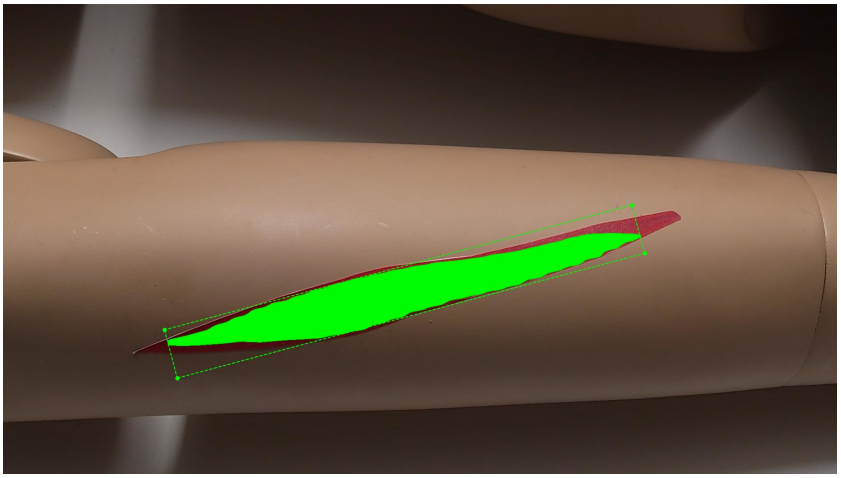


Abbildung 8.2: Wunde 6

echten Wunden evaluiert wurde, wäre eine Untersuchung dieses Verfahrens in einem echten klinischen Szenario ratsam. Zugang zu größeren Bilddatenbanken könnte die Segmentierungsleistung des Mask R-CNN erheblich verbessern. Ein für weitere Forschung geeigneter Aspekt der Wundmessung ist außerdem die Entwicklung eines Verfahrens zur Ermittlung der Wundtiefe, die mit klassischen 2D-Bildanalysetools bisher nicht gemessen werden konnte.

9 Fazit

Die vorliegende Masterarbeit beschäftigte sich mit der Entwicklung und Evaluation eines praxistauglichen Wundmessungsverfahrens auf der HoloLens 2 mittels Convolutional Neural Networks. Ziel war es, eine innovative Lösung zu entwickeln, die traditionelle Methoden der Wundmessung übertreffen und die Effizienz der Wunddokumentation in der medizinischen Praxis steigern soll. Außerdem sollte diese Lösung empirisch evaluiert werden. Zunächst wurden in dieser Arbeit die grundlegenden Konzepte und Technologien besprochen, die in dieser Arbeit von Relevanz waren. Im darauffolgenden Kapitel „Stand der Technik“ wurde ein Überblick über aktuelle wissenschaftliche Arbeiten geschaffen, die die Notwendigkeit eines automatisierten Wundmessungsverfahrens betont und entsprechende Lösungen entwickelt haben. Im Anschluss wurden die Anforderungen an das Projekt vorgestellt, die von der Tiplu GmbH, die diese Arbeit in Auftrag gegeben hat, gestellt wurden. Die Entwicklung des Wundmessungsverfahrens auf der HoloLens 2 wurde im entsprechenden Kapitel „Implementierung“ detailliert präsentiert. Anschließend wurde das Versuchsdesign im Kapitel „Evaluation“ und die Ergebnisse im gleichnamigen Kapitel vorgestellt. Zum Schluss wurden diese Ergebnisse diskutiert und eingeordnet, um die Nutzbarkeit der umgesetzten AR-Anwendung in Verbindung mit dem Wundmessungsverfahren zu bewerten und die Praxistauglichkeit des Messungsverfahrens zu überprüfen. Die Anwendung erhielt insgesamt eine positive Bewertung, wobei der SUS-Score mit 80,79 als „exzellent“ eingestuft wurde. Es wurden jedoch einige Schwächen identifiziert, die durch weitere Arbeit am Prototyp behoben werden könnten, wie die Verbesserung der Fotofunktion. Bezüglich des Wundmessungsverfahrens wurden jedoch schlechtere Ergebnisse für die Genauigkeit der Größenbestimmung festgestellt. Insbesondere die Wundlängenmessung wies einen größeren Fehler auf, möglicherweise aufgrund von der Form der unechten Wunden in der Untersuchung. Der Messfehler sollte möglichst niedrig sein, um das Verfahren als praxistauglich einzustufen. Eine Evaluation des Verfahrens in einem echten klinischen Szenario mit echten Patienten und Wunden wäre wünschenswert und der Zugang zu größeren Bilddatenbanken könnte die Segmentierungsleistung verbessern. Insgesamt kann jedoch das Ergebnis dieser Arbeit als Grundlage für die Entwicklung einer in der Praxis einsetzbaren Anwendung betrachtet werden.

A Fragebogen

Allgemeine Fragen

1. Ihre Teilnahme-ID

   _____

2. Wie alt sind Sie?

   o 17 oder jünger

   o 18 - 20

   o 21 - 29

   o 30 - 39

   o 40 - 49

   o 50 - 59

   o 60 oder älter

3. Bitte geben Sie Ihr Geschlecht an.

   o weiblich

   o männlich

   o diver

4. Haben Sie bereits Erfahrung im Umgang mit der HoloLens 2 oder anderen AR-Systemen?

   o ja

   o nein

5. Haben Sie bereits berufliche Erfahrung mit der Messung von Wunden?

   o ja

   o nein

6. Haben Sie Sehprobleme, die die Nutzung der HoloLens 2 beeinträchtigen haben könnten?

   o ja

   o nein

7. Hatten Sie während der Nutzung der HoloLens 2 körperliche Beschwerden wie Kopfschmerzen oder Übelkeit erfahren?

   o ja

   o nein

8. Wenn ja, welche?

   _____

System Usabilty Score

1. Ich denke, ich würde die Anwendung regelmäßig nutzen.

   Stimme überhaupt nicht zu o o o o o *Stimme voll und ganz zu*

2. Die Anwendung erscheint mit unnötig kompliziert.

   Stimme überhaupt nicht zu o o o o o *Stimme voll und ganz zu*

3. Ich finde, die Anwendung ist einfach zu benutzen.

   Stimme überhaupt nicht zu o o o o o *Stimme voll und ganz zu*

4. Ich denke, ich bräuchte technische Unterstützung, um die Anwendung nutzen zu können.

   Stimme überhaupt nicht zu o o o o o *Stimme voll und ganz zu*

5. Ich finde, dass die verschiedenen Funktionen der Anwendung gut integriert sind.

   Stimme überhaupt nicht zu o o o o o *Stimme voll und ganz zu*

6. Die Anwendung erscheint mir zu uneinheitlich.

   Stimme überhaupt nicht zu o o o o o *Stimme voll und ganz zu*

7. Ich glaube, dass die meisten Leute die Benutzung der Anwendung schnell erlernen können.

   Stimme überhaupt nicht zu o o o o o *Stimme voll und ganz zu*

8. Die Anwendung erscheint mir sehr umständlich zu benutzen.

   Stimme überhaupt nicht zu o o o o o *Stimme voll und ganz zu*

9. Ich fühle mich bei der Benutzung der Anwendung sehr sicher.

   Stimme überhaupt nicht zu o o o o o *Stimme voll und ganz zu*

10. Ich musste einiges lernen, um mit der Anwendung zurecht zu kommen.

    Stimme überhaupt nicht zu o o o o o *Stimme voll und ganz zu*

Anmerkungen

Haben Sie noch weitere Anmerkungen?

Literatur

Balzert, H. (2009). *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. Spektrum Akademischer Verlag. https://doi.org/10.1007/978-3-8274-2247-7

Bohn, D. (2019). Microsoft's HoloLens 2: a $3,500 mixed reality headset for the factory, not the living room. https://www.theverge.com/2019/2/24/18235460/microsoft-hololens-2-price-specs-mixed-reality-ar-vr-business-work-features-mwc-2019

Brooke, J. (1995). SUS: A quick and dirty usability scale. *Usability Eval. Ind.*, *189*. https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale

Çamönü, İ. (2019). Coordinate Spaces and Transformations Between Them - codinBlack. https://www.codinblack.com/coordinate-spaces-and-transformations-between-them/

Carrión, H., Jafari, M., Bagood, M. D., Yang, H.-y., Isseroff, R. R., & Gomez, M. (2022). Automatic wound detection and size estimation using deep learning algorithms (Z. Yu, Hrsg.). *PLOS Computational Biology*, *18*(3), e1009852. https://doi.org/10.1371/journal.pcbi.1009852

Cooley, S. (2023). Hololens 2 Hardware. https://learn.microsoft.com/en-us/hololens/hololens2-hardware

Fulton, W. (2023). Calculator to compute the Distance or Size of an Object in a photo Image. https://www.scantips.com/lights/subjectdistance.html

Furht, B. (Hrsg.). (2011). *Handbook of Augmented Reality*. Springer New York. https://doi.org/10.1007/978-1-4614-0064-6

Girshick, R. B., Donahue, J., Darrell, T., & Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, *abs/1311.2524*. https://doi.org/10.48550/arXiv.1311.2524

Gurucharan, M. (2022). Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network | upGrad blog — upgrad.com. https://www.upgrad.com/blog/basic-cnn-architecture/

Hartmann. (2023). Wunddokumentation - Klassifikationen und Anleitungen. https://www.hartmann.info/de-de/loesungen/l/de/medien/broschueren-downloads/wunddokumentation-hilfe

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. https://doi.org/10.48550/ARXIV.1703.06870

Hunt, A., & McGlashan, S. (2004). Speech recognition grammar specification version 1.0. https://www.w3.org/TR/speech-grammar/

Jung, A. (2020). Examples: Basics x2014; imgaug 0.4.0 documentation — imgaug.readthedocs.io. https://imgaug.readthedocs.io/en/latest/source/examples_basics.html

Klinker, K., Wiesche, M., & Krcmar, H. (2019). Digital Transformation in Health Care: Augmented Reality for Hands-Free Service Innovation. *Information Systems Frontiers*, *22*(6), 1419–1431. https://doi.org/10.1007/s10796-019-09937-7

Matterport. (2017). GitHub - matterport/Mask RCNN. https://github.com/matterport/Mask_RCNN

Mattzmsft. (2023). Spatial Mapping - Mixed Reality. https://learn.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping

Microsoft. (2022a). GitHub - microsoft/MixedReality-QRCode-Sample: A single repository of Mixed Reality samples in Unity. — github.com. https://github.com/microsoft/MixedReality-QRCode-Sample

Microsoft. (2022b). Microsoft/MixedRealityToolkit-Unity. https://github.com/microsoft/MixedRealityToolkit-Unity

Microsoft. (2022c). Speech - MRTK 2. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/speech

Milgram, P., Takemura, H., Utsumi, A., & Kishino, F. (1994). Augmented reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies*, *2351*. https://doi.org/10.1117/12.197321

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. https://doi.org/10.48550/arXiv.1511.08458

Pallets. (2023). Welcome to Flask x2014; Flask Documentation (2.3.x). https://flask.palletsprojects.com/en/2.3.x/

Partsch, H. A. (2010). *Requirements-Engineering systematisch*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-05358-0

Protz, K. (2023). Wunddokumentation | Mölnlycke — molnlycke.de. https://www.molnlycke.de/unsere-expertise/expertenwissen-moderne-wundversorgung/wunddokumentation/

Ren, S., He, K., Girshick, R. B., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR*, *abs/1506.01497*. https://doi.org/10.48550/arXiv.1506.01497

Semple, K. (2023). MRTK2-Unity Developer Documentation - MRTK 2. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/

Tiplu. (2023). Intelligente Entscheidungen für eine bessere Gesundheit. https://tiplu.de/

Unity-Technologies. (2023a). Plans and pricing. https://unity.com/pricing

Unity-Technologies. (2023b). Unity - Manual: Cameras. https://docs.unity3d.com/Manual/CamerasOverview.html

Unity-Technologies. (2023c). Unity - Manual: Creating and Using Scripts. https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

Unity-Technologies. (2023d). Unity - Manual: Important Classes - GameObject. https://docs.unity3d.com/Manual/class-GameObject.html

Unity-Technologies. (2023e). Unity - Manual: Prefabs. https://docs.unity3d.com/Manual/Prefabs.html

Unity-Technologies. (2023f). Unity - Manual: Scenes. https://docs.unity3d.com/Manual/CreatingScenes.html

Unity-Technologies. (2023g). Unity - Manual: Transforms. https://docs.unity3d.com/Manual/class-Transform.html

Wang, C., Anisuzzaman, D. M., Williamson, V., Dhar, M. K., Rostami, B., Niezgoda, J., Gopalakrishnan, S., & Yu, Z. (2020). Fully automatic wound segmentation with deep convolutional neural networks. *Scientific Reports*, *10*(1), 21897. https://doi.org/10.1038/s41598-020-78799-w

Wen, Q. (2022). QR code tracking overview - Mixed Reality. https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/qr-code-tracking-overview

Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D. (2016). Understanding data augmentation for classification: when to warp? https://doi.org/10.48550/arXiv.1609.08764

Hinweis: Alle hier aufgelisteten Web-Links wurden zuletzt am 31. Juli 2023 überprüft!

Anhang USB-Stick

Auf dem beiliegenden USB-Stick befinden sich folgende Verzeichnisse und Dateien:

Verzeichnis/Datei	**Beschreibung**
wagner_edgar_masterarbeit.pdf	PDF-Kopie der Masterarbeit
tiplu_patient_hololens.unitypackage	Unity-Package-Datei der Anwendung mit allen Assets
/MASK_RCNN	Mask-R-CNN-GitHub-Projekt
/wound_webserver	flask-Webserver mit allen in der Untersuchung erstellten Fotos
/measurements	Verzeichnis mit den Messwerten der Untersuchung, mit sämtlichen Bildern