Universität Duisburg-Essen
Fakultät für Wirtschaftswissenschaften
Arbeitsgruppe Mensch-Computer Interaktion

Masterarbeit

Kollaboratives Schreiben mit Unterstützung von Optical Character Recognition

Vorgelegt der Fakultät Wirtschaftswissenschaften der Universität Duisburg-Essen (Campus Essen) von

Jan-Hendrik Terbrack

Scharnhorststraße 2, 47229 Duisburg

Studiengang:	Master Lehramt Informatik GyGe
Matrikelnummer:	3055169
Erstgutachter/in:	Prof. Dr. Stefan Schneegass
Zweitgutachter/in:	Prof. Dr. Torsten Brinda
Betreuer/in:	Nick Wittig, M.Sc. Noro Schlorke, M.Sc.
Datum:	10. April 2023

Kurzfassung

Die Digitalisierung schreitet voran und auch das kollaborative Schreiben findet immer häufiger über Online-Anwendungen wie Wikis, Blogs oder Cloud Anwendungen wie Google Docs statt. Dennoch fehlt es an digitalen Lernsystemen für SchülerInnen, die diese individuell unterstützen. Die individuelle Förderung ist besonders wichtig, da beispielsweise das Vorwissen für den Lernerfolg eine große Rolle spielt und dieses von SchülerIn zu SchülerIn unterschiedlich ist. Eine Lehrkraft kann zwar versuchen, die SchülerInnen individuell zu unterstützen, jedoch gestaltet sich dies schwierig durch die Anzahl der SchülerInnen mit jeweils unterschiedlichem Vorwissen. Ein solches Lernsystem könnte Lernmaterial zur Verfügung stellen und je nach SchülerIn zusätzliche Informationen zur Verfügung stellen. Geschehen sollte dies in einer erweiterten Realität.

In dieser Masterarbeit wurde dafür ein erstes grundlegendes Softwaresystem geschaffen, das kollaboratives Schreiben mit Stift und Papier mit dem kollaborativen Schreiben über Cloud-Anwendungen verbindet. Neu dabei ist, dass durch das implementierte Softwaresystem mit Unterstützung einer (Hand)-Schrifterkennung, im Englischen Optical Character Recognition genannt, arbeitet. Ohne dieses Lernsystem müssten beim kollaborativen Schreiben entweder Fotos des Geschriebenen ausgetauscht werden, oder das Schreiben müsste durch eine digitale Eingabe, wie über ein Tablett, Smartphone oder den Computer erfolgen. Das hier geplante und implementierte Softwaresystem ermöglicht es, dass die SchülerInnen ihren handschriftlich geschriebenen Text innerhalb der implementierten App fotografieren und an den Server übermitteln. Der fotografierte Text wird mithilfe der (Hand-)Schrifterkennung in digitalen Text umgewandelt und den beiden SchreibpartnerInnen angezeigt.

Ein großes Problem ist hierbei die Genauigkeit der Handschrifterkennung. Dieser liegt eine durch Deep Learning trainierte künstliche Intelligenz zugrunde, die anhand ihres Trainings in der Lage ist, einzelne Buchstaben und Wörter zu erkennen. Handschriften unterscheiden sich häufig stark voneinander, wodurch die vorausgesagten Wörter oftmals andere sind, als handschriftlich verfasst.

Daher wurde in diesem System im späteren Verlauf von einer selbst programmierten, lauffähigen Version zu einer open source (Hand-)Schrifterkennung (EasyOCR) gewechselt, da diese besser den fotografierten Text voraussagt und zudem weitere nützliche Implementierungen enthält. Außerdem ist für die nahe Zukunft für (Hand-)Schrifterkennung EasyOCR ein Update geplant, das vorausgesagte Wörter bei handschriftlich geschriebenem Text deutlich verbessern soll.

In dem hier entwickelten Softwaresystem wurden Funktionen implementiert, die bei ungenauer Texterkennung dem Nutzer diese Stellen mitteilen und eine digitale Eingabe für diese Stelle ermöglichen. Die Handschrifterkennung birgt ein noch großes Problem bei der individuellen Förderung der SchülerInnen, ist aber bei einem Lernsystem, das auf einer Eingabe mit Stift auf Papier basiert, essenziell, besonders wenn in Echtzeit zusätzliche Informationen dem Nutzer in einer erweiterten Realität angezeigt werden.

Abstract

Digitization is advancing and collaborative writing is also increasingly taking place via online applications such as wikis, blogs, or cloud applications like Google Docs. Yet, there is a lack of digital learning systems for students to support them individually. Individual support is particularly important because, for example, prior knowledge plays a major role in learning success and this varies from student to student. A teacher can try to support the students individually, but this is difficult due to the number of students with different prior knowledge. Such a learning system could provide learning material and provide additional information depending on the student. This should be done in an augmented reality.

In this master thesis a first basic software system was created for this purpose, which combines collaborative writing with pen and paper with collaborative writing via cloud applications. The novelty is that the implemented software system works with the support of a handwriting recognition, called Optical Character Recognition. Without this learning system, collaborative writing would either require sharing photos of what is written, or writing would have to be done through digital input, such as via a tablet, smartphone, or computer. The software system planned and implemented here allows students to photograph their handwritten text within the implemented app and submit it to the server. The photographed text is converted into digital text using the handwriting recognition and displayed to the two writing partners.

A major problem here is the accuracy of handwriting recognition. This is based on artificial intelligence trained by Deep Learning, which is able to recognize individual letters and words on the basis of its training. Handwriting often differs greatly from one another, which means that the predicted words are often different from those written by hand.

Therefore, in this system, we later switched from a self-programmed, executable version to an open source handwriting recognition (EasyOCR), as it better predicts photographed text and also contains other useful implementations. In addition, an update is planned for the near future for (hand) character recognition EasyOCR, which should significantly improve predicted words for handwritten text.

In the software system developed here, functions have been implemented that, in the case of inaccurate text recognition, inform the user of these locations and allow digital input for that location. Handwriting recognition still poses a major problem in individual student support, but is essential in a learning system based on pen-on-paper input, especially when additional information is displayed in real time to the user in an augmented reality.

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Listings

1 Einleitung

Dieses Kapitel umfasst die Motivation und die Gliederung der Arbeit. In der Motivation wird die Problemstellung, die Zielsetzung, die Forschungsfrage und der mögliche Nutzen der Masterarbeit aufgegriffen.

1.1 Motivation

SchülerInnen stehen vor immer größeren Herausforderungen, wobei die Kluft zwischen leistungsstarken und leistungsschwachen SchülerInnen immer größer wird. Die Digitalisierung der Welt schreitet voran und das Lernen findet nicht mehr nur im Klassenzimmer statt. Die Kultusministerkonferenz erwähnt dabei explizit digitalisierungsbezogene und informatische Kompetenzen, die für SchülerInnen in der heutigen Welt eine große Rolle spielen. Dafür ist der Einsatz von digitalen Lernsystemen erforderlich [25]. Das Bundesministerium für Bildung und Forschung hat ein Verbundprojekt namens FederLeicht [15] gestartet, das den Anforderungen der Kultusministerkonferenz gerecht werden könnte. Dieses Verbundprojekt beschäftigt sich mit der Erforschung und Entwicklung eines Lernsystems in einer erweiterten Realität. Ziel ist es, SchülerInnen individuell beim Lernprozess zu unterstützen, indem zusätzliche Informationen in einer erweiterten Realität zur Verfügung gestellt werden. Für eine erfolgreiche Erforschung und Entwicklung eines Lernsystems in einer erweiterten Realität braucht es Vorarbeit, die verschiedene Aspekte und deren Effekte auf das Lernen untersucht.

Ziel dieser Masterarbeit ist es, Vorarbeit für das Projekt FederLeicht zu leisten. Dabei soll der Aspekt des kollaborativen Schreibens mithilfe einer Handschrifterkennung ermöglicht werden. Dass sich traditionelles kollaboratives Schreiben positiv auf das Lernen von SchülerInnen auswirkt, wurde in der Vergangenheit durch verschiedene Arbeiten nachgewiesen. Doch es fehlt an Grundlagenforschung, ob sich diese positiven Effekte auch unter anderen Bedingungen, wie in etwa einer erweiterten Realität, belegen lassen.

Diese Arbeit orientiert sich an der Forschungsfrage „Erhöht kollaboratives Schreiben in einer erweiterten Realität die Motivation und die Schreibqualität?“. Dafür soll in dieser Arbeit ein Softwaresystem geplant und implementiert werden, dass kollaboratives Schreiben mit Stift auf Papier ermöglicht, obwohl die NutzerInnen räumlich voneinander getrennt sind.

Sollten sich positive Effekte, wie eine erhöhte Motivation und/oder eine höhere Schreibqualität aufzeigen, sollte in Betracht gezogen werden, im Lernsystem des Projekts FederLeicht kollaboratives Schreiben zu ermöglichen. Außerdem wird dem Lernsystem, genau

wie dem Softwaresystem in dieser Masterarbeit, eine Handschrifterkennung zugrunde liegen. Daher lassen sich nach dieser Masterarbeit erste Rückschlüsse ziehen, was eine solche Handschrifterkennung leisten muss und welche möglichen Probleme auftreten können.

1.2 Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Hintergrund und verwandte Arbeiten: Hier werden die Grundlagen dieser Arbeit beschrieben. Dafür wird im theoretischen Hintergrund grundlegendes Wissen über drei Kernbereiche vermittelt, verwandte Arbeiten und deren Erkenntnisse genannt und anschließend das Thema der Arbeit konkretisiert.

Kapitel 3 – Konzept: Beinhaltet das Konzept, aufgeteilt auf die Idee hinter der Implementierung und das Systemdesign. Konkret wird in diesem Kapitel das Softwaresystem geplant und Rückschlüsse für die Implementierung gezogen.

Kapitel 4 – Implementierung: Befasst sich mit der Implementierung des Softwaresystems. Unterteilt wird dies in die Komponenten Server, App und die Handschrifterkennung. Für jede Komponente wird der Nutzen für das System, teilweise unterstützt durch Abbildungen und Quellcode, erläutert.

Kapitel 5 – Evaluation: Thematisiert den Aspekt der Evaluation.

Kapitel 6 – Diskussion: In diesem Kapitel werden verschiedene Aspekte aus vorherigen Kapiteln diskutiert, besonders unter dem Aspekt der Limitierung. Zudem erfolgt ein Ausblick.

Kapitel 7 – Zusammenfassung: Hier wird ein kurzer Durchgang durch die gesamte Arbeit gegeben.

2 Hintergrund und verwandte Arbeiten

Hier werden die Grundlagen dieser Arbeit beschrieben. Dafür wird im theoretischen Hintergrund grundlegendes Wissen über drei Kernbereiche vermittelt, verwandte Arbeiten und deren Erkenntnisse genannt und anschließend das Thema der Arbeit konkretisiert.

2.1 Theoretischer Hintergrund

Für den theoretischen Hintergrund werden Kernbereiche, die der Arbeit zugrunde liegen, erläutert.

2.1.1 Optical Character Recognition

Eine Optical Character Recognition, kurz OCR, ist zu deutsch eine optische Zeichenerkennung. Diese Zeichenerkennungen beruhen auf Modellen, die durch maschinelles Lernen entstanden sind. Das bedeutet, dass das Modell anhand von Trainingsdaten trainiert wurde, dabei Muster zu erkennen. Zusätzlich gibt es noch Daten zum Validieren und zum Testen. Ziel ist es dabei, dass Text in einem Bild anhand der Mustererkennung erkannt, interpretiert und identifiziert wird. Dieser Text wird noch in maschinenlesbare Sprache dekodiert. Besonders anspruchsvoll ist eine optische Zeichenerkennung, die handschriftlich verfasste Texte erkennen und schließlich dekodieren kann. Dies liegt daran, dass Handschriften stark variieren und Buchstaben teilweise verbunden werden [8, 37, 43]. Klassifizierungsfehler könnten verringert werden, wenn die trainierte OCR Kontextinformationen nutzen könnte, was sich bei einer Handschrifterkennung als besonders schwierig herausstellt [2]. In diesem Artikel wird geschlussfolgert, dass zu dem Zeitpunkt (2002) die OCR nur in begrenzten Einsatzbereichen einsetzbar ist. Dazu zählt beispielsweise das Erkennen und Lesen von Postadressen. Heutzutage ist die Entwicklung fortgeschritten und verschiedene maschinelle Lernverfahren wurden und werden kombiniert, um die Genauigkeit des OCRs zu erhöhen. Durch den Einsatz von Deep Learning als Unterkategorie des maschinellen Lernens wird ein künstliches neuronales Netz aufgebaut. Dieses künstliche neuronale Netz ist dem neuronalen Netz aus der Biologie nachempfunden. Den Verarbeitungsprozess übernehmen sogenannte Neuronen. Diese klassifizieren die eingegebenen Daten anhand vordefinierter Klassen. Besonders vielversprechend sind zum Zeitpunkt der systematischen Literaturrecherche sogenannte Convolutional Neural Networks, kurz CNNs, die mit mehreren Schichten zur Mustererkennung arbeiten. Mehrere Schichten meint, dass nicht nur auf einer quadratischen Fläche die Verarbeitung der Eingabedaten geschieht, sondern auf mehreren Ebenen, beziehungsweise Schichten, mit teils anderen vordefinierten Klassen und Bezeichnungen. Dies wird Multi-Layer Perceptron, kurz MLP genannt.

Dadurch soll die Fehlerquote sinken [37]. Die Dekodierung unterscheidet sich je nach gewählten Schichten. Eine bekannte und erprobte ist die Connectionist Temporal Classification, kurz CTC Dekodierung, die anhand der jeweils höchsten Wahrscheinlichkeit für jeden Buchstaben das Wort / den Text zusammensetzt [47]. Für das Training wird auf teils große Datenbanken mit verschiedenen Handschriften zurückgegriffen, damit die Handschrifterkennung eine große Anzahl an Handschriften erkennt. Teilweise werden Handschrifterkennung in Studien anhand der Handschriften der StudienteilnehmerInnen trainiert, um eine noch höhere Genauigkeit zu erreichen [37]. Die Eingabedaten, in diesem Fall Bilder oder Videos, können vor dem Training und der Zeichenerkennung verändert werden, beispielsweise durch das Umwandeln der Farben in Graustufen. Dabei sollte beachtet werden, dass die Eingabedaten des Trainings und der Zeichenerkennung ähnliche Eigenschaften aufweisen, wie beispielsweise das Format und die Farbgebung. Dafür werden Bildbearbeitungsalgorithmen verwendet [30].

2.1.2 Kollaboratives Schreiben

Die Bedeutung des Wortes kollaborativ wird im Duden als gemeinsam, zusammen arbeitend, entwickelnd beschrieben [14]. Kollaboratives Lernen ist eine Lehr-Lernform eines kooperativen Unterrichts, beziehungsweise eine Unterart des kooperativen Unterrichts [7]. Im Englischen wird dies nicht direkt differenziert [31].Nach der Definition eines Fachlexikons [11], die auf der Definition der Leuphana Universität Lüneburg [33] aufbaut, ist eine Differenzierung der Begriffe möglich, aber dieser wird nicht immer gefolgt. Eine Differenzierung erfolgt beispielsweise anhand des entstehenden Lernprodukts. Das kooperative Schreiben zielt auf eigene Lernprodukte hin, die durch arbeitsteiliges Arbeiten erstehen und durch Zusammenfügen ein „großes" Lernprodukt bilden. Beim kollaborativen Schreiben zielt es auf ein gemeinsam erarbeitetes Lernprodukt ab. Ein Beispiel, an dem die hier gebräuchliche Differenzierung klar wird, ist der Vergleich eines Glossars und einem online Wiki, wie Wikipedia. Für ein Glossar erarbeitet jeder einen eigenen Eintrag und am Schluss werden alle Einträge in das Glossar übernommen. Bei einem Wiki gibt es nicht nur eine/einen AutorIn, sondern auch zusätzliche Rollen, wie einen/einen LektorIn oder einen/eine ReviewerIn. Beide können die Arbeit nachträglich korrigieren, erweitern, verbessern oder auch kommentieren. Dies unterstützt nicht nur die Sozialkompetenz, sondern auch eine intensive und reflexive Auseinandersetzung mit dem Thema. Einige Definitionen für kollaboratives Lernen und Arbeiten gehen dabei noch mehr ins Detail. Lowry et al. beschreiben den Prozess des kollaborativen Arbeitens als reziprok, also wechselseitig, wobei hier klar definiert ist, dass sich die weitere Arbeit klar auf die des Vorgängers beziehen und konstruktiv sein muss. Als „Rollen" gibt es auch die des/der Ko-AutorIn. Dies verdeutlicht, dass nicht ein Lernprodukt entwickelt wird, das vor Abgabe überprüft und leicht verändert wird, sondern der gesamte Prozess, beispielsweise das Schreiben, wechselseitig geschehen kann. Zudem wird ein Vergleich zu einem Wissensaustausch per Dialog (kollaborative Interaktion) angestellt, der ebenfalls wechselseitig ablaufen kann und von dem beide vom Gegenüber profitieren [31]. Viele Softwareprodukte, wie beispielsweise Google Docs, unterstützen kollaboratives Schreiben. Dabei muss nicht wechselseitig gearbeitet werden, sondern ein gleichzeitiges Arbeiten ist möglich. Eine Kommentarfunktion unterstützt ebenso den Interaktionsprozess zwischen den Mitarbeitenden.

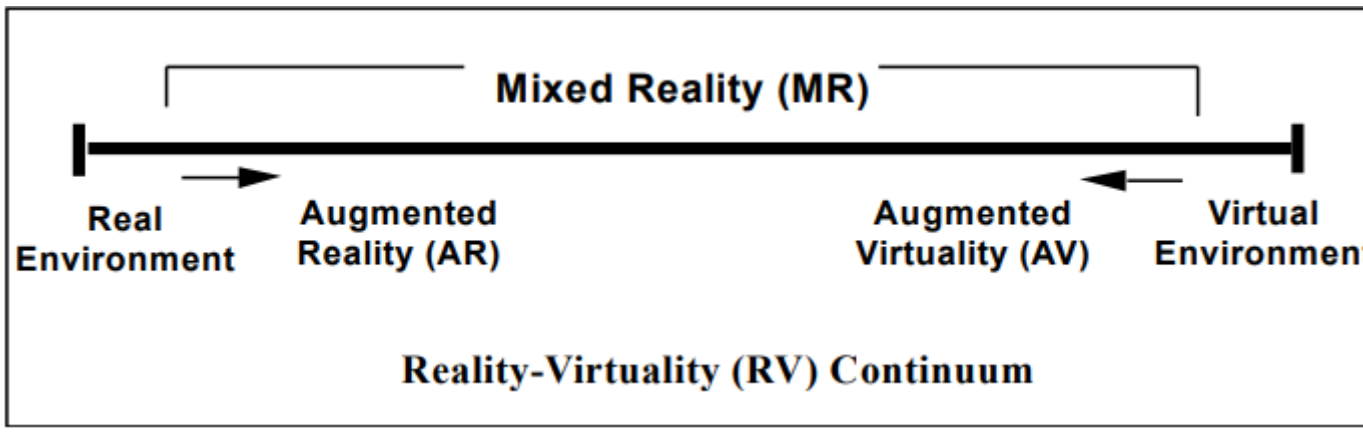


Abbildung 2.1: [Milgram: Reality-Virtuality Continuum aus [38]

2.1.3 Mixed-Reality

Nach Milgram ist die Mixed Reality, kurz MR, der Bereich zwischen unserer realen (Um-) Welt und einer virtuellen (Um-)Welt. Die Abbildung 2.1, genannt Reality-Virtuality Continuum, zeigt diesen Bereich auf. Augmented Reality, kurz AR, steht für eine erweiterte Realität, bei der die reale (Um-welt) um visuelle Objekte / Informationen ergänzt wird [38]. Nach Azuma müssen diese virtuellen Informationen / Objekte im dreidimensionalen Raum der Realität platziert sein und das in Echtzeit. Die gesamte Interaktion mit der AR muss in Echtzeit möglich sein [3]. Augmented Virtuality spiegelt das Konzept von AR. Eine virtuelle Welt wird um reale Objekte ergänzt. Virtual Reality, kurz VR ist eine virtuelle Welt, in die der/die NutzerIn eintaucht und interagiert. Desto mehr „eingetaucht" wird, desto näher ist VR an einer virtuellen Welt und nach der Abbildung2.1 immer weiter rechts [38]. AR ist ebenso nicht fest auf dem Kontinuum verordnet, da auch hier ein gewisses Eintauchen möglich ist. Dies hängt unter anderem mit den verschwendeten AR-Bildschirmen und Techniken zusammen. Die Arbeitsdefinition leitet sich anhand des Projektes FederLeicht ab [15]. Das Projekt wird durch das Bundesministerium für Bildung gefördert und lässt sich auf deren Webseite für interaktive Technologien finden[1]. Laut der Webseite für interaktive Technologien ist eine Mixed-Reality, kurz MR, eine gemischte Realität. Zu dieser gehören die erweiterte Realität (AR) und die virtuelle Realität (VR). Dies folgt der Bekanntmachung „Interaktive Systeme in virtuellen und realen Räume - Innovative Technologien für die digitale Gesellschaft" [16]. Genauere Definitionen, geschweige denn Abgrenzungen zwischen den Begriffen, erfolgt dabei nicht. Dennoch wird in der Bekanntmachung erwähnt, dass explizit Technologien gefördert werden sollen, die AR und VR um physische Interaktionsmöglichkeiten ergänzen. Im Projekt FederLeicht wird als Ziel der Aspekt des Darstellens von virtuellen Informationen in einer AR genannt. Dies ist nach Milgram [38] und Azuma [3] definitiv einer AR zuzuordnen.

[1] Interaktive Technologien https://www.interaktive-technologien.de/ abgerufen am 8.4.2023

2.2 Verwandte Arbeiten

In diesem Unterkapitel werden Arbeiten und Forschungserkenntnisse, die Schnittpunkte zu dieser Arbeit aufweisen, genannt.

2.2.1 Optical Character Recognition

Hinweis: Die Handschrifterkennung steht in dieser Masterarbeit nicht im Fokus. Daher werden zwei Quellen genannt, von denen aus eine weitere Recherche möglich ist.

Regelmäßig erscheinen zum Thema Handschrifterkennung neue Forschungsarbeiten. Einen guten Überblick verschafft die systematische Literaturrecherche von Memon et al. [37], in der 257 Studien rund um die Handschrifterkennung analysiert wurden. Typische Datensätze, die zum Training benutzt werden, grundlegende theoretische Prinzipien und auch Problematiken von Handschrifterkennung in realen Situationen werden angesprochen. Viele Handschrifterkennungen erzielen nur mit den typischen Datensätzen, die optimiert wurden für maschinelles Lernen, hohe Genauigkeiten bei der Zeichenerkennung. Daher sollten die prozentualen Genauigkeiten, die in Forschungsarbeiten genannt werden, kritisch betrachtet werden. Auf der internationalen Konferenz „International Conference on Document Analysis and Recognition", kurz ICDAR, werden im Zwei-Jahres-Rhythmus neueste Arbeiten zu OCRs, meist speziell zu Handschrifterkennungen vorgestellt. Auf dieser Konferenz werden nicht nur neue Forschungsarbeiten präsentiert, sondern auch Auszeichnungen vergeben[2]. Der Einsatz von OCRs spielt auch in AR-Systemen eine Rolle. Ein Beispiel ist Google Lens, das für Übersetzungen genutzt wird. Tatwany und Quertani fanden im Jahre 2016 gerade einmal 12 Forschungsarbeiten zu diesem Thema und entschieden sich, eine eigene Software zu entwickeln [54]. Die Möglichkeiten von OCR in AR-Systemen erscheinen vielfältig. Diese wird für die Hilfe bei der Navigation genutzt [42], oder auch um reale Informationen zu virtualisieren und speicherbar zu machen [52].

2.2.2 Kollaboratives Schreiben

Traditionelles kollaboratives Schreiben wurde in den vergangenen Jahrzehnten vielfach erforscht. Dabei wurde nicht nur das kollaborative Schreiben im schulischen Umfeld, sondern auch im universitären Kontext erforscht. Es wurden häufig eine höhere Schreibleistung in Bezug auf Qualität und Quantität nachgewiesen sowie eine Erweiterung des Wortschatzes beim Sprachenlernen und die Förderung der sozialen Kompetenzen, die als Folge der kollaborativen Interaktion gesehen wird [33]. In der Arbeit von McAllister [36] wird besonders die Erhöhung der Schreibleistung und Schreibmotivation festgestellt. Zudem verbleiben Studierende eher in Kursen, in denen kollaboratives Arbeiten und Schreiben ermöglicht wird. In der Forschungsarbeit von Shehadeh [48] werden besonders die Schreibqualität und dahingehend auch der Wortschatz und die Qualität des Inhalts des Geschriebenen betrachtet. Dabei wird die Schreibqualität operationalisiert über den Inhalt, die

[2] ICDAR 2023:https://icdar2023.org/ abgerufen am 8.4.2023

Organisation des Geschriebenen, die Grammatik, das Vokabular und die Mechaniken des Schreibens. Es wurden dabei durchgehend positive Auswirkungen ermittelt. Lehnen [26] bringt dazu noch Faktoren für das Gelingen von kollaborativen Arbeiten ins Spiel. Laut den dort ermittelten Erkenntnissen haben die bisher entwickelten sozialen Kompetenzen einen großen Einfluss auf das Gelingen und die auftretenden positiven Effekte. Eine zu geringe soziale Kompetenz im Bereich der Interaktion und des gemeinsamen Arbeitens kann zum Scheitern von kollaborativen Arbeiten führen. Ebenso, wenn die MitarbeiterInnen Schreibanfänger von wissenschaftlichen Arbeiten sind. Nach Lingnau et al. kann das Scheitern bei Schreibanfängern (hier werden allgemeine Schreibanfänger betrachtet) umgangen werden, wenn zusätzliche Software verwendet wird [29]. In dieser Arbeit geht es um eine solche Software, die neuartiges kollaboratives Schreiben ermöglicht. Während sich traditionelles kollaboratives Schreiben meist im Klassenraum von Angesicht zu Angesicht abspielte, wird dieses heute über Wikis oder Software wie Google Docs ermöglicht. Besonders im Bereich Sprachenlernen eignet sich dies, wobei zu beachten ist, dass Feedback, besonders direktes Feedback fehlt [51]. Schreiben wird als wichtiger Teil des Lernens gesehen, wobei besonders beim Schreiben in Gruppen dies zum Austausch von Wissen genutzt werden kann [32]. Software zum kollaborativen Schreiben wird nicht nur zum eigentlichen Schreiben genutzt, sondern es werden ebenso Materialien und Links hinzugefügt [40]. Bei Softwaresystemen, die kollaboratives Arbeiten ermöglichen sollen, sind zusätzliche Funktionen zum Austausch wie beispielsweise eine Chatfunktion sehr nützlich [31]. Das Schreiben wird durch kollaboratives Schreiben flüssiger [41]. Weitere Forschungsarbeiten beschäftigen sich auch mit dem Einfluss des kollaborativen Schreibens auf die Schreibqualität. In den Fokus rücken hier die Prozesse während des kollaborativen Arbeitens. Dies meint nicht direkt den Prozess, dass kurz vor Abgabe die einzelnen Schreibteile zunehmen, sondern die Schreibprozesse im Detail. Google Docs und auch Softwaresysteme wie WriteProc registrieren und speichern jede einzelne Änderung am Dokument eines jeden/jeder NutzerIn [49, 50]. Entscheidend ist ebenso das Gruppenbewusstsein, -beteiligung und die Koordination innerhalb der Gruppe [31]. Ebenso führt Quantität nicht zu Qualität, sondern eine ausgeglichene Teilnahme der TeilnehmerInnen führt zu einer erfolgreichen Arbeit [40, 58]. Southavilay et al. nennen als Prozesse / Aktivitäten des kollaborativen Schreibens das Brainstorming (Ideenfindung), Outlining (Gliederung), Drafting (Entwerfen), Revising (Überarbeiten) und Editing (Bearbeiten). Dies geschieht anhand verschiedener Textveränderungen, die auch als Schreibaktivitäten deklariert werden. Zudem werden auch Änderungen an der Textstruktur eingeordnet [49]. Ein großer Kritikpunkt an neuartigem kollaborativen Schreiben ist das fehlende direkte Feedback. Die Arbeiten von Grimes & Warschauer [18] und Khoshnevisan [23] beschäftigen sich mit einem direkten Feedback des Geschriebenen im Hinblick auf die Schreibqualität. Grimes & Warschauer listen mehrere Tools für automatisches Feedback aus, wobei die Arbeit von Khoshnevisan weitergeht. Hier wird das automatic writing evaluation, kurz AWE-, Tool Grammarly[3] untersucht. Dabei zeigte sich, dass die Motivation und die Schreibqualität der Lernenden messbar anstieg. Die Grammatik, der Wortschatz und die Zeichensetzung verbesserten sich. Dabei sind die Ergebnisse nicht repräsentativ, aber es wird dennoch empfohlen, dass Grammarly im Unterricht und beim traditionellen kollaborativen Schreiben eingesetzt wird.

[3] Grammarly Webseite: https://www.grammarly.com/ abgerufen am 8.4.2023

2.2.3 Augmented Reality

Der Fokus dieser Arbeit liegt in der Verbindung zwischen Augmented Reality und dem Bildungsbereich. Viele Forschungsarbeiten befassen sich mit dem aktuellen Wissensstand über Augmented Reality im Bildungsbereich. Dafür werden systematische Literaturrecherchen und / oder Meta-Analysen präsentiert, die Trends, Vor- und Nachteile, sowie nützliche Informationen zu diesem Thema kompakt zusammengefasst bieten. Der allgemeine Trend zeigt, dass das Interesse an AR-Systemen und auch an VR-Systemen im Bildungsbereich steigt und stetig Forschung betrieben wird. Besonders im Bereich der Naturwissenschaften und der Mathematik wird geforscht (ca. 50 %) [1, 12, 17, 44]. Ein interessanter Ansatz ist in der Forschungsarbeit Design Considerations for Haptic-Enabled Virtual Reality Simulation for Interactive Learning of Nanoscale Science in Schools zu finden [57]. Hier wurde eine virtuelle Realität, beziehungsweise ein virtuelles Modell einer Zellmembran erschaffen. Die SchülerInnen konnten in Zweiergruppen dieses virtuelle Modell erkunden und es zeigte sich, dass ein deutlicher Wissenszuwachs über die Konzepte der Zellmembran erreicht wurde. Ebenso wurde nach Unterschieden gesucht, wie sich haptische Bedingungen auswirken. Es zeigten sich keine signifikanten Anzeichen, dass eine haptische Bedingung einen Vorteil bringt. Aber die Wahrnehmung der TeilnehmerInnen war diesbezüglich unterschiedlich. Besonders der Vorteil, dass ein erhöhter Wissenszuwachs ermöglicht wird, wird als der am häufigsten auftretende Vorteil genannt. Eine gesteigerte Motivation folgt kurz danach auf dem zweiten Platz [1, 17]. Besonders die Literaturrecherche von Garzon et al. verdeutlicht die Chancen von AR-Systemen. In jeder der Studien wurde mindestens ein Vorteil gegenüber einer traditionellen Herangehensweise ohne AR-System herausgearbeitet, wobei nur 15% der Arbeiten einen negativen Einfluss feststellen konnten. Dabei werden als Nachteile die Komplexität (des AR-Systems), technische Schwierigkeiten und Multitasking hervorgehoben. Dies scheinen behebbare Faktoren zu sein. Ein AR-System muss für das entsprechende Alter und die kognitiven Leistungen der AnwenderInnen geplant und implementiert werden. Probleme mit Multitasking als Nachteil zu sehen ist schwierig, da es auch ein Vorteil ist, wenn sich der/die AnwenderIn voll auf das Lernsystem konzentrieren kann. Beispielsweise zeigte das Lernsystem MOW einen positiven Einfluss auf das Lernen von Vokabeln und Erlernen der englischen Sprache hat [5, 27]. Hierbei werden zu Vokabeln das entsprechende Bild gezeigt. Ist die Vokabel „Fliege” wird eine Fliege virtuell angezeigt. Beide Studien hatten keine große Anzahl an VersuchsteilnehmerInnen. Dennoch zeigte sich, dass die SchülerInnen nicht nur besser lernten, sondern sie auch das System gegenüber dem traditionellen bevorzugten. Vielleicht könnte das System auch das Lernen außerhalb der Schule verbessern. Bujak et al. beschäftigen sich mit der psychologischen Perspektive von AR-Systemen auf den Mathematikunterricht [10]. Ebenso wie die Arbeit von Kaufmann und Schmalstieg [20] zum Einsatz von Construct3D wird die Möglichkeit des kollaborativen Lernens und Arbeitens und dessen Nutzen thematisiert. Hierbei ist der Fokus auf einer kollaborativen Zusammenarbeit zwischen Lernenden. Sydorenko et al. verfolgen einen anderen Ansatz. Es bilden jeweils zwei Englisch-Lernende mit einem bereits Englisch-Sprechenden eine Gruppe, die zusammen ein Spiel (AR-System) bestreiten. Es zeigte sich, dass diese Gruppenkonstellation und das kollaborative Zusammenarbeiten sich positiv auf den Wortschatz der Englisch-Lernenden auswirkte. Diese Arbeit zeigt also, dass AR-Systeme und Spiele einen positiven Einfluss im Bildungsbereich haben können, ohne dass die Aufgaben traditionellen Schulaufgaben gleichen [53]. In der Arbeit von Matcha und Rambli, die sich explizit mit

kollaborativen Interaktionen auseinandersetzt, wird nochmals darauf hingewiesen, dass zur Kollaboration auch Kommunikation und Austausch gehört, was in vielen AR-Systemen nur mangelhaft umgesetzt wird [35]. Genaue und wissenschaftlich fundierte Kriterien zur Umsetzung eines AR-Systems für den Bildungsbereich wurden nicht gefunden. Besonderes Potenzial scheinen mobile AR-Systeme zu haben, da diese eine hybride Lernumgebung teils unabhängig vom Standort des/der AnwenderIn erzeugen können. Herpich et al. führten 2019 dazu eine separate Literaturrecherche durch, die nochmals verdeutlicht, dass die Anzahl an Forschungsarbeiten zu mobilen AR-Systemen stark gewachsen ist und weiter wächst [19]. Interessant für diese Arbeit sind auch die Forschungsergebnisse von Wang. Hier wurde ein AR-System mit AR-Lernmaterialien geschaffen und der Einfluss auf die Schreibaktivitäten der SchülerInnen untersucht. Das AR-System reagierte dabei individuell und mit direktem Feedback auf die SchülerInnen. Das AR-System förderte besonders bei leistungsschwachen SchülerInnen das Ideenreichtum dieser und verhalf zu einem einfacheren Start in den Schreibprozess. Auch die Schreibqualität und die Richtigkeit des Inhalts verbesserten sich [56].

Die Einsatzmöglichkeiten von AR-Systemen im Bildungsbereich sind vielfältig. Viele Systeme sind auf das Verständnis von wissenschaftlichen Zusammenhängen ausgelegt und wenig auf kollaboratives Arbeiten / Schreiben. In der Arbeit von Lin et al. wird das kollaborative Lernsystem AR-Physics genutzt und ein Wissenszuwachs festgestellt, wobei nicht explizit die Effektivität des kollaborativen Lernens untersucht wurde [28].

2.3 Abgrenzung des eigenen Themas zu verwandten Arbeiten

Diese Masterarbeit soll die Grundlage für die Beantwortung der Fragestellung, ob die positiven Effekte des kollaborativen Schreibens in einer erweiterten Realität mithilfe einer Handschrifterkennung zu beobachten sind, schaffen. Die bisherigen Forschungsergebnisse bezogen sich nur auf Teilaspekte dieses Themas. Dafür wird ein eigenes Softwaresystem entwickelt. Es kann nicht auf Systeme wie Google Docs, oder Augmented Reality Systeme zurückgegriffen werden, da beispielsweise das Schreiben mit Stift auf Papier geschehen soll. Aspekte wie eine mögliche Heuristik zur Auswertung der Qualität des Geschriebenen oder ein Fragebogen zur Motivation beim Schreiben werden sich nicht explizit von anderen Arbeiten unterscheiden müssen. Im Gegenteil, es sollten sich diese zwei Aspekte zur Auswertung an anderen Arbeiten orientieren. Denn so ist eine bessere Vergleichbarkeit und Interpretation der Ergebnisse möglich.

2.4 Schlussfolgerung

Die hier gewonnenen Erkenntnisse anhand der durchgeführten Literaturrecherche können nicht als ultimativ angesehen werden. Die schiere Menge an Forschungsarbeiten zu den verschiedenen Themen konnten nicht in Gänze gesichtet werden. Besonders im Hinblick auf eine Weiterentwicklung des Softwaresystems, das folgend geplant und entwickelt wird, sollte immer wieder nach neuen Erkenntnissen in neuen Forschungsarbeiten geschaut werden. Es wurden sowohl für AR-Systeme, als auch das neuartige kollaborative

Schreiben und Arbeiten viele positive Effekte gefunden, die teilweise aus kleinen Pilotstudien stammen und somit nicht repräsentativ sind. Dabei wurde oftmals vom Spracherwerb und anderen Lernsituationen gesprochen und nicht wie in dieser Arbeit vom Geschichten schreiben. Zusätzliche Aspekte wie das Bewerten von multimedialen Lernumgebungen werden erst im späteren Verlauf der Arbeit angesprochen. Die Tools zur Schreibqualität wie beispielsweise Grammarly könnten innerhalb der Studie zur Auswertung des Geschrieben genutzt werden, aber auch in einer Erweiterung des Softwaresystems in dieses direkt integriert werden, da dies scheinbar positive Effekte auf die Lernenden / AnwenderInnen hat. Im Hinblick auf die Handschrifterkennung sollte stets der aktuelle Forschungsstand betrachtet werden. Eine Open-Source-Lösung, die stetig weiterentwickelt wird, scheint vielversprechend zu sein, da es für einzelne Entwickler schwierig ist, eine bestmögliche Handschrifterkennung zu entwickeln.

3 Konzept

In diesem Kapitel wird das Konzept des Systems erläutert. Unterteilt wird dies zum einen in die Idee hinter der Implementierung und zum anderen in die Planung des Systemdesigns.

3.1 Idee der Implementierung

Die Idee der Implementierung ist es, ein System zu schaffen, mit dem die NutzerInnen kollaborativ eine Geschichte schreiben können. Dabei soll das kollaborative Geschichtenschreiben in Fernarbeit, auch Remote Work genannt, funktionieren. Das bedeutet, dass die NutzerInnen zwar räumlich voneinander getrennt sind, sie aber durch das System miteinander verbunden sind. Das Besondere an diesem System ist, dass die NutzerInnen ihre Geschichte mit einem Stift auf Papier schreiben können, der Text mithilfe einer Schrifterkennung erkannt wird und an den/die PartnerIn gesendet wird. Wichtig ist hierbei auch, dass der Text auch an die StudiendurchführerInnen gesendet wird, um eine spätere Evaluation zu ermöglichen.

Ein weiterer Aspekt ist die Darstellung des Geschriebenen. Dem/der NutzerIn soll eine vereinfachte Mixed-Reality (Augmented Reality) Erfahrung ermöglicht werden. Dafür soll der Text über das Kamerabild gelegt werden.

3.2 Systemdesign

Hinweis: Alle Dateien befinden sich in der beigelegten Dokumentation (CD) und im GitHub Repository [1]. Im Folgenden werden im jedem Abschnitt dieses Kapitels und dem Kapitel 4 Implementierung jeweils die Dateipfade zu den Ordnern der Dokumentation, die das Unterkapitel behandelt und der Dateipfad zu den Dateien, beispielsweise zu einem bestimmten Papierprototyp, angegeben.

Das Systemdesign beinhaltet verschiedene Planungsaspekte, die der Idee der Implementierung folgen. Es umfasst dabei die gesamte Planung des Softwaresystems, wobei dies nicht bedeutet, dass diese Aspekte vollständig im fertigen Softwaresystem umgesetzt wurden. Nach der Implementierung wurde die Planung nicht angepasst oder geschönt. Dennoch sind diese Schritte für ein erfolgreiches Softwaresystem essenziell. Das Beschreiben eines Softwaresystems und der zugrundeliegenden Abläufe gestaltet sich häufig schwierig. Die

[1] GitHub Repository https://github.com/JanotronUni/MasterthesisOCR

folgend aufzeigten Teile einer Softwareentwicklung sorgen nicht nur für Klarheit beim Programmierer selbst, sondern helfen auch bei der Kommunikation mit den Stakeholdern, in diesem Fall mit den Betreuern.

3.2.1 Proposal

Nach Klärung der Rahmenbedingungen (zeitlicher Umfang, Thema der Arbeit) wurde als Erstes ein Proposal, zu Deutsch Vereinbarung, erarbeitet. Das Proposal beinhaltet mehrere für die Planung wichtige Aspekte.

Unter anderem beinhaltet das Proposal eine Einleitung, die Einordnung des Themas inklusive verwandter Arbeiten sowie den eigenen Ansatz. Dies wurde in Kapitel Hintergrund und verwandte Arbeiten übernommen und überarbeitet.

Des Weiteren werden Voraussetzungen bezüglich inhaltlicher Abhängigkeiten und der Verfügbarkeit von benötigter Hard- und Software definiert. Hierbei wird die Heuristik zur Bewertung der Schreibqualität angesprochen und der für das Training benutze Datensatz [34]. Weitere inhaltliche Abhängigkeiten werden im Proposal nicht genannt. Als benötigte Hard- und Software werden ein Server sowie jeweils ein Tablet pro Teilnehmer definiert. (Zu diesem Zeitpunkt war noch geplant, dass die App für Tablets entwickelt werden sollte.)

Eine vorläufige Gliederung gab dem Projekt eine gewisse Struktur, wobei sich diese im Nachhinein von der Gliederung der Masterarbeit unterscheidet. Dabei ist auffällig, dass der theoretische Hintergrund auf drei wesentliche Kernbereiche reduziert wurde und das Kapitel Methoden unterteilt wurde in Konzept und Implementierung.

3.2.2 Erste Modellierung

Als erste Modellierung wurde ein unkonventionelles, sich nicht an UML-Richtlinien orientierendes Komponentendiagramm (Abbildung 3.1) skizziert[2]. Dies diente der Visualisierung und Absprache zwischen Betreuenden und dem an der Masterarbeit Arbeitenden.

Auf diesem lässt sich bereits eine Struktur erkennen, die im weiteren Verlauf nicht mehr stark variiert wurde. Ein Server, hier als main programm bezeichnet, verbindet zwei AnwenderInnen, und deren Apps miteinander. Die AnwenderInnen machen mit der App ein Foto des Geschriebenen und diese leitet dies an den Server weiter. Der Server übernimmt die Handschrifterkennung, die Analyse des erkannten Textes und leitet einerseits an das Research-Team die analysierten Daten und andererseits den erkannten Text an den / die andere AnwenderIn. Das Research-Team hat ebenfalls die Möglichkeit, Aufgaben an die AnwenderInnen mithilfe des Servers zu senden.

[2]Dateipfad: MasterthesisOCR/ZusaetzlicheDatein/ersteModellierung.png

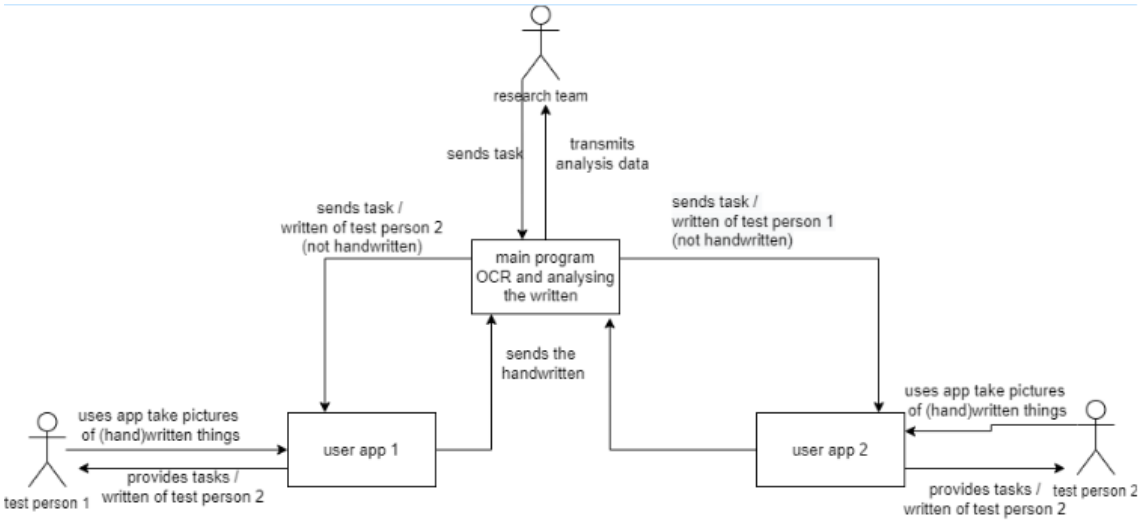


Abbildung 3.1: Erstes unkonventionelles Diagramm. Erste Visualisierung der Abläufe innerhalb des Softwaresystems

3.2.3 Papierprototypen

In der Planungsphase wurden drei Papierprototypen mit Stift auf Papier skizziert[3].

Papierprototyp - Server

Der erste Prototyp stellt eine Skizze der grafischen Oberfläche auf Seite des Servers dar[4].

Dieser Papierprototyp zeigt sechs mögliche Szenen (bzw. Zustände) und mögliche Funktionen, die ein Server mit grafischer Oberfläche erfüllen würde.

Die erste Szene zeigt sich beim Start der grafischen Oberfläche und ist ein Anmeldefenster. Nach der erfolgreichen Anmeldung wechselt die Ansicht zur zweiten Szene. Ansonsten soll eine Fehler-Meldung kommen und die Anmeldung kann wiederholt werden. In allen anderen Szenen befindet sich in der oberen rechten Ecke ein Logout-Button, über den wieder zur ersten Szene gewechselt wird.

Die zweite Szene birgt das Menü, von welchem auf alle anderen Szenen per Button gewechselt werden kann. Außerdem kann von jeder der folgenden Szenen über einen Zurück-Button wieder zum Menü gewechselt werden. Folgende Menüpunkte (Szenen) werden gezeigt: Gemeldete Probleme / Meldungen – Hier werden gemeldete Probleme angezeigt und Funktionen in Form von Buttons um auf diese zu reagieren. Beispielsweise könnte bei einem gemeldeten Problem eine Meldung an alle AnwenderInnen gesendet werden. Stand der Studie – Hier werden nur Daten der Studie angezeigt. Dazu gehört ein

[3] Dateipfad: MasterthesisOCR/ZusaetzlicheDatein/Papierprototypen/
[4] Datei: ServerGUI.pdf

Zeitplan, der bei verschiedenen aufeinanderfolgenden Aufgaben anzeigt, wann die Aufgabe startet, wie die Aufgabenstellung lautet und wann die Aufgabe endet. Neben dem Zeitplan sollen bereits durch eine Analyse herausgefundenen Daten angezeigt werden. Beispielweise, wie viele AnwenderInnen noch keinen Text abgegeben haben und wie viele bereits fertig sind mit dem geschriebenen Text. Aufgabenplanung – Diese Szene hat die Funktion, neben dem Anzeigen des aktuellen Aufgabenstandes Aufgaben hinzuzufügen, zu bearbeiten und zu löschen. Analyse – Über eine Verbindung mit der Datenbank können hier alle Daten (Texte und Analyseergebnisse) abgefragt und downgeloadet werden.

Papierprototyp – App

Als Nächstes wurde ein Papierprototyp für die grafische Oberfläche der App entworfen[5]. Dieser Papierprototyp beinhaltet zwölf Szenen / Ansichten, skizziert für ein Smartphone. Diese Szenen enthalten teils Hinweise, Kommentare, mögliche Erweiterungen, aber auch bereits fest eingebaute Funktionen und Szenenwechsel. Folgend werden die Szenen genannt, ohne dabei auf Szenenwechsel und die genauen Beschreibungen aller Funktionen einzugehen. In der Start-Szene ist der Wechsel zu verschiedenen Szenen möglich. Um die Arbeitsbereich-Szene auszuwählen muss vorher in die ID-Szene gewechselt werden. Die Hilfe/Anleitung-Szene, sowie die Problem-Melde-Szene sind direkt zu erreichen. Die ID-Szene ist im Optimalfall nur beim ersten Starten vonnöten. Hier wird die an den/die TeilnehmerIn vergebene ID eingegeben. Des Weiteren die Hilfe/Anleitung-Szene, in der die Bedienungseinleitung und die Antworten auf eventuell häufigere Fragen aufgelistet sind. In der Problem-Melde-Szene ist vonnöten, falls im Verlauf der Studie Probleme auftreten, kann der / die AnwenderIn das Problem melden. Bisher bekannte Probleme inkl. Lösungen werden ebenfalls angezeigt. es folgt die Arbeitsbereich-Szene, von der sich zu allen Szenen wechseln lässt, die für das kollaborative Schreiben mit dem Partner wichtig sind. In der Aufgabenstellung-Szene werden allein die Aufgabenstellung wird angezeigt. Die Aktueller-Stand-Szene zeigt nochmals die Aufgabenstellung sowie den von den gepartnerten AnwenderInnen bisher verfasste Text. Es folgt die Kommentar-Szene. In dieser wird es den AnwenderInnen ermöglicht, bisher verfasste Textstellen zu kommentieren. Kommentare können ebenfalls, zum Beispiel vom PartnerIn kommentiert, bearbeitet oder gelöscht werden. Ziel ist es, dass unklare Textstellen mit dem/der PartnerIn besprochen werden können. Abgabe-Szene – Hier wird ein Foto des Textes gemacht, welches dann auch versendet werden kann. In der Foto-Status-Szene wird auf die Antwort des Servers auf ein abgesendetes Foto gewartet und anschließend angezeigt. Hier wird gezeigt, ob es einen Fehler gab, das Foto zu unscharf war, der Text nicht erkannt wurde oder, dass die Verarbeitung funktionierte. Die Chat-Szene ermöglicht einen Austausch unter den PartnerInnen. Die Kommentar-Szene dient einzelner Textstellen und die Chat-Szene einem allgemeinen Austausch während des kollaborativen Schreibens. Als Zusatz gibt es noch die Untätig-Szene, in der sollte eine Szene eingebaut werden, die beschreibt, was passiert, wenn lange keine Eingabe erfolgt, obwohl der/die AnwenderIn dies ankündigte.

Da dieser Papierprototyp viele Funktionen enthält, die weit über die in dieser Arbeit verlangten hinausgeht, wurde ein weiterer Papierprototyp entworfen.

[5] Datei: App.pdf

Minimaler Papierprototyp – App

Der minimale Papierprototyp zeigt alle festgelegte Funktionen, die in dieser Masterarbeit umgesetzt werden sollen[6]. Eine große Änderung in der Planung lässt sich beim minimalen Papierprototypen erkennen. Er zeigt den damals mit den Betreuenden besprochene Wechsel zum Einsatz von Smartphones zum Einsatz von Tablets. Der Prototyp zeigt sechs Zustände der App, die sich auf drei Szenen runterbrechen lassen. Die genannten Bilder lassen sich in der Datei finden. Die Lese-Szene (Bild 1,3,4) startet die App. Die Aufgabenstellung wird inklusive der bisher verfassten Teile der Geschichte angezeigt. Wenn es nötig ist, ist das Scrollen durch die verfassten Teile möglich. Von dieser Szene kann nur in die Foto-Szene gewechselt werden. In der Foto-Szene (Bild 2) lässt sich ein Foto machen, während eine Vorschau des Kamerabildes angezeigt wird. Durch das Klicken des Buttons „Foto“ wird mit der Kamera fotografiert und das Foto weitergeleitet. Falls im Foto kein Text erkannt wird, wird diese Szene neu gestartet. Als Letztes die Einzeleingabe-Szene (Bild 5,6), die dazu dient, einzelne Wörter, die schlecht erkannt wurden, also die Genauigkeit der Wiedererkennung gering ist, zu korrigieren. Dafür werden Fotos der nicht erkannten Textstellen angezeigt. Der/die AnwenderIn kann dann das gemeinte Wort eingeben.

3.2.4 Komponentendiagramm

Das Komponentendiagramm[7] soll die Strukturen und Beziehungen dieses Systems zur Laufzeit darstellen. Für die Visualisierung der Prozesse und dem Zusammenspiel der Komponenten wird das System zur Laufzeit mit zwei gepartnerten AnwenderInnen modelliert. Die AnwenderInnen benutzen jeweils ein identisches System, wobei sie dabei auf die Model-View-Controller Einheit der App zugreifen. Von dort aus wird auf die Komponente Cam zugegriffen, die für das Aufnehmen eines Bildes verantwortlich ist. Auffällig ist hier, dass in diesem Diagramm die Handschrifterkennung, kurz OCR, nicht nur als Komponente des Servers, sondern auch als Komponente der App und damit Client- / Anwenderseitig ist.

Der Server wird skizziert als eine Komponente, die aus zwei Komponenten besteht, dem main program (Hauptprogramm) und der database (Datenbank) oder als Alternative Google Docs. Dies zeigt auf, dass zu diesem Zeitpunkt die Planung noch nicht abgeschlossen war, sondern noch andauerte. Eine eventuelle Speicherung in Google Docs erschien eine Alternative zu einer Datenbank darzustellen. Die Analyse soll nach diesem Komponentendiagramm zur Laufzeit stattfinden. Das Hauptprogramm würde anhand der durch die Handschrifterkennung und die ID des / der AnwenderIn eine Analyse durchführen und alle Dateien an die Datenbank leiten. Die Datenbank würde noch die ID des Partners mit verarbeiten und alles entsprechend in der Datenbank speichern.

[6] Datei: MinimalApp.pdf
[7] Dateipfad: MasterthesisOCR/ZusaetzlicheDatein/KomponentenDiagramm.png

3.2.5 Anforderungen

In den Anforderungen[8] wurde das anfänglich festgelegte Ziel, die Beantwortung der Forschungsfrage: *„Erhöht kollaboratives Schreiben in einer erweiterten Realität die Motivation und die Schreibqualität?“* auf sieben Arbeitsschritte verteilt. Als Erstes wurde die Implementierung einer eigenen Handschrifterkennung (Schritt 1) festgelegt. Die Implementierung einer App, die das kollaborative Schreiben ermöglicht (Schritt 2), ebenso. Zudem schien die Implementierung und Aufsetzen eines Servers (Schritt 3)notwendig, wobei dafür auch das Mieten eines Servers in Betracht gezogen wurde, damit die AnwenderInnen die Möglichkeit haben, miteinander zu kommunizieren. Der Server stellt damit die Verbindungseinheit zwischen den AnwenderInnen dar. Für die Speicherung des Geschriebenen wurde die Anforderung „Aufsetzen einer Datenbank” (Schritt 4) verfasst, die noch nicht die genaue Struktur der Datenbank beschreibt. Für die Auswertung des Geschriebenen anhand einer Heuristik würde ein Programm zur Auswertung der Schreibqualität (Schritt 5)einerseits den weiteren Arbeitsaufwand erleichtern und andererseits eine objektive Auswertung ermöglichen. Als weiterer Arbeitsschritt wurde das Planen und die Durchführung einer Nutzerstudie (Schritt 6) festgelegt, in der das kollaborative Schreiben mithilfe der App untersucht werden soll. Schlussendlich ist der letzte Schritt die Dokumentation des gesamten Prozesses (Schritt 7), wobei schon während der einzelnen Arbeitsschritte dokumentiert wird und im letzten Schritt die Fertigstellung der Dokumentation stattfindet.

3.2.6 Meilensteine

Die Meilensteine[9] folgten den ersten zwei Papierprototypen und bilden die verschiedene Stufen der Softwareentwicklung ab. Der erste Meilenstein stellt die Mindestanforderungen an das Softwaresystem dar und hat somit die höchste Priorität bei der Implementierung. Alle Meilensteine, die dem ersten folgen, bilden lediglich nützliche Features, die die Usability für die NutzerInnen erhöhen könnten, ab. Die Meilensteine wurden im Planungsprozess leicht geändert, inklusive der Priorisierung. Während der reinen Softwareimplementierungsphase wurden diese nicht verändert.

Im ersten Meilenstein wurde festgelegt, dass die TeilnehmerInnen die Möglichkeit haben, kollaborativ eine Geschichte zu schreiben. Dafür wird eine App genutzt, die jedem TeilnehmerIn durch das Aushändigen eines Tablets bereitgestellt wird. Die Apps interagieren mit dem Server und nicht direkt untereinander. Der Server koppelt zwei TeilnehmerInnen aneinander, die folgend eine Geschichte zusammen schreiben. Gespeichert wird dies in einer Datenbank. Die TeilnehmerInnen erhalten die Aufgabe auf ihre App. Haben Sie einen Abschnitt fertig geschrieben und wollen diesen an den gekoppelten PartnerIn senden, können Sie von ihrem Abschnitt ein Foto machen. Vorerst kann das Foto überprüft werden, damit kein verschwommenes Foto übermittelt werden. Dieses wird entweder direkt über die App oder durch den Server von einem OCR-Programm geprüft. Sollte das OCR-Programm Probleme haben, handgeschriebene Wörter zu erkennen, wird der Nutzer aufgefordert, dieses nachträglich per Tastatur zu korrigieren. Festgemacht werden

[8] Dateipfad: MasterthesisOCR/ZusaetzlicheDatein/MeilensteineAnforderungen.pdf
[9] Dateipfad: MasterthesisOCR/ZusaetzlicheDatein/MeilensteineAnforderungen.pdf

die „Probleme“ anhand vordefinierter Metriken. Eine solche Metrik ist die „accuracy”, zu Deutsch Genauigkeit. Diese gibt an, wie sicher das OCR-Programm ist, dass die Ausgabe richtig ist. Bei einer niedrigen accuracy hat das Programm Probleme, die Schrift zu erkennen und eine Auswertung durch ein Programm zur Schreibqualität wäre hinfällig, da dies die Werte verfälschen würde. *Beispiel: Teilnehmer schreibt „Hello“. Das OCR-Programm erkennt das Wort nicht und würde es mit einer accuarcy von 70 % als „Hello“ erkennen. Dadurch würde das Programm zur Schreibqualität einen Rechtschreibfehler erkennen und die Schreibqualität würde sinken. Durchgestrichenes würde ebenfalls erkannt werden. Dafür sollte ein Löschenknopf vorhanden sein.* Das Foto des Abschnitts wird an den Server gesendet und an den/die gekoppelten PartnerIn geschickt. Die (gegebenenfalls korrigierte) Ausgabe des OCR-Programms und die Auswertung zur Schreibqualität werden in der Datenbank gespeichert. Diese Daten können jederzeit durch eine einfache App der StudiendurchführerInnen abgefragt und downgeloadet werden. Der / die gekoppelte TeilnehmerIn bekommt das Foto des Abschnitts unterhalb der Aufgabenstellung angezeigt. Jeder weiter geschriebene und verschickte Abschnitt wird unter dem letzten hinzugefügt. Die Verarbeitung des Fotos läuft wie im Vorhinein beschrieben. Sollte ein / eine TeilnehmerIn das Geschriebene des / der PartnerIn nicht lesen können, soll statt des Fotos die (gegebenenfalls korrigierte) Ausgabe des OCR-Programms erscheinen. Gegebenenfalls soll es einen Button für eine Anleitung geben. Mit zunehmenden Features sollte es diesen Button auf jeden Fall geben. Laut dem zweiten Meilenstein soll eine Chat-Funktion hinzugefügt werden, mit der die gekoppelten TeilnehmerInnen kommunizieren können. Dies liegt der fehlenden Möglichkeit zur Kommunikation zugrunde, die es im traditionellen Schreiben, beispielsweise durch Gespräche gegeben ist. Zudem sind sich kollaborative SchreibpartnerInnen meist nicht gänzlich unbekannte und es gibt Wege, wie ein Telefonat, Austausch über soziale Medien, Chats oder eingeschränkter über eine Kommentarfunktion, für einen Austausch unter diesen. TeilnehmerInnen bekommen Benachrichtigungen wurde als dritter Meilenstein bestimmt. *„Beispiel: Dein Partner / Deine Partnerin haben einen neuen Teil zur Geschichte hinzugefügt” könnte eine solche Benachrichtigung lauten.* Dies ermöglicht Rückschlüsse zu ziehen, um Benachrichtigungen die Motivation zum Schreiben erhöhen. Dazu könnte untersucht werden, ob TeilnehmerInnen häufiger und in schnellerem Abstand etwas zu Geschriebenen beitragen, wenn Sie Benachrichtigung über den aktuellen Stand des Erarbeiteten erhalten. Im dritten Meilenstein wurde festgelegt, dass das Kommentieren des Geschriebenen ermöglicht. Eine Chatfunktion würde durch den zweiten Meilenstein bereits bestehen und damit die Möglichkeit, dass die TeilnehmerInnen miteinander kommunizieren. Eine Kommentarfunktion würde die Kommunikationsmöglichkeiten deutlich verbessern, da präzise im Geschriebenen eine Stelle markiert werden, zu der einer der TeilnehmerInnen Kommunikationsbedarf hat. Besonders bei längeren Texten könnte es zur schnelleren und effektiveren Aufklärung von Missverständnissen kommen. Status einfügen für TeilnehmerInnen als letzter Meilenstein. *Beispiel: Aktuell am Schreiben.* Wenn es einen eigenen Arbeitsbereich geben würde, würde das verhindern, dass beide am gleichen Teil der Geschichte schreiben. Weitere Status sind möglich, wie zum Beispiel ein AFK-Status, der den „am Bearbeiten“ Status nach einiger Zeit ohne Eingabe, trotz Aufforderung, zurücksetzt. Dadurch wäre der Arbeitsbereich nicht durchgehend belegt, wenn ein/eine der TeilnehmerInnen in den Arbeitsbereich begibt und den Status auf „Aktuell am Schreiben” setzt und dann nicht dran arbeitet und es gegebenenfalls einfach vergisst.

3.3 Zusammenfassung und Ausblick

In diesem Kapitel wurde die Idee hinter der Implementierung, beziehungsweise hinter dem angestrebten Softwaresystem erläutert. Es soll kollaboratives Schreiben inklusive einer vereinfachten Mixed-Reality Erfahrung ermöglicht werden. Dafür wurde folgend ein Softwaresystem geplant. Es wurden verschiedene Aspekte zur Softwareplanung ausgearbeitet. Dazu gehörte ein Proposal, das keinen üblichen Punkt eines Softwareentwicklungsprozesses darstellt, aber Punkte wie beispielsweise den zeitlichen Rahmen einzelner Schritte sowie Voraussetzungen für das Softwaresystem beinhaltet. Verschiedene Modellierungen wurden in der Planungsphase vorgenommen. Diese dienten nicht nur dem Bewusstwerden über notwendige Komponenten und Funktionen des Softwaresystems, sondern auch für die bessere Kommunikation mit den Stakeholdern. Die erste Modellierung eines Komponentendiagramms bot den ersten Ansatz. Ein ausgearbeitetes Komponentendiagramm wurde im Nachhinein ausgearbeitet und erläutert. Außerdem wurden drei Papierprototypen skizziert. Einer für den Server und zwei für die App. Besonders der erste Papierprototyp zeigt viele mögliche Erweiterungen der App auf, beispielsweise ein Hilfs-/Anleitungsbereich auf Seite der App oder auch das Senden von Benachrichtigungen serverseitig durch die Studiendurchführenden. Ob diese das Arbeiten mit der App für die AnwenderInnen erschwert, würde eine zusätzliche Evaluation erfordern. Diese möglichen Erweiterungen werden auch in den Anforderungen und Meilensteinen aufgegriffen. Die Erweiterungen wurden im vorherigen Kapitel nicht mit betrachtet. Eine Literaturrecherche zu möglichen verwandten Arbeiten zu diesen Erweiterungen ist vonnöten, sowie eine ausführliche Planung. Es zeigte sich, dass verschiedene Komponenten notwendig sind: eine App, ein Server, eine Datenbank und eine Handschrifterkennung. Es ist auch darauf hinzuweisen, dass es verschiedene Vorgehensweisen für Softwareentwicklungsprozesse gibt. Dabei unterscheiden sich die Schritte und auch die Planungsaspekte. Horizontale und vertikale Prototypen inklusive Implementierung und Evaluation und das Erstellen eines Softwarestacks sind nur Beispiele für weitere Planungsaspekte.

4 Implementierung

In diesem Kapitel wird die Implementierung erläutert, die auf die im vorherigen Kapitel beschriebene Planung folgte. Dabei werden die drei Kernbereiche Handschrifterkennung, Server und App genauer betrachtet. Unterstützt wird dies durch Abbildungen und Quelltexten aus den verschiedenen Bereichen. Die vollständigen Quelltexte lassen sich in der beigelegten Dokumentation (CD) oder im GitHub Repository [1] finden. Dort befinden sich alle Dateien des Projekts inklusive der APK (Android Package Kit), mit welcher sich die App auf einem Smartphone installieren lässt.

Hinweis: In Kapitel 4.4 erfolgt ein Durchlauf durch die App, mit dem Klarheit über das Zusammenspiel der einzelnen Komponenten gebracht werden soll.

4.1 Handschrifterkennung

Die Implementierung einer Handschrifterkennung stellt den ersten zentralen Punkt der Implementierung dar[2]. Dafür wurde in den ersten Wochen nach und während der Planungsphase versucht, eine zuverlässige Handschrifterkennung zu implementieren. Bereits existierende Open-Source Lösungen sollten vorerst nicht genutzt werden, wurden aber im Prozess der Implementierung des Systems ausprobiert und später mit eigenen Schreibproben evaluiert. Entschieden wurde sich für die Arbeit mit Keras und TensorFlow.

4.1.1 Training des Models

Grundlage für die Handschrifterkennung war die auf der Keras-Webseite vorgeschlagene Handschrifterkennung [39], basierten auf dem Datensatz [34]. Der für Google-Colab entworfene Quellcode wurde an Python angepasst, der Datensatz downgeloadet und das erste eigene Modell zur Handschrifterkennung konnte trainiert werden[3]. Die Geschwindigkeit konnte deutlich erhöht werden, nachdem das Nvidia CUDA Toolkit downgeloadet und entsprechend installiert wurde. „Gute" Ergebnisse sollten ab 50 Epochen eintreten. Eine Epoche entspricht einem Durchlauf durch die Trainingsdaten.

[1] GitHub Repository https://github.com/JanotronUni/MasterthesisOCR
[2] Dateipfad: MasterthesisOCR/Handschrifterkennung/
[3] Datei: ModelTraining.py

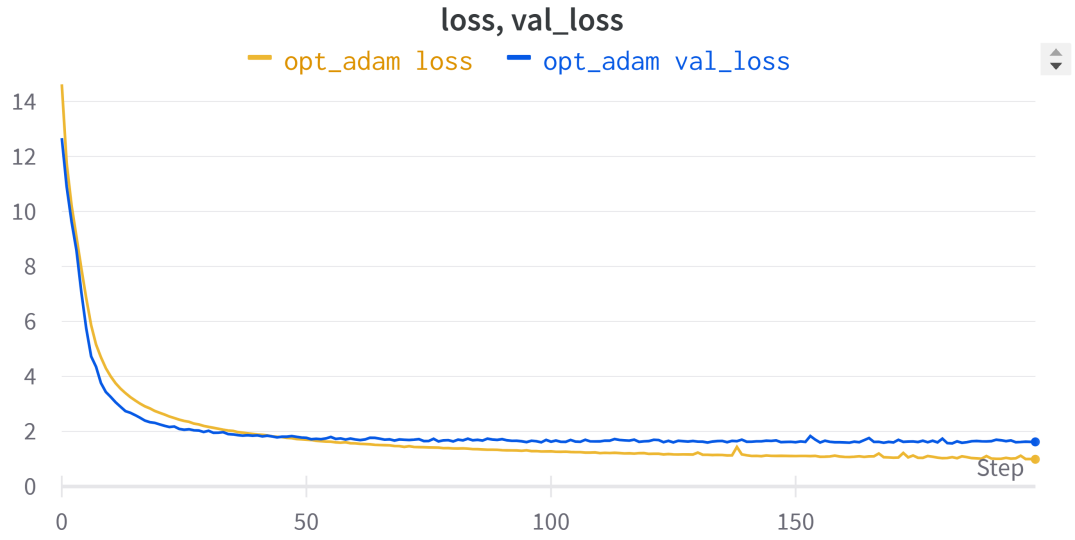


Abbildung 4.1: Weights & Bias: Daten eines Trainings

4.1.2 Speicherung und Umwandlung zu TF-Lite-Modell

Im nächsten Schritt wurde eine Speicherung des Modells eingebaut, um das Modell wiederzuverwenden und nicht jedes Mal neu trainieren zu müssen. Durch den Hinweis im Google-Colab, dass sich sogenannte TF-Lite Modelle besonders für mobile Anwendungen eignen, wurde das Umwandlungsskript implementiert. Dieses öffnet das bereits gespeicherte Modell, bildet das sogenannte prediction_model und speichert es als Tf-Lite Modell (.tflite) ab[4].

4.1.3 Lauffähiges Skript zur Handschrifterkennung

Als vorerst finales Skript zur Handschrifterkennung wurde das Texterkennungskript[5] entwickelt. Hier musste auf verschiedene Methoden aus dem Trainingsskript zurückgegriffen werden. Dazu gehören die Funktionen zur Vorbereitung der Bilder, sowie die leicht veränderte Funktion zum Dekodieren der Ergebnisse aus dem Google-Colab. Der Array, der im Trainingsskript verwendet wird, wurde ebenfalls manuell eingefügt. Schlussendlich wurde Beispielcode zum Tf-Lite Interpreter von der offiziellen Webseite von TensorFlow entnommen.

4.1.4 Weiterentwicklung des Trainingsskripts

Ein bis dahin bestehendes Problem war, dass nach dem Start erst nach allen durchlaufenden Epochen ein Modell gespeichert wurde. Das Trainieren über viele Epochen dauerte trotz CUDA Nutzung und starker GPU lang und das Beobachten des Trainingsprozesses war nicht möglich. Daher wurde im nächsten Schritt Wandb, kurz für Weights & Biases eingebunden und deren Beispielcode [6] an das Trainingsskript angepasst[6]. Durch das Einbauen der Callbacklist 4.1, wird nach jeder Epoche verglichen, ob val_loss niedriger ist als beim bisher gespeicherten Modell. Dies steht für Validation-Loss und gibt vereinfacht an, wie hoch die Fehlerquote bei den Testdaten ist. Der Loss-Wert bezieht sich auf die Trainingsmenge. Falls der Val_loss-Wert niedriger ist, wird das aktuelle Modell gespeichert und überschreibt damit das vorher gespeicherte. Das Training würde über 2000 Epochen laufen, sollte sich der Wert stetig verringern. Die EarlyStopping-Funktion sorgt dafür, dass nicht weiter trainiert wird, wenn nach zehn Epochen der niedrigste val_loss Wert sich nicht um 0,001 verbessert.

```
checkpoint = ModelCheckpoint(filepath=file_path,
                      monitor='val_loss',
                      verbose=1,
                      save_best_only=True,
                      mode='min')

   callbacks_list = [checkpoint,
                WandbCallback(monitor="val_loss",
                          mode="min",
                          log_weights=True),
                PlotPredictions(frequency=1),
                EarlyStopping(min_delta =0.001, patience=10, verbose=1),
                ]
   history = model.fit(train_ds,
                 epochs =2000,
                 validation_data=validation_ds,
                 verbose = 1,
                 callbacks = callbacks_list,
                 shuffle=True)
```

Listing 4.1: Modell-Training mit Stopp-Bedingung

Ein Durchlauf mit 199 Epochen ist in der Grafik 4.1 abgebildet. Der Durchlauf wurde opt_adam genannt, da, wie von Keras empfohlen, der Adam Optimierer genutzt wurde. Ab etwa 25 Epochen stagnierte der Val_loss-Wert. Auch der Loss-Wert sank ab da nur sehr langsam. Hier ist Overfitting aufgetreten, und zwar bereits vor den empfohlenen 50 Epochen. Das Modell ist zu stark auf die Trainingsdaten angepasst.

[4]Datei: ModelZuTfLite.py
[5]Datei: TexterkennungTfLite.py
[6]Datei: ModelTraining.py

4.1.5 Problematiken bei der Texterkennung

Die Wiedererkennung von Wörtern funktioniert mit dem TF-Lite-Modell[7] nur zu einem gewissen Maße. Außerdem braucht dieses im besten Fall ein Foto von nur einem Wort. Daher war ein zusätzliches Modell zur Textdetektion vonnöten, welches auf einem Foto alle Wörter einzelnen ausschneidet und nacheinander in das TF-Lite-Modell zur Texterkennung einfügt.

TensorFlow stellt auf ihrer Webseite[8] eine Beispielapp zur Verfügung und empfiehlt, TF-Lite-Modelle mit Android Studio oder Swift zu verwenden.

Daher startete die weitere Entwicklung in Android-Studio. Nach der Empfehlung eines Betreuers wurde jedoch zu Unity gewechselt. Dort gab es mit den verwendeten TF-Lite Modellen große Kompatibilitätsprobleme, da diese nicht offiziell unterstützt werden und Funktionen, die in Android-Studio enthalten sind, hier fehlen. Dennoch ist eine Weiterentwicklung der App mit Hinblick auf eine Augemented-Reality oder Virtual-Reality durch zahlreiche vorgefertigte Pakete in Unity deutlich einfacher. Auch das Erstellen der grafischen Oberfläche für die App ist in Unity simpel.

Aufgrund der Kompatibilitätsprobleme wurde auf eine Handschrifterkennung innerhalb der App verzichtet und diese auf den Server ausgelagert. Wegen der anfangs erwähnten Recherche zu verschiedenen (Hand-)Schrifterkennungen wurden diese nochmal anhand verschiedener Schriftproben getestet. Die Ergebnisse waren durchwachsen und Schreibschrift wurde deutlich schlechter erkannt als Druckschrift. Einzig OneDrive (Eingabe erfolgt über einen Stift auf einem Tablet) lieferte recht genaue Ergebnisse. Dies lässt sich aber nicht in ein solches Softwaresystem einbauen. Trotz des bereits großen Aufwandes, der für die Entwicklung einer eigenen Handschrifterkennung aufgebracht wurde, fiel deshalb die Entscheidung, auf die Opensource-Lösung der Handschrifterkennung EasyOCR zu wechseln.

4.1.6 EasyOCR

EasyOCR liefert für verschiedene Handschriften zuverlässigere Zeichen- und Wortvorschläge. Sie funktioniert allerdings nur in Python. Aus diesem Grund wurde sich endgültig für die serverseitige Handschrifterkennung entschieden. Ein großer Unterschied ist hierbei, dass EasyOCR nicht nur Text, sondern auch die Position auf dem Bild erkennt. Diese Art von Texterkennung wird auch als Detektion bezeichnet- EasyOCR erkennt komplett den Text auf einem Bild und prüft ihn danach auf Wiedererkennung. In der konkreten Implementierung bietet es dadurch den zusätzlichen Nutzen, Werte zu sogenannten Bounding Boxen anzugeben. Diese geben für ein jedes Wort Koordinaten für ein Rechteck um den erkannten Text an. Diese Bounding Boxen werden im späteren Verlauf genutzt, um die Genauigkeit des Textes zu erhöhen und um den Text zu korrigieren.

[7] Datei: /MasterthesisOCR/ZusaetzlicheDatein/tflite/tf.tflite

[8] Tensorflow: Optical character recognition: https://www.tensorflow.org/lite/examples/optical_character_recognition/overview abgerufen am 8.4.2023

EasyOCR wurde in den Server eingearbeitet. Nach der Initialisierung des Readers (siehe 4.2), kann die Funktion zur Textdetektion und Wiedererkennung von Wörtern ausgeführt werden. Die Readtext-Methode mit einem vorher als Opencv eingelesenen Bild als Parameter entspricht dieser Methode. Diese Funktion gibt eine Ergebnisliste zurück. Darin enthalten sind die Bounding Boxen (bbox), das erkannte Wort (text) und die Wahrscheinlichkeit, mit der das Wort korrekt erkannt wurde (prob). Diese Funktion wird innerhalb der Overlay_ocr_text-Methode verwendet. Die Ergebnisliste wird durchlaufen und das wiedererkannte Wort und die Wahrscheinlichkeit für das korrekte Erkennen angegeben. Jedes Wort (text) wird der Liste texte hinzugefügt, die beim Aufruf der Methode geleert wurde. Falls die Wahrscheinlichkeit für ein korrektes Wort unter 85% ist, wird der aktuelle Index in der Ergebnis-Liste mithilfe eines Zählers in einer Liste namens NiedrigeProb hinzugefügt. Somit enthält die NiedrigeProb-Liste die Indizes der Ergebnisliste, die für die Auswertung zu gering sind. Mithilfe der Bounding Boxen wird aus dem Foto ein Teilfoto, genannt crop, das nur das aktuelle Wort enthält. Dieses wird mehrfach umgewandelt, um als Utf-8 String in der Bilder-Liste abgespeichert zu werden. Anschließend wird der Zähler erhöht. Die Print-Methoden dienen als Hilfe für einen besseren Überblick. Wurde die Ergebnisliste vollständig durchlaufen, sind in der NiedrigProb-Liste alle Indizes mit schlecht erkannten Wörtern, in der Bilder-Liste diese schlecht erkannten Wörter als String und in der Texte-Liste alle Texte enthalten.

4.2 Server

Der Server[9] bildet die zentrale Einheit in diesem Projekt. Er ist zuständig für die Interaktion mit den Clients und dem Austausch zwischen gepartnerten Clients. Clients meint in diesem Fall die Apps der einzelnen AnwenderInnen. Zudem laufen die Schrifterkennung sowie die Datenspeicherung auf dem Server.

Hinweis: Durch die passende CUDA Installation (unterscheidet sich je nach Betriebssystem und Python-Version) wird die Schrifterkennung teils stark beschleunigt.

Der Server ist aufgeteilt in zwei Python-Skripte, um die Übersichtlichkeit und Handhabung zu vereinfachen. Das Datenbankskript (datenbank.py) bietet dabei die Funktion zur Initialisierung der Datenbank und Hilfsfunktionen, die mithilfe von SQL-Befehlen mit der Datenbank interagieren. Das Webserverskript (server.py) beinhaltet Methoden zur Schrifterkennung mittel EasyOCR, die Verbindungen zwischen den AnwenderInnen und eine Flask-Anwendung, die HTTP-Anfragen verarbeitet.

Eine grafische Oberfläche wurde nicht implementiert. Zum Starten des Servers reicht es, das Webserverskript auszuführen. Dies geschieht entweder über die Eingabeaufforderung oder über eine IDE. Der EasyOCR-Reader wird mit den Einstellungen für die englische Sprache gestartet. Bei der erstmaligen Initialisierung von EasyOCR werden alle nötigen Einstellungen für den Reader in der englischen Sprache downgeloadet. Weitere Informationen und hier genutzte Methoden von EasyOcr in 4.1.6.

[9] Dateipfad: MasterthesisOCR/Server/

```
if __name__ == '__main__':
   Datenbank.erstelleDB()
   app.run(host='192.168.1.44',port = 8000,debug=True)
```

Listing 4.2: Initialisierung der Flask-Anwendung und der Datenbank

Außerdem erfolgt die Ausführung des Codes 4.2. Dadurch wird innerhalb der Datenbank der Code 4.3 ausgeführt, wodurch eine Datenbank namens DB.db innerhalb des Server-Ordners erstellt wird. App.run startet die Flask-Anwendung. In diesem Fall als Webserver mit der IP-Adresse 192.168.1.44, dem Port 8000 und einem eingeschalteten Debug-Modus. Dieser ist erreichbar über http://192.168.1.44:8000. Der Debug-Modus sorgt für das Anzeigen von eingehenden HTTP-Anfragen und deren Statuscodes. Die IP-Adresse und gegebenfalls der Port müssen angepasst werden an die IP-Adresse des Systems, auf dem das Webserverskript ausgeführt wird.

```
@classmethod
   def erstelleDB(cls):
      db = sqlite3.connect(db_path)
      c = db.cursor()
      c.execute(
        "CREATE TABLE IF NOT EXISTS " +
         "datenbank(id INTEGER PRIMARY KEY, " +
         "name TEXT, " +
         "foto TEXT, " +
         "text TEXT)")
      db.commit()
      c.close()
```

Listing 4.3: Code zur Tabellenerstellung

SQLite sorgt dabei durch die Connect-Methode für die Verbindung zu Datenbank. Der db_path wurde im Vorhinein als DB.db festgelegt. Der Cursor bildet die Verbindung zur Datenbank und über ihn werden die SQL-Queries ausgeführt. Diese Queries werden bei allen Befehlen an die Datenbank übertragen und aufgeführt. Die hier vorliegende Query (Der String innerhalb der Execute-Methode) 4.3 sorgt bei der gefolgten Ausführung dafür, dass, falls innerhalb der DB-Datenbank noch keine Tabelle „datenbank" vorhanden ist, diese erstellt wird. Dabei beinhaltet die Tabelle vier Spalten: id als Integer, der den primären Schlüssel der Tabelle bildet, gefolgt von name, foto und text, die jeweils Text, beziehungsweise Strings enthalten.

Ein Blick in das Datenbankskript zeigt, dass dort eine Klasse zu finden ist, inklusive Initialisierungsmethode, die sich an den Spalten der Tabelle angleicht. Ebenso die Methode erstelleDbObjekt(). Eine Instanz der Klasse Datenbank im Datenbankskript entspricht somit einer Zeile in der Tabelle der Datenbank. Dies erleichtert den weiteren Einsatz, wie das Speichern einer Datenbankinstanz in die Datenbank.

Durch die im Webserverskript festgelegten @app.route() Funktionen, gefolgt von je einer Methode, definieren den Ablauf für die in @app.route() angegebene HTTP-Anfragen bis zur HTTP-Antwort, die der Server an den Client zurücksendet.

Ein leicht verständliches Beispiel für den gesamten Ablauf wäre eine Anfrage an https://192.169.1.44:8000/text (4.4), bei dem ein Json-Objekt mit dem Inhalt 'name' = "personid1" im Body dieser wäre. Dies ist ein HTTP-POST-Request, also eine HTTP-Anfrage, bei der Daten übermittelt werden. Das Ziel ist es, für die übergebene ID und

den möglichen Partner alle bisher geschriebenen Texte als einen Text an den Client zurückgeben.

```
@app.route("/text", methods=["POST"] )
def gettext():
   json = request.get_json()
   if not json:
      return "Keine Daten erhalten"
   name = json.get('name')
   print(name)
   if (hatPartner(name) ==False):
      results = Datenbank.getTextohnePartner(name)
   else:
      results = Datenbank.getTextmitPartner(name, getPartnerName(name))
   ausgabe = "";
   for result in results:
      ausgabe = ausgabe +"".join(str(result)[2:-3]) +" "
   return jsonify(ausgabe), 200
```

Listing 4.4: HTTP-POST-Request /text

Die Methode gettext() wird daraufhin ausgeführt. Das Json-Objekt wird aus der Anfrage mittels get_json() extrahiert. Im Json-Objekt hinterlegte Name, hier „personid1", wird als name gespeichert. Die If-Abfrage ruft die Methode hatPartner(„personid1") auf. Diese gibt True zurück und somit wird results = Datenbank.getTextmitPartner(„personid1", getPartnerName(„personid1")) ausgeführt. Die Methode getPartnerName(„personid1") gibt „personid2" zurück. Innerhalb des Datenbankskripts wird die Methode aus 4.5 ausgeführt.

```
def getTextmitPartner(nameMY, partnername):
      db = sqlite3.connect(db_path)
      c = db.cursor()
      result = c.execute(f"SELECT text FROM datenbank WHERE name='{nameMY}' OR name =
          '{partnername}'").fetchall()
      db.commit()
      c.close()
      return result
```

Listing 4.5: SQL-Query: Text mit Partner

Dadurch wird die Query „SELECT text FROM datenbank WHERE name= "personid1" OR name = "personid2" ausgeführt und gibt alle Texte (Spalte text), die für die Namen personid1 und personid2 hinterlegt sind, zurück.

Im Webserverskript werden diese Daten als Strings aneinandergereiht und Teile abgeschnitten, die nur Zeichen enthalten, die aus der Abfrage aus der Datenbank resultieren und nicht nur den geschriebenen Text enthalten. Zum Schluss gibt der Server als HTTP-Antwort den aus allen Texten zusammengefügten Text an den Client zurück. Der ebenfalls übermittelte Statuscode 200 zeigt, dass die Übertragung erfolgreich war.

Weitere HTTP-Request des Servers sind: Die Anfrage /sendeFoto hat das Ziel, dass das Foto vom Client übermittelt wird und die Handschrifterkennung dieses verarbeitet. Zum Ablauf: Der Server erhält ein Json-Objekt, welches den Namen des Clients, beziehungsweise seiner ID als String, sowie ein Foto in Form eines Base64-Strings, enthält. Die globale Variable workingonOCR wird auf True gesetzt (Dies spielt für /checkServerStatus eine Rolle). Die globale Variable foto wird ebenfalls initialisiert und auf den Base64 String gesetzt. Aus dem Base64 String wird über mehrere Hilfsmethoden das Bild decodiert und

gedreht. Die Drehung ist nötig, da Android-Geräte das Bild um 90 Grad drehen und die Liste texte wird geleert. Es folgt das Ausführen der Methode overlay_ocr_text (Erläutert wird diese Methode in dem Unterkapitel 4.1.6) mit dem Bild als Parameter. Die vorher geleerte Liste texte wird durchgegangen und alle Strings in der Liste werden im String ausgabe gespeichert. Die Variable workingOnOCR wird auf False gesetzt und „OCR wurde gestartet und ist fertig" wird an den Client zurückgegeben.

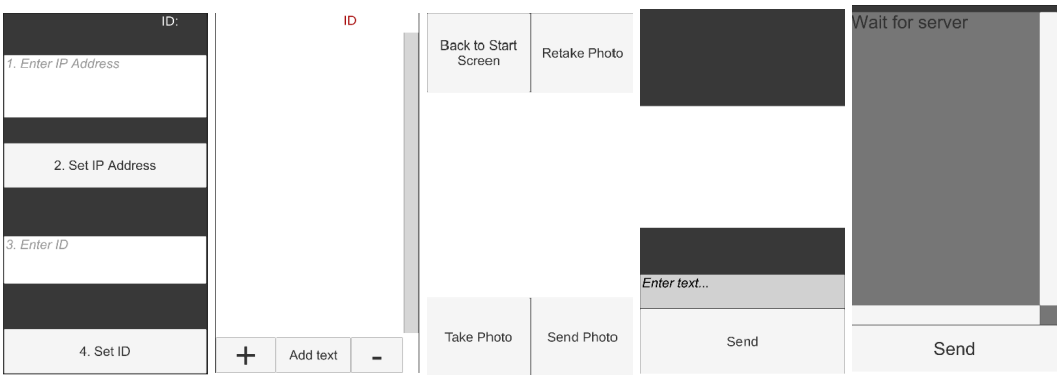
Die Anfrage/checkServerStatus hat das Ziel, den Status des Servers zurückzugeben. Zum Ablauf: Dies ist eine einfache HTTP-GET-Anfrage, bei der keine zusätzlichen Daten übermittelt werden. Hier erfolgt eine einfache If-Else-Abfrage. Wenn workingOnOCR True, also wahr ist, dann wird dem Client „Server beschaeftigt" zurückgegeben, ansonsten „Server frei". Die Variable workingOnOCR ist nur wahr, während in der Methode zu /sendeFoto die Methode overlay_ocr_text ausgeführt wird. Hinweis: Die Dauer des Ausführens der Methode kann je nach Qualität des gesendeten Fotos und den Limitierungen des Servers variieren. Vor dem Senden eines Fotos an den Server sollte der Client vorher den aktuellen Status des Servers abfragen, damit es zu keinen Fehlern kommt, da der Server HTTP-Requests bearbeitet, auch wenn die vorhergehende Bearbeitung noch nicht fertig ist.

Die Anfrage /fotoAnzeigen hat das Ziel, dass das erste Foto aus der Bilderliste zurückgegeben wird. Zum Ablauf: Dies ist ebenfalls eine einfache HTTP-GET-Anfrage. Diese überprüft, wie die vorherige Funktion den Status des Servers anhand der workingOnOCR Variable. Falls workingOnOCR nicht wahr ist, hängt der nächste Schritt von der Liste bilder ab. Falls diese leer ist, dann wird „Alle Bilder verarbeitet" zurückgegeben, ansonsten wird das erste Element (Foto als String) der Liste zwischengespeichert und anschließend gelöscht. Das zwischengespeicherte Element wird an den Client zurückgegeben.

Die Anfrage /getFinalText hat das Ziel, dass der Text ausgegeben wird, der aus OCR und den Einzeleingaben entstanden ist. Zum Ablauf: Diese HTTP-GET-Anfrage setzt den String ausgabe auf „", durchläuft anschließend alle Objekte der Liste texte und fügt alle Strings aneinander. Anschließend wird ausgabe an den Client zurückgeschickt.

Die Anfrage /setFinalText hat das Ziel, dass der finale Text, der übergeben wird, gespeichert wird, sowie die personid und das Foto als Base64 String(FinalerTextSzene). Zum Ablauf: Diese HTTP-POST-Anfrage prüft erst, ob ein Json-Objekt übermittelt wurde und speichert, wenn es eines gibt, den Text des Json-Objekts in der Variable Text. Anschließend wird ein Datenbankobjekt initialisiert. Die id (der Primärschlüssel der Datenbank / der Tabelle er Datenbank) wird ein leeres Objekt (None) gesetzt, die Variable name, foto und die im vorigen Schritt gespeicherte variable Text werden dabei also Variablen zum Initialisieren verwendet. Anschließend wird die Funktion speichern des Datenbankskripts durch die initialisierte Datenbankinstanz ausgeführt und über die Query „INSERT INTO datenbank VALUES(null, name, foto, text" in die Datenbank hinzugefügt. Dem Client wird „Geklappt" zurückgegeben.

Die Anfrage /sendeText hat das Ziel, dass der Text, den der User eingegeben hat (zum übergebenen Foto aus der Anfrage/fotoAnzeigen) an der entsprechenden Stelle gespeichert wird. Zum Ablauf: Bei dieser HTTP-POST-Anfrage prüft der Server erneut, ob ein Json-Objekt übergeben wurde und setzt die Variable text gleich dem Wert, der im Json-Objekt für den Text notiert ist. Anschließend wird die Liste niedrigeProb überprüft. Wenn



(a) Start-Szene **(b)** Lese-Szene **(c)** Foto-Szene **(d)** Einzeleingabe-Szene **(e)** Finaler-Text-Szene

Abbildung 4.2: Die fünf Szenen der App

diese nicht leer ist, wird das erste Objekt mit der Methode print angezeigt. Anschließend wird in der Liste texte, an dem Index, der an der ersten Stelle der Liste niedrigeProb steht, gleich dem übermittelten Text gesetzt. Dem Client wird „Alles geklappt" zurückgegeben.

4.3 App

Die App[10] bildet die Schnittstelle zwischen den ProbandInnen / AnwenderInnen und dem Server. Sie ermöglicht das kollaborative Schreiben. Entwickelt wurde sie mit Unity und ist aktuell auf das Android Betriebssystem zugeschnitten. Hierüber wurde auch die apk (Android Package Kit)[11] gebildet, die als Installationsdatei für Androidgeräte genutzt wird.

Im Entwicklungsprozess wurden fünf Szenen entwickelt, die jeweils ein eigenes Skript beinhalten. Die Szenen lassen sich innerhalb des Assets-Ordners im Ordner Scenes finden und die Skripte im Ordner Scripts. Anhand der Namensgebung der folgenden Unterkapitel lassen sich die jeweiligen Szenen und Skripte finden.

4.3.1 Start Szene

Die App startet in der Start-Szene 4.2a. Hier wird durch den/die AnwenderIn im ersten Eingabefeld die IP-Adresse (inklusive Port) eingegeben und durch das Drücken des Button „2. Set IP-Address" bestätigt. Dies wird statisch als ipaddress hinterlegt, damit von anderen Skripten darauf zugegriffen werden kann. „http://" wird noch vor der IP-Adresse eingefügt. Zudem wird die durch die StudiendurchführerInnen vergebene ID, im Quelltext personID genannt, eingetragen und über den Button „4. Set ID" statisch als personid gespeichert. Danach wird über den Scene-Manager die Lese-Szene geladen.

[10] Dateipfad: MastersthesisOCR/App/
[11] Apk-Datei: /MasterthesisOCR/ZusaetzlicheDatein/App.apk

4.3.2 Lese Szene

Diese Szene 4.2b dient dazu, das bereits Geschriebene des Anwenders, inklusive dem Geschriebenen des/der PartnerIn, wenn vorhanden, anzuzeigen. Das Scrollen durch den Text wurde eingebaut. Beim Start wird die Start()-Methode des Skripts 4.6 ausgeführt.

```
void Start()
  {
     idAnzeige.text = StartBildschirm.personid;
     devices = WebCamTexture.devices;
     WebCamDevice device = default;
     StartCamera(device);
     InvokeRepeating("Aktualisiere", 0f, 10f);
  }
```

Listing 4.6: Start-Methode des Lese-Skripts

Die ID-Anzeige, oben rechts in Rot, wird auf die in der Start-Szene festgelegte ID gesetzt. Die drei darauf folgenden Zeilen sind für die Kamera und deren Start zuständig. Dabei wird der Hintergrund (Raw Image) der Kameratextur gleichgesetzt. Dadurch sieht der/die AnwenderIn das Kamerabild. Die Funktion InvokeRepeating(”Aktualisiere”, 0f, 10f); ruft direkt (0f) die Funktion Aktualisiere() auf und danach wiederkehrend alle zehn Sekunden. Dadurch wird alle zehn Sekunden ein HTTP-POST-Request an die hinterlegte ipaddress mit „/test” gesendet. Dazu wird im Body noch ein Json-Objekt, bestehend aus der personid mit der Bezeichnung „name”, verschickt. Die HTTP-Antwort wird dann über das Kamerabild gelegt, um einen ersten Schritt in Richtung erweiterter Realität zu gewährleisten.

Der „+” Button vergrößtert den angezeigten Text und der „-” Button verkleinert diesen.

Über den „Add text” Button wird zur Foto-Szene gewechselt.

4.3.3 Foto Szene

Die Foto-Szene 4.2c dient dazu, das Geschriebene, welches hinzugefügt werden soll, zu fotografieren und an den Server zu übermitteln. In der Mitte wird das Kamerabild angezeigt. Über den Button „Take Photo” wird das aktuelle Kamerabild temporär gespeichert und angezeigt. Sollte das Foto nicht wie gewollt sein, kann über den Button „Retake Photo” die Foto-Szene neu geladen werden, wonach dann das aktuelle Kamerabild wieder kontinuierlich angezeigt wird . Der „Back to Start” bringt den/die AnwenderIn zurück zur Lese-Szene. Wurde ein Foto gemacht, welches an den Server geschickt werden soll, wird der Button „Send Photo” betätigt. Das Foto muss serialisiert und anschließend konvertiert werden, um an den Server geschickt werden zu können 4.7.

```
byte[] bytes = temp.EncodeToPNG();
string fotoString = Convert.ToBase64String(bytes);
Dictionary<string, string> JZeug = new Dictionary<string, string>()
{
   {"name", StartBildschirm.personid },
   {"foto", fotoString },
   {"text", "" },
};
jsonDaten = JsonConvert.SerializeObject(JZeug);
```

```
StartCoroutine(SendeFotoRequest());
```

Listing 4.7: Umwandlung eines Foto in ein übermittelbares Json-Objekt

Zu diesem Zeitpunkt ist das Foto als Texture2D (temp) gespeichert. Über die Hilfsfunktion der Klasse Texture2D EncodeToPNG() wird das Bild in einen Bytestream bytes kodiert. Dieser wird in einen Base64-String konvertiert. Anhand des Dictionarys wird ein Json-Objekt erstellt. Als name wird wieder die personid eingefügt und unter foto der Base64-String. Der text ist an dieser Stelle nicht relevant. Über JsonConvert.SerializeObject wird das Json-Objekt so umgewandelt, dass es in der aufgerufenen Co-Routine an den Server übermittelt werden kann. Anschließend wird zur Einzeleingabe-Szene gewechselt.

4.3.4 Einzeleingabe Szene

In der Szene 4.2d soll der/die AnwenderIn Fotos von schlecht erkannten Wörtern nacheinander angezeigt bekommen. Im Eingabefeld soll das richtige Wort eingeben werden und über den Button „Send" soll der eingegebene Text an den Server gesendet werden. Der Serverstatus wird über die statische Ip-Adresse ipaddress+„/checkServerStatus" abgefragt. Solange der Server mit der Schrifterkennung beschäftigt ist, wird dem / der AnwenderIn „Wait for Sever" angezeigt. Alle fünf Sekunden wird die genannte HTTP-Anfrage an den Server geschickt. Ist der Server nicht mehr beschäftigt, beginnt das Durchgehen der schlecht erkannten Wörter. Die HTTP-Antwort auf die HTTP-Anfrage „/fotoAnzeigen" enthält entweder einen String, der zu einem Bild dekodiert werden kann, einen Connection Error, die Nachricht, dass der Server beschäftigt sei, oder die Nachricht, dass alle Bilder verarbeitet seien. Bei Letzterem wechselt die Szene in die Finale-Text-Szene. Der/die AnwenderIn gibt für jedes Foto das gemeinte Wort ein. Dieses wird über die TextSenden()-Methode an den Server geschickt und solange ein neues Foto angefragt, bis keins mehr hinterlegt ist.

4.3.5 Finaler Text Szene

In der Szene 4.2e wird der für das gesendete Foto, nach Schrifterkennung und Korrektur in der Einzeleingabe, der gesamte Text angezeigt, um diesen final zu korrigieren. Scrollen ist hier ebenso möglich. Nach der Übermittlung des finalen Textes wechselt die App in die Lese-Szene.

4.4 Nutzung des Softwaresystems

In diesem Kapitel wird vereinfacht erläutert, wie das Softwaresystem und dessen Komponenten untereinander funktionieren.

Die StudiendurchführerInnen tragen im Webserverskript für die personids, beziehungsweise die Teilnehmerkennungen, ein, ob diese einen/eine PartnerIn haben und wenn ja, wie dessen Teilnehmerkennung lautet. Danach wird der Server gestartet.

Ein/eine AnwenderIn startet die App, trägt die IP-Adresse und die Teilnehmerkennung ein und gelangt dadurch in die nächste Szene. Diese Angaben werden folgend genutzt für die Anfragen an den Server.

In der Lese-Szene sehen die AnwenderInnen ihr Kamerabild. Der Client fragt beim Server den Text für die Teilnehmerkennung an. Innerhalb des Servers wird anhand der Teilnehmerkennung und, falls es einen/eine PartnerIn gibt, auch für deren Teilnehmerkennung der gespeicherte Text abfragt und an den Client zurückgesendet. Der übergebene Text wird über das Kamerabild gelegt. Dies wird immer wieder aktualisiert. Falls der/die PartnerIn etwas hinzugefügt hat, wird dieser Text ebenso angezeigt. Soll ein neuer Text hinzugefügt werden, wird über den „Add-Text"-Button zur Foto-Szene gewechselt.

Hier wird wieder das Kamerabild angezeigt und es gibt die Möglichkeit, ein Foto zu machen. Dieses Foto wird an den Server geschickt, die Schrifterkennung analysiert das Bild mittels EasyOCR und erstellt eine Liste mit den Ergebnissen. Diese Ergebnisliste enthält die Position (in Form eines Vierecks) eines Wortes, ist die Wahrscheinlichkeit, dass das Wort richtig erkannt wurde (Wert 0-1), und das Wort selbst. Weiterhin werden alle Texte werden separat in einer Text-Liste gespeichert. Wenn ein Ergebnis eine Wahrscheinlichkeit für ein Wort unter 85 % (<0.85) hat, wird der Index der Ergebnisliste an eine andere Liste, die vor der Schrifterkennung leer war, angehangen. Zusätzlich wird ein Foto des nicht erkannten Wortes in Form eines Strings in einer Bilderliste gespeichert.

In der Einzeleingabe-Szene wird durch eine Server-Anfrage und die erfolgte Antwort das erste Bild aus der Bilderliste angezeigt. Nach Eingabe des richtigen Wortes und Absenden an den Server wird das gesendete Wort an der Stelle eingefügt, an der es auf dem Foto war. Dazu wird die Hilfsliste verwendet, die die Indizes der Ergebnisliste speicherte, die zu korrigieren waren. Sind alle Wörter korrigiert, wechselt die App in die Finale-Text-Szene.

Hier wird der korrigierte Text für eine finale Korrektur angezeigt. Nach dem Absenden des finalen Textes wird dieser in der Datenbank gespeichert und zur Lese-Szene gewechselt. Der eben fotografierte und korrigierte Text wird an den bereits bestehenden Text angehangen.

4.5 Zusätzlicher Quellcode

In diesem Kapitel werden kurz implementierte Methoden genannt, die in der finalen Version dieses Softwaresystems nicht genutzt werden, aber bei einer Weiterentwicklung des Systems nützlich sein könnten. Dafür gibt es ein eigenes Skript [12]. Hier ist die Klasse Datenbank implementiert, die gleich aufgebaut ist, wie die im Datenbankskript (datenbank.py). Die im Datenbankskript hinterlegte Methode erstelleDbObjekt(self) erstellt übergebenen Daten eine Instanz der Datenbank. Wird diese an den Client geschickt, wird diese serialisiert.

[12] Datei: MasterthesisOCR/Assets/Scripts/Nuetzliches.cs

```
var results = JsonConvert.DeserializeObject<Datenbank>(request.downloadHandler.text);
byte[] k = Convert.FromBase64String(results.Foto);
```

Listing 4.8: Datenbankinstanz Deserialisieren

Die Methode in 4.8 kann eine Instanz der Datenbank deserialisieren. Das heißt, der Server kann eine solche Instanz an den Client senden und dieser kann es weiter verarbeiten. Im Quellcode wird ein beispielhafter Nutzen gezeigt, in dem auf den Wert Foto zugegriffen wird.

Die Methode GetIP entstammt einem vorherigen Ansatz, bei welchem keine personID, beziehungsweise Teilnehmerkennung, nötig war, sondern die IP-Adresse genutzt wurde. Diese Funktion gibt die IP-Adresse des Clients als String zurück. Die Methode switchCam kann genutzt werden, um zwischen der vorderen und der hinteren Kamera zu wechseln.

Zudem gibt es ein Benachrichtigungsskript [13]. In diesem Skript wird erst abgefragt, ob die App die Erlaubnis hat, Benachrichtigungen anzuzeigen. Bei neueren Android-Versionen ist dies nicht mehr vonnöten. Die ShowNotification()-Methode zeigt eine vorgefertigte Benachrichtigung an. Innerhalb der Methode lassen sich verschiedenste Parameter der Nachricht (die Wichtigkeit, der Titel, der Text, Icons, Zeitpunkt der Benachrichtigung) einstellen.

4.6 Zusammenfassung und Ausblick

In diesem Kapitel werden die Ergebnisse der Implementierung vorgestellt. Durch sie entsteht ein nutzbares Softwaresystem, welches das kollaborative Schreiben ermöglicht. Unterteilt ist das Softwaresystem in die Komponenten Handschrifterkennung, Server und App, wobei die Handschrifterkennung in den Server inkludiert ist. Es wird der Entstehungsprozess hinter der selbst entwickelten und lauffähigen Handschrifterkennung aufgezeigt, bis hin zur Wahl von EasyOCR und der Einbindung in den Server. Weiterführend werden alle HTTP-Request, seien es Get- oder Post-Anfragen, die der Server nach dem Empfangen verarbeitet, aufgelistet und erläutert. Dazu gehört auch die Interaktion mit der Datenbank. Diese wird über ein weiteres Skript, das ebenfalls zur Komponente Server gehört, initialisiert. Der Server, inklusive Handschrifterkennung und Datenbank, könnte in Zukunft weiterentwickelt werden. Eine Verbesserung der Handschrifterkennung von EasyOCR wird zu einer höheren Anzahl an nicht zu korrigierenden Wörtern führen. Dadurch wird die Effizienz und damit auch ein Teil der Usability deutlich verbessert werden. Ein solches Update ist von EasyOCR geplant [14]. Zum jetzigen Zeitpunkt können parallele, zeitgleiche Interaktionen von verschiedenen AnwenderInnen zu Fehlern führen, insbesondere, wenn ein/eine AnwenderIn einen Interaktionsprozess, wie die Einzeleingabe von Wörtern, nicht abschließt. Das Einführen von Transaktionen bei der Kommunikation mit dem Server ist beim jetzigen Aufbau nicht vonnöten, da keine bereits gespeicherten Daten

[13] Datei: MasterthesisOCR/Assets/Scripts/Notification.cs
[14] https://github.com/JaidedAI/EasyOCR

verändert werden. Für eine mögliche Erweiterung des Softwaresystems, bei der Änderungen von bereits gespeicherten Daten in der Datenbank vonnöten sind, sollten Transaktionen in Betracht gezogen werden. Allgemein kann das Softwaresystem auf der Seite des Servers im Hinblick auf paralleles Arbeiten verbessert werden. In der jetzigen Version des Softwaresystems wird die Datenbank über das Serverskript initialisiert. Im Hinblick auf die Skalierung des Systems sollte dies getrennt werden, unter anderem, weil ein Ansatz gewählt werden könnte, bei dem Server, Handschrifterkennung und Datenbank voneinander getrennt sind. Dafür könnten verschiedene Datenbankmanagementsysteme eingesetzt werden. Mehrfache, parallel ausgeführte Skripte, beispielsweise der Handschrifterkennung, oder das Verwenden von Threads innerhalb eines Handschrifterkennungsskripts würden zu einer niedrigeren Fehlerrate führen. Mit steigender Anzahl an AnwenderInnen sollte ein genügend leistungsstarker Server gewählt werden, auf dem im besten Fall eine CUDA-Nutzung möglich ist, da dies die Performance deutlich verbessert.

Die App-Komponente braucht zum jetzigen Zeitpunkt nicht verändert werden im Hinblick auf paralleles Arbeiten. Die fünf Szenen mit dem jeweiligen Skript werden hier umfassend erläutert, genauso wie das Zusammenspiel zwischen ihnen und dem Server. Zudem wird zusätzlicher Quellcode zur Verfügung gestellt, der eine Erweiterung in zukünftigen Arbeiten erleichtert. Vor einer Erweiterung, beziehungsweise einer Überarbeitung der App, wäre ein Usability-Test in Form eines Fragebogens empfehlenswert. Anhand dieses Tests lässt sich auf die Gebrauchstauglichkeit der App Rückschlüsse ziehen und somit auf mögliche Verbesserungen. Die Fragen sollten dabei die Qualitätskomponenten nach Nielson[15] abdecken und den Schwierigkeitsgrad beim Ausführen von einzelnen Prozessen abfragen. Die System Usability Scale [9], kurz SUS, ist eine häufig verwendete Skala, die aus den Ergebnissen eines Fragebogens gebildet wird [4]. Es ist ein quantitatives und häufig genutztes Mittel, um grundlegende Aussagen über die Gebrauchstauglichkeit eines Softwaresystems zu geben. Um genauere Erkenntnisse über die Usability zu erlangen, benötigt es genauere Fragen, bei denen der Schwierigkeitsgrad für das Ausführen von verschiedenen Aufgaben abgefragt wird. Außerdem können die Befragten direkte Verbesserungsvorschläge und Anmerkungen notieren. Dies müsste dann qualitativ ausgewertet werden. Durch die Implementierung in Unity und deren vorgefertigte Pakete und Kompatibilität zu einigen Augmented-Reality- und Virtual-Reality-Headsets ist ein Umstieg in eine klar definierte erweiterte Realität ohne zu großen Aufwand möglich. Ein möglicher Ansatz ist das Einfügen von Assets, geknüpft an Geschriebenes, im zwei- oder dreidimensionalen Raum. Dies könnte ebenfalls die Motivation steigern. Beispielsweise schreibt der/die AnwenderIn das Wort Wolf und ein Wolf wird neben dem Geschriebenen projiziert. Projektionen, die passend zu einem Wort projiziert wurden, zeigten bereits in einer anderen Studie positive Effekte auf die Motivation [12]. Das Projekt FederLeicht strebt das Anzeigen von zusätzlichen Informationen in einer solchen erweiterten Realität an. Die hier entworfene App bietet sich für genau dieses Anzeigen von zusätzlichen Informationen an. Eine weitere Erweiterung / Veränderung wäre es, die Handschrifterkennung innerhalb der App zu realisieren. Dafür wäre ein zuverlässiges Modell zur Texterkennung und Worterkennung nötig, das mit verschiedensten Datensätzen trainiert wurde. Dadurch würde der Server entlastet werden und ein paralleles Arbeiten / Kommunizieren mit dem Server wäre deutlich weniger fehleranfällig. Zudem würde die Performance steigen, da nicht serialisierte Bil-

[15] Usability nach Nielson: https://www.usability.ch/news/usability-einmaleins.html abgerufen am 8.4.2023

der zwischen Server und AnwenderIn versendet werden würden, sondern direkt der Text. Dadurch sinkt die Datengröße drastisch. Außerdem würde die Einzeleingabe des Textes zu den Bildern, die den schlecht erkannten Text enthalten, rein auf AnwenderIn-Seite stattfinden. Dadurch fiele sehr viel Kommunikation mit dem Server weg. Daraus resultiert dann ein besseres paralleles serverbezogenes Arbeiten. Eine schlechte Internetverbindung mit geringer Datenübertragungsgeschwindigkeit spielt eine weniger große Rolle.

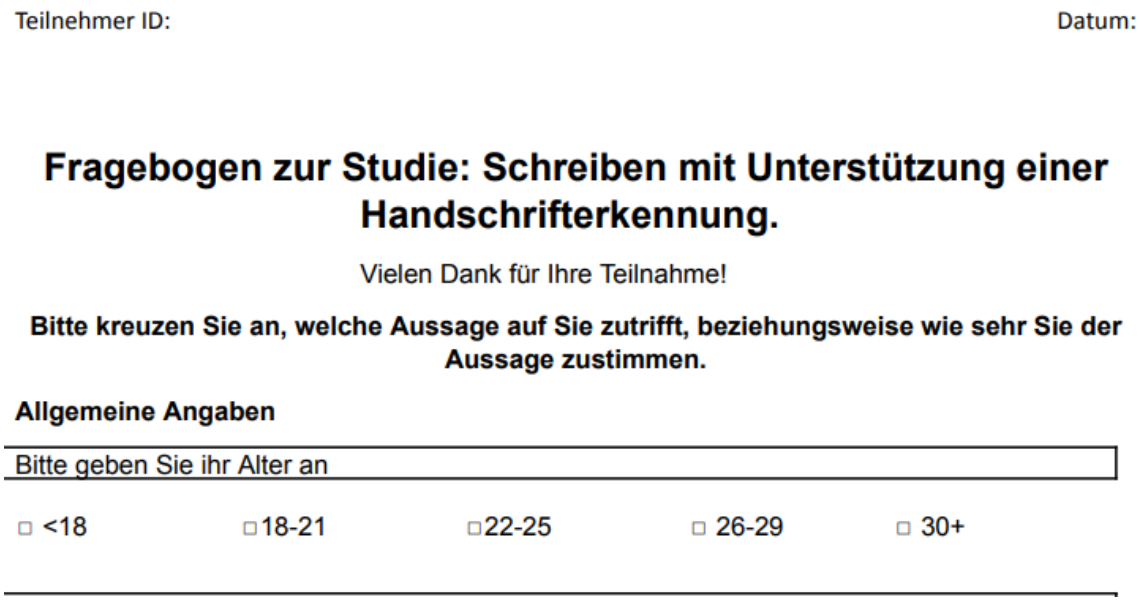
Durch den Einsatz von Unity ist es möglich, die App auch für iOS zu bilden, wodurch es eine größere Menge von potenziellen BenutzerInnen gibt.

5 Evaluation

Das Produkt, beziehungsweise das Softwaresystem der Masterarbeit wird im Anschluss mit einem Studiendesign evaluiert werden, welches im Rahmen der Arbeit entworfen wurde. Ziel der Evaluation wird es sein, zu untersuchen, ob das kollaborative Schreiben zu einer Steigerung der Motivation und darüber auch zu einer Steigerung der Schreibqualität führt. Das soll die Forschungsfrage beantworten: „Erhöht kollaboratives Schreiben in einer erweiterten Realität die Motivation und die Schreibqualität?".

5.1 Studiendesign

Das Design der Studie ist ein Experiment, in dem die Kontrollgruppe alleine und die Experimentalgruppe zu zweit im Wechsel eine Kurzgeschichte in englischer Sprache schreibt. Dies soll nicht innerhalb der Universität geschehen, sondern in Heimarbeit. Ein zeitlicher Rahmen für das Schreiben der Kurzgeschichte wurde nicht festgelegt. Nach Kubbe [24] wird pro Gruppe eine Größe von 30–40 Versuchspersonen angestrebt, wobei dies in der Praxis oftmals aufgrund von Limitierungen geringer ausfällt. Für eine genaue Fallzahlberechnung spielt auch die Effektstärke eine Rolle. Je kleiner die Effektstärke (Unterschied der Messungen der abhängigen Variablen zwischen der Experimental- und Kontrollgruppe) ist, desto mehr Versuchspersonen werden benötigt, um eine Signifikanz nachzuweisen. Zudem steigt die Chance, wenn die Anzahl der Versuchspersonen steigt, dass abweichende Fälle die (Mess-) Ergebnisse nicht (stark) verzerren. Für dieses Studiendesign wurde eine Studiengröße von 40 Versuchspersonen gewählt. Die Versuchspersonen werden nicht wie bei repräsentativen Studien zufällig oder per Quote ausgewählt, sondern es wird innerhalb einer Universität dafür geworben werden, bis vierzig Studierende, wahrscheinlich alle Psychologiestudierende, gefunden sind. Psychologiestudierende deshalb, weil sie sogenannte Versuchspersonenstunden erhalten, die für ihr Studium nötig sind. Deshalb handelt es sich nicht um eine „repräsentative" Stichprobe. Ergebnisse der Studie sollten somit nicht verallgemeinert werden. In dieser experimentellen Studie bilden die Motivation und die Schreibqualität die abhängigen Variablen. Gemessen werden beide am Ende der Studie, nachdem die Kurzgeschichte geschrieben wurde. Um die Schreibqualität zu messen, wird diese operationalisiert. Dies geschieht anhand der Messung von Satzlängen, Satzlängenvarianz, der Nutzung von Adjektiven und Attributen, Wortwiederholungen und Nebensätzen innerhalb eines geschriebenen Textes. Die Motivation wird über einen Fragebogen ermittelt. Hierbei wird nicht, wie bei Thomas und Müller [55] nach den Regulationsstilen der Motivation aufgeschlüsselt. Aber dennoch werden Fragen verwendet, die auf extrinsische Regulation wie den Druck, etwas zu Schreiben, durch Teilnehmen an der Studie erfragt. Die Frage, ob Druck auf einer Person lag, da der/die PartnerIn nicht „enttäuscht" werden soll, fällt weg, da nur die Experimentiergruppe einen/eine PartnerIn hat. Wie im Academic



Teilnehmer ID: Datum:

Fragebogen zur Studie: Schreiben mit Unterstützung einer Handschrifterkennung.

Vielen Dank für Ihre Teilnahme!

Bitte kreuzen Sie an, welche Aussage auf Sie zutrifft, beziehungsweise wie sehr Sie der Aussage zustimmen.

Allgemeine Angaben

Bitte geben Sie ihr Alter an

□ <18 □ 18-21 □ 22-25 □ 26-29 □ 30+

Abbildung 5.1: Ausschnitt des Fragebogens

Self-Regulation Questionnaire [46], kurz SQR-A wird eine vierteilige Skala gewählt von „Very true", also sehr wahr, beziehungsweise deutliche Zustimmung zur vorherigen Aussage bis hin zu „Not at all true", also gar nicht wahr, beziehungsweise deutliche Ablehnung. Andere Fragebögen [13, 45, 55], die ebenfalls die Motivation untersuchen, haben eine fünfteilige Skala, obwohl Fragen aus dem SQR-A teilweise übernommen wurden. In dem Fragebogen von Dengel und Heuer wird das Fachinteresse und die Schulnote mit in die Auswertung einbezogen [13]. In dem hier entworfenen Fragebogen werden als vorläufige Kontrollvariablen einerseits die Abiturnote im Fach Englisch festgelegt und andererseits das wöchentliche Schreiben in der englischen Sprache, inklusive Chatten. Es erfolgt eine quantitative Auswertung / Analyse mithilfe eines Statistikprogramms, wie SPSS oder R. Die Gewichtung der einzelnen Fragen des Fragebogens sowie der Indikatoren, an denen die Schreibqualität bemessen wird, folgt im Nachhinein. Das Erfragen von Verbesserungsvorschlägen und sonstigen Anmerkungen erscheint sinnvoll. Dies erfordert eine individuelle qualitative Auswertung.

Die Abbildung 5.1 zeigt einen Ausschnitt des für die Studie entworfenen Fragebogens. Der vollständige Fragebogen befindet sich in der Dokumentation [1].

5.2 Zusammenfassung und Ausblick

In diesem Kapitel wird ein Studiendesign in Form eines Experiments vorgestellt. Mit 40 Versuchspersonen, gleichmäßig aufgeteilt auf eine Experimentier- und eine Kontrollgruppe, soll das kollaborative Schreiben und dessen Effekte untersucht werden. Ziel ist es,

[1] Datei: MasterthesisOCR/ZusaetzlicheDatein/Fragebogen.pdf

die Forschungsfrage „Erhöht kollaboratives Schreiben in einer erweiterten Realität die Motivation und die Schreibqualität?" zu beantworten. Der Fragebogen ist angepasst an verschiedene Quellen, die sich mit der Untersuchung von Motivation befassen. Der Fragebogen könnte später für den Schuleinsatz erweitert werden. Dort spielt die Gewichtung der Fragen eine größere Rolle. Besonders die extrinsische Motivation wird über vier Regulationen aufgeschlüsselt. Ein Beispiel ist die extrinsische Regulation, in der auch Aspekte wie der Druck von Zuhause (beispielsweise durch die Erziehungsberechtigten), oder die Furcht vor einer möglichen schlechten Note berücksichtigt werden. Die Erläuterungen zur Theorie hinter der Motivation finden sich verschiedene Quellen [21, 55]. Am geeignetsten schien für diesen Fragebogen eine angepasste und übersetzte Version des Academic Self-Regulation Questionaire, kurz SRQ-A zu sein, da es hier um das Erarbeiten der Hausaufgaben geht und das Studiendesign auf ein selbständiges Arbeiten zu Hause abzielt. Fragen, die sich im Zuge einer quantitativen Auswertung / Analyse als unzureichend aussagekräftig erweisen, sollten in zukünftigen Fragebögen gestrichen werden. Allgemein sollte die Auswertung / Analyse von mindestens zwei Personen unabhängig voneinander erfolgen, um eine höhere Objektivität zu gewährleisten. Ein zusätzlicher Fragebogen zur Usability wurde im vorherigen Kapitel erläutert. Anstatt der eigenen Implementierung eines Programmes, welches die Schreibqualität auswertet, kann auch unter Berücksichtigung von Grimes [18] und Khoshnevisan [23] ein bereits fertiges Softwareprodukt genutzt werden. Dafür sollten die aufgelisteten Quellen geprüft werden. Es sollte verglichen werden, welches der Programme die größte Schnittmenge zu der hier gewählten Operationalisierung bietet. In der Beschreibung des Studiendesigns wird erwähnt, dass das Schreiben in der Experimentiergruppe abwechselnd erfolgt. Theoretisch könnte eine Versuchsperson auch mehrere Teile nacheinander hinzufügen. Im Hinblick auf die Motivation und die Schreibqualität wären auch andere Studiendesigns denkbar. Beispielsweise ein Prä-Post-Studiendesign mit einer Gruppe, bei der die Entwicklung von Motivation und Schreibqualität vor dem Verwenden und bei / nach dem Verwenden des Produktes der Masterarbeit untersucht wird. Eine mögliche Erweiterung des Softwaresystems (beispielsweise durch das Senden von Benachrichtigungen und das Speichern von Zeitstempeln für diese und darauf folgende Abgaben) eröffnet neue Möglichkeiten. Zum Beispiel sind anhand einer Zeitmessung Rückschlüsse möglich, ob diese Push-Benachrichtigungen förderlich für die Motivation sind. Im Detail wird ein Vergleich angestellt, ob die Versuchspersonen nach einer Push-Benachrichtigung schneller einen neuen Teil zur Geschichte beitragen als ohne Benachrichtigung.

Es könnte das Softwaresystem mit dem ARCS-Modell [22] und der Ausarbeitung von Zander und Heidig [59], mit denen neue, besonders auch multimediale Lernumgebungen, überprüft werden können und Hinweise auf zu verbesserungswürdige Aspekte gegeben werden, überprüft werden. Sollte beispielsweise die Motivation gegen Ende sinken, sollte die Lernumgebung, aber auch eventuell die Aufgabenstellung angepasst werden.

6 Diskussion

In diesem Kapitel werden verschiedene Aspekte aus vorherigen Kapiteln diskutiert, sowie eine Schlussfolgerung aus dieser Arbeit gezogen, welche vorherige Schlussfolgerungen aufgreift.

6.1 Diskussion einzelner Aspekte aus vorherigen Kapiteln

Dieses Unterkapitel wird genutzt, um einzelne Betrachtungsweisen und Schlussfolgerungen aus den vorherigen Kapiteln zu ergänzen.

6.1.1 Kapitel 2 Hintergrund und verwandte Arbeiten

Wichtig ist, dass sich diese Arbeit nicht anmaßt, alle Studien gefunden zu haben, die verwandt sind mit diesem Thema oder sich gar mit dem gleichen Thema auseinandersetzen. Täglich gibt es neue Arbeiten zu verschiedensten Themata, auch zu denen, die Schnittstellen zu dieser Arbeit bilden. Trotz einer Schlagwortsuche über Google-Scholar[1] IEEE Xplore[2] kann nicht sichergestellt werden, dass alle Arbeiten, die für die Vorarbeit relevant sein könnten, gefunden und mit eingebunden werden können. Daher wird in dieser Arbeit nur die gefundene Literatur betrachtet.

6.1.2 Kapitel 3 Konzept

Während der Konzipierung der Arbeit wurden gleichzeitig erste Implementationsschritte vorgenommen. Deshalb gibt es keine klare Trennung zur Implementierung, wie die Arbeit gegebenenfalls impliziert. Die fortschreitende Planung wurde stark beeinflusst von den Erkenntnissen aus den ersten Implementierungsversuchen. Probleme bei der Implementierung, wie beispielsweise das Einbinden der (Hand-) Schrifterkennung, sorgten für eine Umstrukturierung.

[1] Google Scholar: https://scholar.google.com/ abgerufen am 8.4.2023
[2] IEEE Xplore: https://ieeexplore.ieee.org/Xplore/home.jsp abgerufen am 8.4.2023

6.1.3 Kapitel 4 Implementierung

Die hier entstandenen Probleme wurden im Kapitel genannt. Viele neue Fähigkeiten wurden während der Implementierung gesammelt, da in dieser Arbeit verschiedene Entwicklungsumgebungen und Sprachen genutzt wurden. Schade war es am Ende, dass die Arbeit rund um die eigene Handschrifterkennung abgesehen von den gelernten Fähigkeiten keinen Mehrwert für diese Arbeit hatte und durch EasyOCR ersetzt wurde.

Bei der Implementierung einer Handschrifterkennung ist zu beachten, dass viele (Trainings-)Datensätze in englischer Sprache sind. Die deutsche Sprache enthält Umlaute, die gegebenenfalls nicht oder falsch klassifiziert werden.

6.1.4 Kapitel 5 Evaluation

Hier ist noch einmal anzumerken, dass sich kurz vor Ende der Arbeit gegen eine Tagesstudie mit 5–6 Personen entschieden wurde, da dies keine auswertbaren Ergebnisse geliefert hätte. Dies wäre eher eine Studie gewesen, nur damit eine Studie in dieser Arbeit stattfindet. Stattdessen wurden einige kleine Verbesserungen an der App und dem Server vorgenommen, die auch der wissenschaftlichen Mitarbeiterin der TU Dortmund die spätere Arbeit mit dem Softwaresystem erleichtert. Es ist geplant, dass das Studiendesign übernommen wird für die folgende Studie. Dennoch kann dies nicht garantiert werden, da eine Anpassung an entstehende Limitierungen möglich ist. Der bereits FAM-Fragebogen könnte auch für eine Eingangsbefragung genutzt werden [45].

Bei der Auswahl der VersuchsteilnehmerInnen sollte beachtet werden, dass Schreibschrift deutlich schlechter erkannt wird als Druckschrift.

6.2 Zusammenfassung und Ausblick

In diesem Kapitel werden die Schlussfolgerungen und Ausblicke aus diesem und vorherigen Kapiteln kurz zusammengetragen. Die ausführlichen Schlussfolgerungen lassen sich im jeweiligen Kapitel finden. Die Nützlichkeit der vorherigen Planung eines Softwaresystems wurde dargelegt und im Zuge der Arbeit auch bestätigt. Im Kapitel Implementierung wurden verschiedene Erweiterungsmöglichkeiten des Softwaresystems aufgezeigt. Die Papierprototypen, Meilensteine und Anforderungen ergänzen diese zum Teil. Sollte eine Erweiterung des Softwaresystems in Betracht gezogen werden, so sollte im Zuge des Softwareentwicklungsprozesses eine Planung inklusive der hier erfolgten Planungsaspekte durchgeführt werden. Nicht zu vernachlässigen ist dabei eine weiterführende Literaturrecherche, die neu veröffentlichte verwandte Arbeiten umfasst. Zudem kann es bei Erweiterungen neue Schnittpunkte zu vorher nicht verwandten Arbeiten geben, beispielsweise, wie sich das Empfangen von Benachrichtigungen auf die Motivation der AnwenderInnen auswirkt. Eine Evaluation wird in Form einer Studie in Zukunft stattfinden. Dies übernimmt eine wissenschaftliche Mitarbeiterin der TU Dortmund. Sie erhielt bereits eine Installationsanleitung und eine Einführung in die Funktionsweisen von App und Server. Das Studiendesign der dieser Arbeit folgenden Studie kann sich von dem Studiendesign aus

5.1 unterscheiden. Sollte die Anzahl an VersuchsteilnehmerInnen steigen, sollte das Softwaresystem, besonders im Hinblick auf den Server und die parallele Kommunikation mit diesem angepasst werden. Inwiefern sich die positiven Effekte, die sich sowohl im Bereich kollaboratives Schreiben als auch im Bereich Augmented-Reality-Lernsysteme hier finden lassen, ist nicht gesichert. Anhand der fast durchweg beobachteten positiven Effekte auf die ProbandInnen der Studien ist es aber zu erwarten.

7 Zusammenfassung

In diesem Kapitel wird ein kurzer Durchgang durch diese Arbeit gegeben.

Ziel dieser Arbeit ist die Planung, Entwicklung und Implementierung eines Softwaresystems, mit dem sich kollaboratives Schreiben in einer erweiterten Realität untersuchen lässt. Dies wird grundlegende Vorarbeit für das Projekt FederLeicht leisten. Dafür wird in der Einleitung die Ausgangslage, beziehungsweise die Problemlage genauer erläutert. Es folgt die Zielsetzung und die Forschungsfrage.

Das zweite Kapitel trägt den Namen Hintergrund und verwandte Arbeiten. In diesem wird zuerst der theoretische Hintergrund erläutert. Dafür werden drei Kernbereiche, Deep Learning, kollaboratives Schreiben und Mixed-Reality genauer aufgezeigt.

Es folgt das Aufzeigen von verwandten Arbeiten und die Abgrenzung des eigenen Themas. Es wird anhand verschiedener Arbeiten berichtet, dass die Implementierung einer Handschrifterkennung ein aufwendiges Unterfangen ist mit noch teils niedrigem Erfolg in Bezug Genauigkeit der erkannten Wörter. Besonders schwierig gestaltet sich das Erkennen von Schreibschrift. Weiter werden die Ergebnisse verschiedener Studien aufgezeigt. Die positiven Effekte des kollaborativen Schreibens lassen sich auch beim Schreiben über Cloud-Anwendungen wie Google Docs wiederfinden. Ebenso ergaben Studien, dass der Einsatz von Mixed-Reality Lernsystemen meist einen positiven Einfluss auf den Lernerfolg erzielt. Arbeiten, die das kollaborative Schreiben in Verbindung mit einem Mixed-Reality Lernsystem untersuchen, wurden nicht gefunden.

Im Kapitel Konzept wird das gesamte Konzept für das Softwaresystem detailliert dargestellt. Im ersten Abschnitt wird die Idee hinter der Implementierung erläutert. Im zweiten Teil geht es um das Systemdesign. Dafür werden die verschiedenen Planungsschritte des Softwareentwicklungsprozesses aufgezeigt und erklärt. Dazu gehört eine erste unkonventionelle Modellierung, die den gesamten Ablauf vereinfacht darstellt. Es folgen die im Proposal erfassten Informationen, wie in Kapitel 2 Hintergrund und verwandte Arbeiten. Das Proposal enthält zudem die anfangs erfassten Voraussetzungen und einen groben Zeitplan, dargestellt durch ein Gantt-Diagramm. Für die weitere Planung werden drei Papierprototypten, zwei für die App und einer für die Server GUI, skizziert, wobei diese mit Stift und Papier angefertigt sind. Das Komponentendiagramm, welches den Papierprotoypten folgt, modelliert das ganze System und dessen Abläufe. Eine wichtige Komponente ist die Definition der Meilensteine. Anhand derer wurde gleichzeitig eine Reihenfolge, sowie die Prioritäten für die Implementierung festgelegt.

Die Umsetzung des geplanten Softwaresystems wird in Kapitel Implementierung aufgeschlüsselt. Dabei werden die implementierten Softwarekomponenten einzeln erläutert. Dazu gehört auch die selbst entwickelte und lauffähige Handschrifterkennung, die im finalen Softwaresystem nicht verwendet wird, da die opensource Software EasyOCR in den

Server inkludiert wurde. Ebenso werden der Server, der zusätzlich die Datenbank beinhaltet, und die App dargestellt. Wegen der komplizierteren Struktur der App im Vergleich zu den anderen beiden Komponenten fällt diese Aufschlüsselung umfangreicher aus. Das Zusammenspiel der Komponenten und der Ablauf bei der Bedienung werden ebenfalls aufgezeigt.

Im nächsten Kapitel Evaluation wird die Evaluation thematisiert. Da diese nicht in der Arbeit inkludiert wird, wird darin nur ein vorab definiertes Studiendesign aufgezeigt. Da sich eine Evaluation zur Beantwortung der Forschungsfrage dieser Arbeit anschließen wird, erfolgt ein kurzer Verweis auf verwandte Studiendesigns. Genutzt werden wird für die geplante Evaluation das hier geplante und implementierte Softwaresystem.

Zuletzt wird im Kapitel Diskussion auf Limitierungen innerhalb der Masterarbeit eingegangen und ein Ausblick gegeben, was sich dieser Arbeit anschließen wird. Der wichtigste Aspekt der Limitierung ist dabei, dass sich das Implementieren einer eigenen KI-basierten Handschrifterkennung als zu zeitintensiv erwiesen hat, sowohl, was das Trainieren als auch das Vorbereiten von Datensätzen betrifft. Bereits erprobte Opensource (Hand-)Schrifterkennungen beinhalten in der Regel mehrere Trainingssätze und ein ausgiebig stattgefundenes Training. EasyOCR will explizit Handschrifterkennung unterstützen und kann somit weiterhin genutzt werden. Zudem bietet diese Software nützliche Implementationen wie Bounding Boxen und die Angabe von Wahrscheinlichkeit der Richtigkeit einer Wortvorhersage. Eine Handschrifterkennung wie bei OneDrive sollte ebenfalls in Betracht gezogen werden. Dort erfolgt die Eingabe durch das Schreiben mit einem Stift auf dem Tablet. Eventuell wäre eine Lösung im Projekt FederLeicht, dass beim Schreiben der genutzte Stift mit der Handschrifterkennung interagiert und daraus eine höhere Wahrscheinlichkeit bei der Wortwiedererkennung resultiert. Ebenso könnte die Handschrifterkennung anhand der SchülerInnen stetig weiter trainiert werden. Weitere mögliche Erweiterungen werden im jeweiligen Kapitel thematisiert.

Im Ausblick wird ebenfalls thematisiert, dass eine Evaluation nicht gänzlich wegfällt, sondern in Absprache mit der wissenschaftlichen Mitarbeiterin der TU Dortmund in einem größeren Umfang mithilfe des implementierten Softwaresystems durchgeführt werden wird.

Literaturverzeichnis

[1] Murat Akçayır and Gökçe Akçayır. Advantages and challenges associated with augmented reality for education: A systematic review of the literature. *Educational Research Review*, 20:1–11, February 2017. URL: https://www.sciencedirect.com/science/article/pii/S1747938X16300616, doi:10.1016/j.edurev.2016.11.002.

[2] N. Arica and F.T. Yarman-Vural. Optical character recognition for cursive handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):801–813, June 2002. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. doi:10.1109/TPAMI.2002.1008386.

[3] Ronald T Azuma. A Survey of Augmented Reality.

[4] Aaron Bangor, Philip T. Kortum, and James T. Miller. An Empirical Evaluation of the System Usability Scale. *International Journal of Human–Computer Interaction*, 24(6):574–594, July 2008. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10447310802205776. doi:10.1080/10447310802205776.

[5] João Barreira, Maximino Bessa, Luciana C. Pereira, Telmo Adão, Emanuel Peres, and Luís Magalhães. MOW: Augmented Reality game to learn words in different languages: Case study: Learning English names of animals in elementary school. In *7th Iberian Conference on Information Systems and Technologies (CISTI 2012)*, pages 1–6, June 2012. ISSN: 2166-0735.

[6] Rajesh Shreedhar Bhat. Weights & Biases, November 2021. URL: https://wandb.ai/authors/text-recognition-crnn-ctc/reports/Text-Recognition-With-CRNN-CTC-Network--VmlldzoxNTI5NDI.

[7] Frank Borsch. *Kooperatives Lernen: Theorie - Anwendung - Wirksamkeit*. Lehren und Lernen. Verlag W. Kohlhammer, Stuttgart, 3., aktualisierte auflage edition, 2019.

[8] Thomas M. Breuel, Adnan Ul-Hasan, Mayce Ali Al-Azawi, and Faisal Shafait. High-Performance OCR for Printed English and Fraktur Using LSTM Networks. In *2013 12th International Conference on Document Analysis and Recognition*, pages 683–687, August 2013. ISSN: 2379-2140. doi:10.1109/ICDAR.2013.140.

[9] John Brooke. SUS: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, November 1995.

[10] Keith R. Bujak, Iulian Radu, Richard Catrambone, Blair MacIntyre, Ruby Zheng, and Gary Golubski. A psychological perspective on augmented reality in the mathematics classroom. *Computers & Education*, 68:536–544, October 2013. URL: https://www.sciencedirect.com/science/article/pii/S0360131513000560, doi:10.1016/j.compedu.2013.02.017.

[11] Astrid Buschmann-Göbels and Tushar Chaudhuri. Fachlexikon. URL: https://www.esvcampus.de/ce/fachlexikon-20/search/kollaborativ/target/search/detail.html.

[12] Hsin-Yi Chang, Theerapong Binali, Jyh-Chong Liang, Guo-Li Chiou, Kun-Hung Cheng, Silvia Wen-Yu Lee, and Chin-Chung Tsai. Ten years of augmented reality in education: A meta-analysis of (quasi-) experimental studies to investigate the impact. *Computers & Education*, 191:104641, December 2022. URL: https://www.sciencedirect.com/science/article/pii/S0360131522002123, doi:10.1016/j.compedu.2022.104641.

[13] Andreas Dengel and Ute Heuer. Motivation, Fachinteresse und Schulleistung in Informatik. 2021. ISBN: 9783885797074 Publisher: Gesellschaft für Informatik, Bonn. URL: http://dl.gi.de/handle/20.500.12116/36937, doi:10.18420/INFOS2021_F265.

[14] Duden. kollaborativ - Rechtschreibung, Bedeutung, Definition, Herkunft | Duden. URL: https://www.duden.de/rechtschreibung/kollaborativ.

[15] Bundesministerium für Bildung und Forschung. FederLeicht — Miteinander durch Innovation. URL: https://www.interaktive-technologien.de/projekte/federleicht.

[16] Bundesministerium für Bildung und Forschung. Bekanntmachung - BMBF, April 2020. URL: https://www.bmbf.de/bmbf/shareddocs/bekanntmachungen/de/2020/04/2912_bekanntmachung.html.

[17] Juan Garzón, Juan Pavón, and Silvia Baldiris. Systematic review and meta-analysis of augmented reality in educational settings. *Virtual Reality*, 23(4):447–459, December 2019. URL: http://link.springer.com/10.1007/s10055-019-00379-9, doi:10.1007/s10055-019-00379-9.

[18] Douglas Grimes and Mark Warschauer. Utility in a Fallible Tool: A Multi-Site Case Study of Automated Writing Evaluation. *The Journal of Technology, Learning and Assessment*, 8(6), March 2010. Number: 6. URL: https://ejournals.bc.edu/index.php/jtla/article/view/1625.

[19] Fabrício Herpich, Felipe Becker Nunes, Giani Petri, and Liane Margarida Rockenbach Tarouco. How Mobile Augmented Reality Is Applied in Education? A Systematic Literature Review. *Creative Education*, 10(07):1589–1627, 2019. URL: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/ce.2019.107115, doi:10.4236/ce.2019.107115.

[20] Hannes Kaufmann and Dieter Schmalstieg. Mathematics and geometry education with collaborative augmented reality. In *ACM SIGGRAPH 2002 conference abstracts and applications*, pages 37–41, San Antonio Texas, July 2002. ACM. URL: https://dl.acm.org/doi/10.1145/1242073.1242086, doi:10.1145/1242073.1242086.

[21] John M. Keller. The Arcs Model of Motivational Design. In John M. Keller, editor, *Motivational Design for Learning and Performance: The ARCS Model Approach*, pages 43–74. Springer US, Boston, MA, 2010. doi:10.1007/978-1-4419-1250-3_3.

[22] John M. Keller. ARCS Model of Motivation. In Norbert M. Seel, editor, *Encyclopedia of the Sciences of Learning*, pages 304–305. Springer US, Boston, MA, 2012. doi:10.1007/978-1-4419-1428-6_217.

[23] Babak Khoshnevisan. The Affordances and Constraints of Automatic Writing Evaluation (AWE) Tools: A Case for Grammarly. 2:12–25, December 2019.

[24] Ina Kubbe. Experimente und experimentelle Forschungsdesigns. In Claudius Wagemann, Achim Goerres, and Markus Siewert, editors, *Handbuch Methoden der Politikwissenschaft*, Springer Reference Sozialwissenschaften, pages 1–28. Springer Fachmedien, Wiesbaden, 2018. doi:10.1007/978-3-658-16937-4_7-1.

[25] KMK Kultusministerkonferenz. Lehren und Lernen in der digitalen Welt — Die ergänzende Empfehlung zur Strategie "Bildung in der digitalen Welt". URL: https://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/2021/2021_12_09-Lehren-und-Lernen-Digi.pdf.

[26] Katrin Lehnen. Kooperative Textproduktion : zur gemeinsamen Herstellung wissenschaftlicher Texte im Vergleich von ungeübten, fortgeschrittenen und sehr geübten SchreiberInnen. 2000. URL: https://pub.uni-bielefeld.de/record/2301399.

[27] Shanshan Li, Yang Chen, David M. Whittinghill, and Mihaela Vorvoreanu. A Pilot Study Exploring Augmented Reality to Increase Motivation of Chinese College Students Learning English. pages 24.85.1–24.85.15, June 2014. ISSN: 2153-5965. URL: https://peer.asee.org/a-pilot-study-exploring-augmented-reality-to-increase-motivation-of-chinese-college-students-learning-english.

[28] Tzung-Jin Lin, Henry Been-Lirn Duh, Nai Li, Hung-Yuan Wang, and Chin-Chung Tsai. An investigation of learners' collaborative knowledge construction performances and behavior patterns in an augmented reality simulation system. *Computers & Education*, 68:314–321, October 2013. URL: https://www.sciencedirect.com/science/article/pii/S0360131513001322, doi:10.1016/j.compedu.2013.05.011.

[29] A. Lingnau, H.u. Hoppe, and G. Mannhaupt. Computer supported collaborative writing in an early learning classroom. *Journal of Computer Assisted Learning*, 19(2):186–194, 2003. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1046/j.0266-4909.2003.00019.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1046/j.0266-4909.2003.00019.x, doi:10.1046/j.0266-4909.2003.00019.x.

[30] Cheng-Lin Liu and Ching Y. Suen. A new benchmark on the recognition of handwritten Bangla and Farsi numeral characters. *Pattern Recognition*, 42(12):3287–3295, December 2009. URL: https://www.sciencedirect.com/science/article/pii/S0031320308004457, doi:10.1016/j.patcog.2008.10.007.

[31] Paul Benjamin Lowry, Aaron Mosiah Curtis, and Michelle Rene Lowry. A Taxonomy of Collaborative Writing to Improve Empirical Research, Writing Practice, and Tool Development, 2004. URL: https://papers.ssrn.com/abstract=666141.

[32] Sten 1 Ludvigsen, Nancy 2 Law, Carolyn P. 3 Rose, Gerry 4 1 University of Oslo Stahl, and Pittsburgh HCI Institute. Frameworks for mass collaboration, adaptable scripts, complex systems theory, and collaborative writing. pages 127–131, June 2017. Num Pages: 127-131 Publisher: Springer Nature B.V. URL: https://www.proquest.com/docview/1919693597/citation/77AB92D8A2D24586PQ/1, doi:10.1007/s11412-017-9257-7.

[33] Leuphana Universität Lüneburg. Kollaboratives Lernen. URL: https://www.leuphana.de/lehre/didaktische-konzepte/kollaboratives-lernen.html.

[34] U.-V. Marti and H. Bunke. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, November 2002. doi:10.1007/s100320200071.

[35] Wannisa Matcha and Dayang Rohaya Awang Rambli. Exploratory Study on Collaborative Interaction through the Use of Augmented Reality in Science Learning. *Procedia Computer Science*, 25:144–153, January 2013. URL: https://www.sciencedirect.com/science/article/pii/S1877050913012234, doi:10.1016/j.procs.2013.11.018.

[36] Carole H. McAllister. Collaborative Writing Groups in the College Classroom. In Triantafillia Kostouli, editor, *Writing in Context(s): Textual Practices and Learning Processes in Sociocultural Settings*, Studies in Writing, pages 207–227. Springer US, Boston, MA, 2005. doi:10.1007/0-387-24250-3_10.

[37] Jamshed Memon, Maira Sami, Rizwan Ahmed Khan, and Mueen Uddin. Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR). *IEEE Access*, 8:142642–142668, 2020. Conference Name: IEEE Access. doi:10.1109/ACCESS.2020.3012542.

[38] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies*, 2351, January 1994. doi:10.1117/12.197321.

[39] A.K. Nain and S. Paul. Keras documentation: Handwriting recognition. URL: https://keras.io/examples/vision/handwriting_recognition/.

[40] Judith S. Olson, Dakuo Wang, Gary M. Olson, and Jingwen Zhang. How People Write Together Now: Beginning the Investigation with Advanced Undergraduates in a Project Course. *ACM Transactions on Computer-Human Interaction*, 24(1):4:1–4:40, March 2017. doi:10.1145/3038919.

[41] Vu Phi Ho Pham. The Effects of Collaborative Writing on Students' Writing Fluency: An Efficient Framework for Collaborative Writing. *SAGE Open*, 11(1):2158244021998363, January 2021. Publisher: SAGE Publications. doi:10.1177/2158244021998363.

[42] Ilya Pivavaruk and Jorge Ramón Fonseca Cacho. OCR Enhanced Augmented Reality Indoor Navigation. In *2022 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 186–192, December 2022. ISSN: 2771-7453. doi:10.1109/AIVR56993.2022.00037.

[43] R. Plamondon and S.N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, January 2000. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. doi:10.1109/34.824821.

[44] Anna Carolina Muller Queiroz, Alexandre Moreira Nascimento, Romero Tori, and Maria Isabel da Silva Leme. Immersive Virtual Environments and Learning Assessments. In Dennis Beck, Anasol Peña-Rios, Todd Ogle, Daphne Economou, Markos Mentzelopoulos, Leonel Morgado, Christian Eckhardt, Johanna Pirker, Roxane Koitz-Hristov, Jonathon Richter, Christian Gütl, and Michael Gardner, editors, *Immersive Learning Research Network*, Communications in Computer and Information Science, pages 172–181, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-23089-0_13.

[45] Falko Rheinberg, Regina Vollmeyer, and Bruce D Burns. FAM: Ein Fragebogen zur Erfassung aktueller Motivation in Lern- und Leistungssituationen12 (Langversion, 2001).

[46] R. M. Ryan and J. P. Connell. Perceived locus of causality and internalization: examining reasons for acting in two domains. *Journal of Personality and Social Psychology*, 57(5):749–761, November 1989. doi:10.1037//0022-3514.57.5.749.

[47] Harald Scheidl, Stefan Fiel, and Robert Sablatnig. Word Beam Search: A Connectionist Temporal Classification Decoding Algorithm. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 253–258, Niagara Falls, NY, USA, August 2018. IEEE. URL: https://ieeexplore.ieee.org/document/8583770/, doi:10.1109/ICFHR-2018.2018.00052.

[48] Ali Shehadeh. Effects and student perceptions of collaborative writing in L2. *Journal of Second Language Writing - J SECOND LANG WRIT*, 20:286–305, December 2011. doi:10.1016/j.jslw.2011.05.010.

[49] Vilaythong Southavilay, Kalina Yacef, and Rafael Calvo. Process Mining to Support Students' Collaborative Writing. pages 257–266, October 2010.

[50] Vilaythong Southavilay, Kalina Yacef, and Rafael Calvo. Process Mining to Support Students' Collaborative Writing. pages 257–266, October 2010.

[51] Neomy Storch. Collaborative writing. *Language Teaching*, 52(1):40–59, January 2019. Publisher: Cambridge University Press. URL: https://www.cambridge.org/core/journals/language-teaching/article/collaborative-writing/142B6E59CED32789A826D43923869EAB, doi:10.1017/S0261444818000320.

[52] Jannis Strecker, Kimberly García, Kenan Bektaş, Simon Mayer, and Ganesh Ramanathan. SOCRAR: Semantic OCR through Augmented Reality. In *Proceedings of the 12th International Conference on the Internet of Things*, IoT '22, pages 25–32, New York, NY, USA, January 2023. Association for Computing Machinery. URL: https://dl.acm.org/doi/10.1145/3567445.3567453, doi:10.1145/3567445.3567453.

[53] Tetyana Sydorenko, John Hellermann, Steven L. Thorne, and Vanessa Howe. Mobile Augmented Reality and Language-Related Episodes. *TESOL Quarterly*, 53(3):712–740, 2019. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/tesq.507. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/tesq.507, doi:10.1002/tesq.507.

[54] Lamma Tatwany and Henda Chorfi Ouertani. A review on using augmented reality in text translation. In *2017 6th International Conference on Information and Communication Technology and Accessibility (ICTA)*, pages 1–6, December 2017. ISSN: 2379-4402. doi:10.1109/ICTA.2017.8336044.

[55] Almut E. Thomas and F. Müller. Skalen zur motivationalen Regulation beim Lernen von Schülerinnen und Schülern. 2011. URL: https://www.semanticscholar.org/paper/Skalen-zur-motivationalen-Regulation-beim-Lernen-Thomas-M%C3%BCller/00ed7a175917aa68fc86aeccb88efbbb315400b3.

[56] Yi-Hsuan Wang. Exploring the effectiveness of integrating augmented reality-based materials to support writing activities. *Computers & Education*, 113:162–176, October 2017. URL: https://www.sciencedirect.com/science/article/pii/S0360131517300969, doi:10.1016/j.compedu.2017.04.013.

[57] Mary Webb, Megan Tracey, William Harwin, Ozan Tokatli, Faustina Hwang, Ros Johnson, Natasha Barrett, and Chris Jones. Design Considerations for Haptic-Enabled Virtual Reality Simulation for Interactive Learning of Nanoscale Science in Schools. In Dennis Beck, Anasol Peña-Rios, Todd Ogle, Daphne Economou, Markos Mentzelopoulos, Leonel Morgado, Christian Eckhardt, Johanna Pirker, Roxane Koitz-Hristov, Jonathon Richter, Christian Gütl, and Michael Gardner, editors, *Immersive Learning Research Network*, Communications in Computer and Information Science, pages 56–67, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-23089-0_5.

[58] Soobin Yim, Dakuo Wang, Judith Olson, Viet Vu, and Mark Warschauer. Synchronous Collaborative Writing in the Classroom: Undergraduates' Collaboration Practices and their Impact on Writing Style, Quality, and Quantity. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, pages 468–479, New York, NY, USA, February 2017. Association for Computing Machinery. URL: https://dl.acm.org/doi/10.1145/2998181.2998356, doi:10.1145/2998181.2998356.

[59] Steffi Zander and Steffi Heidig. Motivationsdesign bei der Konzeption multimedialer Lernumgebungen. In Helmut Niegemann and Armin Weinberger, editors, *Handbuch Bildungstechnologie: Konzeption und Einsatz digitaler Lernumgebungen*, pages 393–415. Springer, Berlin, Heidelberg, 2020. doi:10.1007/978-3-662-54368-9_37.

Alle URLs wurden zuletzt am 08. 04. 2023 geprüft.

Eidesstattliche Erklärung

Ich versichere an Eides statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen haben, als solche kenntlich gemacht haben, mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient haben. Die Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Ort, Datum, Unterschrift