

TECHNISCHE UNIVERSITÄT BERLIN

MASTERARBEIT

**Weiterentwicklung eines AOI Editors zur
Erstellung dynamischer Areas of Interest
in 360° Videos in Unity**

Verfasser:

Kilian Justus Severin
SANCHEZ HOLGUIN
Matrikelnr.: 401229

Gutachter:

Prof. Dr.-Ing M. RÖTTING
M.Sc. S.-C. FREYTAG

Zur Erlangung des Grades M.Sc. Human Factors

Institut für Psychologie und Arbeitswissenschaft
Fachgebiet Mensch-Maschine-Systeme

24. August 2023

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 24. August 2023

K. Sanchez Holguin

Unterschrift

Zusammenfassung

Diese Masterarbeit beschäftigt sich mit der Weiterentwicklung und Validierung einer Software zur Vorbereitung und Durchführung von Eyetracking-Experimenten in Virtual Reality (VR) mit 360° Videos. Das Programm ermöglicht es, ein beliebiges 360° Video mit dynamischen Areas of Interest (AOIs) zu versehen. Im Anschluss können Versuchspersonen das Video in VR unter Verwendung einer HTC Vive Pro Eye VR-Brille ansehen, woraufhin eine Auswertung relevanter Eyetracking-Parameter bezüglich der zuvor definierten AOIs bereitgestellt wird. Die Validierung der Software erfolgte durch Versuche, bei denen die Blickbewegungen von sechs Teilnehmer*innen aufgezeichnet, manuell codiert und anschließend mit dem Datenexport der Software abgeglichen wurden. Zusätzlich wurde die Fixationsdetektion der Software mit einem bestehenden Softwarepaket verglichen. Die Ergebnisse zeigen, dass eine hohe Übereinstimmung zwischen der Software, dem manuellen Rating sowie dem zum Vergleich herangezogenen Softwarepaket besteht, was auf das Potenzial der Software als effektives Erhebungsinstrument hinweist.

This master's thesis presents the development and validation of a software tool for preparing and conducting eyetracking experiments in virtual reality (VR) with 360° videos. The program allows for the annotation of dynamic Areas of Interest (AOIs) on any 360° video. Subsequently, participants can view the video in VR using an HTC Vive Pro Eye VR headset, and relevant eyetracking parameters related to the predefined AOIs are analyzed and provided. The software validation was carried out through experiments in which the gaze movements of six participants were recorded, manually coded, and then compared with the data export from the software. Additionally, the software's fixation detection was compared with an existing software package. The results demonstrate a high level of agreement between the software, manual ratings, and the comparison software package, indicating the software's potential as an effective data collection tool.

Inhaltsverzeichnis

Eidesstattliche Erklärung	iii
Zusammenfassung	v
1 Einleitung	1
2 Problemstellung	5
3 Theoretische Grundlagen	7
3.1 Visuelles System	7
3.1.1 Sehschärfe	8
3.1.2 Blickbewegungen	9
3.1.3 Akkommodation-Konvergenz Reflex	11
3.1.4 Visuelle Aufmerksamkeit	11
3.2 Eyetracking	14
3.2.1 Funktionsweise	14
3.2.2 Areas of Interest	15
3.2.3 Parameter	16
3.3 Head Mounted Displays und virtuelle Realität	19
3.3.1 Funktionsweise	19
3.3.2 Immersion und Präsenz in VR	20
3.3.3 Limitationen	21
3.4 360° Videos	24
4 Verwandte Arbeiten	27
4.1 Open-Source-Software	27
4.2 Kommerzielle Software	29
4.3 Bestehende Programmgrundlage	30
5 Entwicklung	33
5.1 Anforderungen	33
5.1.1 Abgrenzung zu verwandten Arbeiten	33
5.1.2 Angestrebte Verbesserungen gegenüber dem Prototyp	34
5.1.3 Funktionalitäten außerhalb des Projektumfangs	35
5.1.4 Vorgehensweise	36

5.2 Vorstellung der entwickelten Software	37
5.2.1 Funktionsumfang	37
5.2.1.1 Projektmanagement	37
5.2.1.2 AOI Erstellung und Management	38
5.2.1.3 Videowiedergabe und AOI Animation	40
5.2.1.4 Versuchsvorbereitung	45
5.2.1.5 Versuchsdurchführung	48
5.2.1.6 Datenexport	48
5.2.1.7 Fehlerbenachrichtigungen	49
5.2.1.8 Offline Analyse	50
5.2.2 Programmlogik	52
5.2.2.1 360° Video Import und Playback	52
5.2.2.2 Interne AOI Repräsentation	54
5.2.2.3 Animation und Interpolation	58
5.2.2.4 Eyetracking und Event Detection	58
5.2.2.5 Visualisierung	62
6 Validierung	65
6.1 Methodik	65
6.1.1 Simulation	65
6.1.2 Versuche	65
6.1.2.1 Stichprobe	65
6.1.2.2 Versuchsaufbau und Instruktionen	66
6.1.3 Vergleich zu bestehendem Softwarepaket	68
6.2 Hard- und Software	68
6.2.1 Software	68
6.2.2 Hardware	69
6.3 Ergebnisse	70
6.3.1 Ergebnisse der Versuche	70
6.3.2 Ergebnisse des Vergleichs mit dem vr-idt Python-Paket	72
7 Diskussion	75
7.1 Limitationen	77
7.1.1 Limitationen der Validierung	77
7.1.2 Limitationen der Software und Verbesserungspotenzial	79
8 Ausblick	83
A Installationsanweisungen	85
A.1 Ausführung ohne Unity	85
A.1.1 Steam und SteamVR	85
A.1.2 Vive Pro HMD Setup	85
A.1.3 SRanipal und Tobii VRU02 Runtime	85

A.1.4 Python und Bibliotheken	86
A.1.5 AOI Editor	87
A.2 Setup des Unity Projekts	87
A.3 Hardwarevoraussetzungen	87
B Unity Projekt Aufbau	89
B.1 Gameobjects in Szene	89
B.2 Assets	91
B.3 Skripte	93
C HTC Vive Pro Eye Spezifikationen	95
D Übersicht eingesetzter Software und Plugins	97
E Studienergebnisse	99
E.1 Versuchsperson 1	99
E.2 Versuchsperson 2	101
E.3 Versuchsperson 3	103
E.4 Versuchsperson 4	105
E.5 Versuchsperson 5	107
E.6 Versuchsperson 6	109
Literatur	111

Abbildungsverzeichnis

3.1	Verteilung der Stäbchen (Rods) und Zapfen (Cones) auf der Retina	8
3.2	Visualisierung abnehmender Sehschärfe bei zunehmendem visuellem Winkel	9
3.3	Relative Position der Pupille und Cornealreflexion zueinander bei 9-Punkt-Kalibrierung	15
3.4	Schematischer Aufbau einer VR-Brille (Oculus Rift)	20
3.5	Nichtübereinstimmung von Akkommodation und Konvergenz bei der Betrachtung eines Objekts in virtual Reality	22
3.6	Erstellung und Wiedergabe eines 360° Videos	24
3.7	Scanpath auf Rektangularprojektion einer 360° Umgebung	25
4.1	Benutzeroberfläche der prototypischen Programmgrundlage	30
5.1	Erste Schritte nach Ausführung des Programms	37
5.2	Benutzeroberfläche des Programms nach Anlegen eines neuen Projekts	39
5.3	Annotierte Benutzeroberfläche der Software	40
5.4	Interaktionen zur Anpassung von AOIs	41
5.5	Skalierte Timeline („gezoomt“)	42
5.6	Rechtsklick-Kontextmenü von AOIs	44
5.7	Die beiden Untermenüs des Programms	45
5.8	Testszene aus Sicht der Versuchsleitung	46
5.9	Beispiel CSV-Datenexport	49
5.10	Beispiel Sequence Chart	49
5.11	Log-Konsole des Programms	50
5.12	Ablaufdiagramm der Videoladesequenz	53
5.13	AOI-Gameobjects in Unity's Scene-Ansicht	55
5.14	Virtuelle Kamera mit Blick auf zwei AOIs	56
5.15	Ablaufdiagramm der Datensammlung	60
6.1	AOIs für Versuch	67
6.2	Verteilung der absoluten Abweichungen zwischen Datenexport und manuellen Rating	72
A.1	Aktivierung des Eyetrackings in SteamVR	86
B.1	Übersicht über die wichtigsten Skripte	93

Tabellenverzeichnis

6.1	Mittlerer absoluter Fehler zwischen CSV-Datenexport und manuellem Coding	71
6.2	Einstellungen der Algorithmen	73
6.3	Prozentuale Übereinstimmung der Klassifikation aller Data-Samples mit dem vr-idt Paket	73
A.1	Minimale Hardwarevoraussetzungen	88
B.1	Beschreibung der Spielobjekte der Unity Szene des Projekts.	89
B.3	Übersicht über Assets	92
C.1	Spezifikationen der HTC Vive Pro Eye	96
D.1	Software und Plugins	97
E.1	VP1: Ergebnisse des manuellen Codings	99
E.2	VP1: Werte des Datenexports	100
E.3	VP1: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding	100
E.4	VP2: Ergebnisse des manuellen Codings	101
E.5	VP2: Werte des Datenexports	101
E.6	VP2: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding	102
E.7	VP3: Ergebnisse des manuellen Codings	103
E.8	VP3: Werte des Datenexports	103
E.9	VP3: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding	104
E.10	VP4: Ergebnisse des manuellen Codings	105
E.11	VP4: Werte des Datenexports	105
E.12	VP4: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding	106
E.13	VP5: Ergebnisse des manuellen Codings	107
E.14	VP5: Werte des Datenexports	108
E.15	VP5: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding	108
E.16	VP6: Ergebnisse des manuellen Codings	109

E.17 VP6: Werte des Datenexports	109
E.18 VP6: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding	110

Abkürzungsverzeichnis

AOI	Area of Interest
CSV	Comma separated Values
FPS	Frames per Second
HMD	Head Mounted Display
IDE	Integrated Development Environment
I-DT	Identification Dispersion Threshold
JSON	Java Script Object Notation
MAE	Mean Absolute Error
px	Pixel
SDK	Software Development Kit
UI	User Interface
VR	Virtual Reality

Kapitel 1

Einleitung

Die Untersuchung der visuellen Aufmerksamkeit von Menschen beschäftigt Forschende bereits seit den Anfängen der Psychologie und Kognitionsforschung. Wissen darüber, wie Menschen ihre Aufmerksamkeit in der Umwelt verteilen, um sich z. B. zu orientieren (Kiefer et al., 2017) oder Multitasking-Situationen zu bewältigen (Marchewka et al., 2020), eröffnet Möglichkeiten in einer Vielzahl interessanter Forschungs- und Anwendungsbereiche. Erkenntnisse aus der Aufmerksamkeitsforschung erlauben uns, unsere Umwelt, Systeme und Objekte menschengerechter und besser zu gestalten.

Die Herausforderung besteht darin, zu ermitteln, worauf genau die visuelle Aufmerksamkeit einer Person zu einem gegebenen Zeitpunkt gerichtet ist. Da es sich bei der Ausrichtung von Aufmerksamkeit um einen kognitiven Prozess handelt, ist dieser nicht unmittelbar messbar. Dennoch existieren physiologische Maße, die Aufschlüsse über den Fokus visueller Aufmerksamkeitsprozesse liefern können.

Bereits im 19. Jahrhundert unternahmen Wissenschaftler*innen erste Experimente, in welchen die Blickrichtung von Versuchspersonen anhand der Bewegung ihrer Augen ermittelt wurde (Javal, 1878). Die Weiterentwicklung dieses Vorgehens brachte sogenannte Eyetracker zutage. Dabei handelt es sich um technische Systeme, die es erlauben, die Position und Bewegung des menschlichen Auges zu messen und daraus die Blickrichtung eines Menschen zu bestimmen. Unter der Annahme, dass Menschen den Reiz verarbeiten, auf den ihr Blick gerichtet ist, lassen sich aus Blickbewegungsdaten Erkenntnisse über die visuelle Aufmerksamkeit ableiten.

Eyetracker sind in vielen Forschungsbereichen weitverbreitet. Speziell in der Human Factors Forschung ermöglichen Blickbewegungsexperimente die Analyse visueller Aufmerksamkeit hinsichtlich der Interaktion mit Benutzeroberflächen und technischen Systemen. Eine umfangreiche Auflistung verschiedener Anwendungsbereiche für Eyetracker liefert Duchowski (2002).

In Eyetracking-Studien kann der zu untersuchende Stimulus prinzipiell auf zwei Arten präsentiert werden. Entweder ist der Stimulus digital auf einem Bildschirm

zu sehen (z. B. Prototypen der Benutzeroberfläche eines Systems, Werbematerial, Fotos, Videos etc.), oder es werden die Blickbewegungen von Versuchspersonen bei der Betrachtung von Stimuli in einer realen Umgebung erfasst (z. B. in einem Supermarkt).

Ersteres hat den Vorteil, dass nahezu beliebige Stimuli kontrolliert präsentiert werden können. Dies erlaubt Versuche konsistent und mit vergleichsweise geringem Aufwand durchzuführen, birgt jedoch den Nachteil, dass je nach Art des Stimulus wenig ökologische Validität gegeben ist, da die Präsentation des Materials auf einen Bildschirm begrenzt ist (Rossetti & Hurtubia, 2020).

Im Gegensatz dazu verleitet die Durchführung von Eyetracking-Experimenten in realen Umgebungen zu natürlicherem Verhalten von Versuchspersonen, sodass eine höhere ökologische Validität gegeben ist. Jedoch ist es wesentlich komplexer, vollständige Kontrolle über solch eine reale Umgebung beizubehalten (Meißner et al., 2019). Verschiedenen Teilnehmenden eines Experiments eine identische Umgebung zu präsentieren, stellt bei der Forschung um Feld eine große Herausforderung dar. Weiterhin können nur Szenarien und Stimuli präsentiert werden, zu denen Forschende und Versuchspersonen physischen Zugang haben. Deshalb können fiktive, riskante oder schwer zugängliche Reize und Umgebungen auf diese Art und Weise nicht untersucht werden (Pan & Hamilton, Antonia F. de C., 2018). Folglich müssen Forschende bei der Gestaltung von Eyetracking-Experimenten und der Auswahl von Stimuli stets zwischen den zwei dargestellten Optionen abwägen.

Eine technische Neuerung, die es ermöglicht, einen Mittelweg zwischen den beiden genannten Ansätzen zu verfolgen, sind sogenannte Head Mounted Displays (HMDs), welche die Darbietung einer virtuellen Realität (Virtual Reality, VR), ermöglichen. VR-Brillen mit integriertem Eyetracker erlauben es, nahezu beliebige Stimuli immersiv und realitätsgetreu abzubilden, wobei Forschende die vollständige Kontrolle über die dargebotene Umgebung behalten (Meißner et al., 2019; Pan & Hamilton, Antonia F. de C., 2018). Die Kombination aus ökologischer Validität und Kontrolle macht die Technologie für Forschungszwecke hinsichtlich visueller Aufmerksamkeit sehr interessant.

Eine virtuelle Umgebung kann auf zwei Arten erstellt werden. Sie kann entweder in 3D-Software modelliert oder durch die Wiedergabe eines 360° Videos abgebildet werden. Ersteres erfordert Kenntnisse einer 3D-Software, und die Modellierung kann sich sehr zeitintensiv gestalten. Diese Vorgehensweise bietet jedoch den Vorteil, dass, sofern ausreichend Zeit und Softwarekenntnisse vorhanden sind, nahezu jedes beliebige Szenario kreiert werden kann. 360° Videos hingegen erlauben es, ohne Kenntnisse komplexer 3D-Software immersive Umgebungen in VR zu erzeugen. Mit einer entsprechenden 360° Kamera können reale Umgebungen gefilmt und anschließend in VR wiedergegeben werden. Dieser Ansatz zur Erstellung virtueller

Umgebungen ist wesentlich simpler als die 3D-Modellierung und erfordert keine besonderen Softwarekenntnisse. Die Kombination aus Head Mounted Displays und 360° Videos birgt das Potenzial, zukünftig kostengünstig realitätsgetreue und immersive virtuelle Umgebungen darbieten zu können und dabei volle Kontrolle über den Stimulus zu behalten. Da die Technologien jedoch vergleichsweise neu sind, gibt es nur wenige Softwarelösungen, die es ermöglichen, Eyetracking-Experimente mit 360° Videos durchzuführen. Ziel der Arbeit ist deshalb die Entwicklung einer Open-Source-Software zu Gestaltung und Durchführung von Eyetracking-Experimenten mit 360° Videos in Virtual Reality.

Kapitel 2

Problemstellung

Die vorgestellte Arbeit erfolgt am Fachgebiet Mensch-Maschine-Systeme an der Technischen Universität Berlin. Das Fachgebiet verfügt über eine VR-Brille mit integriertem Eyetracker (HTC Vive Pro Eye), die für die Durchführung von Eyetracking-Experimenten verwendet wird. Um in Zukunft Versuche in virtueller Realität mit 360° Videos durchführen zu können, wurden bereits erste Bemühungen unternommen, eine prototypische Software zu diesem Zweck zu gestalten. Die Arbeit von Schicks (2022) zeigt, dass mit ausreichend Entwicklungsaufwand eine Implementierung einer solchen Software mit der Videospielentwicklungsplattform Unity möglich ist. Ziel dieser Arbeit ist, die von Schicks entwickelte prototypische Programmgrundlage zu einer am Fachgebiet nutzbaren Software weiterzuentwickeln.

Das Programm soll es Forschenden erlauben, ein beliebiges 360° Video mit dynamischen Areas of Interest (AOIs) zu versehen. Im Anschluss sollen Versuchspersonen das Video in VR ansehen können, woraufhin eine Auswertung relevanter Eyetracking-Parameter bezüglich der zuvor definierten AOIs bereitgestellt werden soll.

Dafür werden in Kapitel 3 zunächst theoretische Grundlagen erläutert, die für die Arbeit mit Eyetracking-Systemen, Areas of Interest und virtuellen Umgebungen hilfreich sind. Im Anschluss werden in Kapitel 4 verwandte Arbeiten aus dem aktuellen Stand der Forschung sowie die bereits bestehende prototypische Programmgrundlage vorgestellt. Kapitel 5 schildert die Entwicklung der neu programmierten Software, ihren Funktionsumfang sowie ihre Limitationen. Im darauffolgenden Kapitel 6 werden die Schritte, die zur Validierung der Software getätigten wurden, aufgezeigt und die Ergebnisse präsentiert. Kapitel 7 diskutiert die Ergebnisse der Validierung und die sich daraus ergebenden Implikationen. Abschließend gibt Kapitel 8 einen Ausblick auf zukünftiges Forschungs- und Entwicklungspotenzial.

Kapitel 3

Theoretische Grundlagen

In diesem Kapitel werden die für das Projekt relevanten Grundlagen erläutert. Es werden theoretische Konzepte eingeführt, die verwendeten Technologien im Einzelnen näher erläutert sowie Implikationen, die sich aus dem Stand der Technik ergeben, aufgeführt.

3.1 Visuelles System

Ein grundlegendes Verständnis über die physiologischen Mechanismen der visuellen Wahrnehmung hilft dabei, die Funktionsweise von Eyetracking-Systemen zu verstehen. In dieser Sektion wird lediglich eine kurze Einführung zu relevanten Konzepten präsentiert. Für einen detaillierten Überblick über das menschliche visuelle System siehe Tovée (2009).

Der Begriff des visuellen Systems umfasst alle physiologischen Komponenten, die Sehen ermöglichen. Dazu gehören das Auge als sensorisches Organ, die Augenmuskeln, welche Blickbewegungen erzeugen sowie Teile des Zentralen Nervensystems wie der Sehnerv, die Sehbahn und der visuelle Kortex. Das visuelle System ist die physiologische Basis für die visuelle Wahrnehmung des Menschen (Tovée, 2009).

Für das Verständnis der in dieser Arbeit behandelten Themen sind folgende Eigenschaften des visuellen Systems, insbesondere der Augen, relevant:

- Die Sehschärfe ist am höchsten im Zentrum des Blickfeldes.
- Deshalb sind unsere Augen ständig in Bewegung, um neue Objekte in das Zentrum des visuellen Feldes zu rücken.
- Um ein Objekt räumlich und scharf wahrzunehmen, richten sich unsere Augen durch eine gegensätzliche Bewegung (Konvergenz) auf das Objekt aus, während sich die Brechkraft der Augenlinsen zeitgleich entsprechend der Distanz des Objekts anpasst (Akkommodation).

Die drei genannten Aspekte haben allesamt Implikationen für den Einsatz von Head Mounted Displays und Eyetracking-Systemen, weshalb sie jeweils kurz näher erläutert werden.

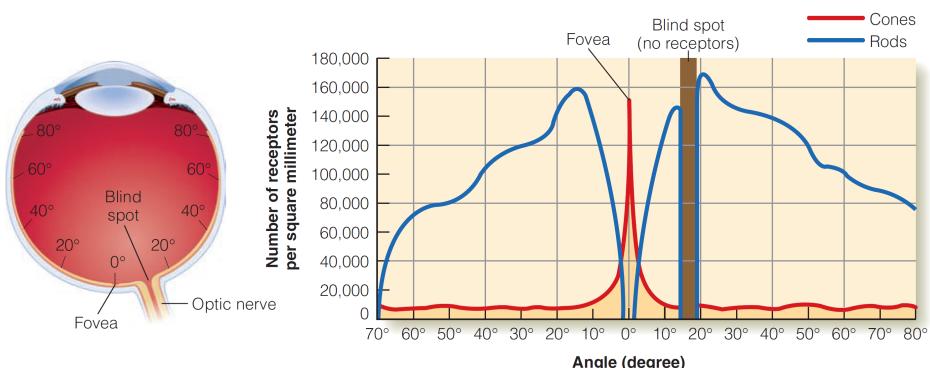
3.1.1 Sehschärfe

Mit beiden Augen gerade nach vorn gerichtet, ohne die Augen oder den Kopf zu bewegen, umfasst das menschliche Blickfeld horizontal etwa 190 Grad (Howard & Rogers, 1995). Das menschliche Auge nimmt Informationen aus dem gesamten visuellen Feld auf, jedoch unterscheidet sich die Wahrnehmung sehr stark zwischen dem zentralen und peripheren Sehen. Die Ursache dieses Phänomens befindet sich in der Verteilung der Fotorezeptoren auf der Netzhaut des Auges.

Licht fällt aus der Umgebung auf die Augenlinse und wird von dieser auf die Netzhaut (lat. Retina) projiziert. Licht, das auf die Retina fällt, initiiert eine Kaskade chemischer und elektrischer Vorgänge, die letztlich eine Weiterleitung von Nervenimpulsen über den Sehnerv an den visuellen Kortex auslöst. Nach Goldstein (2010) besteht die Netzhaut aus zwei Arten lichtempfindlicher Fotorezeptoren, sog. Stäbchen und Zapfen. Das Auge enthält ca. 120 Mio. Stäbchen, welche primär für die Wahrnehmung von Helligkeit zuständig sind. Etwa 6–7 Mio. Zapfen hingegen reagieren auf Licht kurzer, mittlerer oder langer Wellenlänge. Die Integration der Signale unterschiedlicher Zapfenarten ermöglicht das Farbensehen sowie die Wahrnehmung optischer Details, sofern dafür ausreichend Helligkeit gegeben ist. Die ungleiche Verteilung von Stäbchen und Zapfen auf der Netzhaut hat zur Folge, dass verschiedene Bereiche des Blickfeldes unterschiedlich wahrgenommen werden. In der Fovea centralis, dem Zentrum der Netzhaut, ist die Dichte der Zapfen am höchsten und nimmt mit steigendem Abstand drastisch ab (Goldstein, 2010). Diese ungleiche Verteilung von Stäbchen zu Zapfen ist in Abbildung 3.1 zu sehen.

Abbildung 3.1

Verteilung der Stäbchen (Rods) und Zapfen (Cones) auf der Retina

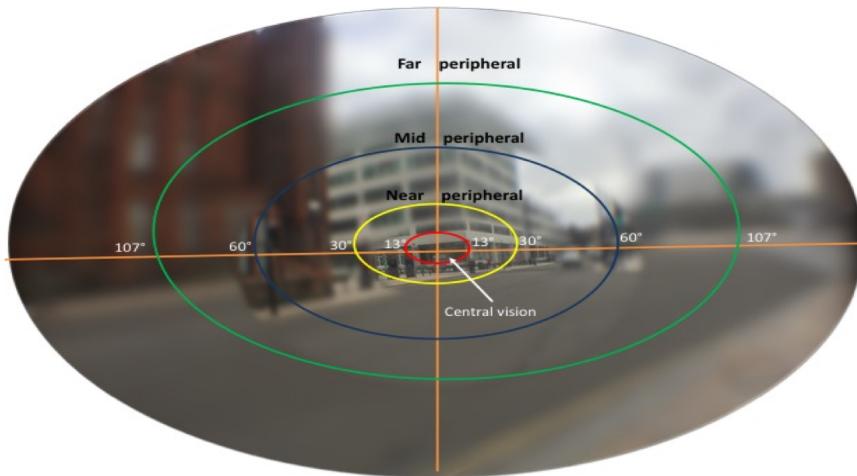


Anmerkung. Aus Goldstein (2010).

Wie einleitend beschrieben hat dies den Effekt, dass die Sehschärfe im Zentrum des Blickfeldes deutlich höher ist als in der Peripherie. Bereits bei 6° visuellem Winkel ist die Sehschärfe um 75 % reduziert (Koulieris et al., 2019). Eine Visualisierung dieses Umstandes zeigt Abbildung 3.2. Es ist zu erkennen, wie drastisch der Abfall der Sehschärfe mit zunehmendem visuellem Winkel ist.

Abbildung 3.2

Visualisierung abnehmender Sehschärfe bei zunehmendem visuellem Winkel



Anmerkung. Aus Younis et al. (2019).

Dieser Umstand der menschlichen Wahrnehmung hat erhebliche Implikationen hinsichtlich der Verteilung und Ausrichtung von Aufmerksamkeit. Er begründet, warum Menschen bei der Wahrnehmung ihrer Umgebung eine Vielzahl von Blickbewegungen tätigen, um einzelne Objekte sowie deren Bestandteile nacheinander in den Fokus der Fovea zu rücken. Gleichzeitig liefert er ein erstes Indiz dafür, dass die Aufmerksamkeit eines Menschen zu einem gegebenen Zeitpunkt wahrscheinlich dem Objekt gilt, auf welches ihr*sein Blick derzeit gerichtet ist.

3.1.2 Blickbewegungen

Das menschliche Auge ist ständig in Bewegung, um die Abbildung einzelner Objekte auf der Netzhaut in den Bereich den Fovea centralis zu lenken. Für diesen Zweck ist das Auge von sechs Muskeln umgeben, welche eine schnelle und präzise Ausrichtung des Auges zulassen (Shumway et al., 2023).

Um ein neues Areal in das Zentrum des Blickfeldes zu rücken, bewegt sich das Auge sprunghaft für eine kurze Zeit mit hoher Geschwindigkeit (Duchowski, 2017). Solche Bewegungen werden Sakkaden genannt. Typischerweise dauern Sakkaden zwischen 30 und 80 Millisekunden, wobei der Blick zwischen 4 und 20 Grad des visuellen Feldes überquert (Holmqvist et al., 2011). Während der Durchführung einer Sakkade werden keine visuellen Informationen aufgenommen.

Die visuelle Informationsaufnahme geschieht in Phasen relativen Stillstands der Augen, sogenannten Fixationen (Duchowski, 2017). Fixationen variieren in ihrer Dauer von wenigen Millisekunden bis zu mehreren Sekunden. Typische Werte liegen zwischen 150 und 600 Millisekunden (Holmqvist et al., 2011).

Die Betrachtung einer Umgebung, virtuell oder real, geschieht folglich durch eine Abfolge abwechselnder Sakkaden und Fixationen, wobei das Blickfeld schrittweise ‚abgetastet‘ und fixiert wird, um visuelle Informationen aus der Umgebung aufzunehmen. Da die visuelle Informationsaufnahme während Fixationen stattfindet, ist die Beantwortung der Fragen ob und wie oft ein bestimmtes Areal oder Objekt fixiert wurde, ein zentraler Parameter in der Eyetracking-Forschung.

Das Auge befindet sich während einer Fixation nicht in vollkommenem Stillstand. Über die Dauer einer Fixationen gleitet das Auge langsam vom Fixationszentrum weg, weshalb kleine Mikrosakkaden stattfinden, um diesen Drift zu korrigieren (Holmqvist et al., 2011). Obgleich diese Mikrobewegungen für die Zwecke dieser Arbeit nicht von großer Bedeutung sind, müssen sie bei der Implementierung von Algorithmen, welche der Fixationsdetektion dienen sollen, berücksichtigt werden. Es kann nicht davon ausgegangen werden, dass das Auge während einer Fixation vollständig unbewegt ist.

Wenn unsere Augen ein bewegtes Objekt verfolgen, geschieht dies nicht durch eine Abfolge abwechselnder Sakkaden und Fixationen, sondern durch eine gleichmäßige Verfolgung des Objekts, einem sogenannten Smooth Pursuit (Duchowski, 2017). Smooth Pursuits dienen dem Zweck, das Zielobjekt konstant im Zentrum des schärfsten Sehens abzubilden. Typischerweise bewegt sich der Blick während eines Smooth Pursuits mit 10 bis 30 Grad/s über das visuelle Feld (Holmqvist et al., 2011). Um zu überprüfen, ob ein bewegtes Objekt Fokus der Aufmerksamkeit einer*eines Betrachter(s)*in ist, ist deshalb ratsam, neben Fixationen auch solche Smooth Pursuits algorithmisch erkennen zu können.

Aus dem in diesem Abschnitt beschriebenen Vorwissen geht hervor, warum besonders die Detektion von Fixationen und Smooth Pursuit Bewegungen für die in dieser Arbeit vorgestellten Software zur Untersuchung der Aufmerksamkeitsverteilung in 360° Videos von Relevanz ist. Die beschriebenen Zustände und Blickbewegungen des Auges sind die physiologische Grundlage der in Sektion 3.2.3 vorgestellten Eyetracking-Parameter.

3.1.3 Akkommodation-Konvergenz Reflex

Die Augen des Menschen bewegen sich ständig, um Objekte in der Umgebung zu betrachten. Bei dieser Ausrichtung der Augen auf ein Zielobjekt findet eine reflexartige gegensätzliche Bewegung beider Augen statt, sodass der Punkt der Überschneidung ihrer Blickachsen auf dem fokussierten Objekt liegt. Diese gegensätzliche Bewegung der Augen nennt sich Konvergenz und ist unverzichtbar, um eine Doppelbildwahrnehmung zu vermeiden und stattdessen ein einziges dreidimensionales Bild des Objekts wahrzunehmen (Cassin & Rubin, 2006). Das Maß der Konvergenz beider Augen zueinander ist abhängig von der Distanz des Objekts zur* zum Betrachter*in. Je näher ein betrachtetes Objekt, desto größer ist die Konvergenz.

Um Objekte in verschiedenen Entfernungen scharf auf der Netzhaut abbilden zu können, nimmt das visuelle System zeitgleich eine Anpassung der Brechkraft der Linsen der Augen vor. Dieser Vorgang nennt sich Akkommodation und ist eine wesentliche Voraussetzung für das scharfe Sehen von Details (Levin & Adler, 2011).

Der sogenannte Akkommodations-Konvergenz Reflex steuert beide Vorgänge. Bei der Betrachtung einer natürlichen Umgebung stehen beide Vorgänge in enger Beziehung, da sie beide von der Entfernung des Objekts zur* zum Betrachter*in abhängig sind. Situationen, in denen Akkommodation und Konvergenz nicht im Einklang miteinander stehen, treten im Normalfall nicht auf.

Speziell bei der Nutzung von Head Mounted Displays kann jedoch genau solch eine Nichtübereinstimmung zwischen beiden Prozessen vorkommen. Der hier beschriebene Akkommodations-Konvergenz Reflex dient als Grundlage für das Verständnis des später erläuterten Vergenz-Akkommodation-Konflikts bei dem Einsatz von VR-Brillen und Head Mounted Displays.

3.1.4 Visuelle Aufmerksamkeit

Die beschriebenen Kenntnisse über das visuelle System, wie der Fakt, dass nur ein kleiner Bereich des visuellen Feldes, das der Fovea centralis, scharf sichtbar ist, geben Aufschlüsse darüber, dass die Aufmerksamkeit einer* eines Betrachter(s)* in wahrscheinlich dort vorzufinden ist, wo ihr* sein Blick hingerichtet ist. Die visuelle Aufmerksamkeit von Menschen nachzu vollziehen und herauszufinden, welche Stimuli für eine* einen Beobachter* in von Interesse sind, ist eine der Hauptmotivationen für die Eyetracking-Forschung.

Im Jahre 1980 formulierten Just und Carpenter die sog. „Eye-Mind“ Hypothese im Kontext der Leseforschung. Die Hypothese besagt, dass das Auge so lange auf ein Wort fixiert ist, solange es gedanklich verarbeitet wird. Weiterhin gäbe es keine nennenswerte Zeitdifferenz zwischen dem, was fixiert und dem, was verarbeitet wird

(Just & Carpenter, 1980). Eine allgemeine Gültigkeit der Hypothese würde bedeuten, dass Menschen immer dann wenn sie ein Objekt fixieren, auch über dieses nachdenken, und zwar für genauso lange, wie der Blick auf dem Objekt ruht. Eyetracking-Daten würden demnach direkt Aufschluss darüber geben, wann, wie oft und für wie lange ein bestimmtes Objekt Fokus der Gedankenwelt einer Person ist.

Dennoch ist es Menschen möglich, Objekte in der Peripherie des Blickfeldes mit Aufmerksamkeit zu belegen, ohne den Blick auf das Objekt zu richten. Posner (1980) nennt diese Art der Aufmerksamkeit „covert“, also verdeckte Aufmerksamkeit. In diesem Fall stimmt die Blickrichtung nicht mit dem Fokus der Aufmerksamkeit überein. Weiterhin ist das Betrachten visueller Sinneseindrücke nicht die einzige Sinnesmodalität, die unserer Aufmerksamkeit bedarf. Während Menschen prinzipiell dazu in der Lage sind, visuelle und auditive Stimuli gleichzeitig zu verarbeiten, ist die Performanz von Menschen in visuellen Aufgaben reduziert, wenn parallel auditive Stimuli präsentiert werden (Bonnel & Hafter, 1998). Versuchspersonen identifizieren so etwa bestimmte Zielobjekte langsamer, wenn währenddessen Textpassagen auditiv präsentiert werden, die sich eingeprägt werden sollen (Gherri & Eimer, 2011). Die Autoren schlussfolgern, dass aktives Zuhören die Verarbeitung visueller Reize beeinträchtigt. Die mögliche Verteilung von Aufmerksamkeit auf andere Sinne kann die Interpretation der Blickrichtung erschweren, da das, was gesehen, gegebenenfalls nicht mit voller Kapazität verarbeitet wird. Bei der Darbietung eines 360° Videos in VR ist vorstellbar, dass sich eine Versuchsperson auf Gesprochenes im Video fokussiert, während der Blick in einem zufälligen Areal ruht. Dieses Areal wäre in diesem Moment nicht Fokus der Gedankenwelt der Versuchsperson, da sie mit Zuhören beschäftigt ist, auch wenn die Eyetracking-Daten im Nachhinein ein hohes Interesse an dem Areal vermuten lassen würden. Auch ohne die Konzentration auf andere Sinnesmodalitäten kann die Aufmerksamkeit einer Person nahezu vollständig von der externen Umgebung abschweifen, hin zu einem ‚Verlieren‘ in eigenen Gedanken, sogenanntes „Mind Wandering“ (Smallwood & Schooler, 2015). Obwohl die Blickrichtung in solchen Momenten nicht mit dem Fokus der Aufmerksamkeit gleichzustellen ist, ist es interessanterweise möglich, solch ein Abschweifen anhand von Mustern in den Blickbewegungen zu erkennen (Brishtel et al., 2020; Lee et al., 2021). Sollten während der Durchführung von Eyetracking-Studien viele verdeckte Aufmerksamkeitsverschiebungen, Fokus auf andere Sinnesmodalitäten, oder gar gedankliches Abschweifen auftreten, so würden die Daten lediglich Aufschluss darüber geben, worauf der Blick zu einem gegebenen Zeitpunkt gerichtet war, im starken Kontrast zur Eye-Mind Hypothese, nicht jedoch, worauf die Aufmerksamkeit der Person lag.

Trotz der vorgestellten Limitationen ist unbestreitbar, dass Augenbewegungen und visuelle Aufmerksamkeit stark in Zusammenhang stehen. Nach Hoffman (2016) ist der derzeitige Konsens, dass die visuelle Aufmerksamkeit 150 bis 200 Millisekunden vor dem Blick auf ein neues Objekt gerichtet wird, aber dass,

sobald sich die Aufmerksamkeit verschiebt, die Augen natürlich und reflexartig folgen. Dennoch müssen eventuelle Faktoren, die die Interpretation der Ergebnisse erschweren, bei der Versuchsdurchführung kontrolliert und abgefragt werden. Häufig werden Eyetracking-Daten mit Ergebnissen von Fragebögen kombiniert, um eventuelle Störfaktoren zu identifizieren und Blickrichtungsdaten mit den Meinungen der Versuchspersonen anzureichern (z. B. Sullivan et al., 2017). Dieses Vorgehen der sogenannten methodischen Triangulation (Denzin, 2017) ist aufgrund der genannten Störfaktoren für die Interpretation von Eyetracking-Studien unerlässlich (Holmqvist et al., 2011).

3.2 Eyetracking

Die in den vorherigen Abschnitten beschriebenen Eigenschaften des visuellen Systems und der visuellen Aufmerksamkeit motivieren die Existenz technischer Systeme, die es erlauben, die Blickrichtung von Menschen zu bestimmen, um daraus den Fokus der derzeitigen Aufmerksamkeit abzuleiten. Eyetracker haben eine lange Geschichte und kamen bereits im 19. Jahrhundert in ersten Experimenten zum Einsatz (Płużycka, 2018). Heutzutage sind fortgeschrittene Eyetracking-Systeme dazu in der Lage, Blickbewegungen mit erstaunlicher Präzision, Genauigkeit und temporaler Auflösung zu messen. Die dabei meistverbreitete Technologie sind videobasierte Pupillen-Cornealreflex-Systeme (Duchowski, 2017). Da ein ebensolches System in der VR-Brille verbaut ist, die in dem vorgestellten Projekt verwendet wurde, wird kurz auf die Funktionsweise dieser Systeme eingegangen. Im Anschluss wird das Konzept von Areas of Interest, welches für diese Arbeit von großer Bedeutung ist, eingeführt und es werden relevante Eyetracking-Parameter vorgestellt.

3.2.1 Funktionsweise

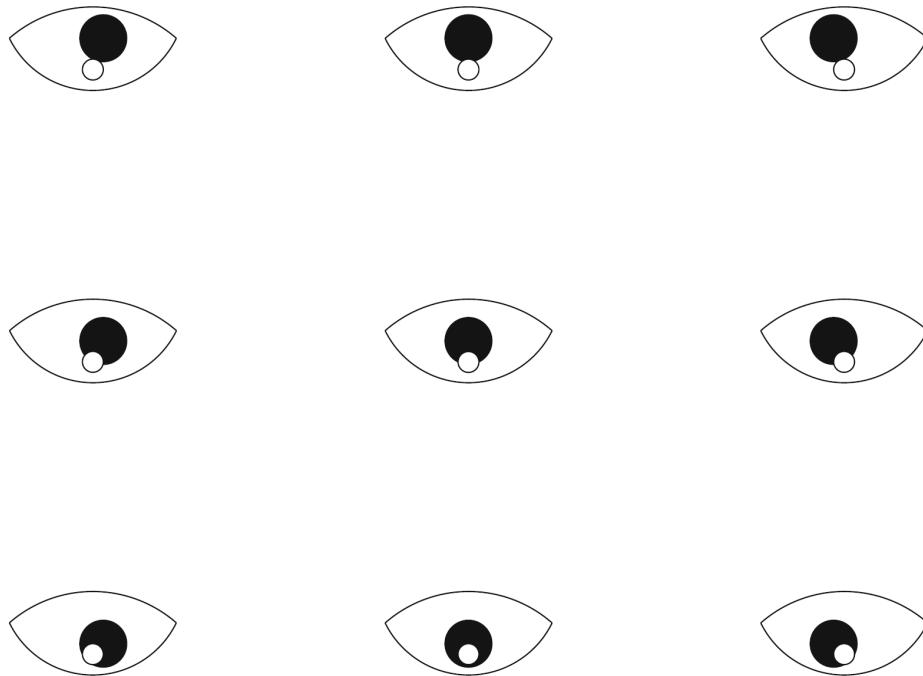
Videobasierte Pupillen-Cornealreflex-Systeme sind im Vergleich zu anderen Technologien relativ kostengünstig und haben den Vorteil, nicht-invasiv zu sein. Dabei wird das Auge von einer Lichtquelle, typischerweise einer Infrarot-LED, beleuchtet und mit einer Kamera gefilmt. Die Videoaufzeichnungen enthalten neben anatomischen Merkmalen Reflexionen der Lichtquelle auf den Grenzflächen des Auges (Duchowski, 2017; Holmqvist et al., 2011). Diese typischen Reflexionen, die sogenannten Purkinje-Bilder, dienen zusammen mit anderen anatomischen Features, wie z. B. der Pupille, als Anhaltspunkte für Algorithmen, welche die Rotation des Auges basierend auf der Position dieser Features bestimmen. Das in Eyetrackern häufig eingesetzte Purkinje-Bild ist die Reflexion der Lichtquelle auf der Oberfläche der Cornea, der Hornhaut des Auges. Die Bestimmung der Position der Cornealreflexion in Relation zu dem Zentrum der Pupille erlaubt die Ermittlung der Rotation des Auges (Duchowski, 2017). Abbildung 3.3 zeigt, wie sich die beiden Features bei einer 9-Punkt-Kalibrierung relativ zueinander bewegen.

Aufgrund ihrer Funktionsweise sind videobasierte Eyetracker anfällig für Problemquellen, welche Features wie die Pupille oder Purkinje-Reflexionen verdecken oder die Bildqualität reduzieren. Dazu gehören z. B. anatomische Merkmale der Versuchsperson wie Schlupflieder oder nach unten gerichtete Wimpern, aber auch Mascara, Brillen oder Kontaktlinsen können für störende Artefakte und Reflexionen in den Aufzeichnungen sorgen. Zudem sollten andere Infrarot-Lichtquellen, welche mit der Lichtquelle des Eyetracking-Systems interferieren könnten, vermieden werden (Holmqvist et al., 2011).

Die in diesem Projekt verwendete VR-Brille (HTC Vive Pro Eye) enthält genau solch ein videobasiertes Eyetracking-System, sodass die beschriebenen Problemquellen

Abbildung 3.3

Relative Position der Pupille und Cornealreflexion zueinander bei 9-Punkt-Kalibrierung



Anmerkung. Aus Duchowski (2017).

bei der Durchführung von Experimenten kontrolliert und wenn möglich eliminiert werden sollten.

3.2.2 Areas of Interest

Während das Auftreten bestimmter Ereignisse wie die in 3.1.2 beschriebenen Sakkaden oder Fixationen aus Blickbewegungsdaten ohne Informationen über den dargebotenen Stimulus errechnet werden können, bedarf es für die Untersuchung vieler Fragestellungen einen Bezug zwischen den Blickbewegungsdaten und dem gezeigten Stimulus. So möchte man z. B. oft nicht nur wissen, dass eine Fixation stattgefunden hat, sondern auch was fixiert wurde. Eine Möglichkeit Eyetracking-Daten mit Stimulus bezogenen Informationen anzureichern, ist der Einsatz sogenannter Areas of Interest (AOIs). AOIs sind definierte Regionen in dem dargebotenen Stimulus und ermöglichen die Bestimmung weiterer Parameter, z. B. wie lange oder wie oft eine bestimmte AOI über die Dauer eines Experiments angesehen wurde (Holmqvist et al., 2011).

Dabei können AOIs je nach Art des Stimulus statisch oder dynamisch sein. Statische AOIs bleiben in ihrer Form, Größe und Position über den Verlauf des Experiments unverändert, da sich der zugrunde liegende Stimulus nicht bewegt (z. B. Regionen in einem Bild). Dynamische AOIs hingegen passen ihre Form und Position stetig an

einen sich verändernden Stimulus an (z. B. Regionen in einem Video) (Holmqvist et al., 2011).

Die Nutzung von AOIs erlaubt die Untersuchung vielfältiger Fragestellungen, jedoch sollten auch hierbei einige methodische Überlegungen und Probleme berücksichtigt werden. Es existieren wenige Richtlinien zur Positionierung von AOIs innerhalb des Stimulus-Materials (Hessels et al., 2016). Dies hat zur Folge, dass Forschung bezüglich desselben Sachverhalts mit demselben Material gegebenenfalls unterschiedliche Ergebnisse erzielt, da verschiedene AOIs im gleichen Stimulus definiert wurden. Holmqvist et al. (2011) argumentieren, dass die Positionierung von AOIs durch Algorithmen objektiver ist und gegenüber der manuellen Positionierung vorgezogen werden sollte. Weiterhin sollten sich AOIs nicht überlappen, um die für statistische Tests nötige Annahme der Unabhängigkeit der Daten zu gewährleisten (Holmqvist et al., 2011). Besonders bei der später beschriebenen Animation von AOIs durch Interpolation kann es vorkommen, dass sich AOIs kurzzeitig überlappen, ohne dass sich die Versuchsleitung darüber bewusst ist, da die Animation halb-automatisiert erstellt wurde. Ein Bewusstsein über die methodischen Bedenken bei dem Einsatz von AOIs ist für die Nutzung der vorgestellten Software hilfreich. Für eine vollständige Übersicht relevanter Überlegungen bezüglich der statistischen Auswertung von AOI-Daten sowie weiterer Verweise auf relevante Literatur siehe Holmqvist et al. (2011).

Die Erstellung und Animation von AOIs erfolgt für gewöhnlich in einem sogenannten AOI Editor. Diese werden oft kostenpflichtig von dem Hersteller des verwendeten Eyetracking-Systems zur Verfügung gestellt. Meist handelt es sich dabei um proprietäre Software, die durch Anwender*innen nicht angepasst werden kann. Die Algorithmen zur Bestimmung der relevanten AOI-Parameter sind in diesem Fall nicht frei zugänglich und können deshalb nicht überprüft oder modifiziert werden. Dieser Umstand motiviert die in dieser Arbeit vorgestellte Entwicklung eines Open-Source AOI Editors, speziell für die Erstellung von AOIs in 360° Videos zur Versuchsdurchführung in Virtual Reality.

3.2.3 Parameter

In dem folgenden Abschnitt wird eine Auswahl häufig genutzter Eyetracking-Parameter vorgestellt. Dabei wird sich auf Maße beschränkt, die im Zuge der Entwicklung eines AOI Editors von Bedeutung sind. Für eine umfangreiche Erläuterung weiterer Parameter siehe Holmqvist et al. (2011). Die Interpretation jedes Parameters ist stark abhängig von dem Kontext und den Instruktionen eines Versuchs, weshalb jeweils nur wenige Anwendungsbeispiele angeschnitten werden. Weiterhin ist anzumerken, dass in der Literatur viele verschiedene Begriffe für die unten stehenden Maße zu finden sind. Im Folgenden wird die Nomenklatur von Holmqvist et al. (2011) verwendet. Zuletzt ist hervorzuheben, dass obwohl

sie methodisch interessant sind, keine Parameter bezüglich Smooth Pursuit Bewegungen gelistet werden. Dies ist dem Umstand geschuldet, dass die Detektion von Smooth Pursuits nicht Teil des Umfangs dieser Arbeit ist.

AOI Hit AOI Hits sind der simpelste AOI-Parameter und stellen die Datengrundlage für die Bestimmung komplexerer Parameter dar. Sofern die Blickkoordinaten eines einzelnen Blick-Datenpunkts innerhalb einer definierten Area of Interest sind, spricht man von einem ‚Treffer‘ auf die AOI, also einem AOI Hit (Holmqvist et al., 2011).

Dwells Ein Dwell stellt einzelnes ‚Ansehen‘ einer AOI, von Eintritt des Blickes in den Bereich der AOI bis zum Verlassen der AOI dar (Holmqvist et al., 2011). Ein Dwell besteht also aus einer Kette von AOI Hits auf die gleiche AOI, mit einem Start, Ende und einer Dauer. Dabei wird nicht zwischen der genauen Art der Blickbewegung unterschieden. Dwells beinhalten nach dieser Definition also Sakkaden, Fixationen sowie Smooth Pursuit Bewegungen über dem Areal der AOI.

- *Dwell Count:* Die Anzahl an Dwells pro AOI ist ein häufig genutzter Parameter in der Human Factors Forschung. Wie oft ein bestimmtes Areal besucht wurde, kann je nach Kontext z. B. Aufschlüsse über die Komplexität oder den Informationsgehalt des Areals geben (Holmqvist et al., 2011).
- *Dwell Time:* Die Dauer eines Dwells von Eintritt bis Austritt des Blickes wird Dwell Time genannt. Eine längere Verweildauer kann je nach Kontext z. B. erhöhtes Interesse, höheren Informationsgrad oder höhere Komplexität des Areals bedeuten (Holmqvist et al., 2011).
- *Total Dwell Time:* Die Summe der einzelnen Verweildauern aller Dwells auf eine AOI wird Total Dwell Time genannt (Holmqvist et al., 2011).
- *First Pass Dwell Time:* Die Verweildauer des ersten Dwells ist die sogenannte First Pass Dwell Time (Holmqvist et al., 2011). Die Dauer des ersten Dwells auf eine AOI ist deshalb interessant, weil sie ähnlich zur ersten Fixation mit frühen kognitiven Prozessen wie der Objekterkennung in Verbindung steht (Holmqvist et al., 2011).
- *Entry Time:* Die Entry Time ist der Zeitpunkt, zu dem eine AOI das erste Mal angesehen worden ist (Holmqvist et al., 2011). Je nach Operationalisierung werden in der Literatur Zeiten bis zum ersten Dwell oder der ersten Fixation ermittelt. In dieser Arbeit wird die Entry Time deswegen in die zwei Parameter *Time To First Dwell* und *Time To First Fixation* getrennt. Dabei handelt es sich jeweils um den Startzeitpunkt des ersten Dwells bzw. der ersten Fixation auf eine AOI.

Fixationen Wie in Abschnitt 3.1.2 erläutert, sind Fixationen Zeiten relativen Stillstands der Augen, zu welchen die visuelle Informationsaufnahme stattfindet. Die beschriebenen Parameter sind eng mit den dwellbezogenen Messgrößen verwandt, jedoch enthalten die zu ihrer Bestimmungen herangezogenen Datenpunkte z. B. keine Sakkaden und lassen sich dadurch präziser interpretieren. Theoretisch kann die Anzahl und Dauer von Fixationen ohne die Verwendung von Areas of Interest bestimmt werden. Der Einsatz von AOIs erlaubt jedoch, das Auftreten von Fixationen direkt Arealen bzw. Objekten im Stimulus zuzuordnen. Jedoch können innerhalb einer AOI Fixationen an unterschiedlichen Stellen auftreten, sodass je nach Größe der AOI ohne Weiteres keine genaue Angabe über den exakten Ort der Fixation gegeben werden kann.

- *Number Of Fixations:* Die Anzahl der Fixationen gibt Auskunft darüber, wie oft eine Area of Interest in einem Versuchsdurchlauf fixiert worden ist. Eine höhere Anzahl von Fixationen auf eine AOI kann Aufschluss über die Bedeutsamkeit und Informationsfülle des Areals geben (Holmqvist et al., 2011). Der Einsatz des Parameters ist vielfältig. In Studien zu Expertise zeigt sich z. B., dass Expert*innen in ihrer Domäne weniger Fixationen als Noviz*innen benötigen, um eine Aufgabe zu bewältigen (Duchowski, 2017; Holmqvist et al., 2011).
- *Time To First Fixation:* Die Zeit zur ersten Fixation gibt Auskunft darüber, wann das erste Mal eine Fixation im Bereich einer AOI aufgetreten ist (Holmqvist et al., 2011). In Studien zur Suchzeit kann sie z. B. signalisieren, dass das aufzufindende Objekt gefunden worden ist (Rötting, 2001).
- *First Fixation Duration:* Ähnlich wie die *First Pass Dwell Time* kann die Dauer der ersten Fixation Aufschlüsse über frühe visuelle Informationsverarbeitungsprozesse wie die Objekterkennung geben (Holmqvist et al., 2011). Aufgrund der engen Kopplung von Fixationen und Informationsaufnahme ist die Fixationsdauer allgemein ein vielfach verwendeter Parameter in vielen unterschiedlichen Kontexten (Rötting, 2001).

Die beschriebenen Parameter sollen von der in dieser Arbeit entwickelten Software bestimmt und ausgegeben werden. In Kombination geben sie Aufschluss darüber, wie oft und für wie lange eine definierte AOI angesehen und fixiert worden ist. Die Interpretation der Parameter variiert stark je nach Kontext und Instruktionen und ist deshalb den Anwender*innen der Software überlassen.

3.3 Head Mounted Displays und virtuelle Realität

Neben Eyetracking-Systemen sind Head Mounted Displays (HMDs) die zweite wichtige Technologie, die im Zuge dieser Arbeit verwendet wird. Dabei handelt es sich um Bildschirme, die am Kopf einer Person befestigt werden, oft mit dem Zweck, virtuelle Umgebungen darzustellen (Shibata, 2002). Häufig werden HMDs mit weiteren Tracking-Technologien zu einer VR-Brille kombiniert, sodass die VR-Brille auf Kopf-, Hand- und Augenbewegungen des Menschen reagieren kann (LaValle, 2023). Für eine technische Beschreibung aktuell verwendeter Display- und Tracking-Technologien siehe Koulieris et al. (2019). Es ist anzumerken, dass HMDs bzw. VR-Brillen nur eine von vielen Möglichkeiten sind, eine virtuelle Welt abzubilden. Einen Überblick über weitere Technologien sowie einer Taxonomie von VR-Systemen liefert Muhanna (2015).

In dieser Sektion wird zunächst die Funktionsweise von Head Mounted Displays beschrieben. Da die Technologie aufgrund ihres Potenzials, virtuelle Umgebungen immersiv zu präsentieren, interessant ist, wird im Anschluss auf die Konzepte Immersion und Präsenz in Virtual Reality eingegangen. Abschließend wird ein Überblick über die Limitationen von HMDs und VR-Brillen gegeben, die bei der Gestaltung von Studien zu berücksichtigen sind.

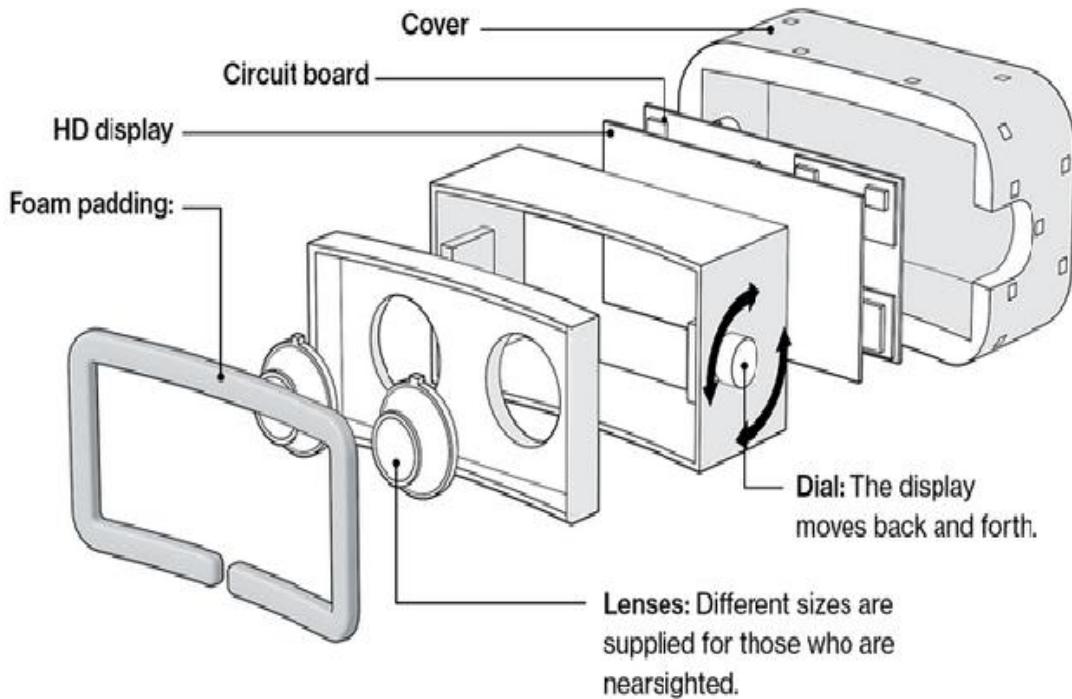
3.3.1 Funktionsweise

Head Mounted Displays verfügen über einen oder mehrere Bildschirme, welche durch ein Headset sehr nah vor dem Gesicht der tragenden Person positioniert werden. Stereoskopische HMDs sind dazu in der Lage, beiden Augen zwei leicht versetzte Ansichten der gleichen Szene zu präsentieren, um so den Eindruck räumlicher Tiefe zu schaffen (Shibata, 2002). Die Positionierung der Bildschirme nur wenige Zentimeter vor den Augen ist wichtig, um ein möglichst großes Sichtfeld zu erzeugen (LaValle, 2023). Da selbst junge Menschen einen Bildschirm, der 10 cm vor dem Auge positioniert ist, bereits nicht mehr scharf fokussieren könnten, werden die Displays in VR-Brillen durch eine konvexe Linse betrachtet (LaValle, 2023). Die Linsen sind so eingestellt, dass das Auge, um die Bildschirme scharf abzubilden, „unendlich“ in die Ferne fokussiert. Diese Fernakkommodation erfolgt durch eine Entspannung der Augenmuskulatur und macht das Tragen von HMDs weniger ermüdend für die Augen. Ein simpler Aufbau ist in Abbildung 3.4 zu sehen.

Die beschriebenen Einzelteile werden von einem Headset umschlossen, das auf dem Kopf der Person getragen wird, sodass das gesamte System den Kopfbewegungen des Anwenders folgt. Weiterhin sorgt das Umschließen der Bildschirme durch eine Headsetkonstruktion dafür, dass kein Licht aus der Umgebung in den Blick der*des Träger(s)*in fällt. So kann sich die Person vollständig auf die virtuelle Umgebung fokussieren. Das vorgestellte Hintergrundwissen über den Aufbau von HMDs ist

Abbildung 3.4

Schematischer Aufbau einer VR-Brille (Oculus Rift)



Anmerkung. Aus Desai et al. (2014).

hilfreich für das Verständnis der in 3.3.3 vorgestellten Limitationen der Technologie.

3.3.2 Immersion und Präsenz in VR

Head Mounted Displays bzw. VR-Brillen sind für Forschungszwecke interessant, da sie es erlauben, virtuelle Umgebungen realitätsgerecht abzubilden. Um zu argumentieren, dass dadurch eine höhere ökologische Validität erzielt werden kann, setzt dies voraus, dass sich Menschen in einer virtuellen Umgebung natürlicher verhalten als bei der Darbietung des gleichen Materials auf einem Desktop-Bildschirm.

Zur Untersuchung dieses Umstands werden die beiden Begriffe Immersion und Präsenz unterschieden. Immersion beschreibt die objektive Eigenschaft eines VR-Systems, natürliche Bedingungen sowohl für die Wahrnehmung von sensorischen Reizen, als auch für die Interaktion mit der virtuellen Umgebung zu schaffen (Mangianti et al., 2017; Slater, 2018; Wilkinson et al., 2021). Faktoren wie die Größe des Sichtfelds (Field of View), Displayauflösung, Surround-Klang, Kopf- und Körpertacking sowie multisensorische Feedback-Mechanismen sind Beispiele für technische Eigenschaften eines Systems, die die Immersion bestimmen (Slater, 2018).

Der Begriff Präsenz hingegen ist eng mit der Erfahrung verknüpft, die ein*e Benutzer*in einer VR-Umgebung erfährt (Wilkinson et al., 2021). Sich in einer virtuellen

Umgebung präsent zu fühlen, bedeutet physiologisch und emotional auf die dargebotene Umgebung zu reagieren, als wäre diese echt, auch wenn man sich kognitiv vollkommen darüber bewusst ist, dass es sich um eine Simulation handelt (Slater, 2018).

Studien in verschiedenen Kontexten zeigen, dass sich Versuchspersonen bei der Darbietung einer virtuellen Umgebung mit VR-Systemen mehr in dieser präsent fühlen, als bei der Betrachtung der gleichen Umgebung auf einem Desktop-Bildschirm. Pallavicini et al. (2019) untersuchen dieses Phänomen z. B. beim Spielen von Videospielen, Shu et al. (2019) bei der Nutzung eines Erdbeben-Edukationssystems. Meißner et al. (2019) stellen in ihrer Arbeit die ökologische Validität von Desktop-Eyetracking, Eyetracking in Virtual Reality und Eyetracking in realen Umgebungen gegenüber. Die Autoren argumentieren, dass unter der Voraussetzung, dass die virtuelle Umgebung realitätsgetreu gestaltet wurde, Eyetracking-Studien in VR gegenüber Desktop-Eyetracking-Studien mehr ökologische Validität erzielen. Für die in dieser Arbeit vorgestellte Software bedeutet das, dass, sofern das verwendete 360° Video eine realistische Umwelt abbildet, davon ausgegangen werden kann, dass die ökologische Validität der Untersuchung höher ist, als bei der Präsentation des gleichen Stimulus auf einem Bildschirm.

3.3.3 Limitationen

Die in VR wahrgenommene Präsenz ist stark mit der Immersion, die ein VR-Headset bieten kann, verknüpft. Aus ihrer Funktionsweise und dem Stand der Technik ergeben sich einige Limitationen, die VR-Brillen bezüglich ihrer Immersion, der von Versuchspersonen erlebten Präsenz und ihrem Einsatz für Studien mit sich bringen.

Kommerziell erhältliche VR-Brillen dienen primär der Präsentation visueller und auditiver Stimuli. Die Darbietung von realistischem haptischem oder gar olfaktorischem Feedback ist derzeit sehr herausfordernd. Weiterhin sind die Interaktionsmöglichkeiten mit der virtuellen Umgebung in VR stark beschränkt. Sich z. B. in einer VR-Umgebung durch natürliches Gehen zu bewegen ist technisch herausfordernd und benötigt weitere Hardwarekomponenten. Fehlende haptische, vestibuläre und propriozeptive Reize können zu einem Mismatch zwischen virtueller und physischer Erfahrung führen. Im besten Falle reduziert dies lediglich die erlebte Präsenz, häufig führt dies jedoch zu Übelkeit in Form von Motion- und Simulator-Sickness. In ihrer Arbeit stellen Clay et al. (2019) methodische Überlegungen zu Eyetracking-Studien in VR vor. Die Autoren zeigen in ihrer Pilotstudie, dass besonders Motion Sickness und die, um Motion Sickness zu reduzieren, unnatürliche Art und Weise, um sich in virtuellen Welten zu bewegen, die Nutzung von VR für Forschungszwecke verkomplizieren. Diese Limitationen in der Interaktion beschränken

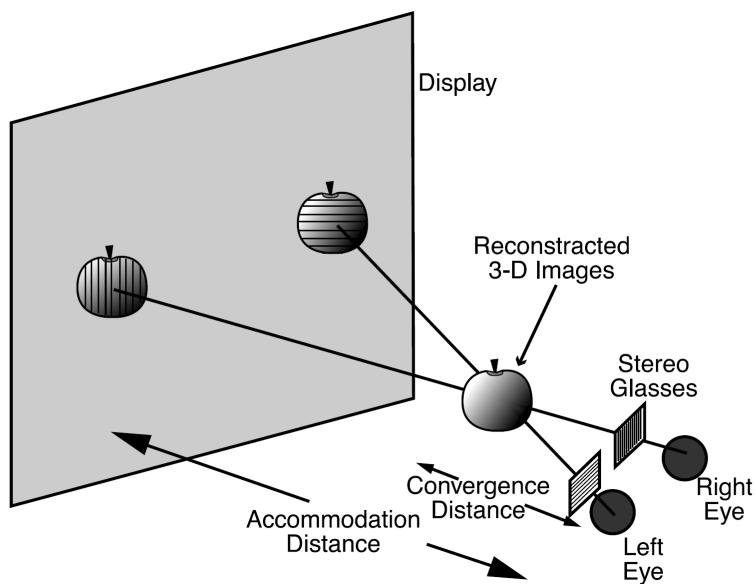
die Anwendung von VR häufig auf Fragestellungen, in denen sich Versuchspersonen vergleichsweise passiv verhalten.

Die Nutzung von VR zur Erforschung von sozialen Interaktionen ist ebenfalls limitiert. Andere Menschen und soziale Interaktion realistisch zu reproduzieren, ist herausfordernd. Nonverbale Kommunikation wie Gesichtsausdrücke und Körpersprache lassen sich derzeit nur mit viel Aufwand realistisch virtuell modellieren (Pan & Hamilton, Antonia F. de C., 2018).

Weiterhin ist selbst die Nachahmung realistischer visueller Stimuli nicht perfekt. In Sektion 3.1.3 wurde der Akkommodations-Konvergenz Reflex erläutert. Beim Tragen einer VR-Brille ist die Brechkraft der Linse des Auges (Akkommodation) nicht mit dem Winkel zwischen den Augen (Konvergenz) im Einklang. Dies ist der Fall, da die Linsen der Augen auf ein Display mit gleichbleibender Distanz fokussieren, während sich die Konvergenz der Augen auf Objekte in der virtuellen Umgebung, welche in verschiedenen Abständen zu dem Betrachter*in erscheinen, einstellt (Kramida, 2016). Abbildung 3.5 zeigt die Akkommodation auf das Display mit gleichzeitiger divergierender Vergenz auf das 3D-Objekt in der virtuellen Umgebung. Die Folge dieses Konflikts kann zu einer Überanstrengung der Augen und Übelkeit führen und stellt eine erhebliche Limitation in der Nutzung von HMDs dar.

Abbildung 3.5

Nichtübereinstimmung von Akkommodation und Konvergenz bei der Betrachtung eines Objekts in virtual Reality



Anmerkung. Aus Shibata (2002).

Zusätzlich kommt es aufgrund des beschriebenen Einsatzes von Linsen in

VR-Brillen zu linsenbedingten optischen Abbildungsfehlern, sogenannten Aberrationen (LaValle, 2023). Diese können Verzerrungen des wahrgenommenen Bildes sowie Unschärfe und Farbfehler verursachen. Weiterhin unterliegen die in HMDs verbauten Bildschirme Beschränkungen in ihrer Auflösung. Da sie sehr nah vor den Augen der*des Anwender(s)*in positioniert sind, müssten sie ausreichend hoch aufgelöst sein, damit einzelne Pixel nicht mehr sichtbar wären. Bildschirme mit derartig hoher Auflösung (bei sehr hoher Bildwiederholrate) zu betreiben, würde jedoch eine extrem hohe Bandbreite des Systems sowie sehr hohe Rechenleistung erfordern (Koulieris et al., 2019). Dementsprechend sind bei der Betrachtung der in HMDs verbauten Bildschirme derzeit einzelne Pixel auszumachen, wodurch die Immersion der Systeme reduziert ist. Abgesehen von der Auflösung der Bildschirme sind die Farbwiedergabe, Helligkeit und der Kontrast weitere ausschlaggebende Faktoren für den wahrgenommenen Realismus der betrachteten Szene. Derzeit verwendete Bildschirme sind nicht dazu in der Lage, kontrastreiche Szenen mit der in der Realität vorzufindenden Licht- und Dunkelintensität wiederzugeben. Meta, ein großes US-amerikanisches Unternehmen und Hersteller von VR-Brillen, sieht die Bewältigung dieser vier Probleme, des Vergenz-Akkommodations Konflikts, der zu geringen Bildschirmauflösung, das Auftreten linsenbedingter Abbildungsfehler sowie die unzureichende Farbwiedergabe als höchste Priorität für die Darbietung von virtuellen visuellen Reizen, die nicht von der Realität zu unterscheiden sind (Tech at Meta, 2022)

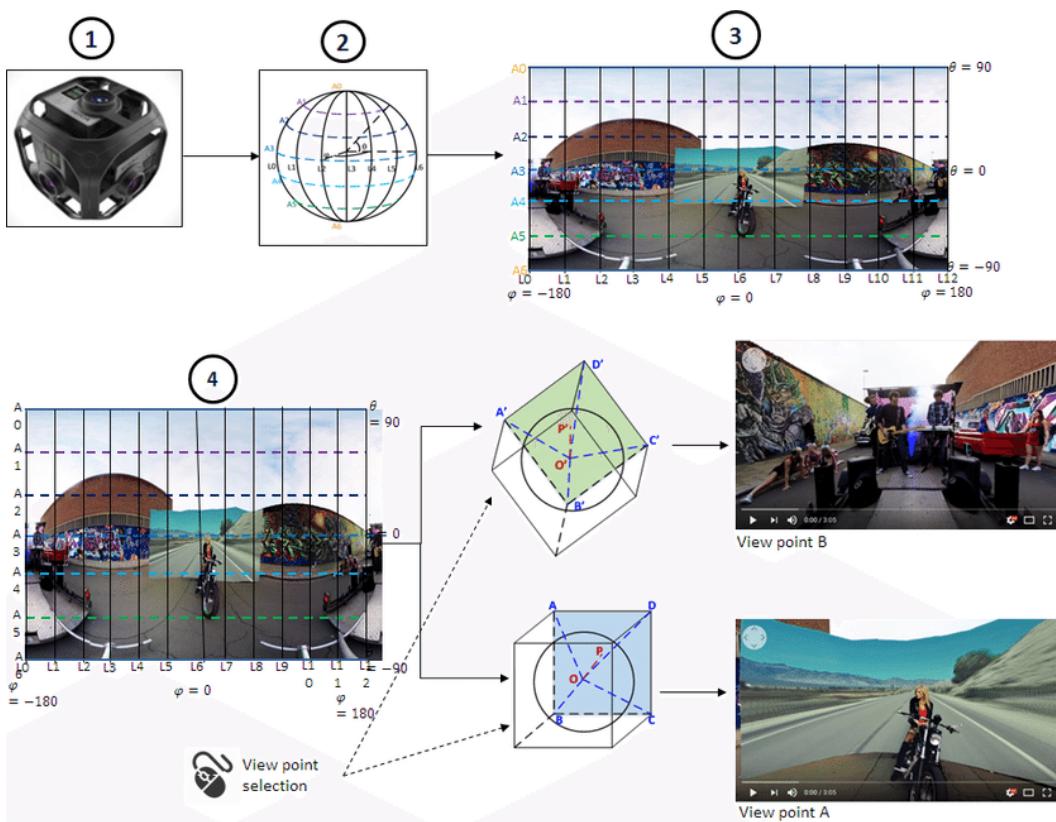
Die dargestellten Limitationen der Technologie reduzieren die Anwendbarkeit von HMDs und müssen bei der Gestaltung von Studien mit der in dieser Arbeit vorgestellten Software berücksichtigt werden. Vor allem das Auftreten von Motion-Sickness sollte antizipiert werden, sodass mit Schwindel oder Übelkeit von Versuchspersonen richtig umgegangen werden kann. Die Untersuchung sozialer oder anderweitig komplexer Interaktionen mit virtuellen Umgebungen, die über ein passives Betrachten der Szene hinaus gehen, erfordern ausreichend technische Kenntnisse und erheblichen Aufwand, um mit angemessener ökologischer Validität untersucht werden zu können.

3.4 360° Videos

Zu guter Letzt werden in dieser Sektion 360° Videos, oft auch immersive Videos genannt, vorgestellt. Dabei handelt es sich um Videos, bei denen Videomaterial durch eine spezielle Kamera oder durch ein Array mehrere Kameras in alle Richtungen aufgezeichnet und zu einem einzigen Video zusammengesetzt wird. Ein*e Betrachter*in kann sich in dem resultierendem Video umsehen und die vollständige 360° Szene in alle Richtungen erkunden. In Abbildung 3.6 ist der Erstellungs- und Wiedergabeprozess eines 360° Videos abgebildet.

Abbildung 3.6

Erstellung und Wiedergabe eines 360° Videos



Anmerkung. Aus Mangiante et al. (2017).

Interessant zu wissen ist, dass immersive Videos, wie in Schritt 3 in Abbildung 3.6 zu sehen ist, meist als Abfolge von zweidimensionalen rektangular-projizierten Panorama-Bildern kodiert sind und bei der Wiedergabe des Videos auf eine Kugelform abgebildet werden. Ein Frame eines solchen Videos kann also, wenn auch verzerrt, zweidimensional betrachtet werden, ohne eine spezielle 360° Wiedergabesoftware zu verwenden. Martin et al. (2021) stellen in ihrer Arbeit Scanpaths, also eine zeitliche Abfolge von Blickbewegungen, zweidimensional auf der Rektangularprojektion dar (siehe Abbildung 3.7). Dieses Verständnis ist für den Kontext dieser Arbeit hilfreich, da es Möglichkeiten aufzeigt, Blickbewegungen zukünftig z. B. in

Form von Scanpaths oder Heatmaps auf dem Stimulus zu visualisieren.

Abbildung 3.7

Scanpath auf Rektangularprojektion einer 360° Umgebung



Anmerkung. Aus Martin et al. (2021).

Bei der Betrachtung eines 360° Videos mit einer VR-Brille sieht die*der Anwender*in einen Ausschnitt des kugelförmig um sie*ihn herum projizierten Videos. Durch das Drehen des Kopfes kann die*der Betrachter*in den sichtbaren Ausschnitt des Videos anpassen und sich in dem Video umsehen. Die Nutzung von 360° Videos in VR erlaubt realistische Stimuli einfach zu erstellen und kontrolliert darzubieten. Saad et al. (2021) demonstrieren den Einsatz von 360° Videos in Eyetracking-Studien zur Untersuchung von sogenannten Shoulder-Surfing Attacks in öffentlichen Verkehrsmitteln. Die Autoren beschreiben, dass der Forschungsgegenstand aus Gründen des Datenschutzes und der Privatsphäre nur schwer im Feld untersuchbar ist und zeigen das Potenzial der Technologien zur Erforschung solcher Fragestellungen. Saad et al. haben das Studiensetup selbst in der Videospielentwicklungsplattform Unity implementiert, da es wenige Softwarelösungen gibt, die Eyetracking-Studien mit AOIs in 360° Videos unterstützen. Die vorgestellte Arbeit schließt direkt an diesen Umstand an, mit dem Ziel, dafür eine Softwarelösung bereitzustellen.

Kapitel 4

Verwandte Arbeiten

Es existieren bereits verschiedene Softwarelösungen für die Arbeit mit Eyetracking-Systemen, Areas of Interest und Videomaterial. Im folgenden Abschnitt wird ein Überblick über die Landschaft bestehender Software und Tools gegeben. Dabei werden vor allem Arbeiten vorgestellt, in welchen Open-Source-Software entwickelt wurde, die es erlaubt, Areas of Interest in Stimuli zu definieren. Es wird jeweils zunächst das von den Autoren vorgestellte Tool beschrieben und im Anschluss betrachtet, ob es für die Arbeit mit 360° Videos nutzbar ist. Zum Schluss werden der Vollständigkeit halber proprietäre, kostenpflichtige Softwarelösungen gelistet.

4.1 Open-Source-Software

Vosskühler et al. (2008) stellen in ihrer Arbeit den Open Gaze And Mouse Analyzer (OGAMA) vor. Dabei handelt es sich um eine umfangreiche Open-Source-Software zur Planung, Durchführung und Analyse von bildschirmbasierten Eyetracking-Experimenten. Die Software unterstützt die*den Benutzer*in bei dem gesamten Experiment-Workflow, von Studiendesign bis zu statistischer Auswertung, ohne dass Programmierkenntnisse erforderlich sind. Die Software ermöglicht die Erstellung und Analyse von AOIs, jedoch können diese nur für statisches Bildmaterial definiert werden. Dynamische AOIs innerhalb von Videos zu animieren, ist nicht möglich. Weiterhin wurde das Programm speziell für die Präsentation von Stimuli auf einem Bildschirm entwickelt. 360° Videos und die Nutzung von VR-Brillen werden von dem Tool nicht unterstützt.

Papenmeier und Huff (2010) entwickelten bereits 2010 ein Open-Source-Tool zur Definition von dynamischen AOIs für Videomaterial. Das Programm DynAOI erlaubt die Erstellung und Animation von 3D-Modellen, deren Form und Bewegung an dynamische Objekte in dem zu untersuchenden Video angenähert werden. Das Tool selbst ermöglicht keine Durchführung von Experimenten, sondern arbeitet mit dem Rohdatenexport einer anderen, von der*dem Benutzer*in selbst gewählten Software, mit der zuvor Blickrichtungsdaten erhoben wurden. Anwender*innen müssen

die Rohdaten dafür zunächst manuell in das von DynAOI genutzte Format bringen. DynAOI ist für zweidimensionale, nicht immersive Videos entwickelt worden und nutzt zweidimensionale Blickrichtungsdaten. Es ist unklar, mit wie viel Entwicklungsaufwand eine Anpassung zur Analyse von AOIs in 360° Videos gestaltet werden könnte.

Dalmaijer et al. (2013) stellen in ihrer Arbeit die Open-Source Bibliothek Pygaze vor. Pygaze ist ein umfangreiches Paket zur Erstellung und Analyse von Eyetracking-Experimenten. Die Bibliothek ist eine Erweiterung der Programmiersprache Python und setzt dementsprechend Programmierkenntnisse voraus. Pygaze ist auf allen gängigen Betriebssystemen nutzbar und unterstützt eine Vielzahl von Eyetracking-Systemen unterschiedlicher Hersteller. Durch die Kombination mit weiteren Paketen sowie der Möglichkeit eigene Funktionalitäten programmatisch zu implementieren, bietet das Tool einen extrem großen Funktionsumfang. Trotz dessen ist die Erstellung eines VR Experiments mit 360° Videos in Pygaze nicht trivial. Die Bibliothek ist vorwiegend für die Darbietung zweidimensionaler Stimuli auf Desktop-Bildschirmen entwickelt worden. Die Implementierung eines Experiments in VR wäre mit erheblichem Entwicklungsaufwand verbunden.

Bonikowski et al. (2021) stellen in ihrer Arbeit ein Open-Source-Programm zur Animation dynamischer AOIs für Videos vor. Im Unterschied zu den anderen vorgestellten Arbeiten nutzen die Autoren Algorithmen zur automatischen Animation der AOIs. Die*der Benutzer*in muss ein Areal also nur ein einzelnes Mal als AOI kennzeichnen. Für alle weiteren Frames erfolgt die Animation der AOI algorithmisch anhand des zugrunde liegenden Videomaterials. Das Programm selbst kann nicht für die Durchführung der Eyetracking-Experimente genutzt werden, sondern stellt die animierten AOIs in Form eines CSV-Datenexports bereit. Im Anschluss an die Versuchsdurchführung kann die*der Anwender*in die erhobenen Blickrichtungsdaten in einem beliebigen Analysetool mit den im Datenexport definierten AOIs abgleichen. Obwohl das Tool nicht für 360° Videos entwickelt wurde, ist es durchaus vorstellbar, dass mit dem Programm AOIs auch für immersive Videos definiert werden könnten. Wenn ein 360° Video in das Programm geladen wird, erscheint dieses in dem Videoplayer als Rektangularprojektion (siehe Sektion 3.4 Rektangularprojektion). Im Anschluss können AOIs als Rechtecke in dem Video positioniert werden. Es ist jedoch unklar, wie gut die Performanz der Algorithmen zur Objekterkennung auf dieser Rektangularprojektion des 360° Videos wäre, da Objekte durch die Projektion stark verzerrt werden. Weiterhin müssten Benutzer*innen die in VR erhobenen dreidimensionalen Blickrichtungsdaten in den zweidimensionalen Raum übersetzen, da die mit dem Tool erstellen AOIs zweidimensional sind.

Tawa et al. (2022) stellen in ihrer Arbeit die Entwicklung eines Open-Source-Tools mit dem gleichen Ziel wie die in dieser Arbeit entwickelte Software vor. In ihrer

Case Study untersuchen sie den Entscheidungsprozess von Polizisten, Schusswaffen einzusetzen, anhand eines 360° Videos. Die Studie zeigt das Potenzial der Technologien, komplexe Fragestellungen in Risikosituationen mit hoher experimenteller Kontrolle zu untersuchen. Der von Tawa et al. (2022) vorgestellte Workflow erfordert das manuelle Aufrufen verschiedener Python-Skripte und Softwarekomponenten, um AOIs zu definieren und Blickrichtungsdaten zu analysieren. Eine Software zur Ansicht von 360° Videos und der eigentlichen Durchführung der Versuche zur Erhebung der Blickbewegungen wird nicht bereitgestellt und muss von der*dem Anwender*in selbst gestaltet werden. Die Arbeit der Autoren bietet keine Funktionalität zur Animation der AOIs durch Interpolation oder Objekterkennung, so dass eine zeitaufwendige Frame-by-Frame Definition der AOIs nötig ist. Die Analyse der Blickrichtungsdaten ist auf Dwells beschränkt. Fixationen im Sinne der in Sektion 3.1.2 vorgestellten Definition von Fixationen werden von den Algorithmen der Autoren nicht detektiert. Weiterhin nutzen die Autoren zur Durchführung ihrer Pilotstudie ein Software Development Kit des Herstellers Tobii, welches eine kostenpflichtige Forschungslizenz benötigt. Im Gegensatz zu der vorgestellten Arbeit zeigt die Case Study von Tawa et al. (2022) konkrete Einsatzmöglichkeiten der ange strebten Software und demonstriert die Kombination von in VR erhobenen Blickbewegungsdaten mit anderen Maßen, wie empfundenem Stress und Reaktionszeiten der Versuchspersonen.

Die vorgestellten Werkzeuge zur Durchführung und Analyse von Eyetracking-Studien sind allesamt für unterschiedliche Anwendungszwecke mit unterschiedlichen Anforderungen entwickelt worden. Bei der Literaturrecherche konnte lediglich eine einzige Veröffentlichung gefunden werden, die ein Open-Source-Werkzeug speziell zur Durchführung von Eyetracking-Studien mit 360° Videos vorstellt (siehe Tawa et al., 2022). Stattdessen konnten mehrfach Arbeiten gefunden werden, die für das Studiensemester eigens programmierte Lösungen verwenden (z. B. M. Brescia-Zapata et al., 2023; Saad et al., 2021). Die geringe Anzahl an relevanter Literatur und Open-Source-Software zeigt, dass weiterhin viel Forschungsbedarf zur Entwicklung von Tools besteht, welche Forschenden zukünftig erlauben, Eyetracking-Experimente in Virtual Reality durchzuführen. Dennoch ist eine Betrachtung bestehender Werkzeuge und deren Funktionalitäten sinnvoll, um Anforderungen an die zu entwickelnde Software abzuleiten. Eine Abgrenzung zu den hier gelisteten Arbeiten und die Darstellung der aus der Literaturrecherche abgeleiteten Anforderungen folgt in Sektion 5.1.

4.2 Kommerzielle Software

Aus dem oben genannten Grund arbeitet eine Vielzahl an Studien mit kommerziell erhältlichen Programmen, welche Lösungen für die Durchführung von Eyetracking-Experimenten in Virtual Reality anbieten. Dazu gehören zum Beispiel Tobii Pro Lab

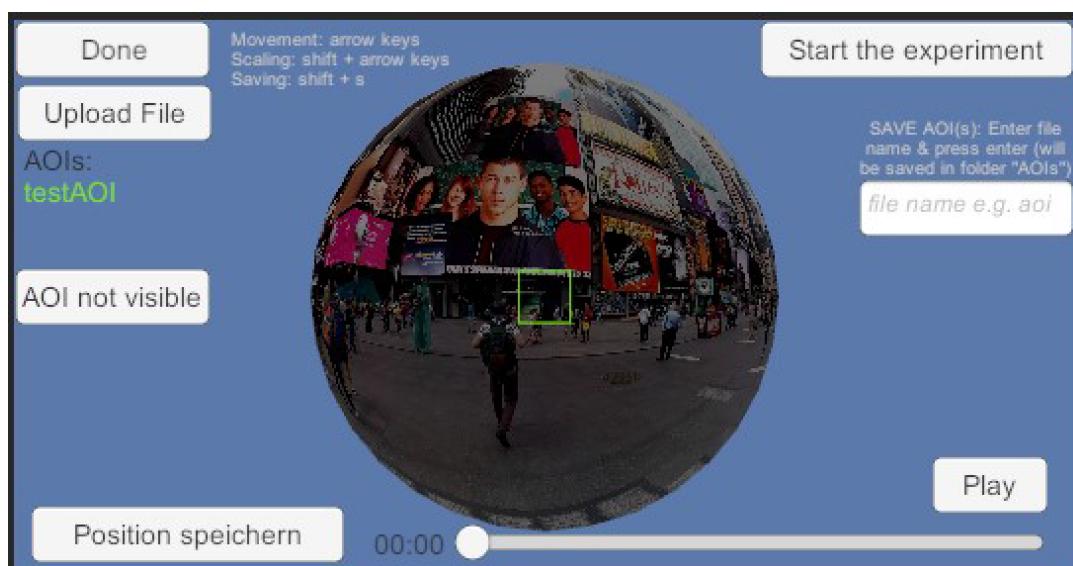
(Tobii Pro Lab, 27/07/2023), iMotions mit dem 'Eye Tracking Virtual Reality' Modul (iMotions, 6/15/2022) sowie Sightlab VR Pro von Worldviz (Sightlab VR Pro, 27/07/2023). Alle drei Applikationen erlauben die Definition von AOIs in 360° Videostimuli, jedoch handelt es sich um kostenpflichtige, proprietäre Software. Beispielsweise kostet die Verwendung von iMotions mit dem nötigen VR Modul für Forschungszwecke 6000 € pro Jahr (iMotions, 10/31/2022). Keiner der genannten Anbieter stellt den Quellcode der Software zur Verfügung, sodass keine Anpassungen entsprechend der eigenen Bedürfnisse und Fragestellung vorgenommen werden können. Dennoch bieten die Werkzeuge einen weitaus größeren Funktionsumfang als die hier vorgestellte Arbeit. Weiterhin gibt es für die kommerziellen Lösungen einen Kundenservice, welcher bei Rückfragen und Problemen behilflich sein kann. Diese Vorteile müssen bei der Auswahl einer Software berücksichtigt werden. Der Funktionsumfang der drei genannten Softwarelösungen wird im Einzelnen nicht vorgestellt und kann auf den entsprechenden Websites nachgelesen werden.

4.3 Bestehende Programmgrundlage

Die bestehende prototypische Programmgrundlage basiert auf der Arbeit von Schicks (2022). Der Prototyp wurde mithilfe der Videospielentwicklungsplattform Unity implementiert und erlaubt die Definition und Animation von AOIs sowie die Aufnahme von Blickbewegungen beim Betrachten eines beliebigen 360° Videos. Eine Abbildung der Benutzeroberfläche des Programms ist in Abbildung 4.1 zu sehen.

Abbildung 4.1

Benutzeroberfläche der prototypischen Programmgrundlage



Anmerkung. Aus Schicks (2022).

Das Programm stellt ein durch die*den Anwender*in gewähltes 360° Video auf einer Kugel dar, welche von außen betrachtet wird. Neue AOIs werden durch einen Klick auf den *Create AOI* Button erstellt. Eine solche AOI ist in der Abbildung 4.1 als grün umrandetes Rechteck zu sehen. Über Tastatureingaben kann die neu erstellte AOI skaliert und die Kugel, die das Video abbildet, rotiert werden, um die AOI an der gewünschten Position im Video zu platzieren.

Ein Keyframe wird durch einen Klick auf den *Position Speichern* Button erstellt. Ein Keyframe speichert die Größe und Position einer AOI zu der derzeitigen Wiedergabe position des Videos (Timestamp) und dient als Ankerpunkt für die Interpolation der AOI für Frames, für die kein Keyframe definiert wurde. Durch die in Abbildung 4.1 rechts unten zu sehenden Bedienelemente, dem *Play* Button und Slider, kann das Video abgespielt, pausiert sowie vor- und zurückgespult werden. Die Lautstärke des Videos kann nicht angepasst werden. Die*der Anwender*in setzt über den zeitlichen Verlauf des Videos verschiedene Keyframes. Für jeden Keyframe sollte die AOI in ihrer Position und Form jeweils so angepasst werden, dass das Zielobjekt im Video überdeckt wird. Nach dem Klick auf den *Done* Button wird die Position der AOI zwischen den gesetzten Keyframes interpoliert, sodass nun bei erneuter Wiedergabe des Videos die Animation der AOI zu sehen ist. Sofern ausreichend viele Keyframes gesetzt wurden, sollte die Animation der AOI das zugrundeliegende Objekt im Video mit hinlänglicher Genauigkeit tracken. In dem Prototyp können Keyframes, nachdem sie definiert worden sind, nicht angepasst werden. Die Benutzeroberfläche besitzt kein Interface zur Einsicht, Bearbeitung oder Löschung von Keyframes und die Animation kann erst überprüft werden, nachdem auf *Done* geklickt wurde. Im Anschluss kann die AOI nicht mehr bearbeitet werden, weswegen AOIs immer in einem Durchlauf vollständig definiert werden müssen. Ein Set aus AOIs kann mithilfe des rechts sichtbaren Textfeldes benannt und anschließend durch einen Druck auf die Enter-Taste gespeichert werden. Ein Klick auf den *Upload File* Button öffnet einen Systemdialog, der es der*dem Anwender*in erlaubt, eine zuvor gespeicherte AOI-Textdatei einzulesen und die darin gespeicherten AOIs zu laden.

Mit einem Klick auf den Button *Start the experiment* wird das Video in VR abgespielt. Das VR-Headset betrachtet die Kugel, auf welche das Video projiziert wird, von innen. Dadurch ist das Video bei der Wiedergabe in der VR-Brille gespiegelt. Während des Experiments werden Blickbewegungen der Versuchsperson aufgezeichnet. Der Prototyp detektiert Dwells und Fixationen auf die definierten AOIs. Dabei wird ein Dwell ab einer Dauer von 150ms als Fixationen gezählt, unabhängig davon, wie viel sich der Blick während dieser Zeit bewegt hat. Aufgrund der konkreten programmativen Implementierung werden die Blickrichtungsdaten mit einer Frequenz von 90 Hz ausgelesen, während der Eyetracker der HTC Vive Pro Eye mit 120 Hz taktet (siehe Anhang C für technische Spezifikationen der VR-Brille). Nach Beendigung eines Versuchs exportiert das Programm die Ergebnisse in Form einer CSV-Datei. Der Prototyp kann unter <https://gitlab.com/franziska.schicks/>

360videoassistent heruntergeladen werden. Das Programm erfordert die Ausführung innerhalb des Unity Editors (Entwicklungsumgebung der Videospiel-Engine Unity). Eine Ausführung als Standalone-Software ist nicht möglich.

Schicks (2022) zeigt mit der Entwicklung des Prototyps, dass eine Umsetzung eines solchen Programms in der Videospiel-Engine Unity möglich ist. Da es sich bei der Implementierung um einen Prototyp handelt, sind sowohl Funktionsumfang als auch Bedienung zum Teil eingeschränkt und sollen im Zuge der vorgestellten Arbeit verbessert werden. Die konkret angestrebten Verbesserungen sind in Sektion 5.1 gelistet.

Kapitel 5

Entwicklung

Dieses Kapitel schildert die Entwicklung der vorgestellten Software. Zunächst werden die aus dem Stand der Forschung sowie der bestehenden Programmgrundlage abgeleiteten Anforderungen gelistet. Im Anschluss wird der Funktionsumfang der Software und der angedachte Arbeitsablauf ausführlich dargestellt. Zuletzt wird auf einige konkrete programmatische Details bezüglich der Implementierung eingegangen und es werden Limitationen sowie zukünftige Verbesserungsmaßnahmen aufgezeigt.

5.1 Anforderungen

Zu Beginn des Projekts erfolgte eine Einarbeitung in die bereits bestehende Programmgrundlage mithilfe der bereitgestellten Dokumentation. Basierend auf den von Schicks (2022) gesammelten Erkenntnisse und vorgestellten Verbesserungsmöglichkeiten sowie der in Kapitel 4 dargestellten Literaturrecherche, wurden die folgenden Anforderungen definiert.

5.1.1 Abgrenzung zu verwandten Arbeiten

Ziel dieser Arbeit ist es, eine kostenlose und modifizierbare Software zur Verfügung zu stellen. Deshalb ist darauf zu achten, dass die eingesetzten Entwicklungsumgebungen, Plugins, Bibliotheken und weitere Dependenzen frei erhältlich sind und in das Programm für Forschungszwecke eingebunden werden dürfen. Die von Schicks (2022) und Tawa et al. (2022) verwendete Entwicklungsumgebung Unity erlaubt die kostenlose Nutzung einer ‚Unity Personal‘ Lizenz, solange mit der Nutzung in Zusammenhang stehende Produkte und Projekte einen jährlichen Umsatz von weniger als 100.000 \$ erzielen (Unity Technologies, 20/07/2023). Problematisch ist jedoch, dass für die Nutzung des von Schicks (2022) und Tawa et al. (2022) verwendeten TobiiXR SDK für akademische Zwecke eine ‚Academic License‘ benötigt wird (Tobii Devzone, 10/07/2023a). Das SDK wurde in beiden Arbeiten verwendet, um in Unity auf das Eyetracking-System zuzugreifen. Um der Anforderung der freien Verfügbarkeit gerecht zu werden, soll die Weiterentwicklung auf die Nutzung dieses

SDK verzichten und stattdessen das offen zur Verfügung stehende SRanipal SDK verwenden (Vive Developers, 20/07/2023).

Weiterhin soll das Programm im Gegensatz zu z. B. Pygaze (Dalmaijer et al., 2013) vollständig ohne Programmierkenntnisse genutzt werden können. Das Tool soll zusätzlich als Standalone-Software fungieren. Das bedeutet, dass für die Ausführung des Programms keine Installation der Unity Entwicklungsplattform benötigt werden soll und alle Funktionalitäten innerhalb einer einzigen Software gebündelt sein sollen. Ein manuelles Aufrufen unterschiedlicher Executables und Skripte wie in der Arbeit von Tawa et al. (2022) soll nicht erforderlich sein.

Im Gegensatz zu den Arbeiten von Papenmeier und Huff (2010), Bonikowski et al. (2021) und Tawa et al. (2022), soll das Programm sowohl für die Definition der AOIs als auch für die eigentliche Versuchsdurchführung genutzt werden können. Das Hinzuziehen einer weiteren Software für den Abgleich zwischen den erstellten AOIs und den erhobenen Blickdaten soll nicht nötig sein.

Anders als in der Arbeit von Tawa et al. (2022) soll die Animation der AOIs, wie bereits von Schicks (2022) demonstriert, durch eine Interpolation zwischen Keyframes unterstützt werden. Eine zeitaufwendige Definition der AOIs für jeden einzelnen Videoframe soll dadurch entfallen. Im Gegensatz zu Bonikowski et al. (2021) ist eine automatische Animation von AOIs durch Algorithmen, wenn auch wünschenswert, in dieser Arbeit nicht vorgesehen.

Zusätzlich soll das Programm nicht nur Dwells, sondern auch Fixationen auf AOIs bestimmen können. Dabei sollen Fixationen nicht wie in den Arbeiten von Schicks (2022) und Tawa et al. (2022) nur anhand eines Zeitkriteriums, sondern auch anhand einer Streungs- und Geschwindigkeitsgrenzwerts klassifiziert werden. Wie beschrieben sind Fixationen Zeiten relativen Stillstands der Augen. Das Hinzuziehen dieser Kriterien erlaubt zu untersuchen, ob auf Grundlage der gemessenen Eyetracking-Daten geschlussfolgert werden kann, dass der Blick derzeit relativ unbewegt ist. Erst dadurch wird sichergestellt, dass es sich bei den detektierten Fixationen um Fixationen im eigentlichen Sinne handelt (siehe 3.1.2).

Im Unterschied zu dem Großteil der in Kapitel 4 vorgestellten Arbeiten soll die Software speziell für die Nutzung von 360° Videos gedacht sein. Eine Unterstützung von nicht-immersivem Video- und Bildmaterial ist nicht angedacht.

5.1.2 Angestrebte Verbesserungen gegenüber dem Prototyp

Auch wenn eine optimierte Benutzeroberfläche keine grundlegende Anforderung an die vorgestellte Arbeit ist, soll das Interface des Prototyps verbessert werden. Vor allem die Darstellung des Videos auf einer Kugel sowie die Bearbeitung von AOIs

per Tastatureingaben sollen angepasst werden, um AOIs einfacher und präziser im Video positionieren zu können. Weiterhin soll das Programm dahin gehend verbessert werden, dass die Bearbeitung von AOIs und Keyframes jederzeit möglich sein soll, sodass eventuell aufgetretene Fehler im Nachhinein problemlos angepasst werden können.

Um die von dem Programm erzielte Datenqualität zu erhöhen, soll das Eyetracking-System in der Weiterentwicklung mit den vollen 120 Hz ausgelesen werden, anstatt der derzeitigen 90 Hz (siehe Beschreibung der HTC Vive Pro Eye in Anhang C). Neben der bereits beschriebenen besseren Klassifikation von Fixationen, sollen Einstellungen zur Verfügung gestellt werden, die es Anwender*innen erlauben, Grenzwerte für die Fixationsdetektion selbst entsprechend dem eigenen Anwendungskontext zu definieren.

Weiterhin existieren einige Bugs in dem Prototyp. So sind die Eyetracking-Daten in den ersten 150 ms jedes Experiments invalide und das Abspielen der Animation von AOIs funktioniert nur unter gewissen Voraussetzungen (Schicks, 2022). Diese Fehler sollen in der Weiterentwicklung nicht mehr auftreten.

Abgesehen von den genannten, bereits zu Beginn des Projekts geplanten Anforderungen, wurde eine Vielzahl weiterer Anpassungen und Verbesserungen vorgenommen. Die vollständige Beschreibung der Software erfolgt in Sektion 5.2.1.

5.1.3 Funktionalitäten außerhalb des Projektumfangs

Um den Umfang des Projekts in einem angemessenen Rahmen zu halten, wurden einige Funktionalitäten definiert, die nicht Bestandteil der vorgestellten Arbeit sein sollen. Sie sind primär im Rahmen der Betrachtung zukünftigen Entwicklungspotenzials interessant.

Eine umfangreiche Optimierung der Gebrauchstauglichkeit der Software steht nicht im Vordergrund des Projekts. Dementsprechend soll keine Validierung der Gebrauchstauglichkeit vorgenommen werden.

Die von der Software bestimmten Blickbewegungsparameter sollen sich auf Parameter bezüglich Dwells und Fixationen beschränken. Sakkaden oder Smooth Pursuit Bewegungen sollen von dem Tool nicht klassifiziert werden, um den Umfang der Arbeit in einem vernünftigem Maß zu halten.

Die Visualisierung von Blickbewegungen auf dem Stimulus, z. B. in Form von Heatmaps oder Scanpaths, ist durch das Programm derzeit nicht vorgesehen.

Weiterhin soll das Programm zu diesem Zeitpunkt nicht dazu in der Lage sein, Anwender*innen bei der statistischen Auswertung oder der Analyse von Parametern über mehrere Versuchsdurchläufe hinweg zu unterstützen.

5.1.4 Vorgehensweise

Nach der Einarbeitung in den bestehenden Prototyp wurden die Arbeitsschritte für die Weiterentwicklung geplant. Da die Programmgrundlage zeigt, dass eine Implementierung eines AOI Editors in Unity grundsätzlich möglich ist, wurde die Unity Plattform weiterhin als Entwicklungsumgebung für das Projekt gewählt. Da die Implementierung des Prototyps jedoch aufgrund der Programmstruktur einigen Limitationen unterlag, wurde sich dazu entschieden, das Programm von Grund auf neu zu programmieren. Dies erlaubte, diese Einschränkungen durch andere programmatiche Ansätze zu umgehen. Weiterhin ermöglichte der Neustart eine bessere Einarbeitung in die Entwicklungsumgebung, ohne die Erfordernis, eine bestehende Co-debase zu verstehen. Die von Schicks gesammelten Erkenntnisse und vorgestellten Verbesserungen wurden dennoch für die Entwicklung berücksichtigt und eingearbeitet.

5.2 Vorstellung der entwickelten Software

5.2.1 Funktionsumfang

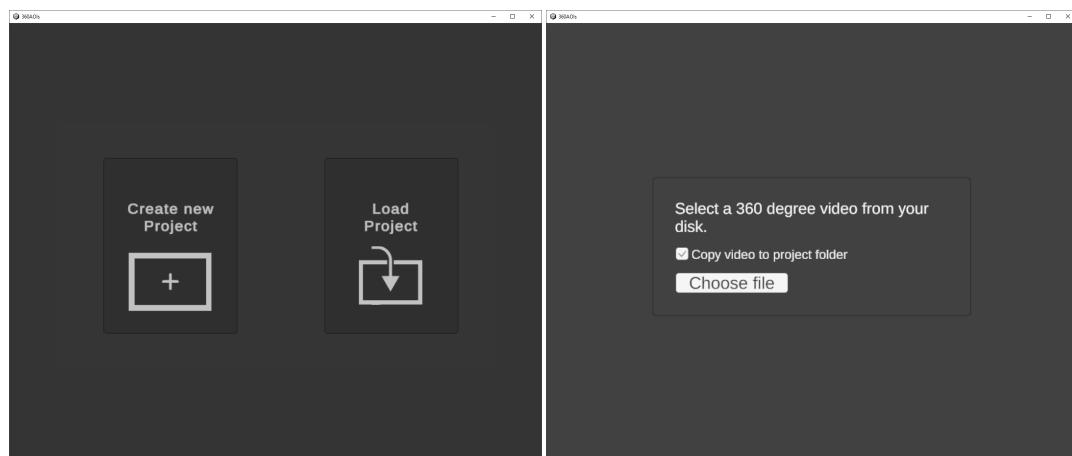
In dieser Sektion wird der Funktionsumfang und der damit einhergehende Arbeitsablauf der Software vorgestellt. Das Kapitel bietet eine vollständige Übersicht aller Funktionen und erläutert die Beweggründe hinter den getroffenen Entscheidungen.

5.2.1.1 Projektmanagement

Der erste Screen, der einer*einem Benutzer*in bei dem Start des Programms begegnet, ist in 5.1a abgebildet. Er stellt die*den Nutzer*in vor die Wahl, ein neues Projekt zu erstellen oder ein bestehendes Projekt zu laden.

Abbildung 5.1

Erste Schritte nach Ausführung des Programms



(a) Startbildschirm

(b) Dialog für den Upload des 360° Videos

Anmerkung. Die Bedienelemente wurden zwecks besserer Darstellung vergrößert abgebildet.

In dem Kontext des vorgestellten Programms ist ein Projekt ein Ordner auf dem System der*des Benutzer(s)*in, welcher bestimmte Dateien und Unterordner beinhaltet, die von dem Programm erkannt, eingelesen und bearbeitet werden können. Das Projekt-Konzept erlaubt es, ein Video samt den dafür definierten AOIs in einem einzigen Schritt zu laden, das Projekt einfach mit Kollaborateuren zu teilen und eine übersichtliche Sammlung der erstellten Exporte für jeden Versuchsdurchlauf an einem zentralen Ort zu haben. Ein Projekt-Ordner enthält (mit wenigen Ausnahmen) die folgenden Elemente:

- **project_settings.txt:** Eine Textdatei, welche die für das Laden des Projekts relevanten Informationen enthält (z. B. Dateipfad des Videos).

- **Results:** Ein Ordner, in welchem die Ausgaben der Versuchsdurchläufe gespeichert werden. Das Programm erstellt für jeden Versuch einen eigenen Unterordner in diesem Verzeichnis.
- **AOI Save-File:** Eine Textdatei gespeicherter AOIs im JSON-Dateiformat. Die Datei muss sich nicht zwangsläufig in dem Projekt-Ordner befinden. Um ein Projekt einfach teilen und verschieben zu können, wird jedoch empfohlen, die Datei in dem Projekt-Ordner abzulegen.
- **360° Video:** Sofern von der*dem Benutzer*in gewünscht, enthält der Ordner eine Kopie des 360° Videos, welches für das Projekt verwendet wird.

Um ein bestehendes Projekt zu laden, wählt die*der Anwender*in die entsprechende Schaltfläche (siehe rechts in Abb. 5.1a) und klickt danach in einem Windows-Systemdialog auf einen existierenden Projekt-Ordner. Im Anschluss wird das Projekt samt 360° Video und bereits angelegten AOIs geladen.

Entscheidet sich die*der Benutzer*in stattdessen dazu, ein neues Projekt zu erstellen, wird sie*er dazu aufgefordert, das 360° Video für das Projekt hochzuladen (siehe Abb. 5.1b). Standardmäßig wird eine Kopie der ausgewählten Videodatei in dem Projekt-Ordner abgelegt, wodurch das Projekt portabler ist, da so alle Dateien in einem einzigen Verzeichnis gelagert werden. Sollte dies nicht gewünscht sein, kann die*der Benutzer*in diese Option abwählen.

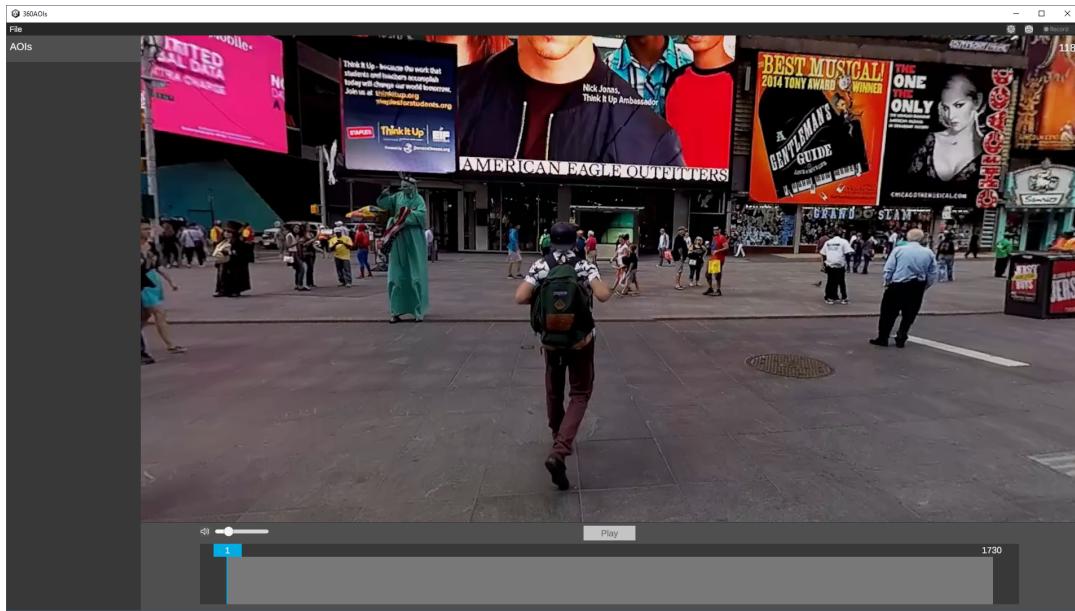
5.2.1.2 AOI Erstellung und Management

Nachdem das Video hochgeladen wurde, erscheint die in 5.2 abgebildete Benutzeroberfläche. Der Großteil der Benutzeroberfläche wird von einem Ausschnitt des 360° Videos eingenommen. Die*der Benutzer*in kann den sichtbaren Ausschnitt des 360° Videos anpassen, also ‚sich umsehen‘, in dem sie*er die rechte Maustaste über dem Video gedrückt hält und dabei die Maus bewegt. Der sichtbare Ausschnitt ändert sich dann entsprechend der Mausbewegung. Diese Darstellung und Interaktion wurde gewählt, da sie bei der Wiedergabe von 360° Inhalten bereits Standard bei großen Plattformen wie YouTube (YouTube VR, 22/07/2023) und dem in Microsoft Windows inkludiertem Videoplayer ist (Pidgeon, 2017). Die Nutzung bekannter Interaktionen reduziert die Lernkurve des Programms (Nielsen Norman Group, 08/01/2021). Weiterhin sorgt diese Art der Darstellung dafür, dass das Bild gegenüber dem Prototyp wesentlich weniger verzerrt ist, da es nicht auf eine Kugel projiziert wird (siehe 4.3).

Durch einen Linksklick innerhalb des Videos wird eine neue AOI erstellt und im Video eingeblendet. Zeitgleich wird ein neuer Eintrag in der Liste aller AOIs (*AOI List*) angelegt, welche sich links im UI befindet. Durch einen Doppelklick auf den Listen-Eintrag kann die AOI beliebig umbenannt werden. Eine beschriftete Abbildung

Abbildung 5.2

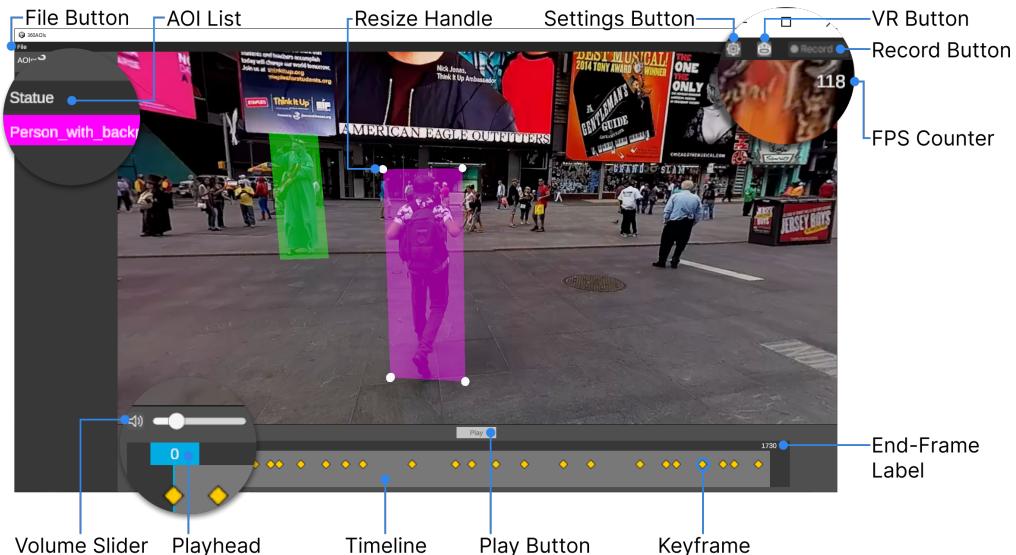
Benutzeroberfläche des Programms nach Anlegen eines neuen Projekts



aller relevanten Bedienelemente ist in Abb. 5.3 dargestellt. Die Abbildung zeigt unter anderem eine derzeit ausgewählte AOI in Magenta sowie den dazugehörigen Listeneintrag, welcher ebenfalls in Magenta eingefärbt ist.

Bestehende AOIs können in ihrer Größe und Position angepasst werden, um ein Objekt oder Areal innerhalb des Videos abzudecken. Um eine AOI zu bewegen, hält die*der Benutzer*in die linke Maustaste über der AOI gedrückt und bewegt dabei die Maus. Die Position der AOI folgt dann dem Mauszeiger. Die Form einer AOI kann angepasst werden, indem die kreisförmigen Eckpunkte der AOI (*Resize Handle* in Abbildung 5.3) auf die gleiche Art bewegt werden. Durch die Bewegung eines Eckpunkts verformt sich die Fläche der AOI, wodurch diese vergrößert oder verkleinert werden kann. Die beiden Interaktionen sind in Abb. 5.4 dargestellt.

AOIs können nach Auswahl entweder durch einen Tastendruck auf die Entfernen-Taste oder per Rechtsklick → *Delete* gelöscht werden. Ein Rechtsklick auf eine AOI öffnet ein Kontextmenü an der Position des Mauszeigers (siehe Abb. 5.6). Das Kontextmenü enthält neben der Option zum Löschen der AOI eine Schaltfläche, mit der die AOI als unsichtbar markiert werden kann. Diese Funktion kann genutzt werden, um eine AOI temporär inaktiv zu setzen, um Zeiten zu überbrücken, in welchen das zugrundeliegende Zielobjekt nicht im Video sichtbar ist. Um die angelegten AOIs zu speichern oder um gespeicherte AOIs zu laden, klickt die*der Benutzer*in oben links auf den *File* Button. Danach erscheint das in Abbildung 5.7a dargestellte Menü mit den Optionen *Save AOIs* und *Load AOIs*, welche jeweils einen Systemdialog öffnen, in dem eine Datei zum Speichern oder Laden ausgewählt werden kann. Um das Projekt einfach per Dateiübertragung des Projekt-Ordners teilen zu können, empfiehlt

Abbildung 5.3*Annotierte Benutzeroberfläche der Software*

Anmerkung. Bestimmte Bereiche der Benutzeroberfläche wurden zwecks besserer Darstellung vergrößert abgebildet.

es sich, die AOI-Datei stets in dem Projekt-Ordner abzulegen.

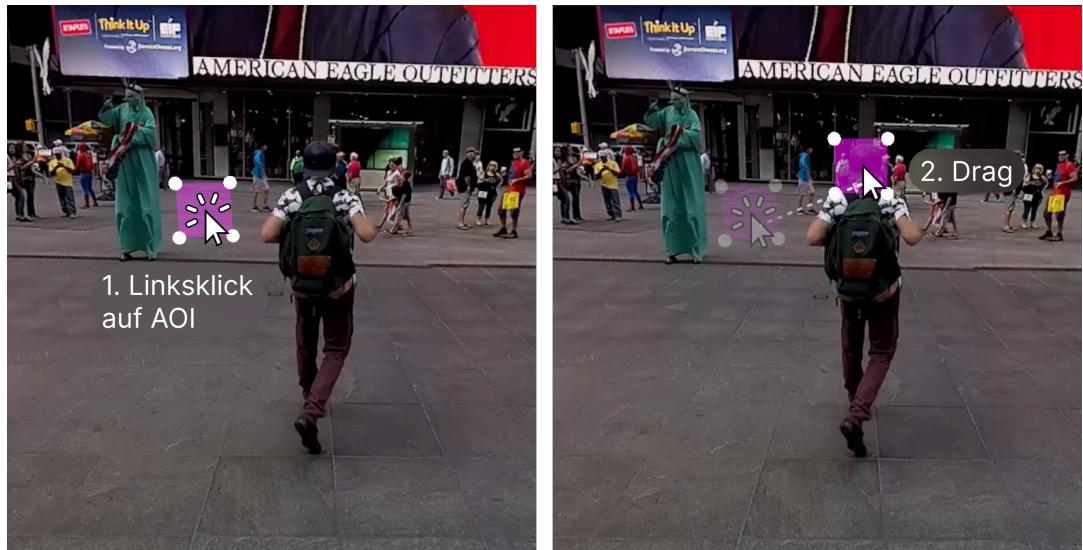
Zusammengefasst können AOIs erstellt, benannt, positioniert und in ihrer Form und Größe angepasst werden, um Areale im Video zu überdecken. Die Interaktionen wurden so gestaltet, dass die Notwendigkeit für die Nutzung der Tastatur auf ein Minimum reduziert wurde. Wo möglich, wurden bekannte Interaktionen anderer Softwarelösungen übernommen, um das Programm intuitiv bedienbar zu machen. Einzelne AOIs können temporär deaktiviert oder gelöscht werden, während alle AOIs gemeinsam gespeichert oder bereits gespeicherte AOIs geladen werden können.

5.2.1.3 Videowiedergabe und AOI Animation

Alle Bedienelemente für die Steuerung der Videowiedergabe befinden sich im unteren Teil der Benutzeroberfläche. Das Video kann per Klick auf den *Play* Button oder per Tastendruck auf die Leertaste gestartet und pausiert werden. Der Lautstärke-Slider, gekennzeichnet durch das Lautsprecher-Symbol, dient der Anpassung der Lautstärke des Videos. Direkt unter dem *Play* Button befindet sich die Timeline. Die Timeline ist der Bereich der Benutzeroberfläche, in dem die Keyframes der aktuell ausgewählten AOI dargestellt werden. Der blaue Playhead zeigt die aktuelle Wiedergabeposition des Videos auf der Timeline an. Dabei wird die Wiedergabeposition nicht als Timestamp, sondern als Frame abgebildet, da die Timeline bei der Erstellung und dem Management von Keyframes behilflich sein soll. Durch einen Klick auf die Timeline kann zu einem gewünschten Frame im Video navigiert werden. Da

Abbildung 5.4

Interaktionen zur Anpassung von AOIs



(a) Interaktion zur Positionierung von AOIs



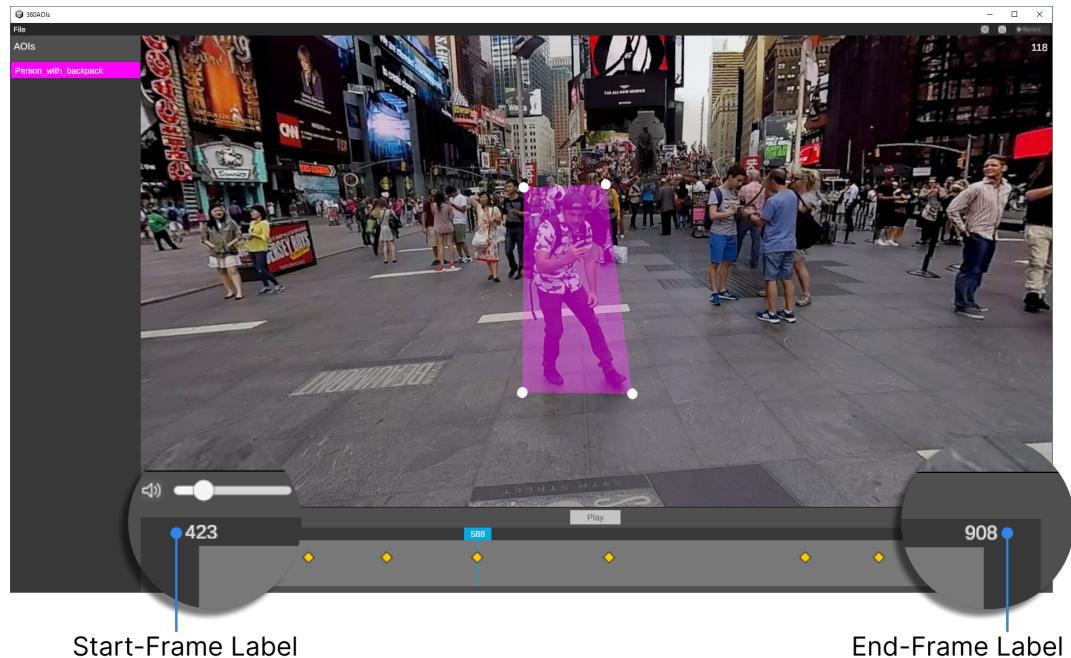
(b) Interaktion zur Anpassung der Form und Größe von AOIs

Videos bereits bei wenigen Minuten Länge aus tausenden Frames bestehen, kann es sich schwierig gestalten, zu einem gewünschten Frame zu navigieren. Aus diesem Grund kann die Skalierung der Timeline durch den Benutzer angepasst werden, bzw. in die Timeline „hineingezoomt“ werden, sodass die Auswahl eines einzelnen Frames einfacher ist. Dafür wird der Mauszeiger über der Timeline positioniert und per Drehung des Mausrads die Auflösung der Timeline erhöht oder verringert. Eine skalierte Timeline ist in Abbildung 5.5 dargestellt. In dem Beispiel bildet die Timeline nach dem Zoom nicht mehr die gesamte Länge des Videos von Frame 0 bis Frame 1730 ab, sondern stellt einen Ausschnitt des Videos von Frame 423 bis 908 dar. Da nun mehr Fläche für weniger Frames bereitsteht, ist die Navigation zu einem

einzelnen Frame stark vereinfacht.

Abbildung 5.5

Skalierte Timeline („gezoomt“)



Anmerkung. In der skalierten Timeline entspricht der Start-Frame nicht dem ersten Frame des Videos und der End-Frame nicht dem letzten Frame des Videos. Stattdessen wird ein zeitlicher Ausschnitt des Videos auf der Timeline dargestellt.

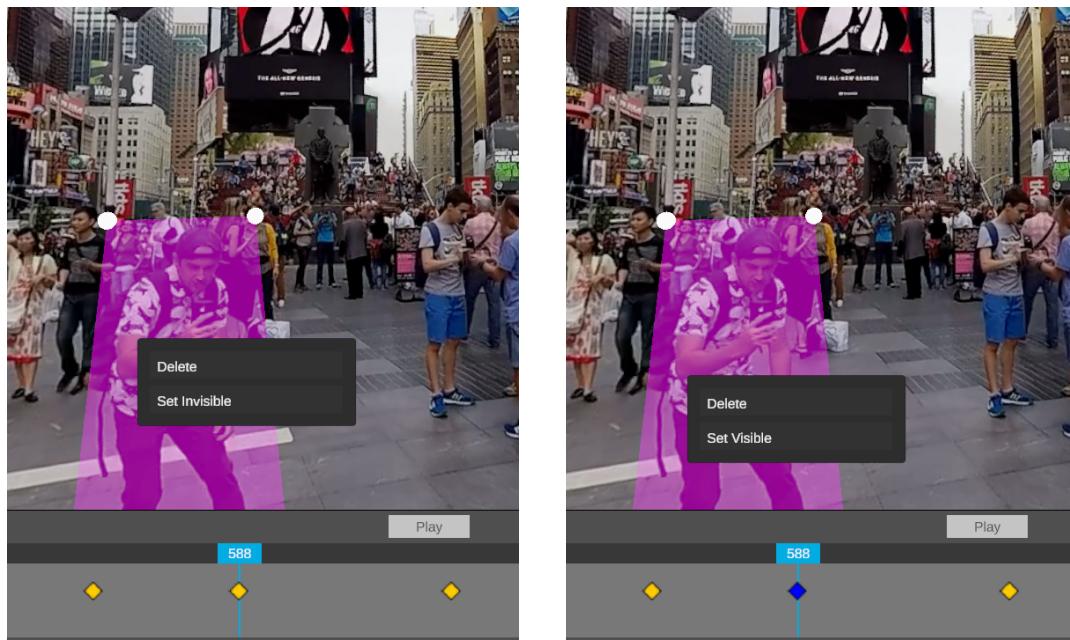
Immer wenn die Position oder Größe einer AOI durch die*den Benutzer*in angepasst wird, wird diese Anpassung für den aktuellen Frame (gekennzeichnet durch die Position des blauen Playheads) als Keyframe gespeichert. Die Timeline stellt alle Keyframes der derzeit ausgewählten AOI jeweils mit einem Rauten-Icon dar (siehe Abb. 5.3 und 5.5). Sollte die*der Benutzer*in eine Anpassung einer AOI für einen Frame vornehmen, für den bereits ein Keyframe existiert, wird dieser automatisch aktualisiert. Nach jeder Interaktion wird die Form und Position der AOI für alle Frames des Videos kalkuliert. Dies geschieht anhand linearer Interpolation zwischen den benutzerdefinierten Keyframes. Wenn die*der Anwender*in ein Objekt im Video mit einer dynamisch AOI tracken möchte, funktioniert dies also wie folgt:

1. Navigation zu dem ersten Videoframe, in dem das Zielobjekt sichtbar ist.
2. Erstellung einer neuen AOI per Linksklick.
3. (optional) Umbenennung der AOI mit einem eindeutig zuordenbaren Namen per Doppelklick auf Listeneintrag.
4. Anpassung der Position und Größe der AOI, sodass sie das Zielobjekt bestmöglich überdeckt. Hierbei wird automatisch ein Keyframe erstellt.

5. Wiedergabe oder Vorspulen im Video, bis sich die Größe oder Position des Zielobjekts im Video hinreichend verändert hat.
6. Erneute Anpassung der Position und Größe der AOI, sodass sie das Zielobjekt auch zu diesem Zeitpunkt im Video bestmöglich überdeckt. In diesem Moment wird erneut automatisch ein Keyframe erstellt.
7. Wiederholung der Schritte 5 und 6, bis das Video vorbei ist oder das Zielobjekt nicht mehr zu sehen ist. In letzterem Fall kann die AOI durch *Rechtsklick -> Set Invisible* deaktiviert werden.
8. Überprüfung der Interpolation durch Wiedergabe des Videos von Anfang bis Ende. Während der Wiedergabe ist die Animation der AOI zu sehen.
9. (Bei Bedarf) Setzen weiterer Keyframes zwischen den bereits erstellten Keyframes, wie in Schritt 5 und 6 beschrieben, um die Genauigkeit der Animation zu verbessern.

Nach Durchlaufen dieses Prozesses hat die*der Benutzer*in eine dynamische AOI erstellt und vollständig animiert. Sollte ein Keyframe unerwünscht erstellt worden sein, kann dieser durch einen Rechtsklick auf das entsprechende Rauten-Icon wieder gelöscht werden. Besonderer Wert wurde darauf gelegt, dass die*der Benutzer*in die Animation jederzeit editieren kann, um Fehler zu korrigieren und das Tracking zu verbessern. Durch die grafische Darstellung von Keyframes auf der Timeline ist stets eine Übersicht darüber gegeben, ob und mit welchem Detailgrad eine AOI bereits animiert wurde.

Wie bereits angeschnitten kann eine AOI als ‚unsichtbar‘ gekennzeichnet werden, für den Fall, dass das Zielobjekt nicht über die gesamte Länge des Videos sichtbar ist. Dies geschieht, indem die*der Benutzer*in in dem Frame, in dem das Objekt das Video verlässt, einen Rechtsklick auf die AOI tätigt und anschließend auf *Set Invisible* klickt. Der Keyframe in der Timeline wird nach dem Klick blau eingefärbt, wodurch gekennzeichnet wird, dass die AOI ab diesem Zeitpunkt nicht mehr auf das Eyetracking reagiert (siehe 5.6b). Falls die AOI zu einem späteren Zeitpunkt im Video wieder aktiviert werden soll, kann sie per Rechtsklick → *Set Visible* erneut als sichtbar gekennzeichnet werden.

Abbildung 5.6*Rechtsklick-Kontextmenü von AOIs*

(a) Kontextmenü mit Optionen Delete und Set Invisible

(b) Kontextmenü mit Optionen Delete und Set Visible

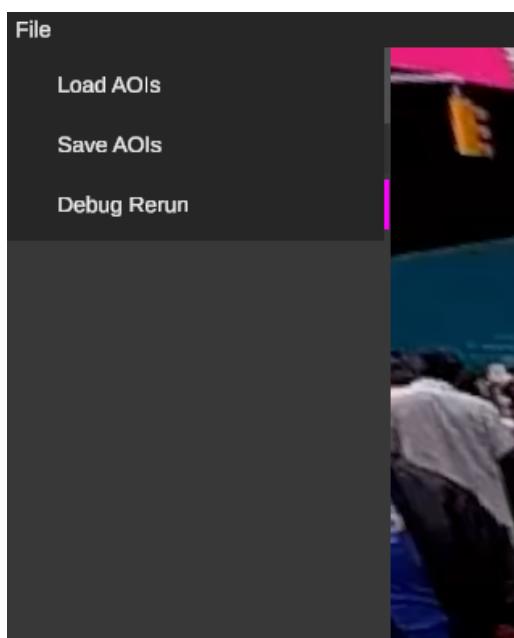
Anmerkung. Das Menü erlaubt, AOIs zu löschen und für das Eyetracking als „unsichtbar“ oder „sichtbar“ zu markieren.

5.2.1.4 Versuchsvorbereitung

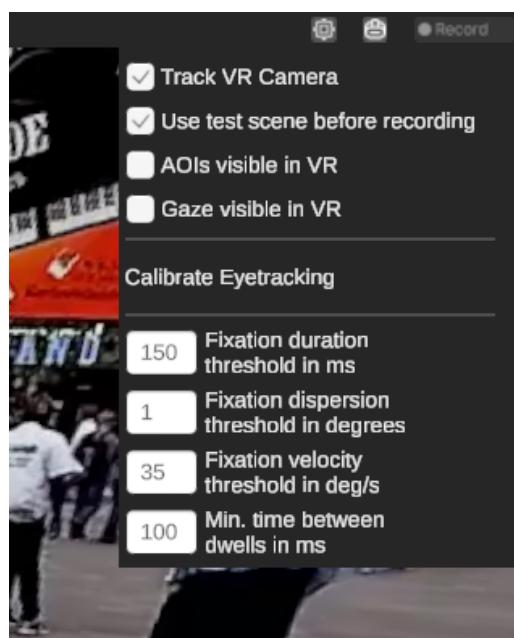
Nachdem alle AOIs angelegt und animiert wurden, ist der nächste Schritt, die Durchführung der Versuche vorzubereiten. Im oberen rechten Teil der Benutzeroberfläche befindet sich ein Button mit einem Icon einer VR-Brille (fortan VR Button genannt) sowie ein *Settings* Button, gekennzeichnet durch ein Zahnrad-Icon (siehe Abb. 5.3). Der VR Button aktiviert alle Hintergrundprozesse, die für die Bildausgabe an die VR-Brille zuständig sind. Sofern die VR-Funktionalitäten nach Klick auf den Button erfolgreich gestartet worden sind, wird dieser grün eingefärbt. Ab diesem Zeitpunkt wird von dem Programm ein Bild an die HTC Vive PRO Eye ausgegeben. Ein Klick auf den *Settings* Button öffnet ein Menü. Dieses enthält Einstellungen zum Test der angelegten AOIs, zur Durchführung und Überprüfung der Eyetracking-Kalibrierung sowie für bestimmte Grenzwerte, die für die Detektion von Dwells und Fixationen genutzt werden. Das Menü ist in 5.7b abgebildet.

Abbildung 5.7

Die beiden Untermenüs des Programms



(a) File Menü zum Laden und Speichern von AOIs.



(b) Settings Menü zur Versuchsvorbereitung sowie weiteren Einstellungen.

Für das Verständnis dieser Optionen ist wichtig anzumerken, dass die Versuchsperson, welche die VR-Brille trägt und die Versuchsleitung, die auf den Monitor blickt, zwei getrennte Ansichten präsentiert bekommen. In der sogenannten Versuchsleitungsansicht wird weiterhin die Timeline, AOI-Liste, das Video mit allen darin erstellten AOIs sowie die derzeitige Blickrichtung der Versuchsperson angezeigt. Im Gegensatz dazu sieht die Versuchsperson in VR zunächst lediglich eine Testszene

oder das 360° Video. Standardmäßig sind das UI, AOIs und die eigene Blickrichtung für die Versuchsperson nicht sichtbar.

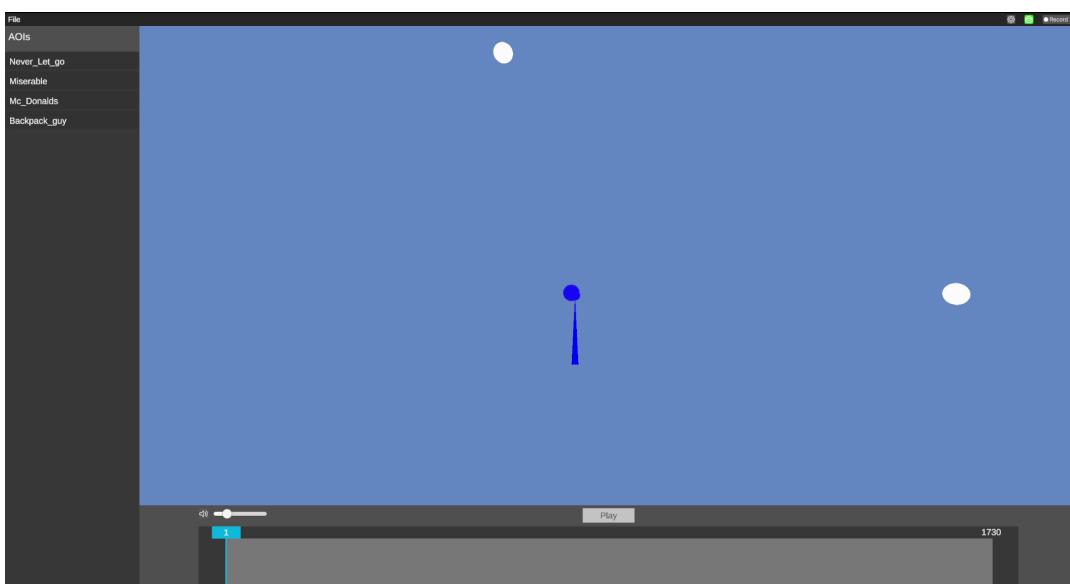
Die genauen Menüoptionen des Settings-Menüs lauten wie folgt:

Track VR Camera: Toggle, Default On Wenn die Option ‚Track VR Camera‘ aktiviert ist, wird in der Versuchsleitungsansicht immer der Ausschnitt des 360° Videos abgebildet, den die Versuchsperson gerade ansieht. Die Kopfbewegungen der Versuchsperson werden also auf die Kamera der Versuchsleitungsansicht übertragen. Wenn die Option deaktiviert ist, kann sich die Versuchsleitung unabhängig von der Versuchsperson per Rechtsklick innerhalb des Videos umsehen.

Use Test Scene: Toggle, Default On Es kann gegebenenfalls unerwünscht sein, dass die Versuchsperson sofort den ersten Frame des Videos sieht, sobald sie die VR-Brille aufsetzt, obwohl der Versuch noch nicht gestartet wurde. Dies würde der Versuchsperson bereits die Chance geben, sich die Umgebung anzusehen und mit dem Video vertraut zu machen, ohne dass Blickbewegungsdaten bezüglich der AOIs aufgezeichnet werden. Wenn die Option *Use Test Scene* aktiviert ist, sieht die Versuchsperson statt des Videos zunächst eine Testumgebung. Die in 5.8 abgebildete Testumgebung enthält Kugeln, die ihre Farbe ändern, wenn der Blickrichtungsvektor auf sie fällt. Diese Testszene erlaubt es, die Kalibrierung zu überprüfen und verbirgt das 360° Video so lange, bis der Versuch gestartet wird.

Abbildung 5.8

Testszene aus Sicht der Versuchsleitung



Anmerkung. Die Blickrichtung der Versuchsperson ist als blauer Blickrichtungsvektor abgebildet und ‚trifft‘ eine der Test-Kugeln, welche sich deshalb auch blau einfärbt.

AOIs visible in VR: Toggle, Default Off Wenn diese Option aktiviert ist, sind die AOIs in der VR-Umgebung sichtbar. Diese Option kann genutzt werden, um zu überprüfen, ob die AOIs aus Perspektive der Versuchsperson korrekt positioniert sind.

Gaze visible in VR: Toggle, Default Off Die Aktivierung dieser Einstellung bewirkt, dass der Blickrichtungsvektor in der VR-Umgebung sichtbar wird. Die Funktion kann verwendet werden, um die Versuchsperson zu fragen, ob sie das Gefühl hat, dass der Blickrichtungsvektor ihre eigene Blickrichtung gut repräsentiert. Unabhängig von dieser Einstellung wird der Blickrichtungsvektor permanent in der Versuchsleitungsansicht eingezeichnet. In Abbildung 5.8 ist dieser als blauer Strahl zu sehen.

Calibrate Eyetracking: Button Ein Klick auf diese Schaltfläche startet die Eyetracking-Kalibrierung des SRanipal SDK. Die Fünf-Punkt-Kalibrierung erfolgt in einem separaten Tool, das nach Klick auf den Button ausgeführt wird. Nach Beendigung der Kalibrierung kehrt die Versuchsperson automatisch zurück zu der VR-Ansicht der hier vorgestellten Software. Es ist zu empfehlen, die Kalibrierung vor jedem Versuchsdurchlauf durchzuführen (Holmqvist et al., 2011).

Fixation duration threshold in ms: TextInput, numeric, Default 150 ms Der Wert, der in dieses Textfeld eingetragen wird, bestimmt die minimale Fixationsdauer in Millisekunden. Sofern die Blickbewegungen der Versuchsperson für diese Dauer innerhalb des Streuungs- und Geschwindigkeitsgrenzwerts liegen, wird der Blick als Fixation klassifiziert.

Fixation dispersion threshold in degrees: TextInput, numeric, Default 1° Dieser Wert dient der Festlegung des Streuungs-Kriteriums. Er bestimmt den maximalen Winkel, der zwischen den Blickrichtungen aufeinanderfolgender Datenpunkte einer potenziellen Fixation liegen darf, damit diese gemeinsam als Fixation klassifiziert werden.

Fixation velocity threshold in °/s: TextInput, numeric, Default 35°/s Dieser Wert dient der Festlegung des Geschwindigkeits-Kriteriums. Er bestimmt, wie schnell die Geschwindigkeit der Blickbewegungen aufeinanderfolgender Data-Samples sein darf, um eine Fixation bilden zu können.

Min. time between dwells in ms: TextInput, numeric, Default 100 ms Dieser Wert bestimmt, wie viel Zeit zwischen zwei AOI Hits auf die gleiche AOI vergangen sein muss, damit der zweite AOI Hit als Beginn eines neuen Dwells gezählt wird. Dieser Wert sollte entsprechend hoch eingestellt werden, damit kurzzeitige Aussetzer

von AOI Hits, die z. B. durch Blinzeln oder invalide Datenpunkte auftreten, nicht fälschlicherweise neue Dwells erzeugen.

Ein typischer Ablauf für die Versuchsvorbereitung mit der vorgestellten Software, unter der Annahme, dass alle AOIs bereits angelegt worden sind, sieht wie folgt aus:

1. Über einen Klick auf den VR-Button wird das Unity XR-Framework gestartet und die Bildausgabe an die VR-Brille eingeleitet.
2. Die Versuchsleitung trägt die gewünschten Fixations- und Dwellgrenzwerte in die Textfelder des *Settings*-Menüs ein.
3. Die Versuchsperson setzt die VR-Brille auf.
4. Die Versuchsleitung startet die Kalibrierung über das *Settings*-Menü.
5. Die Versuchsperson folgt den Anweisungen der Kalibrierung.
6. Die Versuchsleitung und Versuchsperson überprüfen das Ergebnis der Kalibrierung anhand der Testszene sowie dem Blickrichtungsvektor. Gegebenenfalls lässt die Versuchsleitung den Blickrichtungsvektor kurzzeitig in der VR-Ansicht anzeigen, damit die Versuchsperson ihren subjektiven Eindruck bezüglich der Genauigkeit des Eyetrackings schildern kann.
7. Sofern die Kalibrierung zufriedenstellende Ergebnisse liefert, kann der Versuch nun gestartet werden. Andererseits empfiehlt es sich, die Schritte 4 bis 6 zu wiederholen.

5.2.1.5 Versuchsdurchführung

In der oberen rechten Ecke der Benutzeroberfläche befindet sich der *Record* Button. Ein Klick auf den Button bewirkt, dass statt der Testszene das 360° Video in der VR-Ansicht dargestellt wird. Das Video wird automatisch auf den ersten Frame zurückgesetzt und abgespielt. Bis zu dem Zeitpunkt, an dem die Versuchsleitung den Versuch durch einen erneuten Klick auf den *Record* Button beendet, werden die Blickbewegungen der Versuchsperson aufgezeichnet und auf AOI Hits hin überprüft. Derzeit sorgt das Erreichen des Endes des Videos nicht sofort für eine Beendigung des Versuchs. Der Versuch muss manuell beendet werden.

5.2.1.6 Datenexport

Nachdem die Versuchsleitung den Versuch beendet, werden die Blickbewegungsdaten zunächst bereinigt. invalide Blickrichtungsdaten werden von dem SRanipal SDK als solche gekennzeichnet und entfernt. Aus den gefilterten Blickbewegungsdaten und AOI Hits werden Fixations- und Dwellparameter für die angelegten AOIs

berechnet. Im Anschluss wird eine Visualisierung des Versuchsdurchlaufs in Form eines Sequenzdiagramms erstellt. Die Rohdaten, gefilterten Daten sowie die daraus bestimmten Parameter und Visualisierungen werden in einem neuen Ordner in dem Verzeichnis *Projekt-Ordner\Results* abgelegt. In Abbildung 5.9 ist ein Beispiel eines solchen CSV-Datenexports der Parameter zu sehen. Abbildung 5.10 zeigt das automatisch generierte Sequenzdiagramm für denselben Versuchsdurchlauf.

Abbildung 5.9

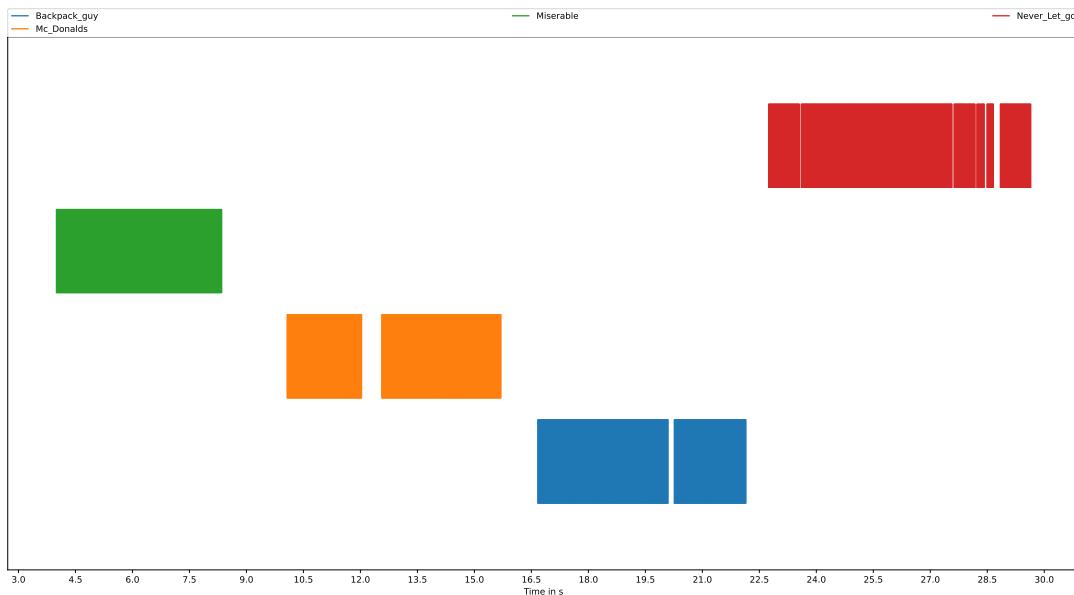
Beispiel CSV-Datenexport

AOI	Dwell_Count	Entry_Time	First_Pass_Dwell_Time	Total_Dwell_Time	Proportion_Of_Total_Dwell_Time
null	15	0	891	10021	0.3182887
Miserable	1	3999	4349	4349	0.1381337
Mc_Donalds	2	10073	1958	5091	0.1617012
Backpack_guy	2	16671	3425	5308	0.1685936
Never_Let_go	2	22745	5915	6715	0.2132829

AOI	Fixation_Count	Time_To_First_Fixation	Duration_Of_First_Fixation	Proportion_Of_Fixations
null	9	0	508	0.2307692
Miserable	8	4641	175	0.2051282
Mc_Donalds	10	10589	284	0.2564103
Backpack_guy	6	17496	150	0.1538462
Never_Let_go	6	24170	150	0.1538462

Abbildung 5.10

Beispiel Sequence Chart



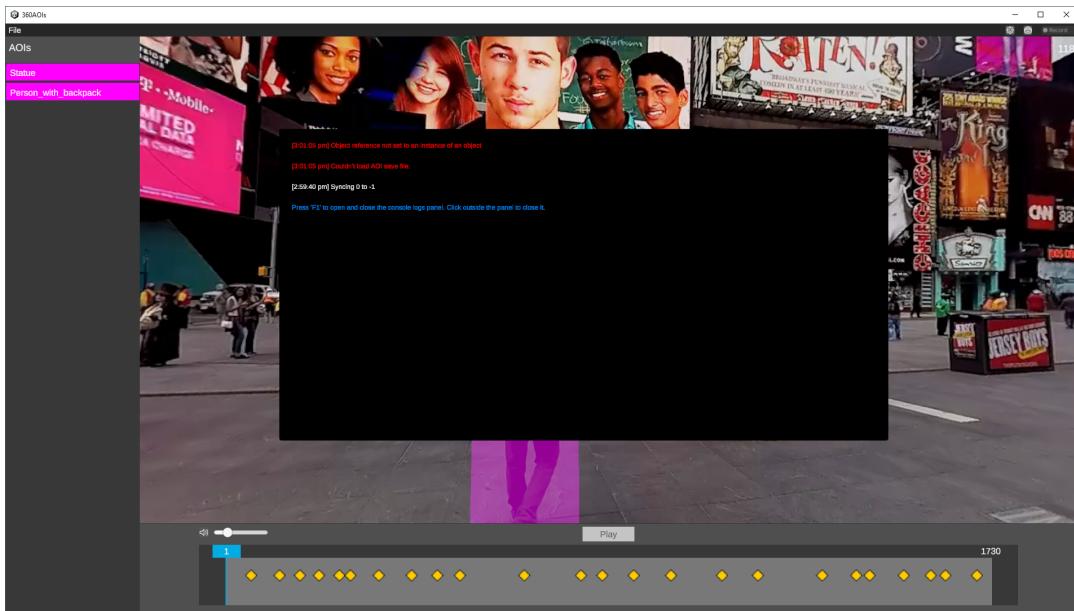
5.2.1.7 Fehlerbenachrichtigungen

Sollte während der Nutzung ein Fehler im Programmablauf auftreten, öffnet sich das in 5.11 abgebildete Fenster. Das Fenster enthält alle Logeinträge, Warnungen

sowie Fehlermeldungen, die bis zu diesem Zeitpunkt aufgetreten sind. Die Log-Konsole kann jederzeit durch einen Tastendruck auf die F1-Taste geöffnet sowie durch einen Klick außerhalb des schwarzen Bereichs oder durch einen erneuten Tastendruck auf die F1-Taste wieder geschlossen werden. Die Funktion gibt Aufschlüsse darüber, was die Ursachen eventuell auftretender Probleme und Bugs sein könnten.

Abbildung 5.11

Log-Konsole des Programms



5.2.1.8 Offline Analyse

Das Programm enthält eine prototypische Implementierung einer offline Analyse. Neben den AOI-Parametern gibt die Software nach jedem Versuchsdurchlauf zwei CSV-Dateien mit den ungefilterten und gefilterten Blickrichtungsdaten aus. Die offline Analyse erlaubt der*dem Anwender*in, die CSV-Datei der gefilterten Blickrichtungsdaten zu reimportieren und die AOI-Parameter erneut bestimmen zu lassen. Dabei werden die in dem *Settings*-Menü derzeit definierten Grenzwerte für die Dwell- und Fixationsdetektion genutzt. Mit dieser Funktion kann die*der Anwender*in durch mehrfaches Anpassen der Grenzwerte bei gleichbleibenden Rohdaten die Ergebnisse verschiedener Grenzwerteinstellungen miteinander vergleichen. Dabei ist zu beachten, dass keine erneute Bestimmung von AOI Hits durchgeführt wird. Es werden die AOI Hits aus der importierten CSV-Datei verwendet. Eine im Nachhinein stattgefundene Anpassung der AOIs wird von der offline Analyse also nicht berücksichtigt. Um die Funktion aufzurufen, klickt die*der Anwender*in auf die Schaltfläche *Debug Rerun* innerhalb des *File* Menüs (siehe Abb. 5.7a). Die Funktion ist derzeit so benannt, da sie in ihrem jetzigen prototypischen Zustand primär zu Debugging-Zwecken dient. Im Anschluss öffnet sich ein Systemdialog, in welchem

die CSV-Datei der gefilterten Daten ausgewählt und eingelesen werden kann. Nach einer kurzen Ladezeit öffnet sich erneut ein Systemdialog, in welchem der Datenexport der offline Analyse gespeichert werden kann.

Die vorgestellten Funktionalitäten unterstützen Anwender*innen bei der Versuchsvorbereitung, der Eyetracking-Kalibrierung und der eigentlichen Versuchsdurchführung. Das Programm exportiert die implementierten Eyetracking-Parameter und stellt sofort nach dem Versuch eine Visualisierung des Blickverhaltens der Versuchsperson bereit. Der CSV-Datenexport der Software kann anschließend genutzt werden, um statistische Auswertungen der Versuche durchzuführen. Es wurde darauf geachtet, der Versuchsleitung möglichst viel Kontrolle zu geben. So wurde eine Versuchsleitungsansicht implementiert, mithilfe derer die Blickrichtung der Versuchsperson überwacht werden kann und ein *Settings* Menü gestaltet, welches unterschiedliche Optionen für die Anpassung relevanter Einstellungen und Grenzwerte beinhaltet.

5.2.2 Programmlogik

Nachdem in der letzten Sektion ein Überblick über die Funktionalitäten der Software gegeben wurde, werden in dieser Sektion konkrete Ansätze der Implementierung vorgestellt. Ein Verständnis der beschriebenen Konzepte setzt Vorkenntnisse der Entwicklungsplattform Unity voraus. Unity bietet umfangreiche Kurse und Tutorials auf ihrer eigenen Lernplattform UnityLearn an, mithilfe derer eine Einarbeitung in die Entwicklungsumgebung erfolgen kann (Unity Learn, 12/06/2023). Bei der Nennung von Konzepten der Unity Plattform werden Verweise zu der entsprechenden Dokumentation in Klammern verlinkt. Die Entwicklung erfolgte mit Version 2021.3.16f1 des Unity Editors. Für eine Übersicht aller verwendeten Softwarepakte siehe Anhang D. Die vollständige Dokumentation des Programmcodes ist unter https://kiliansanchez.github.io/360AOIs_public/index.html abrufbar. Anhang B gibt einen Überblick über die *GameObjects* in der Unity Szene, um die Einarbeitung in das Unity Projekt zu vereinfachen.

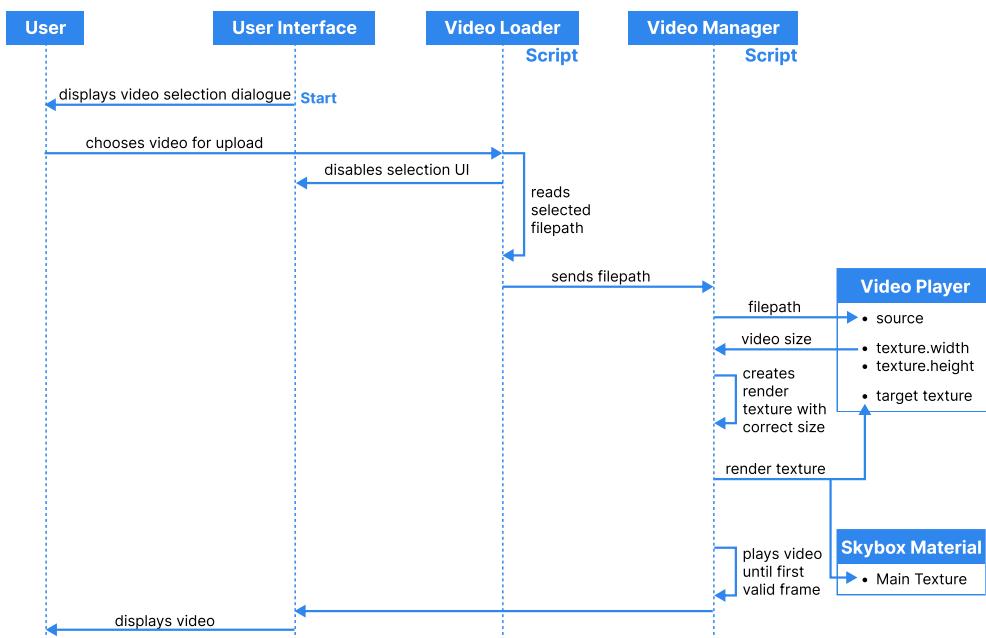
5.2.2.1 360° Video Import und Playback

Eine der ersten programmatischen Überlegungen, die bei der Implementierung des Projekts auftrat, war die Frage, auf welche Art und Weise 360° Videos in Unity abgebildet werden sollen. Wie Schicks (2022) und Tawa et al. (2022) zeigen, ist eine verbreitete Methode zur Darstellung von 360° Videos, sie als Textur auf eine Kugel zu projizieren (siehe Abb. 4.1 für ein 360° Video auf einer Kugel). Ein anderer Ansatz, der auf dem offiziellen Unity-Blog vorgestellt wird, ist die Darstellung des 360° Videos in der Skybox der Spielwelt (de Margerie, 2017). In Videospiel-Engines ist die Skybox eine Textur, die normalerweise zur Darstellung des Himmels in Videospielen genutzt und in 360° um die Spielwelt herum projiziert wird (Unity Technologies, 02/06/2023b). Deswegen bietet sie sich dazu an, als Zieltextrur für das Video zu dienen. Ein Vorteil dieser Methode ist, dass keine besonderen Anpassungen stattfinden müssen, damit das Video nicht spiegelverkehrt dargestellt wird. Die Darstellung des 360° Videos erfolgt sofort korrekt. Ein Nachteil ist, dass es nicht ganz trivial ist, den Schnittpunkt eines Blickrichtungsvektors mit der Textur auf der Skybox zu bestimmten. Man stelle sich zum Beispiel vor, man möchte auf einem Standbild des Videos visualisieren, welche Bereiche besonders oft angesehen worden sind. Dafür möchte man jeden Blickrichtungsvektor auf einen Pixel des Standbilds mappen und Pixel, die häufig betrachtet worden sind, mit einer höheren Intensität abbilden. Diese Übersetzung von einer Blickrichtung zu einem Pixel auf der Videotextur ist mit der Nutzung der Skybox komplexer, da Unity dafür keine bereits vorgefertigte Funktion bereitstellt. Es wurde jedoch eine Methode gefunden, einen Vektor auf die Textur der Skybox abzubilden und daraus den genauen Punkt im Video abzuleiten. Eine prototypische Implementierung ist in dem Skript ‚GazeToTextureMapper‘ zu finden. Da die Methode somit keine ersichtlichen Nachteile aufweist und von Unity empfohlen wird, wurde die Skybox zur Darstellung des 360° Videos gewählt.

Der Ablauf des Einlesens und der Darstellung des 360° Videos ist in Abbildung 5.12 zu sehen und lautet wie folgt: Beim Laden eines 360° Videos wird der Dateipfad des Videos von einem Skript namens ‚VideoLoader‘ eingelesen und an das Skript ‚VideoManager‘ übergeben. Der ‚VideoManager‘ setzt den gewählten Dateipfad als Quelle für einen *Video Player Component* (siehe Source Property in Unity Technologies, 02/06/2023a). Damit Videos beliebiger Auflösung korrekt dargestellt werden können, wird im Anschluss die Auflösung des Videos ausgelesen. Daraufhin wird eine *Render Texture* (Unity Technologies, 14/07/2023d) mit der ausgelesenen Auflösung initialisiert und als Zieltextur für den *Video Player Component* gesetzt. Die *Render Texture* wird dem *Skybox Material* als *Main Texture* zugewiesen (Unity Technologies, 14/07/2023c) und nun auf der Skybox abgebildet. Dieser Ablauf stellt sicher, dass sich die Auflösung der Skybox-Textur dynamisch an das hochgeladene Video anpasst.

Abbildung 5.12

Ablaufdiagramm der Videoladesequenz



Anmerkung. Der Beginn des Ablaufs ist gekennzeichnet mit dem ‚Start‘-Tag.

Das Testen verschiedener 360° Videos brachte zum Vorschein, dass Videos nicht zwangsläufig bei Frame '0' oder '1' beginnen. Je nachdem wie die Zeitinformationen des Videos kodiert sind, ist es möglich, dass der erste Frame eines Videos etwa Frame '4' ist. Der Programmcode musste dementsprechend angepasst werden, da ursprünglich davon ausgegangen wurde, dass jedes Video mit Frame '0' startet.

Nachdem das Video in den *Video Player Component* geladen wurde, wird es automatisch bis zu dem ersten validen Frame abgespielt und dann wieder pausiert. Dieser Prozess ist für Anwender*innen nicht sichtbar und erlaubt den ersten validen Frame in einem Großteil der Fälle korrekt auszulesen. Ein Nachteil ist, dass für Videos, die mit Frame '0' starten, gegebenenfalls fälschlicherweise Frame '1' oder '2' als erster Frame bestimmt wird. Dieser Trade-off wurde in Kauf genommen, da sonst Keyframes für Frames definiert werden könnten, die nicht wirklich existieren. Dies würde zu unvorhergesehenen Programmabläufen führen und wurde dementsprechend umgangen.

Eine weitere Eigenheit der Implementierung betrifft die Navigation innerhalb des eingelesenen Videos. Unity's *Video Player Component* erlaubt es, zu einem bestimmten Frame oder Timestamp in einem pausierten Video zu springen, garantiert jedoch nicht, dass das Standbild des Videos, das nach dem Sprung dargestellt wird, tatsächlich dem Frame des Videos an der gewählten Position entspricht. Praktisch bedeutet das, dass die*der Anwender*in etwa zu Frame '148' navigieren möchte, im Anschluss jedoch das Standbild für Frame '36' dargestellt wird, bis das Video abgespielt wird und sich der Videoplayer aktualisiert. Dieser Umstand ist höchst problematisch, da die*der Anwender*in so Anpassung an der Position und Größe von AOIs basierend auf der Darstellung eines vollkommen falschen Frames vornehmen würde. Um dieses Problem zu umgehen, „tastet“ sich die gewählte Implementierung an den Zielframe heran, um den Videoplayer mehrfach zur Aktualisierung der dargestellten Textur zu zwingen. In dem Beispiel würde das Programm zunächst zu Frame '147' navigieren und anschließend einen erneuten Sprung auf Frame '148' vornehmen. Mit diesem Vorgehen stellt das Programm den korrekten Frame zuverlässig dar.

5.2.2.2 Interne AOI Repräsentation

AOIs wurden als zweidimensionale Flächen, sogenannte *Quads* implementiert (Unity Technologies, 14/07/2023b), die in der 3D-Spielwelt von Unity positioniert werden. *Quads* wurden gewählt, da sie ein sehr simples, zweidimensionales und dadurch performantes Spielobjekt sind. Für die gewählte Implementierung würde der Einsatz von dreidimensionalen Objekten, wie *Cubes* keinen Mehrwert erzeugen. Die virtuellen Kameras, die dazu dienen, das 360° Video in der Versuchsleitungsansicht und in der VR-Brille anzuzeigen, befinden sich an der Position (0, 0, 0), also im Zentrum der Spielwelt und blicken von dort auf die AOIs und Skybox. Die AOIs halten einen konstanten Abstand von zehn Einheiten zum Weltmittelpunkt und befinden sich somit zwischen den Kameras und der Skybox. Der Abstand von zehn Einheiten ist eine relativ willkürlich gewählte Größe. Der genaue Abstand ist für die gewählte Implementierung nicht relevant. Das Grundprinzip ist, dass anhand der Blickrichtungsdaten ein *Raycast* von der Kamera in die Blickrichtung durchgeführt wird. Ein *Raycast* ist das Aussenden eines Strahls von einem Startpunkt in eine

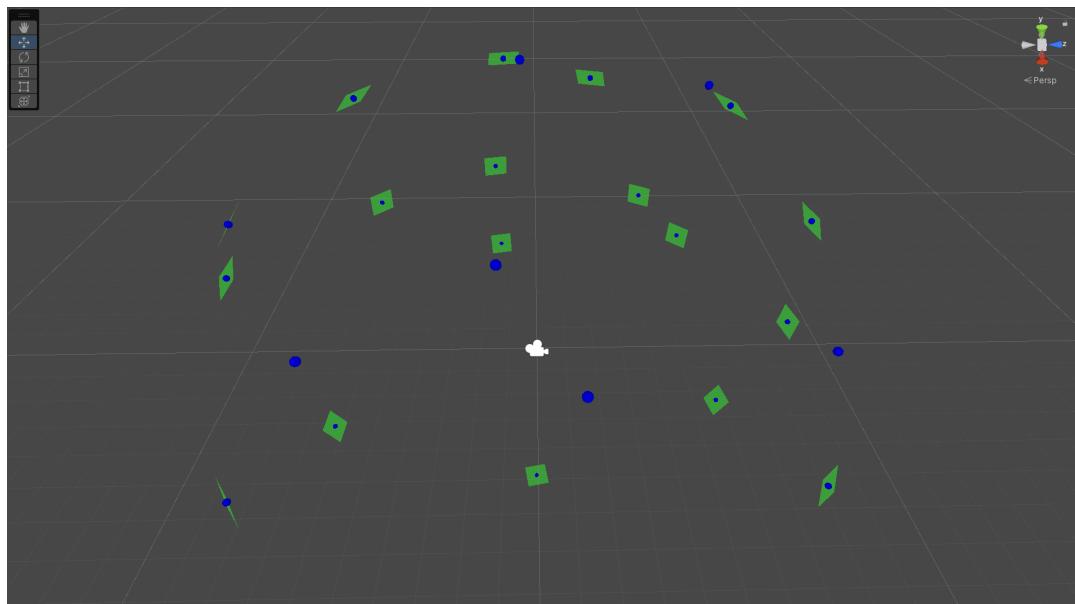
definierte Richtung (Unity Technologies, 14/07/2023c). Wenn der Strahl mit einem Objekt in der Spielwelt kollidiert, können über diese Kollision bestimmte Informationen abgefragt werden, z. B. welches Objekt genau getroffen worden ist. Solange ein AOI-Gameobject das Areal der Skybox (und damit des Videos) aus Sicht der Kamera überdeckt, in dem sich der abzudeckende Stimulus befindet, ist der genaue Abstand des AOI-Gameobjects zur Kamera für den Raycast irrelevant. Je größer der Abstand zwischen Kamera und AOI, desto größer muss das AOI-Gameobject sein, um das Zielobjekt aus Sicht der Kamera, bzw. Sicht der*des Betrachter(s)*in zu überdecken. Formal lautet der geometrische Zusammenhang zwischen visuellem Winkel θ (horizontal oder vertikal), Größe und Distanz eines Objekts wie folgt:

$$\theta = 2 * \arctan(\text{size}/2 * \text{distance}) \quad (5.1)$$

Wären die AOI-Gameobjects statt zehn Einheiten 20 Einheiten von den Kameras entfernt, müssten sie einfach doppelt so groß sein, um die gleiche Area of Interest im Video abzudecken. Wichtig ist lediglich, dass der Abstand konstant gehalten wird, sodass sich die wahrgenommene Größe der AOI, bzw. der visuelle Winkel, den sie einnimmt, nicht unvorhergesehenen ändert.

Abbildung 5.13

AOI-Gameobjects in Unity's Scene-Ansicht



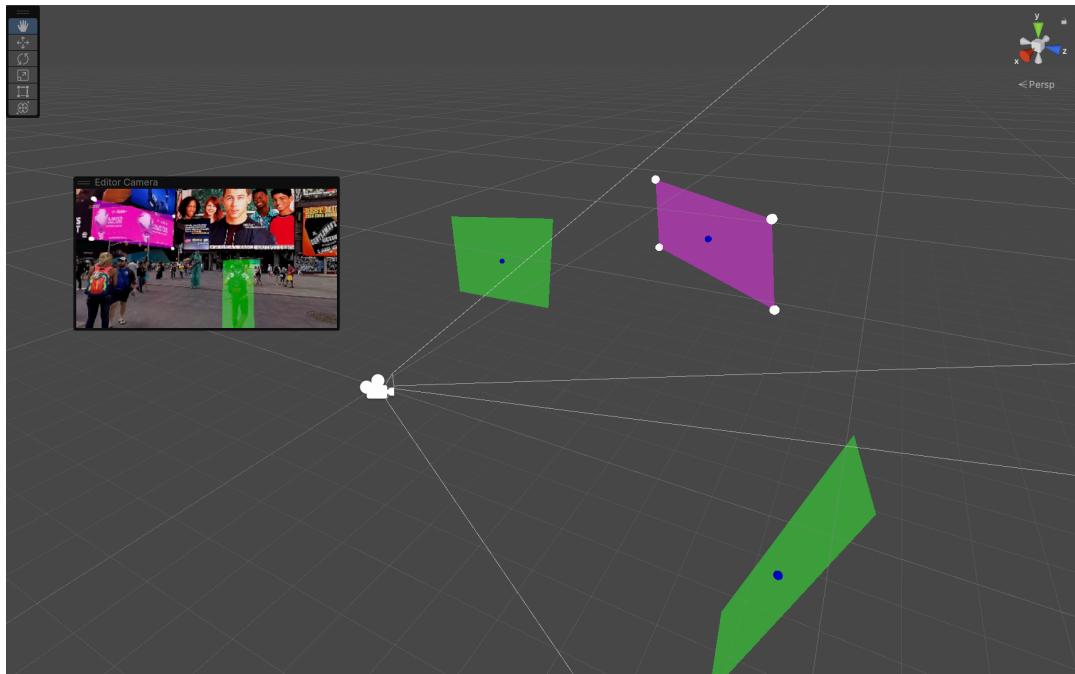
Anmerkung. Es ist zu sehen, wie die AOIs mit einem konstanten Abstand um den Weltmittelpunkt herum positioniert sind. In dem Beispiel überdecken die AOIs nicht wirklich sinnvolle Zielobjekte in einem 360° Video, sondern wurden zu Darstellungszwecken willkürlich rundum im Video verteilt.

Jedes AOI-Gameobject besitzt mehrere Skripte, in welchen die für die AOI nötigen Funktionalitäten implementiert wurden (Unity Technologies, 14/07/2023a). Die

Skripte sind z. B. für die Änderung der Position und Form der AOI zuständig. AOIs lassen sich in 360° um die Kamera herum bewegen, um sie im Video zu positionieren. Abbildung 5.13 zeigt die Positionierung verschiedener AOIs in der 3D-Spielwelt. Die Kamera, von welcher aus sowohl die Versuchsperson als auch die Versuchsleitung die Szene betrachten, ist im Zentrum als Kamera-Symbol dargestellt.

Abbildung 5.14

Virtuelle Kamera mit Blick auf zwei AOIs



Anmerkung. Die weißen Linien zeigen das Kamera-Frustum der ‚EditorCamera‘. Die Kamera blickt vom Weltmittelpunkt aus auf die AOIs. Hinter den AOIs bildet die Skybox das 360° Video ab, jedoch ist dieses in der *Scene View* derzeit nicht sichtbar. In dem schwarz umrandeten Rechteck ist die Ansicht aus Perspektive der Kamera eingeblendet, in welcher das 360° Video samt AOIs zu sehen ist.

Eine Standard-AOI mit allen nötigen Funktionalitäten wurde als *Prefab* (vergleichbar mit einem *Template*) namens ‚AOI Quad‘ gespeichert (Unity Technologies, 14/07/2023a). Ein Skript namens ‚AOIManager‘ kreiert Instanzen dieses *Prefabs*, wenn die*der Anwender*in eine neue AOI erstellen möchte und koordiniert die instantiierten AOI-Objekte untereinander. So handhabt der AOIManager zum Beispiel, welche AOI auf Eingaben reagiert, wenn zwei oder mehr überlappende AOIs von der*dem Benutzer*in angeklickt werden.

Ein alternativer Ansatz zur Implementierung der AOIs wäre gewesen, sie nicht als 3D-Objekte in der Spielwelt zu repräsentieren, sondern ausschließlich als UI-Elemente darzustellen. Die AOIs wären also für die*den Benutzer*in in der Benutzeroberfläche sichtbar, würden aber nicht wirklich im dreidimensionalen Raum der Spielwelt existieren. Dieser Ansatz hätte wahrscheinlich zu visuell

ansprechenderen AOIs geführt und gegebenenfalls den Programmcode für die Änderung von Position und Größe der AOIs vereinfacht. Jedoch müsste eine Übersetzung von den im UI angelegten AOIs zu Arealen in Weltkoordinaten stattfinden, um Überschneidungspunkte mit der Blickrichtung zu kalkulieren, da das SRanipal SDK die Blickrichtungsdaten in Weltkoordinaten zur Verfügung stellt. Aufgrund der möglicherweise besseren Handhabbarkeit der AOIs wäre es zukünftig interessant, diesen Ansatz zu verfolgen.

5.2.2.3 Animation und Interpolation

Die Animation der Areas of Interest erfolgt anhand benutzerdefinierter Keyframes. Jedes AOI-Gameobject besitzt ein Skript namens ‚Animation‘, welches die Daten aller Keyframes der AOI in einer Liste speichert und diese verwaltet. Immer wenn die*der Anwender*in einen neuen Keyframe erstellt oder einen bestehenden Keyframe anpasst, kalkuliert das Skript die korrekte Position und Größe der AOI für jeden Frame des Videos, indem zwischen den zwei naheliegendsten Keyframes linear interpoliert wird. Während der Videowiedergabe kontrolliert das Skript die Form und Position der AOI und setzt diese entsprechend der kalkulierten Interpolation.

Dieser Ansatz könnte verbessert werden, indem nur solche Frames neu berechnet werden, die von der Erstellung bzw. Aktualisierung des zuletzt angepassten Keyframes abhängig sind. Weiterhin wird bei der Interpolation nicht darauf geachtet, einen konstanten Abstand von zehn Einheiten zum Weltmittelpunkt einzuhalten. Die bei der Animation auftretenden Distanzabweichungen sind sehr gering, sodass sie für das Eyetracking unproblematisch sind. Dieser Umstand sorgt jedoch dafür, dass AOIs einen kleinen ‚Sprung‘ machen, sobald die*der Benutzer*in die Position einer AOI für einen interpolierten Frame anpassen möchte, da der Programmcode dann wieder darauf achtet, einen Abstand von zehn Einheiten einzuhalten. In einer zukünftigen Version sollte die Interpolation nur Positionen kalkulieren, die zehn Einheiten vom Weltmittelpunkt entfernt sind.

5.2.2.4 Eyetracking und Event Detection

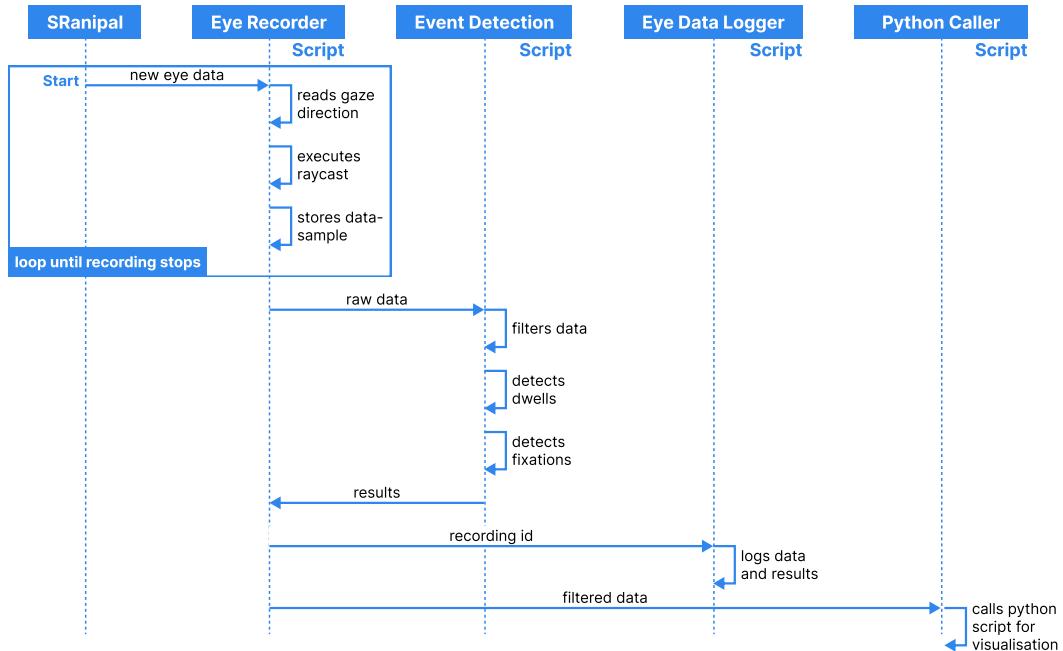
Das Eyetracking erfolgt durch das SRanipal SDK. Über das SDK werden die Blickrichtungsdaten der HTC Vive Pro Eye in einem separaten Thread ausgelesen (Unity Technologies, 24/02/2021). Durch das Auslagern in einen separaten Thread wird sichergestellt, dass die Datenabfrage mit der gleichen Frequenz wie der des Eyetracking-Systems, also mit 120 Hz läuft, sofern dafür ausreichend Hardwareressourcen vorhanden sind.

In dem Moment, in dem ein Unity Projekt eine VR-Brille ansteuert, übernimmt das Unity XR-Framework Kontrolle über die Framerate der Applikation. Da die Displays der HTC Vive Pro Eye eine Bildwiederholrate von 90 Hz haben, setzt das Unit XR-Framework die Framerate des Programms zum Zweck vertikaler Synchronisation auf konstante 90 Hz. Aufgrund dessen können Befehle und Funktionsaufrufe, die mit den Eyetracking-Daten arbeiten, nicht in der meistverwendeten *Update*-Methode von Unity’s *MonoBehaviour* aufgerufen werden, da die *Update*-Methode zu Beginn jedes Frames ausgeführt wird (Unity Technologies, 14/07/2023d). Die Frequenz von 90 Hz wäre zu langsam, um mit den 120 Hz Eyetracking-Daten zu arbeiten. Die Nutzung von Unity’s *Update*-Methode ist die Fehlerursache für die Abtastrate von 90 Hz in dem von Schicks (2022) entwickelten Prototypen. Dennoch können nicht alle Funktionalitäten einfach in einen separaten Thread ausgelagert werden, da die

Unity API nicht ‚thread-safe‘ ist. Das bedeutet, dass Aufrufe der Unity API, also aller von Unity bereitgestellten Funktionalitäten, nicht in einem separaten Thread ausgeführt werden können. Für die Verarbeitung der Eyetracking-Daten ist die Unity API jedoch zwingend erforderlich. Um diese Limitation der Plattform zu umgehen, wurde die von Unity zur Verfügung gestellte Funktion *InvokeRepeating* verwendet (Unity Technologies, 14/07/2023e). *InvokeRepeating* erlaubt es, Funktionsaufrufe mit einer definierten Taktrate zu tätigen und wurde genutzt, um Funktionen, die mit den Eyetracking-Daten arbeiten, mit 240 Hz aufzurufen.

Der Ablauf der Datensammlung ist in Diagramm 5.15 dargestellt. Während der Durchführung eines Versuches überprüft das Programm alle vier Millisekunden, ob ein neues Eyetracking-Data-Sample von dem SDK zur Verfügung gestellt wurde. Aus dem Data-Sample wird die Blickrichtung ausgelesen und ein *Raycast*, ausgehend von dem Weltmittelpunkt (bzw. der Position der VR-Kamera) in Richtung der Blickrichtung durchgeführt. Sollte der *Raycast* ein AOI-Objekt treffen, wird das Data-Sample mit einem AOI Hit auf diese AOI gekennzeichnet und in die Liste der Rohdaten aufgenommen. Sollte keine AOI getroffen werden, so wird das Data-Sample intern mit einem AOI Hit auf die AOI ‚null‘ versehen. Dementsprechend sollte die*der Benutzer*in keine AOI mit dem Namen ‚null‘ bezeichnen, da dieser AOI-Name intern bereits vergeben ist. In einer zukünftigen Version sollte das Programm den Nutzer warnen, wenn dieser versucht, eine AOI ‚null‘ zu nennen.

Nach Beendigung des Versuches werden die Rohdaten an das Skript ‚EventDetection‘ übergeben. Das Skript filtert zunächst alle invaliden Data-Samples. Ob ein Sample valide ist oder nicht, wird von dem SRanipal SDK angegeben. Anschließend werden alle Dwells mit ihrem Start, ihrer Dauer, und ihrem Ende bestimmt. Der Algorithmus zur Bestimmung von Dwells ist in Algorithmus 1 simplifiziert dargestellt.

Abbildung 5.15*Ablaufdiagramm der Datensammlung*

Anmerkung. Der Beginn des Ablaufs ist gekennzeichnet mit dem ‚Start‘-Tag. Der eigentliche Name des Skripts ‚Python Caller‘ lautet ‚Sequence Chart Python Process‘, wurde jedoch in der Abbildung aus Platzgründen abgekürzt.

Algorithmus 1 : Dwell Detection

Data : Filtered Data-Samples

Result : Detected Dwells

```

foreach AOI in Data do
    get all Data-Samples with AOI Hit for current AOI
    foreach AOI Data-Sample do
        get next Sample
        if delta time between current Sample and next Sample is within Dwell Time
            Threshold then
                | store current Sample as part of current Dwell
            end
        else
            | compute Dwell Start, Duration and End based on stored Sample
            Timestamps
            save current Dwell for current AOI
        end
    end
end

```

Nachdem alle Dwells detektiert wurden, werden im Anschluss Fixationen aus den Blickbewegungsdaten bestimmt. Der Algorithmus basiert auf dem von Llanes-Jurado et al. (2020) vorgestellten I-DT (Identifikation - Dispersion Threshold) Algorithmus zu Detektion von Fixationen in Virtual Reality. Der Algorithmus der Autoren verwendet ein Dispersions- bzw. Streuungskriterium zur Detektion von Fixationen. Alle Data-Samples einer Fixation müssen zueinander eine Distanz (bzw. einen Winkel) innerhalb des definierten maximalen Streuungswerts aufweisen, um gemeinsam als Fixation klassifiziert zu werden. Weiterhin überprüft der Algorithmus die Frequenz der Eyetracking-Data-Samples und verwirft Datenbereiche, in denen die Frequenz unter einem definierten Grenzwert liegt. Sollten die Blickrichtungsdaten z. B. aufgrund einer kurzzeitigen Überlastung des PCs oder aus sonstigen Gründen eine unzureichende temporale Auflösung aufweisen, werden diese nicht verwendet. Der Algorithmus wurde als Grundlage für die Implementierung gewählt, da er speziell für den Einsatz in Virtual Reality unter Verwendung der HTC Vive Pro Eye entwickelt und validiert wurde. Die Autoren schlussfolgern, dass ein Dispersionsgrenzwert von 1° optimal für die Detektion von Fixationen mit der in dieser Arbeit verwendeten Hardware ist.

Eine weitere gängige Methode zur Detektion von Fixationen ist die Nutzung eines Geschwindigkeitsgrenzwerts (Holmqvist et al., 2011). Deswegen wurde der Algorithmus dahin gehend erweitert, neben einem Dispersions- auch ein Geschwindigkeitskriterium zu nutzen, um zu entscheiden, ob Data-Samples Teil einer Fixation sein könnten. Wie in 5.2.1.4 dargestellt wurde, können beide Grenzwerte von Anwender*innen eingestellt werden. Sollte die*der Anwender*in nur eins von beiden Kriterien nutzen wollen, ist es möglich, das andere Kriterium beliebig hoch einzustellen, sodass es als Grenzwert praktisch irrelevant wird. Sofern genügend Data-Samples aufeinanderfolgen, die allesamt innerhalb der benutzerdefinierten Grenzwerte liegen und gemeinsam die definierte minimale Fixationsdauer erreichen, wird eine Fixation vermerkt. Daraufhin werden alle dieser Data-Samples auf AOI Hits hin überprüft und die Fixation der AOI (oder Whitespace-AOI ‚null‘) zugeordnet, die während der Fixation am häufigsten getroffen wurde. Eine simplifizierte Darstellung ist in Algorithmus 2 abgebildet.

Algorithmus 2 : Fixation Detection**Data :** Filtered Data-Samples**Result :** Detected Fixations

```

while New Gaze Data-Sample is available do
    Initialize Window to cover the Time Threshold
    while True do
        Computation of  $\theta$  for each Pair of Points
        Computation of Velocity for adjacent Points
        if All Points in Window within Thresholds then
            Mark Window as Fixation
            Increase window size by one
            Continue
        end
        else
            Break
        end
    end
    if Window marked as Fixation then
        Computation of Fixation Start, End and Duration
        Assign Fixation to AOI with most AOI Hits or Whitespace
        Remove all Samples of Window except last
    end
    else
        Remove first Sample of Window
    end
end

```

Aus den detektierten Dwells und Fixationen werden die AOI-Parameter bestimmt und als CSV-Datei ausgegeben (siehe Abbildung 5.9).

5.2.2.5 Visualisierung

Nachdem die Event Detection alle Dwells und Fixationen bestimmt und ausgegeben hat, wird ein Sequence Chart (Sequenzdiagramm) des Versuchsdurchlaufs erstellt (siehe Abbildung 5.10). Die für Python verfügbare Bibliothek *Matplotlib* besitzt eingebaute Funktionalitäten für die Erstellung eines sogenannten *Eventplots* („*matplotlib.pyplot.eventplot — Matplotlib 3.1.2 documentation*“, 03/06/2023). Da dieser Eventplot mit sehr wenig Programmieraufwand zu einem Sequence Chart ummodelliert werden kann, wurde sich dazu entschieden, Python für die Visualisierung zu nutzen. Das Unity-Skript ‚Sequence Chart Python Process‘ ruft dafür ein Python-Skript namens ‚visualization_test‘ auf und über gibt diesem zwei Dateipfade als Parameter. Ersterer verweist auf die gefilterten Rohdaten des Versuchsdurchlaufs, letzterer zu dem Dateipfad, an dem die Visualisierung als PDF-Datei abgelegt

werden soll. Wie dem Namen des Python-Skripts zu entnehmen ist, handelt es sich bei der Visualisierung um eine erste prototypische Test-Implementierung. Die Visualisierung funktioniert, wurde jedoch nicht auf ihre Robustheit für verschiedene Anzahlen von AOIs oder unterschiedliche Versuchsdauern hin überprüft. Es kann z. B. sein, dass die Visualisierung bei der Nutzung vieler verschiedener AOIs nicht genug Platz auf der y-Achse bereitstellt, um alle AOIs ohne visuelle Artefakte abzubilden.

Kapitel 6

Validierung

Das folgende Kapitel stellt die Methoden und Ergebnisse der Validierung des Programms dar.

6.1 Methodik

6.1.1 Simulation

Während der Entwicklung wurden bereits erste Schritte unternommen, um sicherzustellen, dass bei der Implementierung neuer Funktionalitäten sowie der Anpassung bestehenden Codes keine größeren Fehler in der Event Detection auftreten. Dafür wurden zwei Testskripte geschrieben, die jeweils bekannte Blickbewegungsdaten erzeugen und diese an den Teil des Programms, der die Eyetracking-Parameter bestimmt, übergeben. So konnte während der Entwicklung regelmäßig überprüft werden, ob der CSV-Datenexport mit den zu erwartenden Werten übereinstimmt. Die Implementierung dieser Simulation ist jedoch sehr rudimentär und generiert starre, künstliche Blickbewegungen. Angesichts dessen wurden Versuche durchgeführt, um die Software mit menschlichen Blickbewegungen zu validieren.

6.1.2 Versuche

6.1.2.1 Stichprobe

Die sechs Teilnehmenden waren zwischen 18 und 40 Jahre alt. Zwei der Versuchspersonen identifizierten sich als weiblich, vier als männlich. Vier der sechs Teilnehmenden trugen während der Durchführung weder Brille noch Kontaktlinsen. Ein*e Teilnehmer*in trug eine Brille, ein*e anderer Teilnehmer*in trug Kontaktlinsen. Die Teilnehmenden wurden darum gebeten, auf das Tragen von Make-up im Augenbereich zu verzichten.

6.1.2.2 Versuchsaufbau und Instruktionen

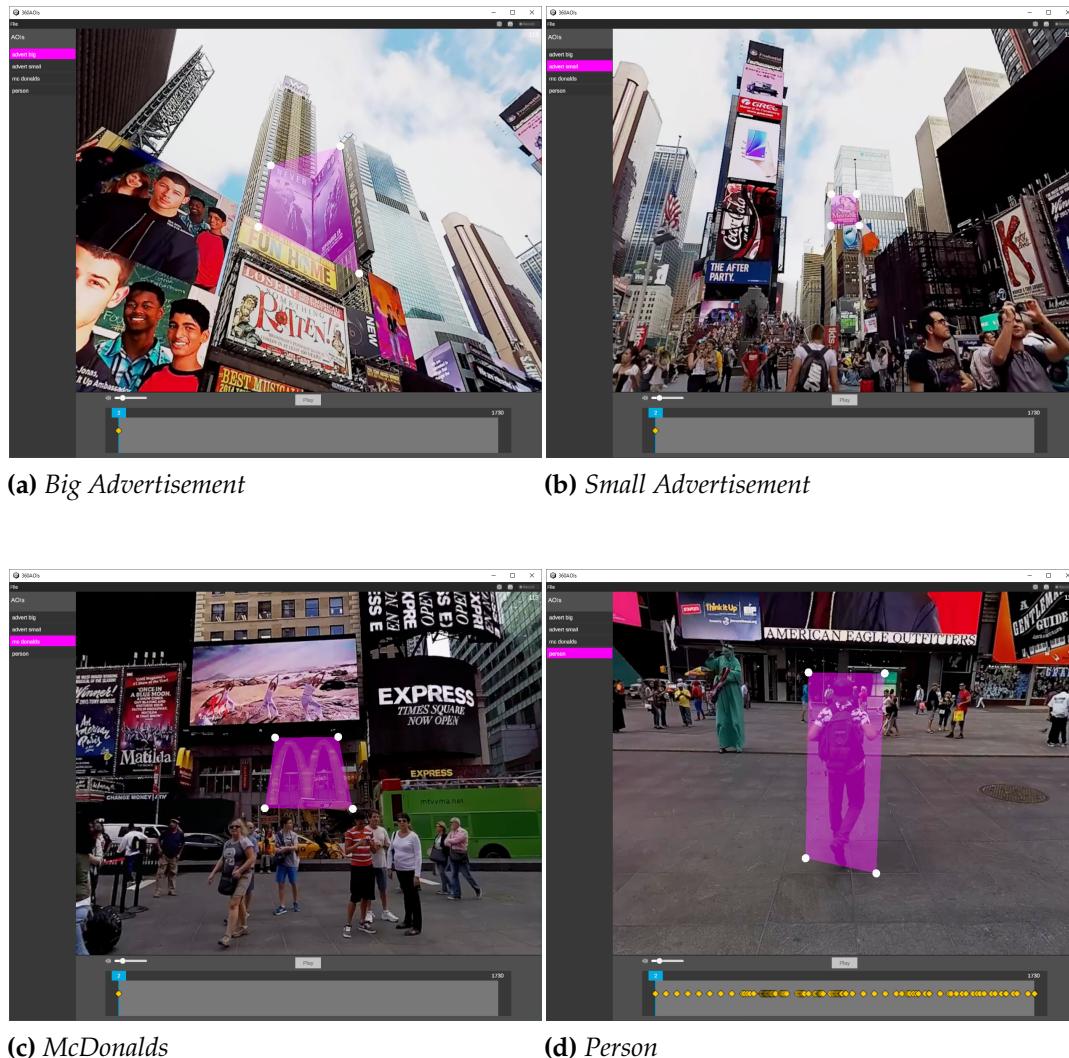
Die Grundidee des Versuchsaufbaus ist, dass die von dem Programm bestimmten AOI-Parameter nicht nur algorithmisch, sondern ebenfalls durch eine*n menschliche*n Rater*in bestimmt und die Ergebnisse im Anschluss miteinander verglichen werden. Bei jedem Versuchsdurchlauf wurde eine Bildschirmaufnahme der Versuchsleiteransicht des entwickelten Programms getätigt. Die Versuchsleiteransicht bildet die aktuelle Blickrichtung der Versuchsperson anhand eines Strahls ab. Im Anschluss an den Versuch kann ein*e Rater*in anhand dieser Bildschirmaufnahme ein manuelles Coding der Blickrichtungsparameter vornehmen, indem sie*er die Blickrichtung in der Bildschirmaufnahme verfolgt und Treffer auf AOIs Frame-by-Frame auszählt. Zur Validierung des Programms können die manuell anhand der Bildschirmaufnahme bestimmten Parameter mit dem Datenexport des Programms verglichen werden. Eine hohe Übereinstimmung zwischen den manuell bestimmten Parametern und dem Datenexport würde bedeuten, dass das Programm ähnliche Ergebnisse wie ein*e menschliche*r Rater*in liefert. Für den Versuch wurde dieser Ansatz gewählt, da der Wertebereich der zu überprüfenden Parameter teilweise im Bereich weniger hundert Millisekunden liegt. Versuchsdesigns, die von Reaktionszeiten und der Fähigkeit der Versuchspersonen, Instruktionen zu befolgen, abhängig sind, sind nicht dafür geeignet, Ergebnisse mit einer derart hohen zeitlichen Genauigkeit zu liefern. In dem gewählten Versuchsaufbau ist es irrelevant, ob eine Versuchsperson Instruktionen schnell oder langsam befolgt, oder eine bestimmte AOI zu einem bestimmten Zeitpunkt ansieht oder nicht. Ausschlaggebend ist lediglich, ob das manuelle Coding und der Datenexport des Programms dahin gehend übereinstimmen, wann und für wie lange AOIs betrachtet worden sind.

Für den Versuch wurde ein 360° Video des New York Times Square verwendet. In dem Video wurden vier Areas of Interest definiert. Es wurde darauf geachtet, AOIs unterschiedlicher Größe in unterschiedlichen Bereichen des Videos zu wählen. Weiterhin wurden für den Bereich, den Versuchspersonen direkt zu Beginn des Videos sehen, AOIs gewählt, die sich relativ weit oben im Blickfeld befinden. So wurde vermieden, dass Teilnehmende direkt zu Beginn des Versuchs mehrere AOIs ansehen, obwohl noch keine Instruktionen gegeben wurden. Für die Datengüte der Studie wäre dies unproblematisch gewesen, jedoch hätte es die Auswertung wesentlich zeitintensiver gestaltet, wenn Versuchspersonen bereits ungeplant AOI Hits auf mehrere AOIs in zufälliger Reihenfolge generiert hätten, welche alle manuell codiert werden müssten. Bei zwei der definierten AOIs handelt es sich um Werbeplakate, die Text beinhalten. Eines dieser Werbeplakate ist relativ groß, das andere relativ klein. Als dritte AOI wurde eine große Leuchtreklame des Logos einer McDonalds Filiale gewählt. Zuletzt wurde eine Person, die sich im Laufe des Videos um die Kamera herum bewegt, als AOI definiert. Bei den ersten drei genannten AOIs handelt es sich um statische, unbewegte AOIs. Bei letzterer handelt es sich um eine

dynamische AOI, die mit einer Vielzahl von Keyframes animiert wurde. Die AOIs sind in Abbildung 6.1 dargestellt.

Abbildung 6.1

AOIs für Versuch



Zu Beginn des Versuchs wurde jede*r Teilnehmer*in über den Ablauf informiert. Im Anschluss wurde der Versuchsperson die VR-Brille übergeben und bequem, fest sitzend aufgesetzt. Danach erfolgte eine Kalibrierung des Eyetrackers sowie eine Überprüfung der Kalibrierung anhand der Testszene (siehe Abb. 5.8 für eine Darstellung der Testszene). Zu diesem Zeitpunkt wurde der Versuchsperson die eigene Blickrichtung über das *Settings* Menü kurzzeitig eingeblendet, um eine subjektive Einschätzung darüber zu erhalten, ob die Blickrichtung gut abgebildet wird. Sofern dies nicht der Fall war, wurde erneut kalibriert. Im Anschluss wurde der Versuch gestartet. Während des Versuchs erhielten die Teilnehmenden vier Instruktionen, die sie jeweils dazu aufforderten, eine der vier AOIs aufzufinden und anzusehen. So wurde sichergestellt, dass für jede AOI Blickbewegungsdaten erhoben wurden. Ein Beispiel

für eine solche Instruktion lautet: „Rechts von dir befindet sich ein sehr großes McDonalds Logo. Kannst du dieses bitte finden und dir kurz ansehen?“ Nachdem die Versuchsperson die AOI gefunden und für wenige Sekunden inspiziert hatte, wurde die nächste Instruktion gegeben. Nachdem alle vier AOIs betrachtet wurden, wurde der Versuch beendet.

Das manuelle Coding erfolgte im Anschluss anhand der Aufnahmen, ohne davor den Datenexport des Programms zu sichten. Während alle Dwells manuell codiert wurden, wurde lediglich die erste Fixation händisch bestimmt, da das manuelle Coding von Fixationen sehr zeitaufwendig ist. Um dennoch bewerten zu können, ob die Fixationsdetektion des Programms korrekt funktioniert, wurde sie mit einer bestehenden Softwarelösung zur Klassifikation von Fixationen verglichen.

6.1.3 Vergleich zu bestehendem Softwarepaket

Da die manuelle Bestimmung der Parameter anhand der Videoaufzeichnung durch eine*n menschlichen Rater*in erfolgt, ist davon auszugehen, dass hierbei Fehler auftreten. Besonders Fixationen sind manuell relativ schwer und nur mit viel Zeitaufwand zu bestimmen. Deshalb wurde für jede AOI pro Versuchsdurchlauf nur die erste Fixation manuell ausgewertet. Um dennoch die Robustheit der Fixationsdetektion objektiv überprüfen zu können, wurde ein bestehendes Softwarepaket zur Klassifizierung von Fixationen herangezogen. Das Python-Paket *vr-idt* stellt einen für VR Anwendungen entwickelten Klassifikationsalgorithmus für die Identifikation von Fixationen bereit (PyPI, 24/07/2023). Der I-DT-Algorithmus des Pakets basiert ebenso wie der in diesem Projekt implementierte Algorithmus auf der Veröffentlichung von Llanes-Jurado et al. (2020). Der Rohdatenexport der Blickrichtungsdaten jedes Versuchsdurchlaufs wurde mit einem Python-Skript eingelesen und mit Hilfe des *vr-idt* Pakets auf Fixationen hin untersucht. Das Ergebnis wurde mit den Fixationen verglichen, die von der entwickelten Software detektiert wurden.

6.2 Hard- und Software

6.2.1 Software

Für die Versuchsdurchführung wurden die in dem entwickelten AOI Editor einstellbaren Grenzwerte gleichbleibend eingestellt. Der Dispersion-Threshold wurde auf 1° gesetzt, der Velocity-Threshold auf 35°/s. Die minimale Fixationsdauer wurde auf 150 ms festgelegt, die minimale Dauer zwischen zwei Dwells auf 100 ms.

Die Aufnahme erfolgte mit der Open-Source Software OBS Studio in Version 29.1.3 („Open Broadcaster Software | OBS“, 24/07/2023). Aufgenommen wurde in 1920×1080 px mit 120 FPS.

Die Frame-by-Frame Analyse der Videoaufzeichnungen erfolgte mit dem Open-Source Programm Kinovea in Version 0.9.5 (Charmant, 24/07/2023). Das Programm wurde ursprünglich für Sport-Analysen entwickelt und erlaubt den Einsatz von framebasierten Stopuhren für die Bestimmung zeitlicher Parameter in Videos.

Für den Vergleich zwischen dem eigens implementierten Fixationsalgorithmus und dem vr-idt Python-Paket (Version 0.0.5) wurde ein Python-Skript mithilfe der Spyder IDE (Version 5.4.2) erstellt (Spyder, 09/06/2023). Anschließend wurde die Übereinstimmung zwischen der Ausgabe des Skripts und des AOI Editors mit Microsoft Excel verglichen.

6.2.2 Hardware

Die Hardwarespezifikationen des verwendeten PCs lauten: 32 GB RAM, Intel i7-9700K Prozessor, Nvidia RTX 2070-Super Grafikkarte. Mit dieser Hardware traten keine Performanzprobleme auf. Die Versuchsleitungsansicht wurde auf einem 144 Hz Monitor abgebildet. Das 360° Video wurde von den Versuchspersonen mit der HTC Vive Pro Eye VR-Brille angesehen (siehe technische Spezifikationen in Anhang C). Das Eyetracking erfolgte über den in der VR-Brille verbauten Tobii-Eyetracker mit 120 Hz. Die VR-Brille wurde mit zwei SteamVR Base Stations 2.0 betrieben.

6.3 Ergebnisse

6.3.1 Ergebnisse der Versuche

Für jeden Versuchsdurchlauf wurden die Ergebnisse des manuellen Codings mit den Daten des CSV-Exports verglichen. Als Maß für die Abweichung zwischen beiden Methoden wurde der mittlere absolute Fehler gewählt. Zur Bestimmung des mittleren absoluten Fehlers wurde der absolute Differenzwert zwischen den manuell bestimmten Werten und den algorithmisch bestimmten Werten jedes Parameters für jede AOI gebildet. Die absoluten Fehler wurden über die Versuchsdurchläufe hinweg aufsummiert und durch die Anzahl der Versuchsdurchläufe n geteilt.

Die Formel für die Bestimmung des mittleren absoluten Fehlers (mean absolute error, MAE) lautet:

$$MAE = \frac{\sum_{i=1}^n |y_i - x_i|}{n} \quad (6.1)$$

Dabei ist n die Stichprobengröße, x der anhand des manuellen Codings bestimmte Wert und y der durch das Programm bestimmte Wert. Die errechneten mittleren Fehler sind in Tabelle 6.1 zu sehen. Die letzte Zeile der Tabelle enthält jeweils den Durchschnittswert des MAEs für jeden Parameter über alle AOIs.

Tabelle 6.1

Mittlerer absoluter Fehler zwischen CSV-Datenexport und manuellem Coding

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	0	0.67	3.33	6.50
Big Advert.	0	7.50	8.83	10.33
McDonalds	0	7.50	5.33	3.83
Person	0	5.00	2.83	7.17
\bar{x} of MAEs	0	5.17	5.08	6.96

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	25.17	22.17
Big Advert.	44.17	34.67
McDonalds	11.33	18.50
Person	11.00	2.83
\bar{x} of MAEs	22.92	19.54

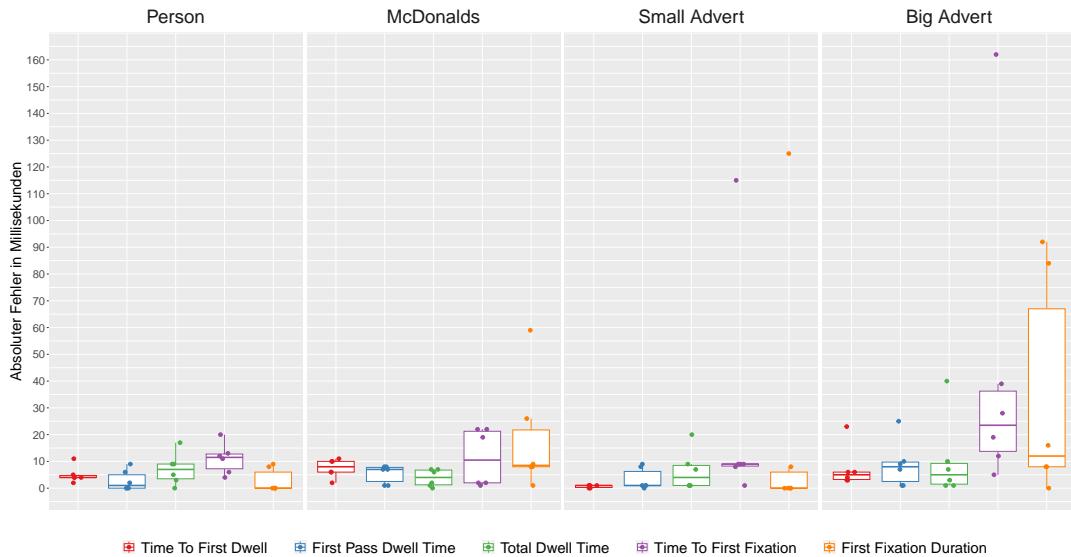
Die Anzahl der Dwells stimmte in jedem Versuchsdurchlauf für jede AOI zwischen beiden Verfahren überein, sodass der mittlere absolute Fehler jeweils 0 beträgt. In Tabelle 6.1 zeigen sich für alle weiteren Parameter leichte Abweichungen zwischen Datenexport und manuellem Coding. Für die Dwell-Parameter befindet sich der durchschnittliche Fehler jeweils unter 11ms, also unter der Zeit zwischen zwei Frames des Programms, da dieses mit einer Bildwiederholrate von 90 Hz lief. Die größten Abweichungen zwischen den algorithmisch und manuell bestimmten Parametern sind bei der *Time To First Fixation* und *First Fixation Duration* aufgetreten. Es ist zu beobachten, dass der größte Fehler für jeden Parameter bei der AOI *Big Advertisement*, also dem großen Werbeplakat aufgetreten ist.

Die Verteilung der absoluten Fehler ist in Abbildung 6.2 zu sehen. Für die AOIs *Person*, *McDonalds* und *Small Advertisement* clustern sich die einzelnen Abweichungen zwischen dem manuellen Rating und dem Datenexport für nahezu alle Parameter

bis auf wenige Datenpunkte im Bereich unter 15 ms. Im Gegensatz dazu zeigen die beiden Fixations-Parameter, insbesondere für die AOI *Big Advertisement*, einige Male Abweichungen von bis zu 158 ms.

Abbildung 6.2

Verteilung der absoluten Abweichungen zwischen Datenexport und manuellen Rating



Die Ergebnisse des Datenexports, des manuellen Ratings sowie die absoluten Abweichungen zwischen beiden Verfahren für jede Versuchsperson sind in Anhang E einzusehen.

6.3.2 Ergebnisse des Vergleichs mit dem vr-idt Python-Paket

Für die Validierung der Fixationsdetektion wurden die von dem Programm ermittelten Fixationen mit den Ausgaben des vr-idt Python-Pakets verglichen. Die Rohdaten der Versuche wurden für den Vergleich insgesamt dreimal algorithmisch analysiert. Durchlauf 0 erfolgte mit dem vr-idt Paket. In der vorgestellten Software kann sowohl ein Dispersions- als auch Geschwindigkeitskriterium eingestellt werden. Da das vr-idt Paket ausschließlich ein Dispersionskriterium verwendet, wurde das Geschwindigkeitskriterium des AOI Editors in Durchlauf 1 sehr hoch eingestellt, sodass es für die Klassifikation praktisch irrelevant war. Im Anschluss wurde zusätzlich ein Durchlauf 2 mit einem Geschwindigkeitskriterium getätigt. Die Einstellungen der Algorithmen sind in Tabelle 6.2 zu sehen.

Tabelle 6.2*Einstellungen der Algorithmen*

Setting	vr-idt Paket	AOI Editor	AOI Editor
	Durchlauf 0	Durchlauf 1	Durchlauf 2
Fixation duration threshold in ms	150	150	150
Fixation dispersion threshold in °	1	1	1
Fixation velocity threshold in °/s	-	9999	35
Frequency threshold in Hz	30	-	-

Für die Rohdaten aller sechs Versuchspersonen wurde jedes einzelne Data-Sample durch die Algorithmen dahingehend gekennzeichnet, ob es Teil einer Fixation war oder nicht. Die prozentuale Übereinstimmung der Klassifikationen des AOI Editors gegenüber dem vr-idt Paket sind der Tabelle 6.3 zu entnehmen.

Tabelle 6.3*Prozentuale Übereinstimmung der Klassifikation aller Data-Samples mit dem vr-idt Paket*

Versuchsdurchlauf	%-Übereinstimmung Durchlauf 1 vs. vr-idt	%-Übereinstimmung Durchlauf 2 vs. vr-idt
#1	0.99	0.86
#2	0.99	0.88
#3	0.99	0.86
#4	1.0	0.83
#5	0.98	0.87
#6	0.98	0.90
\bar{x}	0.99	0.87

Die Übereinstimmung der detektierten Fixationen des ersten Durchlaufs mit den Ergebnissen des vr-idt Pakets liegt bei mindestens 98 % und im Durchschnitt bei 99 %. Das Hinzuziehen eines Geschwindigkeitskriteriums reduziert die Übereinstimmung um bis zu 17 %, da Datenbereiche, in denen die Blickrichtung innerhalb eines Grads bleibt, aber die ermittelte Geschwindigkeit des Blicks mehr als 35°/s beträgt, in diesem Durchlauf nicht als Fixationen klassifiziert wurden.

Kapitel 7

Diskussion

Die Problemstellung der vorliegenden Arbeit war die Weiterentwicklung eines 360° AOI Editors zur Durchführung von Eyetracking-Experimenten in Virtual Reality. Ziel war es, eine Software zu gestalten, die, wenn auch immer noch in einem prototypischen Zustand, bereits für Eyetracking-Experimente in einem wissenschaftlichen Kontext genutzt werden kann. Um zu quantifizieren, ob und in welchem Ausmaß ein ggf. systematischer Fehler in den von der Software ermittelten Blickbewegungsparametern existiert, wurde eine kleine Validierungsstudie durchgeführt und ein Vergleich zu einem bestehenden Softwarepaket zur Bestimmung von Fixation getätigt.

Der Vergleich der manuell anhand von Bildschirmaufzeichnungen bestimmten AOI-Parametern gegenüber den von der Software ausgegebenen Parametern liefert vielversprechende Ergebnisse. Der Fehler für alle dwellbezogenen Parameter liegt für jede der getesteten AOIs im Schnitt unter 11 ms und somit unter der Zeit zwischen zwei Frames. Die sehr hohe Übereinstimmung zwischen dem menschlichen Rater und der Software ergibt Sinn, da für die Bestimmung von Dwells lediglich von Bedeutung ist, ob sich der Blick im Bereich einer AOI befindet. Bis auf wenige Sonderfälle ist dies auch als Mensch sehr einfach, nahezu objektiv zu bewerten. Die in Tabelle 6.1 dargestellten Ergebnisse legen nahe, dass die Software die Parameter *Dwell Count*, *Time To First Dwell*, *First Pass Dwell Time*, und *Total Dwell Time* korrekt bestimmt.

Eine größere Abweichung zwischen dem manuellen Rating und dem Datenexport der Software ist bei den beiden fixationsbezogenen Parametern zu beobachten. Dies ist wenig überraschend, da die manuelle Bestimmung von Fixationen anhand der Videoaufzeichnungen wesentlich fehleranfälliger ist. Die in der Videoaufzeichnung abgebildete Blickrichtung ist aufgrund von Rauschen, mikroskopischen Blickbewegungen sowie Kopfbewegungen selbst während einer Fixation ständig in Bewegung. Deshalb ist schwer einzuschätzen, wann genau eine Fixation nun beginnt und wann sie aufhört. Obwohl die Übereinstimmung zwischen dem Rater und der Software in vielen Fällen immer noch relativ hoch ist, existieren einige Fälle, in denen

sie jeweils einen anderen Datenbereich als erste Fixation klassifiziert haben. Aus den dargestellten Ergebnissen des manuellen Codings kann nicht mit Sicherheit abgeleitet werden, ob die Software Fixationen korrekt bestimmt und es sich bei der Ursache der Abweichungen um menschlichen Fehler handelt oder ob in der Fixationsdetektion des Programms ein Fehler vorhanden ist. Es ist weiterhin unklar, weshalb besonders für die AOI *Big Advertisement* die größten Abweichungen aufgetreten sind. In Abbildung 6.2 ist zu sehen, dass die Übereinstimmung zwischen dem Rater und der Software für diese AOI teilweise genauso hoch ist, wie für andere AOIs, während für manche Datenpunkte eine große Abweichung aufgetreten ist. Mit der geringen Stichprobengröße ist keine Systematik in dem Fehler zu erkennen. Aufgrund dessen wurde die Fixationsdetektion mit einem bestehenden Softwarepaket verglichen. Die Ergebnisse des Vergleichs zwischen der eigenen Implementierung und dem vr-idt Paket zeigen, dass die Übereinstimmung bei gleichen Einstellungen mit selbigen zugrundeliegenden Rohdaten im Schnitt bei 99 % liegt. Es wurde versucht, anhand Einsicht des Quellcodes des vr-idt Pakets nachzuvollziehen, weshalb geringe Abweichungen zwischen den Implementierungen desselben Algorithmus auftreten, jedoch konnte die Ursache nicht gefunden werden. Es ist möglich, dass die Abweichungen aufgrund des Frequenzkriteriums des vr-idt Pakets zustande kommen, da dieses in der eigenen Implementierung in dieser Form nicht existiert. Dennoch ist das Ergebnis des Vergleichs zufriedenstellend, sodass auf Grundlage der Ergebnisse vermutet werden kann, dass der im Programm implementierte Fixationsdetektionsalgorithmus dem von Llanes-Jurado et al. (2020) vorgestellten I-DT Algorithmus großteils entspricht. Wenig überraschend sorgt das Hinzuziehen eines Velocity-Thresholds in einem weiteren Durchlauf dafür, dass die Übereinstimmung mit dem vr-idt Paket sinkt, da die Algorithmen in diesem Fall nicht mit identischen Kriterien arbeiten.

Die vorgestellten Ergebnisse legen nahe, dass die Software bereits dazu in der Lage ist, Forschende bei der Untersuchung von Fragestellungen mit 360° Videos zu unterstützen. Anstatt aufwendige Versuchsdesign unter Voraussetzung programmatischer Vorkenntnisse selbst anzulegen, können Forschende die Software dazu verwenden, Versuche vergleichsweise einfach zu gestalten und die implementierten Parameter zu erheben. Da sie Open-Source ist, können Forschende zusätzlich Modifikationen und Ergänzungen des Quellcodes vornehmen, um das Programm den eigenen Bedürfnissen anzupassen. Die benutzerdefinierbaren Grenzwerte zur Bestimmung von Dwells und Fixationen erlauben eine einfache Anpassung der Algorithmen an die eigene Untersuchung. Gegenüber der Nutzung anderer in Kapitel 4 vorgestellter Arbeiten setzt die Software keine Programmierkenntnisse voraus. Das Programm bündelt alle darin enthaltenen Funktionalitäten in einem Tool, sodass kein Aufrufen unterschiedlicher Skripte nötig ist. Dadurch ist das Programm für Forschende aus den Sozialwissenschaften, der Psychologie und anderen, weniger

technischen Fachrichtungen einfacher nutzbar als vergleichbare Open-Source Angebote. Die kostenlose Bereitstellung der Software erlaubt die Nutzung mit einer relativ niedrigen Eintrittsbarriere. Lediglich eine HTC Vive Pro Eye und ein Windows-PC mit den in A.3 beschriebenen Mindestanforderungen müssen vorhanden sein. Die Definition von AOIs und die Bearbeitung von Keyframes erfolgt anhand einer grafischen Benutzeroberfläche. Auch wenn keine Evaluation der Benutzerfreundlichkeit des Tools stattgefunden hat, wurde darauf geachtet, Interaktionen so zu gestalten, dass diese eine präzise Definition von AOIs zulassen und Anwender*innen erlauben, Fehler jederzeit zu korrigieren. Es ist zu vermuten, dass diese Art der Bedienung gegenüber der Erstellung von AOIs mit der Tastatur oder per Definition der Koordinaten in Textform vorzuziehen ist. Selbst kommerzielle Programme wie Worldviz Sightlab VR sind zum Teil noch auf eine Textdefinition von AOIs angewiesen (Sightlab VR Pro, 18/08/2023). Die in Sektion 5.1 definierten Anforderungen an das Projekt konnten umgesetzt und darüber hinaus übertroffen werden. Funktionalitäten wie die Testszene zur Überprüfung der Eyetracking-Kalibrierung oder das Erstellen und Laden von Projekten waren zu Beginn nicht vorgesehen, konnten jedoch im Zuge der Entwicklung konzipiert und implementiert werden.

7.1 Limitationen

7.1.1 Limitationen der Validierung

Trotz der beschriebenen Erkenntnisse unterliegen sowohl die Validierung als auch der Funktionsumfang der Software einigen Limitationen, welche im Folgenden vorgestellt werden. Zunächst ist anzumerken, dass die Studie lediglich mit sechs Versuchspersonen durchgeführt wurde. Aufgrund der geringen Stichprobengröße sind die Daten anfällig für Ausreißer und bieten sich aufgrund der geringen Power nicht für eine statistische Analyse der Ergebnisse an. Weiterhin lässt die Stichprobe nicht zu, zu untersuchen, ob und in welchem Ausmaß bestimmte Eigenschaften von Versuchspersonen (z. B. Tragen einer Brille) Auswirkungen auf die Performanz der Software haben. Zusätzlich waren die einzelnen Versuchsdurchläufe sehr kurz, damit der Aufwand der manuellen Bestimmung der Blickparameter in einem angemessenen Maß verblieb. Ob Probleme oder unvorhergesehenes Verhalten bei längeren Versuchsdurchläufen auftreten, ist deshalb unklar.

Ein weiteres methodisches Problem ist, dass der Entwickler der Software sowie der einzige Rater für das manuelle Coding die gleiche Person waren. Auch wenn versucht wurde, das Rating objektiv und unvoreingenommen durchzuführen, ist es methodisch bedenklich, dass der Rater keine unabhängige Person war. Darüber hinaus hatte der Rater keine vorherige Erfahrung mit der Codierung von Blickbewegungen. Um die Ergebnisse mit angemessener Gewissheit zu validieren, ist es erforderlich, eine Replikationsstudie unter Einbezug mehrerer Rater*innen mit unterschiedlichem Erfahrungshintergrund durchzuführen.

Des Weiteren war die Bildwiederholfrequenz der Software während der Bildschirmaufnahme auf 90 FPS begrenzt. In einer zukünftigen Validierung wäre wünschenswert, dass der Blickrichtungsvektor, der in der Bildschirmaufnahme zu sehen ist, mit der gleichen Frequenz wie das Eyetracking aktualisiert wird, während die Bildschirmaufzeichnung mit 240 FPS getätigt wird. Unter diesen Voraussetzungen wäre die Auftretenswahrscheinlichkeit von Aliasing entsprechend dem Nyquist-Shannon-Abtasttheorem stark reduziert (Shannon, 1949). Weiterhin wurde aus Zeitgründen bei der manuellen Bestimmung der Blickparameter jeweils nur die erste Fixation pro AOI händisch bestimmt. Der Parameter *Fixation Count*, also die Gesamtanzahl von Fixationen auf eine AOI wurde dementsprechend nicht direkt überprüft. Aufgrund des Vergleichs mit dem vr-idt Paket ist es jedoch höchstwahrscheinlich, dass dieser Parameter korrekt bestimmt wird.

Zu Bedenken ist, dass die Validierung der Software anhand einer Bildschirmaufnahme derselben Software erfolgt ist. Das bedeutet, dass die Ergebnisse nur dann valide sein können, sofern die Implementierung der Versuchsleitungsansicht mit dem eingezeichneten Blickrichtungsvektor keine programmatischen Fehler enthält, welche die Darstellung der Blickrichtung betreffen. Da die Implementierung des eingezeichneten Blickrichtungsvektors sehr simpel ist und Versuchspersonen die Genauigkeit des Vektors selbst überprüfen konnten, ist es jedoch zweifelhaft, dass hierbei ein Problem aufgetreten ist.

Ein weiterer Kritikpunkt an der Validierung besteht darin, dass es sich bei der Software um einen AOI Editor für die Erstellung dynamischer AOIs handelt, aber keine genaue Untersuchung verschiedener animierter AOIs stattgefunden hat. Wenn auch unwahrscheinlich ist derzeit unklar, ob bestimmte Bewegungen oder bestimmte Positionen einer AOI im Video Fehler im Programm erzeugen, da nur eine geringe Anzahl an AOIs getestet wurde und davon lediglich eine animiert war. Aus diesem Grund sollte eine Überprüfung in Abhängigkeit der Dynamik, Form, Größe und Position von AOIs stattfinden.

Die durchgeführte Validierung liefert dennoch erste aussagekräftige Indizien dafür, dass die Software die gewählten Parameter korrekt bestimmt. Aufgrund der genannten Limitationen sollte dennoch in Zukunft eine umfangreichere Validierungsstudie durchgeführt werden.

7.1.2 Limitationen der Software und Verbesserungspotenzial

Der entwickelte AOI Editor ist nur für das Windows Betriebssystem verfügbar. Obwohl Unity dazu in der Lage ist, Projekte für Linux und macOS Systeme zu kompilieren, sind die VR- und Eyetracking-Laufzeitumgebungen nicht für andere Plattformen verfügbar. Derzeit ist unklar, ob dieses Problem durch zusätzlichen Entwicklungsaufwand behoben werden könnte.

Weiterhin unterstützt das Programm ausschließlich die HTC Vive PRO Eye. Dementsprechend benötigen Anwender*innen des Programms Zugang zu genau diesem Modell. Während andere VR-Brillen mit integrierten Eyetrackern erhältlich sind, z. B. die Produkte von Varjo (Varjo.com, 06/06/2023), müsste das SDK des jeweiligen Herstellers in das Projekt eingebunden und eine Logik implementiert werden, welche die Auswahl des von der*dem Benutzer*in verwendeten Headsets unterstützt. Dies ist mit zusätzlichem Entwicklungsaufwand jedoch möglich und wäre eine sehr sinnvolle Verbesserung.

Die Form der erstellten AOIs ist immer rechteckig, sodass runde oder komplexe Formen nicht passgenau abgedeckt werden können. Im Laufe der Entwicklung wurde versucht, die Möglichkeit zu bieten, beliebig geformte Polygone zu erstellen, indem die*der Anwender*in die Eckpunkte der Form definiert und diese im Anschluss miteinander verbunden werden. Die Triangulation von Eckpunkten zu einem mit Unity kompatiblem *Mesh* ist jedoch nicht trivial (Unity Technologies, 14/07/2023b). Dementsprechend wurde auf die Implementierung dieser Funktionalität verzichtet. In Zukunft könnte die Python Bibliothek *Pymesh* verwendet werden, um die Triangulation zu kalkulieren und das Ergebnis zurück an das Unity Projekt zu übergeben („PyMesh — Geometry Processing Library for Python — PyMesh 0.2.1 documentation“, 29/01/2021).

Ein weiteres Problem bei der Handhabung von AOIs ist, dass diese nicht einfach rotiert oder skaliert werden können. Derzeit müssen Anwender*innen jeden Eckpunkt einzeln anpassen, um eine AOI zu vergrößern oder zu rotieren. Hierfür könnten Interaktionen implementiert werden, um die Erstellung von AOIs komfortabler zu gestalten.

Die Animation der AOIs erfolgt anhand linearer Interpolation zwischen benutzerdefinierten Keyframes. Diese Vorgehensweise sorgt dafür, dass vor allem bei langen 360° Videos viele Keyframes gesetzt werden müssen, um ein bewegtes Objekt gut zu tracken. Die ungenaue Abdeckung von einem Areal im Video durch die AOI kann, je nachdem, ob die AOI gegenüber dem Areal im Video zu klein, zu groß oder gar versetzt ist, zu false-positive und false-negative AOI Hits führen. Veröffentlichungen wie Bonikowski et al. (2021) zeigen, dass die Animation von AOIs durch den Einsatz von Algorithmen schneller und präziser gelöst werden könnte. Wie in den

methodischen Überlegungen zu AOIs beschrieben (siehe Sektion 3.2.2), argumentieren Holmqvist et al. (2011), dass eine algorithmische Definition von AOIs gegenüber einer manuellen Definition vorzuziehen ist. Dieser Umstand sollte berücksichtigt und eine automatisierte AOI-Animation ermöglicht werden, jedoch ist zu erwarten, dass dies mit erheblichem Entwicklungsaufwand verbunden wäre.

Derzeit ist es während der Durchführung eines Versuchs prinzipiell möglich, AOIs in der Versuchsleitungsansicht zu bewegen. Es sollte eine Logik implementiert werden, die die Bearbeitung von AOIs während eines laufenden Versuchs blockiert.

Die derzeitige Implementierung kann einem einzelnen Eyetracking-Data-Sample nur einen einzigen AOI Hit zuordnen. Sollte eine Versuchsperson ihren Blick auf zwei oder mehr überlappende AOIs richten, wird lediglich ein AOI Hit auf die zuerst getroffene AOI verzeichnet. Wie in Sektion 3.2.2 beschrieben, sollten überlappende AOIs zwecks statistischer Auswertung vermieden werden. Jedoch wäre es in Zukunft möglich, dass das Programm erlaubt, AOIs in Teilbereiche zu untergliedern. Bei einem Blick auf einen Teilbereich könnten dann sowohl dieser als auch die übergeordnete AOI einen Treffer registrieren.

Sollte eine AOI in einem Versuchsdurchlauf nicht angesehen worden sein, wird für sie auch kein Eintrag in dem CSV-Datenexport der AOI-Parameter erstellt. Derzeit tauchen AOIs im Datenexport in der Reihenfolge auf, in der sie angesehen worden sind. Um Datenanalysen mit anderen Tools zu vereinfachen, wäre es wünschenswert, dass der Datenexport ein konsistentes Format über mehrere Versuchsdurchläufe hinweg beibehält. Dafür könnten zukünftig immer alle AOIs in gleichbleibender Reihenfolge gelistet werden.

Es gibt derzeit keinen Datenexport der einzelnen Fixationen. Es wäre sinnvoll, neben den bereits ausgegebenen Parametern eine CSV-Datei zu exportieren, die alle aufgetretenen Fixationen mit Start, Ende und Dauer beinhaltet. Dies würde der*dem Anwender*in erlauben, selbst eine Betrachtung der Fixationen vorzunehmen und weitere Statistiken zu berechnen. Die Implementierung eines solchen Exports sollte problemlos möglich sein.

Da in der Software keine Smooth Pursuit Detektion implementiert wurde, werden bei der Betrachtung von bewegten AOIs fälschlicherweise Fixationen detektiert, obwohl es sich um Smooth Pursuit Bewegungen handelt. Die Event Detection des Programms kann und sollte zukünftig Smooth Pursuit Bewegungen klassifizieren.

Die Implementierung des Projektmanagements ist sehr rudimentär und wurde ohne relevantes Vorwissen getätig. Derzeit werden lediglich die Dateipfade zu dem 360° Video und der Datei, in der die AOIs für das Video hinterlegt sind, gespeichert. Um

das 360° Video auszutauschen, muss die*der Benutzer*in ein neues Projekt anlegen oder den in der Project-Settings-Textdatei hinterlegten Dateipfad manuell anpassen. Eine einfache Handhabung über die Benutzeroberfläche existiert nicht. Weiterhin werden die Parameter, welche die*der Benutzer*in über das *Settings* Menü einstellt, nicht gespeichert. Dies kann dazu führen, dass bei der Durchführung von Versuchen über mehrere Sessions unterschiedliche Grenzwerte verwendet werden. Diese Fehlerquelle sollte von dem Programm abgefangen werden, indem es die Einstellungen in der Project-Settings-Datei abspeichert. Weiterhin wäre wünschenswert, in dem Programm zwischen unterschiedlichen Projekten wechseln zu können. Derzeit muss die Software neu gestartet werden, um ein anderes Projekt zu öffnen.

Auch wenn versucht wurde, die Interaktion mit dem Programm möglichst intuitiv zu gestalten, wurden keine Usability Evaluationen des Programms durchgeführt. Es ist anzunehmen, dass es Usability Probleme gibt, die in Zukunft verbessert werden könnten. Zum Beispiel wäre es sinnvoll, die derzeit sehr simple Timeline mit Markierungen in regelmäßigen Abständen zu versehen, um Anwender*innen visuell bei der Positionierung von Keyframes zu unterstützen. Für die Entwicklung der Benutzeroberfläche wurde das *Unity UI (uGUI) Package* verwendet. Zukünftig wird *uGUI* durch Unity's *UI Toolkit* ersetzt (Unity Technologies, 21/07/2023). Es wäre möglich und ratsam, eine Weiterentwicklung der Benutzeroberfläche mit dem neuen Unity UI Toolkit umzusetzen.

Zuletzt ist anzumerken, dass die Software derzeit keine Möglichkeiten zur Interaktion seitens der Versuchsperson bietet. 360° Videos können lediglich passiv betrachtet werden. Um Reaktionszeiten oder andere Interaktionen zu erheben, sollten die VR-Controller (in Händen haltbare Eingabegeräte der VR-Brille) in das Projekt eingebunden und der Versuchsleitung einen Baukasten zur Erstellung von Tastendruckabfragen zur Verfügung gestellt werden.

Gegenüber kommerziellen Lösungen ist die vorgestellte Software in Funktionsumfang, Hardwarekompatibilität, Verfügbarkeit für verschiedene Betriebssysteme und Detailgrad des Datenexports beschränkt. Forschende sind abgesehen von der in Anhang B bereitgestellten Dokumentation bei der Anwendung der Software auf sich allein gestellt. Besonders beim Auftreten von Problemen, Bugs oder Unklarheiten kann dieser Umstand zu unerwünschten Verzögerungen im Arbeitsablauf führen. Insbesondere bei zeitkritischen Forschungsunterfangen oder zur Untersuchung von Fragestellungen, die einen umfassenderen Funktionsumfang benötigen, ist dementsprechend eine kommerzielle Lösung vorzuziehen. Weiterhin ist der Entwickler des Programms kein professioneller Programmierer. Deswegen ist davon auszugehen, dass die konkrete programmatische Implementierung nicht dem Standard einer professionellen Entwicklung gerecht wird und Bugs sowie unvorhergesehene Programmabläufe auftreten können.

Die dargestellten Funktionalitäten, Ergebnisse und Limitationen ermöglichen es For-schenden abzuwägen, ob das Tool für den eigenen Einsatz infrage kommt. Vor allem für Untersuchungen, für die eine kostengünstige Lösung gesucht wird, die ohne die Notwendigkeit von Programmierkenntnissen nutzbar ist, ist die vorgestellte Softwa-re zu empfehlen. Trotz des zukünftigen Weiterentwicklungsbedarfs sowie Bedarfs an weiterer Validierung ist es im Zuge dieser Arbeit gelungen, ein Werkzeug zur Durchführung von Eyetracking-Experimenten in Virtual Reality mit 360° Videos zu entwickeln.

Kapitel 8

Ausblick

Das zukünftige Entwicklungs- und Forschungspotenzial im Kontext der vorgestellten Arbeit ist immens.

In einem ersten Schritt wäre es sinnvoll, eine vollumfassendere Validierungsstudie des bestehenden Stands der Software durchzuführen. Eine Kombination aus komplexer Simulation sowie einer Validierung mit einer ausreichend großen Stichprobe wäre wünschenswert. Dabei sollten sich AOIs in ihrer Größe und Form stärker unterscheiden und mehr dynamische AOIs genutzt werden. Eine solche Studie könnte mit ausreichender Gewissheit validieren, ob das Programm die Blickparameter korrekt bestimmt. Zusätzlich könnten Tests mit verschiedenen 360° Videos in variierender Länge Aufschluss über die Robustheit der Software liefern.

In der vorherigen Sektion wurde eine Vielzahl an Limitationen der Software dargestellt. Ein Großteil dieser Einschränkungen kann durch Weiterentwicklung des Programms behoben und verbessert werden. Insbesondere die Klassifikation von Smooth Pursuit Bewegungen ist im Kontext von dynamischen AOIs erstrebenswert und sollte in einem zukünftigen Weiterentwicklungsprojekt priorisiert werden. Des Weiteren wäre es vorteilhaft, die derzeit hohe programmatische Kopplung mit dem SRanipal SDK zu abstrahieren und weitere VR-Headsets zu unterstützen, um die Zugänglichkeit der Software zu verbessern. Im Anschluss könnten bestehende Funktionalitäten ausgebaut und neue Features, wie zusätzliche Visualisierungsoptionen oder Funktionen zur statistischen Auswertung ergänzt werden. Die Möglichkeiten hierbei sind nahezu unbegrenzt.

Für die in der Arbeit implementierten Algorithmen werden lediglich invalide Datenpunkte gefiltert. Eine Glättung oder sonstige Filterverfahren werden nicht angewendet. Da der Autor über keine einschlägigen Kenntnisse der Signalverarbeitung verfügt, wäre es sinnvoll, diesen Aspekt der Arbeit in Zukunft mit ausreichend theoretischem Wissen zu beleuchten. Interessant wäre weiterhin, ob die Bestimmung von AOI Hits anhand eines simplen Raycasts durch weitere Verfahren verbessert werden könnte. Tobii's Eyetracking SDK verwendet z. B. Machine-Learning-Algorithmen um

akkurate Vorhersagen darüber zu treffen, auf welchem Objekt der Blick ruht (Tobii Devzone, 10/07/2023b). Eine Anwendung ähnlicher Verfahren mit dem kostenfreien SRanipal SDK zur Steigerung der Datengüte wäre es wert, erprobt zu werden.

Da die Software nun weiterentwickelt und bis zu einem gewissen Grad validiert wurde, wäre es von besonderer Bedeutung, das Programm in einer Pilotstudie einzusetzen. So könnten Potenziale, Probleme und Verbesserungsmöglichkeiten identifiziert werden, die nur im wirklichen Einsatz zum Vorschein kommen. Der Austausch mit Wissenschaftlern und Wissenschaftlerinnen aus relevanten Forschungsfeldern ist für eine Weiterentwicklung unverzichtbar.

Anhang A

Installationsanweisungen

A.1 Ausführung ohne Unity

A.1.1 Steam und SteamVR

Um das vorgestellte Programm nutzen zu können, muss zunächst die Videospiel-Vertriebsplattform Steam installiert werden, damit die VR-Umgebung SteamVR auf dem System ausgeführt werden kann. Das Programm nutzt Valves SteamVR-Plugin für Unity als API für die Kommunikation mit der VR-Brille.

- Download und Installation von Steam <https://store.steampowered.com/>
- Installation von SteamVR über Steam <https://store.steampowered.com/app/250820/SteamVR/>

A.1.2 Vive Pro HMD Setup

Die für das Projekt verwendete HTC Vive Pro Eye sowie die dazugehörigen Basisstationen und Controller müssen korrekt mit dem verwendeten PC verbunden und installiert werden. Das von HTC zur Verfügung gestellte Vive Pro HMD Setup hilft der*dem Anwender*in dabei, die dafür notwendigen Schritte durchzuführen.

- Download und Ausführung des HTC Vive Pro HMD Setups <https://www.vive.com/us/setup/vive-pro-hmd/>

A.1.3 SRanipal und Tobii VRU02 Runtime

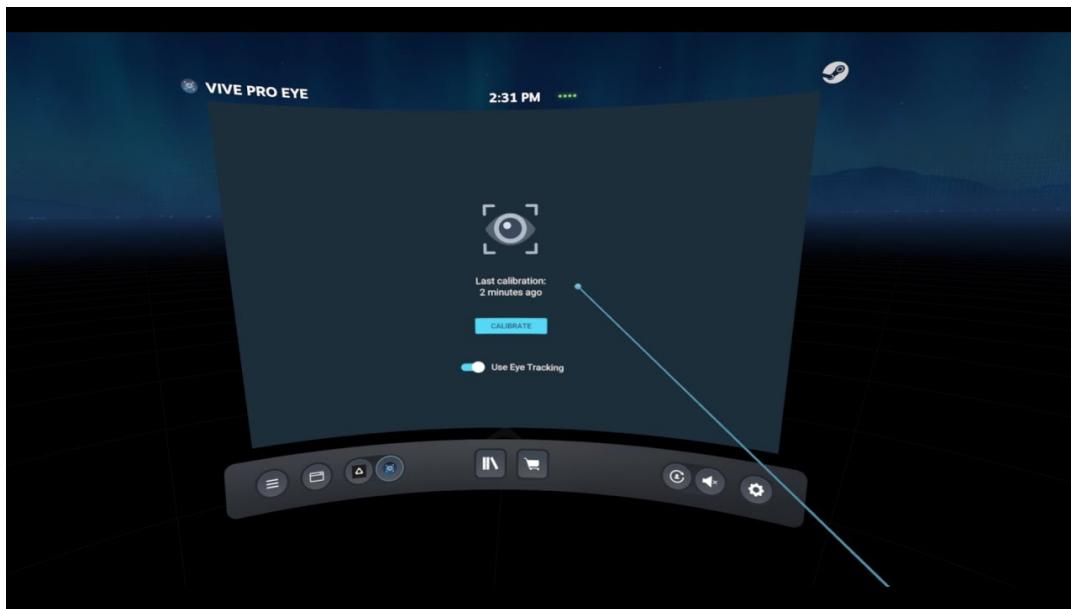
Die SRanipal und Tobii VRU02 Runtimes sind Laufzeitumgebung zur Kommunikation mit dem Eyetracking-System der VR-Brille. Sie müssen installiert werden, um Versuche mit dem Programm durchzuführen. Nachdem die Laufzeitumgebungen installiert wurden, kann das Eyetracking der VR-Brille in den SteamVR-Einstellungen aktiviert werden.

- Registrierung im Vive Developer Portal <https://developer.vive.com/eu/>

- Download und Installation der Laufzeitumgebungen unter <https://hub.vive.com/en-US/download>
- Aktivierung der Eyetracking-Funktionalität in SteamVR
 - SteamVR über Steam starten
 - Controller in die Hand nehmen und VR-Brille aufsetzen
 - Tastendruck auf den Menü-Button des Vive Controllers
 - Klick auf den blauen Vive Pro Eye Button (siehe blaues Icon in Abb. A.1) in VR
 - Aktivierung des Eyetrackings durch Klick auf Toggle (siehe Abb. A.1)
 - Zustimmung zu Nutzungsbestimmungen
 - Durchführung einer ersten Kalibrierung

Abbildung A.1

Aktivierung des Eyetrackings in SteamVR



A.1.4 Python und Bibliotheken

Für die Erstellung des Sequenzdiagramms ruft das Programm ein Python-Skript auf. Dementsprechend muss auf dem System Python3 installiert sein.

- Download und Installation von Python3 unter <https://www.python.org/downloads/>

Die Visualisierung nutzt die folgenden Python-Bibliotheken, welche manuell installiert werden müssen.

- pandas (https://pandas.pydata.org/docs/getting_started/install.html)
- matplotlib (<https://matplotlib.org/stable/users/installing/index.html>)
- argparse (<https://pypi.org/project/argparse/>)

A.1.5 AOI Editor

Zuletzt kann das Programm unter https://github.com/kiliansanchez/360AOIs_public/blob/main/executable.zip heruntergeladen werden. Der Link führt zu einer auf GitHub verfügbaren Zip-Datei. Nach dem Download kann die Datei entpackt werden. Sie enthält einen Ordner, welcher alle für die Ausführung des Programms nötigen Dateien enthält. Durch einen Doppelklick auf die Datei „360AOIs.exe“ wird das Programm ausgeführt.

A.2 Setup des Unity Projekts

Sofern Anpassungen an dem Programm vorgenommen werden möchten, müssen neben den in A.1 beschriebenen Abhängigkeiten zur Ausführung des Programms folgende Schritte getätigt werden.

- Download und Installation von Unity Hub (<https://unity.com/download>)
- Download und Installation von Editor Version 2021.3.16f1 über Unity Hub

Anschließend kann das Projekt geklont oder heruntergeladen und in Unity Hub geöffnet werden. Das Repository des Unity Projekts ist auffindbar unter https://github.com/kiliansanchez/360AOIs_public. Zur Anpassung des Programmcodes empfiehlt es sich, eine IDE wie Visual Studio Community oder Visual Studio Code mit Unity zu verbinden. Eine Anleitung dazu ist hier zu finden <https://docs.unity3d.com/Manual/ScriptingToolsIDEs.html>.

A.3 Hardwarevoraussetzungen

Die genauen Hardwarevoraussetzungen für die Nutzung der Software sind schwer einzuschätzen. Als Richtlinie wurden die Guidelines für die Nutzung der HTC Vive Pro Eye herangezogen („VIVE Pro Eye Specs & User Guide - Developer Resources“, 23/07/2023).

Tabelle A.1*Minimale Hardwarevorraussetzungen*

Prozessor	Intel® Core™ i5-4590 oder AMD FX™ 8350, vergleichbar oder besser.
Grafikkarte	NVIDIA® GeForce® GTX 970 oder AMD Radeon™ R9 290, vergleichbar oder besser.
Arbeitsspeicher	4GB oder mehr
Anschlüsse	1 x DisplayPort 1.2 oder neuer, 1 x USB 3.0 oder neuer

Anhang B

Unity Projekt Aufbau

Dieser Anhang beschreibt den Aufbau des Unity Projekts und soll eine Einarbeitung in dieses vereinfachen. Es handelt sich dabei nicht um eine vollständige Dokumentation des Codes. Für eine Beschreibung aller einzelnen Klassen und Funktionen in englischer Sprache siehe https://kiliansanchez.github.io/360AOIs_public/index.html.

B.1 Gameobjects in Szene

Das Programm besteht aus einer einzigen Unity Szene namens ‚MainScene‘. Die Szene enthält die in Tabelle B.1 gelisteten *Gameobjects*. Sollte die Szene beim ersten Öffnen des Projekts nicht sichtbar sein, muss in den Ordner ‚Scenes‘ navigiert, und ein Doppelklick auf die Datei ‚MainScene‘ getätigt werden.

Tabelle B.1

Beschreibung der Spielobjekte der Unity Szene des Projekts.

GameObject	Skript / Component	Zweck
Application	Setup	Setzen der Zielframerate des Programms auf 120 FPS
	VR Manager	Aktivierung/Deaktivierung der VR Funktionalitäten
	Save And Load AOIs	Speichern und Laden von AOIs
	Gaze To Texture Mapper	(prototypisch) Zuordnung von Blickrichtung zu Pixel in Video

GameObject	Skript / Component	Zweck
	Event Detection Tester	Testskript für Event Detection
	Log Messages Manager	Logging und Anzeige von Fehlermeldungen
	Gaze Simulator	Testskript für Event Detection
	Project Manager	Erstellung und Laden von Projekten
	FPS Counter	Aktualisierung der FPS Anzeige
	Debug Rerun	(prototypisch) Offline Analyse von bereits erhobenen Blickrichtungsdaten
	Visual Angle Tester	(prototypisch) Visualisierung von gegebenem visuellen Winkel (in Grad) in Video.
Editor Camera	Camera Drag	Bewegung der Editor Camera
	Editor Camera	Management der Editor Camera. Setzen von LayerMask und ClearFlags (Skybox vs. Solid Color)
Video Player	Video Player	Video Player Component für Darstellung des Videos
	Video Manager	Management von Videoplayback
	Video Loader	Laden von Video von Dateipfad
AOI Spawner	AOI Manager	Erstellung von AOIs und Management von überlappenden AOIs
UI Canvas		Parent-Object für alle UI Elemente. Buttons, Menüs und Timeline sind Children dieses Objekts und haben jeweils eigene Skripte und Children.
Event System		EventSystem für UI

GameObject	Skript / Component	Zweck
XRRig		Parent Object für VR-Camera. VR-Camera hat eigene Skripte für Setzen von LayerMask und ClearFlags der Kamera.
SRanipal Eye Framework	SRanipal	Management (Aktivierung/Deaktivierung) des SRanipal Frameworks
	Eye_Framework	
	Line Renderer	Component zur Darstellung von Linien in der Spielwelt.
	Eye Recorder	Aufzeichnung der Blickbewegungen
	Gaze Ray Renderer	Darstellung der Blickbewegungen in Versuchsleiteransicht (nutzt Line Renderer)
	Gaze Raycaster	Sendet Raycast der aktuellen Blickrichtung in Spielwelt.
PreVideoSpheres	Pre Video Test Scene	Darstellung der Testszene vor Beginn eines Versuches

B.2 Assets

Die *Gameobjects* und Skripte der ‚MainScene‘ referenzieren Assets, die in verschiedenen Ordnern im Projekt hinterlegt sind. Tabelle B.3 gibt einen groben Überblick über die Struktur der Assets.

Tabelle B.3*Übersicht über Assets*

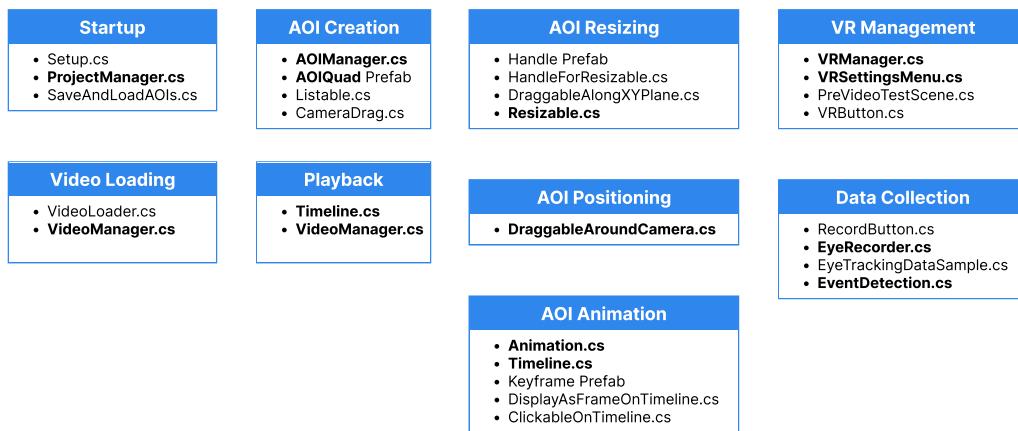
Ordner	Beschreibung
Prefabs	Prefabs (Template Spielobjekte) für AOIs, AOI-Eckpunkte (Handles) und Keyframe-UI-Repräsentationen
Scenes	Enthält die Unity Szene ‚MainScene‘
Scripts	Alle für das Projekt geschriebenen Skripte
SkyboxMaterials	Enthält das Skybox-Material, welches letztlich das Video abbildet
StandaloneFileBrowser	Der UnityStandaloneFileBrowser ist ein Unity Plugin. Es wird in dem Unity-Projekt verwendet, um Windows-Systemdialoge für das Laden und Speichern von Dateien aufzurufen. Der Ordner enthält die von dem Plugin erzeugten Dateien
SteamVR, VR_Input, VR_Resources	Steam- Steam- en Alle von dem SteamVR Plugin erzeugten Dateien
StreamingAssets	Von SteamVR erzeugte Dateien sowie das Python-Skript zur Erstellung von Sequenzdiagrammen
TextMeshPro	Von Unity erzeugte Dateien für die Darstellung von Text
UI Sprites	Sprites (Texturen) für UI-Elemente wie Buttons und Keyframes
UI Toolkit	Von Unity erzeugte Dateien
Videos	Enthält Test-Videos sowie Test-AOI-Dateien
ViveSR	Von dem SRanipal Plugin erzeugte Dateien und Skripte
XR	Von UnityXR Framework erzeugte Dateien

B.3 Skripte

Für das Projekt wurde eine Vielzahl von Skripten geschrieben. Eine vollständige Übersicht aller Skripte ist unter https://kiliansanchez.github.io/360AOIs_public/files.html einzusehen. Abbildung B.1 gibt eine Übersicht über die wichtigsten Skripte des Projekts. Die Skripte sind dabei jeweils nach Funktionalität gruppiert und von links nach rechts in etwa so angeordnet, wie sie im zeitlichen Verlauf der Programmausführung abgerufen werden.

Abbildung B.1

Übersicht über die wichtigsten Skripte



Anmerkung. Besonders relevante und umfangreiche Skripte sind fettgedruckt abgebildet.

Anhang C

HTC Vive Pro Eye Spezifikationen

Die Spezifikationen der HTC Vive Pro Eye sind in Tabelle C.1 gelistet. Die VR-Brille wird mit bis zu vier SteamVR Base Stations 2.0. betrieben. Dabei handelt es sich um kleine quaderförmige Trackinggeräte, die im Raum verteilt werden, um das Headset und die Controller zu tracken. Die Installation und Einrichtung des Headsets erfolgt über den HTC Vive Pro Setup Wizard (<https://business.vive.com/de/setup/vive-pro/>).

Tabelle C.1*Spezifikationen der HTC Vive Pro Eye*

Bildschirm(e)	Bildschirmgröße	Dual OLED 3,5 Zoll (8.89 cm)
	Auflösung	1440 x 1600 px pro Auge (2880 x 1600 px kombiniert)
	Bildwiederholfrequenz	90 Hz
	Sichtfeld	110°
Audio	Stereo Kopfhörer	
	Dual integrierte Mikrofone	
Anschlüsse	USB-C 3.0, DP 1.2, Bluetooth	
Sensoren	SteamVR Tracking	
	G-sensor	
	Gyroscope	
	Proximity	
	Eye Comfort Setting (IPD)	
	Eyetracking (powered by Tobii ®)	
Eyetracking	Frequenz	120 Hz
	Genaugigkeit (innerhalb Sichtfeld von 20°)	0.5° - 1.1°
	Kalibrierung	5-Punkt
	Timestamp (device and system)	
	Gaze origin	
	Gaze direction	
	Pupil position	
	Pupil size	
	Eye openness	
Interface	HTC SRanipal SDK	
SDK Engine Kompatibilität	Unity, Unreal	

Anmerkung. Aus „VIVE Pro Eye Specs & User Guide - Developer Resources“ (23/07/2023).

Anhang D

Übersicht eingesetzter Software und Plugins

Dieser Anhang bietet eine vollständige Übersicht aller eingesetzten Programme und Plugins.

Tabelle D.1

Software und Plugins

Name	Version
Unity Hub	3.4.2
Unity Editor	2021.3.16f1
Steam	1690583737
SteamVR	11805697
SteamVR Unity Plugin	2.7.3
Vive Pro HMD Setup	unbekannt
SRanipal Runtime	VIVE_SRanipal- Installer_1.3.2.0.msi
Tobii VRU02 Runtime	VIVE_SRanipal- Installer_1.3.2.0.msi
SRanipal SDK	1.3.6.6
OpenXR Plugin	1.5.3
Unity Standalone FileBrowser	Nov 6, 2018
Python3	3.11.4
matplotlib	3.7.2
pandas	2.0.3
argparse	3.2
vr-idt	0.0.5
Kinovea	0.9.5
OBS Studio	29.1.3
Spyder	5.4.2

Anhang E

Studienergebnisse

E.1 Versuchsperson 1

Tabelle E.1

VP1: Ergebnisse des manuellen Codings

AOI	Dwell Count	Time To First Dwell in ms	First Pass Dwell Time in ms	Total Dwell Time in ms
Small Advert.	1	3433	3350	3350
Big Advert.	3	16233	33	4150
McDonalds	1	12275	3000	3000
Person	2	7892	3558	5541

AOI	Time To First Fixation in ms	First Fixation Duration in ms
Small Advert.	5000	242
Big Advert.	17308	258
McDonalds	12917	317
Person	8567	258

Tabelle E.2

VP1: Werte des Datenexports

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	1	3433	3341	3341
Big Advert.	3	16230	42	4149
McDonalds	1	12281	2999	2999
Person	2	7890	3558	5532

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	4999	242
Big Advert.	17280	166
McDonalds	12939	291
Person	8573	250

Tabelle E.3

VP1: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	0	0	9	9
Big Advert.	0	3	9	1
McDonalds	0	6	1	1
Person	0	2	0	9

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	1	0
Big Advert.	28	92
McDonalds	22	26
Person	6	8

E.2 Versuchsperson 2

Tabelle E.4

VP2: Ergebnisse des manuellen Codings

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	3	3067	517	4567
Big Advert.	1	19542	4608	4608
McDonalds	2	9325	442	3000
Person	4	13800	8	7141

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	4125	367
Big Advert.	21142	158
McDonalds	10408	317
Person	14242	225

Tabelle E.5

VP2: Werte des Datenexports

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	3	3066	517	4574
Big Advert.	1	19538	4607	4607
McDonalds	2	9315	450	3007
Person	4	13789	8	7124

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	4116	367
Big Advert.	21304	242
McDonalds	10407	316
Person	14231	225

Tabelle E.6

VP2: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	0	1	0	7
Big Advert.	0	4	1	1
McDonalds	0	10	8	7
Person	0	11	0	17

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	9	0
Big Advert.	162	84
McDonalds	1	1
Person	11	0

E.3 Versuchsperson 3

Tabelle E.7

VP3: Ergebnisse des manuellen Codings

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	1	6658	4000	4000
Big Advert.	1	27392	6700	6700
McDonalds	2	12667	4467	4484
Person	1	18650	5758	5758

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	7275	875
Big Advert.	27958	158
McDonalds	13583	150
Person	20058	192

Tabelle E.8

VP3: Werte des Datenexports

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	1	6657	3999	3999
Big Advert.	1	27386	6690	6690
McDonalds	2	12656	4466	4482
Person	1	18646	5749	5749

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	7390	750
Big Advert.	27953	150
McDonalds	13581	158
Person	20038	183

Tabelle E.9

VP3: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms		
Small Advert.	0	1	1	1
Big Advert.	0	6	10	10
McDonalds	0	11	1	2
Person	0	4	9	9

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	115	125
Big Advert.	5	8
McDonalds	2	8
Person	20	9

E.4 Versuchsperson 4

Tabelle E.10

VP4: Ergebnisse des manuellen Codings

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	1	4000	4350	4350
Big Advert.	2	22742	5908	6708
McDonalds	2	10075	1950	5091
Person	2	16675	3425	5308

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	4650	167
Big Advert.	24158	150
McDonalds	10608	275
Person	17500	150

Tabelle E.11

VP4: Werte des Datenexports

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	1	3999	4349	4349
Big Advert.	2	22745	5915	6715
McDonalds	2	10073	1958	5091
Person	2	16671	3425	5308

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	4641	175
Big Advert.	24170	150
McDonalds	10589	284
Person	17496	150

Tabelle E.12

VP4: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	0	1	1	1
Big Advert.	0	3	7	7
McDonalds	0	2	8	0
Person	0	4	0	0

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	9	8
Big Advert.	12	0
McDonalds	19	9
Person	4	0

E.5 Versuchsperson 5

Tabelle E.13

VP5: Ergebnisse des manuellen Codings

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time	Time in ms
Small Advert.	5	2617	567	6409
Big Advert.	3	29792	3858	11717
McDonalds	3	12658	3192	6767
Person	2	20233	7367	8667

AOI	Time To First	First Fixation
	Fixation in	Duration in
	ms	ms
Small Advert.	5500	150
Big Advert.	30808	175
McDonalds	13733	258
Person	20992	192

Tabelle E.14*VP5: Werte des Datenexports*

AOI	Dwell Count	Time To First	First Pass Dwell Time in ms	Total Dwell Time in ms
		Dwell in ms		
Small Advert.	5	2617	566	6389
Big Advert.	3	29786	3857	11714
McDonalds	3	12648	3199	6773
Person	2	20229	7365	8664

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	5491	150
Big Advert.	30769	191
McDonalds	13731	250
Person	20979	192

Tabelle E.15*VP5: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding*

AOI	Dwell Count	Time To First	First Pass Dwell Time in ms	Total Dwell Time in ms
		Dwell in ms		
Small Advert.	0	0	1	20
Big Advert.	0	6	1	3
McDonalds	0	10	7	6
Person	0	4	2	3

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	9	0
Big Advert.	39	16
McDonalds	2	8
Person	13	0

E.6 Versuchsperson 6

Tabelle E.16

VP6: Ergebnisse des manuellen Codings

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	2	4342	1992	4900
Big Advert.	5	28675	500	9516
McDonalds	1	10267	7067	7067
Person	2	18125	9450	9892

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	5158	225
Big Advert.	28958	183
McDonalds	11767	225
Person	19208	233

Tabelle E.17

VP6: Werte des Datenexports

AOI	Dwell Count	Time To First	First Pass	Total Dwell
		Dwell in ms	Dwell Time in ms	Time in ms
Small Advert.	2	4341	2000	4899
Big Advert.	5	28652	525	9556
McDonalds	1	10273	7074	7074
Person	2	18130	9456	9897

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	5166	225
Big Advert.	28977	175
McDonalds	11789	284
Person	19196	233

Tabelle E.18

VP6: Absoluter Fehler zwischen CSV-Datenexport und manuellem Coding

AOI	Dwell Count	Time To First	First Pass Dwell Time in ms	Total Dwell Time in ms
		Dwell in ms		
Small Advert.	0	1	8	1
Big Advert.	0	23	25	40
McDonalds	0	6	7	7
Person	0	5	6	5

AOI	Time To First	First Fixation
	Fixation in ms	Duration in ms
Small Advert.	8	0
Big Advert.	19	8
McDonalds	22	59
Person	12	0

Literatur

- Bonikowski, L., Gruszczyński, D., & Matulewski, J. (2021). Open-source Software for Determining the Dynamic Areas of Interest for Eye Tracking Data Analysis. *Procedia Computer Science*, 192, 2568–2575. <https://doi.org/10.1016/j.procs.2021.09.026>
- Bonnel, A. M., & Hafter, E. R. (1998). Divided attention between simultaneous auditory and visual signals. *Perception & Psychophysics*, 60(2), 179–190. <https://doi.org/10.3758/BF03206027>
- Brishtel, I., Khan, A. A., Schmidt, T., Dingler, T., Ishimaru, S., & Dengel, A. (2020). Mind Wandering in a Multimodal Reading Setting: Behavior Analysis & Automatic Detection Using Eye-Tracking and an EDA Sensor. *Sensors*, 20(9). <https://doi.org/10.3390/s20092546>
- Cassin, B., & Rubin, M. L. (2006). *Dictionary of eye terminology* (5th ed.). Triad Pub.
- Charmant, J. (24/07/2023). Kinovea. <https://www.kinovea.org/>
- Clay, V., König, P., & König, S. (2019). Eye Tracking in Virtual Reality. *Journal of Eye Movement Research*, 12(1). <https://doi.org/10.16910/jemr.12.1.3>
- Dalmaijer, E. S., Mathôt, S., & van der Stigchel, S. (2013). PyGaze: an open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments. *Behavior Research Methods*, 46(4), 913–921. <https://doi.org/10.3758/s13428-013-0422-2>
- de Margerie, T. (2017). How to integrate 360 video with Unity. *Unity Blog*, 2017. <https://blog.unity.com/technology/how-to-integrate-360-video-with-unity>
- Denzin, N. K. (2017). *The research act: A theoretical introduction to sociological methods*. Routledge.
- Desai, P. R., Desai, P. N., Ajmera, K. D., & Mehta, K. (2014). A Review Paper on Oculus Rift-A Virtual Reality Headset. <https://arxiv.org/pdf/1408.1173>
- Duchowski, A. T. (2002). A breadth-first survey of eye-tracking applications. *Behavior research methods, instruments, & computers : a journal of the Psychonomic Society, Inc*, 34(4), 455–470. <https://doi.org/10.3758/BF03195475>
- Duchowski, A. T. (2017). *Eye Tracking Methodology*. Springer International Publishing. <https://doi.org/10.1007/978-3-319-57883-5>
- Gherri, E., & Eimer, M. (2011). Active listening impairs visual perception and selectivity: an ERP study of auditory dual-task costs on visual attention. *Journal of cognitive neuroscience*, 23(4), 832–844. <https://doi.org/10.1162/jocn.2010.21468>

- Goldstein, E. B. (2010). *Sensation and perception* (8. ed., internat. ed.). Wadsworth Cengage Learning.
- Hessels, R. S., Kemner, C., van den Boomen, C., & Hooge, I. T. C. (2016). The area-of-interest problem in eyetracking research: A noise-robust solution for face and sparse stimuli. *Behavior Research Methods*, 48(4), 1694–1712. <https://doi.org/10.3758/s13428-015-0676-y>
- Hoffman, J. E. (2016). Visual attention and eye movements. In H. Pashler (Hrsg.), *Attention: Studies in Cognition* (S. 119–153). Taylor & Francis.
- Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., & de van Weijer, J. (2011). *Eye Tracking: A comprehensive guide to methods and measures*. OUP Oxford.
- Howard, I. P., & Rogers, B. J. (1995). *Binocular vision and stereopsis* (Bd. no. 29). Oxford University Press. <https://doi.org/10.1093/acprof:oso/9780195084764.001.0001>
- iMotions. (6/15/2022). Eye Tracking Software for Virtual Reality - iMotions. <https://imotions.com/products/imotions-lab/modules/eye-tracking-virtual-reality/>
- iMotions. (10/31/2022). Pricing Plans for Academia and Business - iMotions. <https://imotions.com/products/pricing/>
- Javal, L. É. (1878). *Essai sur la physiologie de la lecture*. https://pure.mpg.de/rest/items/item_2350899/component/file_2350898/content
- Just, M. A., & Carpenter, P. A. (1980). A theory of reading: From eye fixations to comprehension. *Psychological Review*, 87, 329–354. <https://doi.org/10.1037/0033-295X.87.4.329>
- Kiefer, P., Giannopoulos, I., Raubal, M., & Duchowski, A. (2017). Eye tracking for spatial research: Cognition, computation, challenges. *Spatial Cognition & Computation*, 17(1-2), 1–19. <https://doi.org/10.1080/13875868.2016.1254634>
- Koulieris, G. A., Akşit, K., Stengel, M., Mantiuk, R. K., Mania, K., & Richardt, C. (2019). Near-Eye Display and Tracking Technologies for Virtual and Augmented Reality. *Computer Graphics Forum*, 38(2), 493–519. <https://doi.org/10.1111/cgf.13654>
- Kramida, G. (2016). Resolving the Vergence-Accommodation Conflict in Head-Mounted Displays. *IEEE transactions on visualization and computer graphics*, 22(7), 1912–1931. <https://doi.org/10.1109/TVCG.2015.2473855>
- LaValle, S. M. (2023). *Virtual Reality*. <https://www.cambridge.org/core/books/virtual-reality/0EC0542C3688B5CA3E9733011DC8DEC2>
- Lee, H.-H., Chen, Z.-L., Yeh, S.-L., Hsiao, J. H., & Wu, A.-Y. A. (2021). When Eyes Wander Around: Mind-Wandering as Revealed by Eye Movement Analysis with Hidden Markov Models. *Sensors*, 21(22). <https://doi.org/10.3390/s21227569>
- Levin, L. A., & Adler, F. H. (Hrsg.). (2011). *Adler's physiology of the eye: [expert consult; activate at expertconsult.com; searchable full text online]* (11. ed.). Saunders

- Elsevier. http://bvbr.bib-bvb.de:8991/F?func=service&doc_library=BVB01&local_base=BVB01&doc_number=022530473&sequence=000002&line_number=0001&func_code=DB_RECORDS&service_type=MEDIA
- Llanes-Jurado, J., Marín-Morales, J., Guixeres, J., & Alcañiz, M. (2020). Development and Calibration of an Eye-Tracking Fixation Identification Algorithm for Immersive Virtual Reality. *Sensors*, 20(17), 4956. <https://doi.org/10.3390/s20174956>
- M. Brescia-Zapata, K. Krejtz, A. T. Duchowski, C. J. Hughes & P. Orero. (2023). Eye-tracked Evaluation of Subtitles in Immersive VR 360° Video. *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, 769–770. <https://doi.org/10.1109/VRW58643.2023.00227>
- Mangiante, S., Klas, G., Navon, A., GuanHua, Z., Ran, J., & Silva, M. D. (2017). VR is on the Edge. *VR/AR Network '17*, 30–35. <https://doi.org/10.1145/3097895.3097901>
- Marchewka, M., Nesterak, J., Sołtysik, M., Szymla, W., & Wojnarowska, M. (2020). Multitasking effects on individual performance : an experimental eye-tracking study. 11082976. <https://www.um.edu.mt/library/oar/handle/123456789/56205>
- Martin, D., Serrano, A., Bergman, A. W., Wetzstein, G., & Masia, B. (2021). ScanGAN360: A Generative Model of Realistic Scanpaths for 360° Images. <https://arxiv.org/pdf/2103.13922>
- matplotlib.pyplot.eventplot — Matplotlib 3.1.2 documentation. (03/06/2023). https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.eventplot.html
- Meißner, M., Pfeiffer, J., Pfeiffer, T., & Oppewal, H. (2019). Combining virtual reality and mobile eye tracking to provide a naturalistic experimental environment for shopper research. *Journal of Business Research*, 100, 445–458. <https://doi.org/10.1016/j.jbusres.2017.09.028>
- Muhanna, M. A. (2015). Virtual reality and the CAVE: Taxonomy, interaction challenges and research directions. *Journal of King Saud University - Computer and Information Sciences*, 27(3), 344–361. <https://doi.org/10.1016/j.jksuci.2014.03.023>
- Nielsen Norman Group. (08/01/2021). Maintain Consistency and Adhere to Standards (Usability Heuristic #4). <https://www.nngroup.com/articles/consistency-and-standards/>
- Open Broadcaster Software | OBS. (24/07/2023). <https://obsproject.com/>
- Pallavicini, F., Pepe, A., & Minissi, M. E. (2019). Gaming in Virtual Reality: What Changes in Terms of Usability, Emotional Response and Sense of Presence Compared to Non-Immersive Video Games? *Simulation & Gaming*, 50(2), 136–159. <https://doi.org/10.1177/1046878119831420>
- Pan, X., & Hamilton, Antonia F. de C. (2018). Why and how to use virtual reality to study human social interaction: The challenges of exploring a new research

- landscape. *British journal of psychology (London, England : 1953)*, 109(3), 395–417. <https://doi.org/10.1111/bjop.12290>
- Papenmeier, F., & Huff, M. (2010). DynAOI: a tool for matching eye-movement data with dynamic areas of interest in animations and movies. *Behavior Research Methods*, 42(1), 179–187. <https://doi.org/10.3758/BRM.42.1.179>
- Pidgeon, E. (2017). Windows 10 Tip: Watch 360° videos with the Windows 10 Creators Update. <https://blogs.windows.com/windowsexperience/2017/06/19/windows-10-tip-watch-360-videos-windows-10-creators-update/>
- Płużyczka, M. (2018). The First Hundred Years: a History of Eye Tracking as a Research Method. *Applied Linguistics Papers*, 4/2018(25), 101–116. <https://doi.org/10.32612/uw.25449354.2018.4.pp.101-116>
- Posner, M. I. (1980). Orienting of attention. *The Quarterly journal of experimental psychology*, 32(1), 3–25. <https://doi.org/10.1080/00335558008248231>
- PyMesh — Geometry Processing Library for Python — PyMesh 0.2.1 documentation. (29/01/2021). <https://pymesh.readthedocs.io/en/latest/>
- PyPI. (24/07/2023). vr-idt. <https://pypi.org/project/vr-idt/>
- Rossetti, T., & Hurtubia, R. (2020). An assessment of the ecological validity of immersive videos in stated preference surveys. *Journal of Choice Modelling*, 34, 100198. <https://doi.org/10.1016/j.jocm.2019.100198>
- Rötting, M. (2001). *Parametersystematik der Augen- und Blickbewegungen für arbeitswissenschaftliche Untersuchungen* (Bd. Bd. 34). Shaker.
- Saad, A., Liebers, J., Gruenefeld, U., Alt, F., & Schneegass, S. (2021). Understanding Bystanders' Tendency to Shoulder Surf Smartphones Using 360-degree Videos in Virtual Reality. *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*, 1–8. <https://doi.org/10.1145/3447526.3472058>
- Schicks, F. (2022). *Entwicklung dynamischer Areas of Interest in 360° Videos in Unity* [Bachelorarbeit]. Technische Universität Berlin.
- Shannon, C. E. (1949). Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1), 10–21. <https://doi.org/10.1109/JRPROC.1949.232969>
- Shibata, T. (2002). Head mounted display. *Displays*, 23(1-2), 57–64. [https://doi.org/10.1016/S0141-9382\(02\)00010-0](https://doi.org/10.1016/S0141-9382(02)00010-0)
- Shu, Y., Huang, Y.-Z., Chang, S.-H., & Chen, M.-Y. (2019). Do virtual reality head-mounted displays make a difference? A comparison of presence and self-efficacy between head-mounted displays and desktop computer-facilitated virtual environments. *Virtual Reality*, 23(4), 437–446. <https://doi.org/10.1007/s10055-018-0376-x>
- Shumway, C. L., Motlagh, M., & Wade, M. (2023). *Anatomy, Head and Neck, Eye Extraocular Muscles*.
- Sightlab VR Pro. (18/08/2023). SightLab VR Documentation - Regions/Areas of Interest for 360 Video. <https://sites.google.com/worldviz.com/sightlab-vr-documentation/regionsareas-of-interest-for-360-video?authuser=0>

- Sightlab VR Pro. (27/07/2023). Virtual Reality Eye Tracking for Research | WorldViz. <https://www.worldviz.com/virtual-reality-eye-tracking-for-research-solutions>
- Slater, M. (2018). Immersion and the illusion of presence in virtual reality. *British journal of psychology (London, England : 1953)*, 109(3), 431–433. <https://doi.org/10.1111/bjop.12305>
- Smallwood, J., & Schooler, J. W. (2015). The science of mind wandering: empirically navigating the stream of consciousness. *Annual review of psychology*, 66, 487–518. <https://doi.org/10.1146/annurev-psych-010814-015331>
- Spyder. (09/06/2023). Home — Spyder IDE. <https://www.spyder-ide.org/>
- Sullivan, H., Boudewyns, V., O'Donoghue, A., Marshall, S., & Williams, P. A. (2017). Attention to and Distraction from Risk Information in Prescription Drug Advertising: An Eye Tracking Study. *Journal of public policy & marketing : JPP&M : an annual publication of the Division of Research, Graduate School of Business Administration, the University of Michigan*, 36(2), 236–245. <https://doi.org/10.1509/jppm.16.013>
- Tawa, J., Lang, Y., & St. John, A. (2022). *Measuring Visual Attention with 360 Degree Video Stimuli in Virtually Immersive Environments: A Case Study of Police Officers' Decisions to Shoot*. <https://doi.org/10.2139/ssrn.4063506>
- Tech at Meta. (2022). Passing the visual Turing test: The inside story of our quest for visual realism in VR.
- Tobii Devzone. (10/07/2023a). Licenses. <https://developer.tobii.com/xr/licenses/>
- Tobii Devzone. (10/07/2023b). Tobii G2OM. <https://developer.tobii.com/xr/solutions/tobii-g2om/>
- Tobii Pro Lab. (27/07/2023). Eye tracking software for behavior research. <https://www.tobii.com/products/software/behavior-research-software/tobii-pro-lab>
- Tovée, M. J. (2009). *An introduction to the visual system* (2. ed., repr. with corr). Cambridge Univ. Press.
- Unity Learn. (12/06/2023). Learn game development w/ Unity | Courses & tutorials in game design, VR, AR, & Real-time 3D | Unity Learn. <https://learn.unity.com/>
- Unity Technologies. (02/06/2023a). Unity - Manual: Panoramic skybox. <https://docs.unity3d.com/Manual/shader-skybox-panoramic.html>
- Unity Technologies. (14/07/2023a). Unity - Manual: Render Texture. <https://docs.unity3d.com/Manual/class-RenderTexture.html>
- Unity Technologies. (21/07/2023). Unity - Manual: UI Toolkit. <https://docs.unity3d.com/Manual/UIElements.html>
- Unity Technologies. (24/02/2021). Unity - Manual: What is multithreading? <https://docs.unity3d.com/2020.1/Documentation/Manual/JobSystemMultithreading.html>

- Unity Technologies. (14/07/2023b). Unity - Scripting API: Mesh. <https://docs.unity3d.com/ScriptReference/Mesh.html>
- Unity Technologies. (14/07/2023c). Unity - Scripting API: MonoBehaviour.InvokeRepeating. <https://docs.unity3d.com/ScriptReference/MonoBehaviour.InvokeRepeating.html>
- Unity Technologies. (14/07/2023d). Unity - Scripting API: MonoBehaviour.Update(). <https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html>
- Unity Technologies. (14/07/2023e). Unity - Scripting API: Physics.Raycast. <https://docs.unity3d.com/ScriptReference/Physics.Raycast.html>
- Unity Technologies (Hrsg.). (02/06/2023b). Unity - Scripting API: VideoPlayer. <https://docs.unity3d.com/ScriptReference/Video.VideoPlayer.html>
- Unity Technologies. (20/07/2023). Unity Personal. <https://unity.com/products/unity-personal>
- Varjo.com. (06/06/2023). High-Resolution Virtual and Mixed Reality Products – Varjo.com. <https://varjo.com/products/>
- Vive Developers. (20/07/2023). Eye and Facial Tracking SDK - Developer Resources. <https://developer.vive.com/resources/vive-sense/eye-and-facial-tracking-sdk/>
- VIVE Pro Eye Specs & User Guide - Developer Resources. (23/07/2023). <https://developer.vive.com/resources/hardware-guides/vive-pro-eye-specs-user-guide/>
- Vosskühler, A., Nordmeier, V., Kuchinke, L., & Jacobs, A. M. (2008). OGAMA (Open Gaze and Mouse Analyzer): open-source software designed to analyze eye and mouse movements in slideshow study designs. *Behavior Research Methods*, 40(4), 1150–1162. <https://doi.org/10.3758/BRM.40.4.1150>
- Wilkinson, M., Brantley, S., & Feng, J. (2021). A Mini Review of Presence and Immersion in Virtual Reality. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 65(1), 1099–1103. <https://doi.org/10.1177/1071181321651148>
- Younis, O., Al-Nuaimy, W., H., M., & Rowe, F. (2019). A Hazard Detection and Tracking System for People with Peripheral Vision Loss using Smart Glasses and Augmented Reality. *International Journal of Advanced Computer Science and Applications*, 10(2), 1–9. <https://doi.org/10.14569/IJACSA.2019.0100201>
- YouTube VR. (22/07/2023). Watch - YouTube VR. <https://vr.youtube.com/watch/>