Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Entwicklung eines Prototyps für Eye-Tracking-Aufnahmen mit zwei Bildschirmen

Developing a prototype for eye tracking recordings for two screens

Bachelorarbeit

im Studiengang Informatik

von

Thorben Kurowski

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Dr. Jil Klünder
Betreuer: M.Sc. Maike Ahrens

Hannover, 06.07.2023

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 06.07.2023

Thorben Kurowski

Zusammenfassung

Die Nutzung von zwei Bildschirmen im Rahmen von Eye-Tracking-Aufnahmen ermöglicht die Abbildung realistischer Arbeitsbedingungen. Viele Eye-Tracker erlauben jedoch nur die gleichzeitige Nutzung eines Eye-Trackers für Eye-Tracking-Aufnahmen. Der in dieser Arbeit entwickelte Prototyp löst dieses Problem, indem er Eye-Tracking-Aufnahmen mit zwei Tobii X3-120 Eye-Trackern ermöglicht.
Durch die Nutzung von zwei Tobii Eye-Trackern kann der Prototyp Eye-Tracking-Daten von zwei Bildschirmen gleichzeitig erfassen. Dazu nutzt die Anwendung das Tobii Pro SDK, um beide Eye-Tracker zu steuern und erstellt gleichzeitig eine Bildschirmaufnahme. Durch einen eigens entwickelten Algorithmus zur Zusammenführung der Eye-Tracking Daten wird eine einzige Ausgabedatei erzeugt, welche die Eye-Tracking-Daten der beiden Bildschirme enthält. Des Weiteren wird die Ausgabedatei um eine Filterung nach Augenbewegungen ergänzt, die eine direkte Interpretation und Weiterverarbeitung der Daten ermöglicht. Dabei ist die Einrichtung und Steuerung der Eye-Tracker über eine grafische Benutzeroberfläche möglich, die bei der Bedienung der Anwendung unterstützt.
Durch eine Evaluation wurde sichergestellt, dass der Prototyp korrekte Gaze-Daten erfasst und dass die Erkennung des aktuell betrachteten Bildschirms sowie die Filterung nach Augenbewegungen zuverlässig funktioniert. Zudem wurden im Rahmen der Evaluation Interferenzen durch die gleichzeitige Nutzung von zwei Eye-Trackern erkannt und durch die Positionierung der Bildschirme mit einem Abstand von mindestens 15cm zueinander vermieden. Der entwickelte Prototyp ermöglicht also Eye-Tracking-Aufnahmen mit zwei Bildschirmen und bietet trotz der Vorgaben hinsichtlich der Positionierung der Bildschirme eine Verbesserung gegenüber der bisherigen Beschränkung auf einen Bildschirm.

Abstract

Developing a prototype for eye tracking recordings for two screens

Supporting eye tracking data collection from two screens enables the recording of gaze data in realistic work environments. However, many eye trackers allow the use of only one eye tracker at a time for eye tracking recordings. The prototype developed in this thesis solves this problem by allowing eye-tracking recordings with two Tobii X3-120 eye trackers.
By using two Tobii eye trackers, the prototype can capture eye tracking data from two screens simultaneously. To do this, the application uses the Tobii Pro SDK to control both eye trackers and creates a screen capture simultaneously. Using a custom algorithm to merge the eye-tracking data, a single output file is created that contains the eye-tracking data from the two screens. Furthermore, the data is filtered by eye movements and the results are added to the output file, allowing direct interpretation and further processing of the data. Thereby, the setup and control of the eye-tracker is possible via a graphical user interface, which supports the operation of the application.
An evaluation ensured that the prototype captures correct gaze data and that the detection of the currently viewed screen as well as the filtering according to eye movements works reliably. In addition, the evaluation detected interference from the simultaneous use of two eye trackers and avoided it by positioning the screens at least 15cm apart. Thus, the developed prototype allows eye-tracking recordings with two screens and offers an improvement over the previous limitation to one screen despite the specifications regarding the positioning of the screens.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der Erfolg vieler Unternehmen basiert auf dem Einsatz von qualitativ hochwertiger Software. Die Entwicklung dieser Software ist jedoch ein komplexer Prozess, der aus vielen verschiedenen Gründen oft zu scheitern droht. So wurde 1995 im CHAOS Report der Standish Group festgestellt, dass nur 16,2% aller Softwareprojekte erfolgreich abgeschlossen werden. Die übrigen Softwareprojekte werden zu spät oder gar nicht fertiggestellt, entsprechen nach der Fertigstellung nicht den erwarteten Anforderungen oder haben ihr geplantes Budget signifikant überschritten. Als ein entscheidender Faktor für das Scheitern von Softwareprojekten werden dabei unvollständige Anforderungen genannt [25].
Um unvollständige Anforderungen im Rahmen eines Softwareprojekts zu vermeiden, zielt das Requirements Engineering auf die systematische Erhebung von Anforderungen ab und kann somit den Grundstein für ein erfolgreiches Softwareprojekt legen [5]. Neben dem Requirements Engineering ist auch die Durchführung von geeigneten Qualitätsmaßnahmen im Gebiet der Softwarequalität ein wichtiger Bestandteil in der erfolgreichen Entwicklung von Software.
Bei der Entwicklung geeigneter Methoden und Modelle, die den Softwareentwicklungsprozess im Rahmen des Software Engineerings unterstützen, kommt Eye-Tracking bereits seit Jahren zum Einsatz, um unter anderem Codeverständnis, Debugging und Traceability zu untersuchen [24, 23]. So hat Bednarik bspw. die Debugging-Strategien von Entwicklern mit unterschiedlichen Erfahrungsniveaus untersucht und nutzte dafür Eye-Tracking-Aufnahmen. Dabei konnte anhand der Aufnahmen ermittelt werden, welche Programmdarstellungen (Code, Output, Projekthierarchie, Spezifikation, etc.) von Entwicklern im Rahmen des Debuggings betrachtet wurden und wie häufig zwischen verschiedenen Darstellungen gewechselt wurde [3].
Ali et. al [1] haben untersucht, wie bestehende Information Retrieval Techniken zur Erstellung von Requirements Traceability (RT) Links mit der Hilfe von Eye-Tracking verbessert werden können. Dabei untersuchten sie anhand

von Eye-Tracking-Aufnahmen, wie Entwickler RT Links verifizieren und leiteten daraus neue Gewichtungen ab, um die Genauigkeit der Information Retrieval Techniken zu verbessern.
Anhand der genannten Beispiele lässt sich also erkennen, dass Eye-Tracking im Rahmen von wissenschaftlichen Arbeiten im Software Engineering gewinnbringend eingesetzt werden kann, um geeignete Methoden und Modelle zu entwickeln und zu verbessern. Diese Methoden und Modelle können den Softwareentwicklungsprozess unterstützen und somit zum erfolgreichen Abschluss eines Softwareprojekts beitragen. Daher ist es förderlich Eye-Tracking in verschiedenen Umgebungen und Szenarien einsetzbar zu machen.

1.1 Problemstellung

Gängige Eye-Tracking Geräte erlauben an einem Computer in der Regel nur die Aufzeichnung von Eye-Tracking-Daten eines Bildschirms gleichzeitig. Dies ist eine entscheidende Limitierung bei der Durchführung von Studien, da im professionellen Arbeitsalltag oft Multi-Monitor-Setups genutzt werden. Somit muss bei der Verwendung eines solchen Setups im Rahmen von Eye-Tracking-Aufnahmen auf einen Teil der Eye-Tracking-Daten verzichtet werden oder man beschränkt sich auf die Nutzung eines einzelnen Monitors. In beiden Fällen leidet die Aussagekraft der gesammelten Daten, da sie entweder unvollständig sind oder nicht aus einer realistischen Arbeitsumgebung stammen. Besonders bei Eye-Tracking-Aufnahmen im Bereich des Software Engineerings führt dies zu Problemen, wenn von Entwicklern in einem Multi-Monitor Setup zwischen Code, Anforderungen und Dokumentation auf mehreren Bildschirmen gewechselt wird und sich diese Arbeitsweise nur unter starken Einschränkungen auf einen einzigen Bildschirm reduzieren lässt.

1.2 Lösungsansatz

Im Rahmen dieser Arbeit soll daher ein Prototyp entwickelt werden, der an einem Computer Eye-Tracking-Aufnahmen von zwei Bildschirmen, an denen jeweils ein Tobii X3-120 Eye-Tracker angebracht ist, gleichzeitig ermöglicht. Dazu wird das Tobii Pro SDK (Software Development Kit) des Herstellers Tobii verwendet, welches Entwicklern die Möglichkeit bietet, direkt mit den Geräten des Herstellers zu interagieren [29].
Die Konfiguration und Steuerung der beiden Eye-Tracker soll mit Hilfe des Tobii Pro SDKs über eine grafische Benutzeroberfläche möglich sein. Dabei kann die Kalibrierung der Geräte weiterhin mit dem Tobii Eye Tracker Manager durchgeführt werden, bei welchem es sich um die bisher verwendete Software des Herstellers zur Konfiguration und Kalibrierung

der Eye-Tracker handelt. Zudem soll eine zu den Eye-Tracking-Aufnahmen synchrone Bildschirmaufnahme durchgeführt werden, um anhand dieser später nachvollziehen zu können, welche Elemente zu einem Zeitpunkt an welcher Position sichtbar waren. Da es aufgrund der Nutzung von zwei Bildschirmen zu unvermeidbaren Kopfbewegungen beim Wechsel zwischen den Bildschirmen kommt, soll eine Überwachung der aktuell erfassten Tracking-Informationen auf einem dritten Bildschirm möglich sein.
Die erfassten Eye-Tracking-Daten werden in einem nächsten Schritt aufbereitet, indem sie den entsprechenden Bildschirmen zugeordnet werden, die Daten des aktuell nicht betrachteten Bildschirms entfernt werden und eine Filterung nach der Betrachtungsdauer einzelner Punkte vorgenommen wird. Anschließend werden die aufbereiteten Eye-Tracking-Daten in ein Format gebracht, welches den exportierten Dateien des Tobii Pro Lab entspricht, und als Datei im tsv-Format abgespeichert. Dies ermöglicht eine Weiterverarbeitung mit kompatiblen Tools, wie dem von Eggers [10] entwickelten Tool zur automatischen Erkennung von Textelementen in Bildschirmaufnahmen für die Eye-Tracking Datenanalyse und dem von Sasse [22] entwickelten Tool zur Erkennung von Blickmustern in Eye Tracking Daten. Auf diese Weise können direkt wertvolle Metriken aus den Daten gewonnen werden, die eine höhere Aussagekraft haben, da sie unter realistischeren Bedingungen gesammelt wurden, als bei Eye-Tracking-Aufnahmen mit einem Bildschirm.

1.3 Ergebnisse der Arbeit

Das Resultat der Arbeit ist ein Prototyp, der Eye-Tracking-Aufnahmen an zwei Bildschirmen gleichzeitig ermöglicht. Während der Eye-Tracking-Aufnahme werden Gaze-Daten über zwei Bildschirme hinweg erfasst und es erfolgt eine parallele Bildschirmaufnahme. Durch die Weiterverarbeitung der Eye-Tracking-Daten werden diese zu einer Datei zusammengefasst und um eine Filterung nach Sakkaden und Fixationen ergänzt. Die Evaluation hat ergeben, dass die Aufzeichnung und Verarbeitung in der Regel korrekte Gaze-Daten liefert und die Erkennung des aktuell betrachteten Bildschirms zuverlässig funktioniert. Durch den gewählten Ansatz entstehen an den angrenzenden Bildschirmrändern Interferenzen, die im Rahmen des Prototypen verhindert werden, indem die Bildschirme mit einem Abstand von mindestens 15cm zueinander positioniert werden. In den meisten Fällen werden die Bildschirme in einem Multi-Monitor-Setup jedoch direkt nebeneinander positioniert. Das bedeutet, dass der entwickelte Prototyp Eye-Tracking-Aufnahmen mit zwei Bildschirmen ermöglicht. Jedoch ist die Rekonstruktion von realistischen Arbeitsbedingungen hinsichtlich der Positionierung der Monitore nicht vollständig möglich.

1.4 Struktur der Arbeit

In Kapitel 2 werden zunächst die Grundlagen erläutert, die für das Verständnis der Arbeit nötig sind. Daraufhin wird in Kapitel 3 verwandte Literatur aufgezeigt und von dieser Arbeit abgegrenzt. Die Analyse der Anforderungen an den Prototypen wird in Kapitel 4 durchgeführt und in Kapitel 5 wird das Konzept des Prototypen beschrieben. Dabei wird auf die grafische Benutzeroberfläche, die Aufnahme der Daten und die Weiterverarbeitung eingegangen, sodass sämtliche konzeptionelle Eigenschaften erklärt werden. Kapitel 6 geht daraufhin auf die verwendeten Technologien und die Architektur der Anwendung ein und in Kapitel 7 wird der Prototyp evaluiert und entsprechende Limitierungen werden aufgezeigt. Zuletzt werden in Kapitel 8 die wichtigsten Punkte der Arbeit zusammengefasst.

Kapitel 2

Grundlagen

In diesem Kapitel werden alle grundlegenden Informationen und Erklärungen aufbereitet, die für das Verständnis der Arbeit nötig sind. Zunächst werden dazu in Abschnitt 2.1 die Grundlagen des Eye-Trackings sowie die im Rahmen des Prototypen verwendete Technik erläutert. Des Weiteren wird in Abschnitt 2.2 auf die Eigenschaften des Tobii Pro X3-120 Eye-Trackers eingegangen und in Abschnitt 2.3 wird die Remote Procedure Call Technik erklärt, welche einen wichtigen Bestandteil des Prototypen darstellt.

2.1 Eye-Tracking

Seit 1990 werden Eye-Tracking-Studien im Bereich des Software-Engineerings eingesetzt [24]. Crosby et al. [8] nutzten Eye-Tracking erstmals, um zu untersuchen, wie Entwickler Algorithmen lesen und verstehen. Seitdem wurde Eye-Tracking in verschiedenen Bereichen des Software-Engineerings eingesetzt [24]. Um ein grundlegendes Verständnis über die in dieser Arbeit verwendete Eye-Tracking-Technik und die Analyse von Eye-Tracking-Daten zu erlangen, werden in Abschnitt 2.1.1 die grundlegende Begrifflichkeiten des Eye-Trackings und wichtige Augenbewegungen beschrieben und eingeordnet. Daraufhin wird in Abschnitt 2.1.2 die Nutzung der Corneareflexion im Eye-Tracking erklärt.

2.1.1 Grundlagen

Mit der Hilfe von Eye-Tracking können die Augenbewegungen von Personen gemessen werden. Anhand der gemessenen Augenbewegungen kann festgestellt werden, auf welchen Punkten die visuelle Aufmerksamkeit einer Person liegt [24]. Weiterführend können aus der visuellen Aufmerksamkeit Rückschlüsse auf die kognitiven Prozesse einer Person gezogen werden, welche interessant für die verschiedensten Forschungsgebiete sind [9].
Bei der Analyse von Eye-Tracking-Daten wertet man diese hinsichtlich der

verschiedenen Augenbewegungen aus. In der Regel werden die Augenbewegungen dabei hauptsächlich in Fixationen, Sakkaden und langsame Augenfolgebewegungen, die sogenannten Smooth Pursuits, unterteilt [9]. Während einer Fixation ruht das Auge auf einem sich nicht bewegenden Objekt [9], was das bewusste Aufnehmen von Informationen ermöglicht [24]. Jedoch befindet sich das Auge während einer Fixation nicht in völligem Stillstand, sondern es kommt zu sehr kleinen unbewussten Augenbewegungen, die charakteristisch für die Fixation sind [9]. Als Sakkaden bezeichnet man sehr schnelle Bewegungen des Auges, die zwischen zwei Fixationen auftreten [19] und somit einen Wechsel der visuellen Aufmerksamkeit zwischen zwei Punkten bedeuten. Während dieser schnellen Augenbewegungen können wir keine bewussten visuellen Informationen aufnehmen [19]. Als Smooth Pursuit bezeichnet man Augenbewegungen, bei denen das Auge ein bewegliches Ziel verfolgt und sich somit langsamer bewegt als bei Sakkaden [9, 19].
Für Eye-Tracking-Studien sind in der Regel vor allem die Fixationen von großer Bedeutung, um aus ihnen abzuleiten, welche Bereiche eines Stimulus für die Informationsverarbeitung einer Person relevant sind [12]. Ein Stimulus ist dabei ein visueller Reiz, dem das Auge ausgesetzt ist. Dieser visuelle Reiz kann ein Bild, ein Video oder eine anderweitige Darstellung auf einem Bildschirm sein. Diesem Stimulus können vorab sogenannte Areas of Interest zugeordnet werden, auf deren Basis dann die Fixationen ausgewertet werden und dadurch Rückschlüsse auf die Informationsverarbeitung der Person gezogen werden können [24].
Es existieren zwei unterschiedliche Arten Augenbewegungen zu messen. Zum einen kann die Position des Auges relativ zum Kopf gemessen werden und zum anderen kann der Blickpunkt einer Person, der sogenannte point of regard, auch Gaze-Point genannt, gemessen werden [32]. Im Rahmen von Eye-Tracking-Aufnahmen im Bereich des Software Engineerings ist man in der Regel daran interessiert herauszufinden, auf welchen Punkt eine Person schaut und nutzt daher letztere Messtechnik [24], welche heutzutage in den meisten Fällen über die Messung der Corneareflexion realisiert wird [9].

2.1.2 Eye-Tracking basierend auf Corneareflexion

Die Hornhaut des Auges, auch Cornea genannt, bildet zusammen mit der Lederhaut (Sclera) die äußere Schicht des Auges und sitzt über der Iris, wie in Abbildung 2.1 zu sehen ist [17]. Wenn das Auge nun mit einer Lichtquelle bestrahlt wird, entsteht eine Reflexion dieser Lichtquelle auf der Außenseite der Cornea [17], die bspw. durch eine Kamera erfasst werden kann. Im Rahmen dieser Arbeit und in einschlägiger Literatur wird diese Reflexion als Corneareflexion oder erstes Purkinje-Bild bezeichnet [7]. Es entstehen auch weitere Reflexionen an unterschiedlichen Punkten des Auges, die als zweites, drittes und viertes Purkinje-Bild bezeichnet werden [7]. Jedoch sind diese weiteren Relexionen weitaus schwieriger zu erfassen, da sie eine deutlich

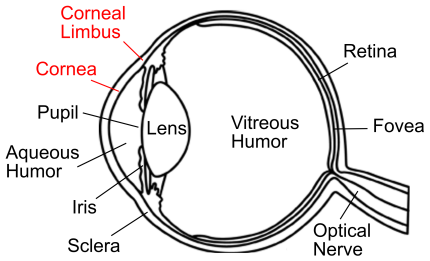


Abbildung 2.1: Queerschnitt des menschlichen Auges nach Nitschke et al. [17]

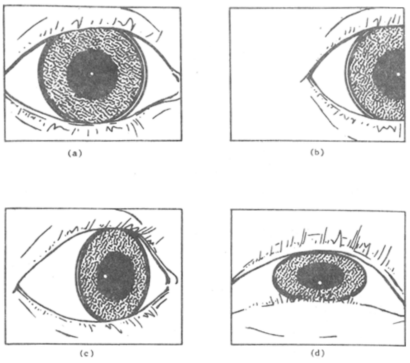


Abbildung 2.2: Corneareflexionen bei verschiedenen Augenbewegungen nach Merchant et al. [15]

geringere Intensität besitzen [32, 9].
Merchant et al. [15] setzten 1974 mit der Entwicklung des Oculometers den Grundstein für Eye-Tracker, die auf der Erfassung der Corneareflexion in Kombination mit dem Mittelpunkt der Pupille basieren [32]. Mit dieser Technik können die Rotationsbewegungen des Auges von Kopfbewegungen unterschieden werden, da sich der Abstand zwischen der Corneareflexion und dem Mittelpunkt der Pupille je nach Bewegungsart anders verhält [15]. In Abbildung 2.2 ist dies exemplarisch in vier verschiedenen Situation dargestellt. Die Corneareflexion ist in der Abbildung als weißer Punkt auf der schwarzen Pupille zu sehen. In Situation (a) und (b) schaut das Auge nach vorne, während es lediglich durch Kopfbewegungen lateral verschoben wird. In Situation (c) und (d) schaut das Auge zur Seite, was jeweils eine Verschiebung der Corneareflexion relativ zum Pupillenmittelpunkt zur Folge hat.
Das beschriebene Verhältnis von Corneareflexion und Pupillenmittelpunkt ermöglicht es, dass Eye-Tracker, die diese Technik verwenden, keine Stabilisierung oder Fixierung des Kopfes benötigen [32]. Zudem gibt es keine störenden Einflüsse durch die Lichtquelle, mit der die Corneareflexion erzeugt wird, da es sich hierbei in der Regel um Infrarotlicht handelt, welches für den Menschen nicht sichtbar ist [32].

2.2 Tobii Pro X3-120

Im Rahmen dieser Arbeit werden zwei Tobii Pro X3-120 Eye-Tracker verwendet. Das Modell X3-120 der Firma Tobii verwendet eine Technik, die auf der, in Kapitel 2.1.2 erläuterten Corneareflexion basiert, und erfasst Daten bzgl. des Gaze-Points mit einer Frequenz von 120Hz [28]. Der Eye-Tracker kann über USB 3.0 direkt mit einem Computer verbunden werden

oder es kann eine External Processing Unit (EPU) des Herstellers genutzt werden, welche über USB mit dem Eye-Tracker und per Ethernet mit dem Computer verbunden wird [28].
Der Hersteller Tobii bietet mit dem Tobii Pro Lab [27] eine Software, mit der Eye-Tracking-Aufnahmen geplant, durchgeführt und analysiert werden können. Diese Software erlaubt jedoch nur die gleichzeitige Nutzung eines Eye-Trackers. Mit dem Tobii Pro Eye Tracker Manager [26] können die Eye-Tracker kalibriert und konfiguriert werden. Zudem können Entwickler mit dem Tobii Pro SDK [29] direkt mit den Eye-Trackern des Herstellers interagieren. Das verwendete Modell X3-120 stellt dabei während der Erfassung der Gaze-Daten über das SDK verschiedene Werte je Sample bereit. Für das Verständnis dieser Arbeit sind vor allem folgende Werte von Bedeutung:

- Zeitstempel in Mikrosekunden
- Augenposition relativ zum Eye-Tracker (x, y, z)
- Augenposition innerhalb der Trackbox (x, y, z)
- Korrektheit der berechneten Augenposition
- Gaze-Point relativ zum Bildschirm (x, y)
- Korrektheit des berechneten Gaze-Points

Der Zeitstempel jedes Samples wird in Mikrosekunden seit dem Hochfahren des Computers angegeben. Alle weiteren Werte werden für beide Augen separat erfasst. Die Positionswerte relativ zum Eye-Tracker werden in einem dreidimensionalen Koordinatensystem mit dem Nullpunkt in der Mitte des Eye-Trackers angegeben, während die Positionswerte relativ zum Bildschirm in einem zweidimensionalen Koordinatensystem mit dem Punkt (0, 0) in der oberen linken Ecke des Bildschirms und dem Punkt (1, 1) in der unteren rechten Ecke des Bildschirms angegeben werden. Der Bereich, in dem der Eye-Tracker die Augen korrekt erfassen kann, wird als Trackbox bezeichnet und durch ein dreidimensionales Koordinatensystem dargestellt [30]. Die Korrektheitswerte werden als 0 oder 1 angegeben, wobei 0 für inkorrekt und 1 für korrekt steht.

2.3 Remote Procedure Call

Das Konzept des Prozeduraufrufs innerhalb eines Programms, bspw. das Aufrufen einer Methode in Python, ist ein allgemein bekanntes Konzept in den meisten Programmiersprachen. Dabei ist die grundlegende Idee der Remote Procedure Call (RPC) Technik, den bekannten Mechanismus des Prozeduraufrufs mittels Netzwerkkommunikation auf mehrere Computer

auszuweiten [4]. Ein Programm soll also die Möglichkeit haben, Prozeduren auf anderen Computern aufzurufen, ohne dass es für den Programmierer einen syntaktischen Unterschied zwischen einem RPC und einem lokalen Prozeduraufruf gibt [31].
Die zwei Parteien eines RPC werden in der Regel als Client und Server bezeichnet. Van Steen [31] beschreibt den Ablauf eines Remote Procedure Calls wie folgt: Der Client ruft die Remote Procedure auf, welche auf dem Server ausgeführt wird und ein Ergebnis an den Client zurückgibt. Für den Aufruf der Prozedur nutzt der Client einen sogenannten Client Stub. Dieser bietet dem Client eine Prozedur, in Python bspw. eine Methode, welche die benötigten Parameter akzeptiert. Anstatt die gewünschten Berechnungen nun lokal auszuführen, sendet der Client Stub die Parameter in einer Nachricht an den Server. Der Server bietet einen sogenannten Server Stub, welcher die Nachricht empfängt, die Parameter entpackt und eine lokale Prozedur mit den entpackten Parametern aufruft. Nach erfolgreicher Ausführung der server-lokalen Prozedur erhält der Server Stub das Ergebnis und sendet dieses in einer Nachricht wieder zurück an den Client Stub, welcher die Nachricht entpackt und zurückgibt.
Client und Server können dabei unterschiedliche Programmiersprachen oder Architekturen haben. Um die Kompatibilität unterschiedlicher Systeme zu gewährleisten, findet das sogenannte Parameter-Marshaling statt [31]. Hierbei werden die an die Remote Procedure übergebenen Parameter vor der Übertragung an den Server in ein plattformunabhängiges Format umgewandelt, um vom Server korrekt interpretiert werden zu können. Außerdem wird das Kommunikationsinterface oft mit Hilfe einer Interface Definition Language beschrieben, aus der Client Stubs und Server Stubs generiert werden können [31]. Es existieren verschiedene Frameworks wie Microsoft Remote Procedure Call [16] oder gRPC [11], welche die Remote Procedure Call Technik implementieren und unterschiedliche Features bieten, die teilweise über den einfachen Aufruf von Prozeduren hinausgehen.

Kapitel 3

Verwandte Arbeiten

Duchowski [9] bietet einen guten Einstieg in den Themenbereich Eye-Tracking und Young et al. schaffen mit ihrer Arbeit [32] einen guten Überblick über die verschiedenen Eye-Tracking-Methoden. Bezogen auf den Bereich des Software-Engineerings ist die Arbeit von Sharafi et al. [24] zu erwähnen, welche einen guten Eindruck über die Nutzung von Eye-Tracking im Software-Engineering vermittelt.
Im weiteren Verlauf des Kapitels werden in Abschnitt 3.1 Arbeiten aufgezeigt, die sich mit Eye-Tracking für Multi-Monitor-Setups beschäftigen. Abschnitt 3.2 geht auf den von Tobii implementierten Algorithmus zur Erkennung von Fixationen ein und Abschnitt 3.3 grenzt diese Arbeit von der bisher erschienenen Literatur ab.

3.1 Eye-Tracking für Multi-Monitor-Setups

In diesem Abschnitt werden wissenschaftliche Arbeiten vorgestellt, die sich mit der Entwicklung und Nutzung von Eye-Tracking im Rahmen von Multi-Monitor-Setups befassen. So haben Kocejko et al. [14] in ihrer Arbeit 2013 eine Methode vorgestellt, Gaze-Daten für die Interaktion mit mehreren Displays und Geräten gleichzeitig zu nutzen. Im Rahmen dieser Arbeit greifen sie auf ein zuvor von Kocejko et al. [13] entwickeltes Eye-Tracking-System zurück und erweitern dieses für die Nutzung mit mehreren Geräten. Hauptbestandteil des Systems für die Erfassung der Eye-Tracking-Daten ist hierbei eine Brille, die über zwei Kameras verfügt. Eine Kamera ist für die Erfassung des Auges zuständig und die andere Kamera wird für die Kompensation von Kopfbewegungen sowie die Erkennung des aktuell betrachteten Bildschirms genutzt. Zusätzlich sind die Bildschirme mit Infrarot-LEDs in den Ecken und LED-Markern am oberen Bildschirmrand markiert, welche zur Identifizierung des aktuell verwendeten Bildschirms genutzt werden. Die entwickelte Methode konnte in der Regel zuverlässig den jeweils betrachteten Bildschirm erkennen und den entsprechenden Gaze-

Point berechnen. Jedoch basiert das Verfahren auf den korrekt platzierte Markern und dem Tragen der Brille, was es eher unflexibel macht.
Coddington et al. [6] nutzten in ihrer Arbeit 2012 ein Dual-Monitor-Setup mit jeweils einem Mirametrix S1 Eye-Tracker unterhalb des Bildschirms für die Interaktion mit einer Image Retrieval Anwendung über Gaze-Daten. Die verwendete Eye-Tracking-Software erlaubte jedoch standardmäßig nur eine aktive Instanz mit einem aktiven Eye-Tracker gleichzeitig. Um diese Limitierung zu umgehen, wurden zwei verschiedene Benutzerkonten des Betriebssystems mit jeweils einer aktiven Instanz der Eye-Tracking-Software genutzt. Im Rahmen der Arbeit wird die nicht standardmäßige gleichzeitige Nutzung der zwei Eye-Tracker nicht explizit evaluiert. Somit lässt sich aus dem Fazit der Arbeit nur vermuten, dass die Eye-Tracker ohne weitere Anpassungen erfolgreich verwendet werden konnten.
2016 veröffentlichten Balthasar et al. [2] eine Arbeit in der sie ein Eye-Tracking-System für die Nutzung mit Dual-Monitor Setups entwickelten. Das System bestand aus zwei eigens entwickelten Eye-Trackern, die auf der in Abschnitt 2.1.2 erklärten Technik der Corneareflexion basieren und jeweils unter einem Bildschirm platziert wurden. Aufgrund der Eigenkonstruktion der Eye-Tracker war eine freie Implementierung der Eye-Tracking-Pipeline möglich. Dies ermöglichte es Balthasar et al. die mehrfach entstehenden Corneareflexionen durch die Nutzung von zwei Eye-Trackern bei der Berechnung des Gaze-Points zu unterscheiden und bestehende Algorithmen dahingehend anzupassen. Das Resultat ist ein Eye-Tracking-System, welches eine ähnliche Genauigkeit bei der Nutzung von zwei Eye-Trackern erreicht wie ein System mit nur einem Eye-Tracker.
Einen anderen Ansatz wählten Saleh et al. [21]. Sie nutzten im Rahmen ihrer Arbeit drei GazePoint GP3 Eye-Tracker, die jeweils unter einem der drei horizontal angeordneten Bildschirme platziert wurden. Der Ansatz von Saleh et al. war die Verbesserung des bestehenden Switching-Algorithmus durch die Verwendung von neuronalen Netzen. Der Switching-Algorithmus ist hierbei zuständig für das Wechseln des aktiven Eye-Trackers beim Wechsel des Gaze-Points zwischen den Bildschirmen. Der Vorteil des gewählten Ansatzes besteht darin, dass keine speziell angefertigten Eye-Tracker oder eine angepasste Berechnung des Gaze-Points nötig sind. Somit lässt sich der Ansatz mit vielen Eye-Trackern verschiedener Hersteller nutzen. Das Ergebnis der Arbeit ist eine signifikante Verbesserung des Switching-Algorithmus im Vergleich zum eingebauten Algorithmus der genutzten Eye-Tracker und potentiell auch mit den Tobii Pro X3-120 Eye-Trackern nutzbar.

3.2 Tobii I-VT Fixation Filter

Olsen beschreibt in einem von Tobii veröffentlichten Paper [18] das Konzept und die Implementierung des von Tobii verwendeten Algorithmus zur

Filterung von Fixationen. Da das Paper als Vorlage für die Implementierung des Algorithmus im Rahmen des Prototypen dient, wird in diesem Abschnitt auf das Paper eingegangen.
Olsen beschreibt den I-VT Classification Algorithmus als einen der populärsten Algorithmen zur Identifikation von Fixationen in Eye-Tracking-Daten. Dabei bietet die von Tobii implementierte Version des Algorithmus auch Informationen bzgl. Sakkaden. Grundsätzlich basiert der I-VT Algorithmus auf der Klassifizierung von Augenbewegungen anhand ihrer Geschwindigkeit. Bewegt sich das Auge bspw. entlang der horizontalen Achse des Bildschirms, dann wird für jeden Datenpunkt anhand der Geschwindigkeit der Augenbewegung entschieden, ob dieser Datenpunkt zu einer Fixation oder einer Sakkade gehört. Dazu wird ein Grenzwert, ein sogenannter Threshold, definiert. Sobald sich die Geschwindigkeit der Augenbewegung über dem Grenzwert befindet, handelt es sich nach dem Algorithmus um eine Sakkade und wenn sich die Geschwindigkeit unterhalb des Grenzwertes befindet, dann handelt es sich um eine Fixation.
Bevor die Geschwindigkeit der Augenbewegung berechnet wird, gibt es allerdings eine Reihe von Schritten, in denen die Eye-Tracking-Daten aufbereitet werden. Während Eye-Tracking-Aufnahmen kann es zu Lücken in den aufgenommenen Daten kommen. Solche Lücken können durch das Wegschauen oder Blinzeln der Person entstehen oder durch kurzzeitige Fehlfunktionen oder Verzögerungen bei der Datenübertragung. Letztere Lücken werden in einem ersten Schritt des Algorithmus zunächst aufgefüllt und anschließend werden die Daten der beiden Augen zu einem Datensatz zusammengefügt. Der daraus entstandene Datensatz enthält in der Regel noch Rauschen, welches bspw. durch Umwelteinflüsse oder kleine unbewusste Augenbewegungen, wie die Mikrobewegungen des Auges während einer Fixation, ensteht. Um dieses Rauschen zu reduzieren nutzt der Algorithmus eine Funktion zur Rauschminderung, die das hochfrequente Rauschen verringert und gleichzeitig die Merkmale der Daten, welche zur Bestimmung von Sakkaden und Fixation genutzt werden, erhält.
Nun wird die Geschwindigkeit der Augenbewegung anhand des aktuellen und vorherigen Datenpunktes berechnet und wie oben beschrieben als Fixation oder Sakkade eingestuft. Zusätzlich werden aufeinanderfolgende Datenpunkte gruppiert, wenn sie die selbe Klassifikation erhalten haben und die Position einer Fixation wird aus der durchschnittlichen Position aller Datenpunkte einer Fixationsgruppe berechnet. Nachfolgend werden Fixationen noch zusammengefügt, wenn sie durch Rauschen als zwei separate Fixationen eingestuft wurden oder Fixationen werden entfernt, wenn sie von zu kurzer Dauer sind, sodass sie nicht als Fixationen einzustufen sind.

3.3 Abgrenzung der Arbeit

Die in Abschnitt 3.1 vorgestellten Arbeiten lassen sich in zwei Kategorien einteilen. Kocejko et al. und Balthasar et al. [14, 2] nutzten im Rahmen ihrer Arbeiten eigens entwickelte oder angepasste Hardware. Dies ist ein entscheidender Unterschied zu dieser Arbeit, in der auf sogenannte off the shelf Geräte, also serienmäßig produzierte Eye-Tracker zurückgegriffen wird. Die von Balthasar et al. verwendeten Eye-Tracker nutzen jedoch ein ähnliches Verfahren zur Bestimmung des Gaze-Points wie die Tobii Pro X3-120 im Rahmen dieser Arbeit. Es lässt sich also daraus ableiten, dass es je nach Positionierung der Tobii Eye-Tracker zu den von Balthasar et al. beschriebenen Interferenzen kommen kann. Die Interferenzen entstehen durch das gleichzeitige Aussenden von Infrarotlicht, wodurch mehrfache Reflexionen auf der Cornea entstehen. Dadurch wird die Bestimmung des Gaze-Points erschwert und es kann somit zu ungenauen oder fehlenden Daten kommen. Dies gilt es im Rahmen dieser Arbeit zu beachten, zu evaluieren und gegebenenfalls zu lösen. Jedoch kann dabei nicht der gleiche Ansatz, wie von Balthasar et al. beschrieben, genutzt werden, da die Tobii Pro X3-120 keine Anpassung des Algorithmus zur Bestimmung des Gaze-Points ermöglichen.
Die zweite Kategorie von Arbeiten umfasst Coddington et al. und Saleh et al. [6, 21]. Im Rahmen dieser Arbeiten werden off the shelf Eye-Tracker ohne angepasste Hardware verwendet. Coddington et al. evaluieren jedoch die gleichzeitige Nutzung von zwei Eye-Trackern und möglicherweise entstehende Interferenzen nicht. Somit lassen sich hieraus keine nützlichen Implikationen für diese Arbeit ableiten. Der Ansatz von Saleh et al. umgeht das Problem der Interferenzen durch das Wechseln des aktiven Eye-Trackers während der Aufnahme. Dies geschieht durch die Nutzung eines Classifiers, der auf einem neuronalen Netz basiert. Dieser Ansatz könnte also auch im Rahmen des zu entwickelnden Prototypen eine mögliche Lösung für entstehende Interferenzen sein.
Insgesamt lässt sich festhalten, dass einige der genannten Arbeiten nützliche Hinweise auf mögliche Komplikationen und entsprechende Lösungsansätze bei der Arbeit mit mehreren Eye-Trackern bieten. Jedoch wurden in keiner der genannten Arbeiten Eye-Tracker der Firma Tobii in Verbindung mit dem Tobii Pro SDK verwendet. Zudem ist es das Ziel des Prototypen, eine ganzheitliche Lösung für Eye-Tracking-Aufnahmen mit zwei Bildschirmen zu bieten. Beginnend bei der Konfiguration der Eye-Tracker, Monitore und Aufnahmeeinstellungen über die Aufnahme der Eye-Tracking-Daten und des Bildschirms bis hin zur Verarbeitung der Eye-Tracking-Daten inklusive der Filterung nach Sakkaden und Fixationen. Die verwandte Literatur behandelt jedoch ausschließlich Teilbereiche der Thematik sehr intensiv und präsentiert bisher keinen Ansatz, der den gesamten Prozess abbildet. An dieser Stelle kann der entwickelte Prototyp einen Schritt in Richtung einer ganzheitlichen Anwendung für Eye-Tracking-Aufnahmen mit zwei Bildschirmen machen.

Kapitel 4

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an die entwickelte Software beschrieben. Die Anforderungen wurden in zwei Hauptschritten herausgearbeitet. Im ersten Schritt wurde die Software Tobii Pro Lab, die für Eye-Tracking-Aufnahmen mit einem Eye-Tracker verwendet wird, sowie das Tobii Pro SDK, welches im Rahmen der Anwendung zur Steuerung der Eye-Tracker benutzt werden soll, analysiert. Aus dieser Analyse ergaben sich grundlegende Anforderungen, die sich an den Funktionen des Tobii Pro Lab orientieren und gleichzeitig die Möglichkeiten und Einschränkungen des Tobii Pro SDK berücksichtigen. In einem zweiten Schritt wurden die grundlegenden Anforderungen in Besprechungen mit der Betreuerin überarbeitet und ergänzt.
In Abschnitt 4.1 werden zunächst die funktionalen Anforderungen beschrieben und in Abschnitt 4.2 werden die nichtfunktionalen Anforderungen dargestellt. Dabei orientiert sich die Formulierung der Anforderungen an den Vorgaben von Rupp und den SOPHISTen [20] und wird teilweise um eine Erläuterung unterhalb der jeweiligen Anforderung ergänzt.

4.1 Funktionale Anforderungen

R01: *Die Anwendung muss Eye-Tracking-Aufnahmen mit zwei Tobii X3-120 Eye-Trackern gleichzeitig durchführen können.*

Die Anwendung ist explizit für die gleichzeitige Verwendung mit zwei Tobii X3-120 Eye-Trackern ausgelegt. Dabei wird einer der Eye-Tracker direkt per USB mit dem Computer verbunden und der andere Eye-Tracker wird über eine External processing unit (EPU) des Herstellers Tobii mit dem Computer verbunden. Grundsätzlich ist die Verwendung einer EPU zu bevorzugen, da diese den Hauptteil der Rechenlast während der Eye-Tracking-Aufnahme übernimmt und somit die CPU des Computers entlastet. Im Rahmen dieser Arbeit stand eine EPU zur Verfügung, weshalb das oben genannte Setup verwendet wurde.

R02: *Die Anwendung muss ein Setup mit zwei nebeneinander stehenden Bildschirmen mit jeweils einem Eye-Tracker am unteren Bildschirmrand unterstützen.*

R03: *Die Anwendung muss dem Benutzer die Möglichkeit bieten, die Eye-Tracker über eine grafische Benutzeroberfläche den Bildschirmen zuzuordnen.*

R04: *Die Anwendung muss dem Benutzer die Möglichkeit bieten, Metadaten für eine Eye-Tracking-Aufnahme anzugeben.*

Die Metadaten einer Aufnahme umfassen den Namen der Aufzeichnung und den Namen der aufgezeichneten Person.

R05: *Die Anwendung muss dem Benutzer die Möglichkeit bieten, die Aufnahmen in Projekten zu organisieren.*

Der Benutzer soll die Möglichkeit haben, Projekte mit einem Projektverzeichnis anzulegen, das Projektverzeichnis zu bearbeiten und Projekte aus der Anwendung zu entfernen. Es werden alle Aufnahmen und Daten, die im Rahmen eines Projektes angefertigt werden im Projektverzeichnis gespeichert.

R06: *Die Anwendung muss dem Benutzer die Möglichkeit bieten, die Kalibrierungsfunktion des Tobii Eye Tracker Managers mit einem ausgewählten Eye-Tracker zu starten.*

R07: *Die Anwendung muss die Kalibrierungsdaten des Tobii Eye Tracker Managers verwenden.*

Wenn der Eye-Tracker bereits im Voraus kalibriert wurde, wird somit automatisch die vorhandene Kalibrierung genutzt.

R08: *Die Anwendung soll das Tobii Pro SDK für die Steuerung der Tobii Pro X3-120 Eye-Tracker nutzen.*

Das Tobii Pro SDK des Herstellers Tobii bietet Entwicklern die Möglichkeit, die Eye-Tracker des Herstellers direkt zu steuern und die aufgenommenen Daten zu verarbeiten.

R09: *Die Anwendung muss dem Benutzer die Möglichkeit bieten, Eye-Tracking-Aufnahmen über eine grafische Benutzeroberfläche zu steuern.*

Die Steuerung der Eye-Tracking-Aufnahmen umfasst das Starten und das Stoppen der Aufnahme über die grafische Benutzeroberfläche.

R10: *Während der Eye-Tracking-Aufnahme, muss die Anwendung eine synchrone Bildschirmaufzeichnung der zuvor konfigurierten*

Bildschirme vornehmen.

R11: *Während der Eye-Tracking-Aufnahmen, muss die Anwendung dem Benutzer die Möglichkeit bieten, die aktuelle Aufnahme zu überwachen.*

Der Benutzer kann im Rahmen der Konfiguration der Aufnahme ein Präsentatorfenster aktivieren, welches während der Aufnahme erscheint und über die Korrektheit der aktuell erfassten Eye-Tracking-Daten informiert und die Dauer der aktuellen Aufnahme anzeigt. Auf diese Weise können fehlerhafte Eye-Tracking-Aufnahmen durch zu große Kopfbewegungen zwischen den Bildschirmen verhindert werden.

R12: *Die Anwendung muss die aufgezeichneten Eye-Tracking-Daten der zwei Eye-Tracker zu einer Datei im tsv-Format zusammenführen.*

Aus den Rohdaten der beiden Eye-Tracker soll eine Datei angefertigt werden, die für jeden Zeitpunkt der Aufnahme nur die Eye-Tracking-Daten des jeweils betrachteten Bildschirms enthält. Die Ausgabedatei soll dabei dem Format der vom Tobii Pro Lab exportierten Dateien entsprechen und wird um eine Spalte erweitert, die den aktuell betrachteten Bildschirm identifiziert.

R13: *Die Anwendung muss die aufgezeichneten Eye-Tracking-Daten nach Sakkaden und Fixationen filtern.*

Die Ausgabedatei soll um eine Filterung nach Sakkaden und Fixationen ergänzt werden, welche auf dem in 3.2 erläuterten Algorithmus basiert, um eine Weiterverarbeitung der Daten zu vereinfachen.

4.2 Nichtfunktionale Anforderungen

NFR13: *Die Eye-Tracking-Aufnahmen der Anwendung müssen eine hohe Genauigkeit aufweisen.*

Die Eye-Tracking-Aufnahmen sollen die bestmögliche Genauigkeit aufweisen, damit sie aussagekräftig sind und im Rahmen von wissenschaftlichen Arbeiten genutzt werden können.

NFR14: *Die grafische Benutzeroberfläche muss eine gute Usability bieten.*

Die grafische Benutzeroberfläche soll einfach zu bedienen sein und dem Benutzer ausreichend Feedback bieten, um die Durchführung von Eye-Tracking-Aufnahmen bestmöglich zu unterstützen.

NFR15: *Solange eine Aufnahme stattfindet, muss die Anwendung die Benutzung des Computer ohne Beeinträchtigungen ermöglichen.*

Während der Eye-Tracking-Aufnahmen wird der Computer von Probanden für verschiedene Aufgaben genutzt. Um diese Nutzung zu gewährleisten, muss sichergestellt sein, dass die Anwendung eine angemessene Rechenleistung benötigt, die nicht zu spürbar verlängerten Ladezeiten oder verzögerten Reaktionen der Benutzeroberfläche führt.

NFR16: *Die Anwendung sollte erweiterbar sein.*

Damit die Erweiterung der Anwendung um weitere Funktionen oder um die Unterstützung anderer Eye-Tracker zu einem späteren Zeitpunkt möglich ist, sollte die Anwendung erweiterbar sein. Dies wird durch einen modularen Aufbau gewährleistet.

Kapitel 5

Konzept

Das grundlegende Konzept des Prototypen ist es, beide Eye-Tracker während der Aufnahme dauerhaft aktiv zu lassen und die aufgezeichneten Daten in einem nächsten Schritt zu verarbeiten, um eine Datei zu erzeugen, welche für die Auswertung der Daten genutzt werden kann. Zum Beginn der Arbeit wurden verschiedene Ansätze zur Nutzung und Steuerung der Eye-Tracker während der Aufnahme evaluiert. Im Rahmen der Ausschreibung wurde zunächst eine potentielle Synchronisation der Eye-Tracker vorgeschlagen, durch welche die Geräte abwechselnd Infrarotlicht aussenden, um Interferenzen vorzubeugen. Diese zeitlich exakte Synchronisation der Eye-Tracker ist jedoch mit dem Tobii Pro SDK nicht umsetzbar, da das SDK keine Option zur zeitlichen Synchronisation von zwei Eye-Trackern bietet. Zudem ist die standardmäßige Aktivierung und Deaktivierung der Eye-Tracker über das SDK mit Verzögerungen verbunden, die eine Synchronisation im Millisekundenbereich verhindert. Des Weiteren wurde eine dynamische Aktivierung der Eye-Tracker in Erwägung gezogen, bei der jeweils nur der Eye-Tracker des aktuell betrachteten Bildschirms aktiviert wird. Da die Aktivierung der Eye-Tracker über das Tobii Pro SDK eine gewisse Verzögerung mit sich bringt, würde dies zu verlorenen Daten beim Wechsel zwischen den Bildschirmen führen. Somit wurde für den entwickelten Prototypen ein Ansatz gewählt, bei dem die Eye-Tracker während der Aufnahme dauerhaft aktiv sind und die aufgezeichneten Daten nach der Aufnahme entsprechend weiterverarbeitet und aufbereitet werden.
Im weiteren Verlauf des Kapitels werden zunächst in Abschnitt 5.1 die verschiedenen Elemente der grafischen Benutzeroberfläche erläutert. Daraufhin wird in Abschnitt 5.2 der Aufbau des genutzten Setups und die Aufnahme der Eye-Tracking-Daten und des Bildschirms beschrieben. Zuletzt wird in Abschnitt 5.3 die Verarbeitung der aufgezeichneten Eye-Tracking-Daten erklärt.

5.1 Grafische Benutzeroberfläche

Die grafische Benutzeroberfläche besteht aus mehreren Fenstern mit unterschiedlichen Funktionen. In den folgenden Abschnitten werden die Elemente des Hauptfensters und des Präsentatorfenster vorgestellt.

5.1.1 Hauptfenster

Beim Starten der Anwendung erscheint zunächst das Hauptfenster, welches in Abbildung 5.1 zu sehen ist. Das Fenster ist in verschiedene Bereiche gegliedert, die dem Nutzer die Konfiguration und Steuerung der Eye-Tracking-Aufnahme ermöglichen. Der Bereich *Setup* ist in der Abbildung mit einem A markiert und ermöglicht die Konfiguration des linken und rechten Bildschirms. Dazu muss der Nutzer zunächst auswählen, welche Monitore im Rahmen der Aufnahme als linker und rechter Bildschirm genutzt werden sollen. Um diese Auswahl zu erleichtern, können die Monitore über den Button *Bildschirme identifizieren* durch die Anzeige einer Zahl in der oberen linken Bildschirmecke identifiziert werden. Nach der Auswahl des entsprechenden Monitors muss der Nutzer jeweils einen Eye-Tracker dem linken und rechten Bildschirm zuordnen. Um die Verbindung zum Eye-Tracker und die korrekte Zuordnung zu überprüfen, kann der Nutzer mit dem *Aktiv*-Schalter den gewählten Eye-Tracker aktivieren und ihn anhand des rötlich leuchtenden Infrarotlichts erkennen. Außerdem ist über den Button *Eye-Tracker Kalibrierung* die Kalibrierung des gewählten Eye-Trackers mit dem Tobii Eye Tracker Manager möglich.
Der Bereich *Aufnahme* ist in Abbildung 5.1 mit einem B gekennzeichnet und ermöglicht die Eingabe von Metadaten für die nächste Aufnahme. Diese Daten umfassen die Bezeichnung der Aufnahme und den Namen der aufgezeichneten Person. Des Weiteren kann der Nutzer einen Shortcut zum Beenden der Bildschirmaufnahme aktivieren und die Visualisierung der Gaze-Daten, welche in Abschnitt 7.1 näher erläutert wird, deaktivieren. Zusätzlich kann das Präsentatorfenster aktiviert und ein Monitor gewählt werden, auf dem das Fenster angezeigt werden soll. Das Präsentatorfenster wird in Abschnitt 5.1.2 näher erläutert.
Sobald die erforderlichen Eingaben getätigt wurden, kann die Aufnahme mit dem Button *Aufnahme starten* am unteren Rand des Fensters gestartet werden. Aufnahmen werden immer im Kontext eines Projekts durchgeführt und somit im Projektverzeichnis gespeichert. Daher ist am oberen Rand des Hauptfensters der Name des aktuellen Projekts zu sehen und mit dem Icon-Button neben dem Projektnamen kann das Projektfenster geöffnet werden, welches die Verwaltung von Projekten ermöglicht.

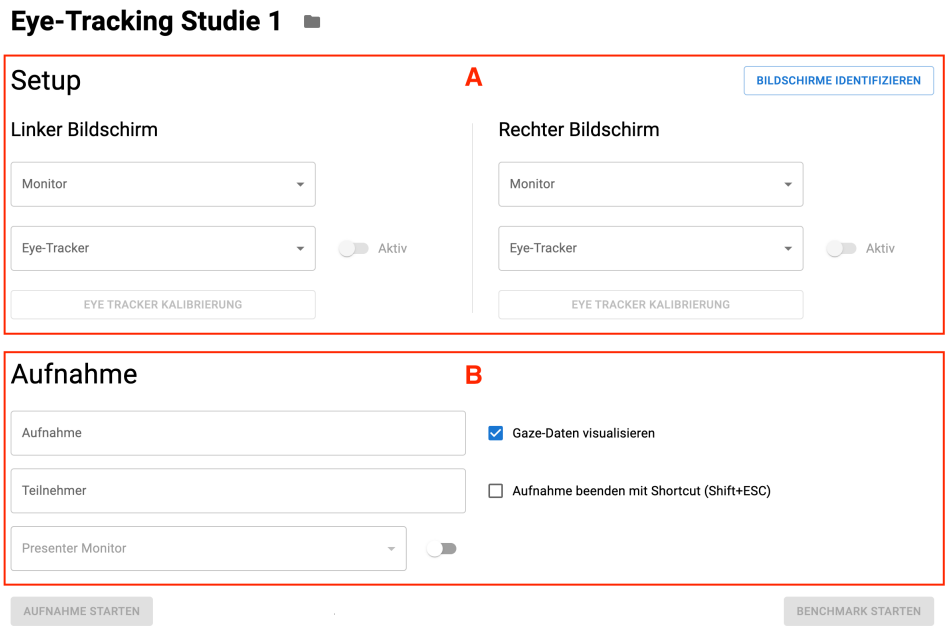


Abbildung 5.1: Hauptfenster der grafischen Benutzeroberfläche

5.1.2 Präsentatorfenster

Das Präsentatorfenster ist in Abbildung 5.2 dargestellt und öffnet sich mit dem Starten der Aufnahme, wenn es zuvor aktiviert wurde. Das Fenster kann genutzt werden, um die korrekte Erfassung von Gaze-Daten zu überwachen. Die Kopfbewegungen einer Person beim Wechsel zwischen den Bildschirmen können dazu führen, dass sich die Person nach einiger Zeit nicht mehr in der Trackbox des Eye-Trackers befindet. Wenn dies geschieht, kann der Eye-Tracker den Gaze-Point nicht mehr korrekt bestimmen und es kommt zum Verlust von Daten. Um dies zu vermeiden, gibt die Statusanzeige des Präsentatorfensters Rückmeldung über den Anteil korrekt erfasster Eye-Tracking Daten und ermöglicht somit die Korrektur der Kopfposition während der Aufnahme. Des Weiteren zeigt das Präsentatorfenster die Dauer und die zuvor konfigurierten Metadaten der aktuellen Aufnahme an sowie einen Button, der das Beenden der Aufnahme ermöglicht.

5.2 Eye-Tracking-Aufnahme

Die Erfassung und Speicherung der Gaze-Daten stellt den Kern der Anwendung dar. Im Rahmen des Prototypen werden zwei Tobii X3-120 Eye-Tracker verwendet, die jeweils am unteren Rand des Bildschirms befestigt sind, wie in Abbildung 5.3 zu sehen. Dabei ist ein Eye-Tracker über USB direkt mit

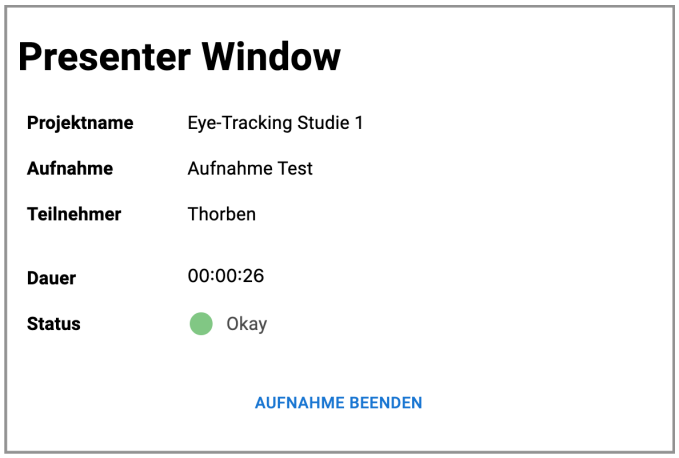


Abbildung 5.2: Präsentatorfenster der grafischen Benutzeroberfläche

dem Computer verbunden und ein Eye-Tracker nutzt eine EPU, die per Netzwerkkabel mit dem Computer verbunden ist. Beide Eye-Tracker über USB direkt mit dem Computer zu verbinden, war während der Entwicklung nicht möglich, da in diesem Fall immer nur ein Eye-Tracker erkannt wurde. Die beiden Bildschirme werden nebeneinander angeordnet, sodass sie sich ungefähr in einem 135° Winkel zueinander befinden. Dadurch kann der Blickpunkt durch das Drehen des Kopfes zwischen den Bildschirmen wechseln, während sich die Augen meist in der Trackbox eines Eye-Trackers befinden. Wenn die Bildschirme jedoch in einem 180° Winkel zueinander genutzt werden würden, dann müsste die aufgenommene Person je nach Größe der Bildschirme zusätzlich zu einer Drehung des Kopfes auch den Oberkörper bewegen, um sich beim Bildschirmwechsel in der Trackbox des jeweiligen Eye-Trackers zu positionieren. Dies könnte zu einem Verlust oder zur fehlerhaften Erfassung von Eye-Tracking-Daten führen. Des Weiteren befindet sich ein Spalt von 15-20cm zwischen den Bildschirmen, um Interferenzen zu vermeiden. Zu Beginn der Entwicklung wurden die Bildschirme ohne Abstand zueinander genutzt. Durch das Evaluieren der Aufnahmen hat sich ergeben, dass der Abstand der Bildschirme zueinander nötig ist, da es sonst in den angrenzenden Bereichen der Bildschirme zu Interferenzen bei der Erfassung der Gaze-Daten kommt. Die Interferenzen äußern sich in Gaze-Punkten, die sich willkürlich über beide Bildschirme verteilen, wenn der tatsächliche Blickpunkt in die Nähe der angrenzenden Bildschirmränder fiel. Durch einen Abstand der beiden Bildschirme von mindestens 15cm werden die meisten Interferenzen vermieden, ohne dass die Nutzbarkeit der beiden Bildschirme zu stark eingeschränkt wird. Da sich die Stärke der Interferenzen und der Abstand, ab dem es zu Interferenzen kommt, von Person zu Person unterscheiden kann, ist eine Verringerung des Abstands der Bildschirme u.U. möglich. Des Weiteren sind die Bildschirme um ca. 5° nach oben geneigt,

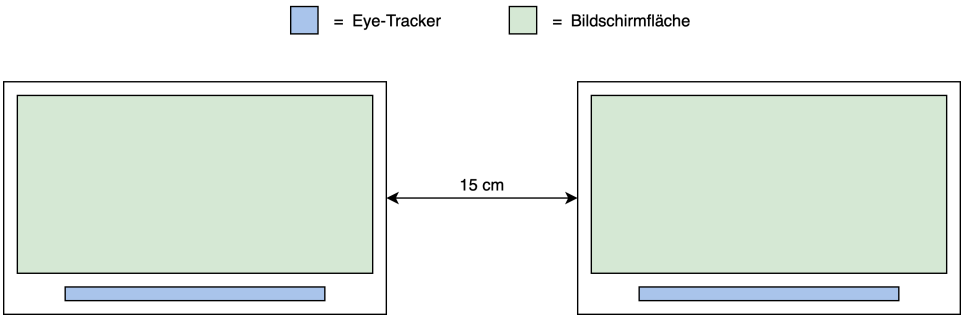


Abbildung 5.3: Bildschirm- und Eye-Tracker Setup

um die Augen in der Trackbox der Eye-Tracker zu positionieren und die aufgenommene Person sitzt mittig vor dem Setup.
Bevor mit den Aufnahmen begonnen werden kann, muss zunächst einmalig für jede Kombination aus Bildschirm und Eye-Tracker ein Bildschirmsetup im Tobii Eye Tracker Manager konfiguriert werden. Vor einer Aufnahme müssen die Eye-Tracker einzeln mit dem Tobii Eye Tracker Manager kalibriert und über die grafische Benutzeroberfläche des Prototypen den jeweiligen Bildschirmen zugeordnet werden. Mit dem Starten der Aufnahme über die grafische Benutzeroberfläche werden beide Eye-Tracker aktiviert und beginnen mit der Aufzeichnung der Gaze-Daten. Die Steuerung der Eye-Tracker erfolgt dabei über das Tobii Pro SDK und die erfassten Gaze-Daten werden während der Aufnahme in einem Buffer gespeichert. Die Daten werden von den Eye-Trackern mit jeweils 120 Hz aufgezeichnet und umfassen die in Abschnitt 2.2 erläuterten Werte je Sample. Sobald die Aufnahme beendet wird, werden die Eye-Tracker über das Tobii Pro SDK deaktiviert und die in den Buffern zwischengespeicherten Gaze-Daten als tsv-Datei im Aufnahmeverzeichnis abgespeichert. Dabei werden die Daten jedes Eye-Trackers in einer separaten Datei gespeichert. Ein Ausschnitt aus einer Ausgabedatei ist exemplarisch in Anhang A.1 dargestellt.
Um die Gaze-Daten der Eye-Tracker im Nachhinein interpretieren zu können, erfolgt eine parallele Bildschirmaufnahme. Bei dieser werden die Bildschirminhalte mit bis zu 30 Bildern pro Sekunde aufgezeichnet, um sowohl eine flüssige Darstellung zu gewährleisten als auch Prozessor und Arbeitsspeicher nicht unverhältnismäßig stark zu belasten. Die tatsächliche Framerate variiert während der Aufzeichnung leicht, je nach Auslastung des Computers. Am Ende der Aufnahme wird eine Videodatei im Aufnahmeverzeichnis erstellt, welche die Aufnahme der beiden zuvor konfigurierten Bildschirme beinhaltet. Um anschließend die Synchronisation der Bildschirmaufnahme mit den Gaze-Daten der Eye-Tracker zu ermöglichen, wird während der Aufnahme zusätzlich eine Textdatei erstellt. Diese Textdatei enthält für jeden Frame der Bildschirmaufnahme den Aufnahmezeitpunkt in Millisekunden und wird ebenfalls im Aufnahmeverzeichnis abgespeichert.

5.3 Verarbeitung der Eye-Tracking-Daten

Nach der Aufnahme werden die Eye-Tracking-Daten in weiteren Schritten aufbereitet und weiterverarbeitet, um sie anschließend besser auswerten zu können. Daher wird in den folgenden Abschnitten zuerst auf das Zusammenführen der Eye-Tracking-Daten und anschließend auf die Filterung nach Sakkaden und Fixationen eingegangen.

5.3.1 Zusammenführung der Daten

Nach der Aufnahme liegen die Gaze-Daten der beiden Eye-Tracker in zwei separaten Dateien vor. Da sich der Gaze-Point einer Person zu jedem Zeitpunkt jedoch immer nur auf einem Bildschirm befinden kann, werden die Gaze-Daten in einem nächsten Schritt zu einer Datei zusammengeführt. Dies ermöglicht im Anschluss eine einfachere Analyse und Interpretation der Daten.
Der Ablauf des Algorithmus ist in Abbildung 5.4 dargestellt. Zunächst werden die Gaze-Daten der beiden Eye-Tracker eingelesen. Da die eingelesenen Daten bereits nach dem Zeitstempel sortiert sind, wird in einem ersten Schritt der Eye-Tracker identifiziert, der zeitlich das erste korrekte Sample erfasst hat. Die Korrektheit des Samples wird in diesem Schritt anhand der Validity Codes bestimmt. Die Samples des auf diese Weise identifizierten Eye-Trackers werden dann bis einschließlich des ersten korrekten Samples für den Anfang der Ausgabedatei übernommen. Dabei werden die Samples bei der Zusammenführung der Daten noch um ein neues Feld mit der Bezeichnung *Side* ergänzt, welches das Sample dem linken oder rechten Bildschirm zuordnet.
Nachdem die ersten Samples in die Ausgabedatei übernommen wurden, werden die Daten der beiden Eye-Tracker nun Sample für Sample verarbeitet. Da beide Eye-Tracker Gaze-Daten mit einer Frequenz von 120 Hz aufzeichnen, folgen zeitlich meist zwei Samples aufeinander, die von zwei unterschiedlichen Eye-Trackern stammen. In diesem Fall werden die aufeinanderfolgenden Samples miteinander verglichen, indem ein Score für jedes Sample berechnet wird. Wenn mindestens eines der Samples einen Score erreicht, der oberhalb des Thresholds liegt, wird das Sample mit dem höheren Score in die Ausgabe übernommen und das Sample mit dem niedrigeren Score wird verworfen. Der Score eines Samples berechnet sich aus verschiedenen Werten. Zum einen werden die Validity Codes der Samples geprüft und zum anderen wird geprüft, ob die Koordinaten des Gaze-Points innerhalb der Fläche des Bildschirms liegen. Des Weiteren werden auch die letzten 15 Samples der Ausgabedatei betrachtet. Je mehr Samples zuletzt einem Bildschirm zugeordnet wurden, desto höher ist der Score für das aktuelle Sample des selben Bildschirms.
Bei der Aufzeichnung der Gaze-Daten erfassen beide Eye-Tracker oft gleich-

zeitig potenzielle Gaze-Punkte. Um zu erkennen, welcher Bildschirm zu einem Zeitpunkt tatsächlich betrachtet wurde, hilft es zu prüfen, ob bei einem Sample für beide Augen jeweils ein Gaze-Point erfasst wurde. Ist dies der Fall, dann ist das ein sehr guter Indikator dafür, dass der zugehörige Bildschirm des Eye-Trackers betrachtet wurde. Durch die Wahl eines verhältnismäßig hohen Thresholds, werden Samples, bei denen diese Entscheidung nicht eindeutig getroffen werden kann, zunächst verworfen. Auf diese Weise wird der aktuell betrachtete Bildschirm sehr effektiv ermittelt und ein Großteil der fälschlicherweise ermittelten Gaze-Points entfernt.
In einem nächsten Schritt werden die zusammengeführten Samples um jene Daten ergänzt, die aufgrund des hohen Thresholds herausgefiltert worden sind, obwohl sie für die weitere Analyse nützlich sind. Dazu werden jeweils zwei aufeinanderfolgende Samples betrachtet. Wenn die betrachteten Samples von dem selben Bildschirm stammen, dann wird in den ursprünglichen Daten des Eye-Trackers nach Samples gesucht, die zwischen den zwei aktuell betrachteten Samples aufgenommen wurden. Wenn entsprechende Samples gefunden worden sind, dann werden diese zwischen den betrachteten Samples eingefügt. Wenn die betrachteten Samples von zwei unterschiedlichen Bildschirmen stammen, werden keine Samples hinzugefügt, da die Erkennung des aktuell betrachteten Bildschirms für Samples in diesem Bereich nicht eindeutig möglich ist.
Die zusammengeführten Samples werden in einem letzten Schritt noch um zwei Felder ergänzt. Zum einen wird zu jedem Sample das Feld *Computer timestamp (Epoch Unix)* mit dem Zeitstempel der Aufnahme in Unix Zeit hinzugefügt und zum anderen erhält jedes Sample das Feld *Recording timestamp*, welches den Zeitpunkt der Aufnahme bezogen auf den ersten Frame der Bildschirmaufnahme enthält. Durch diese Felder können die Gaze-Daten in Kombination mit der Bildschirmaufnahme leichter ausgewertet werden.

5.3.2 Filterung nach Sakkaden und Fixationen

Die Filterung nach Sakkaden und Fixationen ist der letzte Schritt in der Verarbeitung der Eye-Tracking-Daten. Das Konzept des Algorithmus entspricht zum Großteil dem Konzept des Tobii I-VT Fixation Filters 3.2. Der Kern der Filterung besteht darin, die Geschwindigkeit der Augenbewegung zu berechnen. Dies geschieht über die Bestimmung des Winkels $\alpha_{t_1 t_2}$ in dem sich aufspannenden Dreieck zwischen der Augenposition und zwei aufeinanderfolgenden Gaze-Points S_{t_1} und S_{t_2}, wie in Abbildung 5.5 zu sehen. Anhand der Geschwindigkeit der Augenbewegung, wird dann entschieden, ob es sich um eine Fixation oder um eine Sakkade handelt. Als Threshold nutzt der Prototyp den von Olsen [18] empfohlenen Wert von $30°/sec.$
Vor der Berechnung der Geschwindigkeit, werden die Daten, wie von

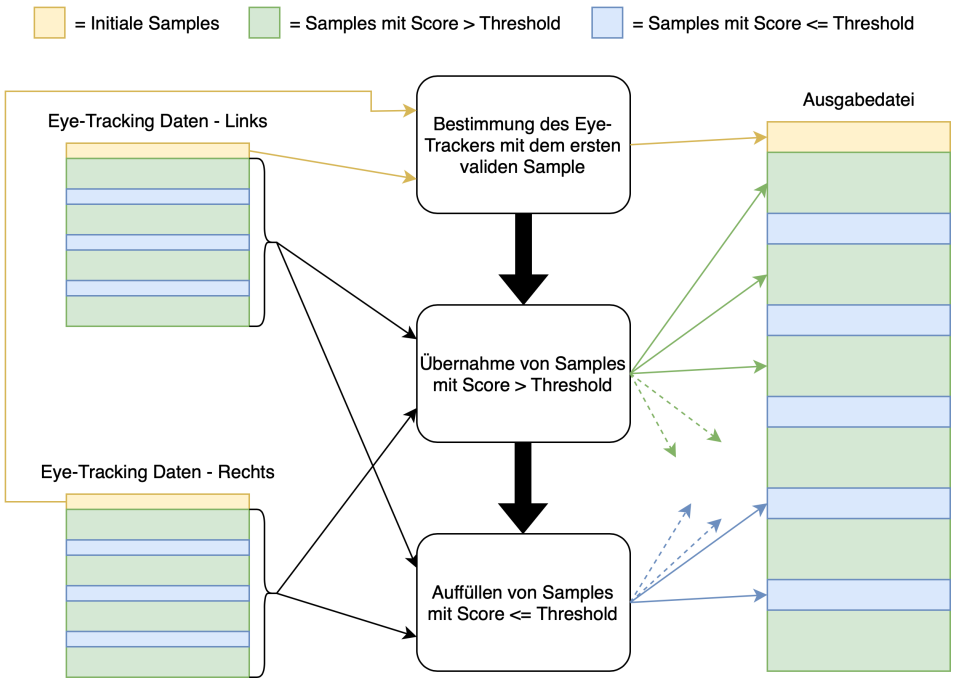


Abbildung 5.4: Ablauf des Algorithmus zur Zusammenführung der Daten

Olsen [18] beschrieben, aufbereitet, um die korrekte Klassifizierung der Augenbewegungen zu optimieren. In einem ersten Schritt werden kleine Lücken in den aufgezeichneten Daten aufgefüllt, wenn diese kürzer als 75ms andauern. Bei solch kleinen Lücken innerhalb der Daten handelt es sich höchstwahrscheinlich um kleine technische Fehler in der Aufnahme, die nicht durch ein bewusstes Wegschauen entstanden sind [18]. Die Werte, mit denen aufgefüllt wird, errechnen sich durch lineare Interpolation zwischen dem letzten validen Sample vor der Lücke und dem ersten validen Sample nach der Lücke. Danach wird in einem weiteren Schritt das Rauschen reduziert. Dazu werden für jedes Sample ein durchschnittlicher Gaze-Point und eine durchschnittliche Augenposition innerhalb eines symmetrischen Fensters berechnet. Das betrachtete Fenster umfasst dabei maximal drei Samples. Das heißt, es werden normalerweise neben dem aktuellen Sample auch das vorherige und das nachfolgende Sample betrachtet. Wenn es sich bei einem der umliegenden Samples um eine Lücke in den Daten handelt, dann schrumpft dass Fenster symmetrisch, sodass nur Samples betrachtet werden, für die ein Gaze-Point erfasst wurde.
Olsen beschreibt mit dem Verschmelzen von Fixationen und dem Verwerfen von zu kurzen Fixationen zwei weitere Verarbeitungsschritte, die nach der Klassifizierung verwendet werden können, um diese zu optimieren. Im Rahmen der Entwicklung des Prototypen haben diese Schritte nicht zu einer erheblichen Verbesserung der Klassifizierung geführt. Daher wurden diese

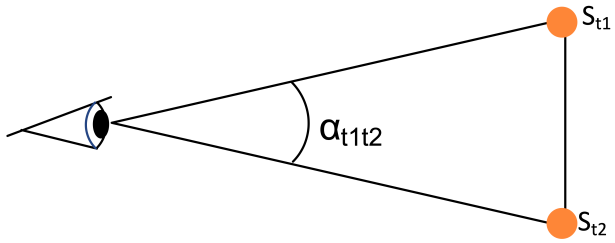


Abbildung 5.5: Berechnung des Gaze-Winkels zwischen zwei Gaze-Points nach Olsen [18]

zwei Verarbeitungsschritte nicht implementiert.
Die finale Ausgabedatei umfasst mehrere Felder, in denen die Ergebnisse der Filterung vermerkt sind. Die Klassifizierung, ob es sich um eine Fixation oder eine Sakkade handelt, ist im Feld *Eye movement* vermerkt und die Dauer der Augenbewegung im Feld *Gaze event duration*. Wenn es sich bei der Augenbewegung um eine Fixation handelt, dann ist der durchschnittliche Fixationspunkt in Bildschirmpixeln in den Feldern *Fixation point X* und *Fixation point Y* angegeben. Ein Ausschnitt aus einer finalen Ausgabedatei ist exemplarisch in Anhang A.2 dargestellt.

Kapitel 6

Implementierung

In diesem Kapitel wird die Implementierung des Prototypen beschrieben. Dazu wird zunächst in Abschnitt 6.1 auf die verwendeten Technologien eingegangen und daraufhin in Abschnitt 6.2 die Architektur der Anwendung vorgestellt.

6.1 Verwendete Technologien

Die Technologieentscheidungen im Rahmen des Prototypen entstanden zu unterschiedlichen Zeitpunkten während der Entwicklung. Zu Beginn stellte sich schnell heraus, dass der Prototyp im Kern Python verwenden wird, da das Tobii Pro SDK als Python-Package verfügbar ist. Um die Komplexität der Architektur möglichst gering zu halten, war die ursprüngliche Idee die grafische Benutzeroberfläche mit tkinter umzusetzen. Dabei handelt es sich um ein Package der Python Standardbibliothek, mit dem grafische Benutzeroberflächen erstellt werden können. Jedoch ist die Erstellung moderner Benutzeroberflächen, die dem Benutzer angemessenes Feedback geben, mit tkinter erheblich komplizierter als mit Web-Technologien. Daher fiel die Entscheidung, die grafische Benutzeroberfläche mit Electron umzusetzen, da dieses Framework die Erstellung von Desktop-Anwendungen mit Web-Technologien ermöglicht. Um die Kommunikation zwischen der Electron-Anwendung und der Python-Engine zu ermöglichen, wird gRPC verwendet, da es für die gewählten Technologien entsprechende Packages gibt und das Framework neben simplen Prozeduraufrufen auch das Streamen von Daten ermöglicht.

6.1.1 Programmiersprachen und Frameworks

In diesem Abschnitt werden die verwendeten Programmiersprachen und Frameworks aufgelistet und um eine kurze Beschreibung ergänzt.

1. **Python**: Python ist eine interpretierte Programmiersprache, die auf

allen gängigen Plattformen verfügbar ist. Die Anwendung verwendet Python in der Version 3.10.10, da das Tobii Pro SDK zum Zeitpunkt der Entwicklung maximal Python 3.10 unterstützt. Python wird zur Einbindung des Tobii Pro SDKs und zur Verarbeitung der Eye-Tracking-Daten genutzt.

2. **Electron**: Electron ist ein Framework zur Erstellung von plattformübergreifenden Desktop-Anwendungen mit Web-Technologien. Dabei verbindet Electron Chromium für die grafische Benutzeroberfläche und Node.js für die Business-Logic. Die Anwendung verwendet Electron 23.2.0, welches die aktuelle Version zum Beginn der Entwicklung war.

3. **React**: React ist eine JavaScript-Bibliothek zur Erstellung von Webanwendungen. React ermöglicht die deklarative Programmierung von Benutzeroberflächen und nutzt dazu die Programmiersprache JavaScript. Die Anwendung nutzt React 18.2.0, welches die aktuelle Version zum Beginn der Entwicklung war.

3. **Javascript**: Javascript ist eine weit verbreitete Skriptsprache, die es ermöglicht interaktive Webinhalte zu erstellen und Anwendungen mit der Hilfe von Laufzeitumgebungen wie Node.js zu entwickeln. Im Rahmen des Prototypen wird Javascript in Kombination mit React innerhalb von Electron genutzt.

4. **gRPC**: gRPC ist ein Remote Procedure Call Framework, das verschiedene Programmiersprachen unterstützt und die Defintion von Interfaces mit Hilfe von Protocol Buffers ermöglicht. Das Framework wird zur Kommunikation zwischen Python und Electron genutzt.

5. **FFmpeg**: FFmpeg ist eine Sammlung von Tools zum Aufnehmen und Verarbeiten von Video- und Audiodateien. Im Rahmen des Prototypen wird FFmpeg für die Bildschirmaufzeichnung verwendet.

6.1.2 Packages und Bibliotheken

Sowohl Javascript, Electron und React als auch Python bieten eine Vielzahl von Bibliotheken und Packages, welche für die Entwicklung von Anwendungen genutzt werden. In diesem Abschnitt werden die verwendeten Packages und Bibliotheken aufgelistet und beschrieben.

1. **Tobii Pro SDK**: Das Tobii Pro SDK wurde bereits in Abschnitt 2.2 näher erläutert. Die Anwendung nutzt die Version 1.11.0 des SDKs, welche zum Zeitpunkt der Entwicklung die aktuelle Version ist.

2. **OpenCV**: OpenCV ist eine Bibliothek für Computer Vision, Machine Learning und Bildverarbeitung. Im Rahmen dieses Prototypen wird das Python Package von OpenCV in der Version 4.7.0.72 verwendet,

um die Gaze-Daten und Augenbwegungen in der Bildschirmaufnahme zu visualisieren.

3. **Material UI**: Material UI ist eine UI-Bibliothek mit React-Komponenten, die Googles Material Design implementieren. Die Bibliothek wird für die grafische Benutzeroberfläche verwendet und ermöglicht somit die effiziente Implementierung einer ansprechenden Benutzeroberfläche.

4. **Moment.js**: Moment.js ist ein Package für Javascript, welches die Arbeit mit Daten und Uhrzeiten vereinfacht.

5. **uuid**: uuid ist ein oft verwendetes Package für Javascript, welches das Erzeugen von zufälligen IDs ermöglicht.

6. **electron-store**: electron-store ist ein Package für Electron, welches das einfache persistieren von Daten zwischen Programmstarts ermöglicht. Im Rahmen des Prototypen wird das Package benutzt, um die Liste von erstellten Projekten zu persistieren.

6.2 Architektur

Die Anwendung setzt sich aus zwei Bestandteilen zusammen: Der Engine und dem Frontend. Da Engine und Frontend unterschiedliche Programmiersprachen nutzen und somit auch in unterschiedlichen Prozessen ausgeführt werden, wird gRPC zur Kommunikation untereinander genutzt. Das gRPC Framework nutzt Protocol Buffers für die Definition eines gemeinsamen Interfaces, welches im Rahmen des Prototypen folgende Prozeduraufrufe definiert:

- **calibrateEyeTrackerManager**: Die Prozedur startet die Kalibrierung des Eye-Trackers, dessen ID übergeben wurde, im Tobii Eye Tracker Manager.

- **getDevices**: Die Prozedur gibt die aktuell angeschlossenen Eye-Tracker zurück.

- **activateTracker**: Die Prozedur aktiviert den Eye-Tracker, dessen ID übergeben wurde und gibt einen Boolean zurück, der die erfolgreiche Aktivierung bestätigt. Dies kann genutzt werden, um den Eye-Tracker zu identifizieren.

- **deactivateTracker**: Die Prozedur deaktiviert den Eye-Tracker, dessen ID übergeben wurde und gibt ebenfalls einen Boolean zurück, der die erfogreiche Deaktivierung bestätigt.

- **startTracking**: Die Prozedur erwartet mehrere Parameter, welche zur Konfiguration der Aufnahme benötigt werden und startet die Aufnahme der Gaze-Daten durch die konfigurierten Eye-Tracker sowie die Aufzeichnung der konfigurierten Bildschirme. Am Ende des Prozeduraufrufs wird ein Boolean zurückgegeben, der den erfolgreichen Start der Aufnahme bestätigt.

- **trackingStream**: Die Prozedur gibt nach dem Aufruf einen Stream zurück, der Informationen über die Qualität der aktuellen Eye-Tracking-Aufnahme überträgt, bis die Aufnahme beendet wird.

- **stopTracking**: Die Prozedur stoppt die Aufnahme der Gaze-Daten sowie die Aufzeichnung der Bildschirme und führt die in Abschnitt 5.3 erläuterte Verarbeitung der Gaze-Daten durch. Im Anschluss gibt die Prozedur einen Boolean zurück, der den erfolgreichen Aufnahmestopp und die erfolgreiche Verarbeitung der Daten bestätigt.

Wie in Abbildung 6.1 dargestellt, bietet die Engine einen Server, der das beschriebene Interface implementiert, während das Frontend einen Client implementiert, der die definierten Prozeduren aufruft. Auf diese Weise kommunizieren Frontend und Engine mit der Hilfe von gRPC über TCP Port 50051 miteinander.
Die Hauptfunktionalität der Engine ist dabei im Core-Modul implementiert. Das Core-Modul umfasst Klassen, welche die Steuerung der Eye-Tracker, den Import und Export von Daten, die Bildschirmaufnahme und die Verarbeitung der Gaze-Daten implementieren. Sämtliche Klassen der Engine sind im Klassendiagramm im Anhang B dargestellt.
Die Electron Anwendung bildet das Frontend des Prototypen. Electron startet initial immer den sogenannten Main-Prozess, der dazu genutzt wird, die verschiedenen Fenster in eigenen Renderer-Prozessen zu öffnen und zu schließen. Zudem kommuniziert der Main-Prozess mit dem gRPC Client des Frontends und mit den einzelnen Renderer-Prozessen der Fenster, um Nachrichten zu übertragen und Prozeduren aufzurufen. Die Renderer-Prozesse starten wiederum jeweils eine eigene React-Anwendung, die dem Benutzer ein interaktives Interface bietet.

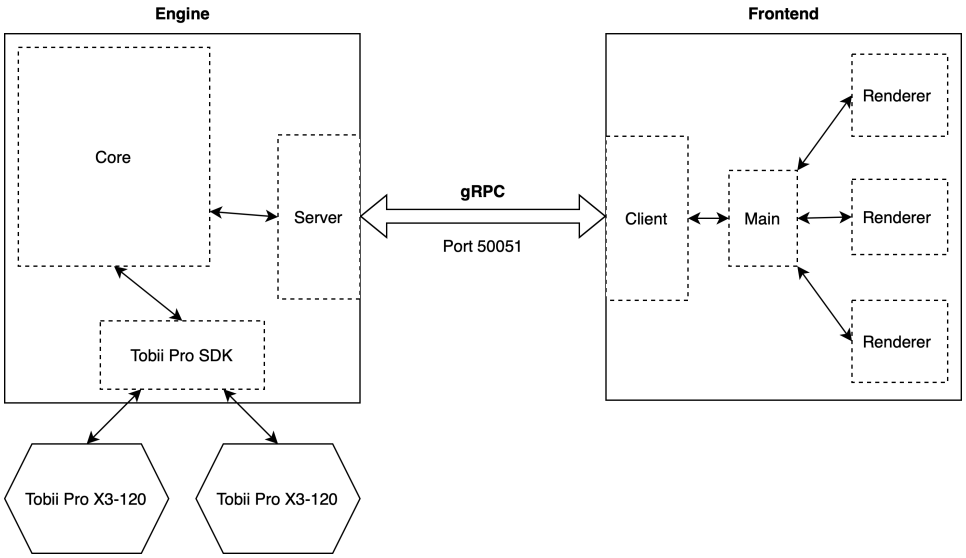


Abbildung 6.1: Architekturdiagramm

Kapitel 7

Evaluation

Dieses Kapitel beinhaltet die Evaluation des entwickelten Prototypen. Dazu wird in Abschnitt 7.1 darauf eingegangen, wie die korrekte Funktionsweise des gewählten Ansatzes evaluiert wurde und in Abschnitt 7.2 werden die Limitierungen des Prototypen dargestellt.

7.1 Proof of Concept

Die Korrektheit der erfassten Daten und somit auch des gewählten Konzepts im Rahmen des Prototypen, wurde während der Entwicklung zunächst anhand der aufgezeichneten Daten evaluiert. Da die ad-hoc Analyse der tsv-Dateien, welche die Eye-Tracking-Daten enthalten, aufgrund ihrer Abstraktheit nur sehr eingeschränkt möglich ist, wurde eine Visualisierung der Gaze-Daten entwickelt. Mit Hilfe der Visualisierung wurde zudem nicht nur die korrekte Erfassung sondern auch die Weiterverarbeitung der Gaze-Daten evaluiert. Bei der Visualisierung der Gaze-Daten werden die Gaze-Points als farbiger Kreis in der Bildschirmaufnahme dargestellt. Die Farbe unterscheidet sich je nachdem, ob der linke oder rechte Eye-Tracker den Gaze-Point erfasst hat. Auf diese Weise kann die Korrektheit der erfassten Gaze-Daten geprüft werden. Zudem werden auch die Mittelpunkte einer Fixation durch die Visualisierung abgebildet. Dies geschieht durch einen transparenten farbigen Kreis und ermöglicht die Überprüfung der Korrektheit des Fixationspunktes. Mit Hilfe der Visualisierung konnte während der Entwicklung die grundlegende Korrektheit des Ansatzes und der Implementierung überprüft werden.
Um das Konzept des Prototypen abschließend zu evaluieren, wurde ein Stimulus entwickelt, der einen Punkt über die beiden Bildschirme hinweg animiert. Fünf Personen erhielten daraufhin die Aufgabe dem Punkt zu folgen und ihn im Stillstand zu fixieren. Dabei trugen zwei der fünf Personen eine Brille, was die Erfassung der Gaze-Daten durch die Eye-Tracker zusätzlich erschwert. Das Setup entsprach dabei den Bedingungen,

wie in Abschnitt 5.2 beschrieben.
Die abschließende Evaluation der visualisierten Gaze-Daten ergab, dass der finale Prototyp Eye-Tracking-Daten erfasst, die dem tatsächlichen Gaze-Point entsprechen. Dazu wurden zunächst Punkte festgelegt, die in einer bestimmtem Reihenfolge betrachtet werden sollten oder es wurde der zuvor beschriebene Stimulus genutzt. Nach der Aufnahme wurden die visualisierten Gaze-Points mit den zuvor festgelegten Punkten verglichen. Aus diesem Vergleich ließ sich ebenfalls erkennen, dass ein Wechsel des Gaze-Points zwischen den Bildschirmen erkannt und der entsprechende Eye-Tracker während der Verarbeitung der Daten korrekt ausgewählt wurde. Des Weiteren funktionierte die Synchronisation zwischen den Gaze-Daten und der Bildschirmaufnahme anhand der Zeitstempel bei den gesammelten Daten korrekt und die Erkennung von Fixationen sowie die Berechnung der Fixationspunkte war ebenfalls valide.
Die Evaluation der visualisierten Daten während der Entwicklung führte zu dem in Abschnitt 5.2 beschriebenen Spalt zwischen den beiden Bildschirmen. Wenn die Bildschirme ohne Abstand zueinander genutzt wurden, zeigte die Visualisierung der Gaze-Daten deutliche Interferenzen. Diese Interferenzen äußerten sich in Gaze-Punkten, die sich über beide Bildschirme willkürlich verteilten, wenn der tatsächliche Blickpunkt in die Nähe des angrenzenden Bildschirmrands fiel. Somit war eine korrekte Erfassung des Gaze-Points nicht mehr gegeben und es kam zu einem erheblichen Verlust von Daten. Um dieser Problematik entgegenzuwirken, ergab sich die Umpositionierung der Bildschirme als wirksame Lösung, welche die beschriebene willkürliche Erfassung falscher Gaze-Points verhindert. Bei der abschließenden Evaluation zeigte sich das Problem der Interferenzen an den angrenzenden Bildschirmrändern bei einer Person mit Brille in abgeschwächter Form erneut. Dies deutet darauf hin, dass der Prototyp Gaze-Daten in der Regel korrekt erfasst und verarbeitet, es jedoch beim Tragen einer Brille zu verstärkten Interferenzen kommen kann, die vorab geprüft werden sollten. Dazu kann bspw. der zur Evaluation entwickelte Stimulus genutzt werden.

7.2 Limitierungen

Im Rahmen des Prototypen gibt es verschiedene Limitierungen, die zum einen konzeptionell bedingt sind und zum anderen von der Implementierung abhängen. Eine entscheidende konzeptionelle Limitierung des Prototypen ist die Positionierung der Bildschirme. Aufgrund von auftretenden Interferenzen müssen die Bildschirme zum einen in einem 135° Grad Winkel und mit einem Abstand von mindestens 15cm zueinander positioniert werden, um die korrekte Erfassung der Gaze-Points zu gewährleisten. Dies schränkt die Nutzbarkeit des Prototypen in Umgebungen, in denen das Setup nicht entsprechend angepasst werden kann, stark ein.

Grundsätzlich ist der Prototyp darauf ausgelegt, mit zwei Eye-Trackern der Firma Tobii genutzt zu werden. Das heißt, dass eine Nutzung mit nur einem Eye-Tracker oder mit mehr als zwei Eye-Trackern zum jetzigen Stand aufgrund der Implementierung nicht möglich ist. Zudem werden auch keine Eye-Tracker eines anderen Herstellers unterstützt. Die Nutzung von nur einem Eye-Tracker oder von mehr als zwei Eye-Trackern kann theoretisch implementiert werden. Besonders die praktische Umsetzung aufgrund der Entstehung von Interferenzen bei zusätzlichen Eye-Trackern müsste jedoch evaluiert werden. Des Weiteren ist die Unterstützung von Eye-Trackern anderer Hersteller, sofern diese die Steuerung über ein entsprechendes Python-Package erlauben, ebenfalls möglich.
Bei der Entwicklung des Algorithmus zur Zusammenführung der Gaze-Daten, wurden Werte und Thresholds zur Berechnung gewählt, die anhand verschiedener Testdaten gute Resultate liefern. Die Anpassung dieser Werte durch Änderungen in einer einfach zu editierenden Konfigurationsdatei oder in der grafischen Benutzeroberfläche sind in der finalen Version des Prototypen nicht möglich. Dies gilt ebenso für Anpassungen der Window Size, des Velocity Thresholds und anderer Werte, die in der Filterung nach Sakkaden und Fixationen genutzt werden. Sämtliche Werte können jedoch im Programmcode geändert werden. Eine Möglichkeit zur Anpassung dieser Werte ohne Änderung im Programmcode würde die Anwendung flexibler machen und potentiell eine einfachere Anpassung an andere Gegebenheiten und Geräte ermöglichen.
Eine weitere implementierungsabhängige Limitierung ist die Visualisierung der Gaze-Daten in der Bildschirmaufnahme. Diese findet automatisiert nach dem Beenden der Aufnahme statt und erlaubt währenddessen keine Interaktion mit der Anwendung. Die Visualisierung könnte bspw. in einem separaten Thread durchgeführt werden, um weitere Aufnahmen im direkten Anschluss zu ermöglichen, ohne auf die Visualisierung verzichten zu müssen.

Kapitel 8

Zusammenfassung und Ausblick

In diesem Kapitel werden in Abschnitt 8.1 die wichtigsten Punkte der Arbeit zusammengefasst und in Abschnitt 8.2 wird ein Ausblick auf ungelöste Probleme des Prototypen sowie Erweiterungsmöglichkeiten gegeben.

8.1 Zusammenfassung

Die Nutzung von zwei Bildschirmen im Rahmen von Eye-Tracking-Aufnahmen ermöglicht die Abbildung realistischer Arbeitsbedingungen, z.B. bei Eye-Tracking-Studien im Software-Engineering. Viele Eye-Tracker erlauben jedoch nur die gleichzeitige Nutzung eines Eye-Trackers für Eye-Tracking-Aufnahmen. Der in dieser Arbeit entwickelte Prototyp löst dieses Problem, indem er Eye-Tracking-Aufnahmen mit zwei Tobii X3-120 Eye-Trackern ermöglicht.
Durch die Nutzung von zwei Tobii Eye-Trackern kann der Prototyp Eye-Tracking-Daten von zwei Bildschirmen gleichzeitig erfassen. Dazu nutzt die Anwendung das Tobii Pro SDK, um beide Eye-Tracker zu steuern und erstellt gleichzeitig eine Aufnahme der genutzten Bildschirme. Mit einem eigens entwickelten Algorithmus zur Zusammenführung der Eye-Tracking-Daten wird eine einzige Ausgabedatei erzeugt, welche die Eye-Tracking-Daten der beiden Bildschirme enthält. Des Weiteren wird die Ausgabedatei um eine Filterung nach Sakkaden und Fixationen ergänzt, die eine direkte Interpretation und Weiterverarbeitung der Daten ermöglicht. Außerdem kann mit Hilfe der Visualisierung der Gaze-Daten die Korrektheit der erfassten Daten geprüft sowie eine ad-hoc Analyse der Aufnahme durchgeführt werden.
Die Einrichtung und Steuerung der Eye-Tracker ist über eine grafische Benutzeroberfläche möglich, die durch angemessenes Feedback und eine klare Struktur bei der Bedienung der Anwendung unterstützt. Dies macht

den Prototypen leicht zugänglich für Personen mit unterschiedlichem Vorwissen bezüglich Eye-Tracking-Aufnahmen.
Die Evaluation des Prototypen hat ergeben, dass die Aufzeichnung und Verarbeitung in der Regel korrekte Gaze-Daten liefert und die Erkennung des aktuell betrachteten Bildschirms sowie die Erstellung einer synchronen Bildschirmaufnahme zuverlässig funktioniert. Jedoch hat der gewählte Ansatz eine Limitierung: Durch die gleichzeitige Nutzung von zwei Eye-Trackern müssen die beiden Bildschirme in einem Winkel von 135° und mit einem Abstand von 15-20cm zueinander aufgestellt werden. Auf diese Weise werden Interferenzen verhindert, welche die Aufnahmen ansonsten teilweise unbrauchbar machen würden. Dies schränkt die Nutzbarkeit des Prototypen in verschiedenen Umgebungen ein, welche die nötigen Anpassungen bei der Bildschirmpositionierung nicht ermöglichen.
Es lässt sich also festhalten, dass der entwickelte Prototyp die Nutzung von Dual-Monitor-Setups im Rahmen von Eye-Tracking-Aufnahmen ermöglicht. Dabei bietet er eine grafische Benutzeroberfläche, die den Benutzer bei der Einrichtung und Durchführung von Eye-Tracking-Aufnahmen unterstützt. Durch die notwendigen Einschränkungen bei der Positionerung der Bildschirme, kann dieser Ansatz jedoch nicht immer die vollständige Nachbildung von realistischen Arbeitsbedingungen, bezogen auf die Positionierung der Bildschirme, ermöglichen. Jedoch bietet der Prototyp durch die nun ermöglichte Nutzung von zwei Bildschirmen im Rahmen von Eye-Tracking-Aufnahmen trotzdem eine deutliche Verbesserung gegenüber der bisherigen Beschränkung auf einen Bildschirm.

8.2 Ausblick

Die konzeptuelle Limitierung des Prototypen aufgrund von Interferenzen, die eine entsprechende Positionierung der Bildschirme erfordert, kann ohne eine generelle Anpassung der Hardware voraussichtlich nicht gelöst werden. Bei der grundlegenden Neuentwicklung entsprechender Hard- und Software ist die Abstimmung der Komponenten auf die gleichzeitige Nutzung von zwei Eye-Trackern, wie bei der Arbeit von Balthasar et al. [2], zu bevorzugen, um technisch bedingte Limitierungen zu verhindern.
Wenn jedoch bereits vorhandene Eye-Tracker für die Eye-Tracking-Aufnahme an mehreren Bildschirmen genutzt werden sollen, die auf einer ähnlichen Technik, wie die Tobii X3 120, basieren, dann kann der in dieser Arbeit präsentierte Ansatz genutzt werden. Besonders die Entstehung von Interferenzen in den angrenzenden Randbereichen der Bildschirme, könnte weiter untersucht werden. Möglicherweise könnte eine andere Positionierung der Eye-Tracker oder Bildschirme Abhilfe schaffen. Zudem kann eine gezielte Aktivierung und Deaktiverung der Tobii X3-120 Eye-Tracker untersucht werden. Hierbei wäre es interessant herauszufinden, wie

groß der Datenverlust beim Wechseln der Eye-Tracker ausfällt und ob der möglicherweise entstehende Datenverlust vernachlässigbar wäre im Vergleich zur aktuellen Beschränkung in der Positionierung der Bildschirme.
Des Weiteren könnte der entwickelte Prototyp um weitere Funktionen und Einstellungsmöglichkeiten, wie in Abschnitt 7.2 aufgeführt, erweitert werden. Dies würde den Funktionsumfang der Anwendung vergrößern. Außerdem könnten die implementierten Algorithmen zur Zusammenführung von Eye-Tracking-Daten sowie die Wahl verschiedener Werte zur Berechnung näher evaluiert werden. Die im Rahmen der Arbeit gewählten Werte orientieren sich entweder an den von Olsen [18] beschriebenen Werten oder sie wurden durch die Analyse von Testdaten ausgewählt. Eine tiefergehende Analyse und Evaluierung der Daten könnte die Algorithmen und somit den Prototypen weiter optimieren.

Anhang A

Ausgabedateien

A.1 Auszug aus Eye-Tracking Rohdaten

- *device_time_stamp*: Zeitstempel des Eye-Trackers in Mikrosekunden
- *system_time_stamp*: Zeitstempel des Computers in Mikrosekunden
- *gaze_point_on_display_area*: Gaze-Point relativ zum Bildschirm je Auge
- *gaze_point_in_user_coordinate_system*: Gaze-Point relativ zum Eye-Tracker je Auge
- *gaze_point_validity*: Korrektheit des Gaze-Points je Auge
- *pupil_diameter*: Pupillendurchmesser je Auge
- *pupil_validity*: Korrektheit des Pupillendurchmesser je Auge
- *gaze_origin_in_user_coordinate_system*: Augenbposition relativ zum Eye-Tracker je Auge
- *gaze_origin_in_trackbox_coordinate_system*: Augenposition innerhalb der Trackbox je Auge
- *gaze_origin_validity*: Korrektheit der Augenposition je Auge
- *project_name*: Projektname
- *recording_name*: Aufnahmebezeichnung
- *participant_name*: Name der aufgenommenen Person
- *system_boot_time*: Zeitstempel des Bootzeitpunkts des Computers in Millisekunden seit Unix Epoch

Auszug aus Eye-Tracking Rohdatei

device_time_stamp	system_time_stamp	left_gaze_point_on_display_area	left_gaze_point_in_user_coordinate_system	left_gaze_point_validity	left_pupil_diameter	left_pupil_validity
7487843608	8041475234	(0.0765693187713623, 1.0224255323410034)	(-225.20896911621094, 14.191741943359375, -3.7086639404296875)	1	nan	0
7487851943	8041483569	(0.09528233855962753, 1.0151623487472534)	(-215.2377166748047, 16.1956787109375, -2.8580474853515625)	1	nan	0
7487860283	8041491911	(0.09450357407331467, 1.0115907192230225)	(-215.65267944335938, 17.18109130859375, -2.4397659301757812)	1	4.3920440673828125	1
7487868606	8041500235	(0.09600579738616943, 1.0042787790298462)	(-214.8522186279297, 19.198486328125, -1.58343505859375)	1	nan	0
7487876950	8041508580	(0.09668983519077301, 1.0023528337478638)	(-214.48773193359375, 19.7298583984375, -1.3578872680664062)	1	nan	0
7487885270	8041516901	(0.0967407301068306, 1.0026211738586426)	(-214.46060180664062, 19.65582275390625, -1.389312744140625)	1	4.38385009765625	1
7487893601	8041525233	(0.09887667745351791, 1.0049961805343628)	(-213.3224639892578, 19.00054931640625, -1.6674575805664062)	1	nan	0
7487901949	8041533582	(0.10818661749362946, 1.0099167823791504)	(-208.3616485595703, 17.6429443359375, -2.2437210083007812)	1	nan	0
7487910269	8041541903	(0.11289749294519424, 1.013147234916687)	(-205.85145568847656, 16.75164794921875, -2.6220550537109375)	1	4.355438232421875	1
7487918601	8041550237	(0.11608492583036423, 1.0092867612838745)	(-204.1530303955078, 17.8167724609375, -2.1699371337890625)	1	nan	0
7487926945	8041558582	(0.11698411405086517, 1.0111393928527832)	(-203.6739044189453, 17.30560302734375, -2.3869094848632812)	1	nan	0
7487935266	8041566904	(0.11870334297418594, 1.009869933128357)	(-202.7578125, 17.655853271484375, -2.2382354736328125)	1	4.3587646484375	1
7487943597	8041575236	(0.11808451265096664, 1.0114789009094238)	(-203.08755493164062, 17.211944580078125, -2.426666259765625)	1	nan	0
7487951942	8041583582	(0.11246427893638611, 1.0112415552139282)	(-206.08230590820312, 17.277435302734375, -2.3988723754882812)	1	nan	0
7487960273	8041591915	(0.12032763659954071, 1.028724193572998)	(-201.89230346679688, 12.453948974609375, -4.446319580078125)	1	4.38482666015625	1
7487968595	8041600238	(0.12021410465240479, 1.0287768840789795)	(-201.95278930664062, 12.43939208984375, -4.452491760253906)	1	nan	0
7487976941	8041608585	(0.12095318734645844, 1.0269362926483154)	(-201.55897521972656, 12.947235107421875, -4.2369384765625)	1	nan	0
7487985259	8041616904	(0.100991390645504, 1.0075949430465698)	(-212.1956329345703, 18.283538818359375, -1.9718017578125)	1	4.374053955078125	1
7487993593	8041625239	(0.10300155729055405, 1.0091537237167358)	(-211.12451171875, 17.85345458984375, -2.15435791015625)	1	nan	0
7488001939	8041633586	(0.10467472672462463, 1.0113965272903442)	(-210.23297119140625, 17.23468017578125, -2.417022705078125)	1	nan	0
7488010270	8041641919	(0.11210241913795471, 1.0110857486724854)	(-206.27511596679688, 17.320404052734375, -2.3806228637695312)	1	4.3724212646484375	1
7488018600	8041650249	(0.11322064697742462, 1.0129872560501099)	(-205.67926025390625, 16.7957763671875, -2.6033172607421875)	1	nan	0
7488026936	8041658587	(0.11072481423616409, 1.008737564086914)	(-207.00917053222656, 17.968292236328125, -2.105621337890625)	1	nan	0
7488035256	8041666908	(0.10916317999362946, 1.0098018646240234)	(-207.84129333496094, 17.674652099609375, -2.2302627563476562)	1	4.383270263671875	1
7488043587	8041675240	(0.11182808130979538, 1.0129156112670898)	(-206.42129516601562, 16.8155517578125, -2.5949249267578125)	1	nan	0
7488051933	8041683587	(0.11572393774986267, 1.015529990196228)	(-204.3453826904297, 16.09423828125, -2.9011077880859375)	1	nan	0
7488060263	8041691919	(0.1108822226524353, 1.0198205709457397)	(-206.92529296875, 14.91046142578125, -3.4035873413085938)	1	4.3936309814453125	1
7488068596	8041700252	(0.12155243754386902, 1.0241130590438843)	(-201.23965454101562, 13.726165771484375, -3.9062957763671875)	1	nan	0
7488076931	8041708589	(0.1147267073392868, 1.0264308452606201)	(-204.8767547607422, 13.086669921875, -4.177742004394531)	1	nan	0
7488085250	8041716909	(0.10763657093048096, 1.0120397806167603)	(-208.65475463867188, 17.05718994140625, -2.4923553466796875)	1	4.3771514892578125	1
7488093594	8041725254	(0.10907090455293655, 1.0137704610824585)	(-207.8904571533203, 16.5797119140625, -2.695037841796875)	1	nan	0

left_gaze_origin_in_user_coordinate_system	left_gaze_origin_in_trackbox_coordinate_system	left_gaze_origin_validity	right_gaze_point_on_display_area
(-60.164424896240234, 13.265389442443848, 660.6546020507812)	(0.6300970315933228, 0.4713154733181, 0.5355153679847717)	1	(0.10928365588188171, 0.9614579677581787)
(-60.216060638427734, 13.24804973602295, 660.529541015625)	(0.6302332878112793, 0.4713475704193115, 0.5350984930992126)	1	(0.10826137661933899, 0.9708627462387085)
(-60.22337341308594, 13.234673500061035, 660.4288940429688)	(0.630268931388855, 0.471372127532959, 0.534762978553772)	1	(0.11092416942119598, 0.9655071496963501)
(-60.154808044433594, 13.274778366088867, 660.2525634765625)	(0.6301554441452026, 0.47127771377563477, 0.5341752171516418)	1	(0.10340239107608795, 0.9728801250457764)
(-60.21377944946289, 13.248783111572266, 660.1161499023438)	(0.6303099393844604, 0.4713280200958252, 0.5337204933166504)	1	(0.09766559302806854, 0.9716123938560486)
(-60.21651840209961, 13.243964195251465, 660.0147094726562)	(0.6303358674049377, 0.4713340401649475, 0.5333823561668396)	1	(0.0978454202413559, 0.9753260612487793)
(-60.16594314575195, 13.265289306640625, 659.8418579101562)	(0.6302605271339417, 0.47128036618232727, 0.5328062176704407)	1	(0.09697730839252472, 0.9667447805404663)
(-60.2200813293457, 13.249722480773926, 659.7323608398438)	(0.6303993463516235, 0.4713093042373657, 0.5324411988258362)	1	(0.10046271979808807, 0.961600661277771)
(-60.262271881103516, 13.262277603149414, 659.6648559570312)	(0.6305040717124939, 0.4712791740894317 6, 0.5322161912918091)	1	(0.10043877363204956, 0.9616171717643738)
(-60.16560745239258, 13.265118598937988, 659.538330078125)	(0.6303197145462036, 0.4712675213813782, 0.5317944288253784)	1	(0.09432735294103622, 0.9870343208312988)
(-60.222381591796875, 13.253567695617676, 659.4719848632812)	(0.6304558515548706, 0.47128966450691223, 0.5315732955932617)	1	(0.10473298281431198, 0.9711012840270996)
(-60.259891510009766, 13.249815940856934, 659.4454956054688)	(0.6305423378944397, 0.47129663825035095, 0.5314849615097046)	1	(0.10482306778430939, 0.9725317358970642)
(-60.15605545043945, 13.267444610595703, 659.355712890625)	(0.6303351521492004, 0.4712545275688171 4, 0.5311856865882874)	1	(0.10505810379981995, 0.9671692848205566)
(-60.205101013183594, 13.263280868530273, 659.3212280273438)	(0.6304482221603394, 0.47126203775405884, 0.5310707688331604)	1	(0.1050865650177002, 0.9652215838432312)
(-60.22396469116211, 13.263500213623047, 659.3201293945312)	(0.6304893493652344, 0.4712615311145782 5, 0.5310670733451843)	1	(0.0952692769184113, 0.9816329479217529)
(-60.104095458984375, 13.279136657714844, 659.2523193359375)	(0.6302430033683777, 0.4712246954441070 6, 0.530841052532196)	1	(0.09723438322544098, 0.9794636964797974)
(-60.14202880859375, 13.28014850616455, 659.2352294921875)	(0.6303285360336304, 0.4712217450141907, 0.5307840704917908)	1	(0.09131225943565369, 0.983528733253479)
(-60.1161003112793, 13.283548355102539, 659.2596435546875)	(0.6302675604820251, 0.4712154269218445, 0.530865490436554)	1	(0.1025342047214508, 0.9664333462715149)
(-60.03887176513672, 13.297319412231445, 659.2127685546875)	(0.630109429359436, 0.47118353843688965, 0.5307092070579529)	1	(0.10699775069952011, 0.9658061265945435)
(-60.05924606323242, 13.30038070678711, 659.2181396484375)	(0.6301525831222534, 0.4711771607398987, 0.5307271480560303)	1	(0.11041078716516495, 0.9717690944671631)
(-60.123233795166016, 13.3075532913208, 659.2412109375)	(0.630286693572998, 0.4711626172065735, 0.5308040380477905)	1	(0.10088671743869781, 0.9732298254966736)
(-59.988555908203125, 13.31546401977539, 659.1864624023438)	(0.6300055980682373, 0.4711430668830871 6, 0.5306215286254883)	1	(0.11089150607585907, 0.974323034286499)
(-60.03117370605469, 13.32096004486084, 659.1616821289062)	(0.6301028728485107, 0.4711300730705261, 0.5305389165878296)	1	(0.1083863377571106, 0.9754819869995117)
(-60.0523796081543, 13.33282470703125, 659.1659545898438)	(0.6301479935646057, 0.4711045622825622 6, 0.5305531620979309)	1	(0.11435101926326752, 0.9839197397232056)
(-59.929603576660156, 13.34238338470459, 659.0924682617188)	(0.6298964023590088, 0.4710806012153625 5, 0.5303082466125488)	1	(0.10488229990005493, 0.9814456701278687)
(-59.96307373046875, 13.346653938293457, 659.0660400390625)	(0.6299741268157959, 0.4710702002048492 4, 0.5302201509475708)	1	(0.10404898226261139, 0.9810085296630859)
(-59.94423294067383, 13.364989280700684, 659.075439453125)	(0.6299314498901367, 0.4710308611392975, 0.5302514433860779)	1	(nan, nan)
(-59.87667465209961, 13.352258682250977, 659.0049438476562)	(0.6297988891601562, 0.4710553586483001 7, 0.5300164818763733)	1	(0.1020655557513237, 0.9628534317016602)
(-59.913089752197266, 13.359718322753906, 658.9838256835938)	(0.6298819780349731, 0.471038281917572, 0.5299460887908936)	1	(0.1048579066991806, 0.9586943984031677)
(-59.98992919921875, 13.34723949432373, 658.9926147460938)	(0.6300468444824219, 0.4710657000541687, 0.5299753546714783)	1	(0.10486451536417007, 0.958694577217102)
(-59.85236740112305, 13.365358352661133, 658.9325561523438)	(0.6297604441642761, 0.4710237979888916, 0.5297752022743225)	1	(0.1064001396298408 5, 0.9709087610244751)

right_gaze_point_in_user_coordinate_system	right_gaze_point_validity	right_pupil_diameter	right_pupil_validity	right_gaze_origin_in_user_coordinate_system
(-207.777099609375, 31.0128173828125, 3.4314498901367188)	1	nan	0	(2.769040107727051, 15.86884880065918, 662.9920043945312)
(-208.3218231201172, 28.41802978515625, 2.3300247192382812)	1	nan	0	(2.71527361869812, 15.87977123260498, 662.9078979492188)
(-206.9029541015625, 29.8956298828125, 2.9572372436523438)	1	4.482421875	1	(2.728591203689575, 15.888901710510254, 662.83349609375)
(-210.9109344482422, 27.861419677734375, 2.0937652587890625)	1	nan	0	(2.749985933303833, 15.899663925170898, 662.6746826171875)
(-213.96778869628906, 28.211181640625, 2.2422332763671875)	1	nan	0	(2.6920361518859863, 15.901037216186523, 662.5713500976562)
(-213.87197875976562, 27.18658447265625, 1.80731201171875)	1	4.4334869384765625	1	(2.7058639526367188, 15.907404899597168, 662.4849853515625)
(-214.3345489501953, 29.554168701171875, 2.8122940063476562)	1	nan	0	(2.7175235748291016, 15.905014991760254, 662.318603515625)
(-212.4773406982422, 30.97344970703125, 3.4147415161132812)	1	nan	0	(2.6676292419433594, 15.897680282592773, 662.2266845703125)
(-212.49009704589844, 30.968902587890625, 3.4128036499023438)	1	4.4329986572265625	1	(2.6768975257873535, 15.903413772583008, 662.1605224609375)
(-215.74658203125, 23.956268310546875, 0.43611907958984375)	1	nan	0	(2.694533109664917, 15.897530555725098, 662.019287109375)
(-210.20193481445312, 28.352203369140625, 2.3020858764648438)	1	nan	0	(2.639421224594116, 15.902438163757324, 661.9547119140625)
(-210.1539306640625, 27.957550048828125, 2.1345672607421875)	1	4.4492034912109375	1	(2.662890911102295, 15.903559684753418, 661.9140625)
(-210.0286865234375, 29.43707275390625, 2.7625808715820312)	1	nan	0	(2.6952569484710693, 15.906756401062012, 661.7959594726562)
(-210.01351928710938, 29.97442626953125, 2.9906768798828125)	1	nan	0	(2.6601474285125732, 15.91098690032959, 661.7525024414062)
(-215.2446746826172, 25.446502685546875, 1.0686874389648438)	1	4.389404296875	1	(2.6840498447418213, 15.915908813476562, 661.7279052734375)
(-214.19757080078125, 26.045013427734375, 1.3227386474609375)	1	nan	0	(2.7217600345611572, 15.918216705322266, 661.62353515625)
(-217.35317993164062, 24.9234619140625, 0.8466720581054688)	1	nan	0	(2.6973037719726562, 15.920424461364746, 661.590576171875)
(-211.37355041503906, 29.640106201171875, 2.8487625122070312)	1	4.4319305419921875	1	(2.709364175796509, 15.925920486450195, 661.586181640625)
(-208.99514770507812, 29.81317138671875, 2.9222183227539062)	1	nan	0	(2.795619249343872, 15.925707817077637, 661.5062866210938)
(-207.17649841308594, 28.16796875, 2.223876953125)	1	nan	0	(2.7714571952819824, 15.931147575378418, 661.5003051757812)
(-212.25140380859375, 27.76495361328125, 2.0528106689453125)	1	4.6070098876953125	1	(2.80593204498291, 15.939556121826172, 661.5091552734375)
(-206.92034912109375, 27.46331787109375, 1.9247817993164062)	1	nan	0	(2.8583474159240723, 15.92404842376709, 661.4276733398438)
(-208.25523376464844, 27.1435546875, 1.7890472412109375)	1	nan	0	(2.8376030921936035, 15.931267738342285, 661.4019165039062)
(-205.0769500732422, 24.815582275390625, 0.800872802734375)	1	4.49237060546875	1	(2.873236894607544, 15.933934211730957, 661.397705078125)
(-210.1223602294922, 25.4981689453125, 1.0906219482421875)	1	nan	0	(2.9048147201538086, 15.928223609924316, 661.3060913085938)
(-210.56640625, 25.6187744140625, 1.141815185546875)	1	nan	0	(2.899731159210205, 15.933929443359375, 661.2841796875)
(nan, nan, nan)	0	nan	0	(nan, nan, nan)
(-211.62326049804688, 30.6278076171875, 3.2680206298828125)	1	nan	0	(2.963212013244629, 15.918842315673828, 661.2085571289062)
(-210.13536071777344, 31.775299072265625, 3.7551040649414062)	1	nan	0	(2.940086841583252, 15.93554973602295, 661.1970825195312)
(-210.1318359375, 31.775238037109375, 3.7550811767578125)	1	4.4564971923828125	1	(2.9579710960388184, 15.940519332885742, 661.2100830078125)
(-209.31358337402344, 28.40533447265625, 2.3246383666992188)	1	nan	0	(3.0059869289398193, 15.941009521484375, 661.1365966796875)

right_gaze_origin_in_trackbox_coordinate_system	right_gaze_origin_validity	project_name	recording_name	participant_name	system_boot_time
(0.49403345584869385, 0.46580684185028076, 0.5433067083358765)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4941485524177551, 0.46577897667884827, 0.543026328086853)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.49411919713020325, 0.4657554626464844, 0.5427783131599426)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4940716624259949, 0.46572405099868774, 0.5422489643096924)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.49419569969177246, 0.4657157361507416, 0.5419045090675354)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4941651225090027, 0.4656975269317627, 0.5416166186332703)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4941385090351105, 0.4656940698623657, 0.5410619974136353)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.494245320558548, 0.46570515632629395, 0.5407556295394897)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.49422475695610046, 0.4656893312931061, 0.5405350923538208)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4941854774951935, 0.46569472551345825, 0.54006427526474)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.494303822517395, 0.4656807780265808, 0.53984942892456)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4942528307437897, 0.4656762480735779, 0.5397135615348816)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.49418193101882935, 0.46566322445869446, 0.5393198728561401)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4942573606967926, 0.46565183997154236, 0.5391750335693359)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4942055344581604, 0.46563994884490967, 0.539093017578125)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4941231906414032, 0.465629518032074, 0.538745105266571)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4941757023334503, 0.46562305092811584, 0.53863525390625)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4941496253013611, 0.46561095118522644, 0.5386205911636353)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4939626455307007, 0.46560725569725037, 0.5383542776107788)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.49401476979255676, 0.46559521555900574, 0.5383343696594238)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.49394041299819946, 0.46557751297950745, 0.5383638739585876)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4938264489173889, 0.4656067490577698, 0.5380922555923462)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4938710331916809, 0.46558982133865356, 0.5380063652992249)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4937940239906311, 0.465583860874176, 0.5379923582077026)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4937249422073364, 0.4655914306640625, 0.5376869440078735)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4937357008457184, 0.46557796001434326, 0.5376139283180237)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(nan, nan, nan)	0	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4935978353023529, 0.46560660004615784, 0.5373618602752686)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4936476945877075, 0.4655699133872986, 0.5373235940933228)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.4936091899871826, 0.46555984020233154, 0.5373669266700745)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792
(0.49350473284721375, 0.46555495262145996, 0.5371220111846924)	1	Test Projekt	benchmark_Benchmark_Do_2	Thorben	1686208772792

A.2 Auszug aus finaler Eye-Tracking Ausgabedatei

- *Recording timestamp*: Zeitstempel der Aufnahme in Mikrosekunden
- *Computer timestamp (Epoch Unix)*: Zeitstempel der Aufnahme seit Unix Epoch
- *Sensor*: Sensor, der das Sample erfasst hat.
- *Project name*: Projektname
- *Export date*: Datum der Erstellung der Ausgabedatei
- *Participant name*: Name der aufgenommenen Person
- *Recording name*: Aufnahmebezeichnung
- *Recording date*: Datum der Aufnahme
- *Recording start time*: Zeitpunkt des Aufnahmestarts
- *Recording resolution*: Anzahl der Pixel des Bildschirms (Breite/Höhe)
- *Eyetracker timestamp*: Zeitstempel des Eye-Trackers in Mikrosekunden
- *Gaze point*: Mittelwert des für beide Augen erfassten Gaze-Points in Bildschirmpixeln
- *Gaze point left*: Gaze-Point des linken Auges in Bildschirmpixeln
- *Gaze point right*: Gaze-Point des rechten Auges in Bildschirmpixeln
- *Validity*: Korrektheit des Gaze-Points je Auge
- *Pupil diameter*: Pupillendurchmesser je Auge
- *Eye movement type*: Bezeichnung der Augenbewegung
- *Gaze event duration*: Dauer der aktuell erkannten Augenbewegung
- *Fixation point*: Fixationspunkt in Bildschirmpixeln

Eye-Tracking Ausgabedatei

Recording timestamp	Computer timestamp (Epoch Unix)	Sensor	Project name	Export date	Participant name	Recording name	Recording date	Recording start time	Recording duration	Recording resolution height left
52057015	1686216848824015	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52065229	1686216848832229	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52073600	1686216848840600	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52081941	1686216848848941	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52091556	1686216848858556	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52098616	1686216848865616	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52106905	1686216848873905	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52115257	1686216848882257	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52123624	1686216848890624	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52131973	1686216848898973	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52142060	1686216848909060	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52148681	1686216848915681	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52156928	1686216848923928	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52165257	1686216848932257	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52173666	1686216848940666	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52181947	1686216848948947	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52190240	1686216848957240	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52198671	1686216848965671	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52206907	1686216848973907	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52215297	1686216848982297	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52223662	1686216848990662	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52231947	1686216848998947	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52240239	1686216849007239	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52248652	1686216849015652	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52256928	1686216849023928	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52265249	1686216849032249	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52273740	1686216849040740	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52281974	1686216849048974	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52292119	1686216849059119	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52298543	1686216849065543	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200
52306907	1686216849073907	Eye Tracker	Test Projekt	08.06.2023	Thorben	benchmark_Beispiel	08.06.2023	11:32:04.000	179910276	1200

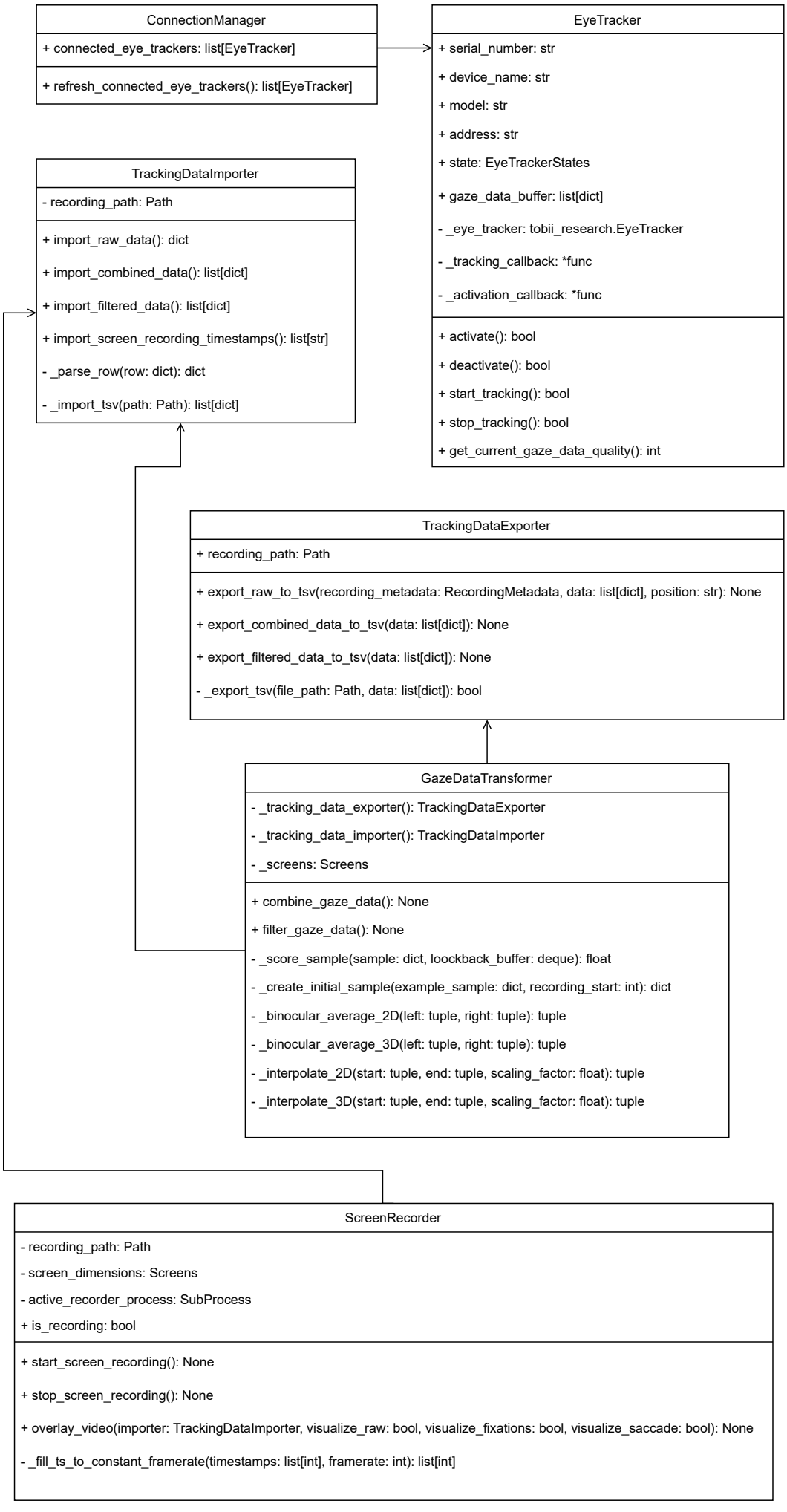
Recording resolution width left	Recording resolution height right	Recording resolution width right	Eyetracker timestamp	Side	Gaze point X	Gaze point Y	Gaze point left X	Gaze point left Y
1920	1200	1920	8076032015	right	1764.9815940856934	1179.7109127044678	1764.4504165649414	1183.9075326919556
1920	1200	1920	8076040229	right	1741.959342956543	1153.7954449653625	1727.764892578125	1133.4079027175903
1920	1200	1920	8076048600	right	1753.2044792175293	1179.2055487632751	1746.7695236206055	1175.6823778152466
1920	1200	1920	8076056941	right	1756.7703437805176	1172.7180004119873	1743.9450073242188	1161.8981838226318
1920	1200	1920	8076066556	right	1760.3788948059082	1179.8313975334167	1748.0208206176758	1172.809624671936
1920	1200	1920	8076073616	right	1759.1588973999023	1180.9038162231445	1744.178466796875	1175.6974697113037
1920	1200	1920	8076081905	right	1759.8350143432617	1181.3886880874634	1743.1729888916016	1175.1049518585205
1920	1200	1920	8076090257	right	1761.320457458496	1168.864095211029	1743.0111694335938	1158.5941314697266
1920	1200	1920	8076098624	right	1763.759994506836	1171.1186170578003	1743.8225555419922	1161.1448764801025
1920	1200	1920	8076106973	right	1758.6784744262695	1170.4912304878235	1739.0473937988281	1158.0613374710083
1920	1200	1920	8076117060	right	1761.8416213989258	1185.5522274971008	1740.4248046875	1179.1387796401978
1920	1200	1920	8076123681	right	1769.2365074157715	1184.282112121582	1736.5842819213867	1176.3178825378418
1920	1200	1920	8076131928	right	1764.5585632324219	1179.7237873077393	1740.4264068603516	1172.4753856658936
1920	1200	1920	8076140257	right	1765.041217803955	1165.4363751411438	1737.4937438964844	1151.487421989441
1920	1200	1920	8076148666	right	1769.0683364868164	1163.7886047363281	1739.1487884521484	1147.9807376861572
1920	1200	1920	8076156947	right	1770.498161315918	1165.9476041793823	1736.4836883544922	1152.0182132720947
1920	1200	1920	8076165240	right	1771.4590644836426	1177.7019739151	1741.589126586914	1167.8670644760132
1920	1200	1920	8076173671	right	1768.9843368530273	1177.600908279419	1733.8639068603516	1171.419095993042
1920	1200	1920	8076181907	right	1747.122459411621	1172.035789489746	1694.3474578857422	1158.8740825653076
1920	1200	1920	8076190297	right	1742.8317260742188	1173.1436848640442	1694.3401336669922	1161.793327331543
1920	1200	1920	8076198662	right	1742.9977798461914	1178.9787769317627	1702.7327728271484	1167.0175552368164
1920	1200	1920	8076206947	right	1759.552001953125	1183.9439034461975	1744.9003601074219	1174.8026132583618
1920	1200	1920	8076215239	right	1760.8791732788086	1185.0355982780457	1741.2210845947266	1177.2510051727295
1920	1200	1920	8076223652	right	1761.0040283203125	1183.1623077392578	1740.1437377929688	1171.5051412582397
1920	1200	1920	8076231928	right	1752.312183380127	1181.5627098083496	1721.0943603515625	1171.1308479309082
1920	1200	1920	8076240249	right	1748.7659454345703	1179.975414276123	1722.6583099365234	1170.1067447662354
1920	1200	1920	8076248740	right	1750.099868774414	1186.540138721466	1727.2727966308594	1188.536024093628
1920	1200	1920	8076256974	right	1765.005226135254	1180.658197402954	1754.3463134765625	1178.3802509307861
1920	1200	1920	8076267119	right	1740.4675483703613	1168.921458721161	1701.5064239501953	1155.2837133407593
1920	1200	1920	8076273543	right	1763.0871963500977	1171.5000629425049	1732.0957946777344	1162.1773481369019
1920	1200	1920	8076281907	right	1764.3782043457031	1169.124948978424	1735.1007843017578	1157.3561668395996

Validity left	Gaze point right X	Gaze point right Y	Validity right	Pupil diameter left	Pupil diameter right	Eye movement type	Gaze event duration	Fixation point X	Fixation point Y
1	1765.5127716064453	1175.51429271698	1	4.3224639892578125	5.0516357421875	Fixation	295769.0	1760	1175
1	1756.153793334961	1174.1829872131348	1			Fixation	295769.0	1760	1175
1	1759.6394348144531	1182.7287197113037	1			Fixation	295769.0	1760	1175
1	1769.5956802368164	1183.5378170013428	1	4.2696533203125	5.192840576171875	Fixation	295769.0	1760	1175
1	1772.7369689941406	1186.8531703948975	1			Fixation	295769.0	1760	1175
1	1774.1393280029297	1186.1101627349854	1			Fixation	295769.0	1760	1175
1	1776.4970397949219	1187.6724243164062	1	4.2615966796875	4.8922576904296875	Fixation	295769.0	1760	1175
1	1779.6297454833984	1179.1340589523315	1			Fixation	295769.0	1760	1175
1	1783.6974334716797	1181.092357635498	1			Fixation	295769.0	1760	1175
1	1778.309555053711	1182.9211235046387	1	4.3002777099609375	5.034088134765625	Fixation	295769.0	1760	1175
1	1783.2584381103516	1191.965675354004	1			Fixation	295769.0	1760	1175
1	1801.8887329101562	1192.2463417053223	1			Fixation	295769.0	1760	1175
1	1788.6907196044922	1186.972188949585	1	4.2570343017578125	5.0845794677734375	Fixation	295769.0	1760	1175
1	1792.5886917114258	1179.3853282928467	1			Fixation	295769.0	1760	1175
1	1798.9878845214844	1179.596471786499	1			Fixation	295769.0	1760	1175
1	1804.5126342773438	1179.87699508667	1	4.2918243408203125	5.0695953369140625	Fixation	295769.0	1760	1175
1	1801.329002380371	1187.536883354187	1			Fixation	295769.0	1760	1175
1	1804.1047668457031	1183.782720565796	1			Fixation	295769.0	1760	1175
1	1799.8974609375	1185.1974964141846	1	4.5469970703125	4.863677978515625	Fixation	295769.0	1760	1175
1	1791.3233184814453	1184.4940423965454	1			Fixation	295769.0	1760	1175
1	1783.2627868652344	1190.939998626709	1			Fixation	295769.0	1760	1175
1	1774.2036437988281	1193.0851936340332	1	4.24859619140625	5.0345458984375	Fixation	295769.0	1760	1175
1	1780.5372619628906	1192.8201913833618	1			Fixation	295769.0	1760	1175
1	1781.8643188476562	1194.8194742202759	1			Fixation	295769.0	1760	1175
1	1783.5300064086914	1191.994571685791	1	5.07257080078125	5.1007537841796875	Fixation	295769.0	1760	1175
1	1774.8735809326172	1189.8440837860107	1			Fixation	295769.0	1760	1175
1	1772.9269409179688	1184.5442533493042	1			Fixation	295769.0	1760	1175
1	1775.6641387939453	1182.936143875122	1	4.2926788330078125	4.9080047607421875	Fixation	295769.0	1760	1175
1	1779.4286727905273	1182.5592041015625	1			Fixation	295769.0	1760	1175
1	1794.078598022461	1180.822777748108	1			Fixation	295769.0	1760	1175
1	1793.6556243896484	1180.8937311172485	1	4.254730224609375	4.91058349609375	Saccade	12492.0		

Anhang B

Klassendiagramm

B.1 Klassendiagramm des Core-Moduls der Engine


ConnectionManager
+ connected_eye_trackers: list[EyeTracker]
+ refresh_connected_eye_trackers(): list[EyeTracker]
EyeTracker
+ serial_number: str
+ device_name: str
+ model: str
+ address: str
+ state: EyeTrackerStates
+ gaze_data_buffer: list[dict]
- _eye_tracker: tobii_research.EyeTracker
- _tracking_callback: *func
- _activation_callback: *func
+ activate(): bool
+ deactivate(): bool
+ start_tracking(): bool
+ stop_tracking(): bool
+ get_current_gaze_data_quality(): int
TrackingDataImporter
- recording_path: Path
+ import_raw_data(): dict
+ import_combined_data(): list[dict]
+ import_filtered_data(): list[dict]
+ import_screen_recording_timestamps(): list[str]
- _parse_row(row: dict): dict
- _import_tsv(path: Path): list[dict]
TrackingDataExporter
+ recording_path: Path
+ export_raw_to_tsv(recording_metadata: RecordingMetadata, data: list[dict], position: str): None
+ export_combined_data_to_tsv(data: list[dict]): None
+ export_filtered_data_to_tsv(data: list[dict]): None
- _export_tsv(file_path: Path, data: list[dict]): bool
GazeDataTransformer
- _tracking_data_exporter(): TrackingDataExporter
- _tracking_data_importer(): TrackingDataImporter
- _screens: Screens
+ combine_gaze_data(): None
+ filter_gaze_data(): None
- _score_sample(sample: dict, loockback_buffer: deque): float
- _create_initial_sample(example_sample: dict, recording_start: int): dict
- _binocular_average_2D(left: tuple, right: tuple): tuple
- _binocular_average_3D(left: tuple, right: tuple): tuple
- _interpolate_2D(start: tuple, end: tuple, scaling_factor: float): tuple
- _interpolate_3D(start: tuple, end: tuple, scaling_factor: float): tuple
ScreenRecorder
- recording_path: Path
- screen_dimensions: Screens
- active_recorder_process: SubProcess
+ is_recording: bool
+ start_screen_recording(): None
+ stop_screen_recording(): None
+ overlay_video(importer: TrackingDataImporter, visualize_raw: bool, visualize_fixations: bool, visualize_saccade: bool): None
- _fill_ts_to_constant_framerate(timestamps: list[int], framerate: int): list[int]

Literaturverzeichnis

[1] N. Ali, Z. Sharafl, Y.-G. Guéhéneuc, and G. Antoniol. An empirical study on requirements traceability using eye-tracking. In *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pages 191–200, 2012.

[2] S. Balthasar, M. Martin, F. van de Camp, J. Hild, and J. Beyerer. Combining low-cost eye trackers for dual monitor eye tracking. In *Human-Computer Interaction. Interaction Platforms and Techniques: 18th International Conference, HCI International 2016, Toronto, ON, Canada, July 17-22, 2016. Proceedings, Part II 18*, pages 3–12. Springer, 2016.

[3] R. Bednarik. Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *International Journal of Human-Computer Studies*, 70(2):143–155, 2012.

[4] A. D. Birrell and B. J. Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems (TOCS)*, 2(1):39–59, 1984.

[5] M. Broy, E. Geisberger, J. Kazmeier, A. Rudorfer, and K. Beetz. Ein requirements-engineering-referenzmodell. *Informatik-Spektrum*, 30(3):127–142, 2007.

[6] J. Coddington, J. Xu, S. Sridharan, M. Rege, and R. Bailey. Gaze-based image retrieval system using dual eye-trackers. In *2012 IEEE International Conference on Emerging Signal Processing Applications*, pages 37–40. IEEE, 2012.

[7] T. N. Cornsweet and H. D. Crane. Accurate two-dimensional eye tracker using first and fourth purkinje images. *J. Opt. Soc. Am.*, 63(8):921–928, Aug 1973.

[8] M. Crosby and J. Stelovsky. How do we read algorithms? a case study. *Computer*, 23(1):25–35, 1990.

[9] A. T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Springer Publishing Company, Incorporated, 3rd edition, 2017.

[10] S. Eggers. Automatische Erkennung von Textelementen in Bildschirmaufnahmen für die Eye Tracking Datenanalyse. Bachelor's thesis, Gottfried Wilhelm Leibniz Universität Hannover, 08 2021.

[11] gRPC Authors. gRPC. `https://grpc.io/`. letzer Zugriff: 04 2023.

[12] M. A. Just and P. A. Carpenter. Eye fixations and cognitive processes. *Cognitive Psychology*, 8(4):441–480, 1976.

[13] T. Kocejko, A. Bujnowski, and J. Wtorek. Eye mouse for disabled. In *2008 Conference on human system interactions*, pages 199–202. IEEE, 2008.

[14] T. Kocejko and J. Wtorek. Gaze tracking in multi-display environment. In *2013 6th International Conference on Human System Interactions (HSI)*, pages 626–631. IEEE, 2013.

[15] J. Merchant, R. Morrissette, and J. L. Porterfield. Remote measurement of eye direction allowing subject motion over one cubic foot of space. *IEEE transactions on biomedical engineering*, (4):309–317, 1974.

[16] Microsoft. Microsoft Remote Procedure Call. `https://learn.microsoft.com/en-us/windows/win32/Rpc/rpc-start-page`. letzer Zugriff: 04 2023.

[17] C. Nitschke, A. Nakazawa, and H. Takemura. Corneal imaging revisited: An overview of corneal reflection analysis and applications. *IPSJ Transactions on Computer Vision and Applications*, 5:1–18, 2013.

[18] A. Olsen. The tobii i-vt fixation filter. `https://go.tobii.com/Tobii-I-VT-fixation-filter-white-paper`, 03 2012. letzer Zugriff: 05 2023.

[19] K. Rayner. Eye movements in reading and information processing: 20 years of research. *Psychological bulletin*, 124(3):372–422, 1998.

[20] C. Rupp and SOPHISTen. *Requirements-Engineering und -Management*. Carl Hanser Verlag GmbH & Co. KG, München, 7., aktualisierte und erweiterte auflage edition, 2020.

[21] K. Saleh, J. Iskander, D. Jia, M. Hossny, S. Nahavandi, C. Best, S. Hosking, B. Rice, A. Bhatti, and S. Hanoun. Reliable switching mechanism for low cost multi-screen eye tracking devices via deep recurrent neural networks. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3492–3497. IEEE, 2018.

[22] D. Sasse. Entwicklung eines Tools zur Erkennung von Blickmustern in Eye Tracking Daten. Bachelor's thesis, Gottfried Wilhelm Leibniz Universität Hannover, 08 2021.

[23] Z. Sharafi, B. Sharif, Y.-G. Guéhéneuc, A. Begel, R. Bednarik, and M. Crosby. A practical guide on conducting eye tracking studies in software engineering. *Empirical Software Engineering*, 25, 06 2020.

[24] Z. Sharafi, Z. Soh, and Y.-G. Guéhéneuc. A systematic literature review on the usage of eye-tracking in software engineering. *Information and Software Technology*, 67:79–107, 2015.

[25] The Standish Group International. The CHAOS Report. 1995.

[26] tobii. Tobii Pro Eye Tracker Manager. `https://www.tobii.com/products/software/applications-and-developer-kits/tobii-pro-eye-tracker-manager`. letzer Zugriff: 04 2023.

[27] tobii. Tobii Pro Lab. `https://www.tobii.com/products/software/behavior-research-software/tobii-pro-lab`. letzer Zugriff: 04 2023.

[28] tobii. Tobii Pro X3-120 - User Manual. `https://go.tobii.com/X3UM`. letzer Zugriff: 04 2023.

[29] Tobii AB. tobiipro/sdk - About the SDK. `https://developer.tobiipro.com/tobiiprosdk.html`. letzer Zugriff: 03 2023.

[30] Tobii AB. tobiipro/sdk - Coordinate Systems. `https://developer.tobiipro.com/commonconcepts/coordinatesystems.html`. letzer Zugriff: 04 2023.

[31] M. Van Steen and A. S. Tanenbaum. *Distributed systems*. CreateSpace Independent Publishing Platform, 3rd edition, 2017.

[32] L. R. Young and D. Sheena. Survey of eye movement recording methods. *Behavior research methods & instrumentation*, 7(5):397–429, 1975.