Fachhochschule Graubünden, FHGR

Visuelle Analyse von Eye Tracking Daten

Name / Vorname: Nyffeler Tamara
Matrikelnr.: 16-666-042
Adresse: Im Husacher 5, 5619 Büttikon
E-Mail: tamara.nyffeler@stud.fhgr.ch

Name / Vorname: Reiser Sharon
Matrikelnr.:17-655-366
Adresse: Alte Römerstrasse 28, 8404 Winterthur
E-Mail: sharon.reiser@stud.fhgr.ch

Name / Vorname: Pellegatta Serge
Matrikelnr.: 18-165-068
Adresse: Usterstrasse 19, 8620 Wetzikon ZH
E-Mail: serge.pellegatta@stud.fhgr.ch

Studiengang: MSc Data Visualization
Modul: Consultancy Project 1
Betreuer: Dr. rer. nat. Michael Burch
Abgabedatum: 14.07.2024

Abstract

Die Analyse von Eye Tracking Daten bietet wertvolle Einblicke in menschliches Verhalten und kognitive Prozesse. Im Rahmen dieses Projekts wurde ein interaktives Dashboard entwickelt, um die Visualisierung und Analyse von Eye Tracking Daten zu erleichtern. Ziel des Projekts war es, ein benutzerfreundliches Werkzeug zu schaffen, das Forschenden und Designern ermöglicht, Eye Tracking Daten effizient zu interpretieren und Entscheidungen basierend auf diesen Daten zu treffen.

Es wurde ein Eye Tracking Datensatz aus einer Studie zu Metrokarten aus verschiedenen Städten verwendet. Dabei lag der Fokus auf der Analyse zwischen farbigen und schwarzweissen Karten. Diese Stimuli wurden mit verschiedenen Visualisierungswerkzeugen dargestellt. Das resultierende Dashboard wurde mit der Python Bibliothek Plotly entwickelt und öffentlich zugänglich gemacht. Es ermöglicht verschiedene Ansichten: einerseits einen globalen Vergleich über alle Karten, andererseits eine Detailansicht pro Stadt mit verschiedenen Interaktions- und Filtermöglichkeiten.

Die Ergebnisse des Dashboards deuten darauf hin, dass es ein nützliches Werkzeug für Forschende im Bereich Usability-Studien und kognitive Psychologie sein kann. Die Entwicklung eines webbasierten Dashboards mit vielfältigen Interaktionen ermöglicht es Forschenden, genauere Einblicke in die Daten zu erlangen und eventuell neue Forschungsfragen aufzuwerfen oder bestehende zu beantworten.

Zukünftige Arbeiten könnten sich darauf konzentrieren, die Funktionalität des Dashboards mit statistischen Kennzahlen zu erweitern, um noch aussagekräftigere Analysen zu ermöglichen und die Anwendungsbereiche zu erweitern. Ein weiterer möglicher Ausbau des Dashboards wäre die Fähigkeit, ähnlich strukturierte Datensätze dynamisch einlesen zu können.

Schlüsselwörter:
Eye Tracking, Datenvisualisierung, interaktives Dashboard, Metrokarten, Python, Plotly

Inhaltsverzeichnis

Bildquelle Titelblatt: https://www.weareconflux.com/en/blog/eye-tracking-in-user-research/, aufgerufen am 01.04.2024

Abbildungsverzeichnis

Codeverzeichnis

Formelverzeichnis

1. Einleitung

In der nonverbalen Kommunikation spielt der Augenkontakt eine entscheidende Rolle. Durch ihn zeigen wir unsere Emotionen, Interesse oder Desinteresse und ermöglichen eine tiefere soziale Interaktion (Burch, 2022). Unsere Augen richten sich auf das, was uns interessiert. Da wir Menschen nicht in der Lage sind, mehrere Sinneseindrücke gleichzeitig zu verarbeiten, bedeutet visuelle Aufmerksamkeit, dass wir uns auf ein sichtbares Objekt konzentrieren (Duchowski, 2007). Darin liegt das grundlegende Konzept der Messung von Eye Tracking Daten. Es erklärt, wie und warum wir unseren Blick auf bestimmte Dinge richten und was dies über unsere Wahrnehmung und Interessen aussagt (Burch, 2022).

Die vorliegende Projektarbeit widmet sich dem Bereich der Visualisierungswissenschaft und der Erstellung eines webbasierten Dashboards für die Exploration von Eye Tracking Daten. Im ersten Teil wird der theoretische Ansatz bezüglich der Messung von visueller Aufmerksamkeit, der damit verbundenen Visualisierungstechniken und der Einbindung von Dashboards aufgezeigt (Kapitel 2). Anschliessend wird auf die verwendete Methodik und den zugrundeliegenden Datensatz näher eingegangen (Kapitel 3). In Kapitel 4 werden die Ergebnisse dargestellt, gefolgt von einer Implikation für die Praxis in Kapitel 5.

1.1. Projektziel

Im Rahmen der vorliegenden Projektarbeit wird ein webbasiertes Dashboard entwickelt, das vielfältige Visualisierungsmöglichkeiten zur Analyse von Eye Tracking Daten bietet. Ziel des Dashboards ist es, einen Datensatz, bestehend aus Eye Tracking Informationen von 24 Metrokarten aus verschiedenen Städten, umfassend zu explorieren und zu interpretieren. Der Datensatz basiert auf einer Studie, die untersucht, wie farbige und graustufenbasierte Metrokarten von Betrachtenden wahrgenommen werden. Die zentrale Forschungsfrage dieser Studie adressiert die Notwendigkeit farbiger Karten im Vergleich zu Graustufendarstellungen und untersucht, ob letztere einen vergleichbaren Nutzen in der Nutzerführung bieten.

Das Zentrum für Data Analytics, Visualization and Simulation (DAViS) der Fachhochschule Graubünden fungiert als Auftraggeber dieses Projekts. Im Projektverlauf wurde die Erstellung eines Wireframes (Handskizze), eines visuellen Mockups mit allen erforderlichen Interaktionsmöglichkeiten sowie die vollständige Entwicklung eines funktionalen Dashboards in Python vereinbart.

1.2. Abgrenzung

Das Ziel dieser Arbeit besteht nicht darin, Rückschlüsse oder Interpretationen aus den zugrundeliegenden Daten zu ziehen. Stattdessen liegt der Schwerpunkt auf der Entwicklung eines Werkzeugs – sprich eines Dashboards – das Forschenden in diesem Bereich

Anwendungsmöglichkeiten bietet. Die eigentliche Evaluierung der Daten wird somit nicht im Rahmen dieser Arbeit durchgeführt, sondern durch die bereitgestellten Tools unterstützt.

1.3. Zielgruppe

Das Dashboard richtet sich primär an Forschende im Bereich Eye Tracking. Es bietet eine Plattform zur Evaluation und zum Vergleich der Effektivität farbiger und graustufenbasierter Kartendarstellungen. Ziel ist es, die Entscheidungsfindung in Studien zu unterstützen, indem es ermöglicht, empirisch zu untersuchen, ob graustufenbasierte Karten gegenüber farbigen Darstellungen gleich oder ähnlich effektiv sind.

2. Theoretischer Ansatz

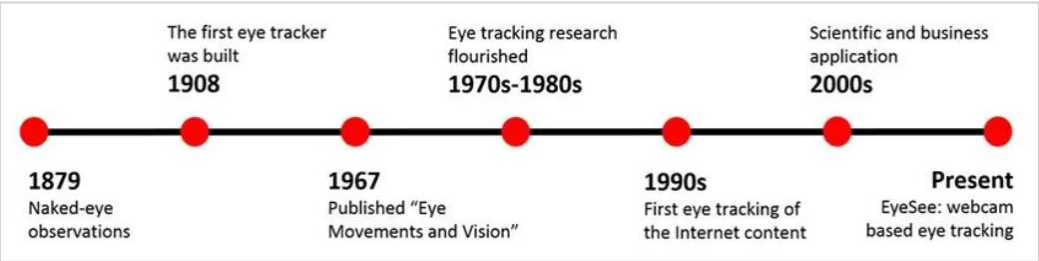
Im folgenden Kapitel wird die relevante Literatur aufgearbeitet, um den theoretischen Rahmen der Studie zu untermauern und bestehende Forschungsergebnisse sowie deren Bezug zur vorliegenden Untersuchung darzustellen.

2.1. Untersuchung der visuellen Aufmerksamkeit

Die Eye Tracking Technologie hat sich als eine zunehmend verbreitete Methode zur Untersuchung des Nutzerverhaltens etabliert und kommt in unterschiedlichen Forschungsfeldern zum Einsatz. Zu den wichtigsten Anwendungsgebieten zählen Marketing, Neurowissenschaften, Mensch-Computer-Interaktion, Visualisierungswissenschaften, Untersuchung von Szenenwahrnehmungen, optischen Suchvorgängen oder Leseforschung. Dabei wird die Augenaktivität von Probanden im Rahmen einer kontrollierten experimenteller Umgebung aufgezeichnet und analysiert (Blascheck et al., 2017).

Die Messung visueller Aufmerksamkeit bei Menschen wird bereits seit mehr als einem Jahrhundert aus qualitativer Sicht erforscht und demnach ist auch die quantitative Analyse keine neue Erfindung der letzten Jahre (Duchowski, 2007). Der russische Psychologe, Alfred Lukyanovich Yarbus, führte in den 1960er Jahre mehrere Studien zur Blickverfolgung in Abhängigkeit von vordefinierten Aufgaben durch. Die Ergebnisse – die 1967 in seinem Werk «Eye Movements and Vision» veröffentlicht wurden – verdeutlichten, dass ein Zusammenhang zwischen den Bewegungen und Fixierungspunkten der Augen und ihrem Interesse aus einer vordefinierten Aufgabe besteht (Duchowski, 2007). Damit wurde belegt, dass sich die Augen der Probanden auf die Bildbereiche fokussieren, die für die gestellten Fragen relevant sind, wenn ihnen mehrere Fragen zu den vorgelegten Bildern gestellt werden. (EyeSee Research, 2014). Abbildung 1**Fehler! Verweisquelle konnte nicht gefunden werden.** zeigt eine Übersicht der wichtigsten Meilensteine in der Forschungsgeschichte von Eye Tracking.

Abbildung 1: Übersicht der Meilensteine in der Geschichte des Eye Trackings



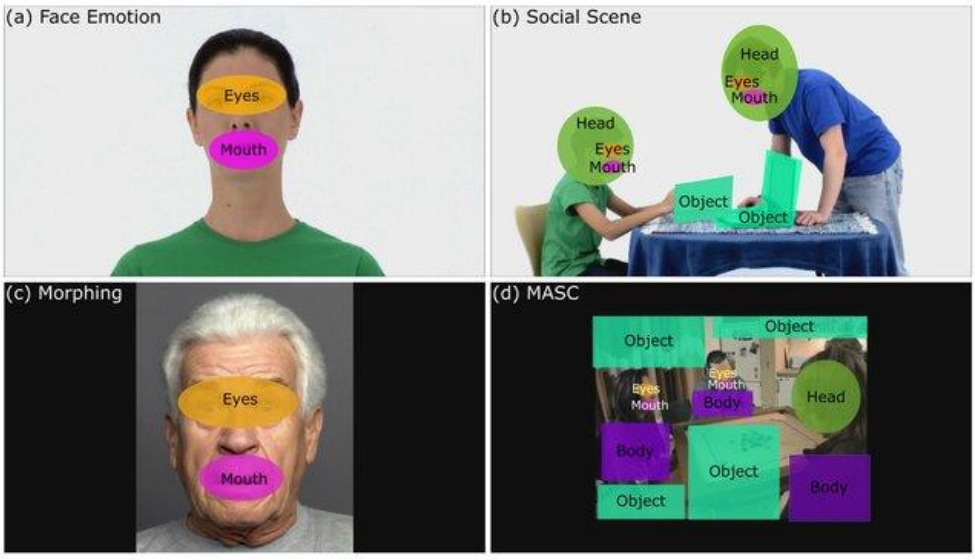
Anmerkung: (EyeSee Research, 2014)

2.1.1. Definition visual Tasks und Areas of Interest

Die aktuelle Eye Tracking Technologie fokussiert sich in erster Linie auf die Messung von Bearbeitungszeiten und Genauigkeitsraten bei der Ausführung definierter visueller Aufgaben, sogenannten visual Tasks. Visual Tasks bezeichnen spezifische Stimuli[1] oder Anforderungen, die Teilnehmenden im Kontext experimenteller Studien präsentiert werden (Burch, 2021).

Damit die Verteilung der Aufmerksamkeit auf einen gezielten Bereich des Stimulus reduziert werden kann, werden in der Praxis Interessensbereiche[2] (AOI) festgelegt. Ein AOI ist demnach eine spezifizierte Region, die von besonderem Interesse für die Quantifizierung des visual Tasks ist (Blascheck, et al., 2017). Abbildung 2 zeigt anhand vier Beispielen, wie AOI's für verschiedene Aufgaben dynamisch oder manuell gruppiert werden können (Prillinger et al., 2023). In diesem Beispiel fokussieren sich die Interessensbereiche je nach Aufgabe auf verschiedene Gesichtsteile und Objekte in sozialen Szenen und morphenden Bildern. Bei «Face Emotion»(a) und «Morphing» (c) werden vorwiegend Augen und Mund betrachtet, während «Social Scene» (b) und «MASC[3]» (d) sowohl soziale (Augen, Mund, Kopf, Körper) als auch nicht-soziale statische Objekte umfassen.

Abbildung 2: Beispiel von vier verschieden gruppierten Interessensbereichen (AOIs)



Anmerkung: (Prillinger et al., 2023)

Im Rahmen eines kontrollierten experimentellen Settings erfolgt die Erfassung der Augenaktivitäten der Teilnehmenden während der Bearbeitung von visual Tasks mittels Eye

[1] Ein Stimulus wird in der Psychologie mit einem Reiz beschrieben, der eine Reaktion auslöst (Pfeifer et al., 1997)
[2] Englisch: Area of Interest, kurz AOI (Blascheck et al., 2017)
[3] «*Multimodal Approach to Studying Cognition*». Dieser Begriff wird häufig bei Eye Tracking-Studien verwendet und bezieht sich auf Methoden, die mehrere Arten von Daten und Analysetechniken integrieren, um kognitive Prozesse besser zu verstehen (Prillinger et al., 2023)

Tracking Geräten. Die dabei generierten Blickdaten dienen als Rohdaten, die für die statistische Analyse und Evaluation der bearbeiteten visuellen Aufgaben verwendet werden (Blascheck, et al., 2017).

2.1.2. Definition Scan Pfad, Fixation und Sakkade

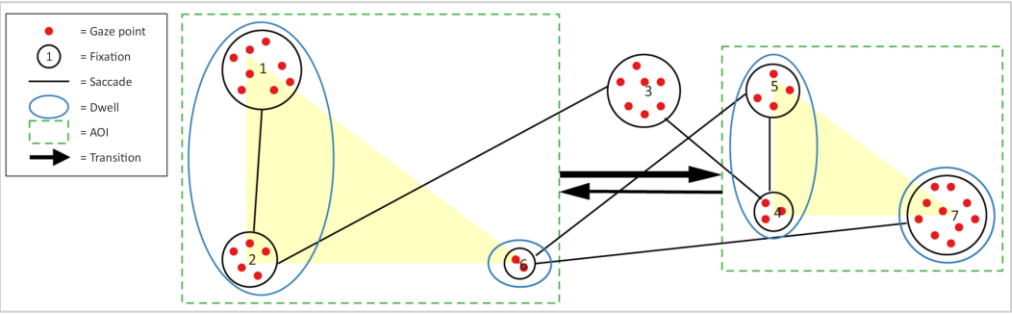
Die gesammelten Rohdaten müssen weiter aufbereitet werden, damit statistische Analysen bezüglich der Bearbeitungszeit und der Genauigkeitsrat bei der Ausführung eines visual Tasks erfolgen können. Der finale Eye Tracking Datensatz enthält schliesslich einen Scan Pfad von jedem Probanden, der in sogenannte Fixationen und Sakkaden unterteilt ist. Scan Pfade, Fixationen und Sakkaden sind die primären Kategorien von Eye Tracking Daten (Blascheck, et al., 2017).

Fixationen: Zeitintervalle, während derer die Augen stabil auf einem bestimmten Blickpunkt verweilen, werden als Fixationen bezeichnet. Während dieser Zeit nimmt das Auge detaillierte Informationen aus dem fixierten Bereich auf. Fixationen sind wichtige Indikatoren dafür, welche Teile eines visuellen Stimulus die Aufmerksamkeit einer Person anziehen (Blascheck et al., 2017).

Sakkaden: Das Springen von einem Fixationspunkt zum nächsten wird als Sakkade bezeichnet. Während dieser Zeit wird die visuelle Wahrnehmung vorübergehend unterdrückt. Das Gehirn verarbeitet während einer Sakkade kaum visuellen Informationen (Blascheck et al., 2017).

Scan Pfad: Die gesamte Abfolge von Fixationen und Sakkaden bilden einen Scan Pfad (Blascheck et al., 2017). Abbildung 3 zeigt eine schematische Darstellung eines Scanpfades, unterteilt in Fixationen und Sakkaden, sowie die Eingrenzung von Interessensbereichen. Die roten Blickpunkte werden zu einer Fixation zusammengefasst, da sie relativ konstant sind. Der Wechsel zum nächsten Fixationspunkt wird als Sakkade bezeichnet und ist als schwarze Linie eingezeichnet. Die Reihenfolge von einer Fixation zur nächsten definiert den Ablauf des Scan Pfades. Der definierte Interessensbereich innerhalb eines Stimulus ist in grün eingezeichnet.

Abbildung 3: Schematische Darstellung eines Scan Pfades



Anmerkung: (Blascheck, et al., 2017).

2.1.3. Motivation für die Erfassung von Eye Tracking Daten

Das Ziel der Erfassung von Augenbewegungen liegt darin, Einsichten in die kognitiven Prozesse der visuellen Aufmerksamkeit des Menschen zu gewinnen (Duchowski, 2007). Dafür wird der Blickverlauf von Testpersonen während einer gestellten visuellen Aufgabe aufgezeichnet. Im Folgenden werden aktuelle Erkenntnisse dargelegt, die durch die Untersuchung von Eye Tracking Daten gewonnen werden können:

- **Usability- und UX-Forschung:** In der Web- und Softwareentwicklung wird Eye Tracking verwendet, um zu analysieren, wie Nutzer mit einer Benutzeroberfläche interagieren und in welcher Reihenfolge und von welcher Dauer gewisse Bereiche betrachtet werden. Es ermöglicht die Identifikation von Usability-Problemen, wie etwa nicht wahrgenommene Elemente oder Bereiche, die potenziell zu Verwirrung führen, sowie die Ableitung weiterer Designentscheidungen. (usability.de GmbH & Co. KG, 2024).

- **Werbung und Marketing:** Angesichts der zunehmenden Bedeutung von Werbung in einer konsumorientierten Gesellschaft wird Eye Tracking zunehmen als Optimierungswerkzeug für Marketingmassnahmen eingesetzt. Beispielsweise lässt sich damit das Verhalten von Konsumenten bei der Auswahl und Suche von Produkten in den Regalen von Supermärkten analysieren (Wedel, Michel & Pieters, Rik, 2008).

- **Forschung:** Ein grosses Einsatzgebiet bildet die Forschung, insbesondere die Leseforschung. In diesem Bereich wird mittels Eye Tracking untersucht, wie lange Lesende auf bestimmten Wörtern oder Textabschnitten verweilen oder welche Rücksprünge gemacht werden. Dadurch werden tiefere Einblicke in kognitive Prozesse vermittelt, die es der Forschung ermöglicht, Unterschiede im Leseverhalten zwischen verschiedenen Personengruppen zu erkennen (Rakoczi, 2012).

- **Visualisierungswissenschaft:** Datenvisualisierungen besitzen nicht notwendigerweise eine aufschlussreiche Aussagekraft (Burch & Schmid, 2023). Die subjektive Wahrnehmungsfähigkeit der Betrachtenden spielt eine entscheidende Rolle, ob spezifische Aufgaben zuverlässig gelöst werden können oder ob zumindest Anhaltspunkte für die Identifikation gesuchter Datenmuster gegeben sind. Die Analyse von Blickdaten liefert Einblicke, wie unterschiedliche Visualisierungsformen – wie Diagramme, Grafiken oder schematische Karten – zur effektiven Informationsvermittlung beitragen (Burch & Schmid, 2023).

2.2. Visualisierungstechniken

Der Typ des Stimulus beeinflusst die Auswahl und Entwicklung von Eye Tracking Visualisierungstechniken massgeblich (Burch, 2022). Stimuli können dabei sowohl statische als

auch dynamische Eigenschaften aufweisen. Als statische Stimuli werden beispielsweise Bilder oder Textobjekte bezeichnet, deren Inhalte sich nicht verändern. Dynamische Stimuli hingegen zeichnen sich durch Interaktivität aus, wie beispielsweise Videos oder Szenen aus der realen Welt (Blascheck et al., 2017). Da diese Projektarbeit den Fokus ausschliesslich auf statische Bilder legt, werden im Folgenden Methoden zur Visualisierung vorgestellt, die speziell für nichtbewegte Stimuli geeignet sind.

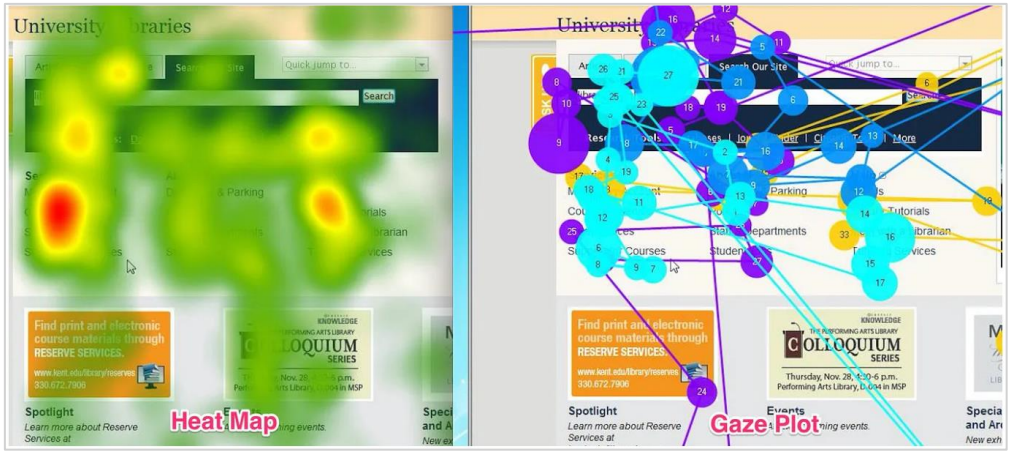
2.2.1. Point-based vs. AOI-based Methods

Die Analyse der in Eye-Tracking-Experimenten erhobenen Blickdaten erlaubt gemäss Burch (2017) die Unterscheidung zwischen punktbasierten (point-based) und Interessensbereich-basierten (AOI-based) Visualisierungstechniken. Die punktbasierte Auswertung befasst sich mit der Gesamtheit der Augenbewegungen sowie deren räumlicher und/oder zeitlicher Verteilung. Das heisst, die X- und Y-Koordinaten von Fixierungspunkten stehen im Fokus. Die zeitliche Abfolge der Sakkaden kann ebenfalls in die Visualisierung inkludiert werden. Die geläufigsten punktbasierten Visualisierungstypen sind Heat Maps und Scatterplots, die die Verteilung und Dichte der Fixationspunkte aufzeigen und zusätzlich auch Scanpfade visualisieren können (Blascheck et al., 2017).

2.2.2. Point-based Methods

Der aktuelle Forschungsstand legt nahe, dass Heat Maps und Gaze Plots zu den am häufigsten verwendeten Visualisierungstechniken für Eye Tracking Daten zählen (Tang, 2016) (Blascheck et al., 2017). Beide Techniken veranschaulichen, welche Bereiche eines Stimulus von den Probanden am intensivsten betrachtet werden. Heat Maps nutzen eine farbliche Kodierung, um die Verteilung der visuellen Aufmerksamkeit darzustellen. Dabei werden intensiv betrachtete Bereiche in Rot gekennzeichnet, während weniger beachtete Regionen in Blau oder Grün erscheinen. Gaze Plots hingegen repräsentieren die Fixationen durch Kreise, deren Grösse proportional zur Fixationsdauer ist. In diesem Sinne deuten grössere Kreise auf eine längere Verweildauer hin. Darüber hinaus visualisieren Gaze Plots die Reihenfolge der Blicksequenzen durch nummerierte Fixationspunkte und die Augenbewegungen (Sakkaden) durch verbundene Linien (Tang, 2016). Abbildung 4 zeigt ein Beispiel einer Heat Map und eines Gaze Plots.

Abbildung 4: Visualisierung einer Heat Map (links) und eines Gaze Plots (rechts)



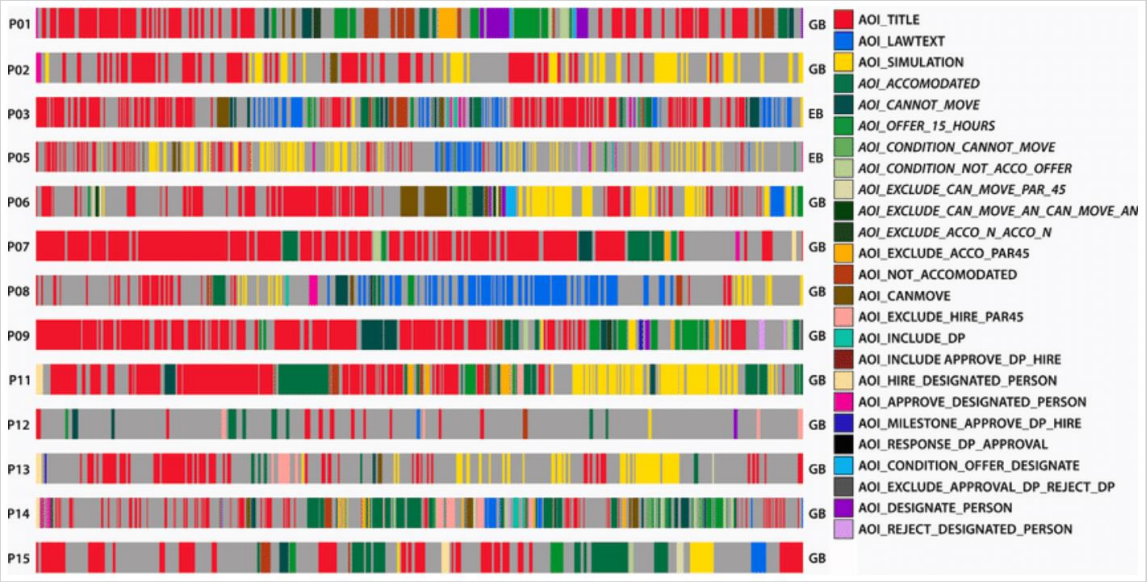
Anmerkung: (Tang, 2016)

Gemäss Burch (2017) sind Heat Maps eine Form von Attention Maps, die besondere Interessensbereiche hervorheben, während Gaze Plots als eine Art von Scan Pfad Visualisierungen gelten und mehr Aufschluss über den Blickverlauf einzelner Probanden liefern. Limitationen werden bei beiden Visualisierungsformen darin gesehen, dass die Überlagerung von mehreren Probanden zu einer visuellen Überlastung und folglich zu Verlust von Klarheit führt. Die Interpretation der überlagerten Daten wird komplex, da es schwierig ist, die Ergebnisse einzelner Probanden von der Gesamtdarstellung zu isolieren (Blascheck et al., 2017).

2.2.3. AOI-based Methods

Im Gegensatz zu den point-based Methoden arbeiten, gemäss Blascheck (2017), die AOI-based Methode mit definierten Interessensbereichen. Das heisst, es werden nur Daten in Bezug auf spezifische Regionen innerhalb eines Stimulus betrachtet. Dabei werden Metriken ausgewertet, die Auskünfte über die Verweildauer oder Anzahl der Fixationen im festgelegten AOI geben (Blascheck et al., 2017). Eine geläufige AOI-basierte Visualisierungstechnik ist gemäss Burch (2017) der Scarf Plot. Diese Darstellung wird verwendet, um die Sequenz und Dauer der Fixationen über verschiedene AOIs – Zeit dargestellt auf der X-Achse – von verschiedenen Probanden – untereinander dargestellt auf der Y-Achse – zu vergleichen. Abbildung 5 zeigt das Visualisierungsbeispiel eines Scarf Plots, der die Fixationssequenzen (X-Achse) verschiedener AOIs und Probanden (Y-Achse) zeigt, die eine definierte Aufgabe gelöst haben. Die AOIs sind farblich codiert.

Abbildung 5: Beispiel eines Scarf-Plots



Anmerkung: (ResearchGate, 2021)

2.3. Dashboard Design

Um die Eye Tracking Daten mit den angewandten Visualisierungstechniken zu verbinden und die Daten benutzbar zu präsentieren, spielen Dashboards eine entscheidende Rolle (Burch & Schmid, 2023). Dieses Kapitel untersucht den theoretischen Nutzen und beleuchtet die wesentlichen Designaspekte von Dashboards im Eye-Tracking-Kontext.

2.3.1. Definition Dashboard

Laut Roberto Capone (2015) wird der Begriff *Dashboard* synonym mit Cockpit verwendet und beschreibt ein Armaturenbrett oder eine Instrumententafel (Capone, 2015). Park und Jo (2015) führen den Begriff *Dashboard* auf seine historische Verwendung im Kontext von Pferdekutschen zurück, indem es ein Brett oder eine Platte an der Vorderseite der Kutsche bezeichnete, das dazu diente, Schlamm und Schmutz daran zu hindern, in den Innenraum zu gelangen (Park & Jo, 2015). Im Bereich der Automobilindustrie wird das Dashboard als Armaturenbrett verstanden, das primär als Informations- und Steuerungstafel konzipiert ist, um den Fahrer bei der Fahrzeugführung zu unterstützen (Malik, 2005). In der allgemeinen Geschäftswelt wird unter einem Dashboard ein Werkzeug verstanden, das zur Visualisierung und Analyse von Daten dient, um die Entscheidungsfindung durch die Bereitstellung übersichtlicher und aussagekräftiger Informationen zu unterstützen (Podgorelec & Kuhar, 2011). Das Ziel ist es, die wichtigsten Indikatoren auf einen Blick zu sehen, wenn die Leistung oder festgelegte Geschäftsziele nicht erreicht werden. Das exponentielle Wachstum des Datenvolumens und die Nutzung von Big Data in den letzten Jahrzehnten haben dazu geführt, dass Dashboards für Unternehmen immer komplexer und umfangreicher werden und in vielen Fachbereichen und Managementebenen nicht mehr wegzudenken sind

(Malik, 2005). Laut Malik wird jedoch die adressatengerechte Darstellung von Kennzahlen und Analyseergebnissen häufig vernachlässigt, da die Datenmenge zu gross und die zeitlichen Ressourcen zu gering sind. Interaktive Dashboards können die Akzeptanz und Nutzung verbessern, indem sie die Visualisierung so gestalten, dass die Empfänger die dargestellten Informationen präzise verstehen und korrekt interpretieren (Burch & Schmid, 2023).

Durch den Einfluss der Informationstechnologie und digitaler Geräte gewinnen Dashboards in der heutigen Arbeitswelt immer mehr an Bedeutung (Few, 2013). Stephen Few definiert den Begriff *Dashboard* wie folgt:

«Eine visuelle Darstellung der wichtigsten Informationen, die zur Erreichung eines oder mehrerer Ziele erforderlich sind, die auf einem einzigen Computerbildschirm konsolidiert werden, so dass sie auf einen Blick überwacht werden können» (Few, 2013, S. 26).

Daraus kann erschlossen werden, dass das Dashboard sowohl als Führungsinstrument als auch als Informationsinstrument eingesetzt werden kann.

2.3.2. Nutzen von Dashboards

Andreas Taschner weist darauf hin, dass Dashboards in der Praxis als grundlegende Dimension zur Entscheidungsfindung dienen, in der Wissenschaft hingegen vorwiegend als Analyseinstrument eingesetzt werden. Beides hat eine entscheidende gemeinsame Komponente: die Visualisierung (Taschner, 2015).

Die Visualisierung von quantitativen Informationen beeinflusst wesentlich die Wahrnehmung und Verarbeitung von Informationen. Durch die visuelle Darstellung mittels Grafiken können Informationen effizienter und aussagekräftiger kommuniziert werden als durch Text alleine (Burch & Schmid, 2023). Wichtig ist dabei, dass die Informationen auf einen Blick erfasst werden können (Podgorelec & Kuhar, 2011). Bildinformationen werden schneller wahrgenommen und bleiben länger in Erinnerung als Zahleninformationen ohne visuelle Komponente (Taschner, 2015).

Nach Stephen Few werden Dashboards je nach Geschäftsaktivität in drei Gruppen eingeteilt (Few, 2013):

- Dashboards, die sich auf die **Darstellung strategischer Ziele** konzentrieren. Dies beinhaltet hochrangige Kennzahlen und Darstellungen von Prognosen. Statistische und einfache Anzeigen ermöglichen einen schnellen Zugriff auf Informationen.
- Die zweite Gruppe unterstützt die **Datenanalyse**. Diese bietet eine Vielzahl von Vergleichen und eine übersichtliche und detaillierte Darstellung. Der Schwerpunkt liegt dabei auf der Leistungsbewertung.

- Die dritte Gruppe ist jene, welche die **Daten des operativen Ziels** in Echtzeit darstellt und hilft, Abweichungen in den Kennzahlen aufzuzeigen.

2.3.3. Designregeln für Datenvisualisierungen

Die Qualität einer Visualisierung hängt stark von der subjektiven Wahrnehmung und den Fähigkeiten des individuellen Betrachters ab (Burch & Schmid, 2023). Auch das Nutzerverhalten variiert, was bedeutet, dass beim Design eines Dashboards zur Datenexploration zunächst die Lesbarkeit und Funktionalität im Vordergrund stehen muss. Die Ästhetik bildet ebenfalls ein wichtiger Aspekt und beeinflusst nicht nur die Attraktivität, sondern auch die Benutzerakzeptanz. Gemäss Burch & Schmid (2023) stehen Funktionalität und Ästhetik oft im Gegensatz zueinander: Die Optimierung eines Aspekts kann zu Einschränkungen des anderen führen. Es gilt also beide Aspekte möglichst zu harmonisieren. Jonathan Schwabish bezieht sich in seinem Werk «Better Data Visuali-zations» insbesondere auf die nachfolgenden fünf Aspekte (Schwabish, 2021):

1) Schwabish betont, dass die primäre Funktion jeder visuellen Darstellung darin bestehen soll, die zugrunde liegenden **Daten klar und deutlich zu präsentieren**. Eine effiziente Datenvisualisierung hebt relevante Informationen hervor, unterstützt fundierte Entscheidungsprozesse und vermeidet die Ablenkung durch unnötige dekorative Elemente. Ziel ist es, die Daten so darzustellen, dass ihre Bedeutung schnell und präzise erfasst werden können (Schwabish, 2021).
2) Ein zentrales Anliegen von Schwabish ist die **Reduktion visueller Unordnung** in Datenvisualisierungen. Überflüssige Elemente und Verzierungen sollten vermieden werden, um die kognitive Belastung des Betrachters zu minimieren (Schwabish, 2021).
3) Laut Schwabish ist die **Integration von Grafiken und begleitendem Text** entscheidend, um die Verständlichkeit der Visualisierungen zu maximieren. Erläuterungen, Titel und Legenden sollten so gestaltet werden, dass sie unmittelbar die dargestellten Daten unterstützen und ergänzen, anstatt separat oder unabhängig von den Visualisierungen zu stehen (Schwabish, 2021).
4) Schwabish warnt vor der Verwendung komplexer Diagramme, die **eine Vielzahl überlappender Datenreihen** enthalten – oft als «Spaghetti-Charts» bezeichnet. Solche Visualisierungen erschweren die Datenerfassung und Analyse, da sie die Klarheit und Verständlichkeit der dargestellten Informationen beeinträchtigen (Schwabish, 2021).
5) Schwabish empfiehlt, Visualisierungen **mit neutralen Farben**, insbesondere Grautönen, zu beginnen. Diese Methode dient dazu, visuelle Hierarchien zu entwickeln,

indem zusätzliche Farben gezielt eingesetzt werden, um wichtige Informationen hervorzuheben. Der Einsatz von Grautönen hilft, die visuelle Unterscheidungskraft zu verbessern und den Fokus auf die bedeutendsten Daten zu lenken (Schwabish, 2021).

Diese fünf Regeln basieren auf den Prinzipien von Edward Tufte, der als einer der Pioniere auf dem Gebiet der Datenvisualisierung gilt. Schwabishs Vorschläge reflektieren und erweitern Tufte's bewährte Ansätze zur effektiven Datenpräsentation aus dem Werk «The Visual Display of Quantitative Information» (1983).

3. Methodik

3.1. Analyse der Rohdaten und Preprocessing

Der erste Schritt, um ein Dashboard erstellen zu können, ist die Analyse und Verarbeitung der Rohdaten. Eine gründliche Datenanalyse ist notwendig, um aussagekräftige Dashboards zu erstellen. Insbesondere die Explorative Datenanalyse bietet eine fundierte Basis für die Datenverarbeitung und Visualisierung (Turkey, 1977).

Der vorliegende Datensatz wurde von Dr. rer. nat Michael Burch zur Verfügung gestellt und stammt aus seinen Forschungsarbeiten im Bereich Eye Tracking. Der Datensatz umfasst Eye Tracking Daten verschiedener Probanden einer Studie und beinhaltet die wesentlichen Variablen, die Informationen zur Analyse von Blickbewegungen und Fixationspunkten liefern. Im Folgenden werden die einzelnen vordefinierten Variablen (Burch, 2024) detailliert beschrieben, um ein umfassendes Verständnis der Datensatzstruktur und ihrer Bedeutung zu ermöglichen.

- **Timestamp:** Diese Variable gibt den genauen Zeitpunkt der Datenaufnahme an. Jeder Eintrag im Datensatz ist mit einem spezifischen Zeitstempel versehen, was eine präzise zeitliche Zuordnung der Blickbewegungen ermöglicht.
- **StimuliName:** Der Name des Stimulus, auf den der Proband blickt, wird hier angegeben. In diesem Fall bezieht sich der Stimulus Name auf eine Metrokarte, die während des Eye Tracking-Experiments präsentiert wurde.
- **FixationIndex:** Diese Variable nummeriert die Fixationen chronologisch. Jede Fixation erhält eine eindeutige Nummer, die die Abfolge der Blickbewegungen angibt.
- **FixationDuration:** Die Dauer, wie lange der Blick auf einem bestimmten Punkt verweilt, wird in Millisekunden gemessen und in dieser Variable erfasst. Dies ist ein wichtiger Indikator für die Aufmerksamkeit und Verarbeitung des betrachteten Stimulus.
- **MappedFixationPointX und MappedFixationPointY:** Diese Variablen geben die genaue Position der Fixation auf dem Stimulus in einem zweidimensionalen Koordinatensystem an. Sie sind essenziell für die räumliche Analyse der Blickbewegungen.
- **user:** Diese Variable identifiziert den Probanden, der die Daten generiert hat.
- **description:** Diese Variable bezieht sich auf den Stimulus und beschreibt dessen Eigenschaften, wie zum Beispiel ob die Metrokarte in Farbe (color) oder in Graustufen (grey) dargestellt wurde. Diese Information ist wichtig für die Analyse, da sie Unterschiede in der visuellen Verarbeitung je nach Darstellungsmodus aufzeigen kann.

- **CityMap:** Diese Variable ist eine der selbst generierten Variablen und enthält den Namen der Stadtkarte ohne die Dateiendung «.jpg». Dies erleichtert die Identifikation der spezifischen Karten, die in den Experimenten verwendet wurden.
- **City:** Auch dies ist eine generierte Variable und enthält den Namen der Stadt, jedoch ohne die Präfixe «S1» und «S2», die möglicherweise auf spezifische Szenarien oder Bedingungen hinweisen. Diese Bereinigung erleichtert die Analyse und Vergleichbarkeit der Daten.

Um die Analyse der Rohdaten weiter zu vertiefen, wurden zusätzliche Variablen aus den bestehenden Daten berechnet. Diese neuen Variablen bieten weiterführende Einblicke in das Blickverhalten der Probanden und ermöglichen eine detailliertere Analyse (Hastie et al., 2009). Im Folgenden werden diese neuen Variablen und ihre Berechnungsmethoden beschrieben:

- **Task Duration:** Diese Variable repräsentiert die Gesamtdauer der Fixationen eines Benutzers pro Karte. Sie wird berechnet, indem die Fixationsdauern für jede Karte und jeden Benutzer summiert werden. Diese Metrik gibt Auskunft über die Gesamtdauer, die ein Benutzer benötigt hat, um eine bestimmte Karte zu betrachten und zu verarbeiten.

Formel 1: Berechnung der Task Duration

$$Task\ Duration = \sum Fixation\ Duration\ [User, Karte]$$

Anmerkung: (Hastie et al., 2009)

- **Task Duration Kategorie:** Diese kategoriale Variable klassifiziert die Task-Dauer in zwei Kategorien:
  - K1 für Task-Dauern unter 10 Sekunden
  - K2 für Task-Dauern von 10 Sekunden oder mehr
  - Diese Kategorisierung hilft dabei, unterschiedliche Betrachtungsstrategien zu identifizieren und zu vergleichen (Schwellwert von 10 Sekunden basiert auf eigenen Annahmen).

Formel 2: Berechnung der Task Duration Kategorie

$$Task\ Duration\ Kategorie = \begin{cases} K1\ , wenn\ Task\ Duration\ \ unter\ 10\ Sekunden \\ K2, wenn\ \ Task\ Duration\ \ \text{über}\ 10\ Sekunden \end{cases}$$

Anmerkung: Eigene Definition

- **Saccade Length:** Diese Variable berechnet die Länge der Sakkaden, also die Wegstrecke zwischen zwei aufeinanderfolgenden Fixationspunkten. Die Berechnung erfolgt mittels der euklidischen Distanzformel zwischen den Koordinaten der Fixationspunkte.

Formel 3: Berechnung der Sakkaden Länge

$$Saccade\ Length = \sqrt{(X2 - X1)^2 + (Y2 - Y1)^2}$$

Anmerkung: (Hastie et al., 2009)

- **Number Fixation Points:** Diese Variable gibt die Anzahl der Fixationspunkte pro Benutzer und pro Karte an. Sie zählt die Fixationen, die ein Benutzer auf einer bestimmten Karte vorgenommen hat (reine Zählung im Datensatz).
- **Avg. Fixation Duration:** Diese Variable gibt die durchschnittliche Fixationsdauer pro Benutzer und pro Karte an. Sie wird berechnet, indem die Gesamtdauer der Fixationen durch die Anzahl der Fixationspunkte dividiert wird.

Formel 4: Berechnung der Average Fixation Duration

$$Avg.Fixation\ Duration = \frac{Anzahl\ der\ Fixation\ Points}{Summe\ der\ Fixation\ Duration}$$

Anmerkung: (Burch, 2024)

Diese zusätzlichen Variablen ermöglichen eine umfassendere Analyse der Eye Tracking-Daten und bieten wertvolle Einblicke in das visuelle Such- und Betrachtungsverhalten der Probanden. Sie bilden die Grundlage für weiterführende Untersuchungen und die Erstellung aussagekräftiger Dashboards (Burch, 2024).

3.1.1. Exploration der Daten

Gemäss Auftraggeber, Dr. rer. nat. Michael Burch, haben wir die folgende Ausgangslage bei den Daten:

Stimuli

- 24 Metrokarten à jeweils 2 Ausrichtungen (S1 und S2) pro Farbausprägung:
  → 48 Stimuli in Farbe vs. 48 Stimuli in Grau

Probanden

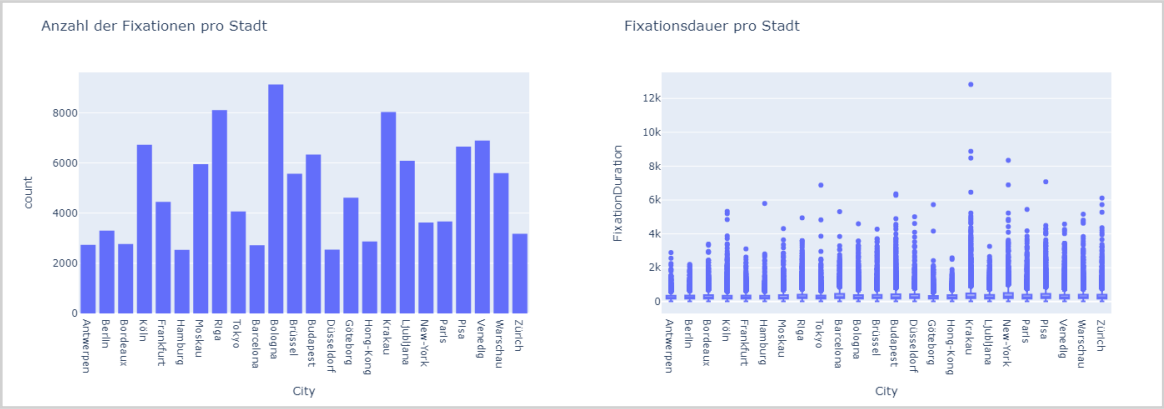
- 40 Probanden haben an der Studie teilgenommen
- Jeder Proband hat eine Kartenausprägung S1 in Farbe und eine Ausprägung S2 in Grau gesehen (oder umgekehrt): 24 Farbkarten vs. 24 Graukarten
- Jeder Stimulus zählt Eye Tracking Daten von 20 Probanden

3.1.2. Verteilung pro Stadt

Die Anzahl der Fixationspunkte sowie die Dauer der Fixationen zeigen erhebliche Unterschiede zwischen den Städten auf. Diese Variationen legen nahe, dass die Komplexität der

Karten in verschiedenen Städten deutlich variiert. Dies wird in Abbildung 6 verdeutlicht, worin die Anzahl Fixationspunkte (links) und die Summe der Fixationsdauer pro Stadt (rechts) dargestellt sind.

Abbildung 6: Überblick der Verteilung der Daten pro Stadt für beide Stimuli

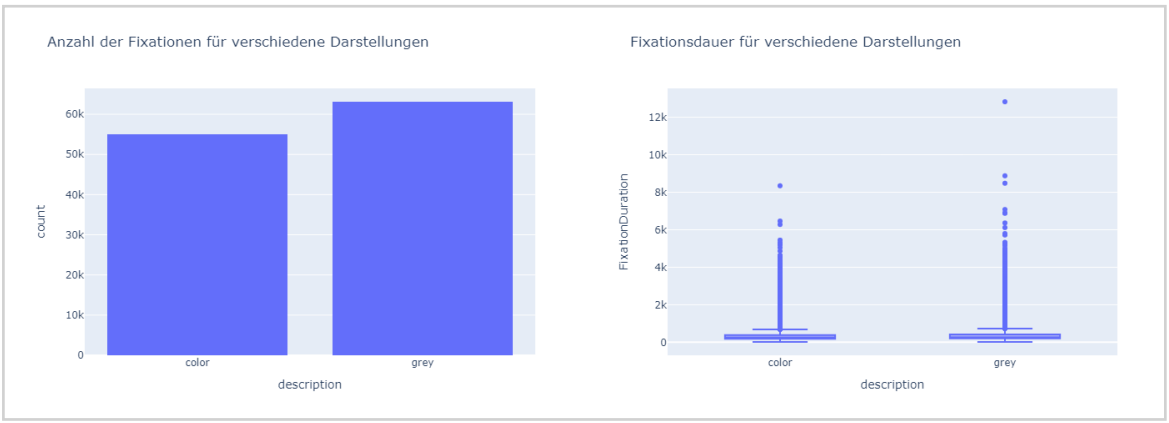


Anmerkung: Eigene Darstellung

3.1.3. Unterscheidung nach grau und farbigen Stimuli

Die Fixationsdauer und die Anzahl der Fixationen unterscheiden sich leicht je nach Stimulus, sei es schwarz/weiss oder farbig. Dies könnte darauf hinweisen, dass farbige Karten möglicherweise einfacher zu lesen sind. Es gibt jedoch kein eindeutiges Ergebnis in dieser Hinsicht, wie aus **Fehler! Verweisquelle konnte nicht gefunden werden.**Abbildung *7* zu entnehmen ist. Darin sind die Anzahl Fixation (links) und die Fixationsdauer pro Stimulus (rechts) dargestellt.

Abbildung 7: Überblick der Verteilung der Daten nach grau und farbigen Stimuli



Anmerkung: Eigene Darstellung

3.2. Analyse der Anspruchsgruppen

Die Entwicklung und Implementierung von Dashboards in Organisationen erfordert eine sorgfältige Analyse der Anspruchsgruppen (Stakeholder), um sicherzustellen, dass die Bedürfnisse und Erwartungen aller relevanten Parteien erfüllt werden. Diese Analyse ist

entscheidend, um Dashboards zu gestalten, die sowohl informativ als auch benutzerfreundlich sind, und um sicherzustellen, dass die bereitgestellten Daten den Entscheidungsprozess effektiv unterstützen (Raposo et al., 2022).

3.2.1. Identifikation der Anspruchsgruppen

Die erste Phase der Analyse besteht darin, alle potenziellen Anspruchsgruppen zu identifizieren, die von dem Dashboard profitieren könnten oder auf irgendeine Weise mit ihm interagieren werden (Freeman & Mcvea, 2001). Diese Gruppen können intern, wie Führungskräfte, Abteilungsleiter und Mitarbeiter, oder extern, wie Kunden, Partner und Investoren, sein. Die systematische Identifikation und Einbeziehung von Stakeholdern ist ein wesentlicher Bestandteil des strategischen Managements (Freeman & Mcvea, 2001).

3.2.2. Bedürfnisse und Anforderungen der Anspruchsgruppen

Nachdem die Anspruchsgruppen identifiziert wurden, besteht der nächste Schritt darin, ihre spezifischen Bedürfnisse und Anforderungen zu analysieren (Davis, 1989). Diese Analyse umfasst das Verständnis der Art und Weise, wie jede Gruppe die Daten nutzt, welche Informationen für sie am relevantesten sind und welche Form der Darstellung bevorzugt wird. Die Benutzerakzeptanz ist entscheidend für die erfolgreiche Implementierung von Informationssystemen, was durch die Berücksichtigung der Benutzerbedürfnisse erreicht werden kann (Davis, 1989).

3.2.3. Personalisierung und Anpassung

Basierend auf der Analyse der Bedürfnisse und Anforderungen können Dashboards personalisiert und angepasst werden, um spezifischen Anspruchsgruppen gerecht zu werden. Diese Anpassung kann die Auswahl relevanter Metriken, die Gestaltung benutzerfreundlicher Schnittstellen und die Integration von interaktiven Elementen umfassen. Benutzerzentrierte Designs verbessern die Effizienz und Zufriedenheit der Nutzer erheblich (Burmester, 2008).

3.3. Vom Wireframe zum Mockup

Die Entwicklung eines Dashboards ist ein mehrstufiger Prozess, der typischerweise mit der Erstellung eines Wireframes beginnt und schliesslich in einem detaillierten Mockup mündet (Few, 2013). Dieser Prozess stellt sicher, dass die Benutzeranforderungen und funktionalen Spezifikationen effektiv umgesetzt werden. Im Folgenden wird der Übergang vom Wireframe zum Mockup im Kontext von Dashboards detailliert beschrieben.

3.3.1. Wireframe-Erstellung

Ein Wireframe ist das erste visuelle Konzept eines Dashboards. Es dient als skizzenhafte Darstellung der grundlegenden Struktur und Layouts, ohne sich auf Design-Details wie

Farben, Schriftarten oder genaue Abstände zu konzentrieren (Raposo et al., 2022). Das Hauptziel eines Wireframes ist es, die Anordnung und Hierarchie der Inhalte festzulegen, um sicherzustellen, dass alle notwendigen Informationen und Funktionen logisch und benutzerfreundlich platziert sind (Raposo et al., 2022).

Ein Wireframe für ein Dashboard könnte folgende Elemente umfassen (Few, 2013):

- Platzierung von Diagrammen und Grafiken
- Navigationselemente und Filteroptionen
- Platz für Überschriften und Beschreibungen
- Grundlegende Anordnung der Datenvisualisierungen

3.3.2. Mockup-Erstellung

Ein Mockup ist eine detailliertere und realistischere Darstellung des endgültigen Dashboards. Es baut auf dem Wireframe auf und integriert detaillierte Designelemente wie Farben, Typografie, Bilder und spezifische Stilrichtlinien. Mockups vermitteln einen präziseren Eindruck davon, wie das fertige Produkt aussehen und funktionieren wird (Raposo et al., 2022). Sie sind entscheidend, um Stakeholdern eine klare Vorstellung des Endprodukts zu geben und frühzeitiges Feedback zu ermöglichen.

Ein Mockup für ein Dashboard könnte folgende verfeinerte Elemente beinhalten (Few, 2013):

- Detaillierte Farbgestaltung und Typografie
- Konkrete Datendarstellungen mit Beispielwerten
- Interaktive Elemente wie Buttons und Dropdown-Menüs
- Endgültige Platzierung und Skalierung aller visuellen Komponenten

3.3.3. Bedeutung des Übergangs vom Wireframe zum Mockup

Der Übergang vom Wireframe zum Mockup ist ein entscheidender Schritt im Dashboard-Designprozess. Er ermöglicht es Designteams, die Struktur und das Layout des Dashboards zu planen und gleichzeitig sicherzustellen, dass das Endprodukt visuell ansprechend und benutzerfreundlich ist (Raposo et al., 2022) (Few, 2013).

Klarheit und Kommunikation:

- Der Übergang vom Wireframe zum Mockup verbessert die Kommunikation zwischen Designern, Entwicklern und Stakeholdern, da Mockups klarer und verständlicher sind als Wireframes. Dies reduziert das Risiko von Missverständnissen und Fehlinterpretationen.

Feedback und Iteration:

- Detaillierte Mockups ermöglichen es den Stakeholdern, fundiertes Feedback zu geben, das in die weitere Entwicklung einfliesst. Diese Iterationen sind entscheidend, um ein benutzerzentriertes Design sicherzustellen.

Design-Verfeinerung:

- Während Wireframes die Struktur festlegen, helfen Mockups dabei, das visuelle Design zu verfeinern und sicherzustellen, dass das Dashboard ästhetisch ansprechend und funktional ist. Dies umfasst die Feinabstimmung von Farben, Schriften und anderen visuellen Details, um eine konsistente Benutzererfahrung zu gewährleisten.

Durch den Einsatz von Wireframes und Mockups können Designer und Entwickler effizient zusammenarbeiten, um ein Dashboard zu erstellen, das sowohl funktional als auch ästhetisch überzeugt (Raposo et al., 2022).

3.4. Dashboard Umsetzung mit Python Plotly

Die Entwicklung eines Dashboards mit Python und Plotly ist ein systematischer Prozess, der verschiedene Schritte umfasst, um sicherzustellen, dass die Daten korrekt verarbeitet und visuell ansprechend präsentiert werden (Burch, 2022). Dieser Leitfaden beschreibt die wesentlichen Schritte von der Datenverarbeitung bis zur Bereitstellung des Dashboards und stützt sich dabei auf Literatur und bewährte Methoden.

3.4.1. Datensatz einlesen und transformieren

Der erste Schritt besteht darin, die Rohdaten zu laden und zu transformieren, um sie für die Visualisierung vorzubereiten. Dies kann die Bereinigung der Daten, das Berechnen zusätzlicher Variablen und das Formatieren der Daten umfassen. Die Datenvorbereitung ein kritischer Schritt, der die Genauigkeit und Aussagekraft der nachfolgenden Analysen bestimmt (McKinney, 2017).

3.4.2. Dash

Dash ist ein Framework für die Erstellung von Webanwendungen mit Python. Es basiert auf Flask und Plotly und ermöglicht die Erstellung interaktiver Dashboards. Nach den Entwicklern von Plotly (Plotly, 2024) bietet Dash eine benutzerfreundliche Schnittstelle für die Entwicklung und das Hosting von Web-Apps.

3.4.3. Dash Core Components

Dash Core Components (dcc) sind grundlegende Bausteine für interaktive Elemente im Dashboard, wie Dropdown-Menüs, Schieberegler und Diagramme. Diese Komponenten ermöglichen die Erstellung dynamischer und interaktiver Benutzeroberflächen (Plotly, 2024)

3.4.4. Dash HTML Components

Dash HTML Components (html) bieten die Möglichkeit, HTML-Elemente wie Divs, Tabellen und Überschriften zu verwenden, um die Struktur und das Layout des Dashboards zu definieren. Diese Komponenten helfen dabei, eine saubere und logische Anordnung der Dashboard-Elemente zu gewährleisten (Plotly, 2024).

3.4.5. Cascading Style Sheets (CSS)

CSS wird verwendet, um das Erscheinungsbild des Dashboards zu gestalten. Es ermöglicht die Anpassung von Farben, Schriftarten und Layouts, um eine ansprechende Benutzeroberfläche zu schaffen. CSS ist ein wesentliches Werkzeug für die Gestaltung von Webanwendungen und bietet flexible Designmöglichkeiten (Meyer, & Weyl, 2023).

3.4.6. Plotly Express

Plotly Express ist eine einfache und schnelle Möglichkeit, interaktive Diagramme und Graphen zu erstellen. Es bietet eine High-Level-API, die die Erstellung von Visualisierungen mit wenigen Codezeilen ermöglicht. Plotly Express ist ideal für die schnelle Entwicklung von Visualisierungen, die in Dash-Apps integriert werden können (Plotly, 2024).

3.4.7. Callbacks

Callbacks sind Funktionen, die die Interaktivität im Dashboard ermöglichen, indem sie auf Benutzereingaben reagieren und die Ausgabekomponenten entsprechend aktualisieren. Callbacks sind das Herzstück der Interaktivität in Dash-Anwendungen und ermöglichen eine dynamische Datenaktualisierung (Plotly, 2024).

3.4.8. Deployen

Der letzte Schritt ist die Bereitstellung des Dashboards auf einem Server, damit es von anderen Benutzern im Web zugänglich ist. Dash-Apps können auf verschiedenen Plattformen wie Heroku, AWS und Google Cloud Platform bereitgestellt werden. Flask ist eine robuste Basis für das Hosting von Dash-Anwendungen und ermöglicht eine einfache Skalierung und Verwaltung (Grinberg, 2018).

4. Ergebnisse

Im vorangegangenen Kapitel wurde die Methodik des Projekts beschrieben. In diesem Kapitel soll nun das erstellte Eye Tracking Dashboard näher erläutert werden. Die folgenden Abschnitte geben einen umfassenden Überblick über den Anwendungsfall des Dashboards, wie es eingesetzt werden könnte, die Erstellung des Wireframes und das Design des Mockups. Des Weiteren wird die Entwicklung des Dashboards in Python beschrieben, sowie dessen Layout, Interaktionstechnik und Performance. Abschliessend wird auf das Dashboard näher eingegangen.

4.1. Use Case

In diesem Abschnitt wird erläutert, wie das Dashboard verwendet werden kann. Das entwickelte Dashboard zur Analyse von Eye Tracking Daten soll als vielseitiges Werkzeug für Forschende dienen, die sich mit Visualisierungswissenschaften beschäftigen. Es soll eine detaillierte Untersuchung und vergleichende Analyse von farbigen und graustufenbasierten Metrokarten verschiedener Städte ermöglichen. Daher ist es notwendig, dass der Vergleich zweier verschiedenfarbiger Karten im Dashboard sichtbar ist. Gemäss Auftraggeber, Michael Burch, muss ausserdem der Aspekt der Interaktionsmöglichkeit berücksichtigt werden. Das Dashboard soll dazu dienen, Daten zu explorieren und zu interpretieren. Ausserdem sollte es Platz auf einer Seite einnehmen, da es sonst zu unübersichtlich werden könnte (Tang, 2016) (Blascheck et al., 2017). Neben Visualisierungen sind auch statistische Kennzahlen, wie beispielsweise die Anzahl Fixationspunkte oder die durchschnittlichen Sakkaden Längen, zur Interpretation geeignet. Um den explorativen Nutzen zu erhöhen, sollten Filterfunktionen vorhanden sein, die eine Interaktion des Nutzers mit dem Dashboard ermöglichen. Gemäss dem vorhandenen Metrokartendatensatz der ist es möglich, eine Filterung nach Testpersonen, Städten und Visualisierungstyp darzustellen.

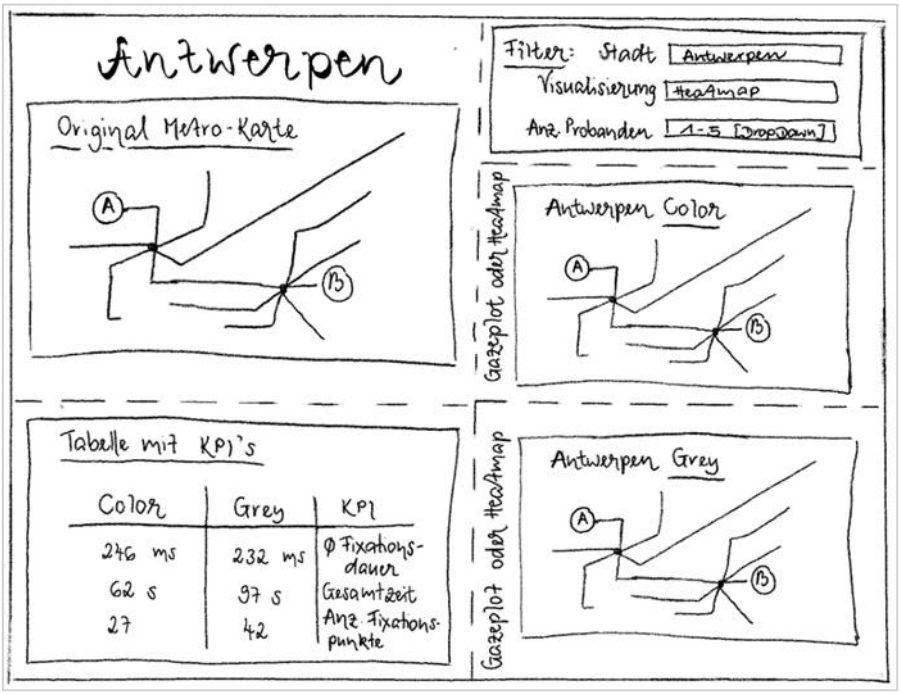
4.2. Wireframe

Wie bereits im vorigen Kapitel beschrieben, ist das Wireframe eine erste Darstellung des Mockups. Das Wireframe wurde zunächst als Handskizze angefertigt und anschliessend digital erfasst (siehe Abbildung 8). Als Beispielbild wurde die Metrokarte von Antwerpen verwendet. Fünf verschiedene Bereiche wurden gezeichnet. Der Hauptbereich, drei Nebenbereiche und ein Filterbereich. Im oberen linken Bereich, dem Hauptbereich, wurde die Kopfzeile (Header) mit dem Titel der ausgewählten Karte erstellt. Darunter befindet sich das zugehörige Originalbild.

Unterhalb des Hauptbereichs befindet sich der Nebenbereich, in dem die Tabelle mit den KPIs (Key Performance Indicators) angezeigt werden. Diese sollte die Spalte Color «grey» und KPIs anzeigen. In der Tabelle sollte die durchschnittliche Fixationszeit, die gesamte

Betrachtungszeit sowie die Anzahl der Fixationspunkte angezeigt werden. Dabei wurde zwischen Farbbilder (Spalte «color») und graustufen Bilder (Spalte «grey») unterschieden. Somit soll ein direkter Vergleich der beiden ausgewählten Karten ersichtlich sein. Oben rechts wurde der Filterbereich (Filter Panel) platziert. Dieser sollte die Option enthalten, die gewünschte Stadt anzuzeigen. Ausserdem sollte die Visualisierung ausgewählt werden können (Gaze Plot und Heat Map). Darunter befindet sich ein Dropdown zur Auswahl der gewünschten Testperson.

Abbildung 8: Darstellung des Wireframes (Handskizze)



Anmerkung: Eigene Darstellung

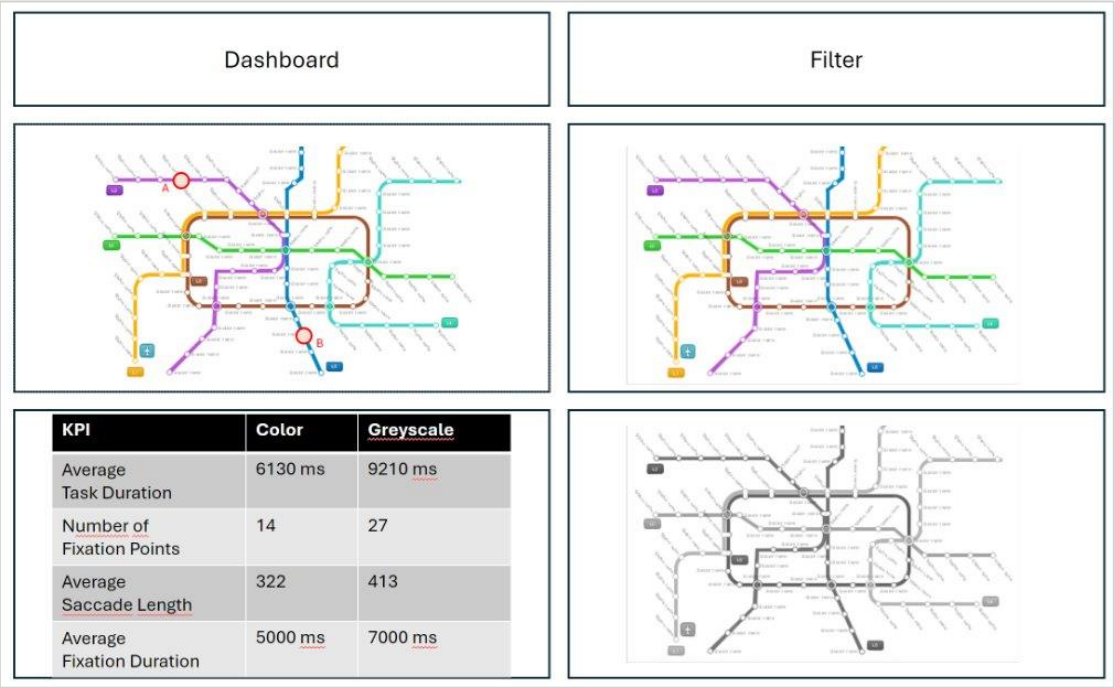
Unter dem Filterbereich befindet sich der dritte und vierte Teilbereich. Im oberen Teil sollte die Metrokarte in Farbe dargestellt werden. Darüber sollte die gewünschte Visualisierung projiziert werden. Dasselbe Prinzip unten rechts, aber mit der grauen Metrokarte. Die Idee ist es, einen direkten Vergleich zwischen dem farbigen und dem grauen Bild zu ermöglichen.

4.3. Mockup

Nach dem Wireframe wurde die erste Version des Mockups erstellt (siehe Abbildung 9). Die Struktur wurde dahingehend geändert, dass eine neue Kopfzeile bündig zum Filterbereich entstanden ist und die KPIs geändert wurden. Der Grund für diese Entscheidung ist, dass das Dashboard übersichtlicher gestaltet werden soll. Ausserdem wurden die Anzahl der Fixationspunkte und die durchschnittliche Sakkaden Länge hinzugefügt. Die Unterteilung von grauen und farbigen Bildern wurde beibehalten. Jedoch wurde die Sprache auf Englisch

geändert, da die verwendeten Fachbegriffe und technischen Ausdrücke in Englisch standardisiert sind.

Abbildung 9: Mockup Version 1



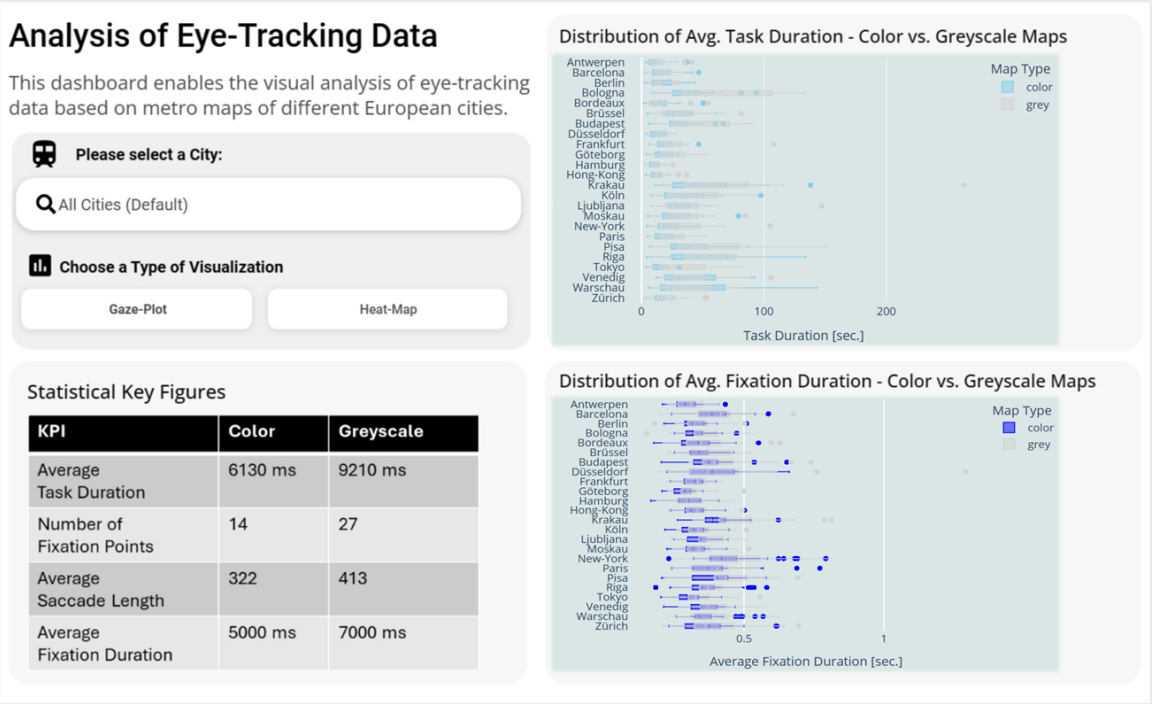
Anmerkung: Eigene Darstellung

Im weiteren Verlauf des Projekts wurden zusätzliche Fragen des Projektteams aufgeworfen. Beispielsweise dahingehend, welche Informationen der Nutzer zuerst sehen würde, nachdem die Daten initial geladen sind. Aus dieser Fragestellung heraus entstand das Mockup einer Landingpage (siehe Abbildung 10). Diese Einstiegseite soll dem Nutzer eine erste Übersicht des Dashboards geben. Darauf zu sehen ist, dass die Landingpage als Gitter-Layout dargestellt wird. Dabei wurde die Anzahl der Bereiche auf vier reduziert.

Der bisherige Hauptbereich mit dem Originalbild wurde entfernt und der Filterbereich in den Hauptbereich verschoben. Diese Änderung wurde vorgenommen, da sich das Originalbild bereits auf der rechten Seite des Dashboards befindet. Der neue Hauptbereich enthält nun eine Kopfzeile, die den Titel des Dashboards und eine kurze Beschreibung des Visualisierungstools enthält. Dies soll dazu dienen, der forschenden Person weitere Informationen über das Dashboard zur Verfügung zu stellen. Ausserdem wurde die Darstellung des Filters verbessert. So ist es nun möglich, eine bestimmte Stadt auszuwählen wobei standardmässig alle Städte ausgewählt sind. Die Auswahl der verfügbaren Visualisierungstypen wurde ebenfalls verbessert. Standardmässig ist keiner dieser Typen ausgewählt, sprich es werden keine zusätzlichen Visualisierungen angezeigt, bis eine Option ausgewählt wird. Ausserdem wurde jedem Bereich ein Titel hinzugefügt, um die Diagramme besser zu beschreiben.

Die Tabelle mit den wichtigsten KPIs im Vergleich zwischen Farb- und Graustufenkarten wurde nicht geändert.

Abbildung 10: Mockup Version 2 – Landing Page



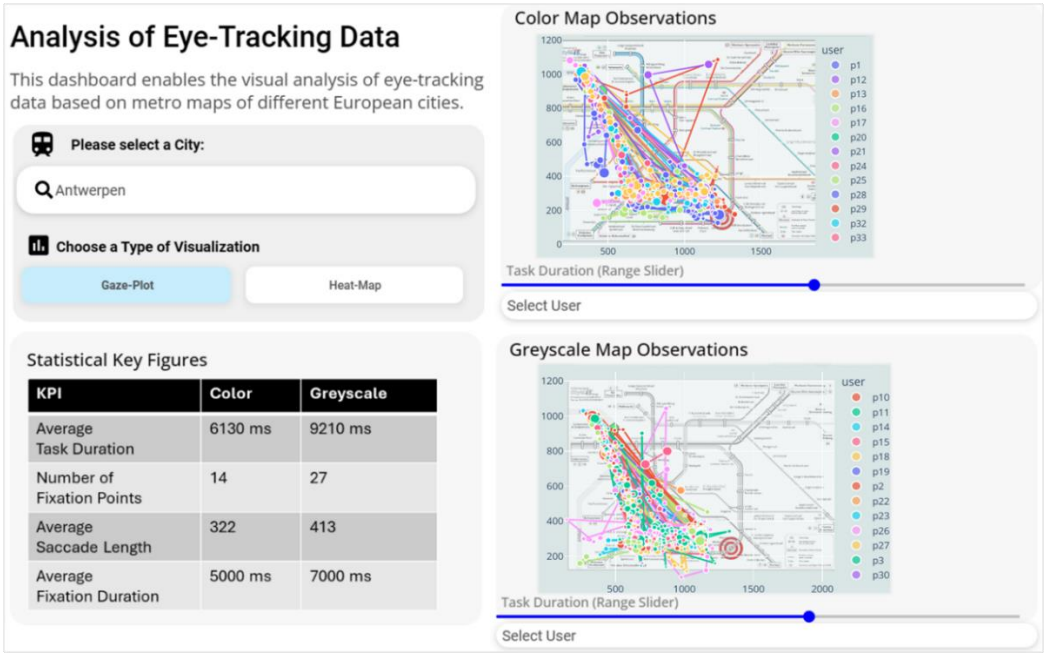
Anmerkung: Eigene Darstellung

Neben dem Filterbereich und der Tabelle befinden sich die neuen Boxplot-Visualisierungen. Der erste (oben rechts) zeigt die Verteilung der durchschnittlichen Dauer der Aufgaben für jede Stadt und vergleicht die Werte zwischen Farb- und Graustufenkarten. Diese sind farblich kodiert: Blau für farbige Karten und grau für graustufige Karten.

Der zweite Boxplot (unten rechts) zeigt die Verteilung der durchschnittlichen Fixationsdauer für jede Stadt und vergleicht ebenfalls farbige und graustufige Karten. Die Farbcodierung ist gleich wie beim ersten Boxplot.

Sobald eine Metrokarte und ein Visualisierungstyp ausgewählt wird, werden die beiden Boxplots durch den gewählten Visualisierungstypen ersetzt (siehe Abbildung 11). Der Titel der Visualisierung ändert sich und ein Schieberegler für die Dauer der Aufgabe ermöglicht es, die Ergebnisse zu filtern. Darüber hinaus gibt es ein Dropdown-Menü, mit dem bestimmte Testpersonen für eine detaillierte Analyse ausgewählt werden können.

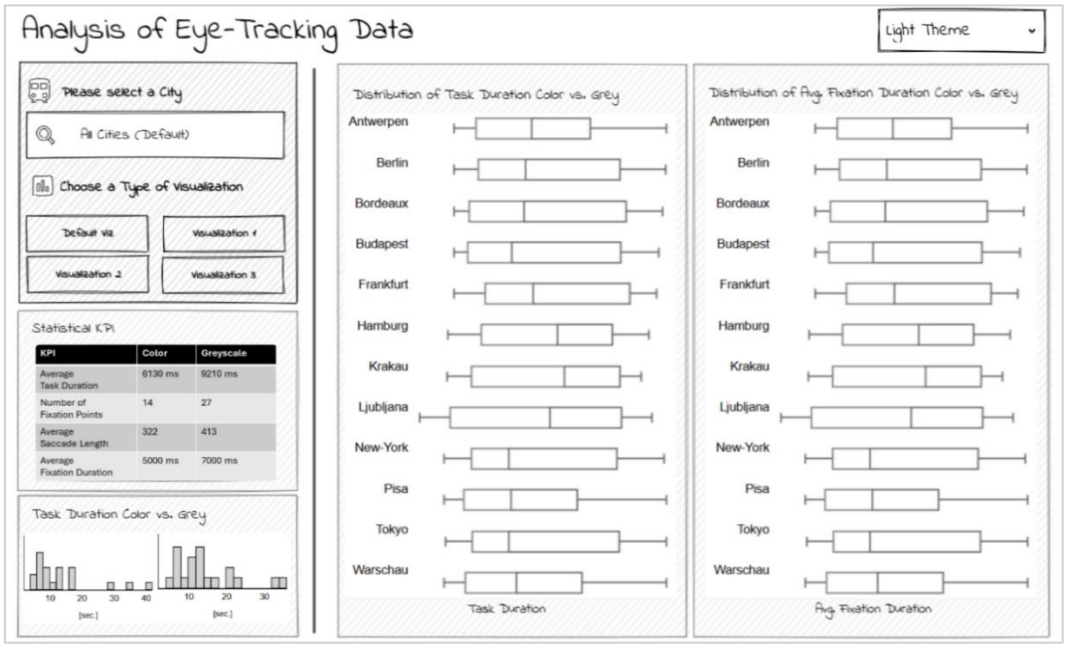
Abbildung 11: Mockup Version 2 – Detail Analyse der Stadt Antwerpen



Anmerkung: Eigene Darstellung

Im weiteren Verlauf des Projekts und aufgrund der Erkenntnisse aus den Modulen «Data Visualization» und «Dashboard Design» wurde das Mockup weiter verfeinert. Auf der Landingpage (siehe Abbildung 12) wurde die Option hinzugefügt, sich für ein helles oder dunkles Design zu entscheiden. Diese Entscheidung wurde getroffen, um den Nutzern die Möglichkeit zu geben, das Dashboard nach ihren persönlichen Präferenzen anzupassen und zu personalisieren. Des Weiteren wurde der Text unter dem Titel entfernt, um Redundanz zu vermeiden.

Abbildung 12: Finale Mockup Version – Landing Page

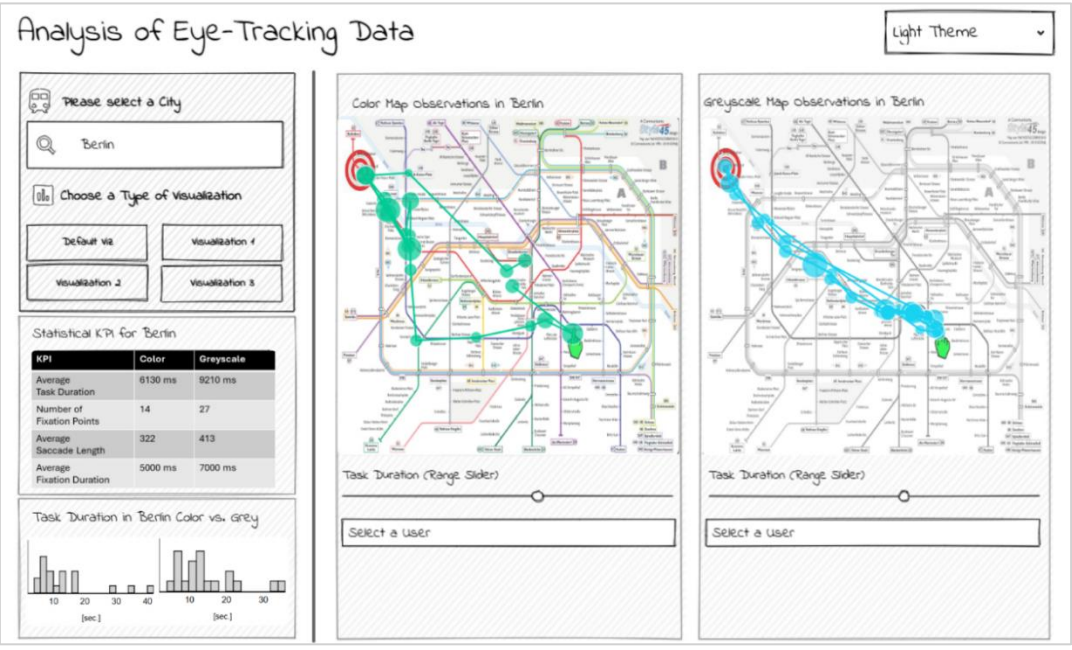


Anmerkung: Eigene Darstellung

Durch das Entfernen des Textes steht mehr Platz für zwei weitere Auswahlmöglichkeiten der Visualisierungsart zur Verfügung. Insgesamt kann zwischen vier Visualisierungstypen gewählt werden. Das Dropdown-Menü für die gewünschte Metrokarte bleibt erhalten. Unterhalb des Eingabebereichs wurde der KPI-Bereich um eine grafische Verteilung der Task Duration unterhalb der Tabelle erweitert. Dadurch bietet das Dashboard noch mehr Erkenntnisse. Rechts neben dem Input- und KPI-Bereich ist der Boxplot vertikal gedreht. Der linke Boxplot zeigt weiterhin die Dauer der Aufgaben aller Karten in Farbe und Grau. Daneben befindet sich der zweite Boxplot mit der durchschnittlichen Fixationsdauer ebenfalls aller Karten in Farbe und Grau. Durch die vertikale Drehung ist es möglich, die Task-Dauer und die durchschnittliche Fixationsdauer der Städte zu vergleichen.

Abbildung 13 zeigt den Visualisierungstypen «Gaze Plot» für die Stadt Berlin. Es ist ersichtlich, dass sich die Überschrift in allen Bereichen dynamisch anpasst. Im rechten Bereich des Dashboards ist nun die hinterlegte Metrokarte von Berlin in Farbe (links) und in Grau (rechts) zu sehen. Des Weiteren ist ein Range Slider für die Task Duration sichtbar, sowie ein Dropdown Menü für die Auswahl eines bestimmten Probanden. Somit ist die Interaktionsmöglichkeit des Dashboards für die Benutzer gegeben.

Abbildung 13: Finale Mockup Version – Detail Analyse einer Stadt



Anmerkung: Eigene Darstellung

4.4. Entwicklung mit Python

Das finale Mockup gemäss Abbildung 12 und Abbildung 13 im vorangehenden Kapitel bildet die Basis für den Python-Code. In diesem Kapitel wird erläutert, wie der theoretische Ansatz in die Programmiersprache übersetzt wurde. Der Hauptcode für das Dashboard liegt in der

Datei ‘app.py’ und die Designkomponente wurden in das CSS-File ‘custom.css’ ausgelagert. Der Python-Code ist in vier Sektionen gegliedert.

4.4.1. Datenimport

In einem ersten Schritt wird der Datensatz mit der Python Bibliothek Pandas als csv-File importiert. Anschliessend werden, wie in Kapitel 3.1 beschrieben, einzelne Kennzahlen berechnet und Aggregationen durchgeführt. Dieses Vorgehen ist in Codeausschnitt 1 dokumentiert.

Codeausschnitt 1: Datenimport mittels Pandas und Kennzahlenberechnung

```
"""
-----------------------------------------------------------------------------------------
Section 1:
Data Import and Preparation
"""
# Data reading:
data_path = 'assets/all_fixation_data_cleaned_up_3.csv'
df = pd.read_csv(data_path, sep=';')

# Add "Task Duration in sec" (per User and Stimulus) to df:
task_duration = df.groupby(['user', 'CityMap', 'description'])['FixationDuration'].sum().re-
set_index()
task_duration['FixationDuration'] = task_duration['FixationDuration'] / 1000
df = pd.merge(df, task_duration, on=['user', 'CityMap', 'description'], suffixes=('', '_aggre-
gated'))

# Add "Average Fixation Duration in sec" (per User and Stimulus) to df:
avg_fix_duration = df.groupby(['user', 'CityMap', 'description'])['FixationDura-
tion'].mean().reset_index()
avg_fix_duration['FixationDuration'] = avg_fix_duration['FixationDuration'] / 1000
df = pd.merge(df, avg_fix_duration, on=['user', 'CityMap', 'description'], suffixes=('',
'_avg'))

# Add Category for Task Duration:
df['TaskDurationCategory'] = pd.cut(df['FixationDuration_aggregated'], bins=[0, 10,
float('inf')], labels=['<10 sec.', '>=10 sec.'])
```

Anmerkung: Eigene Darstellung, Auszug aus File ‘app.py’

4.4.2. Layout

Grundsätzlich wurde ein Layout mit drei Spalten gewählt. Dabei liefert die erste Spalte sowohl Platz für Benutzereingaben (im oberen Bereich), wie auch für kleinere Visualisierungen als Output (im unteren Bereich). Die mittlere Spalte definiert den Output-Bereich für die farbigen Metrokarten und die dritte Spalte repräsentiert die graustufen Karten. Der zweiten und dritten Spalte wird mehr Platz zugeschrieben, da diese Visualisierungen im Fokus stehen. Codeausschnitt 2 zeigt die relevanten Stellen für die Definition des drei-Spalten-Layouts. Eine komplette Ausführung des Codes aus ‘custom.css’ ist im Anhang unter Codeausschnitt 3 zu finden.

Codeausschnitt 2: Definition des Spaltenlayouts

```
/* Split Layout in three columns */
.dash_container {
    display: flex;
    flex-direction: row;
    width: 90%;
    gap: 1px;
    box-sizing: border-box;
    padding-bottom: 10px;
}
/* Erste Spalte */
.first_column {
    flex: 0 0 25%;
    display: flex;
    flex-direction: column;
    box-sizing: border-box;
}
/* Zweite Spalte */
.second_column {
    flex: 0 0 37.5%;
    display: flex;
    flex-direction: column;
    box-sizing: border-box;
}
/* Dritte Spalte */
.third_column {
    flex: 0 0 37.5%;
    display: flex;
    flex-direction: column;
    box-sizing: border-box;
```

Anmerkung: Eigene Darstellung, Auszug aus File 'custom.css'

Innerhalb dieser drei Spalten wurden sieben Container definiert. Deren Anordnung ist in Abbildung 14 schematisch dargestellt.

1. Container – Header Area: Wird verwendet, um die Titelzeile zu gestalten. Die Elemente in der Zeile sind mit dem Parameter 'display = flex' definiert, um die Inhalt gleichmässig zu verteilen.

2. Container – Input City: Ist für das Dorpdown Menü der City Map vorgesehen. Grundsätzlich haben alle Bereiche einen Rahmen und abgerundete Ecken. Mittels den Parametern 'padding' und 'margin' werden Innen- und Aussenabstände festgelegt.

3. Container – Input Visualization Type: Ist für die Auswahl des Visualisierungstyps vorgesehen. Die Klasse 'button_viz_type' gestaltet die Anordnung der Click-Buttons und die Klasse 'viz_button' definiert deren Aussehen, einschliesslich Hover-Effekt und aktiven Zuständen.

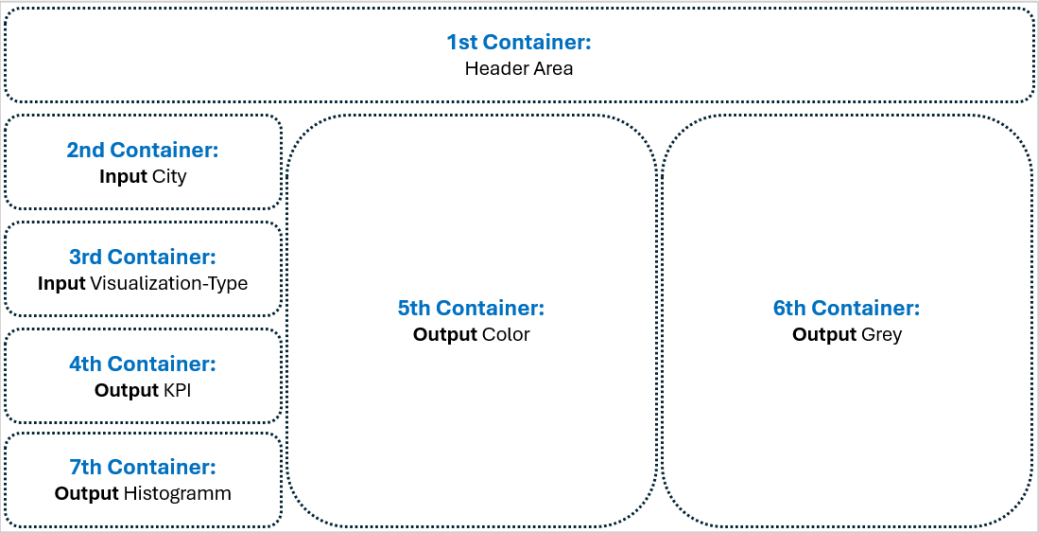
4. Container – KPI Table: Wird verwendet, um das Layout der KPI-Tabelle zu definieren.

5. Container – Color Map: Wird verwendet, um die Visualisierungen der farbigen Karten darzustellen. Um dem Visualisierungen so viel Platz wie möglich zu geben, wurden hier alle Innenabstände minimal gehalten.

6. Container – Grey Map: Ist analog dem fünften Container definiert und wird für die graustufen Visualisierungen verwendet.

7. Container – Histogramm: Wird verwendet, um ein Histogramm darzustellen. Diese Visualisierung ist weniger Platz eingeräumt, da sie mehr als Orientierung für den Output in den beiden Container 5 und 6 steht.

Abbildung 14: Anordnung der sieben Layout-Container



Anmerkung: Eigene Darstellung

Anhand dieser Container wurden die HTML-Komponente im Dash-Layout festgelegt. Der komplette Code zum Dash-Aufbau ist im Anhang unter Codeausschnitt 4 zu finden.

Damit die Designkomponente aus dem File ‘custom.css’ mit dem Dash-Layout verlinkt werden, muss im Python-Code eine Verlinkung stattfinden (Burch & Schmid, 2023). Dies wurde im Hauptprogramm ‘app.py’ mit folgender Codezeile umgesetzt:

```
app = Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP, '/assets/custom.css'])
```

4.4.3. Interaktionen

In der dritten Sektion werden die notwenigen Dash-Anwendungen konfiguriert, damit die Benutzeroberfläche dynamisch auf Benutzereingaben reagiert und die Visualisierungen und Filteroptionen korrekt angezeigt werden. In Codeausschnitt 5 im Anhang ist der komplette Code dazu zu finden. Grundsätzlich sind die folgenden Komponenten enthalten:

- **Interaktive Buttons:** Aktivierung und Speicherung des Click-Buttons für die Wahl des Visualisierungstyps.

- **Anzeigen von Visualisierungen:** Abhängig vom aktiven Button wird die entsprechende Visualisierung im Container 5 und 6 dargestellt. Die Steuerung erfolgt über eine CSS-Klassenänderung.
- **Dropdown-Filter:** Dynamische Aktualisierung der Dropdown-Filter für die Wahl der Probanden, in Abhängigkeit der gewählten Städtekarte (relevant bei Heat Map und Gaze Plot).
- **Range-Slider:** Dynamische Anpassung der Range-Slider für die Filterung nach 'Taskduration', in Abhängigkeit der gewählten Städtekarte (relevant bei Heat Map und Gaze Plot).
- **Theme-Mode:** Wechsel zwischen einem hellen und dunklen Theme-Mode, wobei die entsprechenden CSS-Klassen angewendet werden, um das Aussehen der Anwendung zu ändern.

4.4.4. Visualisierungen

In der vierten Sektion werden die eigentlichen Visualisierungen erstellt. Das Dashboard umfasst insgesamt zehn Visualisierungen. Da die Funktionen und Callbacks aller Visualisierungen einer allgemeinen Struktur folgen, wird an dieser Stelle nicht der komplette Python-Code beschrieben. Codeausschnitt 6 im Anhang zeigt ein Beispiel für die Visualisierung des Gaze Plots für farbige Karten. Der Codeaufbau umfasst die folgenden Schritte:

1. Die Benutzer interagieren mit einer Dash-Komponente, indem die Auswahl einer Städtekarte getroffen wird → der entsprechende Callback wird ausgelöst (Input).
2. Die Funktion verarbeitet die Inputs, filtert die Daten und erstellt oder aktualisiert die Visualisierung mit Plotly.
3. Das Ergebnis wird an die Output-Komponente zurückgegeben, welche im Dashboard aktualisiert wird → der entsprechende Callback wird ausgelöst (Output).

Die nachfolgenden zehn Visualisierungs-Funktionen sind im Code 'app.py' enthalten:

KPI-Tabelle (4.1): Erzeugt eine Tabelle der wichtigsten KPIs

Gaze Plot Color (4.2): Erstellt einen Gaze Plot für Farbkarten, der Fixationspunkte und Sakkaden Pfade ausgewählter Benutzer innerhalb einer Stadt darstellt. Der Plot wird auf einem Hintergrundbild dargestellt, das die jeweilige Stadtkarte zeigt.

Gaze Plot Grey (4.3): Gleich wie 4.2, jedoch für Graustufenkarten.

Density Heat Map Color (4.4): Erzeugt eine Heat Map für Farbkarten, die die Dichte der Fixationspunkte zeigt. Als Hintergrundbild erscheint die jeweilige Städtekarte.

Density Heat Map Grey (4.5): Gleich wie 4.4, jedoch für Graustufenkarten.

Box-Plot Taskduration (4.6): Erstellt ein Box-Plot, der die Verteilung der Taskduration über alle Städte und Kartentypen (Farbe und Graustufen) hinweg zeigt.

Box-Plot Average FixationDuration (4.7): Erstellt ein Box-Plot, der die Verteilung der durchschnittlichen Fixationsdauer pro Benutzer über alle Städte und Kartentypen hinweg darstellt.

Histogramm (4.8): Erzeugt ein Histogramm, das die Verteilung der Taskduration für eine ausgewählte Stadt oder alle Städte zeigt.

Scatter-Plot Color (4.9): Erstellt ein Streudiagramm zur Analyse der Korrelation zwischen Fixationsdauer und Sakkaden Länge für Farbkarten.

Scatter-Plot Grey (4.10): Gleich wie 4.9, jedoch für Graustufenkarten.

Abbildung 16 und 17 im Anhang zeigen die Visualisierungen entsprechend obiger Nummerierung im Code ‘app.py’, unterteilt nach globaler Analyse, sprich aggregiert über alle Städtekarten und nach Detail-Analyse am Beispiel Tokyo.

4.5. Eye Tracking Dashboard

Im vorangegangenen Kapitel wurde der Code-Aufbau mit Python beschrieben. Dabei wurden die einzelnen Komponenten des Dashboards individuell betrachtet. In diesem Kapitel widmen wir uns dem Aufruf des finalen Dashboards, was über zwei Optionen möglich ist.

4.5.1. Ausführung der Applikation ‘app.py’

Um die Applikation ‘app.py’ auszuführen, ist die Installation von Python Version 3.10 oder höher erforderlich. Zudem müssen folgende Packages installier sein, wobei der Import der Bibliotheken bereits im Code ‘app.py’ enthalten ist:

- from dash import Dash, dash_table, dcc, html, Input, Output, State, callback_context
- from dash_iconify import DashIconify
- from PIL import Image
- import dash_bootstrap_components as dbc
- import pandas as pd
- import plotly.express as px
- import glob
- import plotly.graph_objects as go

Die Applikation und für den Aufruf erforderlichen Mittel, wie Stylesheet ‘custom.css’, der Datensatz ‘all_fixation_data_cleaned_up’ sowie die Städtekarten in jpg-Format, sind Teil

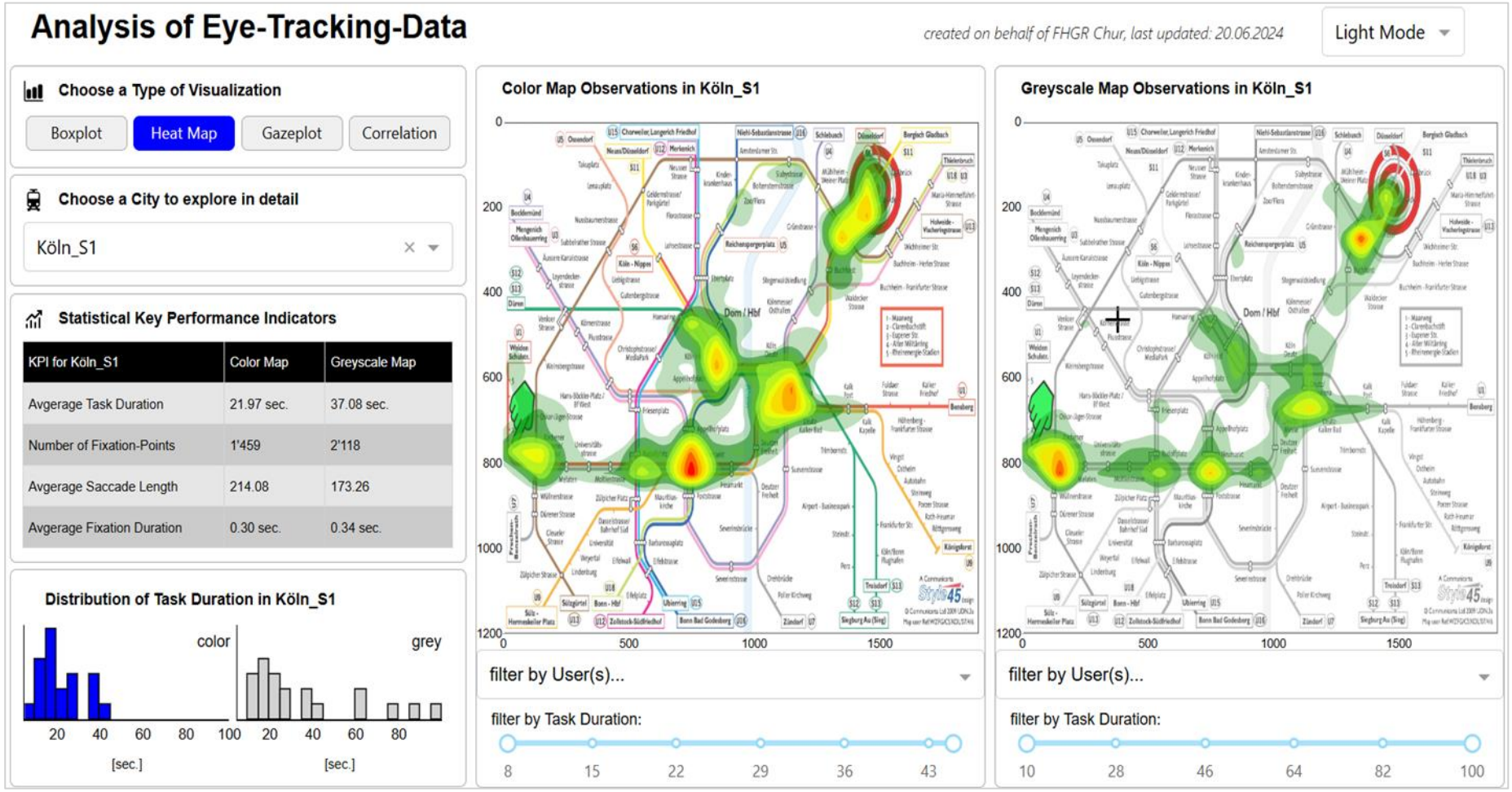
des Github-Repositorys <https://github.com/TamaraFHGR/Consultancy-Project-1.git>. Das Repository ist öffentlich zugänglich.

Sobald die Applikation ‘app.py’ gestartet wird, öffnet sich die Dash-Anwendung automatisch im lokalen Webbrowser mit dem URL http://127.0.0.1:8050/. Abbildung 15 zeigt das finale Dashboard, wie es beim Aufruf dargestellt wird.

4.5.2. Zugang über öffentlichen Server

Alternativ zu obigem Vorgehen kann das Dashboard über einen öffentlichen Server gestartet werden. Für das Deployen, wie in Kapitel 3.4.8 beschrieben, wurde die Plattform ‘pythonanywhere’ verwendet. Ohne weitere Installationen kann die Dash-Applikation über folgenden Link gestartet werden: https://tamarafhgr.pythonanywhere.com/.

Abbildung 15: Finales Dashboard



Anmerkung: Eigene Darstellung

5. Implikationen für die Praxis

Im vorhergehenden Kapitel wurden die Projektergebnisse und deren Entstehung erläutert. In diesem Kapitel geht es darum, den Praxisbezug näher zu beschreiben sowie Limitationen und Potenzial zur Weiterentwicklung aufzuzeigen.

5.1. Praxisbezug

Das entwickelte webbasierte Dashboard kann nun die gegebenen Eye Tracking Daten durch eine Vielzahl von Visualisierungsmöglichkeiten darstellen. Dadurch sollte es möglich sein, die Daten aus verschiedenen Blickwinkeln zu betrachten und zu analysieren. Durch den Einsatz von vier verschiedenen Visualisierungsarten kann ein Mehrwert geschaffen werden, um neue Erkenntnisse aus dem Dashboard zu gewinnen.

Der Boxplot auf der Landingpage bietet bereits einen ersten Überblick und ermöglicht Rückschlüsse auf die verschiedenen Städtekarten. Auffällig ist beispielsweise, dass der Median der Taskduration in Bologna bei der Graustufenkarte nahezu doppelt so hoch ist wie bei den farbigen Karten. Im Gegensatz dazu zeigen sich bei Hamburg kaum Unterschiede. Basierend auf solchen globalen Erkenntnissen, bietet das Dashboard nun die Möglichkeit, eine gewünschte Metrokarte detaillierter zu betrachten.

Durch die Auswahl der Heat Map wird die Dichte der Blickpunkte visualisiert. Es kann für die Forschenden entscheidend sein zu sehen, wie viel Fläche der gesamten Karte betrachtet wurde und wie stark sich die Blickpunkte auf den Ausgangs- und Zielpunkt konzentrieren. Darüber hinaus können durch die Heat Map auch Bereiche identifiziert werden, die häufig übersehen werden, was auf mögliche Probleme in der Kartengestaltung hinweisen könnte.

Mit Hilfe des Gaze Plots wird der Scan Pfad der Blickbewegungen untersucht. Für die Forschenden kann es aufschlussreich sein zu beobachten, welche Bereiche der Karte in welcher Reihenfolge betrachtet wurden. Dies hilft zu verstehen, wie Benutzer visuelle Informationen verarbeiten und welche Elemente der Karte zuerst ihre Aufmerksamkeit erregen. Der Gaze Plot zeigt auch, ob Benutzer einen logischen Pfad zwischen Ausgangs- und Zielpunkt verfolgen.

Mit der Korrelationsanalyse zwischen Sakkaden Länge und Fixationsdauer können zusätzliche Rückschlüsse gezogen werden. Eine globale Betrachtung über alle Städte hinweg zeigt, dass die Fixationsdauer bei den Graustufenkarten generell länger ist. Dies könnte darauf hinweisen, dass die Benutzer mehr Zeit benötigen, um Informationen auf den Graustufenkarten zu verarbeiten. Auf Detailsicht sieht es – um wieder auf das Beispiel von Bologna und Hamburg zurückzukommen – aber nicht durchgehend gleich aus.

Aus finanzieller Perspektive kann daher festgestellt werden, dass bestimmte Karten, die für die Betrachtenden weniger komplex sind, auch in Graustufen gedruckt werden können, ohne dabei wesentliche Einschränkungen in der Nutzererfahrung zu verursachen.

Grundsätzlich liefert das Dashboard ein Werkzeug, das eine Vergleichsanalyse von einer groben Übersicht bis hin zu detaillierten Einblicken ermöglich. Für den Praxisbezug ist jedoch stets entscheidend, welche Forschungsfrage mit den Analysen beantwortet werden soll. Unter diesem Aspekt zeigt das Dashboard einige Einschränkungen, die im nächsten Kapitel erläutert werden.

5.2. Limitation

Das aktualisierte Eye Tracking Dashboard bietet eine Vielzahl nützlicher Informationen und Visualisierungen, weist jedoch einige Einschränkungen auf, die berücksichtigt werden sollten, um die Benutzerfreundlichkeit und die Aussagekraft der Daten weiter zu verbessern.

Zum einen können die Informationen und Visualisierungen überwältigend wirken. Daher ist das Dashboard für Personen geeignet, die sich bereits mit dem Thema Eye Tracking beschäftigt haben.

Ein weiterer Punkt ist die statistische Aussagekraft. Das Dashboard zeigt zwar die Metriken an, es fehlen jedoch Angaben darüber, ob die Unterschiede zwischen den Gruppen (z.B. Farbkarten vs. Graustufenkarten) statistisch signifikant sind. Zum einen sind die in der Tabelle aufgeführten KPIs einfach gehalten, zum anderen gibt es keine statischen Auswertungen mehr. Damit ist auch keine tiefergehende Analyse möglich.

Da der Datensatz bereits im Dashboard integriert ist, besteht keine Möglichkeit, diesen auszutauschen oder anzupassen. Dies bedeutet, dass nur die Ersteller des Dashboards Änderungen an den Daten vornehmen können. Eine weitere Einschränkung liegt in der notwendigen Vorabaufbereitung der Daten, was die Austauschbarkeit des Datensatzes zusätzlich begrenzt.

5.3. Potenzial zur Weiterentwicklung

Im vorhergehenden Kapitel wurden die Grenzen des Dashboards aufgezeigt. Abschliessend werden die gewonnenen Erkenntnisse analysiert, um Entwicklungsmöglichkeiten aufzuzeigen.

Durch die Integration statistischer Signifikanztests könnte das Potenzial der Dashboards erweitert werden. Unterschiede in Farbausprägungen könnten somit nicht nur visuell, sondern auch basierend auf Berechnungen identifiziert werden. Mit diesem zusätzlichen Feature wäre das Dashboard in der Lage, den Anwendern weiterführende Schlussfolgerungen zu ermöglichen.

Generell könnte die Einbindung von weiteren Kennzahlen für Forschende im Eye Tracking Bereich von Interesse sein. Die Literaturrecherche in diesem Projekt hat gezeigt, dass der aktuelle Forschungsstand eine Vielzahl an Metriken bietet. Neben der Fixationsdauer könnte beispielsweise die Untersuchung der Rückkehrzeit – also die Zeit, die benötigt wird, um nach einer Fixation an einem anderen Ort zu einem vorherigen Bereich zurückzukehren – zusätzliche Erkenntnisse liefern (Brunyé et al., 2019).

Das Dashboard ist darauf ausgelegt, einen direkten Vergleich zwischen zwei Stimuli zu ermöglichen. Derzeit liegt der Schwerpunkt auf einem spezifischen Datensatz und analysiert die Farbausprägungen von Metrokarten. Es ist jedoch denkbar, dass das Dashboard auch für andere Vergleichszwecke genutzt werden könnte, wie beispielsweise die Analyse von Verkehrsdaten. Durch Korrelationsanalysen könnten hierbei Zusammenhänge aufgedeckt werden. Wichtig für die Verwendbarkeit ist, dass die Datensätze in ihrer Struktur ähnlich aufgebaut sind und über Zeit-Punkt-Daten verfügen (Zeitstempel und räumliche Koordinaten) (Burch, 2024). Diese Idee könnte noch weiter forciert werden, indem den Nutzern ermöglicht wird, ihre eigenen Datensätze für die Analyse in das Dashboard hochzuladen. Damit diese Funktionalität jedoch genutzt werden kann, ist es notwendig, eine detaillierte Anleitung bereitzustellen, welche die genauen Anforderungen an die Struktur der Daten beschreibt. Zudem sollte eine Schritt-für-Schritt-Checkliste erstellt werden, um sicherzustellen, dass das Dashboard fehlerfrei arbeitet. Darüber hinaus sind Anpassungen im Code erforderlich, um die Variablen neutral zu gestalten und nicht auf den aktuellen Anwendungsfall für Metrokarten zu beschränken.

Die Benutzerfreundlichkeit des Dashboards könnte durch erweiterte Filterfunktionen optimiert werden. Derzeit sind die Filtermöglichkeiten auf die Auswahl der Metrokarten, der Probanden und die Taskduration beschränkt. Zusätzliche Filter, die demografische Merkmale wie Alter oder Geschlecht einbeziehen, würden detailliertere Analysen ermöglichen und den Nutzern gestatten, spezifischere Forschungsfragen zu untersuchen. Es ist jedoch wichtig zu beachten, dass solche Informationen im Datensatz vorhanden sein müssen. Die Ausprägung und Qualität der Daten sind ausschlaggebend für die Aussagekraft des Dashboards.

Abschliessend könnte die Implementierung einer Berichtsfunktion für die Forschenden von zusätzlichem Nutzen sein. Dadurch könnten relevanten Erkenntnisse als Output als PDF oder in einem anderen gewünschten Format exportiert werden, um die Ergebnisse zu teilen oder zu präsentieren.

Diese Weiterentwicklungspunkte wurden vom Projektteam als valide Ansätze anerkannt, die in einem weiteren Studienprojekt möglicherweise weiterverfolgt werden können. Das Dashboard im aktuellen Zustand bietet dafür eine solide Grundlage.

6. Literaturverzeichnis

Blascheck, T., Kurzhals, K., Raschke, M., Burch, M., Weiskopf, D., & Ertl, T. (2017). Visualization of Eye Tracking Data: A Taxonomy and Survey. *Computer Graphics Forum*, *36*(8), 260–284. https://doi.org/10.1111/cgf.13079

Brunyé, T. T., Drew, T., Weaver, D. L., & Elmore, J. G. (2019). A review of eye tracking for understanding and improving diagnostic interpretation. *Cognitive Research: Principles and Implications*, *4*(1), 7. https://doi.org/10.1186/s41235-019-0159-2

Burch, M. (2022). *Eye tracking and visual analytics* (1st Aufl.). River Publishers.

Burch, M. (2024). *All_fixation_data_cleaned_up.csv* [CSV].

Burch, M., & Schmid, M. (2023). *Dashboard Design*. River Publishers.

Burmester, M. (2008). *Usability-Engineering*. https://link.springer.com/chapter/10.1007/978-3-540-69818-0_13

Capone, R. (2015). *Nachhaltiges Vertriebscontrolling für Elektrotechniker: Integrationsmöglichkeiten von BSC und Management Cockpit in die Vertriebs- und Marketingsteuerung*. Springer Fachmedien Wiesbaden GmbH.

Davis, F. (1989). *Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology*. https://www.jstor.org/stable/249008

Duchowski, A. T. (2007). *Eye tracking methodology: Theory and practice* (2. ed). Springer.

EyeSee Research. (2014, Mai 20). Eye Tracking Through History. *Medium*. https://medium.com/@eyesee/eye-tracking-through-history-b2e5c7029443

Few, S. (2013). *Information dashboard design: Displaying data for at-a-glance monitoring* (Second edition). Analytics Press.

Freeman, E., & Mcvea, J. (2001). *A Stakeholder Approach to Strategic Management*. University of Virginia. https://www.researchgate.net/publication/228320877_A_Stakeholder_Approach_to_Strategic_Management

Grinberg, M. (2018). *Flask Web Development*. O'Reilly Media, Inc. https://www.oreilly.com/library/view/flask-web-development/9781491991725/

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer. https://link.springer.com/book/10.1007/978-0-387-84858-7

Malik, S. (2005). *Enterprise dashboards: Design and best practices for IT*. Wiley.

McKinney, W. (2017). *Python for Data Analysis*. O'Reilly Media, Inc. https://www.oreilly.com/library/view/python-for-data/9781491957653/

Meyer, E., & Weyl, E. (2023). *CSS: The Definitive Guide*. O'Reilly Media, Inc. https://www.oreilly.com/library/view/css-the-definitive/9781098117603/

Park, Y., & Jo, I.-H. (2015). Development of the Learning Analytics Dashboard to Support Students' Learning Performance. *Journal of Universal Computer Science*, *Vol. 21*(No. 1).

Pfeifer, W., Braun, W., & Zentralinstitut für Sprachwissenschaft (Hrsg.). (1997). *Etymologisches Wörterbuch des Deutschen* (Ungekürzte, durchges. Ausg. der Taschenbuchausg., 2. Aufl). Dt. Taschenbuch-Verl.

Plotly. (2024). *Dash Documentation & User Guide | Plotly*. Plotly. https://dash.plotly.com/

Podgorelec, V., & Kuhar, S. (2011). Taking Advantage of Education Data: Advanced Data Analysis and Reporting in Virtual Learning Environments. *Electronics And Electrical Engineering*, *114*(8), 111–116. https://doi.org/10.5755/j01.eee.114.8.708

Prillinger, K., Radev, S. T., Amador De Lara, G., Werneck-Rohrer, S., Plener, P. L., Poustka, L., & Konicar, L. (2023). Effects of Transcranial Direct Current Stimulation on Social Attention Patterns and Emotion Recognition Ability in Male Adolescents with Autism Spectrum Disorder. *Journal of Clinical Medicine*, *12*(17), 5570. https://doi.org/10.3390/jcm12175570

Rakoczi, G. (2012). Eye Tracking in Forschung und Lehre. Möglichkeiten und Grenzen eines vielversprechenden Erkenntnismittels. In *Digitale Medien—Werkzeuge für exzellente Forschung und Lehre: Bd. Medien in der Wissenschaft* (Nummer 61, S. 87–98). Waxmann : Münster u.a. https://doi.org/10.25656/01:8301

Raposo, D., Silva, J., & Neves, João. (2022). *Perspectives on Design II*. https://link.springer.com/book/10.1007/978-3-030-79879-6

ResearchGate. (2021). *ResearchGate—Scarf-plot*. ResearchGate. https://www.researchgate.net/figure/Scarf-plot-showing-the-sequences-of-fixations-for-participants-solving-a-scenario-task_fig4_342713292

Schwabish, J. A. (2021). *Better data visualizations: A guide for scholars, researchers, and wonks*. Columbia University Press.

Tang, L. (2016, Dezember 5). Subtleties of Eyetracking Heat Maps and Gaze Plots. *Medium*. https://medium.com/@TheRealTang/subtleties-of-eyetracking-heat-maps-and-gaze-plots-a7ba4207f20f

Taschner, A. (2015). *Management Reporting und Behavioral Accounting: Verhaltenswirkungen des Berichtswesens im Unternehmen*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-07699-3

Turkey. (1977). *Exploratory Data Analysis* |. SpringerLink. https://link.springer.com/referenceworkentry/10.1007/978-0-387-32833-1_136

usability.de GmbH & Co. KG. (2024). *Eye Tracking—Usability-Test mit Eye Tracking*. usability.de. https://www.usability.de/leistungen/ux-testing-nutzerforschung/eyetracking.html

Wedel, Michel & Pieters, Rik. (2008). *A Review of Eye-Tracking Research in Marketing*. *Volume 4*, 123–147.

7. Hilfsmittelverzeichnis

Python Bibliotheken:

- dash
- dash_iconify
- PIL
- dash_bootstrap_components
- pandas
- plotly express
- glob
- plotly graph object

ChatGPT für die Fehlerfindung und Optimierung im Python Code

Fachliteratur gemäss Literaturverzeichnis in Kapitel 6

8. Anhang

Codeausschnitt 3: Komplette Layout Definition (‘custom.css’)

```
/* 1st Container - Header and Theme-Mode */
.first_container {
    width: 90%;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding-top: 3px;
    padding-bottom: 1px;
    padding-left: 20px;
    box-sizing: border-box;
}
/* Container für Header (H1 und H4) */
.header {
    display: flex;
    align-items: center;
    gap: 360px; /* Abstand zwischen H1 und H4 */
}
/* Dropdown für Theme-Mode */
.theme_dropdown {
    width: 220%;
    font-size: 14px;
}
/* ------------------- */
/* Split Layout in three columns */
.dash_container {
    display: flex;
    flex-direction: row;
    width: 90%;
    gap: 1px;
    box-sizing: border-box;
    padding-bottom: 10px;
}
/* Erste Spalte */
.first_column {
    flex: 0 0 25%;
    display: flex;
    flex-direction: column;
    box-sizing: border-box;
}
/* Zweite Spalte */
.second_column {
    flex: 0 0 37.5%;
    display: flex;
    flex-direction: column;
    box-sizing: border-box;
}
/* Dritte Spalte */
.third_column {
    flex: 0 0 37.5%;
    display: flex;
    flex-direction: column;
    box-sizing: border-box;
}
/* ------------------- */
/* 2nd Container - Input City */
.second_container {
    display: block;
    align-items: center;
    border: 1px solid darkgrey;
    border-radius: 5px;
    padding: 10px;
    gap: 2px;
    margin-left: 3px;
    margin-right: 3px;
    margin-top: 5px;
    box-sizing: border-box;
```

```
}
.dropdown {
    width: 100%;
    font-size: 14px;
    cursor: pointer;
}
.dash-dropdown .Select-placeholder {
    font-family: Arial, sans-serif;
    font-style: normal;
    font-size: 14px;
}
/* ------------------- */
/* 3rd Container - Input Visualization Type */
.third_container {
    display: block;
    align-items: center;
    border: 1px solid darkgrey;
    border-radius: 5px;
    padding: 10px;
    gap: 2px;
    margin-left: 3px;
    margin-right: 3px;
    margin-top: 5px;
    box-sizing: border-box;
}
.button_viz_type {
    display: flex;
    gap: 1px;
    justify-content: left;
    border-radius: 5px;
}
.viz_button {
    padding: 3px 3px;
    width: 80px;
    font-size: 12px;
    border: 1px solid darkgrey;
    border-radius: 5px;
    margin: 2px;
}
.viz_button:hover {
    background-color: #e0e0e0;
    cursor: pointer;
}
.viz_button.active {
    background-color: #0000FFFF;
    color: white;
    border: 1px solid blue;
}
/* ------------------- */
/* 4th Container - KPI-Table */
.fourth_container {
    display: block;
    align-items: center;
    border: 1px solid darkgrey;
    border-radius: 5px;
    padding: 10px;
    gap: 2px;
    margin-left: 3px;
    margin-right: 3px;
    margin-top: 5px;
    padding-bottom: 10px;
    box-sizing: border-box;
}
/* ------------------- */
/* 5th Container - Color Map */
.fifth_container {
    align-items: center;
    border: 1px solid darkgrey;
    border-radius: 5px;
    margin-left: 3px;
    margin-right: 3px;
```

```
    margin-top: 5px;
    box-sizing: border-box;
}
/* ------------------- */
/* 6th Container - Grey Map */
.sixth_container {
    align-items: center;
    border: 1px solid darkgrey;
    border-radius: 5px;
    margin-left: 3px;
    margin-right: 3px;
    margin-top: 5px;
    box-sizing: border-box;
}
/* 7th Container - Histogram */
.seventh_container {
    display: block;
    align-items: center;
    border: 1px solid darkgrey;
    border-radius: 5px;
    padding: 10px;
    gap: 2px;
    margin-left: 3px;
    margin-right: 3px;
    margin-top: 5px;
    box-sizing: border-box;
}
/* ------------------- */
/* Theme-Mode */
.light_theme {
    background-color: white;
    color: black;
}
.dark_theme {
    background-color: #454a59;
    color: white;
}
html, body {
    background-color: inherit;
    color: inherit;
}
.header, .first_container, .second_container, .third_container, .fourth_container,
.fifth_container, .sixth_container, .seventh_container, .dash_container {
    background-color: inherit;
    color: inherit;
}
/* Dropdown für Theme-Mode */
/* Überschreiben der Standardklasse von Dash */
.light_theme .dash-dropdown,
.light_theme .dash-dropdown .Select-control,
.light_theme .dash-dropdown .Select-menu-outer {
    background-color: white !important;
    color: black !important;
}
.dark_theme .dash-dropdown,
.dark_theme .dash-dropdown .Select-control,
.dark_theme .dash-dropdown .Select-menu-outer {
    background-color: #485260 !important;
    color: white !important;
}
/* Schriftfarbe für den ausgewählten Wert im Light Theme */
.light_theme .dash-dropdown .Select-placeholder,
.light_theme .dash-dropdown .Select-value-label {
    color: black !important;
}
/* Schriftfarbe für den ausgewählten Wert im Dark Theme */
.dark_theme .dash-dropdown .Select-placeholder,
.dark_theme .dash-dropdown .Select-value-label {
    color: white !important;
}
/* ------------------- */
```

```
/* Font Style */
h1 {
    font-size: 24px;
    font-family: Arial, sans-serif;
    font-weight: bold;
    text-align: left;
    /* margin-bottom: 12px; */
}
h2 {
    font-size: 16px;
    font-family: Arial, sans-serif;
    font-weight: normal;
    text-align: left;
    /* margin-bottom: 12px; */
}
h3 {
    font-size: 12px;
    font-family: Arial, sans-serif;
    font-weight: bold;
    text-align: left;
    /* margin-bottom: 12px; */
}
h4 {
    font-size: 12px;
    font-weight: lighter;
    font-style: italic;
    margin-top: 10px;
}
p {
    font-size: 12px;
    font-family: Arial, sans-serif;
    font-weight: normal;
    text-align: left;
    margin-top: 6px;
    margin-bottom: 1px;
    margin-left: 12px;
}
```

Codeausschnitt 4: Komplette HTML-Layout Definition (‘app.py’)

```
"""
-----------------------------------------------------------------------------------------
Section 2:
Definition of Dash Layout
"""
app.layout = html.Div([
    # Header and Theme-Mode:
    html.Div([
        html.Div([
            html.H1('Analysis of Eye-Tracking-Data'),
            html.H4('created on behalf of FHGR Chur, last updated: 20.06.2024')],
            className='header'),
        dcc.Dropdown(
            id='theme_dropdown',
            options=[
                {'label': 'Light Mode', 'value': 'light'},
                {'label': 'Dark Mode', 'value': 'dark'}],
            value='light',
            clearable=False,
            className='theme_dropdown',
        ),
        dcc.Store(id='current_theme', data='light'),
    ], className='first_container'),

    html.Div([
        # Start first column (Input, KPI, Histogram):
        html.Div([
            # Input-Containers:
            html.Div([
```

```
                # Visualization Type Buttons:
                html.Div([
                    html.H3([
                        DashIconify(icon="ion:bar-chart", width=16, height=16, style={"margin-
right": "12px"}),
                        'Choose a Type of Visualization']),
                    html.Div([
                        html.Button('Boxplot', id='default_viz', n_clicks=0, class-
Name='viz_button'),
                        html.Button('Heat Map', id='heat_map', n_clicks=0, className='viz_but-
ton'),
                        html.Button('Gazeplot', id='gaze_plot', n_clicks=0, class-
Name='viz_button'),
                        html.Button('Correlation', id='scatter_plot', n_clicks=0, class-
Name='viz_button'),
                    ], id='button_viz_type', className='button_viz_type'),
                    dcc.Store(id='active-button', data='default_viz'),
                    html.Div(id='output-section'),
                ], className='third_container'),
            ], className='input_container'),

                # City Dropdown:
                html.Div([
                    html.H3([
                        DashIconify(icon="vaadin:train", width=16, height=16, style={"margin-
right": "12px"}),
                        'Choose a City to explore in detail']),
                    dcc.Dropdown(
                        id='city_dropdown',
                        options=[{'label': city, 'value': city} for city in
sorted(df['CityMap'].unique())],
                        placeholder='Select a City Map...',
                        value=None,
                        clearable=True,
                        className='dropdown'),
                ], className='second_container'),

            # Output-Container KPI-Table:
            html.Div([
                html.H3([
                    DashIconify(icon="fluent:arrow-trending-lines-24-filled", width=16,
height=16,
                                style={"margin-right": "12px"}),
                    'Statistical Key Performance Indicators']),
                html.Div(id='table_container')
            ], className='fourth_container'),

            # Output-Container Histogram:
            html.Div([
                dcc.Graph(id='hist_taskduration'), #style={"height": "150px"}),
            ], className='seventh_container'),
        ], className='first_column'),

        # Start second column (Color Map):
        html.Div([
            # Output-Container Color Plot:
            html.Div([
                html.Img(
                    id='city_image_color'),
                dcc.Graph(id='gaze_plot_color'),
                dcc.Graph(id='heat_map_color'),
                dcc.Dropdown(id='dropdown_user_color', multi=True),
                dcc.RangeSlider(id='range_slider_color', min=1, max=50, step=1, value=[1, 50],
                                marks={i: f'{i}' for i in range(0, 51, 5)}),
                dcc.Graph(id='box_task_duration'),
                dcc.Graph(id='scatter_correlation_color')
            ], id='color_plot_area', className='fifth_container'),
        ], className='second_column'),

        # Start third column (Grey Map):
        html.Div([
```

```
            # Output-Container Grey Plot:
            html.Div([
                html.Img(
                    id='city_image_grey'),
                dcc.Graph(id='gaze_plot_grey'),
                dcc.Graph(id='heat_map_grey'),
                dcc.Dropdown(id='dropdown_user_grey', multi=True),
                dcc.RangeSlider(id='range_slider_grey', min=1, max=50, step=1, value=[1, 50],
                                marks = {i: f'{i}' for i in range(0, 51, 5)}),
                dcc.Graph(id='box_avg_fix_duration'),
                dcc.Graph(id='scatter_correlation_grey')
            ],  id='grey_plot_area', className='sixth_container'),
        ], className='third_column'),
    ], className='dash_container'),
], id='page_content', className='light_theme')
```

Codeausschnitt 5: Definition und Update von Interaktionselementen (‘app.py’)

```
"""
-----------------------------------------------------------------------------------------
Section 3:
Definition of Interaction Elements
"""
# 3.1 - Define and keep active Viz-Button:
@app.callback(
    [Output('default_viz', 'className'),
     Output('heat_map', 'className'),
     Output('gaze_plot', 'className'),
     Output('scatter_plot', 'className'),
     Output('active-button', 'data')],
    [Input('default_viz', 'n_clicks'),
     Input('heat_map', 'n_clicks'),
     Input('gaze_plot', 'n_clicks'),
     Input('scatter_plot', 'n_clicks')],
    [State('active-button', 'data')]
)

def update_active_button(btn1, btn2, btn3, btn4, active_btn):
    ctx = callback_context
    button_id = ctx.triggered[0]['prop_id'].split('.')[0] if ctx.triggered else 'default_viz'
    return [
        'viz_button active' if button_id == 'default_viz' else 'viz_button',
        'viz_button active' if button_id == 'heat_map' else 'viz_button',
        'viz_button active' if button_id == 'gaze_plot' else 'viz_button',
        'viz_button active' if button_id == 'scatter_plot' else 'viz_button',
        button_id
    ]

# 3.2 - Update Output Section and Plot Area based on active button, part I:
@app.callback(
    Output('output-section', 'children'),
    Input('active-button', 'data')
)
def update_output(active_button):
    if active_button == 'default_viz':
        return ''
    elif active_button == 'heat_map':
        return ''
    elif active_button == 'gaze_plot':
        return ''
    elif active_button == 'scatter_plot':
        return ''
    else:
        return ''

# 3.3 - Update Output Section and Plot Area based on active button, part II:
@app.callback(
    [Output('color_plot_area', 'children'),
     Output('grey_plot_area', 'children')],
```

```
    [Input('active-button', 'data'),
     Input('city_dropdown', 'value')]
)
def update_plot_area(visualization_type, selected_city):
    if visualization_type in ['gaze_plot', 'heat_map']:
        min_val_color, max_val_color, value_range_color, marks_color = up-
date_range_slider_color(selected_city)
        min_val_grey, max_val_grey, value_range_grey, marks_grey = up-
date_range_slider_grey(selected_city)

        return [
            dcc.Graph(id=f'{visualization_type}_color'),
            dcc.Dropdown(id='dropdown_user_color', value=None, multi=True, placeholder='filter
by User(s)...'),
            html.P('filter by Task Duration:'),
            dcc.RangeSlider(id='range_slider_color',
                            min=min_val_color, max=max_val_color, value=value_range_color,
marks=marks_color)
        ], [
            dcc.Graph(id=f'{visualization_type}_grey'),
            dcc.Dropdown(id='dropdown_user_grey', value=None, multi=True, placeholder='filter
by User(s)...'),
            html.P('filter by Task Duration:'),
            dcc.RangeSlider(id='range_slider_grey',
                            min=min_val_grey, max=max_val_grey, value=value_range_grey,
marks=marks_grey)
        ]
    elif visualization_type == 'scatter_plot':
        return [
            dcc.Graph(id='scatter_correlation_color')
        ], [
            dcc.Graph(id='scatter_correlation_grey')
        ]
    else:
        return [
            dcc.Graph(id='box_task_duration')
        ], [
            dcc.Graph(id='box_avg_fix_duration')
        ]

# 3.4 - Update Dropdown-Filters in plot area, based on selected city:
@app.callback(
    [Output('dropdown_user_color', 'options'),
     Output('dropdown_user_grey', 'options')],
    [Input('city_dropdown', 'value')]
)
def update_user_dropdowns(selected_city):
    if selected_city:
        # Filter users based on the selected city and description
        filtered_users_color = df[(df['CityMap'] == selected_city) & (df['description'] ==
'color')]['user'].unique()
        filtered_users_grey = df[(df['CityMap'] == selected_city) & (df['description'] ==
'grey')]['user'].unique()

        # Convert filtered users to dropdown options
        color_options = [{'label': user, 'value': user} for user in filtered_users_color]
        grey_options = [{'label': user, 'value': user} for user in filtered_users_grey]

        return color_options, grey_options

    return [[], []]

# 3.5 - Update Range-Slider in plot area, based on selected city and viz-type:
def to_int(value):
    return int(value) if not pd.isna(value) else 0

def update_range_slider(selected_city, description, buffer=5):
    if selected_city:
        filtered_df = df[(df['CityMap'] == selected_city) & (df['description'] == descrip-
tion)]
        if not filtered_df.empty:
```

```
            min_value = to_int(filtered_df['FixationDuration_aggregated'].min())
            max_value = to_int(filtered_df['FixationDuration_aggregated'].max())
            # Fester Wert von 2 zur Max-Grenze hinzufügen
            max_value_with_buffer = max_value + buffer
            marks = {i: f'{i}' for i in range(min_value, max_value_with_buffer + 1, max(1,
(max_value_with_buffer - min_value) // 5))}
            return min_value, max_value_with_buffer, [min_value, max_value_with_buffer], marks
        else:
            return 0, 0, [0, 0], {0: '0'}
    else:
        global_min = to_int(df['FixationDuration_aggregated'].min())
        global_max = to_int(df['FixationDuration_aggregated'].max())
        # Fester Wert von 2 zur globalen Max-Grenze hinzufügen
        global_max_with_buffer = global_max + buffer
        marks = {i: f'{i}' for i in range(global_min, global_max_with_buffer + 1, max(1,
(global_max_with_buffer - global_min) // 5))}
        return global_min, global_max_with_buffer, [global_min, global_max_with_buffer], marks

# 3.5.1 - Update color Slider:
def update_range_slider_color(selected_city):
    return update_range_slider(selected_city, 'color')

# 3.5.2 - Update grey Slider:
def update_range_slider_grey(selected_city):
    return update_range_slider(selected_city, 'grey')

# 3.5.3 - Callback for both Slider:
@app.callback(
    [Output('range_slider_color', 'min'),
     Output('range_slider_color', 'max'),
     Output('range_slider_color', 'value'),
     Output('range_slider_color', 'marks'),
     Output('range_slider_grey', 'min'),
     Output('range_slider_grey', 'max'),
     Output('range_slider_grey', 'value'),
     Output('range_slider_grey', 'marks')],
    [Input('city_dropdown', 'value')]
)
def update_range_sliders(selected_city):
    min_color, max_color, value_color, marks_color = update_range_slider_color(selected_city)
    min_grey, max_grey, value_grey, marks_grey = update_range_slider_grey(selected_city)
    return min_color, max_color, value_color, marks_color, min_grey, max_grey, value_grey,
marks_grey

# 3.6 - Update Theme-Mode based on selected theme:
@app.callback(
    [Output('page_content', 'className'),
     Output('current_theme', 'data')],
    [Input('theme_dropdown', 'value')]
)
def update_theme_mode(theme):
    if theme == 'light':
        return 'light_theme', 'light'
    else:
        return 'dark_theme', 'dark'

# 3.7 - Update Dropdown-Classname based on selected theme:
@app.callback(
    Output('city_dropdown', 'className'),
    [Input('current_theme', 'data')]
)
def update_dropdown_classname(current_theme):
    if current_theme == 'light':
        return 'dropdown light_theme_dropdown'
    else:
        return 'dropdown dark_theme_dropdown'
```

Codeausschnitt 6: Definition der Visualisierung Gaze Plot Color (‘app.py’)

```
"""
---------------------------------------------------------------------------------------------
Section 4:
4.2 - Definition of Scatter-Plot Color (Gaze-Plot)
"""
def get_image_path_color(selected_city):
    file_pattern_color = f'assets/*_{selected_city}_Color.jpg'
    matching_files = glob.glob(file_pattern_color)
    if matching_files:
        image_path = matching_files[0]
        img = Image.open(image_path)
        width, height = img.size
        return 'http://127.0.0.1:8050/' + image_path, width, height
    return None, None, None


@app.callback(
    Output('gaze_plot_color', 'figure'),
    [Input('city_dropdown', 'value'),
     Input('dropdown_user_color', 'value'),
     Input('range_slider_color', 'value'),
     Input('current_theme', 'data')]
)
def update_scatter_plot_color(selected_city, selected_users, range_slider_value, cur-
rent_theme):
    if selected_city:
        # Define a color map for users
        unique_users = df['user'].dropna().unique()
        colors = px.colors.qualitative.Plotly
        color_map = {user: colors[i % len(colors)] for i, user in enumerate(unique_users)}

        # Filter and sort data based on the selected filters (city and user):
        filtered_df = df[
            (df['CityMap'] == selected_city) & (df['description'] == 'color')]

        if selected_users:
            if isinstance(selected_users, str):
                selected_users = [selected_users]
            filtered_df = filtered_df[filtered_df['user'].isin(selected_users)]

        min_duration, max_duration = range_slider_value
        filtered_df = filtered_df[
            (filtered_df['FixationDuration_aggregated'] >= min_duration) & (filtered_df['Fixa-
tionDuration_aggregated'] <= max_duration)]

        # Extract Image Information and normalize data (only applicable for Antwerpen):
        image_path_color, width, height = get_image_path_color(selected_city)
        if image_path_color and width and height:
            # Attention: "Antwerpen_S1_Color" Data are not normalized !!!
            if selected_city == 'Antwerpen_S1':
                filtered_df['NormalizedPointX'] = (
                        filtered_df['MappedFixationPointX'] / 1651.00 * width)
                filtered_df['NormalizedPointY'] = (
                        filtered_df['MappedFixationPointY'] / 1200.00 * height)
            else:
                filtered_df['NormalizedPointX'] = filtered_df['MappedFixationPointX']
                filtered_df['NormalizedPointY'] = filtered_df['MappedFixationPointY']

            # Filter for fixation points within map only
            filtered_df = filtered_df[
                (filtered_df['NormalizedPointX'] >= 0) & (filtered_df['NormalizedPointX'] <=
width) &
                (filtered_df['NormalizedPointY'] >= 0) & (filtered_df['NormalizedPointY'] <=
height)]

        # Set title color based on theme
        title_color = 'black' if current_theme == 'light' else 'white'

        # Create scatter plot using the color map
```

```
        fig = px.scatter(filtered_df,
                         x='NormalizedPointX',
                         y='NormalizedPointY',
                         size='FixationDuration',
                         color='user',
                         color_discrete_map=color_map,
                         labels={
                             'MappedFixationPointX': 'X Coordinate',
                             'MappedFixationPointY': 'Y Coordinate',
                             'FixationDuration': 'FixationDuration (ms)',
                             'FixationDuration_aggregated': 'Task Duration (sec)'
                         },
                         hover_data = {
                                'user': True,
                                'MappedFixationPointX': True,
                                'MappedFixationPointY': True,
                                'FixationDuration': True,
                                'FixationDuration_aggregated': True
                            }
                         )

        # Add line traces for each user
        for user in filtered_df['user'].unique():
            user_df = filtered_df[filtered_df['user'] == user]
            fig.add_trace(
                go.Scatter(
                    x=user_df['NormalizedPointX'],
                    y=user_df['NormalizedPointY'],
                    mode='lines',
                    line=dict(width=2, color=color_map[user]),
                    name=f"Scanpath for {user}",
                    hoverinfo='skip'
                )
            )
        fig.update_xaxes(
            range=[0, width],
            autorange=False,
            showgrid=False,
            showticklabels=True,
            tickfont=dict(color=title_color, size=9, family='Arial, sans-serif'))

        fig.update_yaxes(
            range=[height, 0],
            autorange=False,
            showgrid=False,
            showticklabels=True,
            tickfont=dict(color=title_color, size=9, family='Arial, sans-serif'))

        fig.add_layout_image(
            dict(
                source=image_path_color,
                x=0,
                sizex=width,
                y=0,
                sizey=height,
                xref="x",
                yref="y",
                sizing="stretch",
                opacity=0.8,
                layer="below"
            )
        )

        fig.update_layout(
            plot_bgcolor='rgba(0, 0, 0, 0)',
            paper_bgcolor='rgba(0, 0, 0, 0)',
            xaxis_title=None,
            yaxis_title=None,
            title={
                'text': f'<b>Color Map Observations in {selected_city}</b>',
                'font': {
```

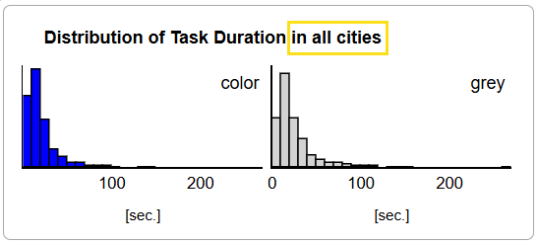
```
                    'size': 12,
                    'family': 'Arial, sans-serif',
                    'color': title_color }
            },
            margin=dict(l=0, r=5, t=40, b=5),
            showlegend=False,
            height=425)
        return fig

    else:
        fig = px.scatter()

        title_color = 'black' if current_theme == 'light' else 'white'

        fig.update_layout(
            plot_bgcolor='rgba(0, 0, 0, 0)',
            paper_bgcolor='rgba(0, 0, 0, 0)',
            title={'text': f"No City Map selected.<br><br>"
                           f"To display the <b>Scan Path Visualization</b> on a specific
map,<br>"
                           f"please select a city from the dropdown on the left.",
                   'y': 0.6,
                   'x': 0.5,
                   'xanchor': 'center',
                   'yanchor': 'middle',
                   'font': dict(
                       size=14,
                       color=title_color,
                       family='Arial, sans-serif')},
            showlegend=False,
            margin=dict(l=0, r=5, t=40, b=5),
            height=425
        )

        fig.update_xaxes(showgrid=False, zeroline=False, showline=False, showticklabels=False)
        fig.update_yaxes(showgrid=False, zeroline=False, showline=False, showticklabels=False)

        return fig
```

Abbildung 16: Visualisierungen der globalen Analyse

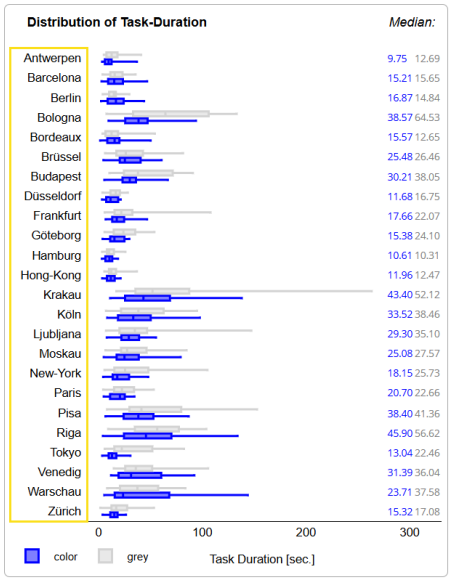
Statistical Key Performance Indicators

KPI for all cities	Color Map	Greyscale Map
Avgerage Task Duration	19.85 sec.	24.54 sec.
Number of Fixation-Points	55'009	63'119
Avgerage Saccade Length	167.70	147.05
Avgerage Fixation Duration	0.35 sec.	0.37 sec.

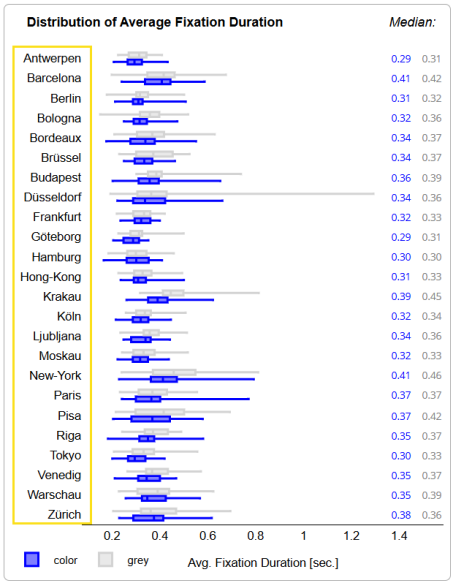
app.py – Sektion 4.1



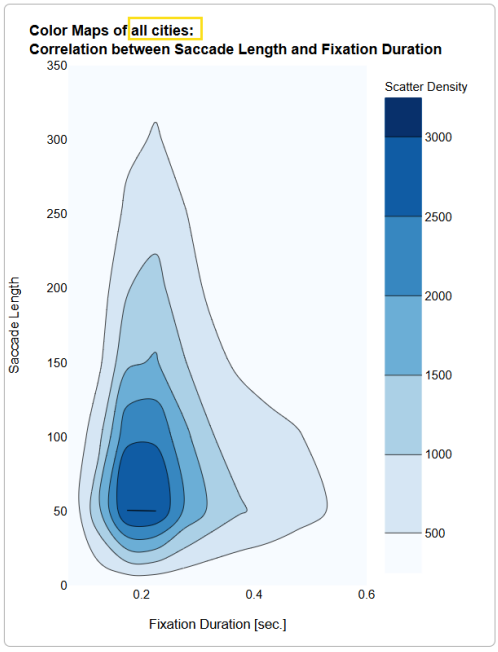
app.py – Sektion 4.8



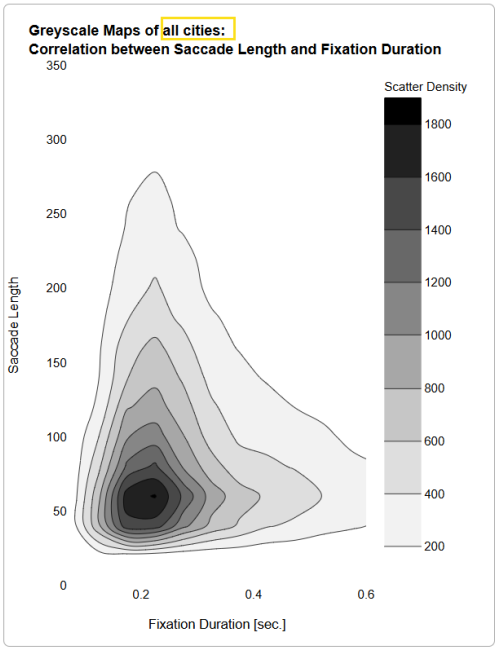
app.py – Sektion 4.6



app.py – Sektion 4.7

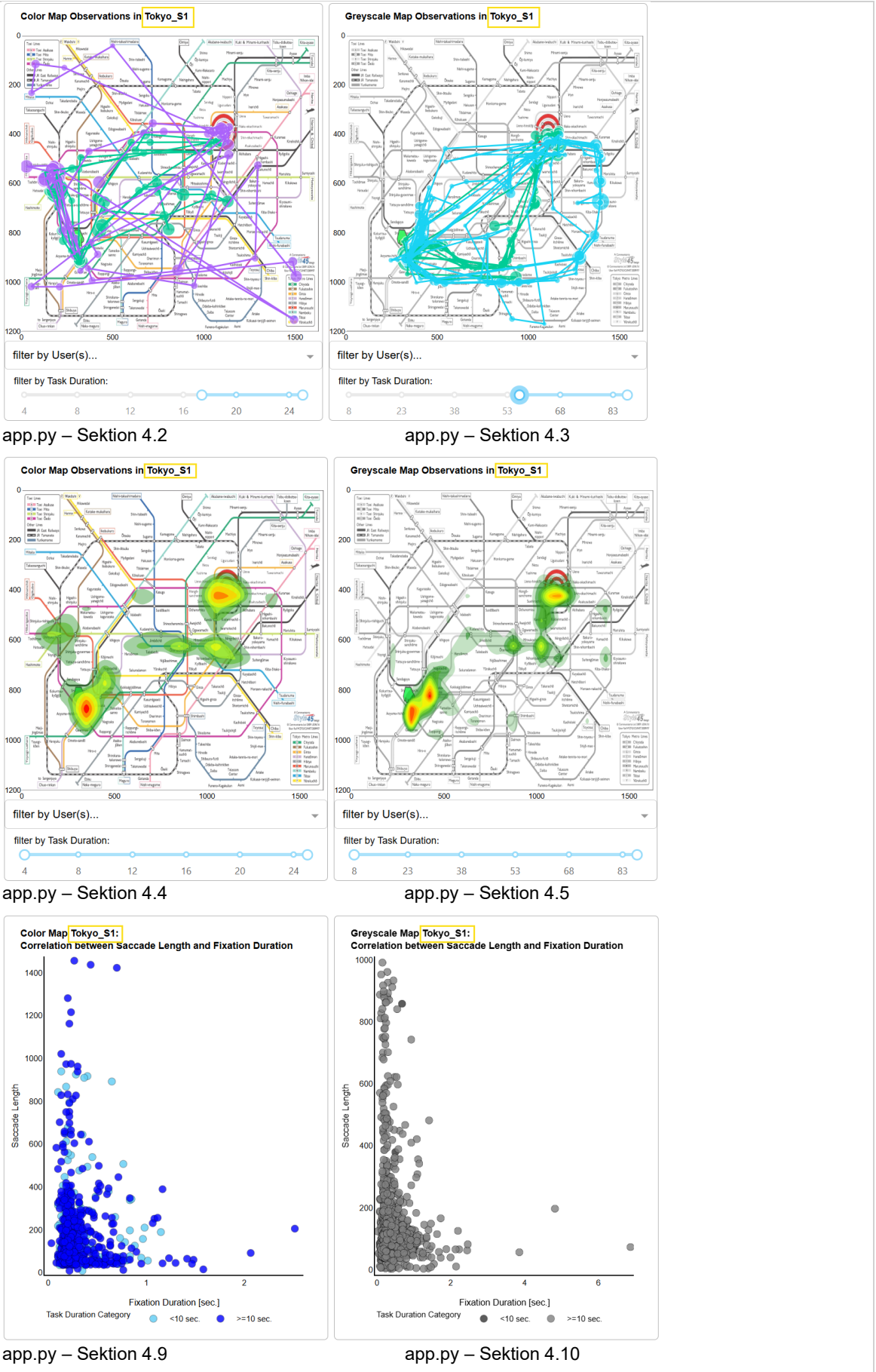


app.py – Sektion 4.9



app.py – Sektion 4.10

Abbildung 17: Visualisierungen der Detail-Analyse (Beispiel Tokyo)



Selbstständigkeitserklärung

Wir erklären hiermit, dass wir diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und erlaubten Hilfsmittel benutzt haben, einschliesslich der Verwendung von KI-Systemen. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen worden sind, haben wir als solche gekennzeichnet. Wir sind den Vorgaben des Leitfadens wissenschaftliches Arbeiten gefolgt. Uns ist bekannt, dass andernfalls die Hochschulleitung zum Entzug der aufgrund unserer Arbeit verliehenen Qualifikation oder des für unsere Arbeit verliehenen Titels berechtigt ist.

Winterthur, 14. Juli 2024

Sharon Reiser

Wetzikon, 14. Juli 2024

Serge Pellegatta

Büttikon, 14. Juli 2024

Tamara Nyffeler