Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3509

Neue visualisierungsbasierte Analysetechniken für Eye-Tracking-Daten

Dominik Herr

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Thomas Ertl
Betreuer/in:	Dipl.-Phys. Michael Raschke
Beginn am:	2013-06-12
Beendet am:	2013-12-12
CR-Nummer:	D.2.13, D.4.7, H.5.2

Kurzfassung

Aufgrund der zunehmenden Nutzung von Eye-Tracking-Systemen in Wirtschaft und Forschung nimmt der Bedarf an Analyse- und Visualisierungstechniken von Eye-Tracking-Daten zu. Diese Arbeit stellt einen auf den Konzepten und Techniken der Visual Analytics basierenden Ansatz zur Filterung und Weiterverarbeitung von Eye-Tracking-Daten vor. Hierbei werden zunächst die Daten vorverarbeitet. Im Anschluss kann der Studienanalysierende mittels konfigurierbarer Filter die Daten analysieren und im Anschluss visualisieren lassen. Die Visualisierung basiert dabei auf den Parallel Scan-Paths und unterstützt Interaktionsmöglichkeiten wie Brushing und Zoomen, um die Daten weiter explorieren zu können. Zu dem Konzept wird eine prototypische Implementierung vorgestellt, die am Ende dieser Arbeit in zwei Anwendungsszenarien evaluiert wird.

Abstract

Due to the increasing usage of eye tracking systems in the economical sector and academic research there is a rising demand for analyzation and visualization techniques for eye tracking data. This work applies the techniques and concepts of visual analytics by proposing a visualization-driven approach to analyze eye tracking data by filtering and analyzing the data before visualizing it. To do so, the data will be pre-processed. Afterwards the data can be analyzed by using configurable filters. The results are visualized using the Parallel Scan-Path visualization technique, enhanced by adding interaction methods such as brushing and zooming to enable further exploration of the data. The concept has been implemented as a prototype, which is evaluated in two scenarios.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1. Einleitung

Mit der zunehmenden Nutzung von Eye-Trackern in der Wirtschaft, im speziellen in der Marktforschung und in Teilen der Industrie, wie beispielsweise der Autoindustrie, in der man das Verhalten von Fahrern untersuchen will, und in der Forschung nimmt die Auswertung von Eye-Tracking Daten einen zunehmenden Stellenwert bei der Evaluation von Arbeitsabläufen und Qualitätsmessungen ein. Insbesondere in den Kognitionswissenschaften, der Visualisierung und der Mensch-Computer-Interaktion bzw. der Mensch-Maschine-Interaktion nimmt die Anzahl der durch Eye-Tracker gestützten Benutzerstudien zu.

Eines der größten Probleme des Eye-Trackings ist dabei die Auswertung der teilweise großen Datenmengen, die von Eye-Trackern erzeugt werden. Um die kognitiven Analysefähigkeiten des Menschen mit vollautomatisierten Filtertechniken aus der Informatik zu vereinen, wurde das Themenfeld der Visual Analytics begründet. Eine der Kernkomponenten, um die zunächst abstrakten Daten des Eye-Trackings einem Analysierenden dieser Daten zu vermitteln, ist dabei die Visualisierung.

Diese Arbeit befasst sich mit bereits bestehenden Visualisierungen für Eye-Tracking-Daten und wird auf Basis der Parallel Scan-Paths und verschiedenen Ansätzen und Techniken der Visual Analytics ein Konzept für ein interaktives System zur Exploration großer Eye-Tracking-Datensätze vorstellen. Dabei werden Techniken wie Filter und das „User in the loop“-Konzept, bei dem neben den automatischen Filtertechniken der Mensch mit Hilfe seines Verständnisses der Daten filtern kann, eingesetzt.

Hierzu werden zunächst die benötigten Grundlagen aus dem Bereich der Datenbanken, des Eye-Trackings, der Stringvergleiche, des Clusterings, der Informationsvisualisierung und der Visual Analytics vorgestellt.

Daraufhin wird auf das Szenario sowie die daraus entstehende Aufgabe und deren Lösungsansatz eingegangen.

Anschließend werden Arbeiten vorgestellt, deren Ziele ähnlich derer dieser Arbeit sind oder deren Techniken für das Erreichen des Ziels nützlich erscheinen.

Im Anschluss wird das Konzept dieser Arbeit vorgestellt. Hierzu wird zunächst eine Übersicht über das Konzept gegeben. Dann wird durch Anpassung der Konzepte der Visual Analytics ein Framework erzeugt, das im Konzept weiter ausgebaut wird. Hierzu werden die Schritte

- der Datenvorfilterung,
- der Analyse und
- der Visualisierung mit deren Interaktionsmöglichkeiten

vorgestellt.

Dabei sollen in der Datenvorfilterung die Eye-Tracking-Daten mit Areas of Interest verbunden werden und unnötige Daten verworfen werden, damit die spätere Datenanalyse schneller

abläuft. In der Analyse sollen die Daten bezüglich verschiedener Metriken gefiltert oder in Gruppen von Probanden unterteilt werden. Die Visualisierung mit deren Interaktionsmöglichkeiten sollen dem Studienanalysierenden die Möglichkeit bieten, die Daten detaillierter zu analysieren und zu explorieren, zum Beispiel indem bestimmte Daten ausgeblendet werden.

Auf das Konzeptkapitel folgt die Vorstellung der Implementierung des Prototyps. Hierbei wird zunächst auf die Architektur des Systems eingegangen. Daraufhin wird das für die Architektur genutzte Plug-in-System MEF kurz vorgestellt und dessen Einsatz verdeutlicht. Anschließend wird auf die Komponenten des Analyseschrittes sowie deren Vorteile und Einschränkungen eingegangen. Abschließend wird die Implementierung und Erweiterung der Parallel Scan-Paths vorgestellt.

Um die Nutzbarkeit des Konzepts und deren Implementierung zu zeigen, werden im Demonstrations- und Evaluationskapitel zwei Szenarien analysiert. Das erste Konzept basiert auf der Studie „Visual Elements III“ des Instituts für Visualisierung und interaktive Systeme der Universität Stuttgart. Das zweite Szenario ist die Aufnahme einer Testfahrt eines Fahrzeugs. Der Datensatz der Testfahrt wurde von der Abteilung R&D, Driver Analysis, der Daimler AG aufgezeichnet und zur Verfügung gestellt.

2. Grundlagen

Dieses Kapitel stellt grundlegende Techniken und Technologien vor, deren Verständnis für die Erfassung und die Lösung der Aufgabenstellung notwendig ist. Zuerst werden grundlegende Kenntnisse über Datenbanken und deren Verwendung (Abschnitt 2.1) vermittelt. Daraufhin werden geschichtliche Hintergründe und die Funktionsweise sowie die messbaren Metriken von modernen Eye-Trackern vorgestellt (Abschnitt 2.2). Daraufhin werden die gängigsten Visualisierungstechniken im Bereich des Eye-Tracking gezeigt (Abschnitt 2.3). Anschließend wird der Algorithmus zur Berechnung der Edit-Levenshtein-Distanz (Abschnitt 2.4) vorgestellt, der als Abstandsfunktion für die anschließend erläuterten Clusteringalgorithmen (Abschnitt 2.5) in dieser Arbeit eine zentrale Rolle spielt. Danach werden die Grundlagen der Informationsvisualisierung behandelt (Abschnitt 2.6). Zuletzt wird die Visual Analytics mit ihren Konzepten eingeführt (Abschnitt 2.7). In diesem Zusammenhang wird auch das erweiterte Visual Information Seeking Mantra vorgestellt.

2.1. Datenbanken

Heutzutage wird die Verwaltung von Daten häufig von Datenbanksystemen übernommen. Hierbei werden die Daten auf einem Server hinterlegt, damit die Daten zentral verwaltet werden können. Eine Datenbank ist eine Sammlung gespeicherter operationaler Daten, die von den Anwendungssystemen benötigt werden.

Beispielsweise nutzen Banken Datenbanksysteme, um monetäre Transaktionen durchzuführen. Eine Datenbank gewährleistet dabei beispielsweise im Falle einer Überweisung von einem Bankkonto zu einem anderen, dass ein Konto nur dann belastet wird, wenn das andere eine entsprechende Gutschrift erhält. Entweder die Überweisung war erfolgreich, in diesem Fall wurde das Geld überwiesen, oder sie schlug fehl und das Ausgangskonto bekommt den Überweisungsbetrag wieder gutgeschrieben.

2.1.1. Gründe für die Verwendung einer Datenbank

Gegenüber der lokalen Verwaltung haben Datenbanken mehrere Vorteile.

ACID: Die Zuverlässigkeit aller Transaktionen werden mittels der ACID (Atomicity, Consistency, Isolation, Durability)-Eigenschaften garantiert.

Atomicity bedeutet, dass eine logische Operation entweder erfolgreich ausgeführt wird oder vollständig annulliert wird. Die Operationen werden dabei so verwaltet, dass jede von außen als eine Transaktion sichtbar ist, die nicht von außen unterbrochen werden kann. Sollte die Transaktion dennoch fehlschlagen, so wird der ursprüngliche Zustand wiederhergestellt.

Consistency heißt, dass die Daten diversen Integritätsbedingungen genügen und dass die Daten normalisiert sind und somit, außer bei Fremd- und Primärschlüsseln, keine Redundanzen enthalten (siehe hierzu auch Abschnitt 2.1.2).

Isolation beschreibt die Isolation der Transaktionen dahingehend, dass nicht gleichzeitig auf Daten gelesen und geschrieben werden kann und somit Inkonsistenzen verhindert werden.

Durability garantiert, dass erfolgreiche Transaktionen dauerhaft in der Datenbank gespeichert werden. Diese Garantie gilt über ein Versagen der Soft- und Hardware hinaus (sofern nicht der Datenträger von dem Schaden betroffen ist).

High-Performance-Zugriffe: Datenbank-Management-Systeme (DBMS) sind stark optimierte Systeme, weshalb sie Daten sehr schnell bereitstellen können. Das liegt unter anderem daran, dass die Anfragesprachen so aufgebaut sind, dass diese sich in wenige logische Operationen auflösen lassen. Diese werden dann vom DBMS weiter vorverarbeitet und optimiert, bevor sie an die Datenbank weitergeleitet werden. Dort können sie dann sehr effizient und schnell verarbeitet werden. Zudem kann man bei Datenbanken Indizes anlegen, um die Abfrage von Daten noch schneller zu verarbeiten.

Verteilter Zugriff: Ein Datenbankserver kann ohne größeren Aufwand über ein Netzwerk, wie beispielsweise das Internet, zur Verfügung gestellt werden. Dort können dann mehrere unterschiedliche Clients und Anwendungen gleichzeitig auf die Datenbank zugreifen.

2.1.2. Relationale Datenbanken

Relationale Datenbanken sind in Tabellen unterteilt, die miteinander in Verhältnissen stehen können.

Tabellen: Tabellen sind Container, in denen die Daten der Datenbank abgelegt werden. Jede Tabelle hat einen Namen und verschiedene Eigenschaften, die deren Inhalt oder deren Relation zu anderen Tabellen beschreiben.

Tupel: Jede Zeile einer Tabelle repräsentiert einen Datensatz und wird Tupel oder Eintrag genannt.

Attribute: Die Spalten einer Tabelle nennt man Attribute. Jedes Attribut hat einen Namen und einen Datentyp, den es repräsentiert. Außerdem können Attribute zusätzliche Eigenschaften haben. Ein Beispiel hierfür wäre die Eigenschaft *Nullable*, die aussagt, dass ein bestimmtes Feld existiert, aber nicht immer einen Wert besitzen muss.

Primary Key (PK): Um ein Tupel in einer Tabelle zu identifizieren, benötigt man ein einzelnes oder mehrere Attribute, die zusammen die Eindeutigkeit eines Tupels sicherstellen. Diese Attribute nennt man Primary Keys. Wenn in einer Tabelle zwei Einträge identische Primary Keys haben, führt dies zu einem Fehler, da dies später eine Mehrdeutigkeit zur Folge hätte. Das widerspricht jedoch dem Konzept der Primary Keys.

Foreign Key (FK): Um Tabellen miteinander zu verknüpfen nutzt man so genannte Foreign Keys. Jeder FK ist eine Referenz auf ein Attribut in einer anderen Tabelle.

Beispiel

Wir haben zwei Tabellen: Person und Telefonnummer. Die Tabelle Person enthält ein Feld `id`, das vom Typ Integer ist und den PK repräsentiert. Zudem gibt es die Felder `Vorname` und `Nachname` als VarChar[1] Die Telefonnummern werden in der zweiten Tabelle verwaltet, da eine Person eventuell mehrere Nummern hat. Zudem gibt es ein Attribut `Geburtstag` vom Typ Date. Dieses ist *Nullable*, das bedeutet, dass der Wert nicht gesetzt werden muss. Die zweite Tabelle heißt `Telefonnummer` und hat ebenfalls einen Integer `id` als PK und einen VarChar `Telefonnummer`. Um auf die Tabelle `Person` zu verweisen gibt es zudem einen `FK FK_Person`, der auf den PK der Tabelle `Person` verweist. In Abbildung 2.1 wird das zugehörige Entity-Relation-Diagramm gezeigt. Tabelle 2.1 zeigt einige exemplarische Daten für eine solche Datenstruktur.

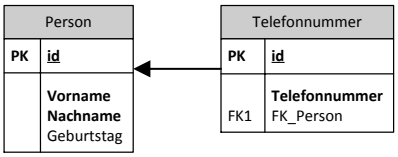

Abbildung 2.1.: Dem Beispiel entsprechendes ER-Diagramm. Ein Entity-Relation-Diagramm, kurz ER-Diagramm, ist eine mögliche Technik, um das Modell der Datenbank zu visualisieren. Es ist zu sehen, dass es Tabellen gibt, welche Attribute diese enthalten und wie die Tabellen zusammenhängen.

id	Vorname	Nachname	Geburtstag
1	Hein	Blöd	NULL
2	Käpt'n	Blaubär	1993-10-02

id	Telefonnummer	FK_Person
1	0711 / 929-0	2
2	0761 / 38 08-0	2

Tabelle 2.1.: Beispiel einer Datenbank. Hein Blöd hat zwar keine Telefonnummer, da in der Telefontabelle kein Eintrag auf seine id zeigt, doch sein Chef Käpt'n Blaubär hat dafür zwei Rufnummern.

2.2. Eye-Tracking

Heutzutage werden viele Studien aus der Forschung sowie Usability- und Marktforschungsstudien durch Eye-Tracking unterstützt. Dabei wird das Blickverhalten der Studienteilnehmer bei einer gegebenen Aufgabenstellung untersucht und überprüft, ob dieses der zuvor aufgestellten Hypothese entspricht. Falls dies nicht der Fall sein sollte, muss überlegt werden, wie man die Anordnung des Versuchs (dies kann sowohl Software als auch Hardware betreffen)

[1] VarChars sind Strings mit vorgegebener Höchstgröße. Dies verbessert die Zugriffsgeschwindigkeit innerhalb der Datenbank.

so ändern kann, dass das gewünschte Ziel erreicht wird, oder ob der generelle Ansatz der Hypothese überdacht werden muss. Diese Hypothesen beziehen sich in der Marktforschung häufig auf das Kaufverhalten und die Manipulation der Aufmerksamkeit einer Zielgruppe, um diese zum Kauf eines bestimmten Produkts zu bewegen. In der Wissenschaft wird häufig die Qualität einer neuen Technik, beispielsweise aus dem Gebiet der Visualisierung, oder die Nutzbarkeit eines Programms untersucht. Letztere Untersuchung findet man auch in der Industrie.

2.2.1. Das menschliche Auge

Die visuelle Wahrnehmung des Menschen hat einen Anteil von über 95% des Informationsflusses der menschlichen Wahrnehmungsorgane [41]. Um mit einem Eye-Tracker den menschlichen Blick nachvollziehen zu können, muss daher der Ablauf des menschlichen Sehens nachvollzogen werden können.

Die Lichtstrahlen, die der Mensch wahrnimmt, gehen durch die Pupille und treffen dann auf die Linse, in der sie gebündelt werden. Dies ermöglicht das Fokussieren auf eine bestimmte Tiefe, wodurch wir bestimmte Objekte scharf sehen können. Nachdem die Lichtstrahlen von der Linse gebündelt wurden, treffen sie auf die Netzhaut. Die Netzhaut besitzt zwei Arten von Rezeptortypen: Die Zapfen und die Stäbchen. Die Stäbchen nehmen nur Helligkeit wahr, wohingegen die Zapfen unterschiedliche Farbtöne wahrnehmen. Es gibt dabei drei Arten von Zapfen: L-, M- und S-Zapfen. Diese können unterschiedliche Farbtöne wahrnehmen. Ein Diagramm der über die Wellenlänge wahrgenommenen Farben findet sich in Abbildung 2.2.

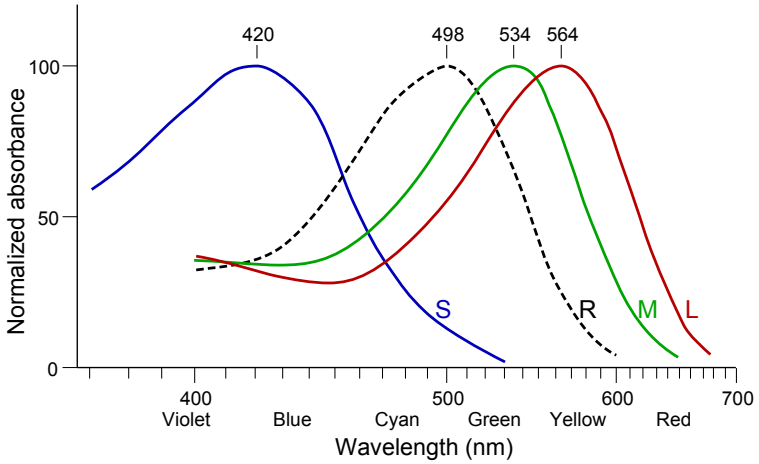

Abbildung 2.2.: Diagramm zur Visualisierung der Sensibilität der drei Zapfenarten. Die drei Zapfenarten werden durch die blaue, die grüne und die rote Linie repräsentiert. Die schwarz gestrichelte Linie repräsentiert die normierte Empfindlichkeitsverteilung. Die y-Achse repräsentiert die Sensibilität, die x-Achse die Wellenlänge. Quelle: [40]

Die Stäbchen treten stark verdichtet in einem kleinen Gebiet der Netzhaut, Fovea genannt, auf. Außerhalb dieses Gebiets treten sie deutlich seltener auf. Dies hat zur Folge, dass das

menschliche Auge nur einen kleinen Teil des Blickfelds scharf wahrnimmt und nur in diesem Farben wahrnimmt. Das scharfe Blickfeld entspricht ungefähr einem Fingernagel mit einer Entfernung von Armlänge vom Auge (siehe [7, Seite 21]. Der Teil, den wir wahrnehmen, obwohl wir unseren Blick nicht darauf fokussieren, nennt man „peripheres Blickfeld“.

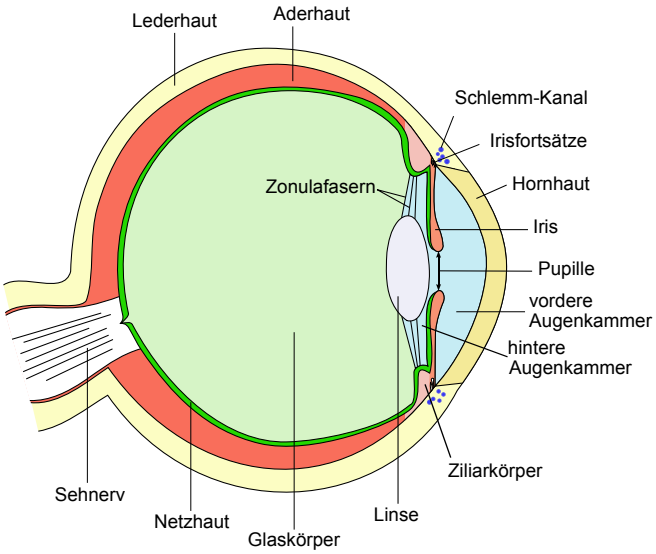

Abbildung 2.3.: Übersicht über den Aufbau des menschlichen Auges. Die Lichtstrahlen fallen durch die Pupille ein, werden dann durch die Linse gebrochen und schneiden sich im Idealfall auf der Netzhaut. Dort wird das Licht von den Zapfen und Stäbchen aufgenommen und in elektrische Signale umgewandelt, die durch den Sehnerv an das Gehirn weitergeleitet werden. Dort werden die Signale dann zu einem Bild zusammengefügt. Quelle: [39]

2.2.2. Augenbewegungen des Menschen

Um das kleine scharfe Blickfeld zu kompensieren, springt das Auge zwischen allen Punkten im Raum, die das Gehirn für die Erstellung eines Bilds benötigt. Holmqvist definiert folgende sechs Metriken zu Augenbewegungen (siehe [7, Seite 21-23]):

Blickpunkt: Ein Blickpunkt ist ein einzelner Blick auf eine bestimmte Stelle. Da diese jedoch von sehr kurzer Dauer sind und aufgrund der nachfolgenden Effekte praktisch nie auf die gleiche Stelle treffen, bedarf es einer Zusammenfassung dieser Blickpunkte.

Fixation: Eine Fixation ist die Zusammenfassung aufeinanderfolgender Blickpunkte auf eine bestimmte Position im Raum. Hierbei wird eine geringe Toleranz zugelassen, sodass auch räumlich sehr eng zusammen liegende Blickpunkte in die gleiche Fixation aufgenommen werden. Hierdurch sollen Effekte wie Beben, Drift und Mikrosakkaden gefiltert werden.

Sakkade: Eine Sakkade ist der Sprung des Auges von einer Fixation zu einer anderen. Da dieser Übergang näherungsweise spontan erfolgt, nimmt das Auge während der

Sakkade vermutlich keine Informationen wahr und der Mensch ist daher in dieser Zeit blind (siehe [7, Seite 23]). Eine Sakkade dauert ca. 30-80 ms.

Beben: Während eines Blickes verweilt das Auge nicht an der gleichen Stelle. Das Auge unterliegt einem ständigen Beben, dessen Zweck nicht geklärt ist. Dieses wird in der englischen Fachliteratur als Tremor bezeichnet.

Mikrosakkade: Zusätzlich zum Tremor treten häufig Sakkaden mit sehr kleinen Sprungdistanzen auf. Diese werden als Mikrosakkaden bezeichnet und dauern ca. $10 - 30$ ms.

Drift: Neben dem Beben und Mikrosakkaden wandert das Auge beständig von seinem eigentlichen Fokus ab. Diesen Effekt nennt man Zug oder Drift. Die Driftdauer beträgt ungefähr $200 - 1000$ ms.

In diesem Zusammenhang sei darauf hingewiesen, dass in der Literatur die Definition von Fixation und Blickpunkt nicht einheitlich ist. In manchen Büchern entspricht die Definition der Fixation dem hier definierten Blickpunkt und der Blickpunkt einer Fixation.

2.2.3. Historische Entwicklung

Die Untersuchung des Zusammenhangs zwischen der Augenbewegung des Menschen und seiner Wahrnehmung geht bis ins Jahr 1879 zurück. In diesem Jahr untersuchte Émile Javal erstmals das Leseverhalten von Menschen durch reine visuelle Beobachtung [11]. Er stellte fest, dass die meisten Menschen nach zirka 10 Buchstaben kurz pausieren. Daraus folgerte er, dass dies die Grenze von Elementen ist, die ein Mensch in seinem Kurzzeitgedächtnis halten kann. Die daraufhin entwickelten ersten Eye-Tracking-Systeme waren stark invasiv und hatten direkten Kontakt mit der Hornhaut. Im Jahr 1901 entwickelten Dodge und Cline den ersten präzisen, nicht-invasiven Eye-Tracker. Dieser nahm die horizontale Augenposition durch die Lichtreflektion der Hornhaut auf einer Fotoplatte auf. Deswegen durften die Probanden zum Zeitpunkt der Aufnahme ihren Kopf nicht bewegen. Vier Jahre später entwickelten Judd, McAlister und Steel mittels Filmaufnahmen das Aufzeichnen von zwei Dimensionen. Sie zeichneten keine Reflektionen auf, sondern einen kleinen weißen Punkt, der auf das Auge der Probanden vor dem Versuch aufgebracht wurde. In den 1930er Jahren untersuchten Miles Tinker und seine Kollegen die Auswirkungen u. a. von Schriftart, Schriftgröße, Seitenaufbau auf das Lesen (siehe [34], [35]). 1948 entwickelten Hartridge und Thompson den ersten Eye-Tracker, der am Kopf von Probanden befestigt werden konnte, wodurch es möglich wurde, den Kopf während der Versuche zu bewegen. Mit dem Aufkommen von Minirechnern[2] wurde es möglich, Eye-Tracking-Daten in Echtzeit zu verarbeiten. Heutige Eye-Tracker sind meist nicht mehr invasiv. Die Rechenleistung eines heutigen Desktopcomputers genügt, um die Daten des Eye-Trackers in Echtzeit zu verarbeiten. Durch die Nutzung von Head-Mounted Eye-Trackern kann man sich heute frei bewegen. Hier wird die Recheneinheit in einen Rucksack ausgelagert oder die Daten werden kabellos an einen stationären Rechner übertragen.

[2]Minirechner sind eine Klasse von Computern, die in den 1960er Jahren aufkamen und eine kleine, kostengünstige Alternative zu Mainframes darstellten.

2.2.4. Die Eye-Mind-Hypothese

Heutzutage wird Eye-Tracking in vielen Gebieten eingesetzt. Dies liegt unter anderem daran, dass Marcel Adam Just und Patricia A. Carpenter im Jahr 1980 [12] die so genannte *Eye-Mind-Hypothese* aufgestellt haben. Diese nimmt an, dass es einen direkten Zusammenhang zwischen dem Blick des Menschen und seiner kognitiven Verarbeitung gibt. Die Grundannahme ist, dass man über das nachdenkt, was man ansieht. Das Eye-Tracking kann also dahingehend interpretiert werden, dass nicht nur beobachtet wird, wo das Auge rein mechanisch hinsieht, sondern auch, worüber die Person gerade nachdenkt und was ihre Aufmerksamkeit erregt.

2.2.5. Funktionsweise eines modernen Eye-Trackers

Moderne Eye-Tracker nutzen Videoanalyse, um die Blickrichtung der Augen ermitteln. Hierzu wird infrarotes Licht vom Eye-Tracker ausgestrahlt, welches von der Hornhaut reflektiert wird. Das reflektierte Licht wird von einer Infrarotkamera aufgenommen, um die Position der Pupille des Auges zu finden. Mit diesen Informationen könnte man bereits die Blickrichtung feststellen. Um die Robustheit der Messung zu erhöhen, wird jedoch zusätzlich die Hornhautreflektion als zusätzliches Merkmal genutzt. Aus diesen Informationen kann mittels eines Algorithmus (beispielsweise der Starburst-Algorithmus [17]) die Blickrichtung errechnet werden.

Abbildung 2.4.: Abbildungen von aktuellen Eye-Trackern der Firma Tobii. Quelle: tobii.com
Links: Das stationäre Modell XL60.
Rechts: Das Head-Mounted System WS619

2.2.6. Messbare Metriken

Heutige Eye-Tracker zeichnen zumeist die Blickpunkte auf und errechnen direkt eine zugehörige Fixation. Es gibt zunächst zwei Möglichkeiten die Information „wo“ hingesehen wurde aufzuzeichnen:

1. Es wird die Position des Kopfes bzw. der Augen im Raum ermittelt. Als weitere Information wird die Blickrichtung durch einen Raumwinkel aufgezeichnet. Dies ist von Vorteil, wenn der Stimulus im 3D-Raum ist oder aus sonstigen Gründen die Blickrichtung von Interesse ist. Der Nachteil dieser Methode ist, dass das Mapping der Blickrichtung auf einen Stimulus zu einem späteren Zeitpunkt erfolgen muss.

2. Wenn klar ist, dass die Aufnahme zu einem flachen Stimulus, beispielsweise auf einem Bildschirm, gehört, so kann man direkt die Fixation als Koordinaten (beispielsweise Pixelkoordinaten) auf dem Stimulus angeben. Der Vorteil hiervon ist, dass das Mapping von dem Blick auf den Stimulus schon in der Vorverarbeitung abgeschlossen ist. Der daraus folgende Nachteil ist, dass nach der Berechnung der Position keine Informationen über die ursprüngliche Augenposition oder deren Blickrichtung verfügbar sind.

Eye-Tracking-Aufzeichnungen enthalten häufig noch eine Untermenge folgender Metriken:

Validität der Aufzeichnung: Manchmal kann der Eye-Tracker nur ein Auge oder gar keines erkennen. Dies kann sowohl durch Effekte wie Blinzeln verursacht werden, als auch durch Störfaktoren bei den Probanden wie glasige Augen, Wimperntusche und dergleichen. Damit in Folge dessen keine Fehlinformationen weiterverarbeitet werden, zeichnen Eye-Tracker häufig die Messvalidität als eine Art Fehlercode mit auf. Diese Daten können später genutzt werden, um den Anteil von fehlerfreien Daten zu ungültigen Daten zu berechnen. Dieser Wert wird häufig als Gütewert für eine Aufnahme genutzt.

Größe der Pupille: Die Größe der Pupille hängt vorrangig vom Lichteinfall ab. Die Pupillengröße wird zudem von folgenden Effekten beeinflusst [7, Seite 393-394]:

- Emotionen und Erwartungen vergrößern die Pupillengröße.
- Verschlafenheit und Erschöpfung verringern die Pupillengröße.
- Diabetes und Alter verringern die Pupillengröße.
- Drogen und Schmerzen vergrößern die Pupillengröße.
- Mentale Arbeitsbelastung vergrößert die Pupillengröße.

Vor allem der letzte Punkt ist von besonderem Interesse für die Feststellung, wie anstrengend oder belastend eine gegebene Aufgabe bzw. Technik ist.

Angezeigter Stimulus: Da eine Eye-Tracking-Aufzeichnung mehr als einen Stimulus enthalten kann, wird meist der aktuell angezeigte Stimulus mit angegeben.

Events: Falls eine Eye-Tracking-Aufnahme Interaktionen, wie etwa Mausklicks oder Tastendrücken, vorsieht, werden diese häufig mit aufgezeichnet.

2.2.7. Areas of Interest

Je nachdem, was bei einem Eye-Tracking-Versuch untersucht werden soll, ist es unter Umständen nicht von Interesse, die einzelnen Fixationen zu betrachten. Es wäre beispielsweise denkbar, dass die Aufenthaltsdauer in einem bestimmten Gebiet des betrachteten Stimulus oder die Übergänge zwischen verschiedenen solchen Gebieten interessant sind. Zu diesem Zweck wurden die so genannten Areas of Interest (kurz AOI, dt. Interessensgebiet) eingeführt. Hierbei werden bestimmte Gebiete des Stimulus definiert (häufig als Rechteck, aber auch anderen Formen wie Ellipsen und Polygone sind denkbar) und semantisch annotiert oder aufgezählt. Hierdurch wird für Fixationen zudem die Überführung von den von Duchowski [3] beschriebenen WO- [6] in den WAS-Raum [10] ermöglicht. Hierbei sind in der Regel folgende Metriken von besonderem Interesse:

AOI-Treffer (hit): Ein AOI-Treffer beschreibt für eine Area of Interest den Zeitpunkt, zu dem eine Fixation zum ersten Mal auf die AOI trifft. Das Ereignis wird nur von einer Fixation ausgelöst, falls die vorhergehende Fixation nicht bereits in der AOI lag. Es ist wichtig, dass hiermit nicht Fixationen innerhalb einer AOI beschrieben werden, sondern ausschließlich jene, die das Eintrittsereignis auslösen.

AOI-Aufenthalte (dwells; gaze duration): Hiermit wird der eigentliche Aufenthalt innerhalb einer AOI beschrieben. Ein Aufenthalt beschreibt alle Fixationen, die zwischen einem bestimmten Eintritt und dessen Austritt aus der AOI liegen.

AOI-Übergänge (transitions): Eine Transition beschreibt den Übergang zwischen zwei Areas of Interest. Sie enthält die letzte Fixation der alten Area of Interest und die erste Fixation der neuen AOI.

2.3. Gängige Eye-Tracking-Visualisierungstechniken

Um Eye-Tracking-Daten zu visualisieren, haben sich, in Abhängigkeit der Art des Stimulus, Techniken etabliert. Hierbei handelt es sich um die Scanpaths und Heatmaps, die bei statischen Stimuli angewendet werden, und den Bee Swarm, der bei videobasierten Stimuli genutzt wird. Diese werden im Folgenden vorgestellt.

2.3.1. Attention maps (dt. Aufmerksamkeitskarten)

Attention Maps [31] stellen ein Hilfsmittel zur visuellen Repräsentation von zweidimensionalen Daten dar. Ziel ist es, die Verteilung einer gegebenen Punktmenge zu visualisieren. Hierzu wird die Attention Map häufig über das Bild gerendert, auf dessen Grundlage die Punktmenge erzeugt wurde[3]. Beim Eye-Tracking können die auf den Stimulus gemappten Fixationen als Punkte aufgefasst werden, sofern der Stimulus eine Ebene darstellt. Ansonsten muss eine virtuelle Ebene definiert werden, auf der das Mapping ausgeführt werden kann. Im folgenden wird davon ausgegangen, dass die zu visualisierenden Daten der Attention Map aus dem Gebiet des Eye-Tracking stammen und dass der Stimulus eine Ebene ist. Es gibt mehrere Möglichkeiten, die Verteilung der Punkte auf einer Attention Map zu visualisieren. Im Allgemeinen folgen jedoch alle derartigen Visualisierungen der in Abbildung 2.5 dargestellten Verarbeitung. Die Attention Map kann auf vielfältige Art aufgebaut werden.

Anzeige mittels einer Höhenkarte: Die Attention Map kann als Höhenkarte aufgefasst werden. Dabei wird an jeder Fixationsposition eine Anhebung erzeugt (optional auch mit Gewichtung auf Basis der Fixationsdauer). Da Attention Maps im Allgemeinen nur eine Übersicht über die Daten geben sollen, kann das Höhenprofil zudem geglättet werden. Einer der großen Nachteile der dreidimensionalen Darstellung der Daten ist die potenzielle Überdeckung von anderen Daten. Deshalb sollten dem Benutzer Interaktionsmöglichkeiten zur Exploration der Visualisierung zur Verfügung gestellt werden.

[3] Die Anzeige auf einem Bild ist nur sinnvoll, wenn die Datengrundlage nicht zeitbasiert ($\triangleq$ ein Bild) ist, also nicht aus Videodaten o. ä. besteht.

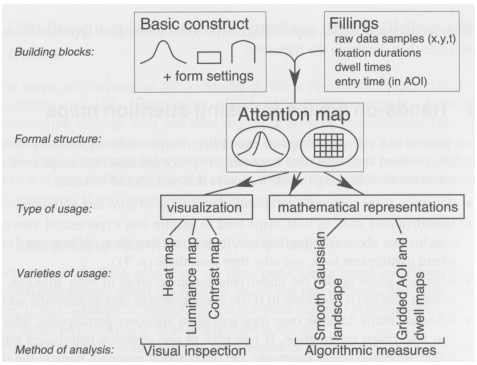

Abbildung 2.5.: Aufbau einer Attention Map. Zunächst werden die verfügbaren Formen und ihre Eigenschaften festgelegt und die Daten der Eye-Tracking-Aufnahme erhoben. Dann werden die Daten und die Formen zusammengeführt und eine Attention Map wird erzeugt. Diese kann intern als kontinuierliche Funktion (format structure, links) oder quantisierte Struktur (format structure, rechts) vorliegen. Danach kann die Attention Map entweder algorithmisch oder visuell ausgewertet werden. Quelle: [7, Seite 237]

Anzeige mittels Heatmap: Alternativ zur Darstellung als Höhenprofil können die Daten auch auf eine Farbskala übertragen und die Verteilung der Fixationen auf diesem Weg visualisiert werden. Hierbei werden häufig Gradienten von blau über grün und gelb nach rot (ähnlich der in der Meteorologie genutzten Farbwerte für Wärme) angelegt. Die daraus entstehende Karte wird als Heatmap bezeichnet. Sie findet häufig Anwendung, um die Verteilung der Fixationen auf einem Stimulus festzustellen. Ihr Vorteil ist, dass es keine Überdeckungen gibt, da diese Visualisierung zweidimensional ist. Der Nachteil ist, dass das menschliche Auge nur einen begrenzten Farbraum erfassen kann. Im Fall der Heatmap kann ein Mensch daher ähnliche Werte nur schwer voneinander unterscheiden. Dieser Nachteil ist jedoch nachrangig, da Heatmaps im Allgemeinen nur zur Gewinnung einer Übersicht genutzt werden.

2.3.2. Scanpaths (dt. Blickpfade)

Um das Verhalten einzelner Probanden genauer zu betrachten, ist es möglich, die Fixationsdaten mittels so genannter Scanpaths zu visualisieren. Eine der Möglichkeiten, einen Scanpath zu definieren ist, ihn als vorgegebenen Blickpfad einer Person für einen bestimmten Stimulus anzusehen. Dies würde bedeuten, dass eine Person einen Stimulus immer auf die gleiche Art und Weise ansehen würde [21] [22]. Die heutige Definition entspricht weitgehend dieser Annahme. Heutzutage wird zusätzlich die Umgebungssituation berücksichtigt, die einen markanten Einfluss auf das Verhalten eines Probanden haben kann. Eine Möglichkeit, die einzelnen Fixationspunkte als Scanpath zu visualisieren, ist, die einzelnen Fixationen als Punkte darzustellen und aufeinander folgende Fixationen mit Linien zu verbinden. Es ist dabei möglich, nur Fixationen innerhalb von bestimmten zeitlichen Abständen anzuzeigen. Außerdem kann man die Fixationsdauer auf die Punktgröße abbilden und diese somit

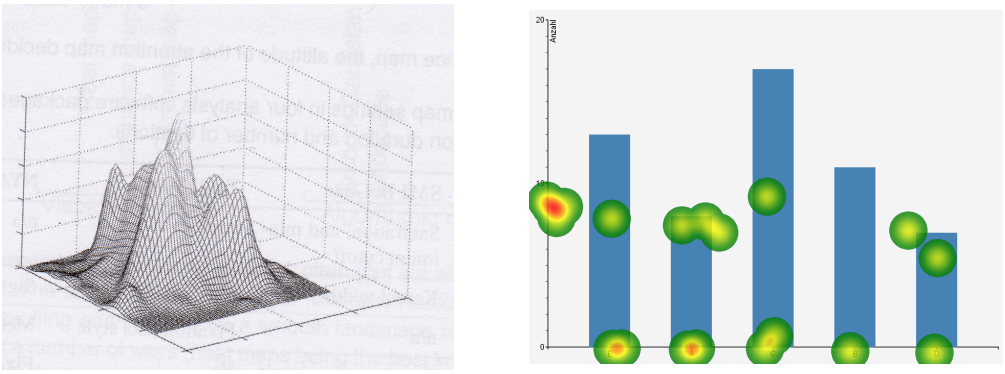

Abbildung 2.6.: Beispiel für eine Attention Map (links) und eine Heatmap (rechts). Die Attention Map stammt aus [7, Seite 235]. Die Heatmap wurde mit Tobii Studio 3 erzeugt. Als Datengrundlage diente ein Balkendiagramm der Studie „Visual Elements III“ des Instituts für Visualisierung und interaktive Systeme.

visualisieren. Neben der statischen Visualisierung ist es möglich, die Zeitdimension durch Verwendung eines Space-Time-Cubes zu repräsentieren (siehe hierzu [15]). Scanpaths eignen sich insbesondere dazu, Aufnahmen mit wenigen Fixationen anzuzeigen und eine geringe Anzahl an Probanden, etwa zwei bis fünf, miteinander zu vergleichen. Eine zu hohe Anzahl an gleichzeitig angezeigten Probanden oder eine sehr hohe Fixationszahl führt zu Überdeckungen und Visual Clutter, was in Abbildung 2.7 zu sehen ist. Unter Visual Clutter versteht man einen Zustand, bei dem die Fülle an Objekten oder deren Desorganisation zu einer Verringerung der Leistung einer bestimmten Aufgabenstellung führt [27]. Zudem erhöht Visual Clutter die kognitive Last einer Visualisierung. Die kognitive Last beschreibt den geistigen Aufwand, der benötigt wird, um eine Aufgabe, wie das Betrachten einer Visualisierung, erfolgreich zu bearbeiten. Eine weitere Ausführung über kognitive Last bei Visualisierungen findet sich in [8].

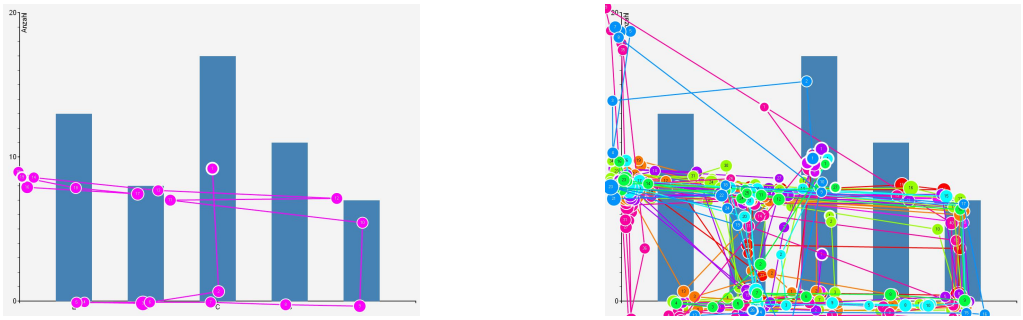

Abbildung 2.7.: Links: Darstellung eines Scanpaths. Es ist gut zu sehen, dass der Proband zunächst die Beschreibungen der Balken abgelesen hat, dann den Balken A mit dem Balken D verglichen hat und dann den Wert von Balken A ausgelesen hat. Rechts: Darstellung des gleichen Stimulus mit mehreren Probanden. Die starke Überdeckung der Punkte und der Linien macht es unmöglich, Erkenntnisse aus der Aufnahme zu gewinnen.

2.3.3. Bee-Swarm

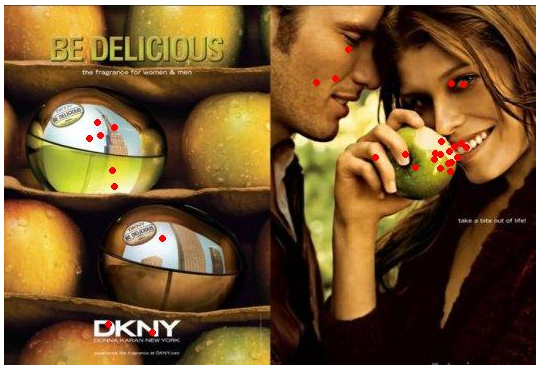

Abbildung 2.8.: Beispiel einer Bee-Swarm-Visualisierung. Der Stimulus ist videobasiert. Die Visualisierung zeigt, an welche Position auf dem Stimulus die einzelnen Probanden gesehen haben. Dies ermöglicht eine Einschätzung über die Verteilung der Interessenspunkte auf dem Stimulus zu dem gegebenen Zeitpunkt. Quelle: [26]

Bei Stimuli, die sich über die Zeit verändern, wie etwa Videodaten, sind Scanpaths nicht praktikabel. Um dem Analysierenden dennoch eine Visualisierung über die Fixationen zu einem gegebenen Zeitpunkt zur Verfügung zu stellen, wurde die „Bee Swarm"-Visualisierung entwickelt. Hierbei werden die Fixationen aller Probanden direkt auf dem Stimulus mit einem Punkt angezeigt. Somit wird es möglich, die Interessensgebiete der Probanden nachzuvollziehen. Einer der Nachteile dieser Technik ist die schwere bis unmögliche Nachvollziehbarkeit der Blickpfade einzelner Probanden, da man die Sakkaden der Probanden zwar visualisieren kann, dies jedoch zu sehr viel Visual Clutter führt. Ein Beispiel einer Bee-Swarm-Visualisierung ist in Abbildung 2.8 gegeben.

2.4. Stringähnlichkeit

Bei Eye-Tracking-Studien will man häufig zwei oder mehr Aufnahmen von Probanden miteinander vergleichen. Um dies auf der Datenebene durchzuführen, muss eine Metrik festgelegt werden, mit welcher die Aufnahmen der Probanden auf einem Stimulus verglichen werden können. Hierbei bietet es sich an, die Aufnahmen als Strings anzusehen und dann die Stringähnlichkeit zu vergleichen. Dabei stellen sich zwei Fragen:

1. Was heißt „ähnlich"?
2. Wie sind die Zeichenfolgen aufgebaut und welches Ziel soll mit dem Vergleich erreicht werden?

Diese beiden Fragen hängen stark voneinander und vom Kontext, in dem sie stehen, ab. Hier ist der Kontext der Vergleich von Eye-Tracking-Fixationen auf einem Stimulus. So könnte beispielsweise die Ähnlichkeit der Strings „Äpfel" und „Birnen" verglichen werden, um

deren Ähnlichkeit zu ermitteln. Hierbei ist das Ziel die Definition eines Abstandsmaßes, das „Äpfel“ und „Birnen“ vergleichbar macht.

Hierfür werden häufig die Stringvergleichsalgorithmen von Levenshtein [16], Needleman und Wunsch [20] sowie Smith und Waterman [30] genutzt. Im Folgenden wird exemplarisch die Edit-Levenshtein-Distanz als Abstandsmaß für Stringvergleiche vorgestellt.

2.4.1. (Edit-)Levenshtein-Distanz

Die Berechnung der Levenshtein-Distanz ist ein Verfahren zur Bestimmung der Ähnlichkeit von zwei Strings, indem es die Kosten berechnet, einen der Strings durch Ersetzen, Einfügen und Löschen von Buchstaben in den anderen String umzuformen [1, Seite 188ff.]. Eine Spezialform der Levenshtein-Distanz ist die Edit-Distanz. Hier gilt die Einschränkung, dass die Einfüge- und Löschkosten gleich sind. Diese Form wird in der Informatik jedoch i. A. nicht gesondert behandelt. Nachfolgend werden zunächst die benötigten Parameter und dann der Ablauf des Algorithmus beschrieben.

Parameter des Algorithmus

Zunächst benötigt man für den Algorithmus zwei Strings A und B. A hat die Länge n und B die Länge m. Mit $A[i]$ beschreibt man das Zeichen an der Stelle i im String A und $B[j]$ das Zeichen an der Stelle j im String B. Bei dem Algorithmus kann prinzipiell festgelegt werden, wie teuer die Operationen

- Buchstabe einfügen (insert) [c_i^i]
- Buchstabe ersetzen (replace) [c_i^r] und
- Buchstabe löschen (delete) [c_i^d]

sein sollen. Häufig werden die Werte $c_i^i = c_i^d = 2$ und $c_i^r = 1$ gewählt. Prinzipiell kann man auch beliebige nicht-konstante Funktionen an die Kosten binden.

Ablauf des Algorithmus

Es wird eine Tabelle angelegt, die $(n+1) \times (m+1)$ Felder groß ist. Dabei ist n die Länge des Ausgangsstrings und m die Länge des Zielstrings. Um die Tabelle einfacher initialisieren zu können, wird jeweils noch ein Leerzeichen vor den Strings hinzugefügt. Die Tabelle wird danach wie folgt gelesen: Aus der Spaltenposition der Tabelle kann abgelesen werden, welcher Substring des Ausgangsstrings betrachtet wird (von Position 0, die dem Leerzeichen entspricht, bis zur Position x). Die Zeilenposition gibt den Teilstring an, der als Zielstring angesehen wird. Die Felder der ersten Zeile werden dabei mit der Formel 2.1 initialisiert.

$$T(x,0) = c^d \cdot x \tag{2.1}$$

Dies entspricht dem Löschen des gesamten Strings (da der Zielstring in der ersten Zeile keine Zeichen enthält). Des Weiteren wird die erste Spalte mit der Formel 2.2 initialisiert.

$$T(0,y) = c^i \cdot y \tag{2.2}$$

Dies entspricht dem Einfügen von allen Buchstaben, bis der gewünschte Teilstring vollständig eingefügt wurde.

Alle nun folgenden Felder werden mit folgender Vorschrift erzeugt:

$$T(x,y) = min \begin{cases} T(x-1,y) + c^i & \\ T(x-1,y-1), & \text{falls } A(x) = B(y) \\ T(x-1,y-1) + c^r, & \text{sonst} \\ T(x,y-1) + c^d & \end{cases} \tag{2.3}$$

Im Feld T(n, m) stehen nun die minimalen Kosten, um die Strings ineinander zu überführen. Es ist möglich, durch eine Analyse der Tabelle festzustellen, welche Schritte ausgeführt werden müssen, um String A in String B umzuwandeln. Dies ist jedoch für eine reine Kostenanalyse unnötig.

		C	H	I	R	U	R	G
	0	2	4	6	8	10	12	14
C	2	0	2	4	6	8	10	12
H	4	2	0	2	4	6	8	10
I	6	4	2	0	2	4	6	8
G	8	6	4	2	1	3	5	7
U	10	8	6	4	3	1	3	5
R	12	10	8	6	4	3	1	3
R	14	12	10	8	6	5	3	**2**

Tabelle 2.2.: Kostentabelle des Beispiels für die Levenshtein-Distanz mit den Strings „CHIRURG“ und „CHIGURR“. Die Kosten, um aus dem String „CHIRURG“ den String „CHIGURR“ zu erzeugen, betragen zwei (das erste R durch ein G ersetzen und das letzte G durch ein R).

2.4.2. Ähnlichkeit mittels Needleman-Wunsch

Eine weitere Möglichkeit die Ähnlichkeit von Strings zu vergleichen ist mit dem Algorithmus von Needleman und Wunsch [20]. Der Algorithmus stammt aus der Bioinformatik und zählt zu den Sequence Alignment Algorithmen. Er wurde entwickelt, um die Ähnlichkeit von DNA-Sequenzen zu bestimmen. Der Algorithmus funktioniert größtenteils analog zu dem Algorithmus von Levenshtein. Der einzige Unterschied ist die Berechnung der Kostenfunktion, sollte ein Zeichen ersetzt werden. Bei Levenshtein wird hier eine Kostenfunktion angewendet die nicht berücksichtig, welche Zeichen miteinander verglichen werden. Bei dem Algorithmus von Needleman und Wunsch gibt es eine Ähnlichkeitsmatrix, die die Kosten für das ersetzen eines Zeichens in ein anderes angibt. Ein Beispiel für eine solche Matrix ist in Tabelle 2.3 gegeben.

	A	**G**	**C**	**T**
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

Tabelle 2.3.: Beispiel einer Ähnlichkeitsmatrix für den Algorithmus von Needleman und Wunsch. Aus der Tabelle kann abgelesen werden, wie hoch die Kosten für das Ersetzen eines Zeichens in ein anderes ist. So sind beispielsweise die Kosten für das Umwandeln des Zeichens „A“ zu „T“ -4.

2.4.3. Ähnlichkeit mittels Musterabgleich

Alternativ zum Vergleich zweier Strings durch eine Umwandlung eines Strings in einen Vergleichsstring ist es möglich, übereinstimmende Substrings zu suchen. Die Anwendung von Musterabgleichen, im Englischen *pattern matching* genannt, reichen von der Linguistik bis zur Informatik. In der Informatik findet es in vielen Gebieten, wie beispielsweise dem Compilerbau und dem Data Mining, Anwendung. Ein Beispiel für einen Musterabgleichsalgorithmus ist der Algorithmus von Knuth, Morris und Pratt [14].

2.5. Clustering

Zur besseren Vergleichbarkeit ist es sinnvoll, einzelne Datensätze zu Gruppen zusammenzufassen. Es kann beispielsweise bei Studien von Interesse sein, Altersgruppen miteinander zu vergleichen. Diese müssen jedoch zunächst in Gruppen zusammengefasst werden. Wenn diese Gruppierung vollautomatisch geschieht, dann spricht man von „Clustering“.

Der folgende Abschnitt befasst sich zunächst mit den dem Ursprung von Clusteringalgorithmen, sowie deren zwei Arten. Daraufhin wird für die beiden Arten jeweils ein Algorithmus exemplarisch vorgestellt.

Ursprünge der Clusteringproblematik

Urspünglich kommen Clusteringalgorithmen aus dem Themengebiet des Information-Retrieval. Hier bestand das Problem, dass beispielsweise bei Suchanfragen im Internet nicht einzelne Ergebnisse, sondern auch damit verwandte Ergebnisse ausgegeben werden sollen. So sollten beispielsweise bei der Suche nach dem Wort „Handy“ sowohl Ergebnisse mit dem Begriff „Handy“ als auch mit „Smartphone“ und „Mobiltelefon“ zurückgegeben werden.

Arten des Clustering

Es gibt verschiedene Ansätze, um Daten zu clustern.

- Die am häufigsten angewendete Form ist das flache Clustering. Hierbei werden alle Daten gleich behandelt und die Anzahl der Gruppen ist häufig a priori festgelegt. Ein prominentes Beispiel hierfür ist der später näher erläuterte *k-means-Algorithmus*.

- Wenn die Anzahl der Gruppen nicht bereits im Voraus bekannt ist, kann man versuchen, mittels adaptiver flacher Clusteringalgorithmen dieses Problem anzugehen. Der *k-means-Algorithmus* kann angepasst werden, sodass er dieses Kriterium erfüllt. Dieser Ansatz nennt sich *Expectation–maximization-Algorithmus*.
- Alternativ zu den flachen Clusteringalgorithmen kann man die Daten hierarchisch anordnen. Mit Hilfe dieser Hierarchie können dann Cluster erzeugt werden.

Genereller Ablauf aller flachen Clusteringalgorithmen

Die vorhandenen Daten müssen auf Basis eines zuvor festgelegten Kriteriums gruppiert werden. Dazu muss ein Abstandsmaß für die Metrik festgelegt werden. Hier wären räumliche Abstände für geometrische Koordinaten, TF-IDF[4] für Suchanfragen oder andere textbasierte Daten oder die Hamming[5]- oder die Levenshtein-Distanz für Stringvergleiche denkbar. Danach werden die Daten auf Grundlage des Abstandsmaßes gruppiert.

Der k-means-Algorithmus

Das Ziel des k-means-Algorithmus ist es, aus N Datensätzen K Gruppen zu bilden. Hierzu werden von dem Algorithmus so genannte Centroide gesucht, die als zentrales Element einer Gruppe angesehen werden. Der Algorithmus ist daher in mehrere Schritte unterteilt, die sich ab dem 2. Schritt wiederholen, bis ein festgelegtes Abbruchkriterium erfüllt ist. Im Abbruchsfall darf der letzte Schritt nicht mehr durchgeführt werden.

1. Wähle zufällig k Centroide.
2. Weise jedem der Centroide eine eigene Gruppe zu.
3. Berechne für jedes Element der Datenmenge den Abstand zu den k Centroiden und weise dem Element die Gruppe des nächsten Centroids zu.
4. Approximiere für jede Gruppe mittels der *sum of squared differences (SSD)* das Element als Centroid, das die SSD minimiert.
5. Setze die Gruppenzugehörigkeit für alle Elemente zurück.

In Abbildung 2.9 wird ein beispielhafter Ablauf illustriert. Dabei ist in (1) der ungeclusterte Datensatz gegeben. Nachdem in (2) zufällig k (hier 2) Centroide ausgewählt wurden, werden in (3) alle Datenpunkte dem nächstliegenden Centroid (und damit seinem Cluster) zugeordnet. Daraufhin wird aus den Datenpunkten, die zum gleichen Centroid gehören ein Cluster gebildet (4) und ein neuer Centroid berechnet (5-6). Weil sich diese neuen Centroide von den in (2) gewählten Centroiden unterscheiden, wiederholen sich die Schritte (7-9). Da sich daraufhin die Centroide nicht mehr verändern, terminiert das Clustering an dieser Stelle.

[4]Term Frequency - Inverse Document Frequency (TF-IDF): Ein Abstandsmaß aus dem Gebiet des Information Retrieval, das aus einer Menge von Dokumenten auf Basis eines gegebenen Terms eine Vorkommenshäufigkeit errechnet [18, Seite 107-109].

[5]Bei der Hamming-Distanz müssen beide Strings gleich lang sein und es dürfen nur Buchstaben ersetzt werden.

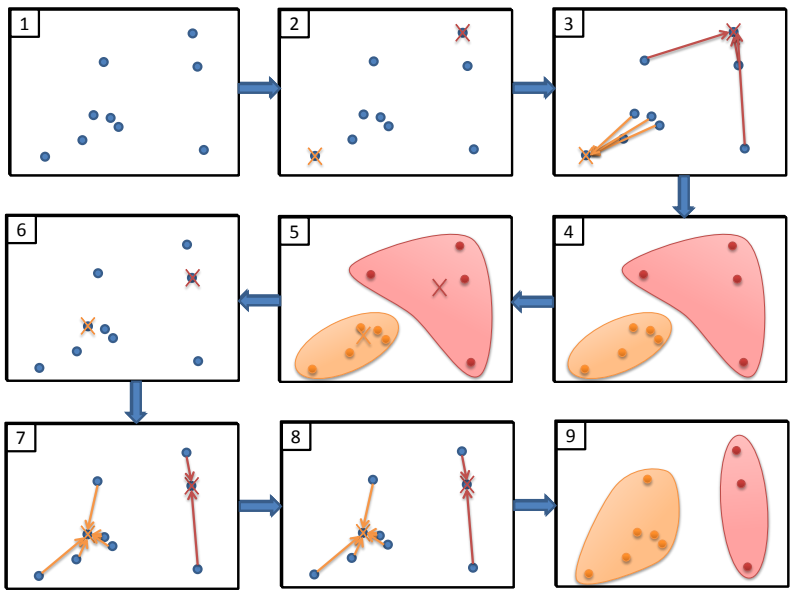

Abbildung 2.9.: Beispielhafter Ablauf des k-means-Algorithmus. Es werden die in 2.5 gegebenen Schritte der Centroidberechnung, Gruppenzuweisung und Neuberechnung der Centroide durchgeführt, bis das Clustering abgeschlossen ist.

Da im ersten Schritt die Centroide zufällig ausgewählt werden, ist das Ergebnis des k-means-Algorithmus nichtdeterministisch.

Hierarchisches Clustering

Einer der größten Nachteile von flachem Clustering ist, dass die Algorithmen unstrukturierte, nichtdeterministische Ergebnisse zurückgeben, die teilweise eine a priori festgelegte Anzahl an Clustern erzeugen.

Beim hierarchischen Clustering werden Cluster in einer Baumstruktur angeordnet und somit ihre Hierarchie festgelegt. Das Clustering kann hierbei auf zwei Arten aufgebaut werden:

Top-Down: Zunächst gehören alle Datensätze dem gleichen Cluster an. Dann werden die Cluster so lange aufgeteilt, bis entweder die gewünschte Anzahl an Clustern erreicht ist oder die Cluster sich ausreichend unterscheiden. Wie „ausreichend unterscheiden“ verstanden werden kann, hängt von der Metrik an, die für die Ähnlichkeit der Datensätze verwendet wird.

Bottom-Up: Zu Beginn stellt jeder Datensatz einen Cluster dar. Die jeweils ähnlichsten Cluster werden so lange zusammengefasst, bis eine minimale Anzahl an Clustern erreicht ist oder die Cluster sich zu stark unterscheiden. Diese Technik wird auch als hierarchisches, agglomeratives Clustering, kurz HAC, bezeichnet.

Da der Top-Down-Ansatz selten Anwendung findet (siehe [18, S. 347], wird im Folgenden von einem Bottom-Up-Ansatz ausgegangen.

Zusammenführung der Cluster

Bei Bottom-Up-Ansätzen werden immer genau die Cluster zusammengeführt, deren Abstand am niedrigsten ist. Nachfolgend wird davon ausgegangen, dass immer nur die zwei ähnlichsten Cluster zusammengeführt werden. Wenn jeder Cluster nur einen Datensatz enthält, entspricht der Abstand der Cluster dem Abstand der Datensätze in den Clustern. Wenn ein Cluster mehrere Datensätze enthält, gibt es mehrere Möglichkeiten, den Abstand festzulegen, wovon die einfachsten zwei im Folgenden kurz beschrieben werden:

Single-linkage: Beim Single-linkage entspricht der Abstand der Cluster dem Abstand der zwei zueinander ähnlichsten Datensätze aus den jeweiligen Clustern.

Complete-linkage: Wenn man den größten Abstand der Datensätze der jeweiligen Cluster als Abstand definiert, so spricht man von complete-linkage.

Ein Beispiel für den Ablauf des hierarchisch-agglomerativen[6] Clusterings findet sich in Abbildung 2.10. In diesem Beispiel wird für die Zusammenführung single-linkage verwendet.

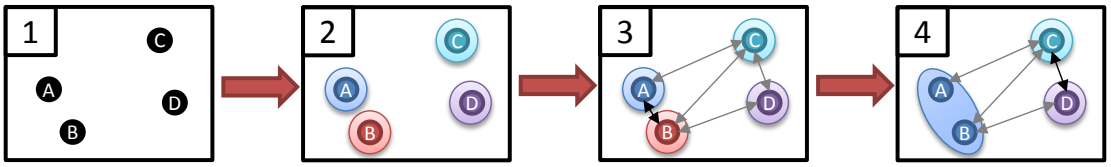

Abbildung 2.10.: Beispielhafter Ablauf des HAC-Algorithmus. In (1) ist der ungeclusterte Datensatz gegeben. In (2) wird jedem Datensatz ein Cluster zugewiesen. Daraufhin werden in (3) die Distanzen zwischen allen Clustern ermittelt und die Cluster mit der geringsten Distanz (in (3) ist diese Verbindung fett hervorgehoben) werden vereinigt.

Dendrogramme - eine Visualisierungstechnik für hierarchisches Clustering

Unter einem Dendrogramm versteht man eine bestimmte Baumvisualisierung, deren Zweck die Visualisierung des Ablaufs eines Clusteringalgorithmus ist. Im Falle eines Bottom-Up-Ansatzes wird die Visualisierung von unten nach oben gelesen. Immer wenn zwei Knoten zusammengeführt werden, entspricht dies der Verschmelzung der entsprechenden Knoten. Abbildung 2.11 zeigt das Dendrogramm, das entstehen würde, wenn Beispiel 2.10 vollständig illustriert wäre (in dem Beispiel fehlt die Vereinigung der letzten beiden Cluster).

[6] Agglomeration ≙ Konzentration, Anhäufung

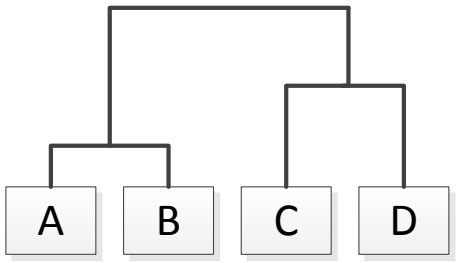

Abbildung 2.11.: Dendrogramm für den in Abbildung 2.10 gegebenen Datensatz. Die Buchstaben entsprechen den einzelnen Datensätzen, die Linien repräsentieren die Cluster. Wenn zwei Linien zusammenlaufen, so wurden die zugehörigen Cluster vereinigt. Wenn die Cluster in der Grafik früher zusammenlaufen, dann sind sie sich ähnlicher.

Zeitkomplexität von hierarchischen Clusteringalgorithmen

Die Komplexität von hierarchischen Clusteringalgorithmen entspricht $\Theta(n^3)$, da in jedem der N - 1 Clusteringdurchläufe eine N $\times$ N Matrix durchlaufen werden muss, um den minimalen Abstand zu ermitteln (siehe [7, S. 253]).

2.6. Informationsvisualisierung

Purchase et al. [24] definieren Informationsvisualisierung folgendermaßen: „Information Visualization utilizes computer graphics and interaction to assist humans in solving problems."

Die Disziplin der Informationsvisualisierung beschäftigt sich mit der Visualisierung von abstrakten Daten. Dieser Abschnitt stellt wichtige Grundlagen und häufig genutzte Techniken der Informationsvisualisierung vor und erläutert diese.

Zunächst werden die Techniken für Datensätze verschiedener Dimensionen vorgestellt. Danach wird die Visualisierungspipeline der Informationsvisualisierung gezeigt. Zuletzt wird das Konzept des *Visual Information Seeking Mantra* vorgestellt.

2.6.1. Techniken zur Visualisierung von Daten

Je nachdem, wie viele Dimensionen ein Datensatz hat, stehen unterschiedliche Techniken zur Visualisierung zur Verfügung:

Eindimensionale Datensätze: Wenn ein Datensatz nur eine Variable hat, dann nennt man diesen Datensatz ***univariat***. Univariate Datensätze können mittels unterschiedlicher Visualisierungen dargestellt werden. Zu diesen zählen unter anderem Balkendiagramm bzw. Histogramm und das Liniendiagramm. Beispiele dieser Visualisierungen werden in Abbildung 2.12 dargestellt.

Zweidimensionale Datensätze: Bei Datensätzen mit zwei Variablen spricht man von ***bivariaten Daten***. Für deren Visualisierung nutzt man häufig Scatterplots (dt. Streudiagramme).

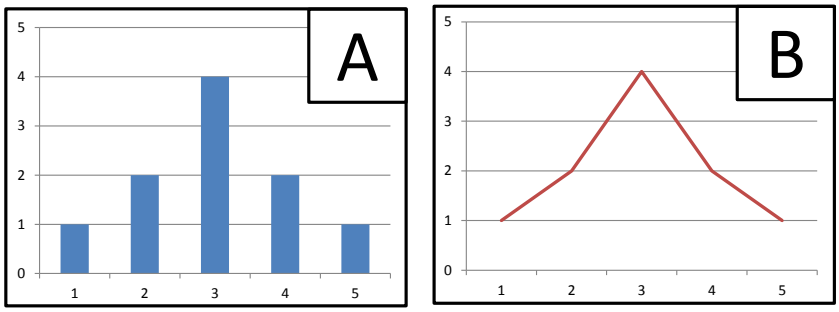

Abbildung 2.12.: Beispiel für die erwartete Punktverteilung einer Prüfung mit 10 Teilnehmern als Balkendiagramm (A) und Liniendiagramm (B).

Hierbei werden die Variablen als Achsen auf einem kartesischen Koordinatensystem aufgefasst. Die einzelnen Daten des Datensatzes werden als Punkte in diesem Koordinatensystem interpretiert aufgetragen. Aus der entstehenden Visualisierung kann man Informationen wie die Korrelation der Variablen oder die Streuung von einem erwarteten Wert ablesen. Eine geeignete Form der Darstellung von bivariaten Datensätzen sind Scatterplots wie in Abb. 2.13 gezeigt.

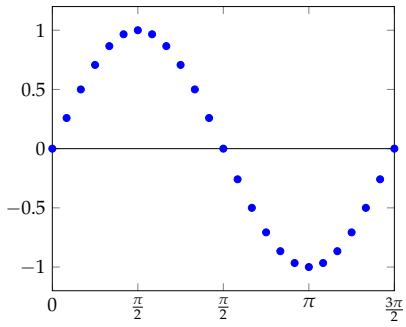

Abbildung 2.13.: Exemplarischer Scatterplot, in dem der Datensatz einer Sinuskurve entspricht.

Dreidimensionale Datensätze: Datensätze mit drei Variablen werden ***trivariate Datensätze*** genannt. Hier entsteht im Gegensatz zu uni- oder bivariaten Datensätzen das Problem, dass die Informationen nicht mehr direkt auf einer planaren Fläche, wie einem Bildschirm, aufgetragen werden können.

Eine Möglichkeit ist es, den Scatterplot als Kubus, also dreidimensional darzustellen. Ein Problem, das immer bei dreidimensionalen Darstellungen auftritt, sind Überdeckungen. Dies wird vor allem bei dichten und großen Datensätzen problematisch.

Als weitere Möglichkeit stellt man die dritte Dimension als Zeitverlauf dar. Dies funktioniert offensichtlich am Besten bei einer Zeitdimension und nicht alle Daten eignen sich hierfür. Ein großer Nachteil dieser Technik ist, dass hierbei die so genannte „Mental Map“ nicht erhalten bleibt. Diese beschreibt den Zusammenhang von verschiedenen

Elementen auf einer kognitiven Ebene. Eine inkonsistente Mental Map hat häufig zur Folge, dass Informationen nicht richtig im Gehirn verknüpft werden können und in Folge dessen auch keine Schlussfolgerungen über den Verlauf oder den Zusammenhang der Daten gezogen werden können.

Die dritte Möglichkeit sind Scatterplot-Matrizen. Hierbei werden jeweils zwei Dimensionen gegeneinander aufgetragen. Die dritte Dimension wird dabei ignoriert. Dies wird für alle möglichen Kombinationen durchgeführt, es werden also alle Dimensionen paarweise dargestellt, und alle entstandenen Scatterplots werden in einer Matrixform nebeneinander aufgetragen.

Mehrdimensionale Datensätze: Wenn ein Datensatz mehr als drei Variablen hat, dann spricht man von ***multivariaten Daten***.

Es ist möglich, die Scatterplot-Matrizen auch bei höherdimensionalen Datensätzen zu nutzen. Es entsteht daher eine $\mathbf{n} \times \mathbf{n}$-Matrix, wobei $\mathbf{n}$ die Anzahl der Dimensionen ist. Des Weiteren kann man so genannte Glyphen verwenden. Hierbei werden die Variablen auf Symbole abgebildet. Dies funktioniert nur bei diskretisierbaren Daten. Außerdem kann man alle Variablen auch mittels so genannter *paralleler Koordinaten* auf eine Grafik abbilden. Diese Technik wird im folgenden Abschnitt genauer erläutert.

Neben den angesprochenen Techniken kann man bei allen Techniken durch zusätzliche Kodierungen wie Punktdicke, Farbe und andere Parameter weitere Informationen in eine Grafik kodieren.

2.6.2. Parallele Koordinaten

Bei den parallelen Koordinaten [9] werden alle Variablen (bei parallelen Koordinaten Attribute genannt) auf jeweils eine eigene vertikale Achse eines Plots aufgetragen. Die einzelnen Tupel der Daten (Fälle genannt) werden dann als Polylinie in den Plot eingetragen. Dabei schneidet die Polylinie die Attribute genau an der Stelle, die dem Wert des Falls für das Attribut entspricht. Der folgende Abschnitt befasst sich zunächst mit den Vorteilen der parallelen Koordinaten. Daraufhin werden die Nachteile und Einschränkungen der parallelen Koordinaten vorgestellt. Anschließend werden Techniken zur Verbesserung der Darstellung der parallelen Koordinaten aufgezeigt. Diese Verbesserungen lassen sich auch auf andere Visualisierungstechniken übertragen.

Vorteile der parallelen Koordinaten

Die parallelen Koordinaten skalieren sehr gut mit der Anzahl der Fälle. Dies liegt unter anderem daran, dass sich die Größe der Visualisierung beim Hinzufügen von weiteren Fällen nicht verändert. Es werden nur neue Polylinien hinzugefügt. Zudem skalieren die parallelen Koordinaten mit der Anzahl der Attribute, da jedes Attribut durch eine weitere Achse in den parallelen Koordinaten dargestellt wird.

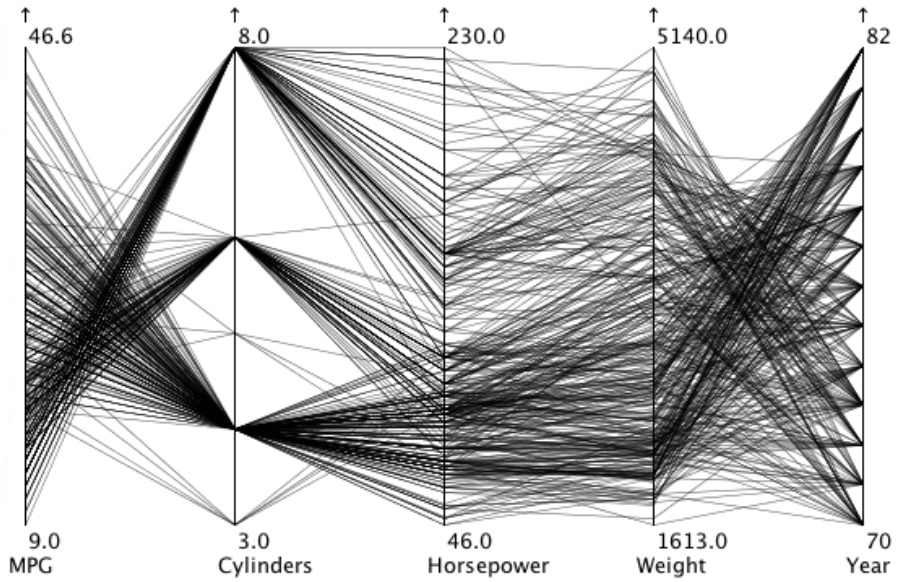

Abbildung 2.14.: Exemplarische Visualisierung eines Datensatzes mit Hilfe von parallelen Koordinaten. Der Datensatz beschreibt Fahrzeugmodelle, die zwischen 1970 und 1982 erschienen sind und enthält den Verbrauch (MPG), die Zylinderanzahl (Cylinders), die Leistung (Hosepower), das Gewicht (Weight) und das Erscheinungsjahr (Year). Quelle: [4]

Nachteile der parallelen Koordinaten

Obgleich die parallelen Koordinaten mit der Anzahl skalieren, so besteht doch ein Problem bezüglich deren Anordnung. Da bei parallelen Koordinaten die Polylinie immer zwei benachbarte Attributen verbindet, kann man nur zwischen diesen Attributen Korrelationen feststellen. Wenn die Attribute also ungünstig angeordnet sind, sind Abhängigkeiten von Attributen nur schwer festzustellen.

Zudem besteht ein Problem bei kategorischen Daten, also Daten, die nur aufzählbare, aber nicht vergleichbare Werte haben (beispielsweise Automarken). Hier besteht die Gefahr, dass auf bestimmte Abhängigkeiten geschlossen wird, die nur aufgrund der Anordnung der Werte, die aufgrund der fehlenden Ordnung willkürlich ist, entstanden sind.

Des Weiteren besteht ein Überdeckungsproblem. Wenn mehrere Fälle für ein bestimmtes Attribut den gleichen Wert haben, so ist es dem Nutzer ohne weitere Interaktions- oder Hilfstechniken nicht möglich, festzustellen, welche der von dem Attribut ausgehenden Linien welcher eingehenden Linie zuzuordnen ist.

Verbesserungen für parallele Koordinaten

Manche der Nachteile der parallelen Koordinaten können mittels Interaktion oder weiterer Visualisierungstechniken teilweise ausgeglichen werden. Hierzu bieten sich folgende Verbesserungen an:

Brushing: Der Nutzer kann durch das Markieren eines oder mehrerer Fällen diese visuell hervorheben (beispielsweise indem diese ihre Farbe oder Dicke verändern) und diese somit einfacher verfolgen (siehe Abb. 2.15).

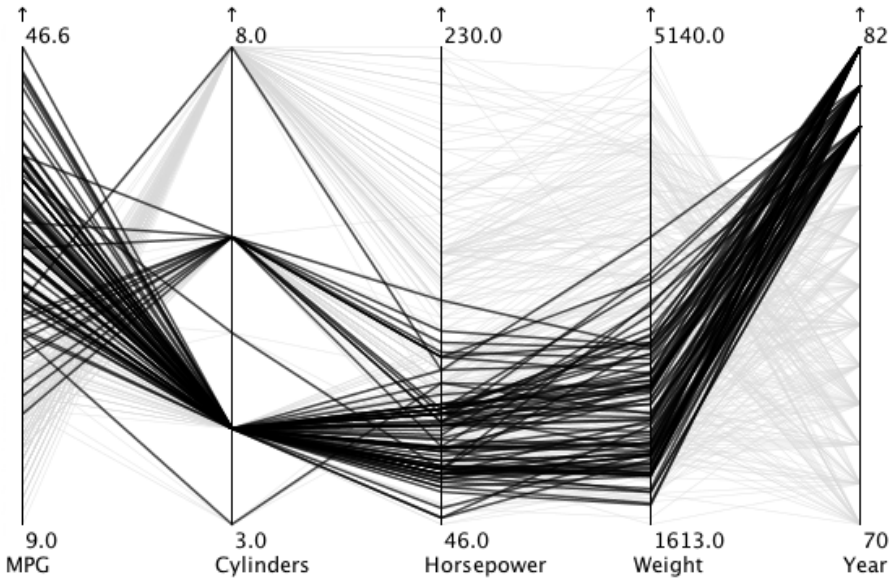

Abbildung 2.15.: Visualisierung des Datensatzes aus 2.14. Hier wurden mittels Brushing einige Datensätze besonders hervorgehoben. Quelle: [4]

Edge Bundling: Um das Problem des Visual Clutters zu beheben, ist es möglich, mehrere Kanten zusammenzufassen. Hierbei gehen zwar Informationen verloren, doch dafür verbessert sich die Übersicht und der generelle Kantenverlauf wird verdeutlicht.

Edge Splatting: Da auch mit Edge Bundling Kanten mit hoher Dichte auftreten können ist es möglich, dass diese mittels Edge Splatting farblich bezüglich ihrer Dichte, ähnlich einer Heatmap, kodiert werden.

2.6.3. Visual Information Seeking Mantra

Ben Shneiderman hat im Jahr 1996 das Visual Information Seeking Mantra [29] vorgestellt. Dieses sieht eine interaktionsgetriebene Verarbeitung von Daten der Informationsvisualisierung in drei Schritten vor:

1. Overview first: Zunächst sollen alle Daten angezeigt werden, damit sich der Nutzer einen Überblick über die Daten verschaffen kann.

2. Zoom and filter: Danach kann der Nutzer die Daten filtern und bestimmte Daten grafisch auswählen bzw. in diese hineinzoomen.

3. Details on demand: Um Visual Clutter zu verhindern werden zusätzliche Daten erst dann angezeigt, wenn der Nutzer diese explizit anfordert.

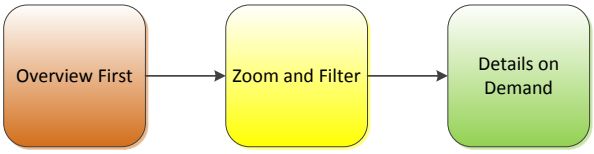

Abbildung 2.16.: Darstellung des Visual Information Seeking Mantra von Ben Shneiderman

Dieses Konzept lässt sich sowohl bei Visualisierungen, als auch beim Design einer grafischen Benutzeroberfläche (GUI) anwenden. Bei Visualisierungen werden die Informationen mittels Interaktionen gefiltert und Details beispielsweise mittels Tooltips angezeigt. Bei GUIs kann dieses Konzept auch auf Menüführungen übertragen werden.

2.6.4. Visualisierungspipeline

Um die vorhandenen abstrakten Daten zu visualisieren, sind zunächst einige Schritte notwendig, die im Folgenden anhand von Abb. 2.17 vorgestellt werden [2]. Diese Stellen eine mögliche Umsetzung des Visual Information Seeking Mantras dar.

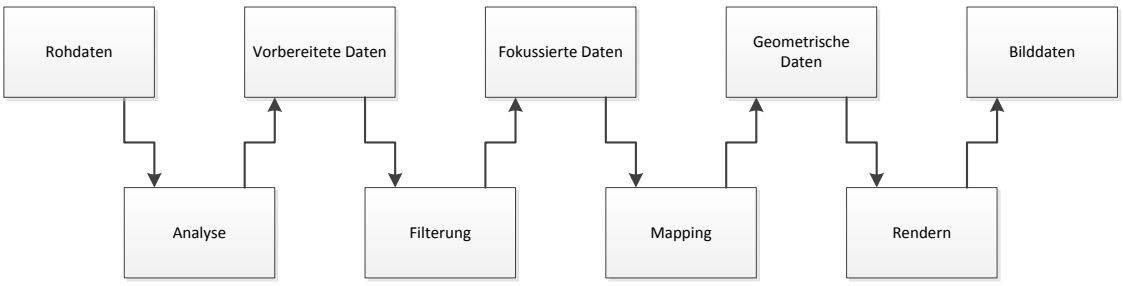

Abbildung 2.17.: Visuelle Darstellung der Pipeline zur Informationsvisualisierung. Die oberen Elemente repräsentieren die Daten und die unteren Elemente die Verarbeitungsschritte. Die Linien repräsentieren den Datenfluss.

1. **Analyse:** Die Daten werden zunächst vorgefiltert, um beispielsweise fehlende Werte zu interpolieren, Messungen bezüglich Ausreißern zu glätten oder fehlerhafte Messungen zu korrigieren.
2. **Filterung:** Die Datenmengen sind nach wie vor sehr groß und müssen zunächst gefiltert werden. Im Gegensatz zur Datenanalyse, bei der der Nutzer fast keinen Einfluss hat, werden die Filter hier vom Nutzer festgelegt.
3. **Mapping:** Beim Mapping werden den Daten geometrische Primitive wie Punkte, Linien oder ähnliches zugewiesen und deren Attribute wie Position, Dicke oder Größe vorgegeben.
4. **Rendern:** Beim Rendern werden die geometrischen Daten zu Bilddaten umgewandelt und angezeigt.

2.7. Visual Analytics

Mit jedem Tag nehmen die von verschiedenen Computersystemen generierten Datenmengen zu. Hierdurch ergibt sich schon seit einiger Zeit das Problem, dass sich diese riesigen Datenmengen mit den bisher verwendeten Techniken nicht mehr analysieren, geschweige denn visualisieren lassen.

Der nachfolgende Abschnitt stellt zunächst die Ursprünge der Visual Analytics vor. Daraufhin wird erklärt, worin sich die Visual Analytics von der Informationsvisualisierung unterscheidet und wie die Visualisierungspipeline für die Visual Analytics angepasst werden muss.

2.7.1. Ursprung der Visual Analytics

Im Zuge der Aufarbeitung der Anschläge am 11. September 2001 auf das World Trade Center wurde klar, dass die Geheimdienste der Vereinigten Staaten schon lange vor ihrer Durchführung Hinweise auf die Anschläge hatten. Diese gingen jedoch in den großen Datenmengen, welche die Geheimdienste ansammelten, unter.

Mit diesem Hintergrund veröffentlichten Thomas und Cook im Jahr 2005 das Buch *Illuminating the Path* [33] und begründeten damit das Feld der Visual Analytics. Der Ansatz war, die Informationen mittels Data Mining-Techniken zunächst zu filtern und dann einem Benutzer die Daten visuell zu präsentieren. Dieser konnte diese Daten dann von Hand manipulieren und das neue Ergebnis automatisiert weiterverarbeiten lassen.

2.7.2. Erweiterung des Visual Information Seeking Mantra

Da die Datenmengen in der Visual Analytics sehr groß sind, ist das Visual Information Seeking Mantra aus der Informationsvisualisierung nicht mehr praktikabel. Daher wurde es für die Visual Analytics abgeändert:

1. **Analyze First:** Da eine Übersicht nicht praktikabel ist werden die Daten zunächst vollautomatisiert gefiltert.

2. **Show the Important:** Es soll eine Übersicht über die gefilterten Daten angezeigt werden.

3. **Zoom, Filter and Analyze Further:** Die Informationen sollen über eine interaktive Visualisierung weiter gefiltert werden können und danach wieder mittels Techniken des Data Mining weiterverarbeitet werden.

4. **Details on Demand:** Es sollen detailliertere Informationen angezeigt werden, sofern es der Anwender wünscht.

2.7.3. Erweiterung der Visualisierungspipeline

Keim et. al. entwickelten im Jahr 2008 eine Visualisierungspipeline, die auf die Eigenschaften der Visual Analytics abgestimmt ist (siehe Abb. 2.18). Hierbei können die Daten mittels interaktiver Visualisierung gefiltert und mit Data-Mining-Techniken automatisiert weiterverarbeitet werden. Die beiden Möglichkeiten stehen in gegenseitiger Wechselwirkung. Das

aus der Manipulation entstehende Wissen kann zur weiteren Analyse der Daten von Mensch und Maschine genutzt werden und zur weiteren Verfeinerung der Ergebnisse wieder in die Pipeline geschickt werden.

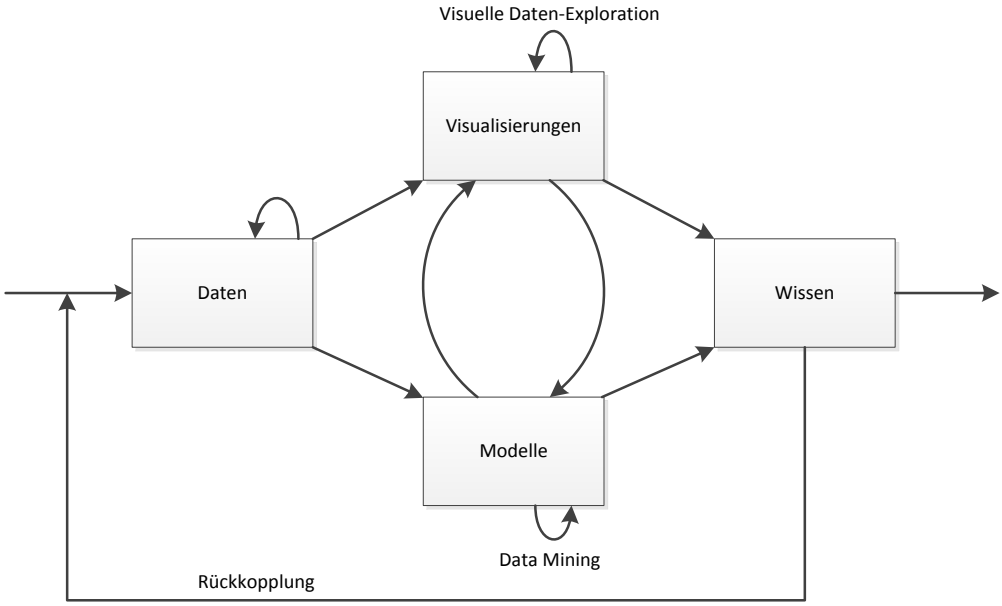

Abbildung 2.18.: Visuelle Darstellung für das Konzept der Visual Analytics [13].

3. Aufgabe und Lösungsansatz

Das folgende Kapitel behandelt das Szenario (Abschnitt 3.1) und die daraus entstehende Aufgabe dieser Arbeit (Abschnitt 3.2). Anschließend wird ein Lösungsansatz vorgestellt (Abschnitt 3.3), der neue visualisierungsbasierte Analysetechniken für Eye-Tracking-Daten skizziert.

3.1. Szenario

Aufgrund der zunehmenden Nutzung von Eye-Trackern in Wirtschaft, Industrie und Forschung steigt der Bedarf an Analysemethoden und Visualisierungstechniken für Eye-Tracking-Daten. Dies ist insbesonders in den Kognitionswissenschaften, der Visualisierung und der Mensch-Computer-Interaktion bzw. der Mensch-Maschine-Interaktion der Fall, da es in diesen Disziplinen eine zunehmende Verbreitung von durch Eye-Tracking-gestützten Benutzerstudien gibt. Auch in der Automobilindustrie nimmt dabei das Interesse am Blickverhalten des Fahrers zu. Da bei Eye-Tracking-Aufnahmen große Datenmengen entstehen, ist eine der aktuell größten Herausforderungen, diese Daten effizient auszuwerten. Heutige Filter glätten zumeist nur die Blickaufnahmen und fassen Blicke zusammen. Zudem skalieren die derzeit am weitesten verbreiteten Visualisierungstechniken für Eye-Tracking-Daten, Heat Maps und Scan-Paths, bei zunehmender Probandenanzahl und zunehmender Aufnahmelänge nur begrenzt. Dabei ist die Vermeidung von Visual Clutter, der aufgrund von häufigen Überdeckungen entsteht, eine der größten Herausforderungen.

Diese Diplomarbeit soll daher visualisierungsbasierte Analysetechniken entwerfen, die sowohl die Analysemöglichkeiten für Eye-Tracking-Daten erweitern, als auch die daraus entstehenden Daten so visualisieren, dass die Daten besser skalieren.

3.2. Aufgabe

Das Ziel dieser Diplomarbeit ist die Entwicklung eines Visualisierungskonzepts, um die Analyse der Eye-Tracking-Daten mittels der Techniken der Visual Analytics zu vereinfachen. Zudem sollen anhand der aufbereiteten Daten mit Hilfe der Parallel Scan-Paths über mehrere Probanden hinweg wiederkehrende Muster erkannt werden können. Das entwickelte Konzept soll danach in einem Prototyp implementiert und dessen Funktionalität anhand von zwei Szenarien evaluiert werden. Um diese Ziele zu erfüllen, ist diese Diplomarbeit in folgende Aufgaben unterteilt:

- Eine Recherche zu existierenden Arbeiten (siehe Kapitel 4)
- Die Erarbeitung eines Analysekonzepts für die visuelle Analyse von Eye-Tracking-Daten (siehe Kapitel 5)

- Die Implementierung des Konzepts in einem Prototyp (siehe Kapitel 6)
- Eine Evaluation des Prototyps anhand von zwei Szenarien (siehe Kapitel 7)

3.3. Lösungsansatz

Analog zum Konzept der Visual Analytics ist diese Arbeit zweigeteilt:

1. Zunächst wird ein Visualisierungskonzept für den Analyseschritt der Daten entwickelt. Dabei sollen die Daten stufenweise durch vom Analysierenden konfigurierbare Filter aufbereitet werden. Um die Filter erweiterbar zu halten, ist es denkbar, die einzelnen Filter als Plug-ins zur Verfügung zu stellen. Die Filter sollten dabei zu einem späteren Zeitpunkt nachkonfigurierbar sein, um den „User in the loop“, der eine der Kernkomponenten der Visual Analytics darstellt, zu realisieren.

2. Anschließend werden die Daten an die Visualisierung übergeben. In dieser sollen mittels Interaktionstechniken wie Zoomen und Brushen die Daten besser exploriert werden können. Zudem ist es denkbar, die Darstellung der Daten zwecks besserer Vergleichbarkeit unterschiedlich aufzubereiten und entsprechend darzustellen. Im Anschluss soll dem Analysierenden die Möglichkeit gegeben werden, anhand der aus der Visualisierung gewonnenen Erkenntnisse die Filter im Analyseschritt anzupassen.

Da Eye-Tracker teilweise sehr große Datenmengen generieren, gibt es einige Aspekte, die in dieser Arbeit zu jedem Zeitpunkt zu beachten sind:

Performanz: Ein wichtiger Aspekt bei der Analyse von großen Datensätzen mittels Interaktion und Visualisierung ist eine schnelle Repräsentation der Veränderungen des Datensatzes an den Nutzer.

Einhaltung des Visual Information Seeking Mantra: Um die unnötige Erzeugung von Visual Clutter zu vermeiden ist die Einhaltung des Visual Information Seeking Mantra von Ben Shneiderman ein wichtiger Aspekt.

Nachvollziehbarkeit: Die Effekte, die der Analysierende durch seine Interaktionen verursacht, sollen nachvollziehbar sein. Dies betrifft nicht notwendigerweise die Funktionsweise bestimmter Funktionen im Detail, sondern deren faktischer Auswirkungen auf den Datensatz bzw. die Visualisierung.

4. Existierende Arbeiten

Dieses Kapitel stellt Arbeiten vor, deren Zielsetzung dieser Arbeit ähnlich sind oder deren Techniken für diese Arbeit relevant sind. Zunächst werden die Parallel Scan-Paths vorgestellt, auf denen die Visualisierung des Konzepts basiert (Abschnitt 4.1). Daraufhin wird das Circular Heat Map Transition Diagram behandelt, dessen Ziel die Visualisierung der Zusammenhänge einzelner Areas of Interest ist (Abschnitt 4.2). Anschließend wird eine Übersicht über die Funktionsweise des eSeeTrack-Programms gegeben, dessen Ziel die visuelle Darstellung von kognitiven Abläufen ist (Abschnitt 4.3). Einen ähnlichen Ansatz verfolgt das Programm EyePatterns, mit dem Unterschied, dass die Abläufe mittels Textanalyse durchgeführt werden (Abschnitt 4.4). Zuletzt werden die Space-Time Cubes vorgestellt (Abschnitt 4.5). Deren Ziel ist die Visualisierung von Fixationen auf Videodaten. Dabei werden die Fixationen über die Zeit visualisiert. Die Arbeit nutzt des Weiteren Techniken für vollautomatisiertes Clustering.

4.1. Parallel Scan-Paths

Das Konzept der Parallel Scan-Paths [25], kurz PSPs, ist die Zusammenfassung von Fixationen innerhalb von Areas of Interest, um Blickmuster einfacher zu erkennen. Zudem skaliert die Technik deutlich besser über die Zeit einer Eye-Tracking-Aufnahme als vergleichbare Techniken, die ebenfalls Blickmuster visualisieren. Bei den Parallel Scan-Paths werden dabei sämtliche AOIs auf der Abszissenachse aufgelistet. An jeder AOI wird eine vertikale Achse aufgezogen, womit später die Zugehörigkeit einer Fixation zu der AOI visualisiert wird. Der zeitliche Verlauf ist auf die Ordinatenachse aufgetragen. Es gibt dabei mehrere Modi, wie die Daten in diesem System aufgetragen werden können: Das Gaze Duration Sequence Diagram, das Fixation Point Diagram und das Gaze Duration Distribution Diagram. Im Folgenden werden diese Modi näher beschrieben.

4.1.1. Gaze Duration Sequence Diagram

Beim Gaze Duration Sequence Diagram wird eine Eye-Tracking-Aufnahme in Form einer Linie dargestellt. Dabei verläuft diese entlang der Ordinatenachse. Sie wechselt immer auf die vertikale Achse der AOI, die der Proband während des Zeitpunkts der Aufnahme fokussiert hat. Eine Darstellung dieser Technik findet sich in Abbildung 4.1. Hierbei wurden auf einer Achse vier Areas of Interest definiert. Diese vier AOIs finden sich auf der Rechtsachse der PSPs wieder. Auf dem PSP ist die Aufnahme eines Probanden abgetragen.

4.1.2. Fixation Point Diagram

Das Fixation Point Diagram ähnelt dem soeben vorgestellten Gaze Duration Sequence Diagram. Hier werden jedoch die einzelnen Fixationen durch das Auftragen einzelner Punkte

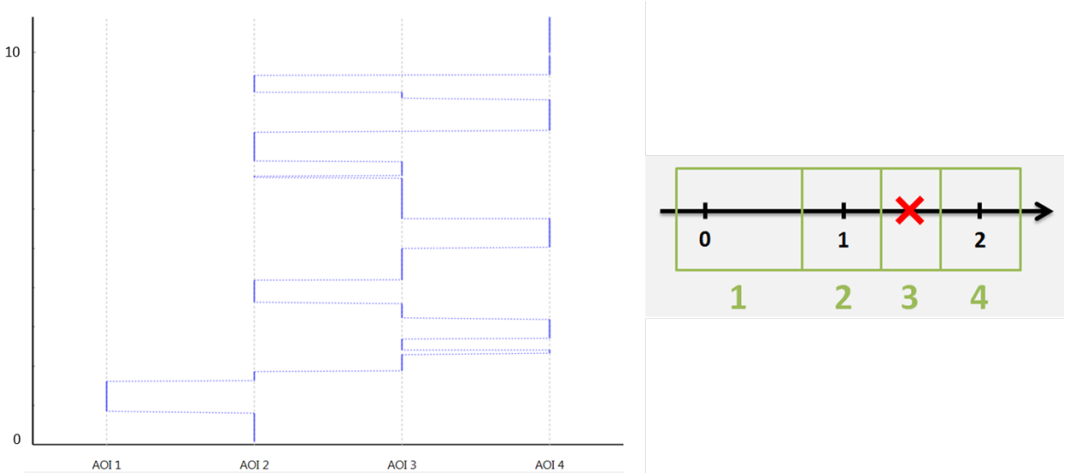

Abbildung 4.1.: Darstellung des Gaze Duration Sequence Diagram. Auf der rechten Seite wird der mit AOIs annotierte Stimulus gezeigt. Zudem wird eine Scanpath-Aufnahme mit vier Probanden dargestellt. Auf der linken Seite ist das dem Scanpath entsprechende Fixation Point Diagram für dieselben vier Probanden zu sehen. Quelle: [25]

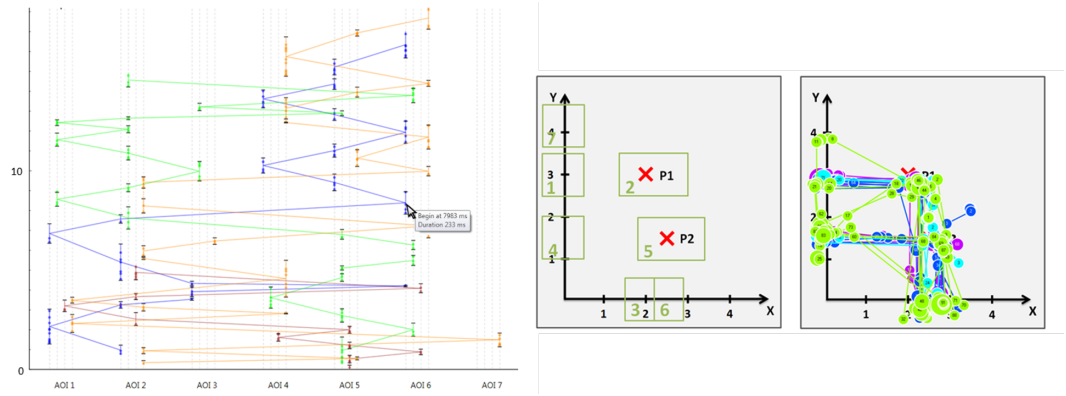

Abbildung 4.2.: Darstellung des Fixation Point Diagram. Auf der rechten Seite wird der mit AOIs annotierte Stimulus gezeigt. Auf der linken Seite ist das Fixation Point Diagram für einen Probanden zu sehen. Quelle: [25]

dargestellt. Zudem trifft die Linie, die den Scanpath repräsentiert, die AOI-Achse jeweils nur an einem Punkt, der die Hälfte der Aufenthaltszeit innerhalb der AOI repräsentiert. Danach springt der Verlauf zur nächsten AOI. Diese Darstellung kann hilfreich sein, um zu erkennen, ob die Probanden immer die gleiche Stelle der Area of Interest betrachtet haben oder ob sie häufig innerhalb der AOI gesprungen sind, beispielsweise um mehr Details der Area of Interest zu erkennen. Ein Beispiel dieser Darstellungsmethode findet sich in Abbildung 4.2. Auf der rechten Seite befinden sich der Stimulus mit sechs definierten AOIs und die Scanpath Visualisierung der Aufnahme mit vier Probanden. Links wird ein Fixation Point Diagram derselben Aufnahme mit den gleichen Probanden dargestellt.

4.1.3. Gaze Duration Distribution Diagram

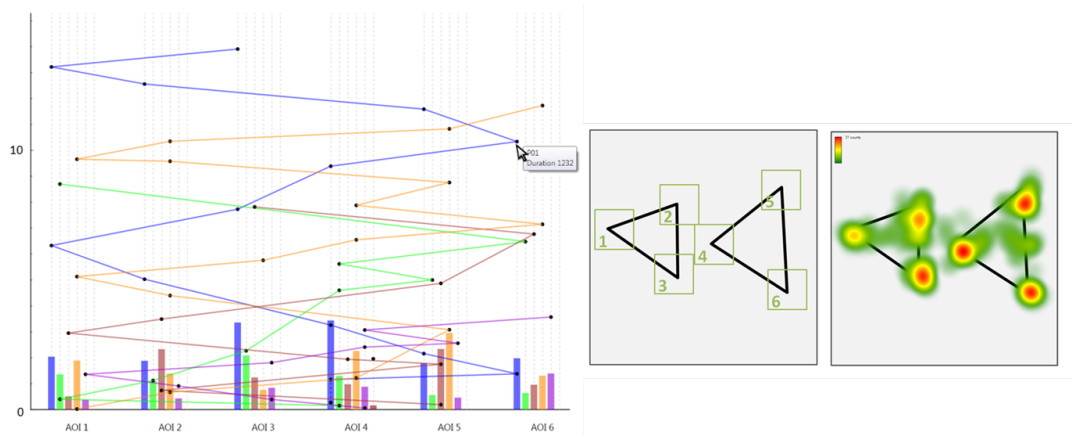

Abbildung 4.3.: Darstellung des Gaze Duration Distribution Diagram mit mehreren Probanden. Die Aufenthaltszeit wird über den Labels der AOIs für jeden Probanden dargestellt. Quelle: [25]

Das Gaze Duration Distribution Diagram stellt weitere Informationen über die Aufenthaltsdauer innerhalb einer AOI zur Verfügung. Die Darstellung des Verlaufs ähnelt derer des Fixation Point Diagram, doch werden hier zusätzliche Informationen bezüglich der Verweildauer innerhalb einer Area of Interest oberhalb der Abzissenachse dargestellt. Ein Beispiel eines Gaze Duration Distribution Diagram ist in Abbildung 4.3 dargestellt. Hier wird die Aufenthaltszeit innerhalb der AOIs durch Histogramme über den AOI-Labels für jeden Probanden dargestellt.

4.1.4. Einschränkungen der PSPs

Obgleich die PSPs eine deutlich bessere Skalierung von Eye-Tracking-Daten bezüglich ihres zeitlichen Verlaufs bieten, so skalieren sie doch nur begrenzt mit der Anzahl der AOIs. Zudem wird es bei langen Aufnahmen zunehmend schwierig, Probanden miteinander zu vergleichen und Muster zu erkennen. Zudem bieten PSPs keinerlei Interaktionsmöglichkeiten, sodass zu jedem Zeitpunkt alle Daten angezeigt werden.

4.2. Circular Heat Map Transition Diagram

Das Circular Heat Map Transition Diagram nutzt als Datengrundlage einen mit AOIs annotierten Stimulus und visualisiert daraufhin die Zusammenhänge der Areas of Interest. Dabei werden die AOIs auf einem Ring dargestellt und Übergänge zwischen den AOIs werden mittels Pfeilen visualisiert. Dabei steht die Pfeildicke für die Anzahl der Transitionen zwischen den AOIs, welche zusätzlich durch ein Label an dem Pfeil angegeben wird. Die Farbe der AOI auf dem Diagramm steht für die Anzahl der Fixationen innerhalb der AOI. Die Zusammenhänge können sowohl zeitabhängig als auch zeitunabhängig dargestellt werden.

Eine beispielhafte Darstellung der Visualisierung ist in Abbildung 4.4 gegeben. Dabei ist das Circular Heat Map Transition Diagram für einen Probanden sowohl zeitabhängig als auch zeitunabhängig dargestellt. Durch die zeitunabhängige Darstellung kann ein Gesamtüberblick gewonnen werden und durch die zeitabhängige Darstellung ist es möglich, den Blickverlauf des Probanden nachzuvollziehen. Die Diagramme wurden vor allem zur Visualisierung der Scanpaths einzelner Probanden einer Eye-Tracking-Aufnahme entworfen und nicht um mehrere Probanden miteinander zu vergleichen.

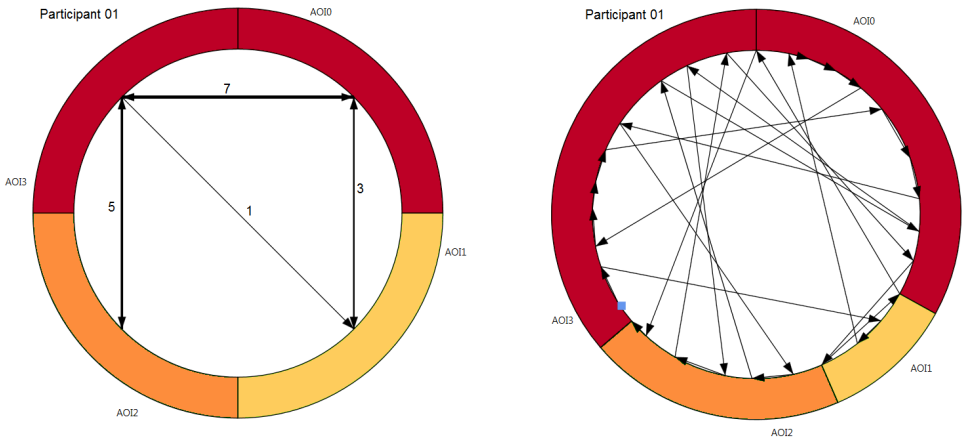

Abbildung 4.4.: Darstellung des Circular Heat Map Transition Diagram. Auf der linken Seite werden die Transitionen zwischen den verschiedenen AOIs mittels Verbindungslinien dargestellt. Die Dicke gibt dabei die Übergangshäufigkeit an. Auf der rechten Seite sind die Übergänge zwischen den AOIs über die Zeit aufgetragen. Dadurch ist es möglich, den Blickverlauf des Probanden nachzuvollziehen

4.3. eSeeTrack

Das Ziel von eSeeTrack [36] ist die Visualisierung von wahrscheinlichen Blickmustern bei Eye-Tracking-Studien. Dazu wird ein mit AOIs annotierter Stimulus genutzt. Der Analysierende kann sich mit einer Zeitleiste einen ersten Überblick über die fixierten Areas of Interest verschaffen und dann mit Hilfe einer Baumdarstellung der wahrscheinlichsten AOI-Transitionen die Daten weiter explorieren. Der Hauptfokus dieser Arbeit liegt sowohl in der Visualisierung des Blickablaufs als auch in der Vergleichbarkeit von Probanden. Um dieses Ziel zu erreichen, ist die Programmoberfläche in vier Abschnitte unterteilt: Die Zeitleiste, die erweiterte Zeitleiste, die Baumvisualisierung und den Kontrollbereich. Die gesamte Programmoberfläche wird in Abbildung 4.5 gezeigt, in der auch die genannten vier Abschnitte markiert sind. Die Abschnitte werden nachfolgend näher erläutert.

Abbildung 4.5.: Gesamtdarstellung des eSeeTrack-Tools. Das Tool ist in vier Bereiche aufgeteilt. (a) zeigt die Zeitleiste. (b) ist die erweiterte Zeitleiste. Hier können optional Vorschaubilder der betrachteten Szenen eingeblendet werden. (c) ist die Darstellung der Baumvisualisierung. (d) ist der Datenkontrollbereich. Hier können die verarbeiteten Daten gefiltert werden. Quelle: [36]

4.3.1. Timeline

In der Zeitleiste werden für einen oder mehrere Probanden die Aufenthalte in den verschiedenen AOIs visualisiert. Jede AOI wird dabei durch eine andere Farbe repräsentiert und die Breite einer Farbe entspricht der Aufenthaltsdauer innerhalb der AOI. Der Hauptzweck der Zeitleiste ist die Gewinnung der Übersicht über alle vorliegenden Daten.

4.3.2. Detailed Timeline

Um dem Analysierenden die Möglichkeit zu geben, einzelne Bereiche näher zu untersuchen, kann man diese selektieren und in der Detailed Timeline näher betrachten. Hierdurch ist es möglich, einzelne Bereiche der Aufnahme deutlich detaillierter zu untersuchen und Ähnlichkeiten im Blickverhalten verschiedener Probanden einfacher festzustellen. In der detaillierten Zeitleiste können zudem optional Vorschaubilder der betrachteten AOIs angezeigt werden.

4.3.3. Tree Visualization

Nachdem die Timeline eine Übersicht gegeben hat, kann der Analysierende einen beliebigen Aufenthalt innerhalb einer AOI selektieren. Hierdurch wird in der Baumvisualisierungskom-

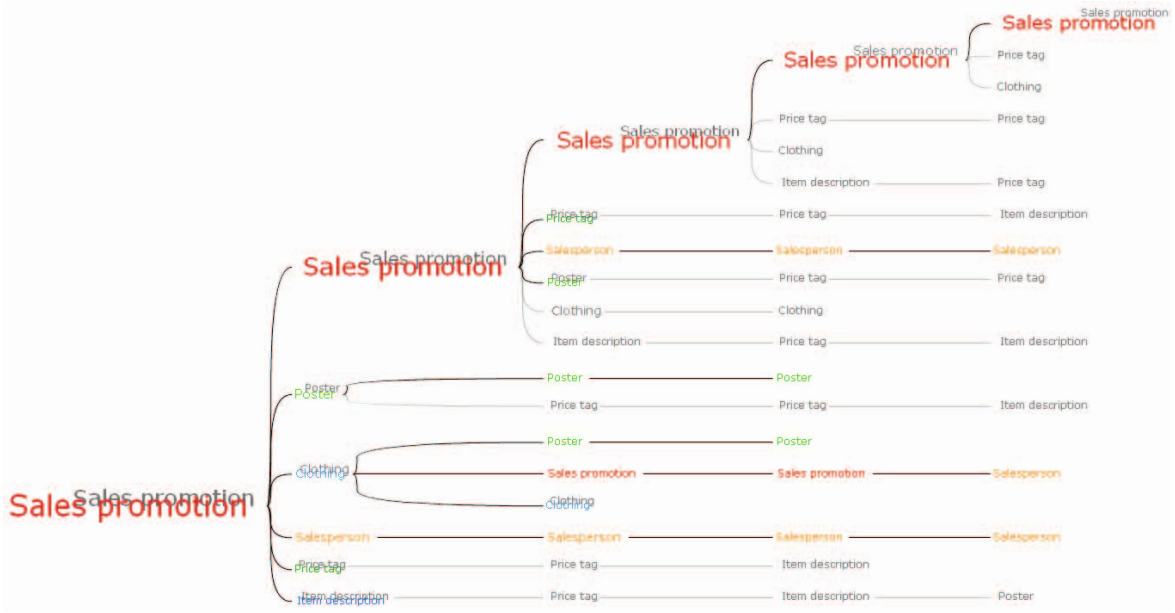

Abbildung 4.6.: Darstellung der Baumvisualisierung von Eye-Tracking-Aufnahmen in zwei Einkaufshäusern für Kleidung. Die farbigen Labels repräsentieren AOIs, die in dem ersten Kaufhaus angesehen wurden, wogegen graue Labels AOIs repräsentieren, die in dem zweiten Laden angesehen wurden. Quelle: [36]

ponente der Blick als Wurzel eines Word-Tree-ähnlichen Baumes angelegt. Jedes Element in dem Baum hat ein Label, das den Namen der AOI bzw. des Tags enthält, das ausgewählt wurde. In eSeeTrack kann jede Gruppe die AOIs unabhängig voneinander benennen. Diese Benennung wird als Tag bezeichnet. Abhängig von den wahrscheinlichsten Übergängen von der gewählten AOI zu anderen AOIs werden Kindknoten angelegt. Jeder Kindknoten hat zwei Labels:

- Das Hauptlabel trägt den Namen des Tags der Gruppe, aus der die Wurzel-AOI gewählt wurde. Die Schriftgröße hängt von der Wahrscheinlichkeit ab, mit der ein Übergang zu dem Kindknoten eintritt. Zudem ist dieses Label farbig.
- Ein kleineres graues Label, das den häufigsten Tagnamen aller anderen Gruppen enthält.

Die maximale Tiefe des Baums wurde in dem Paper auf fünf festgelegt. Eine Darstellung der Baumvisualisierung findet sich in Abbildung 4.6. Hier wurden die Eye-Tracking-Aufnahmen von zwei Einkaufshäusern für Kleidung miteinander verglichen. Dabei sind die farbigen Labels die des ersten und die grauen die des zweiten Einkaufshauses.

4.4. EyePatterns

Mit Hilfe des Programms EyePatterns [38] ist es ebenfalls möglich, das Blickverhalten von Probanden miteinander zu vergleichen. Hier werden ebenfalls AOIs für einen Stimulus definiert.

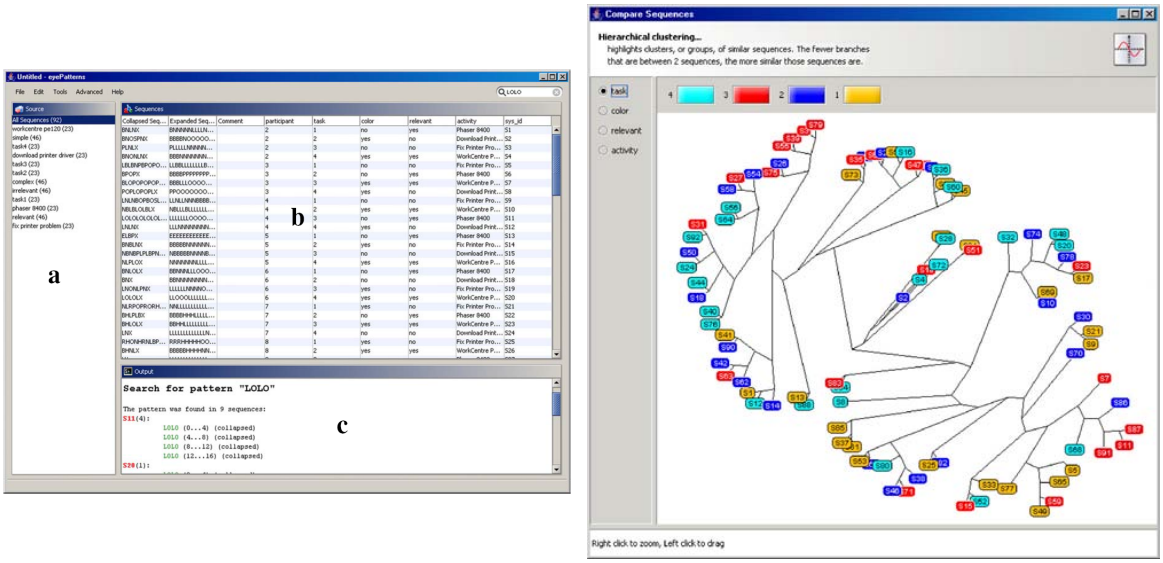

Abbildung 4.7.: Links: Darstellung der Ergebnisse der Datenaufbereitung. Es werden sowohl die ungekürzten als auch die gekürzten Strings angezeigt. Rechts: Darstellung der geclusterten Baumstruktur von EyePatterns. Die Datensätze, die während des Analyseschrittes als ähnlich eingestuft wurden, haben in dem Baum eine höhere räumliche Nähe, wodurch diese auch visuell geclustert werden. Quelle: [38]

Jeder AOI wird des Weiteren ein Zeichen zugewiesen, das die AOI eindeutig identifiziert. Dann wird für jeden Probanden ein String angelegt, der dem Verlauf der Fixationen entspricht. Hierbei wird in einem einstellbaren Intervall geprüft, welche Position auf dem Stimulus der Proband fixiert hat. Wenn diese Position innerhalb einer AOI liegt, wird deren Zeichen an das Ende des Strings für den Probanden angehängt. Anschließend können die entstandenen Strings miteinander verglichen werden. Die Strings können optional auch gekürzt werden, sodass Strings wie „AABCA“ zu „ABCA“ zusammengefasst werden. Der rechte Teil von Abbildung 4.7 zeigt die daraus entstehende Übersicht. In dieser kann man auch nach auftretenden Mustern suchen. Zur Berechnung der Ähnlichkeit der Strings stellt EyePatterns die drei Algorithmen von Edit und Levenshtein [16], Needleman und Wunsch [20] sowie von Smith und Waterman [30] zur Verfügung. Nachdem die Ähnlichkeit der Strings berechnet wurde, können diese durch die Visualisierung von EyePatterns wie in Abbildung 4.7 links als geclusterter Baum dargestellt werden. Dadurch wird dem Analysierenden eine Übersicht darüber gegeben, welche Blickverläufe der Probanden ähnlich sind.

4.5. Space-Time Cubes

Das Ziel der Space-Time Cubes [15], kurz STC, ist es, Eye-Tracking-Daten von dynamischen Stimuli wie Videodaten besser zu visualisieren. Hierzu wird der Space-Time Cube eingeführt. Die Achsen des Würfels entsprechen der Zeitachse des Stimulus und den Fixationsdaten auf einer Ebene für die verbleibenden zwei Dimensionen. Um den aktuellen Zeitpunkt zu

visualisieren, ist in den Würfel eine Ebene eingezogen, die den Stimulus zu einem bestimmten Zeitpunkt wiedergibt. Der Nutzer kann dabei in der Zeitachse frei navigieren. In dem Würfel können zudem die Fixationen der Probanden als einzelne Punkte in dem Würfel eingeblendet werden. Alternativ ist es zudem möglich, die Fixationen zu Clustern zusammenzufassen und den Verlauf der „Fixationscluster“ anzeigen zu lassen. Es ist außerdem möglich, eine zusätzliche Miniaturprojektion der beiden Ebenen einblenden zu lassen, um eine bessere Übersicht über die Daten zu gewinnen. Auf der Stimulusebene werden für den gewählten Zeitpunkt entweder eine Bee-Swarm-Visualisierung oder die geclusterten Fixationsdaten angezeigt. Das Tool ist dazu in der Lage, das Clustering der Fixationsdaten vollautomatisch durchzuführen. Ein Screenshot des Programms ist in Abbildung 4.8 gegeben.

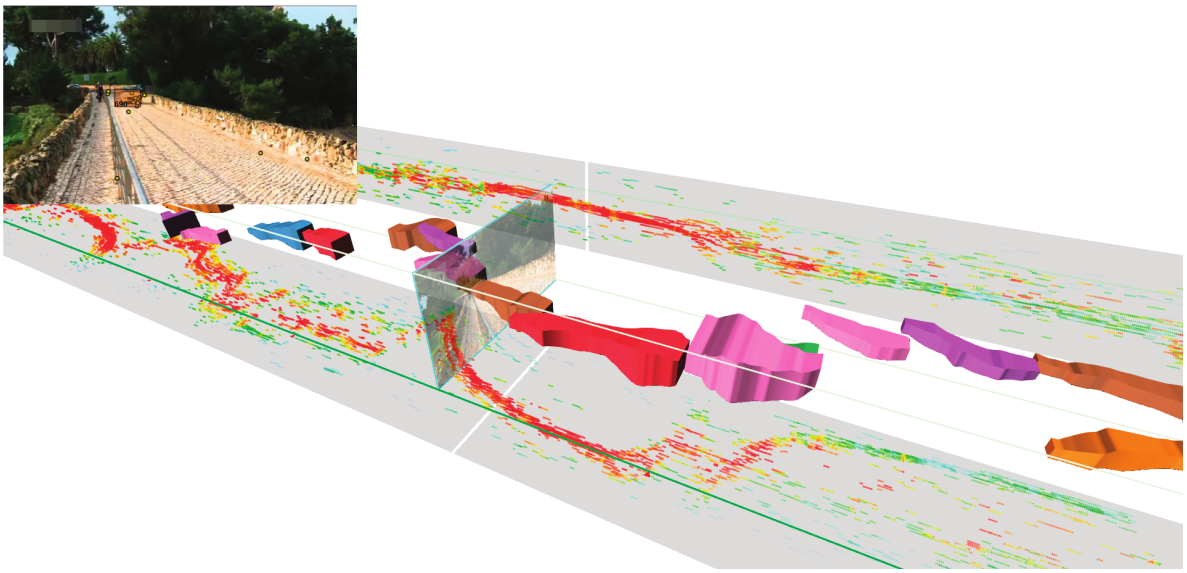

Abbildung 4.8.: Darstellung des Space-Time Cube. Auf dem Stimulus ist eine Straße mit Fahrzeug zu sehen. Der Space-Time Cube stellt die Fixationsdaten der Probanden dabei geclustert dar. Zudem sind zusätzliche Ebenen eingeblendet, die den räumlichen Verlauf der Daten zwecks einer besseren Übersicht auf die jeweils eingeblendete Ebene projizieren. Quelle: [15]

5. Konzept

In diesem Kapitel wird das Konzept für die Eye-Tracking-Analyse-Pipeline-gestützten PSPs, kurz ETA-based PSPs, vorgestellt.

Zunächst wird das allgemeine Konzept dieser Arbeit beschrieben (Abschnitt 5.1). Auf dessen Basis wird das Konzept der Visual Analytics zu einer Pipeline erweitert, was der darauf folgende Abschnitt erläutert (Abschnitt 5.2).

Anschließend werden die technischen Voraussetzungen beschrieben, unter denen die Fixationsdaten des Eye-Trackers mit AOIs kombiniert werden können (Abschnitt 5.3). Dies dient im weiteren Verlauf dazu, dass die Daten später effizient gefiltert und dann mit Hilfe der PSPs analysiert werden können. Dazu wird zunächst diskutiert, warum die Verwendung einer Datenbank sinnvoll ist und welche Daten und Relationen diese enthalten muss. Daraufhin werden verschiedene Ansätze vorgestellt, mit denen die Areas of Interest erzeugt werden können, damit diese anschließend mit den Fixationsdaten von Eye-Tracking-Studien verbunden werden können.

Daraufhin folgen die einzelnen Abschnitte der angepassten Visual Analytics Pipeline. Dazu werden zunächst die Möglichkeiten der Vorfilterung der Daten aufgezeigt (Abschnitt 5.4). Anschließend wird das Analysekonzept vorgestellt (Abschnitt 5.5). Ziel ist hier die automatisierte Verarbeitung der Daten. Dabei wird der Nutzen von verschiedenen Algorithmen zur Ermittlung der Stringähnlichkeit diskutiert. Im Anschluss werden weitere Möglichkeiten zur Ermittlung der Ähnlichkeit von AOI-gebundenen Scanpaths vorgestellt und es wird erläutert, warum eine Gruppierung von ähnlichen Blickpfaden sinnvoll ist. Zum Abschluss der Analyse werden visuelle Ergänzungen, wie eine Annotation der Filterübergange zwischen den verschiedenen Filtern, diskutiert. Nach dem Analysekonzept wird die Möglichkeit einer Filterung durch vom Benutzer vorgegebene statistische Nebenbedingungen mittels Optimierungsverfahren vorgestellt. Im Anschluss folgt das Konzept für die interaktive Visualisierung der Parallel Scan-Paths (Abschnitt 5.6). Hierin werden die angezeigten Daten sowie deren Metadaten wie Szenen- und AOI-Zugehörigkeit gezeigt und deren potenzielle Interaktionsmöglichkeiten diskutiert. Daraufhin wird das Potenzial einer Erkennung von wiederkehrenden Blick- und damit Aufmerksamkeitsmustern verdeutlicht. Des Weiteren wird die Möglichkeit der Einbindung von externen Daten und deren unterschiedliche Typen behandelt. Zum Abschluss der interaktiven Visualisierung wird auf die optionale Anzeige weiterer Statistiken der Eye-Tracking-Daten zur Unterstützung der Visualisierung eingegangen.

Anschließend wird die optionale Rückkopplung der in der Visualisierung gefilterten Daten, also die Nutzung der neuen Erkenntnisse über die Daten durch eine Veränderung der Filter und die erneute Nutzung der gefilterten Daten, mittels Interaktionen auf den visualisierten Daten diskutiert (Abschnitt 5.7).

Zuletzt wird auf die Einschränkungen des Konzepts eingegangen (Abschnitt 5.8).

5.1. Allgemeines Konzept

Eye-Tracker produzieren während einer Studie eine große Menge an Daten. Trotz Vorverarbeitung durch den Eye-Tracker selbst verbleiben große Datenmengen, die der Studienleiter manuell auswerten muss. Durch die Verwendung von Areas of Interest kann eine weitere Abstraktionsebene eingeführt werden. Durch die Aufenthaltsdauer innerhalb von AOIs und den Übergängen zwischen den AOIs können weitere Informationen aus den Eye-Tracking-Daten gewonnen werden. Die aufeinanderfolgenden Übergänge, oder Transitionen, eines Probanden zwischen den AOIs wird in dieser Arbeit als Transitionspfad beschrieben. Entsprechend werden in dieser Arbeit die Fixationen, die die Start- und Endfixationen der einzelnen Teile des Transitionspfades darstellen, als Transitionspfadelemente bezeichnet. Hierdurch wird eine klarere Abgrenzung zu den Blickpfaden der Scan-Paths ermöglicht. Die Parallel Scan-Paths (PSP)[25] nutzen die AOIs, um bei der Auswertung einer Eye-Tracking-Studie ein weiteres Werkzeug zur Verfügung zu stellen, mit dem man den Transitionspfad, dem ein Teilnehmer gefolgt ist, besser nachvollziehen und bezüglich auftretender Muster analysieren kann.

Die Parallel Scan-Paths skalieren besser über die Zeit als etablierte Visualisierungstechniken wie Heat Maps oder Scan-Paths. Dennoch ist es schwer, mehr als zwei bis drei Probanden bezüglich ihres Blickverhaltens zu vergleichen. Um diese Einschränkungen zu beheben, sieht diese Arbeit eine Datenverarbeitung nach Vorbild des VA-Konzepts von Keim et al. [13] vor. In Folge dessen wird die Evaluation der Eye-Tracking-Daten in drei Schritte unterteilt: Die Vorverarbeitung, die Analyse und Filterung der Daten und die Visualisierung der gefilterten Daten. Hierzu wird das VA-Konzept zu einer Pipeline umstrukturiert. Diese Pipeline wird später ausführlicher beschrieben und wird als Eye-Tracking-Analysis-Pipeline, kurz ETA-Pipeline, bezeichnet. Unter Berücksichtigung der VA-Pipeline werden die PSPs sowohl im Datenmodell als auch in der Visualisierung erweitert.

Vor der Visualisierung findet eine maschinelle Analyse statt, welche die Daten nach Kriterien, die von dem Studienanalysierenden festgelegt werden, verarbeitet. Ziele dieser Analyse sind

1. Filtern,
2. Gruppieren oder
3. Anpassen der Daten

Die Visualisierung der Daten mit Hilfe der PSPs wird um mehrere Interaktionsmöglichkeiten erweitert. So soll unter anderem

1. das Zoomen von zeitlichen Abschnitten,
2. das Einklappen zeitlicher Abschnitte,
3. das Zoomen und Einklappen von AOIs sowie
4. das Umschalten der Detailstufe von AOIs

ermöglicht werden.

5.2. Erweiterung zu einer Pipeline

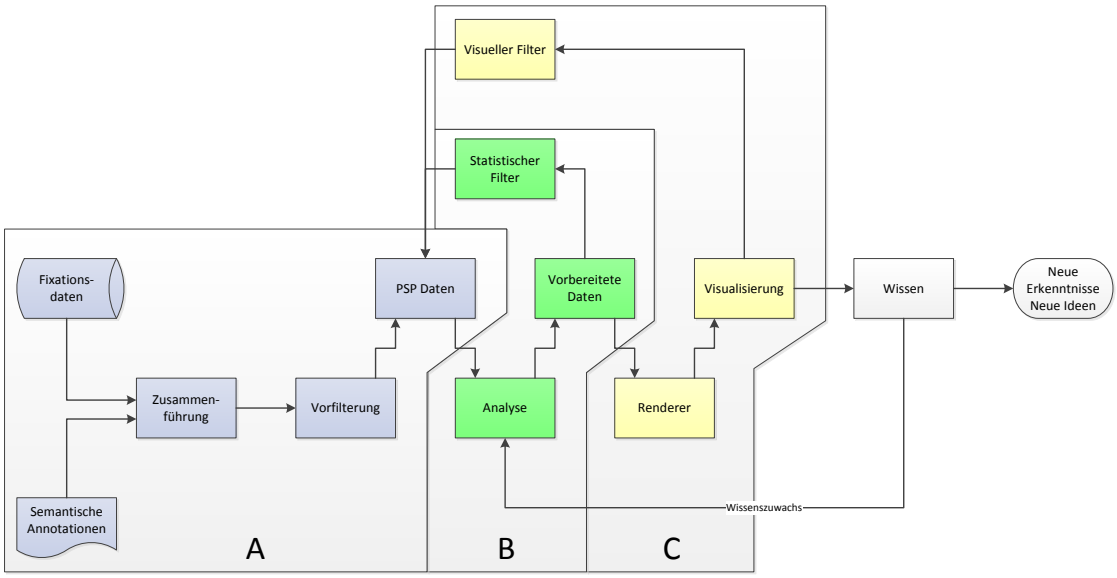

Abbildung 5.1.: Visuelle Darstellung der Eye-Tracking-Analyse-Pipeline (ETA-Pipeline). Die Daten werden in Block A vorverarbeitet. Die für die Analyse der Daten und deren Filterung und Gruppierung relevanten Abschnitte der Pipeline sind in Block B zusammengefasst. Block C stellt die Visualisierung und Interaktionsmechanismen mit der Visualisierung zur Verfügung.

Die Konzept der Visual Analytics ist sehr generisch und muss daher modifiziert werden, um die Anforderungen der Analyse einer Eye-Tracking-Studie zu erfüllen. Daher wird in dieser Arbeit das Konzept zu einer Pipeline erweitert, im Folgenden Eye-Tracking-Analyse-Pipeline, kurz ETA-Pipeline genannt. Diese ist in Abbildung 5.1 zu sehen.

Zunächst müssen sowohl die Fixationsdaten für einen bestimmten Stimulus von einem Eye-Tracking-System zur Verfügung gestellt werden. Zudem werden Informationen bezüglich der Position und Form der AOIs für den Stimulus definiert. Daraufhin werden die Fixationsdaten mit den Areas of Interest verbunden. Hierbei werden Fixationen, die in keiner AOI liegen, verworfen. Die dabei entstehenden Daten werden hier als PSP-Daten bezeichnet. Der Matchingschritt ist nur einmal notwendig.

Anschließend werden die Daten mittels vollautomatischer Algorithmen, welche die Daten anhand bestimmter Metriken mit einstellbaren Parametern selbstständig filtern, analysiert. Das Ziel ist hier eine Gruppierung und Filterung der Daten. Die entstandenen Daten können entweder mittels einer statistischen Filterung wieder in die Analysepipeline zurückgeführt oder visualisiert werden. Die statistische Filterung kann zusätzlich Nebenbedingungen enthalten, sodass Daten, die statistisch zu stark von einem Erwartungswert abweichen, gefiltert werden.

An den Analyseschritt schließt der Renderer, der in diesem Fall Parallel Scan-Paths generiert, an. Hierdurch wird die Visualisierung der Daten ermöglicht. Die Visualisierung besitzt Interaktionsmöglichkeiten, wie das manuelle Filtern von AOIs oder Zeitabschnitten. Hierbei können Teile der AOIs und bestimmte Zeitabschnitte verkleinert oder komplett ausgeblendet werden. Diese Interaktionsmöglichkeiten sind über den visuellen Filter repräsentiert. Aus der Kombination von Analyse und Visualisierung entsteht für den Analysierenden ein Wissenszuwachs. Dieser kann entweder zu neuen Erkenntnissen bzw. Hypothesen führen oder den Analysierenden veranlassen, die Parameter der Vorfilterung zu ändern und somit andere Datensätze zu generieren. Es ist auch möglich, dass die Datensätze von der Visualisierung manipuliert und wieder an die Analyse zurückgegeben werden.

5.3. Technische Voraussetzungen

Um die neuen PSPs effizient zu verwalten, schnell rendern zu können und eine möglichst flüssige Interaktion zu erlauben, müssen zunächst technische Voraussetzungen gegeben sein. Ein wichtiger Aspekt ist die Datenverwaltung und die Zusammenführung von AOIs und Fixationen eines Eye-Trackers.

Im folgenden Abschnitt werden zunächst die Anforderungen an die Datensätze aufgeführt. Anschließend wird auf die Datenverwaltung mittels einer Datenbank eingegangen. Zuletzt werden die verschiedenen Möglichkeiten des Matchings von AOIs mit Fixationen vorgestellt.

5.3.1. Anforderungen an die Datensätze

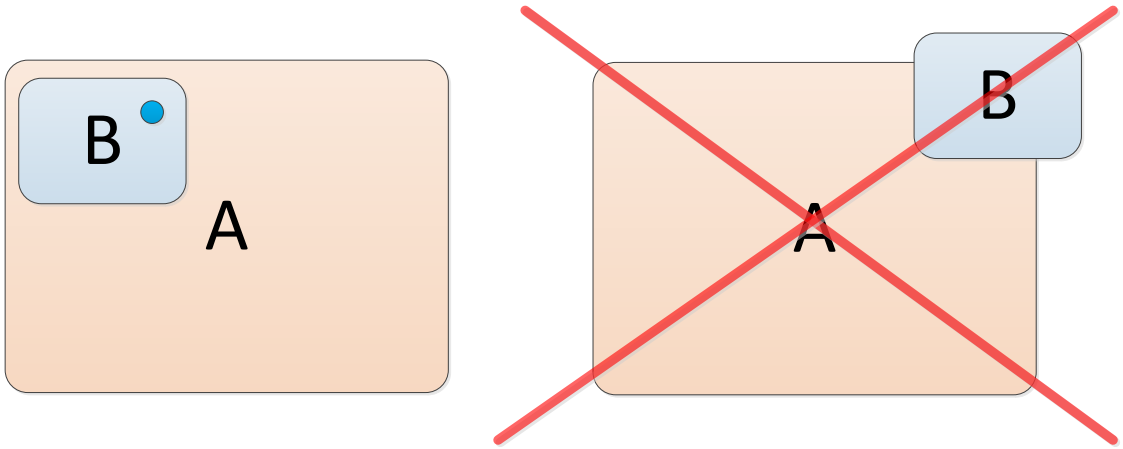

Abbildung 5.2.: Darstellung der Trefferabfrage für AOIs. Auf der linken Seite sind zwei ineinanderliegende AOIs dargestellt. Diese Anordnung der AOIs ist zulässig und die als dunkelblauer Kreis eingezeichnete Trefferabfrage könnte als Trefferergebnis sowohl „A“ als auch „B“ zurückgeben. Da „B“ jedoch die innere AOI ist, wird als Ergebnis „B“ gewählt. Die rechte Seite stellt eine unzulässige Anordnung der AOIs dar, da sich die gegebenen AOIs nicht vollständig überlappen.

Das im Folgenden vorgestellte Konzept nimmt an, dass alle AOIs innerhalb eines Stimulus eindeutig bestimmbar sind. Das bedeutet, dass AOIs zwar ineinander liegen können, ein teilweises Überschneiden jedoch ausgeschlossen ist. Sollte eine AOI innerhalb einer anderen liegen, so muss definiert sein, welche der AOIs im Falle eines AOI-hits getroffen wird. Die intuitivste Lösung des Problems ist, dass die innerste AOI, also die AOI, die von allen anderen relevanten AOIs enthalten ist, die getroffene AOI ist. Abbildung 5.2 zeigt sowohl eine zulässige Anordnung von AOIs als auch eine unzulässige Anordnung.

Außerdem muss die Abhängigkeit der AOIs voneinander, etwa ob eine AOI in einer anderen liegt, eindeutig bestimmbar sein. Dies entspricht in datenstrukturtechnischer Sicht einer Baumstruktur. Dies soll gewährleisten, dass die Relation von ineinander liegenden AOIs bekannt ist. Dies ist für die spätere Darstellung der PSPs unabdingbar.

Zudem müssen alle Fixationen einen Zeitstempel sowie pixelgenaue Koordinaten auf dem Stimulus haben. Zusätzliche Verlaufsdaten, wie etwa Daten eines Brain-Computer-Interfaces, kurz BCI, oder eines Gaspedals, sollten zwischen 0 und 1 normiert sein. Eventdaten müssen einen Zeitstempel und sollten eine Eventbeschreibung haben.

5.3.2. Datenverwaltung mit einer Datenbank

Im Grundlagenkapitel wurden bereits die Vorteile eines Datenbanksystems vorgestellt. Nachfolgend wird beschrieben, welche Daten im Falle eines Datenbankeinsatzes von dieser verwaltet und welche Zusammenhänge abgebildet werden sollen.

Benötigte Daten

Aufgrund der sehr hohen Abfragegeschwindigkeit von Daten aus einer Datenbank ist es sinnvoll, zumindest

- die Studiendaten,
- die Stimuli,
- die Probanden,
- die Fixationen,
- die Areas of Interest und
- die gematchten Fixationsdaten

in der Datenbank zu verwalten. Zum einen lassen sich mittels Datenbankabfragen die Daten hochperformant filtern und zum anderen ändern sich diese Daten nicht. Die Einbindung zusätzlicher Daten, wie etwa Eventdaten, ist rein optional.

Benötigte Verknüpfungen

Um in der Datenbank die korrekten Daten abfragen zu können, sollten sowohl die Fixationen als auch die Areas of Interest einem Stimulus zugeordnet sein. Dieser muss wiederum die Information enthalten, zu welcher Studie er gehört. Zudem müssen die Probanden einer Studie zugeordnet sein und die Fixationen müssen zu einem Probanden gehören.

5.3.3. Datenerzeugung: Matching von AOIs mit Fixationen einer Studie

Um eine AOI-basierte Analyse von Fixationsdaten durchführen zu können, ist es sinnvoll, die Fixationen bereits als Vorverarbeitungsschritt mit AOIs zu verbinden. Hierdurch ergibt sich zum einen ein Performancegewinn bei der Verarbeitung der Daten in der eigentlichen Datenanalyse und zum anderen wird es dadurch möglich, die Zwischenergebnisse der Analyse zu speichern. Dazu wird für die Fixationen eine Trefferabfrage für die AOIs durchgeführt und die AOI, in der die Fixation liegt, wird mit dieser verbunden. Um diese Trefferabfrage durchzuführen, benötigt man jedoch zunächst Areas of Interest. Es gibt drei Möglichkeiten, diese zu generieren:

- Manuell,
- mit Clusteringalgorithmen, die nur auf den Fixationsdaten arbeiten, und
- mit Segmentierungstechniken der Computer Vision, die den Stimulus mittels Bilderkennung in Bereiche unterteilen, die dann als AOIs definiert werden.

Diese werden nachfolgend genauer beschrieben.

Manuelles Erstellen von AOIs

Eine der Möglichkeiten AOIs zu definieren, ist deren manuelle Erstellung durch den Benutzer. Dieses Verfahren hat den großen Vorteil, dass der Studienleiter die seiner Meinung nach interessanten Gebiete a priori kennt und diese entsprechend definieren kann. Zudem kann ein Mensch problemlos visuelle Zusammenhänge zwischen Objekten auf einem Stimulus erkennen und dieses Wissen bei der Definition der Areas of Interest berücksichtigen. Dieser Vorteil bringt jedoch eine Reihe von Nachteilen mit sich. So können unerwartete Interessensgebiete eventuell übersehen werden. Außerdem ist die Erstellung der AOIs von Hand sehr zeitaufwendig.

Erstellen der AOIs mittels eines Clustering-Algorithmus

Eine weitere Möglichkeit, Areas of Interest zu erzeugen ist die Nutzung eines Clustering-Algorithmus. Hierbei wird der zugrundeliegende Datensatz untersucht und auf Basis einer vorgegebenen Metrik werden AOIs in den Gebieten erzeugt, in denen Cluster entstehen. Bei Eye-Tracking-Studien sind eine naheliegende Metrik die Koordinaten der Fixationen. Eine Möglichkeit des Clusterings ist der in Kapitel 2.5 der Grundlagen beschriebene k-means-Algorithmus. Hierdurch werden AOIs an den Orten festgelegt, die Probanden häufig fokussiert haben.

AOIs durch semantische Annotation

In der Arbeit „Entwicklung eines Konzepts zur Annotation von grafischen Elementen in Visualisierungen“ von Stefan Strohmaier [32] wird vorgeschlagen, dass die semantische Annotierung zum Zeitpunkt der Erstellung des Stimulus erfolgen soll. Der Vorteil dieses Ansatzes

ist, dass die Annotationen für unterschiedliche Datensätze mit gleichem Stimulus wiederverwendbar sind. Zudem eignen sie sich gut zur Modellierung von kognitiven Prozessen. Dies ist von Vorteil, wenn man einen Vergleich von Lösungsstrategien beim Betrachten des Stimulus wünscht.

Erstellung mittels Computer Vision

Es ist außerdem möglich, AOIs mittels Bildanalysetechniken zu definieren. Ein Ansatz ist dabei das Annotieren mittels Segmentierung. Hierbei wird der Inhalt des Stimulus in einzelne Regionen unterteilt. Sofern eine Wissensdatenbank verfügbar ist, können die entstandenen Regionen semantisch annotiert werden. Der Vorteil dieser Technik ist eine automatische Annotierung der Stimuli, wodurch die AOIs im Vergleich zu Clustering-Algorithmen semantisch mehr Sinn ergeben, denn selbst ohne Wissensdatenbank entstehen die Regionen entlang von Kanten im Stimulus, da Kanten häufig eine Abgrenzung von Objekten darstellen. Der Nachteil dieses Ansatzes ist die Bildanalyse an sich, da diese eine Verbindung zwischen der segmentierten Region und der Wissensbasis herstellen muss. Zum einen muss die Wissensbasis hinter dem Analysealgorithmus alle in dem Stimulus vorkommenden Objekte beinhalten, um diese dann auf dem Stimulus zu erkennen. Zum anderen muss der Algorithmus sowohl robust als auch zuverlässig sein.

5.4. Vorfilterung der Daten

Im Vorfilterungsschritt können die Daten nach bestimmten Metriken gefiltert werden. Hierbei verarbeitet man nur selten die Aufnahmedaten, die für die Analyse der Fixationen oder der Probanden benötigt werden, sondern die Metadaten. So soll es möglich sein, bestimmte Datensätze von vornherein von der Analyse auszuschließen. So ist es beispielsweise denkbar, dass bei der Analyse nur Probanden betrachtet werden sollen, bei denen der Anteil der gültigen Eye-Tracking-Daten einen bestimmten Prozentsatz übersteigt.

Da die Vorfilterung aufgrund neuer Erkenntnisse während der Analyse eventuell angepasst wird, sollten die Daten auch nach anderen Metriken wie Alterskohorten oder Geschlecht der Probanden gefiltert werden können. Diese Daten kommen nicht vom Eye-Tracker, sondern müssen von einer externen Quelle in die Datenbank eingespielt werden.

5.5. Analyse (Distanz/Überdeckungen)

Die Analyse ermöglicht es dem Anwender, die Daten mittels unterschiedlicher Filter verarbeiten zu lassen. So ist es möglich, die Datensätze der einzelnen Probanden einer Studie miteinander zu vergleichen, um über Blickmuster oder andere Metriken Gruppen zu erzeugen, die dann einzeln weiterverarbeitet, ausgeblendet oder miteinander verglichen werden können. Einer der Kernaspekte hierbei ist das Vergleichen der Blickpfade. Ein ähnliches Blickmuster lässt die Vermutung zu, dass die Probanden die gleiche Strategie zur Lösung des vorgegebenen Problems angewendet haben.

Zusätzlich sollen die verwendeten Daten die Auswahl der Visualisierungskomponente berücksichtigen und somit eine Unterscheidung zwischen allen, global verfügbaren Daten und den durch die Visualisierung vorgegebenen, eingeschränkten Daten ermöglichen.

5.5.1. Datengrundlage: Globale oder eingeschränkte Daten

Die ETA-Pipeline sieht eine Interaktionsmöglichkeit zwischen der Visualisierungskomponente, die ab Abschnitt 5.6 vorgestellt wird, und der Analysekomponente vor. Um dies zu berücksichtigen, sollen die einzelnen Filter der Analyse vorsehen, dass es sich um die globalen oder die bereits vom Benutzer selektierten Daten handelt.

5.5.2. Metrik für die Ähnlichkeit bei AOI-basierten Fixationen

Bevor ein Vergleich der Datensätze möglich wird, muss zunächst definiert werden, was das Maß ist, an dem die Ähnlichkeit der Blickpfade gemessen wird. In Kapitel 5.3.1 auf Seite 42 wurden bereits die Anforderungen an die Datenstruktur vorgegeben. Da jede AOI eindeutig ist, kann man jede AOI $\alpha \in \nu$ als Eingabezeichen für ein Alphabet $\Sigma := \nu \cup \diamond$ mit $\alpha \notin \diamond$ betrachten. Da alle Transitionspfadelemente eindeutig zu einer AOI zugeordnet werden können müssen, ergibt sich hierdurch für jeden Datensatz ein String. Dieser String soll die Grundlage für die einzelnen Vergleichsalgorithmen darstellen.

5.5.3. Pfadvereinfachung

Bei der Analyse von Eye-Tracking-Daten erkennt man häufig das Hin- und Herspringen des Fokus zwischen zwei Punkten auf einem Stimulus. Beispiele hierfür wären Szenarien, in welchen der Proband einen Wert ablesen möchte und deshalb häufig zwischen dem abzulesenden Punkt und einer Werteskala hin- und herspringt oder ein Proband aufgefordert wird, in einem Supermarkt einzukaufen, und daher häufig zwischen verschiedenen Marken und deren Preisschild springt. Dieses Verhalten wird bei einem AOI-String durch die Wiederholung eines bestimmten Substrings repräsentiert. Da dieses Verhalten jedoch nur selten auf eine bestimmte Suchstrategie hinweist, kann es sinnvoll sein, die wiederkehrenden Substrings zusammenzufassen.

Der Vorteil einer Pfadvereinfachung als Vorverarbeitungsschritt für einen Stringvergleich ist, dass wiederkehrende Muster, in der Annahme einer unbedeutenden Wiederholung, zusammengefasst werden. Somit ist es wahrscheinlicher, ähnliche Strategien mit einem Stringvergleich zu finden, da Wiederholungen bei einem Stringvergleich als unterschiedliche Wörter aufgefasst werden.

Die Annahme der unbedeutenden Wiederholungen kann dahingehend eingeschränkt werden, dass eine gewisse Anzahl an Wiederholungen, etwa vier oder mehr, auf ein cross-checking der Werte hinweist. Das kann bedeuten, dass bestimmte Informationen entweder sehr komplex und somit schwer zu merken sind oder dass diese Informationen als sehr wichtig angesehen werden und erneut überprüft werden. Für Vergleichbarkeit der Lösungsstrategie ist das cross-checking jedoch von eher geringer Bedeutung, sodass deren Berücksichtigung rein optional ist.

Diese Art der Stringvereinfachung birgt jedoch auch Nachteile. So werden Datensätze mit sehr viel Rauschen mit hoher Wahrscheinlichkeit nicht beeinflusst. Somit können verrauschte Daten plötzlich ähnlich zu zuverlässigen Daten wirken.

5.5.4. Ähnlichkeit mittels Levenshtein-Distanz

Mit der Levenshtein-Distanz ist es möglich, zwischen zwei Strings ein Ähnlichkeitsmaß zu berechnen. Im Falle der PSP-Daten entsprechen diese Wörter dem Transitionspfad. In Verbindung mit der Eye-Mind-Hypothese kann man hieraus ableiten, dass ähnliche Wörter ähnlichen Gedankengängen und daher ähnlichen Suchstrategien entsprechen.

Im folgenden wird auf die Laufzeit der Levenshtein-Distanz für Transitionspfade eingegangen und es werden Modelle für die Kostenfunktion der Levenshtein-Distanz vorgestellt.

Laufzeit

Die Laufzeit der Berechnung der Levenshtein-Distanz von zwei Strings mit den Längen m und n entspricht $O(m \cdot n)$. Da alle Transitionspfade miteinander verglichen werden müssen entspricht die Laufzeit der Berechnung

$$\underbrace{O(m \cdot n)}_{\text{Laufzeit Levenshtein}} \cdot \underbrace{O(p^2)}_{\text{Vergleich aller Pfade}} = O(m \cdot n \cdot p^2), \text{ wobei} \tag{5.1}$$

m = Anzahl der Zeichen im ersten String

n = Anzahl der Zeichen im zweiten String

p = Anzahl der Pfade

Die Laufzeit steigt demnach am stärksten, wenn sich die Anzahl der Transitionspfade, die der Anzahl der Probanden entspricht, erhöht. Da diese Zahl jedoch häufig nicht sehr groß, etwa 30 bis 100 Probanden, ist, besteht das größte Problem in der Länge der Aufnahme, mit der zwangsläufig die Länge jedes Pfades steigt, da die Probanden mit hoher Wahrscheinlichkeit häufiger zwischen AOIs wechseln.

Modelle für die Kostenfunktionen

Es gibt mehrere Möglichkeiten, worauf man bei einem Vergleich von Wörtern den Fokus legt. So kann man beispielsweise eine höhere Priorität auf einen ähnlichen Wortbeginn legen oder jeden Buchstaben gleich stark gewichten. Daraus leitet sich der Bedarf an unterschiedlichen mathematischen Modellen ab, die diese Vergabe von Prioritäten ermöglichen. Diese Modelle können die Kostenfunktion konstant, logarithmisch, exponentiell oder nicht-stetig definieren. Auch eine Kombination der Modelle ist denkbar. Die genannten Funktionsklassen sind in Abbildung 5.3 abgebildet.

Konstante Kostenfunktion: Es besteht die Möglichkeit, den einfachsten Stringvergleich anzuwenden und den einzelnen Operationen konstante Funktionen, also einen konstanten Wert, zuzuweisen. Dies bietet sich vor allem an, wenn keine Annahmen bezüglich des Blickverhaltens, etwa aufgrund des Studiendesigns, vorherrschen.

Logarithmische Funktion: Falls man die Kosten für die ersten Zeichen des Strings besonders hoch oder niedrig ansetzen möchte, bietet sich eine logarithmische Funktion an. Es erscheint vor allem sinnvoll, die Kosten zu Beginn des Strings höher anzusetzen als im weiteren Verlauf, da die Strings, und damit die Suchstrategien, vor allem am Anfang übereinstimmen müssen und im weiteren Verlauf mit hoher Wahrscheinlichkeit das Rauschen der Eye-Tracking-Daten, und damit der vermeintlichen Suchstrategie, zunimmt und es somit schwieriger wird, in der Mitte der Aufnahme noch eine Strategie festzustellen.

Exponentielle Funktion: Es ist ebenfalls möglich, die Funktionen exponentiell anzusetzen. Hierdurch können sich vor allem am Anfang die Pfade am Anfang unterscheiden, wodurch eine eventuell existierende Übersichtsphase sich nur unbedeutend auf die Kosten des Blickpfads auswirkt.

Kombination verschiedener Funktionen: Um die Vorteile der bereits genannten Funktionen zu nutzen, kann man diese, beispielsweise mittels einer Multiplikation, kombinieren.

Nicht-stetige Funktion: Alternativ zur herkömmlichen Kombination der Funktionen ist es denkbar, zu unterschiedlichen Zeitpunkten unterschiedliche Funktionen anzuwenden. Dies entspricht mathematisch einer nicht-stetigen Funktion. Einer der großen Vorteile dieses Ansatzes ist, dass man die anfänglichen Blicke mit niedrigen oder gar keinen Kosten belegt und später eines der anderen Modelle anwendet.

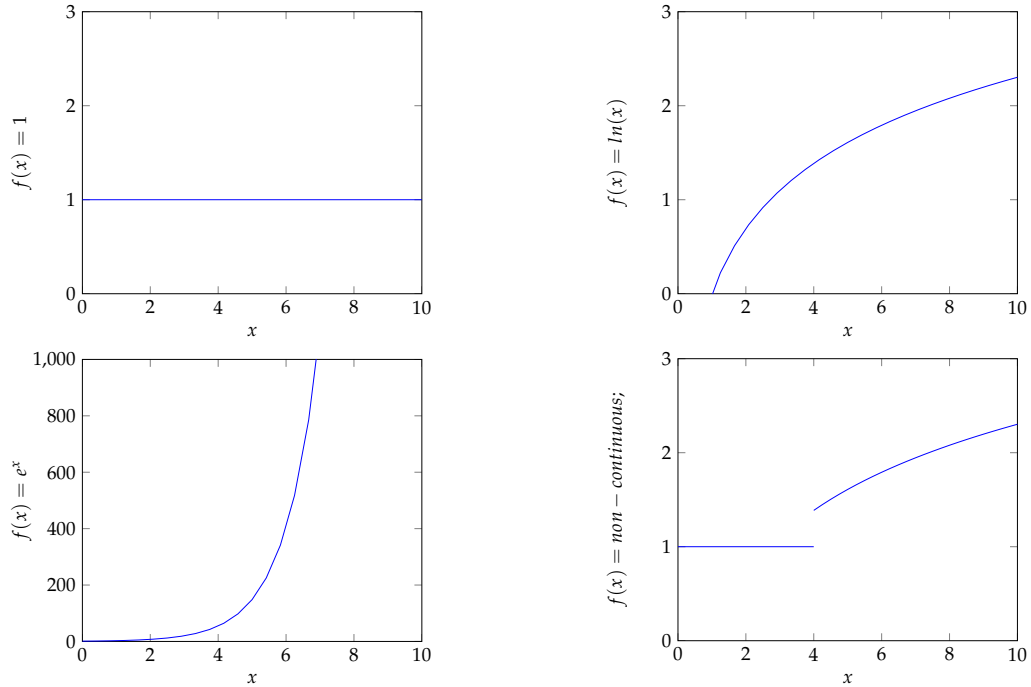

Abbildung 5.3.: Darstellung der verschiedenen Modelle für Kostenfunktionen der Levenshtein-Distanz. Oben links: Konstant; Oben rechts: Logarithmisch; Unten links: Exponentiell; Unten rechts: Nicht-stetig mit einer Funktion für $x <= 1$ und einer logarithmischen Funktion für $x < 1$.

Es ist denkbar, dass die Kostenfunktionen der einzelnen Stringoperationen unterschiedlichen Modellen folgen. So könnte man beispielsweise erzwingen, dass am Anfang nur Zeichen eingefügt, aber weder gelöscht noch getauscht werden.

5.5.5. Wortähnlichkeit durch semantische Abhängigkeit der AOIs

Anstatt die Kostenfunktionen für die Levenshtein-Distanz selbst festzulegen, ist es denkbar, die Funktionen automatisch anpassen zu lassen. Hierzu wird jedoch Vorwissen auf Seiten des Computers benötigt. Eine Möglichkeit, dieses Wissen zu generieren ist, die AOIs des Stimulus semantisch zu Annotieren und eine Ontologie zu hinterlegen. Dieses Wissen kann sowohl für eine Anpassung der Kosten bei den Übergängen zwischen bestimmten AOIs genutzt werden als auch bei der Anordnung innerhalb der Visualisierung. Hierzu müssen die Beziehungen der Areas of Interest in einer Baumstruktur hinterlegt werden. Bei den Kostenfunktionen ist es beispielsweise denkbar, dass Übergänge zwischen miteinander verbundenen Elementen innerhalb der Ontologie als „besser“ eingestuft werden und damit bevorzugt behandelt werden.

5.5.6. Ähnlichkeit mittels Needleman-Wunsch

Der in Kapitel 2.4.2 vorgeschlagene Algorithmus von Needleman und Wunsch kann dahingehend genutzt werden, dass die Kosten von Übergängen zwischen den einzelnen AOIs unterschiedlich gewichtet werden. Dies ist insbesonders dann nützlich, wenn die zuvor beschriebenen semantischen Abhängigkeiten der AOIs bekannt sind.

5.5.7. Ähnlichkeit mittels Mustervergleich

Pattern Matchings ist hilfreich, um bestimmte Transitionssequenzen zu erkennen. Somit ist es möglich, wiederkehrende Blickmuster weiter zu untersuchen. Obgleich diese Technik einige interessante Aspekte hat, bestehen doch einige Probleme:

Beliebige Reihenfolge: Da die Reihenfolge der Substrings beliebig sein kann ist es nicht mehr möglich, eine Aussage über Abhängigkeiten der Elemente in dem Stimulus zu treffen. Man kann nur die Aussage treffen, dass bestimmte „Unterstrategien“ von mehreren Probanden angewendet wurden.

Exakte Übereinstimmung: Das Pattern Matching lässt nur exakte Übereinstimmungen zu. Die daraus folgende Fehlerintoleranz bedeutet, dass in den Transitionspfaden keine Fehlmessungen stattgefunden haben dürfen. Dies ist jedoch aus technischen Gründen bei Eye-Tracking-Studien nur sehr schwer zu bewerkstelligen.

5.5.8. Diskussion der Techniken

Die beiden vorgestellten Techniken zur Bestimmung der Stringähnlichkeit von Levenshtein und Needleman und Wunsch sowie dem Musterabgleich haben jeweils Vor- und Nachteile, die in Tabelle 5.1 zusammengefasst werden. Hierbei wird deutlich, dass das Rauschen der Daten auf Eye-Tracking-Datensätzen ein großes Problem darstellt. Zudem fällt auf, dass viele

der Nachteile der Algorithmen zur Messung der Stringähnlichkeit beim Musterabgleich nicht auf. Dies legt die Überlegung nahe, ob sich die Techniken verbinden oder kombinieren lassen.

	Levenshtein	Needleman-Wunsch	Pattern Matching
Vorteile	Fehlertolerant; Kostenfunktionen ermöglichen verschiedene Schwerpunkte zu unterschiedlichen Zeitpunkten; Ausgabe ist bereits Abstandsmaß;	Fehlertolerant; Kostenfunktionen ermöglichen die Bevorzugung bestimmter Transitionen; Ausgabe ist bereits Abstandsmaß;	Findet Unterstrategien; Keine Kostenfunktion nötig; Substringlänge (und damit Unterstrategielänge) kann festgelegt werden;
Nachteile	Trotz Fehlertoleranz anfällig gegen stückweises Rauschen;	Trotz Fehlertoleranz anfällig gegen stückweises Rauschen;	Keine Fehlertoleranz; Abstandsmaß nicht gegeben; Keine Abhängigkeit der Strategien, da das Matching an beliebigen Stellen stattfinden kann

Tabelle 5.1.: Vergleich der Vor- und Nachteile der Algorithmen von Levenshtein, Needleman-Wunsch und dem Pattern Matching.

5.5.9. Kombination der Techniken

Unter Umständen kann es von Interesse sein, das Verhalten von Probanden zu unterschiedlichen Zeitpunkten miteinander zu vergleichen. Sofern man voraussetzen kann, dass in diesen Zeiträumen beispielsweise das gleiche Objekt angesehen wird, so kann man versuchen, diesen Vorgang zu untersuchen. Hierfür würde sich das Pattern Matching anbieten, doch die Voraussetzung, dass die Daten exakt übereinstimmen müssen, macht ein Matching bei Eye-Tracking-Daten nahezu unmöglich.

Es ist daher sinnvoll zu überlegen, ob sich das Pattern Matching mittels der Fehlertoleranz der Levenshtein-Distanz für Eye-Tracking-Studien anpassen lässt. Hierzu muss man den Vergleich der Substrings dahingehend anpassen, dass die Levenshtein-Distanz der verglichenen Substrings berechnet wird, und sofern sie unter einer zuvor festgelegten Schranke liegt, wird eine Übereinstimmung festgestellt. Wenn man die Begrenzung auf 0 festsetzt, entspricht das dem ursprünglichen Pattern Matching.

5.5.10. Gruppierung

Ein weiterer Aspekt der Analyse ist die Gruppierung der Daten. Dabei sollen Daten, die sich bezüglich einer bestimmten Metrik ähnlich sind, einander zugeordnet werden. So sind Datensätze beispielsweise nach den Alterskohorten der Probanden gruppierbar. Es ist auch möglich,

die Gruppierung anhand der genutzten Löstungsstrategie einer gegebenen Aufgabe aufzubauen. Hierzu gibt es zwei Möglichkeiten, nach welcher Metrik die Daten gruppiert werden sollen. Die Gruppierung findet mittels der in Kapitel 2.5 beschriebenen Clusteringtechniken statt.

Gruppierung nach Metadaten

Die erste Möglichkeit ist, die Daten nach ihren Metadaten zu sortieren. Es ist beispielsweise denkbar nach Kohorten oder Geschlecht zu gruppieren. Es werden bei Studien auch häufig weitere Daten gesammelt. Hierzu gehören oft

- die Erfahrung mit Computern,
- die Kenntnisse eines bestimmten Fachbereichs, wie Mathematik, Psychologie oder einer bestimmten Sprache,
- der Bildungsgrad, wie etwa Schul- und Hochschulabschluss,
- Beruf,
- Kenntnisse über mit der Studie verwandten Techniken.

Gruppierung nach Lösungsstrategie

Eine der Kernideen dieser Arbeit ist, dass die Eye-Tracking-Daten, wenn sie mit annotierten Daten verbunden werden, auch auf einer lösungsstrategischen Ebene miteinander verglichen werden können. Die für die Clusterbildung benötigte Distanzfunktion hängt von der verwendeten Vergleichstechnik ab (siehe hierzu 5.5.4, 5.5.6, 5.5.7 und 5.5.9). Eine geringere Distanz bedeutet auch einer höhere Ähnlichkeit der Probanden bezüglich ihrer Lösungsstrategie. Hieraus ergibt sich bereits eine Hierarchie. Da für die Probanden also bereits eine Hierarchie vorhanden ist und nicht bekannt ist, wie viele Gruppen bei der Anwendung eines Clusteringalgorithmus entstehen, ist die Nutzung eines hierarchischen Clustering-Algorithmus sinnvoll. Um die Distanzen zu vereinheitlichen, ist es naheliegend, die Daten vor der Anwendung des Clusteringalgorithmus zu normieren.

5.5.11. Zusammenfassung der Daten

Es ist denkbar, dass man bei der Untersuchung der Datensätze gemäß Shneiderman [29] zunächst eine Übersicht gewinnen will. Diese Übersicht kann entweder auf den vollständigen Datensätzen basieren oder bereits eine Gruppierung enthalten. Bei der Visualisierung aller Datensätze entsteht sehr schnell das Problem von vielen Überdeckungen der Transitionspfade in der Visualisierung, selbst wenn die Datensätze gruppiert sind. Damit entsteht auch Visual Clutter, welcher die Analyse der Visualisierung für den Menschen erschwert. Sofern bei der Auswertung der Studie daher schon eine Metrik für die Gruppierung festgelegt werden kann, bestehen drei Möglichkeiten, den Visual Clutter zu reduzieren:

1. Es wird eine bestimmte Gruppe an Datensätzen ausgewählt und nur deren Daten werden angezeigt.

2. Man generiert für jede Gruppe einen Durchschnittsverlauf und vergleicht dann die repräsentativen Verläufe der einzelnen Gruppen.

3. Man wählt einen Datensatz der Gruppe aus, der die Gruppe repräsentiert, und vergleicht die Gruppen anhand der repräsentativen Datensätze.

Vor allem die letzten beiden Vorschläge sind vielversprechend. Eine Durchschnittsbildung wirkt glättend und hierdurch werden auch grundsätzlichere Unterschiede zwischen den Gruppen ersichtlich, die zuvor eventuell im Rauschen untergegangen wären.

Erzeugung eines „Durchschnittsdatensatzes“ von Eye-Tracking-Aufnahmen

Wenn man den ersten Ansatz des vorherigen Abschnitts verfolgen und einen „Durchschnittsdatensatz“ erzeugen will, stellt sich die Frage, wie man die zugrunde liegenden Datensätze vereinheitlichen soll. Grundsätzlich ist die Durchschnittsbildung nur sinnvoll, wenn sich die Daten der Transitionspfade ähneln. Dies kann jedoch nur gewährleistet werden, wenn als Metrik die Lösungsstrategie, also der Vergleich der Transitionspfade, gewählt wurde. Da hierbei der generelle Verlauf des Transitionspfades optimalerweise identisch ist, sich aber zumindest ähneln sollte, kann hierbei zunächst eine durchschnittliche Abfolge der besuchten AOIs erzeugt werden. Danach muss man die Verweildauer innerhalb der AOI und die Fixationsanzahl festlegen. Diese Werte können einfach durch das arithmetische Mittel oder den Median der Datensätze, auf die der Besuch der AOI zutrifft, festgelegt werden. Der daraus entstehende Datensatz lässt sich als Repräsentant der Gruppe für die weitere Datenanalyse nutzen.

Nutzbarkeit von repräsentativen Datensätzen

Die Erzeugung eines Durchschnittsdatensatzes bzw. die Wahl eines Repräsentanten für eine Gruppe an Datensätzen gibt eine gute Übersicht über die zu erwartenden Datensätze. Da der angezeigte Datensatz jedoch nur einen exemplarischen oder gar einen approximierten Transitionspfad darstellt, ist es für eine eingehendere Analyse unbedingt notwendig, alle Datensätze einer Gruppe zu betrachten. Hiermit wird es möglich, die Güte des repräsentativen Transitionspfades mit der Gruppe abzugleichen und einzuschätzen. Diese Abschätzung kann mit statistischen Kennwerten wie einer durchschnittlichen Abweichung von dem repräsentativen Datensatz gestützt werden.

Unabhängig von den statistischen Werten scheint es sinnvoll, sich auch die Datensätze, aus denen die Übersicht generiert wurde, anzusehen, sofern man sich entscheidet, die Daten des Übersichtsdatensatzes zu nutzen.

5.5.12. Filtern von Eigenschaften

Um zuverlässigere Datensätze zu erhalten und Ausreißer auszuschließen, kann man Datensätze oder ganze Gruppen ausschließen, die bestimmte Eigenschaften oder statistische Nebenbedingungen nicht erfüllen. So können beispielsweise Datensätze, deren Aufnahmedauer mehr als 30% von der durchschnittlichen Aufnahmedauer abweicht, ausgeschlossen werden.

5.5.13. Visuelle Darstellung der ETA-Pipeline

Um dem Studienanalysierenden die Möglichkeit zu geben, die Filter anzuordnen und zu konfigurieren ist es möglich, diese Filter in Form einer Pipeline zu repräsentieren. Hierdurch erhält der Analysierende die Möglichkeit, schnell und intuitiv die unterschiedlichen Filter zu konfigurieren und somit die Daten zu manipulieren. Anschließend kann die ETA-Pipeline als weitere Komponente die Visualisierung anzeigen. Ein Entwurf der Darstellung der ETA-Pipeline ist in Abbildung 5.4 gegeben.

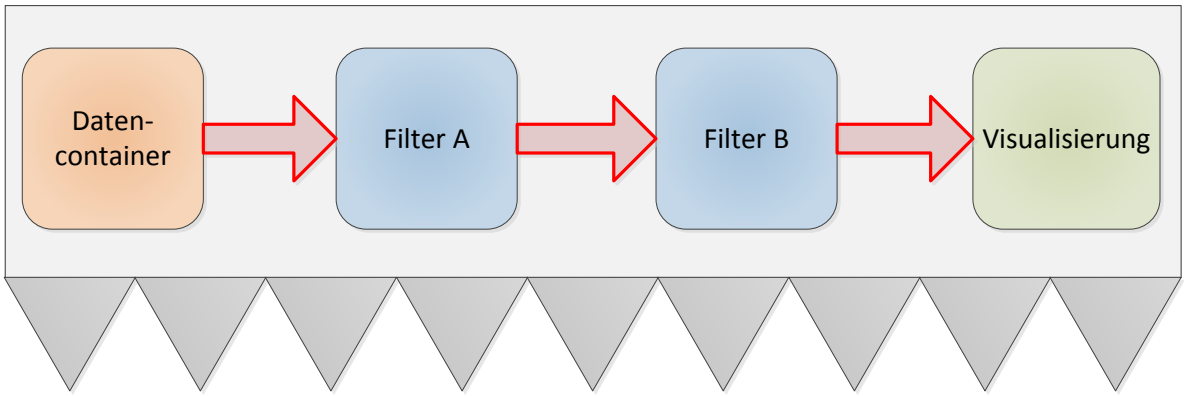

Abbildung 5.4.: Schematische Ansicht der ETA-Pipeline. Das linke Element repräsentiert die Datenquelle, welche die zu analysierenden Datensätze zur Verfügung stellt. Danach lässt sich eine beliebige Anzahl an Filtern, hier zwei, einfügen. Auf der rechten Seite findet sich ein Feld für die Visualisierung der gefilterten Datensätze. Der durch Zacken angedeutete Bereich stellt die Oberfläche des ausgewählten Filters dar.

5.5.14. Semantische Annotation der Filterübergänge

Ein großes Problem der Analysekomponente ist, dass es zunächst zu keinem Zeitpunkt möglich ist, Einsicht in die Daten zu erhalten. Dies ist vor allem problematisch, da man nicht feststellen kann, welche Auswirkungen die einzelnen Analysekomponenten auf den vorliegenden Datensatz haben.

Um diese Problematik anzugehen, werden zwischen den einzelnen Komponenten Übergänge visualisiert, die mit Metainformationen der Datensätze annotiert sind. So ist es möglich, zu sehen, wie viele Gruppen momentan verwaltet werden und wie viele Datensätze noch vorliegen. Es ist auch denkbar, dass weitere statistische Informationen über die einzelnen Gruppen, wie die durchschnittliche Aufnahmezeit, oder die durchschnittliche Aufnahmegüte, angezeigt werden.

5.6. Interaktive Visualisierung

Nachdem die Datensätze mittels der Analyse vorbereitet wurden, werden die Daten visualisiert. Die Visualisierung baut auf dem Konzept der in Abschnitt 4.1 vorgestellten Parallel

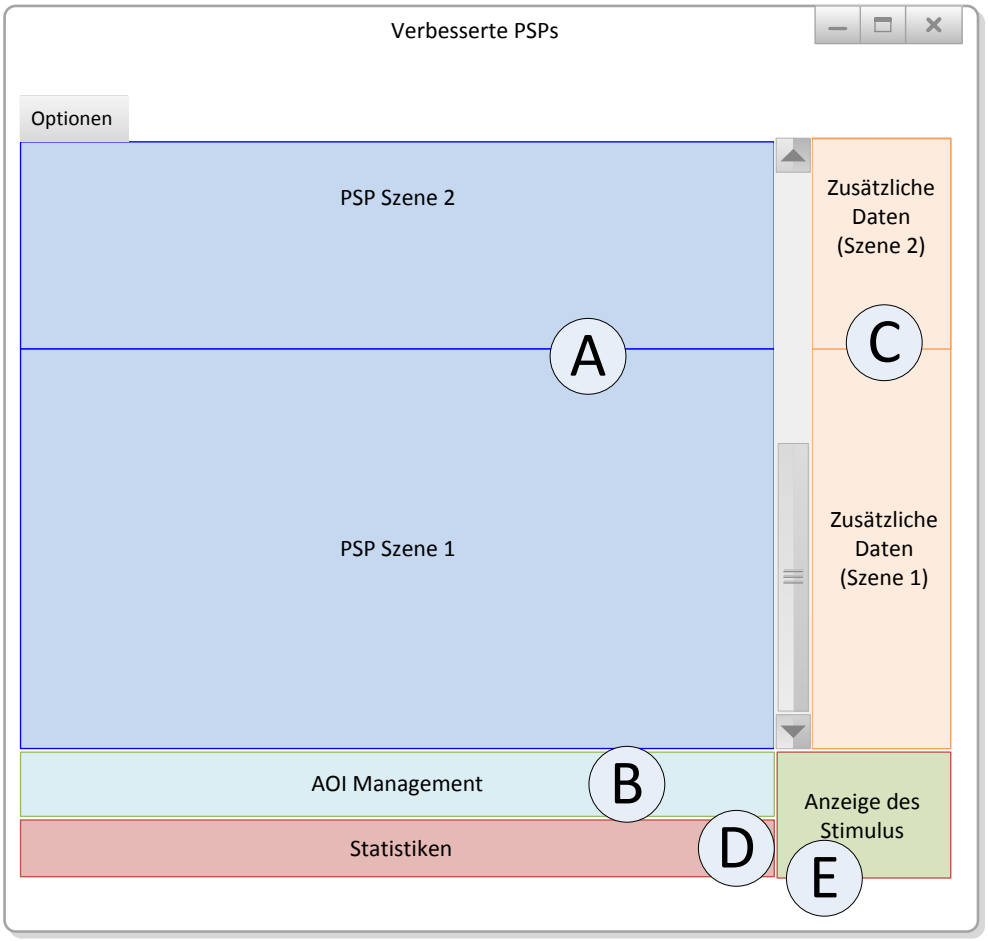

Abbildung 5.5.: Schematische Ansicht des Visualisierungskonzepts. Der in blau hervorgehobene Bereich „A“ enthält die einzelnen Szenen, die jeweils Teile des PSPs darstellen. In dem türkisfarbenen Bereich „B“ werden die einzelnen AOIs angezeigt. Der orangefarbene Bereich „C“ stellt zusätzliche, von außen zur Verfügung gestellte, Daten dar. Der rote Bereich „D“ zeigt zusätzliche Statistiken an. In Bereich „E“ kann optional der betrachtete Stimulus angezeigt werden.

Scan-Paths auf. Diese werden um Interaktions- und Einstellungsmöglichkeiten erweitert. Der folgende Unterabschnitt erläutert, wie die Daten dargestellt werden. Danach werden die, vor allem bei Videodaten anwendbare, Szenen- bzw. Abschnittsfilterung und deren Interaktionsmöglichkeiten vorgestellt. Anschließend werden das AOI-Management und dessen Interaktionsmöglichkeiten aufgezeigt. In diesem Zusammenhang wird auch auf die Anordnung und die Problematik der Skalierbarkeit bei AOIs eingegangen. Der anschließende Abschnitt wird sich mit der Möglichkeit, externe Daten einzubinden, beschäftigen. Hierbei werden sowohl die verschiedenen Arten der Daten als auch deren Interaktionsmöglichkeiten erläutert. Zuletzt wird auf die optionale Anzeige von Statistiken eingegangen.

5.6.1. Gesamtvorschau

Der auf die Analyse folgende Teil soll mit Hilfe der Visualisierungstechnik der Parallel Scan-Paths dem Studienanalysierenden eine visuelle Aufbereitung der Eye-Tracking-Daten bieten. Eine schematische Ansicht der Visualisierung ist in Abbildung 5.5 gegeben. Sie zeigt die verschiedenen Bereiche der Visualisierung, auf die in späteren Abschnitten im Detail eingegangen wird. Der Bereich „A“ zeigt die in einzelne Abschnitte unterteilbaren PSP-Daten. Diese können zusammen mit den optional anfügbaren Daten (in der Abbildung mit „B“ gekennzeichnet) ein- oder ausgeblendet werden. Hierbei handelt es sich um Daten, die aus anderen Quellen als einem Eye-Tracking-System stammen. Hierbei könnte es sich beispielsweise um Eingabeereignisse handeln. Unter den PSPs befindet sich das AOI-Management, das in der Abbildung mit „C“ markiert ist. Hier werden die einzelnen AOIs mit ihren untergeordneten AOIs angezeigt und können dort auch manipuliert werden. Die sich darunter befindlichen Statistiken zeigen zusätzliche Informationen, wie etwa die durchschnittliche Aufenthaltszeit in einer AOI, an. Diese sind in der Abbildung als Bereich „D“ hervorgehoben.

5.6.2. Szenen- und Abschnittsfilterung

In Visualisierungsstudien sind die Aufnahmen für einen Stimulus nur selten länger als eine Minute, da oft die Auswertungsgeschwindigkeit und die Qualität der Visualisierung bewertet werden sollen. Im Falle von Eye-Tracking-Aufzeichnungen aus der Industrie oder dem Marketing werden jedoch häufig entweder Head-Mounted Eye-Tracker oder statische Eye-Tracker mit Videodaten oder einer sich verändernden Umgebung als Stimulus verwendet. Hierbei sind die Aufnahmezeiten potenziell deutlich länger und können mit einer Videoaufnahme verglichen werden. Analog zu Videoaufnahmen kann man dadurch auch die Eye-Tracking-Aufnahmen in Szenen unterteilen und diese getrennt voneinander anzeigen und analysieren. Eine Möglichkeit, die Szenen anzuzeigen ist, diese aneinandergereiht darzustellen. Diese Anordnungsmöglichkeit wird hier als Szenenguide bezeichnet. Der Szenenguide wurde in Form von übereinander angeordneten PSPs in die GUI integriert, wie in Bereich „A“ von Abbildung 5.5 dargestellt. Die Szenen müssen dabei nicht gleich viel Platz in Anspruch nehmen. Es ist beispielsweise denkbar, dass der Platzanteil einer Szene

- abhängig von der Anzahl der in ihr enthaltenen Fixationen im Vergleich zur Gesamtfixationszahl der Aufnahme oder
- von der Dauer der Szene im Vergleich zur Gesamtzeit der Aufnahme ist.

Die Unterteilung in Szenen hat den großen Vorteil, dass lange Aufnahmen in einzelne Abschnitte unterteilt werden können. Diese Unterteilung birgt jedoch je nach Szenario auch Probleme. Wenn die Eye-Tracking-Daten beispielsweise auf einem interaktiven Szenario, wie beispielsweise einer Autofahrt, einem Einkauf in einem Supermarkt oder der Benutzung einer Software mit einem gegebenen Ziel, basieren, so ist es unwahrscheinlich, dass die unterschiedlichen Szenen für alle Probanden gleich lang sind.

Im Folgenden gilt die Annahme, dass alle Probanden die gleichen Szenen in derselben Reihenfolge durchlaufen haben. Das bedeutet, dass diese die Stimuli in der gleichen Reihenfolge betrachtet haben und nicht beispielsweise *Proband 1* zuerst Szene „A“ und dann Szene

„B“ gesehen hat und *Proband 2* zuerst B und dann A. Zunächst wird festgestellt, welcher Proband am längsten in der Szene verweilt hat. Dies ist wichtig, um eine obere Grenze für die Szene festzulegen. Ein generelles Problem des Szenenansatzes ist, dass die Zeitstempel der Aufnahmen nicht mehr global nutzbar sind (sondern bestenfalls innerhalb der Szenen). Danach sind für diese Problematik zwei Ansätze denkbar:

1. Die Aufnahmen aller Probanden werden normiert. Hierdurch würden die Verläufe der Teilnehmer innerhalb der Szene vergleichbar gemacht. Jedoch geht durch die Normierung jegliche Zeitinformation verloren. Zudem ist es wahrscheinlich, dass Probanden, die länger in einer Szene verbleiben, deutlich längere Transitionspfade haben, wodurch der Vorteil der Vergleichbarkeit unter Umständen wieder verloren geht.

2. Wenn ein Proband die Szene früher beendet als die anderen, so endet sein Transitionspfad an dieser Stelle. Falls auf die Szene eine weitere folgt, so fährt der Transitionspfad dort fort. Es ist denkbar, eine Verbindung zwischen den Szenen anzudeuten, um zu verdeutlichen, dass die Daten nicht fehlerhaft sind. Mit dieser Technik würden die Zeitstempel innerhalb der einzelnen Szenen erhalten bleiben, doch sind die Daten für einen Menschen eventuell schwer zu vergleichen.

Detaillierte Untersuchung einer bestimmten Szene

Eine der Möglichkeiten, wie die Szenenunterteilung genutzt werden kann, ist, eine der Szenen im Detail zu untersuchen. Dabei werden alle anderen Szenen, ausgeblendet. Einer der großen Vorteile dieser Interaktionsmöglichkeit ist, dass dem Studienanalysten eine deutlich bessere Übersicht über die einzelne Szene gegeben wird. Das gilt vor allem, wenn die Szene im Verhältnis zu den anderen Szenen der Aufnahme sehr wenig Platz in der Visualisierung zugewiesen bekommt.

Sollte der Studienanalysierende eine Szene genauer untersuchen wollen oder zwei Szenen, die nicht aufeinander folgen, vergleichen wollen, so kann er bestimte Szenen entweder einklappen und hierdurch ihren Platzverbrauch deutlich reduzieren oder einzelne Szenen vollständig ausblenden. Durch diese Funktionalität wird es möglich, einzelne, nicht aufeinanderfolgende Szenen miteinander zu vergleichen. Eingeklappte Szenen können zu einem späteren Zeitpunkt wieder ausgeklappt werden.

5.6.3. Unterstützung zur Erkennung wiederkehrender Muster

Wenn während des Analyseschritts wiederkehrende Muster erkannt wurden, kann dies vom Renderer in der Visualisierung besonders hervorgehoben werden. Dies kann auf unterschiedliche Arten realisiert werden:

Grafisch hervorheben: Es besteht die Möglichkeit, den PSP an den Stellen mit wiederkehrenden Mustern hervorzuheben. Dies kann mittels der Veränderung einer Eigenschaft wie der Dicke oder der Farbe des PSPs an der betroffenen Stelle geschehen. Diese Änderung kann so gestaltet werden, dass die Muster präattentiv erkannt werden können.

Markierung mittels Glyphen: Zudem kann man die Muster gänzlich durch einzelne Glyphen ersetzen, die dann die wiederkehrenden Muster symbolisieren oder die entsprechenden Abschnitte durch Glyphen ergänzen.

Es ist auch möglich, diese beiden Techniken zu kombinieren und somit die wiederholt vorkommenden Muster noch deutlicher hervorzuheben.

5.6.4. Visualisierung des AOI-Managements und seine Interaktionsmöglichkeiten

Um eine Hierarchie der AOIs zu visualisieren, kann man die zuvor bereits beschriebene Baumstruktur der AOIs anzeigen, indem man sie beispielsweise gemäß Abbildung 5.6 anzeigt. Dabei ist die obere AOI die äußere AOI. Falls auch die untergeordneten AOIs berücksichtigt werden sollen, werden in der unteren Reihe alle untergeordneten AOIs und die übergeordnete AOI aneinandergereiht.

Abbildung 5.6.: Schematische Darstellung des AOI-Managements. Das AOI-Management enthält drei AOIs, „AOI 1“, „AOI 2“ und „AOI 2.1“. „AOI 2.1“ liegt dabei in „AOI 2“. Damit die PSPs auch Treffer in „AOI 2“ darstellen können, wenn geschachtelte AOIs dargestellt werden, hat „AOI 2“ auch in der Detailansiche ein eigenes Feld.

5.6.5. AOI-Schachtelung und deren Möglichkeiten

In [7] wird im Allgemeinen davon abgeraten, sich (teilweise) überschneidende AOIs zu definieren. Es gibt jedoch eine Vielzahl an Szenarien, in denen es trotzdem sinnvoll ist, ineinanderliegende AOIs zu definieren. Es ist beispielsweise denkbar, dass der Stimulus die Abbildung einer Wasserflasche mit aufgeklebtem Ettikett ist. Hier wäre es denkbar, dass um die Flasche als Ganzes eine AOI definiert wird, das Etikett im speziellen jedoch ebenfalls mit einer AOI annotiert werden soll. Es bestehen grundsätzlich mehrere Möglichkeiten, diese AOIs in einem Datenmodell abzubilden:

Hierarchische AOIs: Es ist denkbar, die AOIs hierarchisch anzuordnen. Hierbei werden die AOIs im Datenmodell des AOI-Managements in einer Baumstruktur angeordnet und der Hit-Test müsste mittels eines Top-Down-Ansatzes für jede Ebene des Baums einen Hit-Test innerhalb des zuletzt getroffenen Knotens durchführen. Der Vorteil dieser Technik ist, dass die einzelnen AOIs geometrisch deutlich einfacher aufgebaut sind. Der Nachteil ist, dass mehrere Hit-Tests ausgeführt werden müssen, wenn die AOIs ineinander geschachtelt sind. Zudem ist eine teilweise Überlappung der AOIs bei dieser Struktur sehr problematisch, da nicht klar ist, wo die AOI eigentlich im Baum eingetragen werden müsste. Die einzige Möglichkeit, dieses Problem anzugehen, ist, dass die AOI in mehrere AOIs aufgeteilt werden, damit es keine teilweisen Überlappungen mehr gibt. Daraufhin müsste geprüft werden, ob eine getroffene AOI Teil einer zuvor aufgeteilten AOI war, damit die gesamte AOI als Treffer zurückgegeben werden kann.

AOIs mit komplexen Strukturen: Alternativ kann man die Flaschen-AOI mittels eines Linienzuges so zu definieren, dass die Stelle, an der sich die Etikett-AOI befindet, nicht in der Flaschen-AOI enthalten ist. Diese Art AOIs zu definieren ist jedoch sehr aufwendig und es besteht grundsätzlich die Möglichkeit, dass Probleme am Rand des Polygons auftreten. Eines dieser Probleme ist, dass nicht klar ist, ob die Grenze des Polygonzuges in diesem Beispiel noch zur Flaschen-AOI oder zur Etikett-AOI gehört. Ein scheinbarer Vorteil dieser Technik ist, dass die Struktur des Datenmodells der AOIs flach ist und ein Hit-Test für Fixationen nur auf einer Ebene durchgeführt werden muss. Da die Struktur der AOI jedoch nicht notwendigerweise konkav ist, muss gegebenenfalls in einem Vorverarbeitungsschritt die AOI zunächst in mehrere kleine AOIs aufgeteilt werden, um dieses Problem zu umgehen.

Auflösung von komplexen Strukturen mittels Hierarchien: Um das Problem der komplexen Geometrien der AOIs mit komplexen Strukturen zu lösen, ist es denkbar, die komplexen AOIs einmalig zu triangulieren und die entstehenden Dreiecke in einer Baumstruktur als Kindelemente des Polygons einzutragen und bei der Trefferabfrage nur Hit-Tests auf alle Blattknoten des Baums durchzuführen. Der Vorteil dieser Technik besteht darin, dass die getesteten Elemente ausschließlich primitive Geometrien sind, die effizient von einer Grafikkarte bezüglich eines Hit-Tests verarbeitet werden könnten. Andererseits ist die Anzahl der zu testenden Elemente bei komplexen Formen aufgrund der Triangulation vermutlich sehr hoch. Zudem ähnelt dieser Ansatz dann stark den zuvor beschriebenen hierarchischen Areas of Interest.

Detaillierte Untersuchung einer bestimmten AOI

Analog zu einer detaillierten Untersuchung von Szenen ist es ebenfalls möglich, Areas of Interest komplett oder nur ihre untergeordneten AOIs auszublenden. Wenn AOIs als ganzes ausgeblendet werden, besteht die Möglichkeit, die Korrelation weniger AOIs miteinander zu vergleichen. Wenn nur die untergeordneten AOIs ausgeblendet werden, erzielt man eine stärkere Vergleichbarkeit der größeren Regionen eines Stimulus.

5.6.6. Die Bedeutung der Anordnung der AOIs

Es gibt grundsätzlich zwei Szenarien, die bei AOIs in Stimuli zutreffen können:

- Die AOIs haben eine semantische Abhängigkeit, wie beispielsweise das Etikett, das auf einer Flasche klebt.
- Die AOIs haben keine (offensichtliche) semantische Abhängigkeit, wie etwa zwei willkürliche Laubblätter auf einem Herbstbild im Wald.

Im ersten Fall ist es sinnvoll, die AOIs, die zusammenhängen, in der Visualisierung nahe aneinander anzuordnen. Da die AOIs im AOI-Management nebeneinander angeordnet sind, lässt sich die Anordnung der AOIs als optimale topologische Anordnung der AOIs interpretieren. Die Abhängigkeiten der AOIs beschreiben dabei die Verbindungen der Elemente der topologischen Ansicht.

In beiden Fällen besteht das Problem, dass es zwar möglich ist, manuell weitere Abhängigkeiten zu erkennen, doch im Allgemeinen muss man damit rechnen, dass nicht erkannte Korrelationen vorliegen können. In Folge dessen sollten AOIs bezüglich ihrer Korrelationen untersucht und dann entsprechend angeordnet werden. Es sollten also AOIs, zwischen denen es häufig Transitionen gibt, in der Nähe voneinander angeordnet werden. Es ist möglich, die AOIs manuell oder automatisch mit Hilfe einer Transitionsmatrix anordnen.

Manuelle Anordnung der AOIs

Die einfachste Art, AOIs anzuordnen, ist eine manuelle Anordnung. Der Vorteil besteht darin, dass der Studienanalysierende den Inhalt des Stimulus kennt und daher die semantischen Zusammenhänge der enthaltenen Elemente, und damit der AOIs, besser auswerten kann, als dies ein automatisierter Algorithmus, etwa mittels Computer Vision, könnte. Das manuelle Anordnen von AOIs ist jedoch nur noch eingeschränkt möglich, wenn die vorliegenden Datensätze berücksichtigt werden sollen, da ein Computer diese wesentlich schneller und effizienter untersuchen kann.

Automatische Anordnung der AOIs mittels einer Transitionsmatrix

Es bieten sich für die automatische Anordnung der AOIs mit Hilfe einer Transitionsmatrix zwei Ansätze an. Man kann die Daten lokal oder global optimieren. Wenn man die Daten auf einer lokalen Ebene optimiert, bedeutet das, dass nur die direkten Zusammenhänge der AOIs berücksichtigt werden. Dieser Ansatz erkennt schnell direkte Zusammenhänge. So ist es naheliegend, dass beispielsweise ineinander liegende AOIs häufig Übergänge zueinander aufweisen. Aufgrund dieser Eigenschaft würden die AOIs automatisch in der Visualisierung des AOI-Managements sehr nah zueinander angeordnet werden. Auch werden AOIs einander zugeordnet, bei denen Übergänge häufig auftreten. Wenn man beispielsweise einen Wert in einem Balkendiagramm abliest, so ist es sehr wahrscheinlich, dass der Blick, nachdem man den gesuchten Balken gefunden hat, zwischen dem Balken und der Wertachse springt, um den Wert besser abschätzen zu können. In diesem Vorteil liegt gleichzeitig auch ein Nachteil. So werden AOIs einander zugeordnet, die möglicherweise keinen Zusammenhang haben oder deren Übergänge Zufall sind. Bei einer ausreichend hohen Anzahl von Transitionen wird jedoch die Auswirkung von Effekten wie Ausreißern oder zufälligen Übergängen, die man als Rauschen in den Daten interpretieren kann, reduziert. Zudem ist nicht klar, wie die AOIs angeordnet werden sollen, wenn sich bezüglich der Abhängigkeit Gruppen bilden, die zueinander nur schwache Abhängigkeiten aufweisen. In diesem Fall müssten die Gruppen vermutlich zufällig angeordnet werden, da man keine Informationen aus den lokalen Daten extrahieren kann.

Es ist auch möglich, die Daten der Transitionsmatrix global zu optimieren, also nicht nur direkte Zusammenhänge zu betrachten. Hierdurch entsteht nicht das Problem, wie die AOIs, die nicht direkt zusammenhängen, angeordnet werden sollen. Es bietet sich also an, die Transitionsmatrix durch statistische Korrelationen zu unterstützen und somit die Abhängigkeiten der AOIs zueinander zu bestimmen.

Herausforderungen bei der Anordnung von AOIs

Mit der zunehmenden Anzahl an Probanden und AOI-Übergängen nimmt der Visual Clutter unweigerlich zu, wodurch die globale Anordnung der AOIs immer wichtiger wird. Wann dieser Effekt eintritt, hängt maßgeblich von der Häufigkeit von Transitionen der einzelnen Probanden ab und ob diese Transitionen zu ähnlichen Zeitpunkten eingetreten sind. Das Problem ist dabei, dass der Visual Clutter trotzdem zunimmt, da jeder Mensch ein anderes Blickverhalten aufweist. So ist bei Weitem (auch mit den Gruppierungstechniken aus Kapitel 5.5.10) nicht gegeben, dass sich die Blickpfade der Probanden vollkommen überschneiden. Zum einen nimmt die Wahrscheinlichkeit, dass die Blickpfade voneinander abweichen, mit zunehmender Aufnahmedauer zu und zum anderen weichen die Verweildauern auf den AOIs voneinander ab. Hierdurch ergeben sich bei den PSPs Überschneidungen bei den Pfaden, die sich nicht verhindern und deren negative Effekte bezüglich der Nachvollziehbarkeit der Pfade sich nur bedingt abmildern lassen.

5.6.7. Einbindung externer Daten

Die Visualisierung externer Daten ist sinnvoll, wenn während der Eye-Tracking-Aufnahme zusätzliche Daten, wie etwa EEG-Daten[1] oder Interaktionen, aufgezeichnet wurden. Solche zusätzlichen Daten aufzuzeichnen ist aus meheren Gründen sinnvoll:

1. Interaktionen haben häufig ein großes Ablenkungspotenzial, da sie, in unterschiedlicher Stärke, die Konzentration des Probanden von der Eye-Tracking-Aufgabe ablenken.
2. Manche Interaktionen oder Ereignisse lösen kurz vor oder direkt nach ihrem Auftreten bestimmte Blickmuster aus, die entweder wichtige Charakteristika sind oder die genauer untersucht werden können.

Je nachdem, welche Art von Daten von den externen Daten geliefert werden, unterscheidet man zwischen verschiedenen Datentypen, den Verlaufs- und den Eventdaten, die im Folgenden vorgestellt werden.

Verlaufsdaten

Wenn die Daten kontinuierlich sind, kann man diese als parallel zu der Eye-Tracking-Aufnahme aufgezeichnete Verlaufsdaten auffassen. Diese Daten können je nach Kontext der Eye-Tracking-Aufzeichnung unterschiedlichster Natur sein. So könnten Studien aus der Psychologie an

- EEG-Werten,
- Hautwiderstand,
- Puls oder
- Pupillengröße

[1] EEG $\triangleq$ Elektroenzephalografie: eine Technik zur Aufzeichnung der elektrischen Aktivität des Gehirns [5].

interessiert sein. Durch die EEG-Werte ist es beispielsweise möglich, bestimmte Krankheitsbilder oder, in Verbindung mit anderen Kennwerten, physische Eigenschaften wie Stress zu erkennen.

In der Automobilindustrie könnten zudem die verschiedenen Messwerte des Bordcomputers wie

- Bremskraft,
- Drehzahl,
- Geschwindigkeit,
- u.v.m.

von Interesse sein. So mag es beispielsweise untersuchenswert sein, ob ein bestimmtes Blickverhalten mit einem Wert, wie der Bremskraft, korreliert.

Eventdaten

Wenn Daten nicht kontinuierlich vorliegen, sondern nur auf einem Kanal einzelne Signale senden, so kann man diese als Ereignisse oder Events auffassen. Zu diesen Daten gehören

- Tasteneingaben,
- Mausereignisse wie Links- und Rechtsklick,
- das Betätigen eines Hebels, beispielsweise eines Blinkers am Auto.

Diese Daten sind aus Sicht des Analysierenden deshalb so interessant, da sie häufig mit bestimmten Denkprozessen verbunden sind. Aufgrund der Eye-Mind-Theorie besteht die Annahme, dass der Mensch ansieht, über was er nachdenkt. Mit dieser Annahme kann man das Auslösen des Ereignisses als Trigger ansehen, dem ein bestimmter Denkprozess vorausgegangen ist. Dieser Denkprozess lässt sich dann eventuell kurz vor oder nach dem Event durch bestimmte Blickmuster nachvollziehen. Das langfristige Ziel einer solchen Untersuchung ist die Echtzeiterkennung des Beginns eines solchen Blickmusters, um somit auf den aktuellen Denkprozess rückzuschließen.

5.6.8. Zusätzliche Anzeige des Stimulus

Neben der Anzeige zusätzlicher Daten ist die Anzeige des ursprünglichen Stimulus, eventuell mit zusätzlichen, bereits etablierten Visualisierungstechniken, eine sinnvolle Ergänzung der PSPs, da diese die Blickpfade auf einer sehr abstrakten Ebene darstellen.

Darstellung des Stimulus

Der Stimulus sollte neben des eigentlichen Bildes oder Videos auch die definierten AOIs darstellen. Bei Bildern sind diese statisch, doch bei Videos können diese vom Zeitpunkt in dem Video abhängen. Dies muss zum einen bei der Darstellung der AOIs auf dem Video und zum anderen bei einer bei Videos möglichen Rückkopplung auf die PSPs beachtet werden.

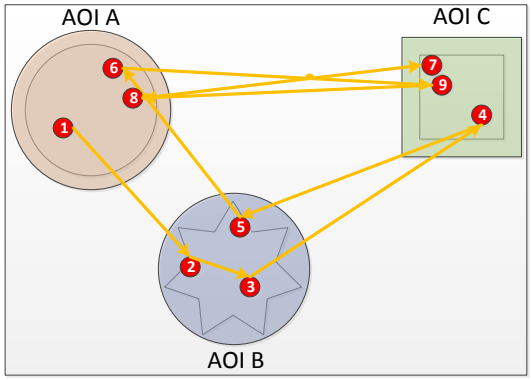

Abbildung 5.7.: Schematische Ansicht der Anzeige des Stimulus. Es werden sowohl der Stimulus als auch die drei darauf definierten AOIs angezeigt. Zusätzlich wird der Scanpath eines Probanden dargestellt.

Zudem kann die Anzeige des Stimulus dahingehend verbessert werden, dass dem Studienanalysierenden eine Verbindung zwischen den Daten aus dem PSP und dem Stimulus gegeben wird. Hierzu kann man den Stimulus um weitere Visualisierungstechniken, wie etwa einen Scanpath für statische oder einen Bee-Swarm für dynamische Stimuli, erweitern. Eine schematische Darstellung des zusätzlich eingeblendeten Stimulus ist in Abbildung 5.7 zu sehen.

5.6.9. Einblendung verschiedener Statistiken

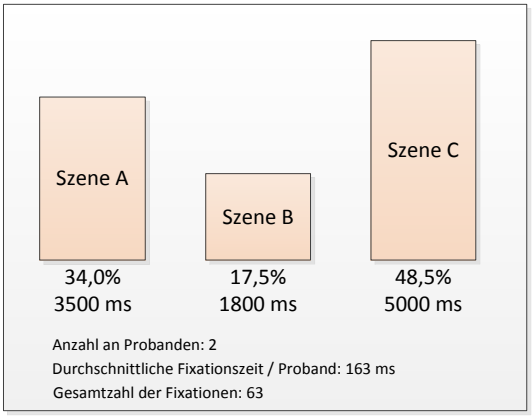

Abbildung 5.8.: Schematische Ansicht der Anzeige der Statistik. Es werden allgemeine Informationen über den geladenen Datensatz, wie etwa die Aufenthaltszeiten in den verschiedenen Szenen oder die Probandenanzahl, ausgegeben.

Unterhalb des AOI-Managements können zusätzliche Statistiken, wie etwa die durchschnittliche Aufenthaltszeit innerhalb einer AOI oder die durchschnittliche Fixationszahl pro Proband, angezeigt werden, wie in Abbildung 5.8. Diese Informationen können auch kon-

textabhängig sein. So kann sich die Statistik beispielsweise nur auf den Probanden beziehen, dessen Transitionspfad momentan hervorgehoben wird. Dabei können die Informationen je nach Metrik unterschiedlich angezeigt werden. Grundsätzlich bieten sich zwei Möglichkeiten an, die Statistiken anzuzeigen:

Numerische Statistiken: Es ist möglich, die Statistiken rein als Text mit Zahlen darzustellen. Dies bietet sich an, wenn die Statistik eine eigenständige Aussagekraft hat, wie etwa die durchschnittliche Aufnahmedauer des Stimulus pro Proband.

Visualisierungsgestützte Statistiken: Statistiken, die erst im Kontext von anderen Werten aussagekräftig werden, lassen sich mit Visualisierungen wie Balkendiagrammen oder Liniendiagrammen gut darstellen. Ein Beispiel hierfür wäre die durchschnittliche Aufnahmelänge der verschiedenen Szenen.

5.7. Optionale Rückkopplung der visuellen Filterung

Ein Kernelement der Visual Analytics ist, dass die aus der Visualisierung gewonnenen Informationen oder Erkenntnisse eine Rückkopplung auf die Analyse haben können. In diesem Konzept können beispielsweise die PSP-Daten dahingehend verändert werden, dass die Analyse nur Daten berücksichtigt, die in der Visualisierung auch angezeigt werden. So wird es dem Studienanalysierenden ermöglicht, mehr Einsicht in bestimmte Teile der Eye-Tracking-Daten zu erhalten.

5.7.1. Absolute und relative Datensätze

Aus der optionalen Rückkopplung der Daten ergibt sich, dass die Analyse-Plug-ins zwischen zwei Arten von Daten unterscheiden können müssen:

Absolute Daten: In dieser Arbeit sind absolute Daten die Daten, die in der Datenbank vorliegen und vollständig aus dieser abgerufen und dann weiterverarbeitet werden. Es ist hierbei nicht vorgesehen, dass die Daten nur teilweise, etwa nur zwei von fünf verfügbaren Szenen, abgerufen werden.

Relative Daten: Relative Daten sind die Daten, die bereits in der Visualisierung gefiltert wurden. Da diese in der Datenbank so möglicherweise nicht vorliegen, müssen diese zunächst so verarbeitet werden, dass sie in der Datenbank abgebildet werden können, damit die Analyse-Plug-ins mit ihnen arbeiten können.

Jedes Plug-in muss also dem Studienanalysierenden die Wahl geben, ob dieser mit den absoluten Daten oder mit relativen Daten arbeiten möchte. Hierbei soll auch gewährleistet werden, dass keine Daten permanent verloren gehen, sodass der Studienanalysierende zuvor gefilterte Daten wiederherstellen kann.

5.8. Einschränkungen des Konzepts

Obwohl die ETA-Pipeline mit den PSPs als Visualisierungstechnik ein gutes Konzept zur Exploration von Blickmustern ermöglicht, so hat dieses Konzept doch Einschränkungen. So

ist beispielsweise der Nutzen der Filter in der ETA-Pipeline stark davon abhängig, welche Visualisierungstechnik verwendet wird. So ist es beispielsweise kaum möglich, mit Hilfe von Heat Maps Gruppierungen darzustellen. Dieses Konzept nutzt daher die Parallel Scan-Paths zum Visualisieren der Daten. Diese skalieren zwar deutlich besser über die Zeit als andere Visualisierungstechniken wie beispielsweise Scan-Paths, doch ist die Anordnung einer hohen Anzahl an AOIs ein bislang ungelöstes Problem. Des Weiteren können die PSPs nicht direkt auf den Stimulus abgebildet werden, was die Bildung einer Mental Map hemmt. Auch mit der zusätzlichen Kopplung des Stimulus an die PSPs, wie in Abschnitt 5.6.8 beschrieben, wird dieses Problem nur teilweise behoben.

6. Implementierung

Das folgende Kapitel stellt die Implementierung des Prototyps vor. Hierbei werden zunächst die verwendeten Programme und Tools aufgezählt (Abschnitt 6.1). Daraufhin wird auf die Datenbank, die für die Datenverwaltung zuständig und für die Performance des Programms unverzichtbar ist, eingegangen (Abschnitt 6.2).

Im Anschluss wird die Funktionsweise des Imports und des Matchings der Daten in die Datenbank erläutert (Abschnitt 6.3). Daraufhin werden die Schnittstellen zur Datenbank vorgestellt (Abschnitt 6.4). Da der Prototyp als Plug-in-System entworfen wurde, wird daraufhin das von Microsoft zur Verfügung gestellte Managed Extensibility Framework, kurz MEF, vorgestellt (Abschnitt 6.5). Anschließend wird auf die Datenverwaltung zwischen den Analysekomponenten eingegangen (Abschnitt 6.6). Daraufhin wird auf den Aufbau der grafischen Oberfläche der Analyse eingegangen (Abschnitt 6.7). Hierbei wird auch beschrieben, wie und wann die Plug-ins geladen werden. Im Anschluss wird die Funktionsweise der einzelnen Plug-ins aufgezeigt (Abschnitt 6.8). Hierbei handelt es sich um die Gruppen- und Probandenauswahl, sowie das hierarchische Clustering, das als Abstandsfunktion die Levenshtein-Distanz nutzt.

Nachdem alle Analysekomponenten vorgestellt wurden, folgt der Aufbau der Visualisierungsoberfläche (Abschnitt 6.9). In diesem Zusammenhang wird auch die Implementierung der Parallel Scan-Paths und deren Erweiterungen aufgezeigt. Zuletzt wird auf die Interaktionsmöglichkeiten der Visualisierungsoberfläche eingegangen (Abschnitt 6.10).

6.1. Verwendete Programme und Tools

Der Prototyp wurde in der Programmiersprache C# programmiert. Als Entwicklungsumgebung wurde hierzu Visual Studio 2012 mit dem .NET Framework 4.5 genutzt. Die genutzte Datenbank ist eine lokal laufende MSSQL-Server-Datenbank, die mittels der von Microsoft zur Verfügung gestellten Entity Framework angesteuert wird. Zur weiteren Datenbankverwaltung wurde das SQL Server Management Studio 2012 genutzt. Der Prototyp wurde unter Windows 8 entwickelt und ist bis Windows Vista SP2 abwärtskompatibel. Windows XP wird aufgrund der fehlenden Unterstützung von .NET 4.5 nicht unterstützt.

6.2. Zugrunde liegendes Datenbankmodell

Das genutzte Datenbankmodell ist sehr komplex. Daher wird in Abbildung 6.1 nur der Ausschnitt der Datenbank, der für die eigentliche Analyse und Visualisierung der Eye-Tracking-Daten relevant ist, abgebildet. Eine vollständige Abbildung der Datenbank findet sich in Anhang A. In Tabelle 6.1 werden die relevanten Datenbanktabellen und deren Zweck aufgelistet:

Tabellenname	Zweck
AreaOfInterest-Group	Dient der Gruppierung von AOIs. Daher enthält eine AreaOfInterestGroup auch mehrere AreaOfInterest.
AreaOfInterest-Point	Dient der geometrischen Definition einer AreaOfInterest. Es werden zwei- und dreidimensionale Punkte zur Definition eines Polygons unterstützt. Um den Polygonzug in der richtigen Reihenfolge auszulesen, wurde ein OrderIndex hinzugefügt. Des Weiteren kann der zwei- bzw. dreidimensionale Punkt als Zentrum einer Ellipse genutzt werden. Um Ellipsen zu unterstützen, können optional Breite und die Rotation der Ellipse um ihren Schwerpunkt gespeichert werden.
AreaOfInterest	Identifiziert alle importierten AOIs, optional mit Namen. Mittels der Verweise AreaOfInterestChild bzw. AreaOfInterestParent kann man bei geschachtelten AOIs zwischen den Eltern und den Kindern wechseln.
Filtered_AOI_ Fixation	Dient dem Speichern von Daten. Um einen bestimmten Zeitpunkt der Analyse wiederherstellen zu können, werden in dieser Tabelle Verweise zu den verschiedenen bereits entstandenen Gruppen abgelegt. Zur besseren Identifikation enthält die Tabelle ein Feld zur Benennung des Zwischenstands, einen Datumsstempel sowie eine Beschreibung.
GroupsInUse	Diese Verbindungstabelle zwischen der Filtered_AOI_Fixation-Tabelle und der GroupedAoiFixations-Tabelle ermöglicht es, für jede Filtered_AOI_Fixation einzelne Gruppen auszuwählen.
GroupedAoi-Fixations	Enthält alle Gruppen, die jemals während der Analyse entstanden sind. Jede Gruppe enthält dabei eine beliebige Anzahl an Probanden.
ParticipantsInUse	Diese Verbindungstabelle zwischen der Participant-Tabelle und der GroupedAoiFixations-Tabelle ermöglicht es, Teilnehmer innerhalb der einzelnen Gruppen auszuwählen oder besonders hervorzuheben.
Participant	Hierdurch werden die einzelnen Teilnehmer einer Studie repräsentiert. Ein Teilnehmer hat eine id und einen „Namen“. Dieser Name entspricht dem Identifikationsnamen innerhalb der Studie, wie etwa „Participant 01“. Zudem werden Start- und Endzeitpunkt der Studie, sowie ein Farbwert, der dem Probanden einen konstanten Farbwert über alle Analysen, zuweist, hinterlegt.
Fixation	Repräsentiert für einen bestimmten Probanden für einen bestimmten Stimulus eine Fixation. Diese Fixation enthält dabei Informationen über das Datum der Aufnahme und deren Zeitzone, einen Zeitstempel innerhalb der Eye-Tracking-Aufnahme, die X- und Y- Position der Fixation auf dem Stimulus in Pixeln, die Pupillengröße, die Dauer der Fixation, den Index der Fixation sowie den GazeCount. Dieser gibt an, wie viele Blicke in die Fixation aufgenommen wurden.

AOI_Fixation	Eine AOI_Fixation gibt ein Element im Transitionspfad für die PSPs an. Damit die Fixationen zwischen der Eintritts- und der Austritts-AOI nicht verloren gehen, gibt es für jede AOI_Fixation eine FixationList, die die in dem Transitionspfadelement liegenden Fixationen beinhaltet. Sollten keine Fixationen enthalten sein, so wird diese Verknüpfung nicht angelegt.
FixationList	Diese Tabelle enthält die Fixationen, die nicht Teil des Transitionspfades sind. Hierdurch können weitere Informationen bei den PSPs, wie Statistiken oder die einzelnen Fixationen, eingeblendet werden.
Scene	Ein Stimulus kann a priori in verschiedene Szenen unterteilt werden. Die Szeneninformationen werden in dieser Tabelle hinterlegt, damit die PSPs später in Szenen unterteilt werden können. Eine Szene enthält einen Namen, Start- und Endzeitpunkt sowie den zugehörigen Stimulus und die enthältenen Transitionspfadelemente.
Stimulus	Die Stimulustabelle enthält die einzelnen Stimuli einer Studie. Sie enthält eine Dateireferenz, die den gezeigten Stimulus bzw. zusätzliches Datenmaterial enthält.
File	Enthält eine binarisierte Datei. In dieser Arbeit handelt es sich bei der enthaltenen Datei ausschließlich um Bilder oder Videos, die entweder direkt den Stimulus anzeigen, oder eine Aufnahme der Studie beinhalten. Die Dateitabelle ist aus dem Stimulus ausgelagert, damit sie auch von anderen Komponenten der Datenbank genutzt werden kann.

Tabelle 6.1.: Auflistung aller relevanter Datenbanktabellen mit Beschreibung. Die Tabellen sind nach thematischer Zusammengehörigkeit, und daher nicht alphabetisch, sortiert.

6.3. Funktionsweise des Imports und des Matchings der Daten in die Datenbank

Der folgende Abschnitt beschäftigt sich zunächst mit dem Import der Eye-Tracking-Daten der verschiedenen Eye-Tracking-Systeme in die Datenbank. Dieser Import wird durch ein Tool bewerkstelligt, dass die Eye-Tracking-Daten aus einer Datei ausliest und in die Daten in die entsprechenden Datenbanktabellen einfügt. Anschließend geht es um den Import der AOIs, der Szenen und welche geometrischen Strukturen unterstützt werden. Danach wird erläutert, wie die Eye-Tracking-Daten mit den AOIs gematcht werden.

6.3.1. Import der unverarbeiteten Eye-Tracking-Daten

Der Importer des Prototyps unterstützt zwei Eye-Tracking-Datenformate:

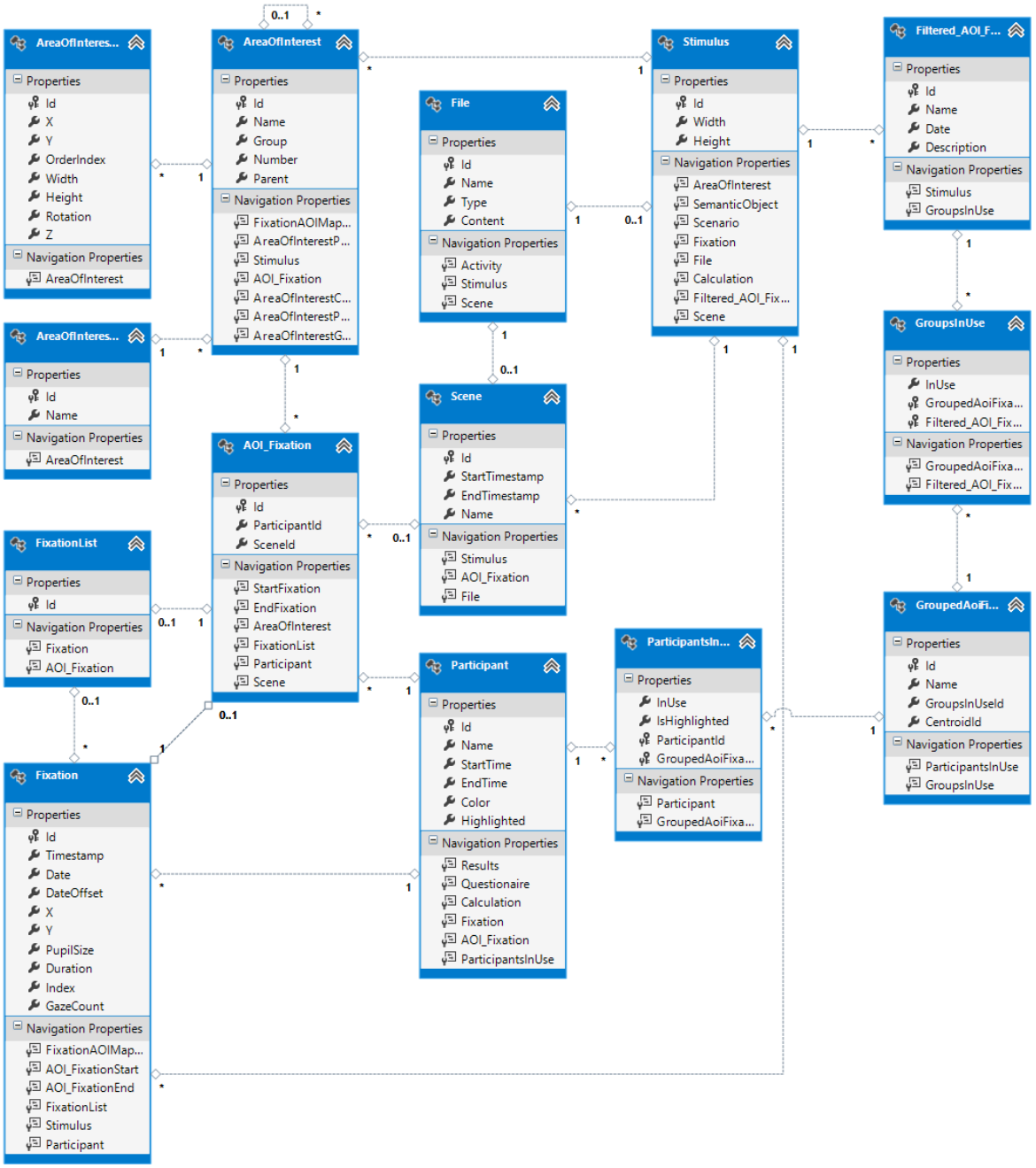

Abbildung 6.1.: Relevanter Ausschnitt des Datenbankschemas des Prototyps. Der Stimulus muss bei der Nutzung dieses Datenbankausschnittes bereits bekannt sein. Die für die PSPs wichtigste Tabelle ist die AOI_Fixation-Tabelle, welche die gematchten Daten der Fixationen mit den AOIs enthält.

1. Die tab-separated value (.tsv) Dateien, die von der Tobii Studio Software bei Aufnahmen mit Eye-Trackern der Firma Tobii erzeugt werden.
2. Das durch die Daimler AG aufbereitete Datenformat eines dort verwendeten DSS Eye-Trackers von SeeingMachines.

Es ist problemlos möglich, weitere Formate hinzuzufügen.

Import von Tobii-Fixationen

Wenn Fixationen von einem Eye-Tracker der Firma Tobii importiert werden, wird eine Grundannahme getroffen: Ein Proband hat für jedes Projekt nur eine Aufnahme durchgeführt. Das bedeutet, dass es zu jedem Probanden auch nur ein Recording gibt. Diese Annahme wurde zur Vereinfachung der Datenbank getroffen und ist möglich, da mehrere Recordings in aller Regel nur entstehen, wenn die vorherigen Recordings fehlgeschlagen sind. Falls ein Proband mehrere Recordings durchführen soll, wird hier davon ausgegangen, dass mehrere Probanden erstellt werden.

Die Software Tobii Studio gibt für jeden Probanden eine eigene .tsv-Datei aus. Jede .tsv-Datei beginnt mit Metainformationen über die Studie, wie etwa den genutzten Filter für die Aufnahme der Daten und den Namen des Probanden, sowie den bei der Aufnahme verwendeten Computer und die verwendete Software. Danach folgen die eigentlichen Aufnahmedaten. Bei den Tobii-Daten gibt es drei relevante Informationsarten, die in eine Zeile kodiert werden können:

Stimulus: Falls ein neuer Stimulus beginnt, wird dies als Event geführt und es wird der Dateiname des angezeigten Stimulus mit angegeben.

Eingabeevent: Falls eine Taste gedrückt wurde oder ein anderes Eingabeevent auftritt, wird dies ebenfalls vermerkt. Bei den in dieser Arbeit genutzten Datensätzen wurden Tasteneingaben jedoch nur für den Wechsel des Stimulus genutzt. Daher werden diese Events momentan nicht in die Datenbank übernommen.

Fixation: Für jede Fixation werden vom Eye-Tracker diverse Informationen, wie beispielsweise die Blickposition, die Pupillengröße und den Zeitstempel, zur Verfügung gestellt. Da viele dieser Informationen jedoch redundant sind, liest der Importer nur die x- und y-Koordinate des rechten Auges, die Fixationsdauer, die Anzahl der in der Fixation enthaltenen Gazes und den Timestamp, sowie den Datumsstempel und die Zeitverschiebung durch die Zeitzone aus.

Falls eine der Fixationen ungültig sein sollte, so wird diese nicht importiert. Mögliche Ursachen hierfür sind beispielsweise, dass der Eye-Tracker die Augen des Probanden nicht mehr erkennen konnte oder dass die Verbindung vom Eye-Tracker zur Aufnahmesoftware nicht mehr vorhanden war.

Import von Daimler-Daten

Die Abteilung R&D, Driver Analysis, der Daimler AG zeichnet mit ihrem Eye-Tracker das Blickverhalten von Probanden im Cockpit eines PKWs auf. Da hierbei kein fester zweidimensionaler Stimulus mit fester Größe definiert ist, kann man das Koordinatensystem auch nicht

in der linken oberen Ecke des Stimulus definieren. Aus diesem Grund wird die Annahme getroffen, dass der Ursprung des Koordinatensystems an der Stelle ist, die getroffen wird, wenn der Proband geradeaus sieht. Die Daimler AG zeichnet des Weiteren die Abweichung von diesem Ursprung nicht in Pixeln, sondern in Metern auf. Da die Datenbank und der Renderer jedoch von einem Pixel-basierten Koordinatensystem ausgehen, werden die Abweichungen um den Faktor 10.000 multipliziert und dann als Pixelkoordinaten angenommen. Diese Annahme ist für das spätere Matching mit den AOIs wichtig. Zudem vereinigt der Eye-Tracker nicht Blickpunkte zu Fixationen. Der Importer nimmt vereinfachend an, dass ein Blickpunkt einer Fixation entspricht. Auch bei den Fixationen der Daimler AG werden ungültige Fixationen nicht übernommen.

6.3.2. Import von AOIs und Szenen

AOIs und Szenen müssen in einem angepassten XML-Format, hier Motif-Format genannt, vorliegen. Dabei muss zunächst ein Name für die Datei und der Stimulus, auf den sich die enthaltenen AOIs und den sich die Szenen beziehen, definiert werden. Anschließend können AOIs als Polygon oder Ellipsen definiert werden und Szenen angegeben werden. In Listing 6.1 findet sich ein kommentiertes Beispiel für eine einzelne Motif-Datei, in der zwei ineinander liegende AOIs und zwei Szenen definiert werden. Die Anordnung der AOIs, die sich hieraus ergibt, ist in Abbildung 6.2 zu sehen.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?> <!-- Definiere XML version mit Encoding
    -->
<Motif Name="Demo" Scale="50"> <!-- Gib den Namen fuer dieses Motif bekannt -->
 <Bitmap File="C:\Demo\MyStimulus.png" /> <!-- Definiere Stimulus, auf dem gearbeitet wird -->
 <AOIPolygon Number="0" Name="AOI 1"> <!-- Eine AOI als Polygon mit dem Namen 'AOI 1' -->
     <Punkt x="0" y="100" /> <!-- Ein einzelner Punkt, der Teil des Polygons ist -->
     <Punkt x="20" y="100" />
     <Punkt x="20" y="80" />
     <Punkt x="0" y="80" />
      <!-- In der folgenden Zeile wird eine AOI definiert, die in der vorherigen AOI liegt -->
     <AOIPolygon Number="1" Name="AOI 2">
       <Punkt x="0" y="100" />
       <Punkt x="8" y="100" />
       <Punkt x="8" y="92" />
       <Punkt x="0" y="92" />
      </AOIPolygon>
 </AOIPolygon>

     <!-- Definition einer Szene mit dem Namen 'Szene 1' und Start- und Endzeit -->
 <Scene StartTimestamp="0" EndTimestamp="50" Name="Szene 1">
 </Scene>
     <!-- Definition einer weiteren Szene mit dem Namen 'Szene 2' und Start- und Endzeit -->
 <Scene StartTimestamp="50" EndTimestamp="100" Name="Szene 2">
 </Scene>
</Motif>
```

Listing 6.1: Codebeispiel einer Motif-Datei mit zwei ineinander liegenden AOIs und zwei Szenen. Die AOIs enthalten dabei jeweils vier Punkte. Diese dienen der Definition des Polygonzugs der AOI.

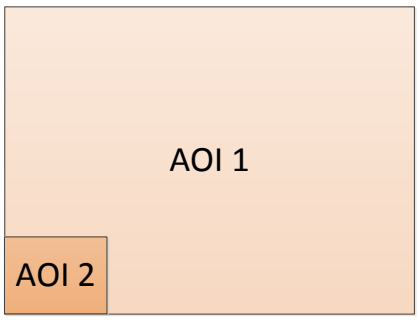

Abbildung 6.2.: Darstellung der in dem Beispiel der Motif-Datei aus Listing 6.1 gegebenen AOIs. Es werden zwei AOIs definiert, wobei „AOI 2“ in „AOI 1“ enthalten ist.

Der AOI- und Szenen-Importer liest die AOIs rekursiv, beginnend bei der äußersten, aus und baut aus ihnen die AOI-Struktur auf, die dann in die Datenbank eingefügt wird. Der Stimulus wird an dem angegebenen Pfad gesucht und dann in binarisierter Form in die Datenbank geladen, sofern die Datei dort noch nicht existiert. Sollte die Datei schon existieren, wird nur ein Verweis auf die Datei angelegt.

6.3.3. Matching der Eye-Tracking-Daten

Nachdem sowohl alle Fixationsdaten als auch die Stimuli und die AOIs in die Datenbank geladen wurden, können diese gematcht werden. Das Matching-Tool lädt alle relevanten AOIs aus der Datenbank und zeichnet diese auf eine Canvas. Anschließend werden alle zu dem Stimulus gehörenden Fixationen durchlaufen. Deren Koordinaten werden an die `HitTest`-Methode von WPF übergeben und die AOI zurückgegeben, in der die Fixation liegt. Sollte die Fixation in keiner AOI liegen, so wird die Fixation ignoriert. Andernfalls überprüft das Tool, ob die zurückgegebene AOI in der AOI-Hierarchie am tiefsten liegt und korrigiert gegebenenfalls das Ergebnis. Das bedeutet, dass der Algorithmus sicherstellt, dass, falls die Fixation zwei ineinanderliegende AOIs gleichzeitig trifft, die am weitesten ’innen’ liegende AOI zurückgegeben wird.

Es gibt mehrere Szenarien für die Verarbeitung der Fixationen, die bedacht werden müssen, um die Daten korrekt in die Datenbank einzutragen:

1. Die Fixation trifft eine AOI und die vorherige Fixation lag nicht in der selben AOI:
   - Es muss ein neuer Eintrag in der AOI_Fixation-Tabelle erzeugt werden, mit der aktuellen Fixation als Startfixation
   - Falls die vorherige Fixation in einer AOI lag, muss die letzte AOI_Fixation die vorherige Fixation als Endfixation erhalten und in die Datenbank geladen werden.
2. Die Fixation trifft eine AOI und die vorherige Fixation lag in der selben AOI:
   - Die Fixation wird zur Fixationsliste der aktuellen AOI_Fixation hinzugefügt.
3. Die Fixation trifft keine AOI

- Falls bereits eine AOI_Fixation begonnen wurde, wird die vorherige Fixation als Endfixation zugewiesen.
- Die aktuelle Fixation wird verworfen.

Nachdem alle Fixationen bearbeitet wurden, muss noch einmalig überprüft werden, ob es noch eine begonnene AOI_Fixation gibt. Wenn dem so ist, so muss diese mit der letzten Fixation als Endfixation abgeschlossen werden.

6.4. Die Schnittstellen zur Datenbank

Um den Umgang mit der Datenbank zu vereinfachen, wurde für den Prototypen eine Drei-Schichten-Architektur gewählt. Das bedeutet, dass es zwischen der Präsentationsschicht, also dem Frontend des Programms und der Datenhaltungsschicht, also in diesem Fall der MSSQL-Datenbank, eine Logikschicht gibt. Die Aufgabe dieser ist die Verwaltung der Anfragen der Präsentationsschicht an die Datenhaltungsschicht. Dies hat nicht nur den Vorteil, dass das Frontend keine Datenbankanfragen stellen muss. Zusätzlich ist das Frontend robust gegenüber Veränderungen des Datenbankmodells, da nur die Logikschicht hiervon betroffen wäre. Eine Illustration dieses Aufbaus ist in Abbildung 6.3 gegeben. Dabei werden die Zusammenhänge der Schichten für den Prototypen dargestellt.

Im Folgenden werden die Klassen der Logikschicht, das Data-Management und das AOI-Management, vorgestellt.

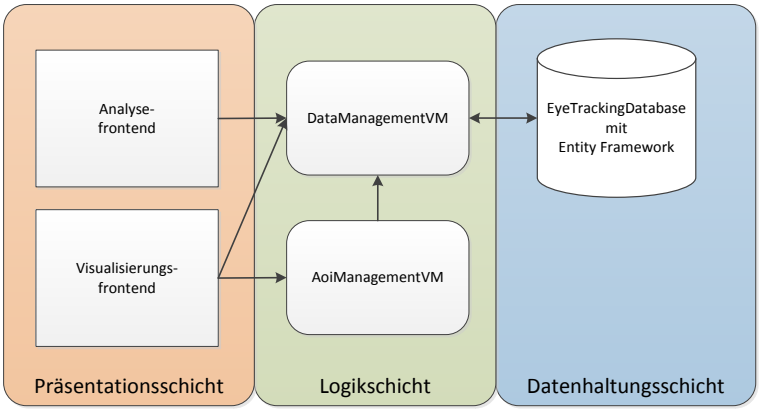

Abbildung 6.3.: Darstellung des Drei-Schichten-Modells des Prototyps. Die Präsentationsschicht besteht aus dem Analyseteil, der im Konzept als ETA-Pipeline eingeführt wurde, und dem Visualisierungsfrontend, das die interaktiven PSPs repräsentiert. In der Logikschicht gibt es das DataManagementVM, welches für jegliche Datenbankabfragen zuständig ist und das AoiManagementVM, das die Interaktionen mit den AOIs ermöglicht. Dieses nutzt wiederum das DataManagementVM, um die AOIs aus der Datenbank abzufragen und zu verwalten. Die Datenhaltungsschicht besteht aus der MSSQL-Datenbank und dem Entity Framework, womit es angesteuert wird.

6.4.1. Die Klasse `DataManagementVM`

Die Klasse `DataManagementVM`, die das Data-Management Viewmodel repräsentiert, ist für die Datenverwaltung zwischen Frontend und Datenbank verantwortlich. Die Klasse enthält Methoden zur Verwaltung der

- Probanden,
- Gruppen und
- Szenen.

Um die Datenbank anzusteuern nutzt die Klasse sowohl Language Integrated Queries (kurz LINQ) als auch Lambda-Expressions. LINQ ähnelt stark der in Datenbankensystemen weit verbreitetend Anfragesprache SQL. Lambda-Expressions filtern die Daten mittels einzelner Methoden, was mitunter zu sehr unübersichtlichen Abfragekonstruken führen kann. Beide Techniken sind dabei gleich mächtig. Es werden dennoch beide Techniken eingesetzt, da sich manche Anfragekonstrukte mit der LINQ einfacher formulieren lassen als mit Lambda-Expressions. Auch der umgekehrte Fall trifft in manchen Fällen zu. In den Listings 6.2, 6.3 und 6.4 findet sich die gleiche Datenbankabfrage formuliert in SQL, LINQ und als Lambda-Expression.

```
01    SELECT *
02    FROM Participant p
03    WHERE p.Name LIKE 'Mustermann'
```

Listing 6.2: Beispiel einer SQL-Anfrage auf einer Datenbank.

```
01    FROM p in ctx.Participant
02    WHERE p.Name == Mustermann
03    SELECT p
```

Listing 6.3: Beispiel einer Datenbankabfrage mittels LINQ.
`ctx` repräsentiert dabei den Datenbankkontext.

```
01    ctx.Participant.Where(p => p.Name == Mustermann).SELECT(p => p);
```

Listing 6.4: Beispiel einer Datenbankabfrage mittels Lambda-Expressions.
`ctx` repräsentiert dabei den Datenbankkontext.

Um die Performance der Datenverwaltung zu erhöhen, wird von der Datenverwaltung häufig ausgenutzt, wie Datenbankenänderungen mit LINQ und Lambda-Expressions vom Entity Framework verwaltet werden: Der Datenbankzugriff folgt mittels eines Datenbankkontextes. Danach werden alle Änderungen der lokalen Daten in dem Changetracker des Kontexts verfolgt. Diese Änderungen werden jedoch erst nach einem Aufruf der `SaveChanges()`-Methode an die Datenbank übergeben. Die lokalen Daten arbeiten aber ständig mit den veränderten Daten. Hierdurch können Daten temporär geändert werden, ohne dass dies Einfluss auf die Datenbank hat.

6.4.2. Die Klasse `AoiManagementVM`

Da Datenbanken eigentlich keine hierarchischen Strukuren unterstützen, muss die Hilfsklasse `AoiManagementVM` die Konvertierung und Verwaltung der Baumstruktur der Areas of Interest übernehmen. Dazu ruft die Klasse zum Zeitpunkt ihrer Initialisierung die relevanten Areas of Interest aus der Datenbank ab und liest deren Abhängigkeiten aus. Die dabei entstehenden Daten werden dann in einer durch Listen realisierten Baumstruktur gehalten. Wenn die Elemente der Liste vertauscht werden, dann entspricht dies einem Umsortieren der angezeigten Reihenfolge der AOIs. Da der Prototyp in erster Linie eine hohe Skalierbarkeit der Fixationen und der Aufnahmelänge des Stimulis gewährleisten soll, wird implizit davon ausgegangen, dass der Baum eine Tiefe von zwei Ebenen nicht übersteigt. Das bedeutet, dass eine AOI höchstens Kinder, aber keine Kindeskinder haben kann.

6.5. Das Managed Extensibility Framework (MEF)

Um mit dem Prototyp eine Plug-in-gestützte Architektur zu unterstützen, verwendet dieser das Managed Extensibility Framework, kurz MEF. Das MEF wurde von Microsoft als eine Bibliothek entwickelt, um leichtgewichtige, erweiterbare Anwendungen zu ermöglichen (siehe [19]).

6.5.1. Vorteile eines Plug-in-Systems

Es ist von großem Vorteil, ein Plug-in-System zu verwenden, denn Plug-ins werden erst zur Laufzeit geladen. Das bedeutet, dass die Anwendung, die das Plug-in-System nutzt, nicht neu kompiliert werden muss, wenn ein neues Plug-in eingebunden wird. Das MEF erfüllt genau diese Bedingungen und wird daher in diesem Prototypen verwendet.

6.5.2. Funktionsweise des MEF

Um das MEF zu nutzen, muss die Klasse, die das Plug-in-System enthält, zunächst die Assemblies *System.ComponentModel.Composition* und *System.ComponentModel.Composition.Hosting* referenzieren und einbinden. Danach funktioniert die Nutzung des MEF wie folgt:

1. Es wird eine globale Variable vom Typ `CompositionContainer` angelegt. Diese enthält später alle importierten Plug-ins.
2. Daraufhin muss eine Instanz der Klasse `AggregateCatalog` initialisiert werden. Ein `AggregateCatalog` sammelt die verschiedenen Kataloge, die im nächsten Schritt geladen werden.
3. Der `AggregateCatalog` wird durch `AssemblyCatalog` gefüllt. `AssemblyCatalog` erhalten in ihrem Konstruktor den Verweis auf die Assembly, aus der die importierten Plug-ins stammen sollen.
4. Alternativ zu 3. kann auch mittels eines `DirectoryCatalog` jedes Plug-in, das in einem bestimmten Pfad enthalten ist, geladen werden.

5. Zuletzt wird der in 1. angelegte `CompositionContainer` initialisiert. Im Konstruktor wird dabei der `AggregateCatalog` übergeben. Ab diesem Zeitpunkt können die Plug-ins über den `CompositionContainer` abgerufen werden.

6.5.3. Anforderungen an die Plug-ins, um die Kompatibilität mit dem MEF zu gewährleisten

Damit das MEF die Plug-ins erkennt, müssen diese sich als Plug-in ausweisen. Hierzu müssen diese zwei Voraussetzungen erfüllen:

1. Sie müssen ein Interface implementieren, das über den `AssemblyCatalog` geladen wird.
2. Sie müssen mittels der Direktive *Export* den Typ ihres Interfaces bekannt geben und mit *ExportMetaData* ihren Namen und ihren Plug-in-Typ angeben.

Ein Beispiel für den Header eines Plug-ins aus dem Prototyp ist in Listing 6.5 gegeben.

```
[...]
namespace Plugins.Analyzation
{
   /// <summary>
   /// Interaction logic for ParticipantFilterUC.xaml
   /// </summary>
   [Export(typeof(IAnalyzationExtension))]
   [ExportMetadata("AnalyzationUC", "ParticipantFilterUC")]
   public partial class ParticipantFilterUC : UserControl, IAnalyzationExtension
   {
      DataManagementVM _DataManagement;
      ICollection<Participant> participants = null;
[...]
```

Listing 6.5: Beispiel des Headers des ParticipantFilter-Plug-ins. Es werden sowohl die Export- und ExportMetaData-Direktiven dargestellt, als auch, welches Interface in diesem Plug-in genutzt wird.

Das benötigte Interface wird im folgenden Unterabschnitt vorgestellt.

6.5.4. Interface für Analysekomponenten

Alle Plug-ins, die im Analyseschritt verwendet werden sollen, müssen die Methoden und Properties aus Tabelle 6.2 implementieren. Dies gewährleistet, dass jedes Plug-in Schnittstellen zur Verfügung stellt, damit andere Komponenten diese nutzen können, ohne deren Implementierung zu kennen.

6.5.5. Interface für Visualisierungskomponenten

Alle Visualisierungs-Plug-ins müssen ebenfalls ein Interface implementieren. Dieses ähnelt zunächst dem Interface der Analyse, doch werden hierbei weitere Daten an das Plug-in übergeben, um ein korrektes Rendering der Daten zu ermöglichen. Da die Properties Description und DataManagement identisch mit denen aus Tabelle 6.3 sind und die Methode

Name	Typ / Rückgabewert	Funktion
Description	String (Property)	Beschreibung des Plug-ins. Wird bei der Auswahl der Plug-in-Komponente genutzt.
ShowAnalyzationUC()	UserControl	Gibt die initialisierte Plug-in-Klasse zurück, damit diese verwendet werden kann.
DataManagement	DataManagementVM (Property)	Hiermit kann dem Plug-in ein Datensatz übergeben und zur Weitergabe an andere Plug-ins wieder entnommen werden.
IsValid	bool (Property)	Gibt bekannt, ob die Verarbeitung der gegebenen Daten möglich bzw. erfolgreich war.

Tabelle 6.2.: Tabellarische Beschreibung des Interfaces für Analyse-Plug-ins. Die erste Spalte enthält den Namen der Methode bzw. Property. Die zweite Spalte gibt an, um welchen Datentyp es sich handelt und ob es eine Property oder eine Methode mit Rückgabewert ist. Die letzte Spalte enthält eine Funktionsbeschreibung.

`ShowVisualizationUC()` den gleichen Zweck erfüllt wie `ShowAnalyzationUC()`, werden diese nicht erneut aufgelistet.

Name	Typ / Rückgabewert	Funktion
AoiManagement	AoiManagementVM (Property)	Wird zur Verwaltung der AOIs genutzt.
SceneId	int?	Gibt die id der Szene an, die von dem Renderer gerendert werden soll. Falls keine Szenen benötigt werden, kann dieser Wert auf *null* gesetzt werden.
TimelineStart	long (Property, nur schreiben)	Hiermit kann der Startzeitpunkt des Renderings als Timestamp übergeben werden.
TimelineStop	long (Property, nur schreiben)	Hiermit kann der Endzeitpunkt des Renderings als Timestamp übergeben werden.

Tabelle 6.3.: Tabellarische Beschreibung des Interfaces für Rendering-Plug-ins. Die erste Spalte enthält den Namen der Methode bzw. Property. Die zweite Spalte gibt an, um welchen Datentyp es sich handelt und ob es eine Property oder eine Methode mit Rückgabewert ist. Die letzte Spalte enhält eine Funktionsbeschreibung.

6.6. Datenverwaltung zwischen den Analysekomponenten

Um die Daten auf der Analyseebene zu verwalten, werden die in Unterkapitel 6.4.1 beschriebenen Eigenschaften des ChangeTrackers genutzt. Die Änderungen, welche von den Plug-ins an den einzelnen Datensätzen vorgenommen werden, werden daher nicht von der Datenbank übernommen. Sie können jedoch auf lokaler Ebene genutzt werden. Die `AnalyzationPipeline`-Klasse erstellt dazu zunächst eine `DataManagementVM` mit den Daten aus der Datenbank. Diese werden an das erste Plug-in übergeben. Für jedes weitere Plug-in bzw. zur Weitergabe der Daten an die Visualisierung wird die `DataManagementVM` aus dem letzten Plug-in wieder entnommen und weitergegeben. Zur Verdeutlichung des Konzepts ist der Ablauf in Abbildung 6.4 noch einmal dargestellt. Da der Prototyp das Entfernen von Plug-ins unterstützt, müssen

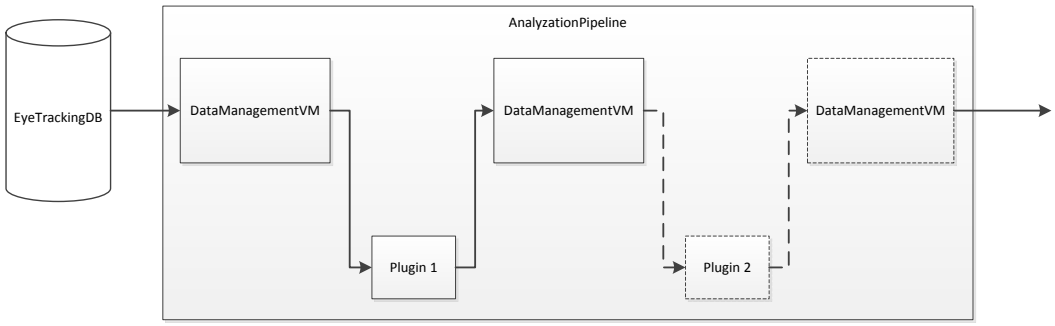

Abbildung 6.4.: Darstellung des Datenflusses der Analyse. Zunächst werden die Daten von einer DataManagementVM-Instanz aus der Datenbank geladen. Diese Instanz wird dann an das erste Plug-in übergeben und die Daten werden manipuliert. Danach werden die Daten aus dem Plug-in entnommen und an das nächste übergeben, bis alle Plug-ins abgearbeitet wurden. Dann wird die DataManagementVM-Instanz weitergegeben.

einige potenzielle Fehlerquellen beachtet werden:

Anzahl der Plug-ins: Es sollte immer mindestens ein Plug-in vorhanden sein. Um diese Beschränkung zu garantieren, wird beim Erstellen der Analysepipeline bereits ein Plug-in erstellt. Jedes Plug-in kann ausgetauscht werden. Plug-ins können gelöscht werden, sofern danach noch mindestens ein Plug-in in der Pipeline verbleibt.

Datenverwaltung beim Löschen von Plug-ins: Wenn ein Plug-in gelöscht wird, müssen die darauf folgenden Plug-ins entsprechend reagieren. Deshalb wird beim Entfernen eines Plug-ins ein Event ausgelöst, das die erneute Auswertung der einzelnen Plug-ins erzwingt. Zudem wählt der Prototyp automatisch das erste neu evaluierte Plug-in aus. Falls dieses nicht verfügbar ist (beispielsweise, wenn das gelöschte Plug-in das letzte der Pipeline war), wird das dem gelöschten vorhergehende Plug-in ausgewählt.

6.7. Allgemeine grafische Oberfläche der Analyse

Der folgende Abschnitt befasst sich mit der Implementierung der grafischen Oberfläche des Analyseteils des Prototyps. Zunächst wird erläutert, wie die einzelnen Bereiche der Analyseoberfläche verbunden sind. Anschließend wird die ETA-Pipeline vorgestellt. Darauf folgt der Anzeigecontainer für die Analyse-Plug-ins.

6.7.1. Aufbau der Analyseoberfläche

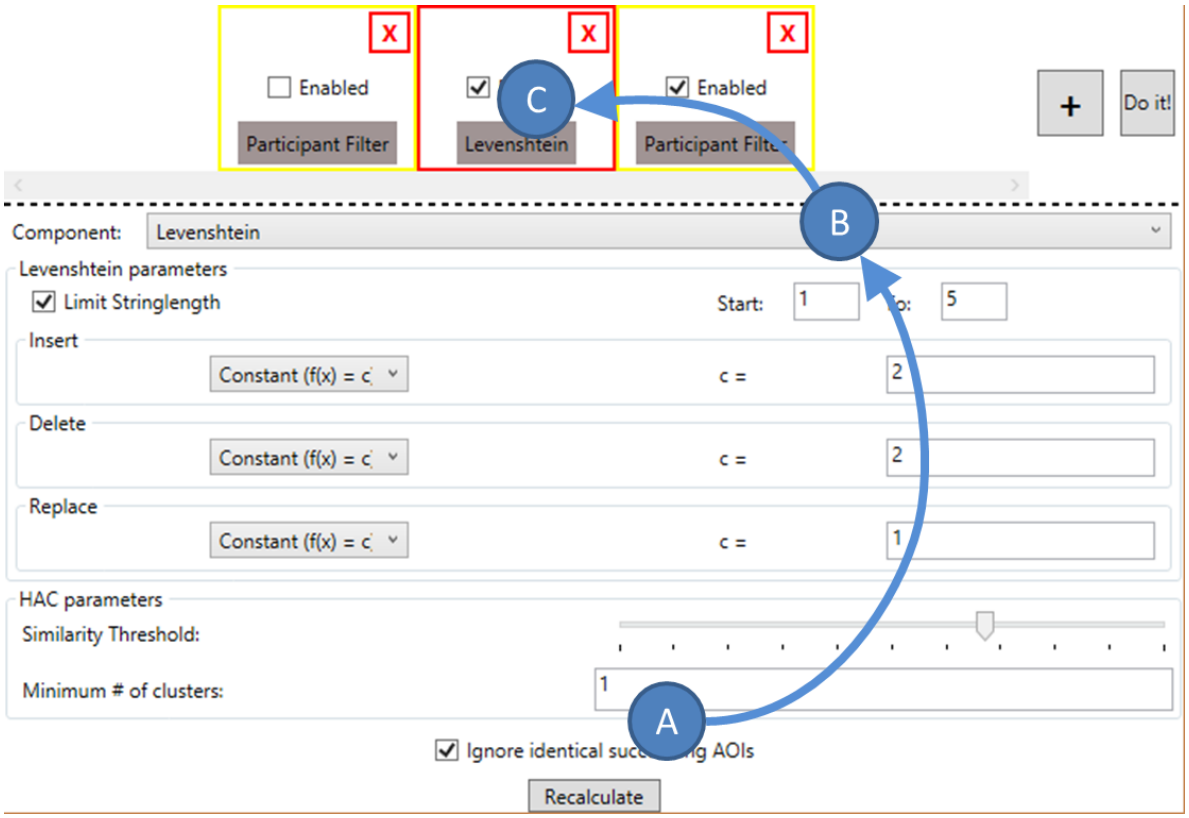

Abbildung 6.5.: Darstellung der Abhängigkeiten der Komponenten der Analyseoberfläche. Die Plug-in-Konfiguration (A) ist an die Combobox (B) gebunden. Diese ist Teil der gesamten Plug-in-Anzeige, welche an ein einzelnes Element (C) der ETA-Pipeline gebunden ist.

Um eine bessere Kontrolle des Datenflusses zu erreichen und die Datenverwaltung zu vereinfachen, wurden die Komponenten der Analyseoberfläche aneinander gebunden. Abbildung 6.5 veranschaulicht die genutzten Bindings. Die Plug-in-Konfiguration ist an die Combobox gebunden, mit der das Plug-in ausgewählt wird. Die Combobox ist wiederum Teil der gesamten Plug-in-Anzeige und ist an ein einzelnes Element der ETA-Pipeline, die nachfolgend näher erläutert wird, gebunden.

6.7.2. Die ETA-Pipeline

Ein Screenshot der ETA-Pipeline ist in Abbildung 6.6 gegeben. Die Implementierung ist dabei in zwei Bereiche unterteilt:

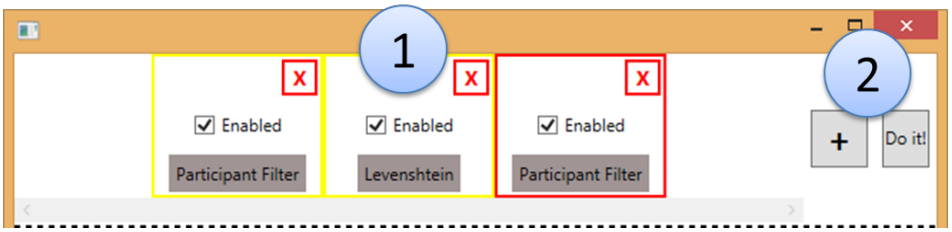

Abbildung 6.6.: Screenshot der Analysepipeline des Prototyps. Abschnitt (1) zeigt dabei die momentan hinzugefügten Plug-ins. Die Analyseverwaltung ist bei (2) zu sehen.

(1) Plug-in-Übersicht: Hier werden die momentan hinzugefügten Plug-ins angezeigt. Da die Elemente selbst diverse Interaktionsmöglichkeiten bieten, werden diese nachfolgend näher beschrieben.

(2) Analyseverwaltung: Dieser Teil bietet Funktionen zur Verwaltung der Analysepipeline. Die unterstützten Tätigkeiten sind

- neue Plug-ins hinzuzufügen (+) und
- die Analyse abzuschließen und die Visualisierung aufzurufen (Do it!).

Interaktionsmöglichkeiten der Plug-in-Elemente

Die einzelnen Plug-in-Elemente haben mehrere Eigenschaften. So ist es möglich, die Plug-ins mit dem rechts oben angebrachten Löschknopf zu entfernen, sofern das Plug-in nicht das letzte ist. Andernfalls wird der Nutzer in der Konsole darüber informiert, dass es nicht möglich ist, das letzte Plug-in zu entfernen. Zudem ist es möglich, die Plug-ins ein- und auszuschalten. Wenn ein Plug-in abgeschaltet ist, leitet es seine Eingangsdaten direkt an den Ausgang weiter und manipuliert die Daten nicht. Das zentriert unten angebrachte Label zeigt den Namen des Plug-ins, welches das Element repräsentiert. Außerdem wird das momentan ausgewählte Plug-in-Element durch einen roten Rahmen hervorgehoben.

6.7.3. Anzeigecontainer für die Analyse-Plug-ins

Der Analysecontainer ist, wie in Abbildung 6.7 dargestellt, in zwei Gebiete unterteilt:

(1) Plug-in-Auswahl: Die Plug-ins werden wie in Abschnitt 6.5 beschrieben geladen. Dann werden bei den Analyse-Plug-ins die Namen ausgelesen und in einer Combobox angezeigt. Das ausgewählte Plug-in wird im Plug-in-Container angezeigt.

(2) Plug-in-Container: Hier wird das ausgewählte Plug-in angezeigt. Weder Aussehen noch die genaue Funktionsweise des Plug-ins sind dieser Komponente bekannt. Sie dient nur als Behälter für die Anzeige.

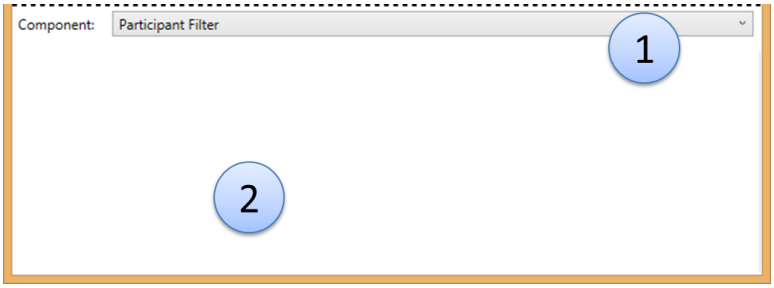

Abbildung 6.7.: Screenshot des Containers für Plug-ins. In der Combobox (1) werden alle verfügbaren Plug-ins aufgelistet. Deren Oberfläche wird dann in dem Bereich (2) dargestellt.

Aufgrund der in Abschnitt 6.7.1 beschriebenen Bindungen der Analyseoberfläche bleiben die Änderungen in einem Plug-in bis zum Programmende, oder bis es gelöscht wird, erhalten.

6.8. Analysekomponenten

Zur Analyse werden im Prototyp drei Analyse-Plug-ins, der Gruppen- und Probandenfilter, das Levenshtein-basierte Clustering sowie die Gruppenvereinfachung, zur Verfügung gestellt. Deren Funktionsweise und Interaktionsmöglichkeiten werden im Folgenden näher erläutert.

6.8.1. Der Gruppen- und Probandenfilter

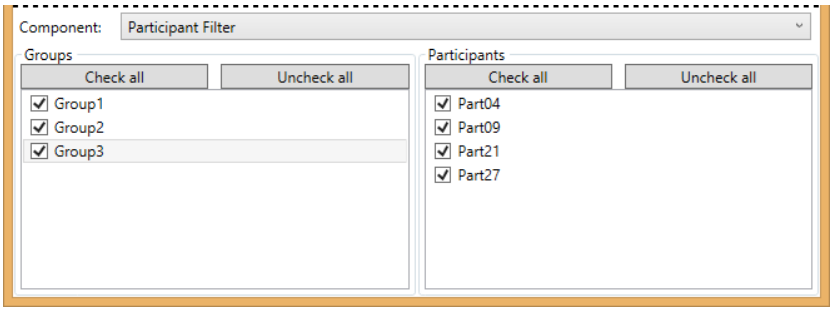

Abbildung 6.8.: Screenshot des Gruppen- und Probandenfilters. Der an das Plug-in übergebene Datensatz enthält drei Gruppen von Probanden: Group1, Group2 und Group3. Alle Gruppen sind angewählt und die dritte Gruppe ist selektiert. Rechts neben den Gruppen befinden sich die Probanden, die zu der ausgewählten Gruppe gehören. Zu sehen sind vier Probanden. Es sind alle Probanden ausgewählt.

Der Gruppen- und Probandenfilter wird in Abbildung 6.8 dargestellt und ist in zwei Abschnitte unterteilt. Auf der linken Seite werden alle Gruppen des an das Plug-in übergebenen Datensatzes in einer ListBox angezeigt, die in dem aktuellen Datensatz existieren. Die einzelnen Gruppen werden dabei durch Checkboxen repräsentiert. Dies ermöglicht eine visuelle

Repräsentation, ob die Gruppe derzeit aktiv verwendet wird oder nicht. Jede Gruppe kann von dem Benutzer an- und abgewählt werden und es ist möglich, eine Gruppe zu selektieren.

Wenn eine Gruppe abgewählt wird, dann wird das `inUse`-Attribut der Gruppe auf *false* gesetzt. Alle Plug-ins können aus der `DataManagementVM`-Klasse die Methode `getGroupIds(bool useFilteredData)` ausführen. Diese gibt, abhängig davon, ob der Parameter `useFilteredData` *true* oder *false* ist, entweder nur die IDs der genutzten oder aller Gruppen zurück. Das Anwählen einer Gruppe setzt respektive das `inUse`-Attribut auf *true*.

Wenn eine Gruppe selektiert wird, werden auf der rechten Seite alle der Gruppe zugehörigen Probanden angezeigt. Diese können ebenfalls an- und abgewählt werden, wobei die Mechanik analog zur Gruppenauswahl ist.

6.8.2. Der HAC mit Levenshtein-Distanz

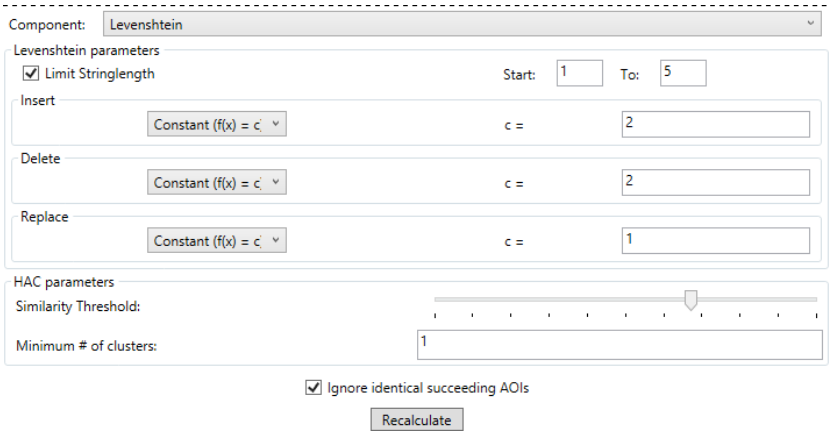

Abbildung 6.9.: Screenshot des HAC-Plug-ins. Es kann die Länge der zu vergleichenden Strings ausgewählt werden. Darunter können die Parameter für die Kostenfunktionen für die Einfüge-, Lösch- und Ersetzungsfunktion gesetzt werden. Darunter kann die Ähnlichkeit und die minimale Clusteranzahl gewählt werden. Zuletzt kann ausgewählt werden, ob die Vergleichsstrings gekürzt werden sollen.

Das Plug-in zum hierarchischen Clustering ermöglicht es, die verfügbaren Probanden zu gruppieren. Das Plug-in setzt voraus, dass sich alle Participants in einer Gruppe befinden, es also eigentlich keine Gruppen gibt. Sollte dies nicht der Fall sein, so gibt das Plug-in immer die ursprünglichen Daten zurück und führt kein Clustering durch. Da das Clustering immer auf den ursprünglichen Daten angewendet wird ist es möglich, die Parameter des Plug-ins zu verändern und die Daten erneut zu clustern. Die zuvor erstellten Gruppen werden dann verworfen. Das hierarchische Clustering-Plug-in nutzt als Distanzmaß die Levenshtein-Distanz und als Clusteringverfahren das Single-Linkage.

Zur Konfiguration der Levenshtein-Distanz stehen verschiedene Parameter, wie die optionale Einschränkung der Stringlänge, die Kostenfunktionen der Levenshtein-Distanz, die Toleranz der Clusterähnlichkeit, die Mindestanzahl an Clustern und das optionale Ignorieren

identischer aufeinanderfolgender AOIs, zur Verfügung, die im Folgenden näher beschrieben werden:

Einschränkung der Stringlänge: Anstatt den gesamten String, und damit den gesamten Transitionspfad der Eye-Tracking-Aufnahme, zu vergleichen, ist es möglich, den String auf einen definierbaren Intervall zu beschränken. Dies ermöglicht, dass beispielsweise nur der Anfang aller Aufnahmen verglichen wird. Ein weiterer Vorteil ist, dass sehr lange Aufnahmen, bzw. Aufnahmen mit häufigen Wechseln der AOI, abgeschnitten werden, um mit den anderen Aufnahmen vergleichbarer zu werden.

Kostenfunktionen der Levenshtein-Distanz: Wie in Kapitel 5.5.4 beschrieben, unterstützt das Modell zur Berechnung der Levenshtein-Distanz verschiedene mathematische Modelle. Die unterstützten mathematischen Modelle und ihre Parameter sind im Folgenden beschrieben. Es können dabei für die Kostenfunktionen in der Regel die Parameter a für die Steigung, b für die Verschiebung auf der x-Achse und c für die Verschiebung auf der y-Achse gesetzt werden. Es stehen folgende mathematische Modelle zur Verfügung:

Konstanter Wert: $f(x) = c$ Es kann ein konstanter Wert angegeben werden. Der einzige Parameter ist hierbei c.

Lineare Funktion: $f(x) = a \cdot (x - b) + c$

Exponentialfunktion: $f(x) = a \cdot e^{(x-b)} + c$

Invertierte Exponentialfunktion: $f(x) = (a \cdot e^{(x-b)} + c)^{-1}$

Es sollte beachtet werden, dass das Ergebnis der Funktion immer größer als 0 ist, da ansonsten Fehler in der Levenshtein-Funktion auftreten und in Folge dessen die berechnete Distanz fehlerhaft sein kann.

Nachdem die Levenshtein-Distanz konfiguriert wurde, ist es noch notwendig, einige Parameter für den HAC festzulegen:

Toleranz der Ähnlichkeit: Mittels eines Sliders kann festgelegt werden, wie hoch die Toleranz für die Mindestähnlichkeit für eine Vereinigung der Cluster sein muss. Ein Toleranzwert von 1,0 weist alle Probanden derselben Gruppe zu. Bei einem Wert von 0,0 werden nur vollkommen identische Probanden der selben Gruppe zugewiesen. Wenn sich die Cluster nicht mehr ähnlich genug sind, bricht der HAC ab.

Mindestanzahl an Clustern: Eine Mindestanzahl an Clustern kann mittels dieses Parameters garantiert werden.

Ignorieren identischer aufeinanderfolgender AOIs: Es ist möglich, die Vergleichsstrings bezüglich direkt aufeinanderfolgender identischer Zeichen zu kürzen, sodass beispielsweise aus dem String „AAABA“ „ABA“ wird. Dies kann nützlich sein, wenn der Eye-Tracker die Augen des Probanden häufig für kurze Zeit verliert und dadurch beim Matching Transitionspfadelemente mit einer identischen AOI entstehen.

Wenn die Parameter gesetzt sind, kann man mittels des „Recalculate“-Buttons das HAC auf dem ursprünglichen Datensatz ausführen lassen.

Abhängig von der gewählten Funktion wird die Methode `calculateDistance(String A, String B, FunctionType insert, FunctionType delete, FunctionType replace)` aufgerufen, die eine Gleitkommazahl als Rückgabewert hat. Ein Auszug aus der Hilfsklasse, die die Funktionsdelegates generiert, ist in Listing 6.6 gegeben. Zur Verdeutlichung der Anwendung der Delegates ist ein Beispiel für den Funktionsaufruf, bei dem die Einfüge- und Löschkosten konstant und die Ersetzungskosten linear sind, in Listing 6.7 gegeben.

```
[...]
public enum FunctionTypes { Constant = 0, Linear = 1, Exponential = 2, InvExponential =
    3 }
private FunctionTypes functionType = FunctionTypes.Constant;
[...]
public FunctionDelegates.FunctionType Function
{
      get {
            switch (functionType) {
                  case FunctionTypes.Constant:
                        return (x) => CValue;
                  case FunctionTypes.Linear:
                        return (x) => (AValue*(x - BValue) + CValue);
                  case FunctionTypes.Exponential:
                        return (x) => (AValue * Math.Exp(x - BValue) + CValue);
                  case FunctionTypes.InvExponential:
                        return (x) => Math.Pow((AValue * Math.Exp(x - BValue) +
                            CValue), -1);
                  default:
                        return (x) => -1; } }
}
[...]
```

Listing 6.6: Auszug aus der Hilfsklasse, welche die Funktionsdelegates für die Levenshtein-Distanz berechnet. Es werden die verschiedenen Funktionsklassen in einem Enumerator aufgelistet. Abhängig von dem in der GUI gewählten Kostenmodell wird das entsprechende Delegate erzeugt.

```
[...]
String A = "‘ABBAB";’
String B = "‘ABAAB"’;
var distance = levenshteinDistance.calculateDistance(A, B, (i) => 2, (d) => 2, (r) => r*5);
[...]
```

Listing 6.7: Beispiel eines Aufrufs der `calculateDistance`-Methode zur Berechnung der Levenshtein-Distanz zweier Strings. Die Einfüge- und Löschkosten sind konstant und die Ersetzungskosten linear. Die Kostenfunktionen werden mit Delegates übergeben.

Um dem Analysierenden eine Rückmeldung über die Auswirkungen des Plug-ins auf den Datensatz zu geben, werden auf einer separaten Konsolenausgabe die entstandenen Cluster mit deren Probanden sowie deren Ähnlichkeit ausgegeben. Eine Darstellung der Konsolenausgabe ist in Abbildung 6.10 gegeben. Bereich A zeigt dabei die Distanzen der einzelnen Cluster und Bereich B zeigt die entstandenen Cluster mit den enthaltenen Probanden. Die Cluster werden dabei nach dem in Abbildung 6.11 gegebenen Schema berechnet.

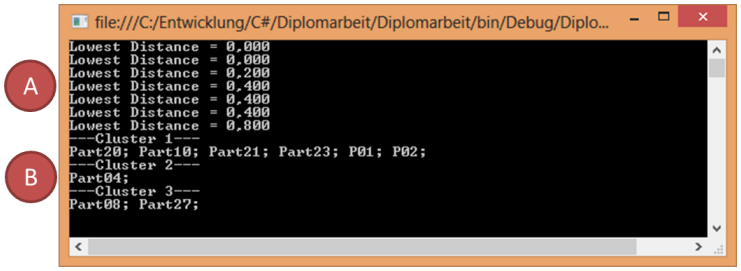

Abbildung 6.10.: Screenshot der HAC-Konsolenausgabe. In Bereich A wird angezeigt, wie ähnlich sich die einzelnen Cluster vor der Vereinigung waren. Bereich B zeigt, welche Cluster erzeugt wurden und welche Probanden in diesen enthalten sind.

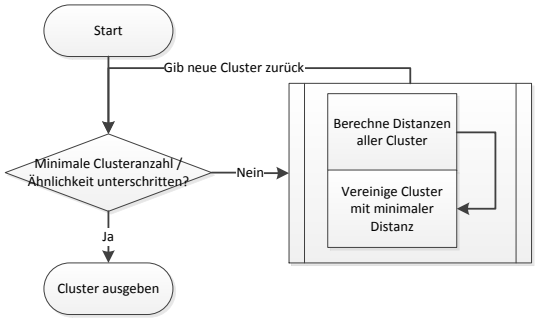

Abbildung 6.11.: Schematische Darstellung des Clusterings. In jedem Durchlauf wird überprüft, ob die Abbruchbedingungen erfüllt sind. Wenn dies nicht der Fall ist, so werden die Distanzen der Cluster berechnet und die Cluster mit der geringsten Distanz vereinigt. Ansonsten bricht der Algorithmus ab und gibt die erstellten Cluster zurück.

6.8.3. Die Gruppenvereinfachung

Das Plug-in zur Gruppenvereinfachung ermöglicht es, einen Repräsentanten für eine Gruppe zu berechnen. Hierbei wird der dritte Ansatz aus Abschnitt 5.5.11 genutzt. Das Plug-in weist jedem Probanden einen Gütewert zu, der die Eignung des Probanden als Centroid angibt. Das Plug-in ermöglicht die Auswahl des Verfahrens, das für die Berechnung des Gütewerts genutzt wird. Als Grundlage für den Gütewert wird von dem Plug-in die Levenshtein-Distanz jedes Probanden zu allen anderen Probanden berechnet. Die Levenshtein-Distanz arbeitet dabei mit konstanten Kosten. Diese betragen für das Ersetzen „1“ und für das Einfügen und Löschen von Zeichen „2“. In Tabelle 6.4 werden die verschiedenen unterstützten Methoden zur Berechnung der Güte eines einzelnen Probanden und deren Eigenschaften beschrieben.

6.9. Aufbau der Visualisierungsoberfläche

Die Visualisierungsoberfläche ist aus verschiedenen Komponenten aufgebaut, die in diesem Abschnitt näher beschrieben werden. Zuerst wird auf die Menüführung eingegangen. Darauf-

Verfahrensname	Berechnungsvorschrift	Eigenschaften
Einfache Summe der Kosten	Güte = $\sum_{AlleProbanden} LD(A,B)$	Ungewichtete Summe über alle Probanden
Quadratische Summe der Kosten	Güte = $\sum_{AlleProbanden} LD(A,B)^2$	Ungewichtete Summe über alle Probanden
Wurzelsumme der Kosten	Güte = $\sum_{AlleProbanden} \sqrt{LD(A,B)}$	Ungewichtete Summe über alle Probanden

Tabelle 6.4.: Tabellarische Beschreibung unterstützten Methoden zur Güteberechnung der Centroide. Links stehen die Namen der jeweiligen Verfahren. In der Mitte ist die Berechnungsvorschrift für das Verfahren angegeben. Dabei steht LD für die Levenshtein-Distanz-Funktion. A ist der String des Probanden, dessen Güte berechnet wird und B ist der String des Vergleichsprobanden.

hin wird auf das AOI-Management eingegangen. Anschließend wird der Szenenguide näher erläutert. Danach wird die PSP-Anzeige vorgestellt.

6.9.1. Die Menüführung

Einige zusätzliche Funktionen wurden zwecks der Reduzierung der kognitiven Last im Menü untergebracht. Dazu gehören Funktionen wie zoomen, speichern und die zusätzlichen Optionen für die PSPs und die AOIs.

6.9.2. Der Szenenguide

Der Szenenguide listet alle Szenen in chronologischer Reihenfolge auf. Die erste Szene ist dabei ganz unten, die letzte ganz oben. Zudem kann für jede Szene optional ein Vorschaubild angezeigt werden, um ihren Inhalt anzudeuten. Die Szenengröße ist dabei abhängig von der Länge der Aufnahme. Technisch gesehen enthält jede Szene einen unabhängigen PSP.

6.9.3. Die PSP-Anzeige

Hier werden die einzelnen PSPs angezeigt. Probanden, die zu einer Gruppe gehören, werden in ähnlichen Farben dargestellt. Dabei werden nie zwei Probanden der selben Gruppe in der gleichen Farbe dargestellt. Die Gruppenfarben liegen eindeutig unterscheidbar auseinander.

Die Wahl der Farbe für die Gruppen

Um die Gruppenfarben eindeutig unterscheiden zu können wurde die Hilfsklasse `ProceduralColorGenerator` angelegt. Diese stellt die Methode `CalculateMaxDifferenceColorBrush(double startOffset, int index)` zur Verfügung, die einen Offset für den Farbraum und eine Indexvariable erwartet. Die Indexvariable soll dabei in diesem Fall den Gruppenindex angeben, damit beispielsweise die Farben der Gruppen „2“ und „3“ sich so stark wie möglich unterscheiden. Der Algorithmus arbeitet dazu im HSV-Farbraum und nutzt dabei als Distanz zwischen den Hue-Werten der Farben den goldenen Schnitt und ist angelehnt an den

Algorithmus aus [37] unter dem Kapiel „4. Selecting from a gradient“. Der Unterschied besteht zum einen darin, dass der Gradient der gesamte Farbraum ist und dass der Algorithmus den HSV-Raum nutzt, um die Werte zu errechnen. Außerdem gibt die Methode aus praktischen Gründen einen `SolidColorBrush` anstatt einer `Color` zurück.

6.10. Interaktionsmöglichkeiten

Abschließend werden in diesem Abschnitt die verschiedenen Interaktionsmöglichkeiten mit der Visualisierungsoberfläche erläutert. Dazu wird zunächst auf das Brushing eingegangen. Danach wird die Funktionsweise der Zoom-Funktion erläutert. Anschließend werden die verschiedenen Konfigurationsmöglichkeiten der PSPs vorgestellt. Daraufhin geht es um die Konfiguration des AOI-Managements. Dann wird die Aus- und Einblendefunktion des AOI-Managements vorgestellt. Abschließend wird auf die Möglichkeit, Szenen ein- und auszublenden, eingegangen.

6.10.1. Brushing

Es ist möglich, einzelne Transitionspfade mittels Brushing hervorzuheben, indem man mit der Maus über den entsprechenden Pfad fährt. Hierdurch wird zum einen die Opazität aller anderen Transitionspfade gesenkt und zum anderen die Dicke des Transitionspfades erhöht. Zusätzlich wird nach kurzer Zeit der Name des Probanden als Tooltip angezeigt. Ein Screenshot der Visualisierung des Prototyps mit gebrushten Daten ist in Abbildung 6.12 zu sehen.

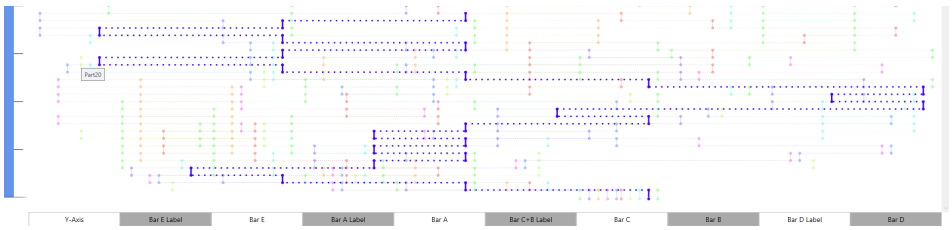

Abbildung 6.12.: Screenshot der Visualiserung mit gebrushten Daten. Es werden zehn Datensätze angezeigt, wovon einer durch Brushing hervorgehoben wird. Der Tooltip zeigt den Namen des Probanden.

6.10.2. Zooming

Wenn ein Transitionspfad sehr lang ist, entstand bei den ursprünglichen PSPs das Problem, dass entweder der ganze PSP angezeigt wurde, dann aber kaum Details erkennbar waren, oder die PSPs gezoomt waren, wodurch die Gesamtübersicht verloren ging. Der Prototyp stellt eine Zoomfunktion zur Verfügung. Diese kann entweder über das Menü aufgerufen werden oder mit Hilfe von Tastaturshortcuts. Die PSPs müssen fokussiert sein, damit die Shortcuts funktionieren. Die zur Verfügung stehenden Zoomfunktionen werden in Tabelle 6.5 aufgelistet.

Funktion	Shortcut
Zoom (1x)	[Strg] + [+]
Zoom (10x)	[Strg] + [Umschalt] + [+]
Zoom (1x)	[Strg] + [-]
Zoom (10x)	[Strg] + [Umschalt] + [-]

Tabelle 6.5.: Tabellarische Beschreibung der Zoomfunktionen für die PSPs. Es kann sowohl mit einem einfachen, als auch mit einem zehnfachen Zoomfaktor gearbeitet werden.

6.10.3. Konfiguration der PSPs

Es ist möglich, über das Menü die PSPs zu konfigurieren. So besteht beispielsweise die Möglichkeit, zwischen einer zeitabhängigen und einer zeitunabhängigen Anzeige zu wechseln. Zudem kann man bestimmte Animationen ein- und ausschalten.

Zeitabhängige und zeitunabhängige Anzeige

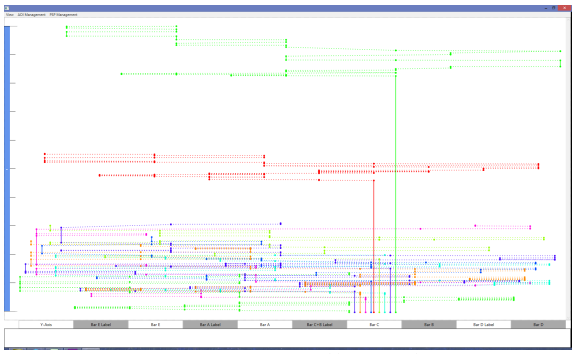

Abbildung 6.13.: Darstellung der PSPs mit zeitabhängigen Daten.

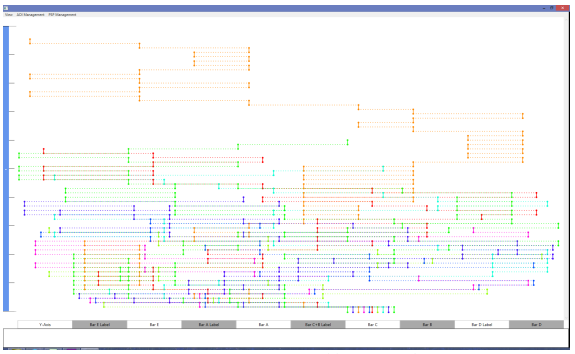

Abbildung 6.14.: Darstellung der PSPs mit zeitunabhängigen Daten.

Wie bereits im Konzept beschrieben, ist es im Prototyp möglich, die Transitionspfade zwecks einer besseren Vergleichbarkeit der Pfade, und damit auch der kognitiven Abläufe, sowohl zeitabhängig, als auch zeitunabhängig darzustellen. Technisch werden hierbei die Punkte der PSPs bezüglich der Zeitachse äquidistant zueinander gerendert. Bei der zeitunabhängigen Anzeige der Daten können die Transitionspfade sowohl gekürzt als auch ungekürzt angezeigt werden. Abbildung 6.13 zeigt einen Datensatz, bei dem die Daten zeitabhängig dargestellt werden. In Abbildung 6.14 wird der gleiche Datensatz zeitunabhängig dargestellt. Ansonsten bestehen keine weiteren Unterschiede.

Animation

Da das Brushing bei PSPs mit vielen Transitionspfaden eine starke Veränderung der visuellen Informationen zur Folge hat, ist es möglich, diese Änderungen animieren zu lassen. Diese Funktion ist jedoch standardmäßig abgeschaltet, da diese viel Rechenleistung benötigt. Wenn

die Animationen eingeschaltet sind, werden die Veränderung der Dicke und der Opazität animiert.

6.10.4. Szenenein- und ausblendung

Es ist ebenfalls möglich, die Szenen aus- und einzublenden. Dies ermöglicht, zeitlich nicht direkt aufeinanderfolgende Szenen trotzdem zu vergleichen, um eventuelle Ähnlichkeiten zu erkennen. Wenn eine Szene ausgeblendet ist, wird ihr Inhalt nicht gerendert und die Szene wird rot dargestellt. Ein denkbares Szenario hierfür wäre, ob sich das Blickverhalten eines Autofahrers beim Setzen des Blinkers in der Stadt mit dem des Blinkersetzens auf der Autobahn überdeckt. Die Größe der ausgeblendeten Szenen entspricht 70% der kleinsten angezeigten AOI.

Abbildung 6.15 zeigt oben den Szenenguide mit ausgeklappten Szenen. Unten werden die gleichen Szenen angezeigt, doch sind die zweite und dritte Szene eingeklappt.

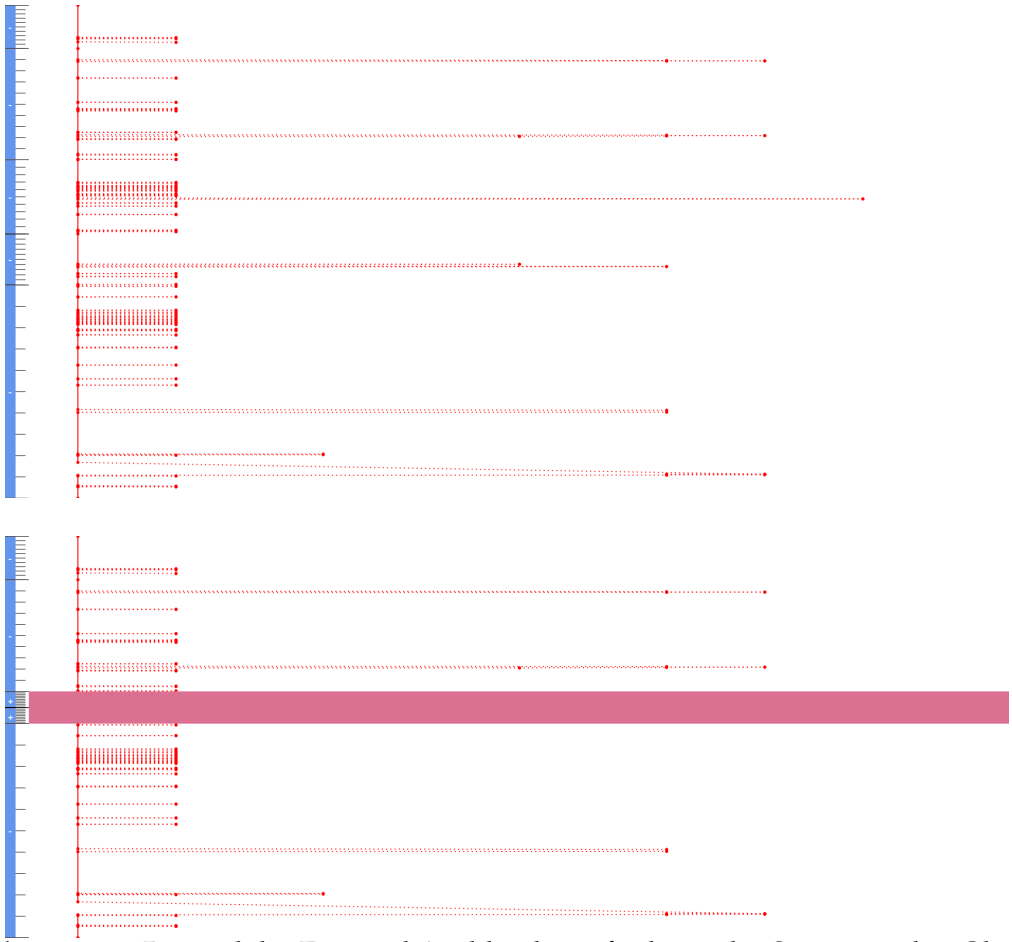

Abbildung 6.15.: Beispiel der Ein- und Ausblendungsfunktion des Szenenguides. Oben wird der Szenenguide mit fünf ausgeklappten Szenen angezeigt. Unten werden die gleichen Szenen angezeigt, wobei die zweite und dritte Szene eingeklappt sind.

7. Demonstration und Evaluation

Dieses Kapitel demonstriert die Analyse von zwei Eye-Tracking-Datensätzen mit unterschiedlichen Szenarien mit Hilfe des zuvor vorgestellten Prototyps. Das erste Szenario ist die Analyse einer Eye-Tracking-Studie aus der Forschung (Abschnitt 7.1). Bei dem zweiten Szenario handelt es sich um eine Aufnahme eines Eye-Trackers in einem Auto der Daimler AG, bei der das Blickverhalten eines Autofahrers während einer Autofahrt in der Stadt und auf der Autobahn aufgenommen wurde (Abschnitt 7.2).

In den Abschnitten 7.1 und 7.2 wird jeweils zunächst das Szenario vorgestellt. Daraufhin werden die Daten der Studie bzw. der Aufnahme, wie etwa die Anzahl der Probanden oder die Aufnahmelänge, erläutert. Anschließend wird auf die Besonderheiten bzw. Details der jeweiligen Datensätze eingegangen. Danach wird die Durchführung der Analyse beschrieben. Zuletzt werden die Ergebnisse der jeweiligen Untersuchung vorgestellt.

7.1. Analyse einer Informationsvisualisierungsstudie

Das erste Szenario nutzt die Studiendaten von „Visual Elements III“ des Instituts für Visualisierung und interaktive Systeme. Von der Studie wurde exemplarisch ein Stimulus zur Analyse mit dem Prototypen ausgewählt. Dabei sollen mit Hilfe des Tools die Lösungsstrategien für das Auslesen und Vergleichen eines Balkendiagramms untersucht werden. Im Folgenden wird zunächst das Szenario erläutert. Daraufhin wird auf die Daten der Studie, wie etwa die Anzahl der Probanden, eingegangen. Anschließend wird der für die Analyse gewählte Stimulus und die dazu gestellte Aufgabe erläutert. Anschließend wird auf die Durchführung der Analyse eingegangen. Zuletzt werden die Ergebnisse der Untersuchung vorgestellt.

7.1.1. Analyse einer Informationsvisualisierungsstudie

An der Studie Visual Elements III, kurz VE3, haben zehn Personen teilgenommen. Von diesen Teilnehmern waren vier weiblich und sechs männlich. Das Durchschnittsalter ist 29,2 Jahre und zwei der Teilnehmer haben angegeben, dass sie Vorkenntnisse im Gebiet der Visualisierung haben. Von den zehn Teilnehmern waren neun zwischen 23 und 30 Jahren alt. Diese Teilnehmer waren ausschließlich Studenten. Die Studie befasst sich mit der Analyse von kognitiven Prozessen bei Visualisierungen. Um diese Prozesse zu analysieren, wurden verschiedene Aufgaben gestellt. Die Aufgabenarten sind

1. das Lösen einfacher Rechenaufgaben,
2. das Bearbeiten von Fragestellungen zu Mengendiagrammen,
3. das Lösen von Rätseln der Art „Sudoku“,

4. das Ablesen von Werten aus Balkendiagrammen,
5. das Ablesen von Werten aus verschiedenartigen Diagrammen und
6. das Vergleichen von Bildern.

7.1.2. Daten der Studie

Die Studie wurde mit einem Tobii XL60 aufgenommen. Dieser hat eine Bildschirmdiagonale von 24 Zoll und eine Bildschirmauflösung von 1920x1200 Pixeln. Die Aufnahmefrequenz der Blickpunkte betrug 30 Frames pro Sekunde. Die Studie wurde mit der Software Tobii Studio 3.1 entworfen und durchgeführt. Während der Studie entstanden 206.968 Fixationen.

7.1.3. Gewählter Stimulus und dazu gestellte Aufgabe

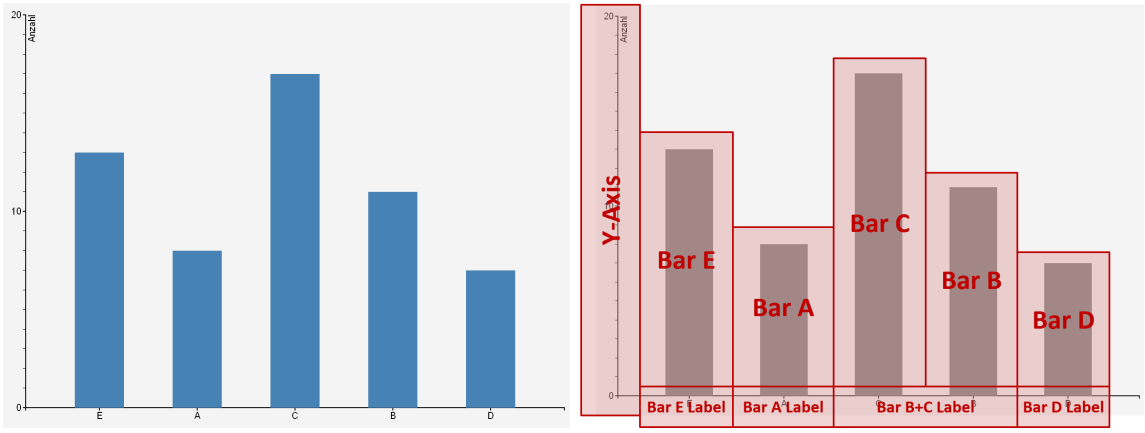

Abbildung 7.1.: Darstellung des für das erste Szenario genutzten Stimulus. Die linke Abbildung stellt den Stimulus dar, wie er den Probanden während der Studie gezeigt wurde. Auf der rechten Seite ist der mit AOIs annotierte Stimulus dargestellt. Es gibt eine AOI für die y-Achse und eine für jeden Balken, sowie für die zugehörigen Labels. Die Labels für die Balken „B“ und „C“ wurden zusammengefasst.

Der zu Demonstrationszwecken gewählte Stimulus entstammt dem Aufgabentyp „Ablesen von Werten aus Balkendiagrammen“. Hierbei wurden neun der zehn Datensätze der Probanden für die Analyse genutzt. Der Stimulus ist in Abbildung 7.1 auf der linken Seite dargestellt. Dieser zeigt zwei Koordinatenachsen und fünf Balken mit den Beschreibungen „A“, „B“, „C“, „D“ und „E“. Die Probanden wurden gebeten, die Balken „A“ und „D“ zu vergleichen und den höheren Wert der beiden Balken anzugeben. Die Areas of Interest wurden entsprechend der rechten Seite von Abbildung 7.1 definiert. Als AOIs wurden die y-Achse und jeder Balken und ihre jeweiligen Labels gewählt. Da die Balken „B“ und „C“ nicht von Interesse waren, wurden deren Labels als „Label C+B“ zusammengelegt. Zum Vergleich mit der Darstellung des Prototyps sind in Abbildung 7.2 die Fixationen als Scanpath und als Heat Map dargestellt.

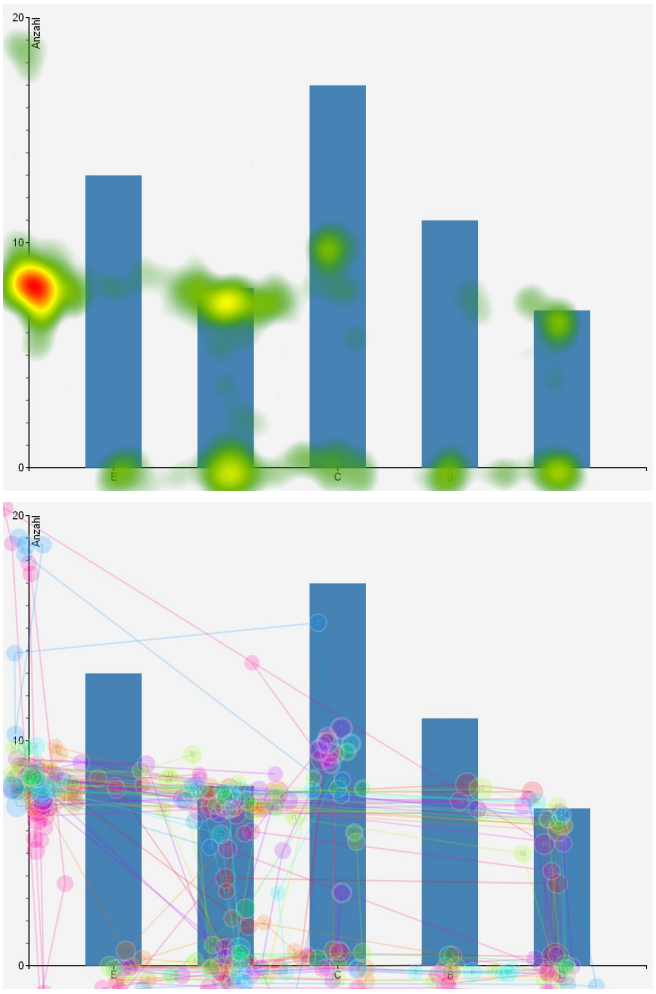

Abbildung 7.2.: Darstellung der Fixationen des ersten Szenarios auf einer Heat Map und als Scan-Path. Oben werden die Fixationen als Heat Map auf dem Stimulus aufgetragen. Unten sind dieselben Fixationen als Scan-Path dargestellt.

Von den 206.968 Fixationen der Studie entfallen 2251 Fixationen auf das gewählte Balkendiagramm. Nach dem Matching verblieben 316 Transitionspunkte für die weitere Analyse in der ETA-Pipeline und der anschließenden Visualisierung.

7.1.4. Durchführung der Analyse

Aus dem Stimulus lässt sich in Verbindung mit der Aufgabenstellung die Hypothese aufstellen, dass es mindestens zwei Gruppen geben muss. Dabei sollten die Probanden einer der Gruppen zunächst den Balken „A“ auslesen und danach den Balken „D“. Die Probanden einer weiteren Gruppe sollten mit Balken „D“ anfangen und dann zu Balken „A“ wechseln.

Zunächst wurde eine Visualisierung aller Daten erzeugt. Diese Visualisierung ist in Abbildung 7.3 abgebildet. Aus dieser geht hervor, dass die Probanden „P01“ und „P02“ eine

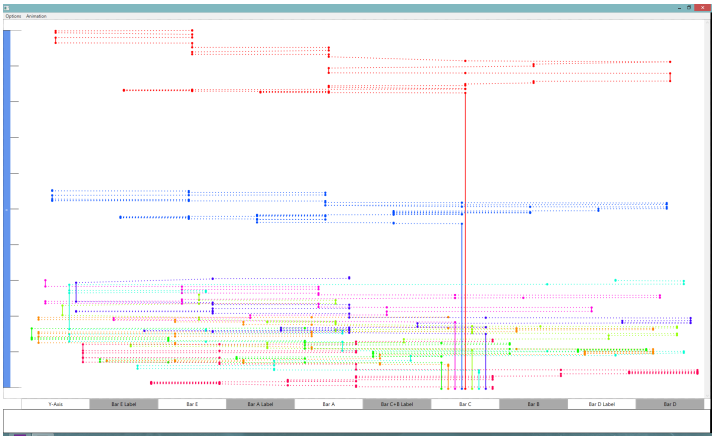

Abbildung 7.3.: Darstellung der Visualisierung des gesamten genutzten Datensatzes. Jeder Proband gehört einer eigenen Gruppe an. Es gibt in dieser Darstellung zwei Ausreißer: „P01“ und „P02“ verbringen deutlich mehr Zeit auf der AOI des Balkens „C“ als die anderen Probanden. Alle Probanden starten bei „C“, da sie gebeten wurden zu Beginn der Aufnahme die Mitte des Bildschirms zu fixieren. Dort ist die AOI des Balkens „C“ ebenfalls definiert.

deutliche Verzögerung der Aufnahme vorweisen. Hierdurch sind diese nur schwer mit den anderen Probanden in der zeitabhängigen Visualisierung zu vergleichen. Da diese jedoch bei der zeitunabhängigen Darstellung der Daten nicht mehr auffallen, wurden diese nicht aus der Studie gefiltert. Anschließend wurden die Probanden mit dem Levenshtein-gestützten Clustering gruppiert. Dabei wurde ein Ähnlichkeitsschwellwert von 0.68 gewählt. Daraus ergaben sich drei Gruppen. Die gruppierten Daten wurden wie in Abbildung 7.4 visualisiert. Um die Gruppen besser vergleichen zu können, wurden mit dem Centroid-Filter Repräsentanten für die Gruppen berechnet. Die daraus entstehende Visualisierung ist in Abbildung 7.5 dargestellt.

7.1.5. Ergebnisse der Untersuchung

Aus Abbildung 7.5 kann man einen Zusammenhang zwischen dem Verhalten der Gruppen und der Aufgabenstellung ziehen. Die Aufgabenstellung forderte, die Balken „A“ und „D“ miteinander zu vergleichen und den höheren Wert auszulesen. Das Verhalten der einzelnen Gruppen war wie folgt:

Rote Gruppe: Die rote Gruppe liest zunächst den Balken „A“ ab und vergleicht diesen dann mit dem Balken „D“. Anschließend wird erneut der Balken „A“ abgelesen und der Wert an der y-Achse abgelesen.

Grüne Gruppe: Die grüne Gruppe beginnt mit Balken „D“ und wechselt dann zu Gruppe „A“. Dann liest sie die y-Achse ab.

Blaue Gruppe: Die blaue Gruppe weist kein spezifisches Muster auf und wechselt häufig zwischen den verschiedenen Balken. Es ist möglich, dass diese Personen die Aufgabe

nicht oder falsch verstanden haben. Aufgrund dieser Annahme wird diese Gruppe als Ausreißergruppe angenommen.

7.2. Analyse von Eye-Tracking-Daten einer Autofahrt

Die Daten des zweiten Szenarios wurden von der Abteilung R&D, Driver Analysis, der Daimler AG zur Verfügung gestellt. Hier wurde der Eye-Tracker in einem PKW angebracht und es wurde mit dem Eye-Tracker das Blickverhalten des Fahrers während einer Autofahrt in der Stadt und auf der Autobahn aufgenommen. Der folgende Abschnitt erläutert zunächst das Szenario. Daraufhin werden die Daten der Eye-Tracking-Aufnahme, wie die Aufnahmelänge, beschrieben. Anschließend wird auf die Erweiterung der zur Verfügung gestellten Daten eingegangen. Danach wird die Durchführung der Analyse beschrieben. Zuletzt werden die Ergebnisse der Untersuchung vorgestellt.

7.2.1. Szenario

Die Daimler AG testet experimentell den Nutzen von Eye-Trackern in PKWs, um festzustellen, ob sich über das Blickverhalten Schlüsse bezüglich des Fahrverhaltens schließen lassen bzw. ob bestimmte Blickmuster einen Schluss auf bestimmte Ereignisse, wie etwa den Beginn eines Staus, zulassen. Dazu hat ein Teilnehmer eine ca. 45 minütige Fahrt unternommen. Diese Fahrt wurde auf einem Video, das ein Bild der Front- und Heckkamera des Wagens sowie das Aufnahmebild des Eye-Trackers und dessen Verwaltungssoftware enthält, aufgenommen. Die Aufnahme startet in einer Tiefgarage. Nach ca. zwei Minuten fährt das Fahrzeug aus der Tiefgarage und folgt einer Straße Richtung Autobahn. Der Fahrer erreicht nach 5:33 Minuten die Autobahn. Zum Zeitpunkt 7:13 min beginnt ein Stau, der sich zum Zeitpunkt 8:20 min wieder auflöst. Nach 27:30 Minuten verlässt der Fahrer die Autobahn und fährt über einen

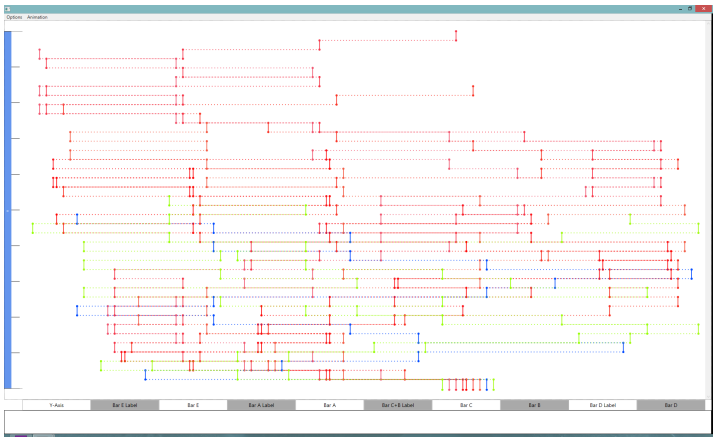

Abbildung 7.4.: Darstellung der Daten, nachdem sie im Analyseschritt gruppiert wurden. Es sind drei Gruppen zu sehen, die in grün, blau und rot dargestellt sind. Die Daten sind zeitunabhängig dargestellt.

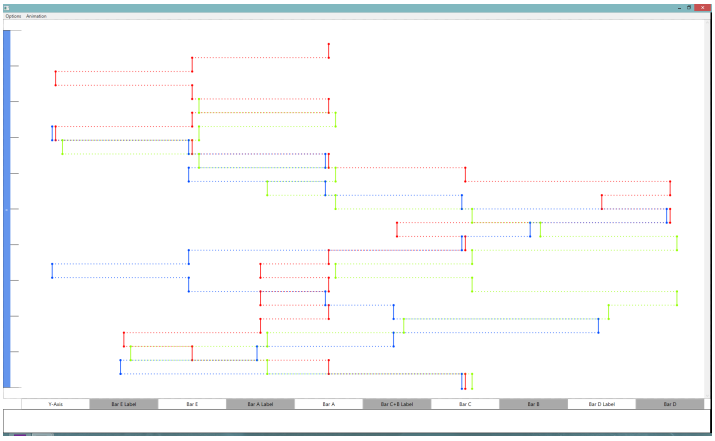

Abbildung 7.5.: Darstellung in 7.4 dargestellten Daten. Hierbei wurden für die Gruppen während der Analyse Repräsentanten berechnet, die nun explizit dargestellt werden. Die rote Gruppe sieht dabei zunächst den Balken „A“ an und danach den Balken „D“. Die grüne Gruppe beginnt mit Balken „D“ und wechselt dann zu Balken „A“. Die blaue Gruppe weist kein erklärbares Blickmuster auf.

Stadtbereich wieder zurück zum Ausgangspunkt. Die Aufnahme endet, nachdem der PKW in der Tiefgarage geparkt wurde.

7.2.2. Daten der Aufnahme

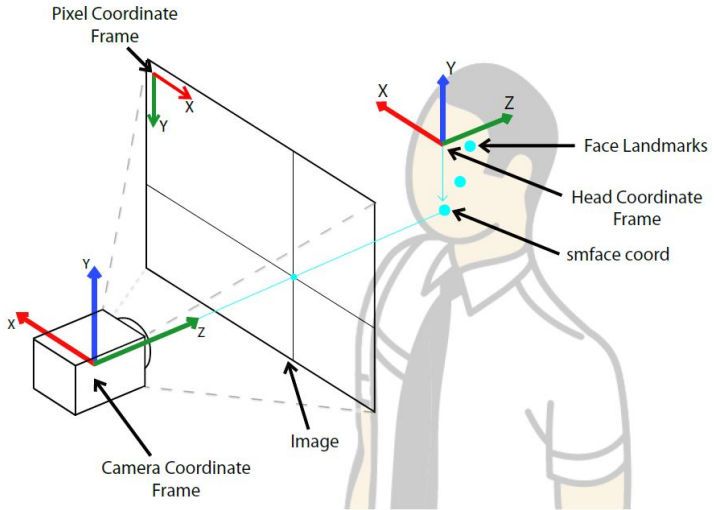

Abbildung 7.6.: Schematische Darstellung des genutzten Koordinatensystems. Der Blick des Probanden wird auf eine Ebene im Raum projiziert. Quelle: [28]

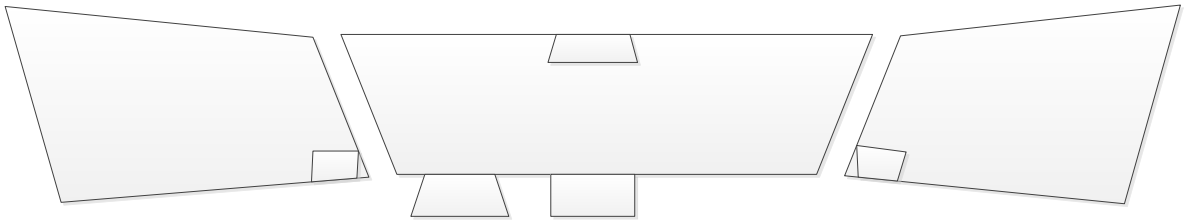

Abbildung 7.7.: Schematische Abbildung der AOIs aus Sicht des Fahrers, die für dieses Szenario genutzt wurden. Es sind AOIs für die Windschutzscheibe und die Außenfenster, jeweils mit Rückspiegeln, vorgesehen. Zudem gibt es AOIs für die Instrumententafel und die Comand-Konsole.

Das im Szenario verwendete Eye-Tracking-System beschreibt die Blickpunkte in einem Koordinatensystem, das in Abbildung 7.6 dargestellt wird. Die Abweichungen den Blickpunkte vom Ursprung werden in metrischen Einheiten aufgezeichnet. Der für diese Arbeit zur Verfügung stehende Datensatz beinhaltet einen Probanden, der eine ca. 45-minütige Autofahrt im Stadtverkehr und auf der Autobahn zurücklegt. Dabei wurde alle 16,66 ms ein Blickpunkt aufgezeichnet. Der Prototyp nimmt beim Import dieser Daten an, dass ein Blickpunkt einer Fixation entspricht. Dies entspricht einer Gesamtfixationszahl von 163.240 Fixationen. Das Cockpit wurde mittels der Areas of Interest modelliert und ist in Abbildung 7.7 dargestellt. Die AOIs decken dabei die Instrumententafel, die Comand-Konsole, die Windschutzscheibe sowie die Außenfenster ab. In der Windschutzscheibe und den Fenstern sind zudem die Rückspiegel als untergeordnete AOI hinterlegt.

Zudem stand ein Video der Autofahrt zur Verfügung. Dieses ist viergeteilt und zeigt

1. das Bild der Frontkamera,
2. das Bild der Heckkamera,
3. das Aufnahmebild des Eye-Trackers und
4. die Verwaltungssoftware des Eye-Trackers.

7.2.3. Anpassung und Erweiterung der zur Verfügung gestellten Daten

Um die Daten mit der vom Prototyp verwendeten Datenbank kompatibel zu machen, wurden die Koordinaten der Blickpunkte um den Faktor 10.000 erhöht und als Pixelkoordinaten angenommen. Die Abbildung der AOIs wurde von Hand angepasst. Anschließend wurden aus den Videodaten manuell Szenen extrahiert. Diese wurden ebenfalls in die Datenbank eingefügt.

7.2.4. Durchführung der Analyse

Da die Daten nur einen Probanden enthalten, können die Daten in der ETA-Pipeline nicht weiter gefiltert werden. Die Daten wurden daher ohne eine weitere Aufbereitung der Daten visualisiert. In der Visualisierung wurden zunächst alle Daten auf den PSPs angezeigt. Der

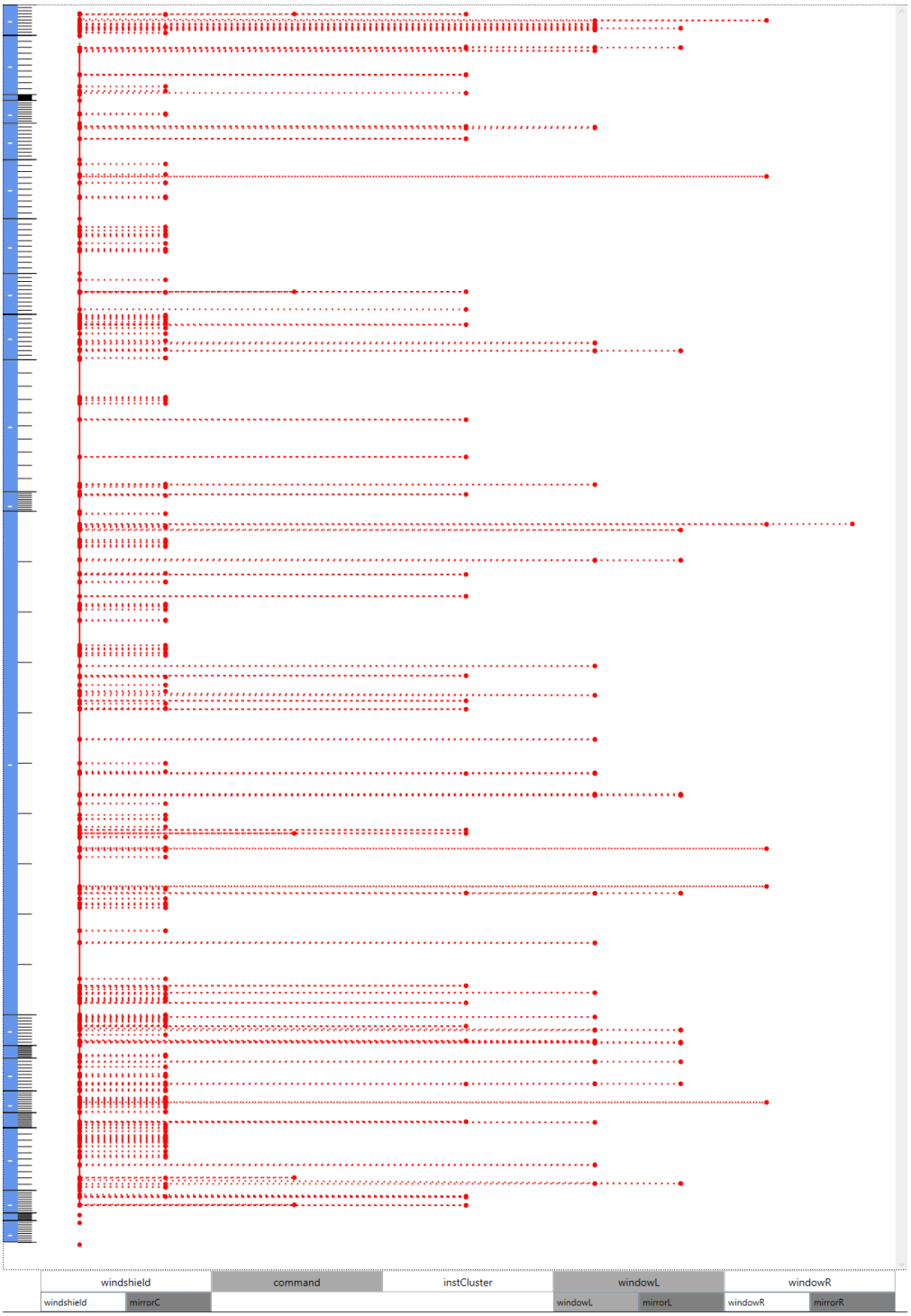

Abbildung 7.8.: Abbildung des vollständigen PSPs der Autofahrt des zweiten Szenarios. Die Szenen sind links aufgelistet und die AOIs sind unten entsprechend der im Datenabschnitt beschriebenen Hierarchie dargestellt.

gesamte Verlauf ist in Abbildung 7.8 dargestellt. Daraufhin wurden die einzelnen Szenen der Aufnahme genauer betrachtet.

7.2.5. Ergebnisse der Untersuchung

Abbildung 7.9.: Darstellung der Fahrtszene im Stadtverkehr. Der Ausschnitt wurde mit der Zoomfunktion der Visualisierung des Prototyps erzeugt.

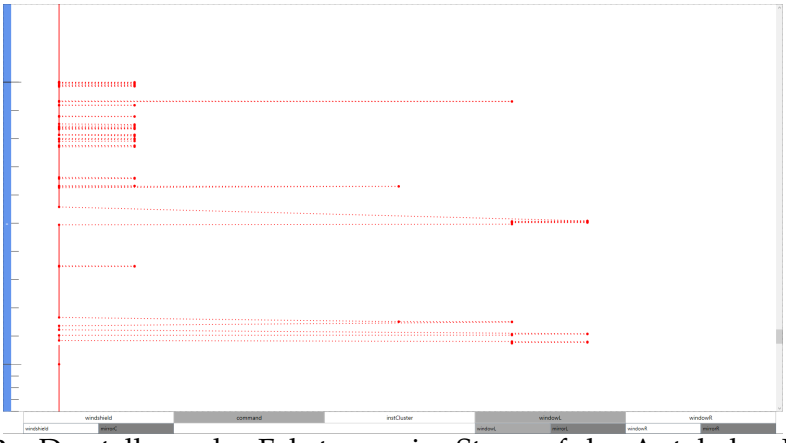

Abbildung 7.10.: Darstellung der Fahrtszene im Stau auf der Autobahn. Der Ausschnitt wurde mit der Zoomfunktion der Visualisierung des Prototyps erzeugt.

Bei der Gesamtansicht fällt auf, dass der Proband nur selten die Comand-Konsole ansieht. Das legt nahe, dass der Proband entweder keine Funktionen der Konsole nutzen wollte oder dass die Nutzung des Systems während der Fahrt einen zu hohen kognitiven Aufwand bedeuten würde. Auch der rechte Außenspiegel hat sehr wenige Fixationen. Dies könnte darauf hinweisen, dass der rechte Außenspiegel nur selten, etwa bei einem Spurwechsel auf die rechte Spur, benötigt wird. Aufgrund der geringen Fixationszahl ist es jedoch wahrscheinlicher, dass der Eye-Tracker die Augen des Probanden häufig verloren hat, wenn dieser in den rechten Außenspiegel gesehen hat. Da Autofahrer bei nach vorne gerichteter Kopfhaltung den rechten Außenspiegel nur schlecht einsehen können, drehen diese häufig beim Blick in den rechten

Außenspiegel den Kopf nach rechts. Hierbei verliert der Eye-Tracker möglicherweise die Augen.

Bei einer näheren Betrachtung der Stauszene fällt zudem auf, dass der Proband deutlich seltener in die Außenspiegel sieht. Dies lässt sich dadurch erklären, dass es nur selten Spurwechsel während eines Staus gibt und daher die angrenzenden Spuren nur von geringem Interesse sind.

Eine genauere Analyse der Szene im Stadtverkehr zeigt, dass hierbei die Instrumententafel deutlich seltener betrachtet wurde als während der Autobahnfahrt. Dies legt nahe, dass die Überprüfung der Werte der Instrumententafel in der Innenstadt, beispielsweise zum Zweck der Einhaltung der Geschwindigkeitsgrenze, als weniger wichtig eingestuft wird. Eine mögliche Erklärung hierfür ist, dass bei normaler Verkehrsdichte das Geschwindigkeitslimit nur selten überschritten werden kann und dass die Aufmerksamkeit verstärkt auf dem Verkehr liegt.

8. Zusammenfassung und Ausblick

Dieses Kapitel fasst zunächst das Konzept, die Implementierung und die Demonstration und Evaluation zusammen. Daraufhin wird ein Ausblick darüber gegeben, wie das Konzept und der Prototyp weiterentwickelt werden können.

Zusammenfassung

Diese Arbeit befasste sich mit dem Entwurf und der Implementierung von neuen visualisierungsbasierten Analysetechniken für Eye-Tracking-Daten. Hierzu wurde unter Nutzung der Konzepte und Techniken der Visual Analytics eine Eye-Tracking-basierte Pipeline, kurz ETA-based Pipeline, entwickelt, um die Daten von Eye-Tracking-Studien zu analysieren und zu visualisieren. Dazu wurden zunächst die Fixationsdaten mit den AOIs der Stimuli der Studie verbunden und dann in eine Datenbank importiert. Diese Daten können mit Hilfe verschiedener Filter aufbereitet und unter Nutzung der Parallel Scan-Paths visualisiert werden.

Um die Umsetzung des Konzepts zu testen, wurde ein Plug-in-basierter Prototyp entwickelt. Dieser Prototyp unterstützt die Analyse der Daten mittels Analyse-Plug-in-Filtern. Die Plug-ins stellen dabei Funktionalität zur Gruppierung mittels eines auf der Levenshtein-Distanz basierten HAC-Algorithmus, der Auswahl der in der Analyse genutzten Gruppen und Probanden, sowie die Berechnung eines Repräsentanten für die jeweiligen Gruppen zur Verfügung. Auf die Analyse folgt die Visualisierung der Daten mittels der Parallel Scan-Paths. Die Visualisierung unterstützt zudem ineinander geschachtelte AOIs und bietet eine Zoomfunktion für die Daten. Außerdem können für die Stimuli vor dem Import in die Datenbank Szenen definiert werden, die in der Visualisierung ein- und ausgeblendet werden können. Des Weiteren unterstützen die PSPs Brushing der einzelnen Probanden.

Zuletzt wurde der Prototyp anhand zweier Szenarien evaluiert. Das erste Szenario entstammt der Studie „Visual Elements III“ des Instituts für Visualisierung und interaktive Systeme und vergleicht das Verhalten der Probanden beim Ablesen von einem exemplarisch ausgewählten Balkendiagramm. Das zweite Szenario zeigt die Aufnahme des PKW-Fahrers während einer Fahrt auf der Autobahn und durch die Innenstadt. Die Daten wurden von der Abteilung R&D, Driver Analysis, der Daimler AG aufgenommen und zur Verfügung gestellt.

Ausblick

Das Konzept und der Prototyp können um diverse Aspekte, wie die Auswertung der verschiedenen mathematischen Modelle der Levenshtein-Distanz, die mögliche Erweiterung der Datenanalyse um weitere Stringvergleichsalgorithmen und die Erweiterung des Konzepts auf

große und mobile Endgeräte erweitert werden. Diese Aspekte werden im Folgenden näher beschrieben.

Auswertung der verschiedenen mathematischen Modelle

Obwohl der Prototyp bereits verschiedene mathematische Modelle zur Modellierung der Kostenfunktion für die Levenshtein-Distanz unterstützt, wurden in der Evaluation nur konstante Kosten für die Distanzberechnung bei der Erstellung der Cluster genutzt. Um die Kosten über die Zeit anpassen zu können, ist eine detailliertere Evaluation der Anwendbarkeit und Auswirkungen der einzelnen mathematischen Modelle erforderlich.

Erweiterung der Datenanalyse um weitere Stringvergleichsalgorithmen

Im Konzept wurden bereits verschiedene Algorithmen für Stringvergleiche vorgestellt. Diese sollten in den Prototyp integriert werden, um deren spezielle Eigenschaften und deren Eignung für verschiedene Eye-Tracking-Studien zu evaluieren. Auch die im Konzept beschriebene Kombination des Pattern Matching mit dem Algorithmus von Levenshtein oder dem Algorithmus von Needleman und Wunsch bietet die Möglichkeit, zukünftig Muster in Blickfolgen besser zu erkennen.

Erweiterung des Konzepts auf große Displays und mobile Endgeräte

Die Trennung der Analyse von der Visualisierung der Daten ermöglicht, dass die Daten dezentral analysiert werden können. So ist es beispielsweise denkbar, dass Studien künftig von einer Gruppe von Analysierenden bearbeitet werden kann, indem die Visualisierung auf einem großen Display, wie etwa einer Powerwall, angezeigt und die Datenanalyse auf mobilen Endgeräten, wie Smartphones oder Tablets, durchgeführt wird. In [23] wird ein Ansatz für ein solches Interaktionskonzept beschrieben.

A. Datenbank

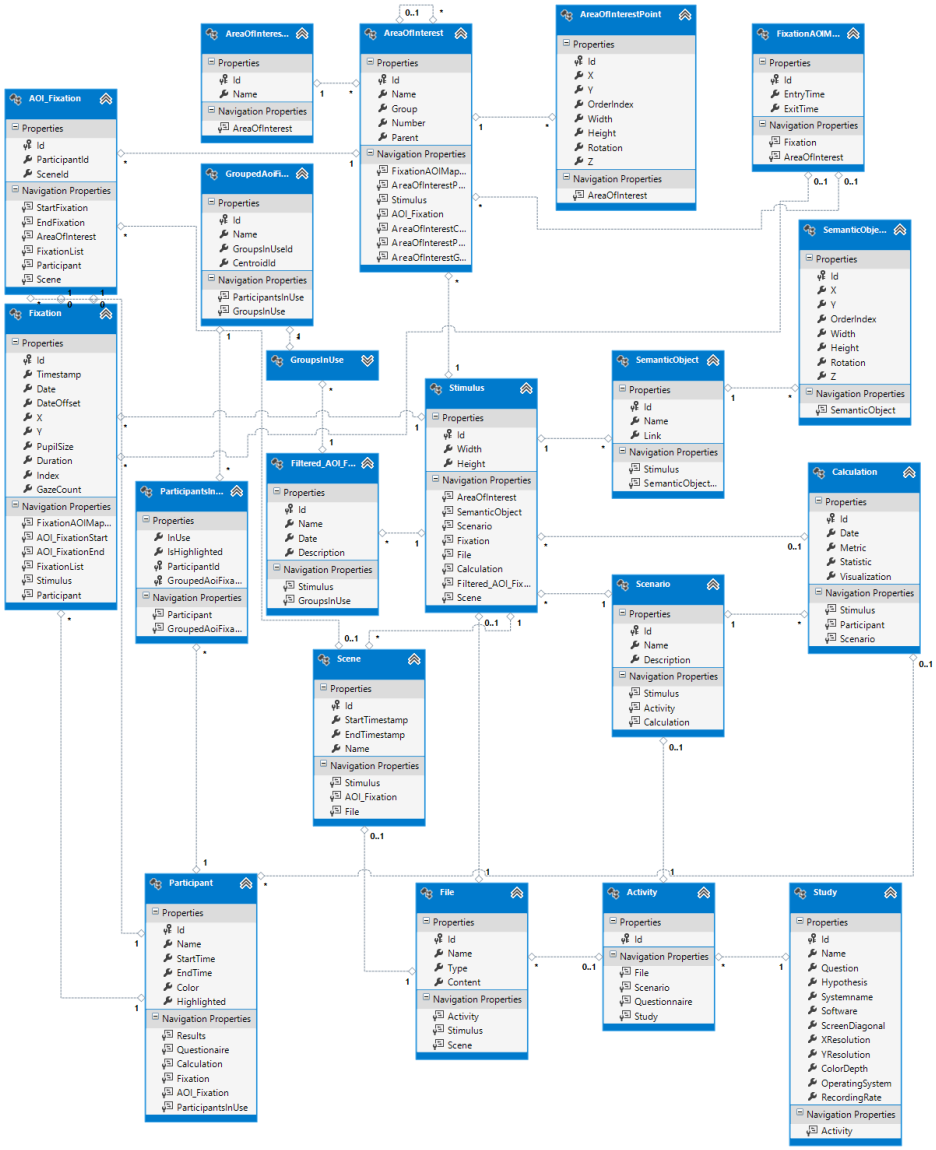

Abbildung A.1.: Darstellung des gesamten Datenbankmodells.

Literaturverzeichnis

[1] BLOCKEEL, H., DEMOEN, B., DUVAL, E., LAVRAC, N., AND JACOBS, N. Relational sequence learning and user modelling. (Zitiert auf Seite 15)

[2] CARD, S. K., MACKINLAY, J. D., AND SHNEIDERMAN, B. *Readings in information visualization: using vision to think*. Morgan Kaufmann Pub, 1999. (Zitiert auf Seite 26)

[3] DUCHOWSKI, A. T. *Eye tracking methodology: Theory and practice*, vol. 373. Springer, 2007. (Zitiert auf Seite 10)

[4] EAGEREYES. Seite mit Parallelen Koordinaten. http://eagereyes.org/techniques/parallel-coordinates. (Zitiert auf den Seiten 24 und 25)

[5] HEINZ PENIN, W. F. Elektroenzephalografie (EEG). http://www.izepilepsie.de/home/showdoc,id,392,aid,650.html. (Zitiert auf Seite 60)

[6] HELMHOLTZ, H. V., AND SOUTHALL, J. P. C. Treatise on physiological optics. iii. the perceptions of vision. (Zitiert auf Seite 10)

[7] HOLMQVIST, K., NYSTRÖM, M., ANDERSSON, R., DEWHURST, R., JARODZKA, H., AND VAN DE WEIJER, J. *Eye tracking: A comprehensive guide to methods and measures*, first edition ed. Oxford University Press, 2011. (Zitiert auf den Seiten 7, 8, 10, 12, 13, 21 und 57)

[8] HUANG, W., EADES, P., AND HONG, S.-H. Measuring effectiveness of graph visualizations: A cognitive load perspective. *Information Visualization 8*, 3 (2009), 139–152. (Zitiert auf Seite 13)

[9] INSELBERG, A., AND DIMSDALE, B. Parallel coordinates. In *Human-Machine Interactive Systems*. Springer, 1991, pp. 199–233. (Zitiert auf Seite 23)

[10] JAMES, W. The principles of psychology, vol i. (Zitiert auf Seite 10)

[11] JAVAL, E. Essai sur la physiologie de la lecture. In *Annales D'Oculistique* (1879). (Zitiert auf Seite 8)

[12] JUST, M. A., AND CARPENTER, P. A. A theory of reading: From eye fixations to comprehension. *Psychological review 87* (1980), 329–354. (Zitiert auf Seite 9)

[13] KEIM, D., ANDRIENKO, G., FEKETE, J.-D., GÖRG, C., KOHLHAMMER, J., AND MELANÇON, G. *Visual analytics: Definition, process, and challenges*. Springer, 2008. (Zitiert auf den Seiten V, 28 und 40)

[14] KNUTH, D. E., MORRIS, JR, J. H., AND PRATT, V. R. Fast pattern matching in strings. *SIAM journal on computing 6*, 2 (1977), 323–350. (Zitiert auf Seite 17)

[15] KURZHALS, K., AND WEISKOPF, D. Space-time visual analytics of eye-tracking data for dynamic stimuli. *IEEE Transactions on Visualization and Computer Graphics 12*, 19 (2013), toappear. (Zitiert auf den Seiten 13, 37 und 38)

[16] LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady* (1966), vol. 10, p. 707. (Zitiert auf den Seiten 15 und 37)

[17] LI, D., WINFIELD, D., AND PARKHURST, D. J. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on* (2005), IEEE, pp. 79–79. (Zitiert auf Seite 9)

[18] MANNING, C. D., RAGHAVAN, P., AND SCHÜTZE, H. *Introduction to information retrieval*, first edition ed., vol. 1. Cambridge University Press Cambridge, 2008. (Zitiert auf den Seiten 18 und 20)

[19] MICROSOFT CORP. Managed Extensibility Framework (MEF). http://msdn.microsoft.com/en-us/library/dd460648.aspx. (Zitiert auf Seite 74)

[20] NEEDLEMAN, S. B., AND WUNSCH, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology 48*, 3 (1970), 443–453. (Zitiert auf den Seiten 15, 16 und 37)

[21] NOTON, D., AND STARK, L. Scanpaths in eye movements during pattern perception. *Science* (1971). (Zitiert auf Seite 12)

[22] NOTON, D., AND STARK, L. Scanpaths in saccadic eye movements while viewing and recognizing patterns. *Vision research 11*, 9 (1971), 929–IN8. (Zitiert auf Seite 12)

[23] PFLEGING, B., RASCHKE, M., PÜTTMANN, E., AND PLONER, N. Powerwall interactions. (Zitiert auf Seite 100)

[24] PURCHASE, H. C., ANDRIENKO, N., JANKUN-KELLY, T., AND WARD, M. Theoretical foundations of information visualization. In *Information Visualization*. Springer, 2008, pp. 46–64. (Zitiert auf Seite 21)

[25] RASCHKE, M., CHEN, X., AND ERTL, T. Parallel scan-path visualization. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (2012), ACM, pp. 165–168. (Zitiert auf den Seiten 31, 32, 33 und 40)

[26] R&B GROUP. Bee-Swarm. http://eyetracking.com.ua/eng/visualization/10.html. (Zitiert auf Seite 14)

[27] ROSENHOLTZ, R., LI, Y., MANSFIELD, J., AND JIN, Z. Feature congestion: a measure of display clutter. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2005), ACM, pp. 761–770. (Zitiert auf Seite 13)

[28] SEEING MACHINES. faceAPI - The Real-Time Face Tracking Toolkit for Developers and OEMs. http://www.seeingmachines.com/pdfs/brochures/faceAPI-techspecs.pdf. (Zitiert auf Seite 94)

[29] SHNEIDERMAN, B. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on* (1996), IEEE, pp. 336–343. (Zitiert auf den Seiten 25 und 51)

[30] SMITH, T. F., WATERMAN, M. S., AND BURKS, C. The statistical distribution of nucleic acid similarities. *Nucleic Acids Research 13*, 2 (1985), 645–656. (Zitiert auf den Seiten 15 und 37)

[31] STELLMACH, S., NACKE, L. E., DACHSELT, R., AND LINDLEY, C. A. Trends and techniques in visual gaze analysis. *arXiv preprint arXiv:1004.0258* (2010). (Zitiert auf Seite 11)

[32] STROHMAIER, S. Entwicklung eines Konzepts zur Annotation von grafischen Elementen in Visualisierungen. Master's thesis, Universität Stuttgart, 2013. (Zitiert auf Seite 44)

[33] THOMAS, J. J., AND COOK, K. A. *Illuminating the path: The research and development agenda for visual analytics*. IEEE Computer Society Press, 2005. (Zitiert auf Seite 27)

[34] TINKER, M. A. Eye movement, perception, and legibility in reading. *Psychological Bulletin 33*, 4 (1936), 275. (Zitiert auf Seite 8)

[35] TINKER, M. A. Reliability and validity of eye-movement measures of reading. *Journal of Experimental Psychology 19*, 6 (1936), 732. (Zitiert auf Seite 8)

[36] TSANG, H. Y., TORY, M., AND SWINDELLS, C. eseetrack-visualizing sequential fixation patterns. *Visualization and Computer Graphics, IEEE Transactions on 16*, 6 (2010), 953–962. (Zitiert auf den Seiten 34, 35 und 36)

[37] TULLEKEN, H. How to Choose Colours Procedurally (Algorithms). http://devmag.org.za/2012/07/29/how-to-choose-colours-procedurally-algorithms/. (Zitiert auf Seite 86)

[38] WEST, J. M., HAAKE, A. R., ROZANSKI, E. P., AND KARN, K. S. eyepatterns: software for identifying patterns and similarities across fixation sequences. In *Proceedings of the 2006 symposium on Eye tracking research & applications* (2006), ACM, pp. 149–154. (Zitiert auf den Seiten 36 und 37)

[39] WIKIMEDIA COMMONS. Aufbau des menschlichen Auges. http://de.wikipedia.org/wiki/Auge. (Zitiert auf Seite 7)

[40] WIKIMEDIA COMMONS. Farbwahrnehmung der Zaepfchen des menschlichen Auges. http://de.wikipedia.org/wiki/Farbwahrnehmung#Die_Sehzellen. (Zitiert auf Seite 6)

[41] ZÜHLKE, D. *Nutzergerechte Entwicklung von Mensch-Maschine-Systemen: Useware-Engineering für technische Systeme*. Springer DE, 2012. (Zitiert auf Seite 6)

Alle URLs wurden zuletzt am 10. 12. 2013 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift