Bachelorarbeit

von

Steffen Goos

Entwicklung eines Tools zur Auswertung relevanter Bereiche in Eyetracking-Daten mittels statischen und dynamischen AOI.

Betreuer:

Prof. Dr. Ansgar Sherp,

M.Sc. Falk Böschen,

Dipl.-Psych. Benjamin Strobel
(IPN Kiel)

Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Knowledge Discovery
Düsternbrooker Weg 120, 24105 Kiel

September 2015

Kurzfassung

Inhalt dieser Bachelorarbeit ist die Entwicklung und Erstellung eines Tools zur Auswertung von statischen und dynamischen Area of Interest (AOI) mittels Eyetracking-Daten. Als dynamische AOI werden dabei Bereiche bezeichnet, die über einen definierten Zeitraum ihre Größe, Form und Position verändern können. Zusätzlich soll die Möglichkeit bestehen, AOI durch eine Clusteranalyse automatisch zu erkennen. Das Analysetool soll dabei als Grundgerüst für Auswertungsmethoden dienen, die beliebig hinzugefügt werden können. Zunächst werden grundlegende Begriffe des Eyetracking und verschiedene Auswertungsmethoden, insbesondere die der Clusteranalyse erläutert. Als Nächstes folgt die Aufstellung der Anforderungen an das Analysetool und die Umsetzung in eine Programmarchitektur. Dieses Konzept wird in ein Java-Framework überführt und softwarespezifische Aspekte beleuchtet. Im Anschluss wird durch die Auswertung eines Beispieldatensatzes die Funktionalität in den Bereichen statische AOI, dynamische AOI und AOI-Erkennung demonstriert.

Abstract

Goal of this thesis is the development and implementation of a tool for analysis of static and dynamic Area of Interest (AOI) by use of eyetracking-data. Dynamic AOI are regions, that can change their size, form and position over a defined time. Furthermore, it should be possible to detect AOI automaticly by the use of Clustering. First, the basic terms of eyetracking will be explained. In the next step, the requirements are set up and transferred to an architecture. Followed by the implementation in Java and the application of a sample data set.

Inhaltsverzeichnis

Kurzfassung iii

Inhaltsverzeichnis vii

1 Einleitung 1
1.1 Anwendungsgebiete . . . 1
1.2 Gegenstand der Bachelorarbeit . . . 1

2 Hintergrund und Stand der Forschung 3
2.1 Eyetracking . . . 3
2.1.1 Funktionsweise . . . 3
2.1.2 EyeMotion-Framework . . . 4
2.2 Fixationen und Sakkaden . . . 5
2.2.1 Erkennung von Fixationen . . . 5
2.3 Area of Interest (AOI) . . . 6
2.3.1 Transitionsmatrix . . . 6
2.3.2 AOI Transitionsdiagramm . . . 7
2.4 Erkennung von AOI mittels Clusteranalyse . . . 8
2.4.1 Partitionierende Clusteranalyse . . . 8
2.4.2 Hierarchische Clusteranalyse . . . 8
2.4.3 Dichtebasierte Clusteranalyse . . . 9
2.5 Scalable Vector Graphics . . . 10
2.5.1 Animationen . . . 11

3 Anforderung an das Analysetool 13
3.1 Anwenderanforderung . . . 13
3.1.1 Funktionale Anforderungen . . . 13
3.1.2 Nichtfunktionale Anforderungen . . . 14
3.2 Systemanforderung . . . 14
3.2.1 Funktionale Anforderungen . . . 14
3.2.2 Nichtfunktionale Anforderungen . . . 15

4 Architektur und Entwurf des Analysetools 17
4.1 Übersicht . . . 17
4.1.1 Prozessablauf . . . 17
4.2 Komponenten . . . 19
4.2.1 Analyzer . . . 19
4.2.2 AOIParser . . . 21
4.2.3 AnalyzerGUI . . . 22

5 Implementierung des Analysetools **25**
5.1 Verwendete Bibliotheken/Klassen 25
5.2 Auswertungsmethoden 26
5.2.1 DrawFixations 26
5.2.2 DrawAOI 26
5.2.3 DefaultAOIAnalysis 27
5.2.4 Clustering 28
5.3 Benutzeroberfläche 30

6 Proof of Concept: Anwendung des Analysetools **33**
6.1 Auswertung von statischen AOI 33
6.1.1 Ergebnis 34
6.2 Auswertung von dynamischen AOI 35
6.2.1 Ergebnis 36
6.3 Erkennung von AOI mittels Clusteranalyse 38
6.3.1 Heatmap 38
6.3.2 AOI 39

7 Fazit **41**
7.1 Entwicklung 41
7.2 Ausblick 41

Literatur **44**

Abbildungsverzeichnis **45**

Abkürzungsverzeichnis **47**

Erklärung **49**

1 Einleitung

Eyetracking erlaubt es, das Verhalten von Probanden beim Analysieren von Bildern und Videos zu erfassen und auszuwerten. Es lassen sich die Problemlösungsstrategien aufzeichnen und vergleichen. In den letzten Jahren sind die entsprechenden Geräte immer kompakter und günstiger geworden und mit dem Tobii EyeX Controller und dem EyeTribe sind zwei sehr preiswerte Eyetracker für den Endanwenderbereich in den Handel gekommen, sodass sich nun auch vermehrt Anwendungsgebiete im privaten Bereich finden lassen.

1.1 Anwendungsgebiete

Es ergeben sich eine Reihe von möglichen Anwendungsgebieten, sowohl im Bereich der Forschung, als auch in dem der Endanwender. Aus den erhobenen Eyetracking-Daten kann das Verhalten der Probanden beim Analysieren von Bildern oder Texten erforscht werden [SG00]. Bilder können anhand der Daten mit Tags versehen und die Suche nach Ihnen vereinfacht werden [WSS14]. Mit Hilfe von Eyetrackern lassen sich zudem Erkenntnisse zu Nutzerfreundlichkeit und Nutzungsstrategien bei der Informationssuche in Web-Portalen gewinnen [PSK06] und zur Verbesserung von Online-Auftritten nutzen.

Im Endanwenderbereich nehmen die Nutzungsmöglichkeiten vor allem im Spieleumfeld zu. Die Firma Steelseries hat 2015 den EyeX Controller von Tobii lizenziert und vertreibt ihn als Sentry Eye Tracker speziell an Computer-Spieler. Diese haben die Möglichkeit ihre Augenbewegungen während des Spielens aufzuzeichnen und im Anschluss analysieren zu lassen. Dadurch lassen sich, unter anderem, Spielabläufe optimieren. Des Weiteren können Spiele auch mit Augenbewegungen gespielt, oder durch diese ergänzt werden, wie zum Beispiel die Steuerung des Fadenkreuzes durch die Augen statt der Maus [Sau15].

1.2 Gegenstand der Bachelorarbeit

Im Rahmen einer Projektgruppe im Jahr 2015 wurde ein Java-Framework zur einheitlichen Ansteuerung von unterschiedlichen Eyetracker-Modellen geschaffen [GJLR15]. Dieses *EyeMotion* genannte Framework soll durch diese Bachelorarbeit um eine Komponente zur Auswertung der Eyetracking-Daten, insbesondere im Hinblick auf die Analyse von statischen und dynamischen Area of Interest (AOI), erweitert werden. Die Auswertung von dynamischen AOI stellt dabei eine Neuerung dar, die in dieser Form noch nicht zur Verfügung steht und eine Reihe neuer Möglichkeiten, wie zum Beispiel die Verfolgung des Blickes auf einem Werbeobjekt in einem Video, schafft. Diese dynamischen AOI

sollen sowohl in ihrer Position, als auch in der Größe veränderbar sein. Hierfür wird die *Synchronized Multimedia Integration Language* (SMIL) in Verbindung mit SVG-Dateien genutzt. Darüber hinaus soll das Auswertungstool durch eine Clusteranalyse die Möglichkeit zur automatischen Erkennung von AOI bieten. Ein wesentlicher Punkt bei der Erstellung des Tools stellt die Erweiterbarkeit da. Neue Auswertungsmethoden sollen auf eine einfache Weise hinzufügen lassen.

2 Hintergrund und Stand der Forschung

2.1 Eyetracking

Als Eyetracking wird das Verfolgen und Aufzeichnen von Augenbewegungen bezeichnet. Es gibt zwei Kategorien von Geräten: Mobile Eyetracker werden am Kopf des Probanden befestigt und zeichnen den Blick, während dieser sich frei bewegen kann, auf. Ein Anwendungsgebiet hierfür liegt in der Marktforschung, bei der eine Testperson durch einen Supermarkt gehen muss. Die zweite Kategorie bilden die stationären Eyetracker. Sie werden zum Beispiel vor einem Bildschirm positioniert und erfassen nur einen begrenzten Bereich.

2.1.1 Funktionsweise

Es gibt eine Reihe von verschiedenen Verfahren zur Bestimmung der Augenposition, darunter fallen Methoden, die sich die Nachbilder zunutze machen, oder mit speziellen Kontaktlinsen arbeiten, auf die hier nicht weiter eingegangen wird. Im Folgenden werden mit der Elektrookulografie und der Cornea-Reflex-Methode zwei weitere Verfahren vorgestellt, die sich im Gegensatz zu den oben genannten Methoden, durch eine einfachere Handhabung auszeichnen.

Elektrookulografie

Zwischen der Vorder- und Rückseite der Netzhaut besteht ein ständiger Spannungsunterschied, der dafür sorgt, dass die Hornhaut positiv und die Netzhaut negativ geladen ist. Die Elektrookulografie (EOG) nutzt diesen Umstand aus, indem Elektroden auf die Haut, an den sich gegenüberliegenden Seiten des Auges, angebracht werden. Eine Bewegung des Augen führt dazu, dass sich die Hornhaut der einen und die Netzhaut der anderen Elektrode annähert. Dies führt zu einer messbaren Spannungsveränderung mit der die Bewegung gemessen werden kann. Nutzt man zwei Paare von Elektroden, die um das Auge angebracht werden, lassen sich die horizontalen Werte EOG_h und die vertikalen Werte EOG_v ermitteln. [BWGT09]

Cornea-Reflex-Methode

Die Cornea-Reflex-Methode oder auch *video based eyetracking* nutzt eine Lichtquelle, zumeist im Infrarotbereich mit einer Wellenlänge von 790-880 nm, um das Auge zu beleuchten und die Reflexionen aufzuzeichnen. In diesem Wellenlängenbereich wird das

Licht nicht vom menschlichen Auge wahrgenommen und daher nicht als störend empfunden. Es existieren zwei Konzepte zur Anordnung der Lichtquellen: Beim *On-Axis* Verfahren liegt die Lichtquelle nah an der Kamera mit der Folge, dass die Pupillen einen großen Teil des Lichts zurück zum Eyetracker werfen und diese sehr hell erfasst werden. Beim Off-Axis Verfahren werden die Lichtquellen in einiger Entfernung zur Kamera angebracht und das aufgenommene Bilder der Pupillen ist dunkler. Studien haben gezeigt, dass Faktoren wie die Kopfposition, Blickrichtung und ethnischer Hintergrund der Probanden erheblichen Einfluss auf die Stärke der Reflexion haben. Aus diesem Grund verwenden viele Eyetracker eine Kombination von On- und Off-Axis Beleuchtung, die je nach Bedarf automatisch umgeschaltet wird [HJ10]. Eyetracker, die mit dieser Methode arbeiten, liefern in der Regel bei Augenbewegungen von bis zu 15° exakte Werte [Lev91]. Diese Technik wird von vielen aktuellen Eyetrackern verwendet, so auch von dem in dieser Bachelorarbeit benutzten Tobii EyeX Controller.

2.1.2 EyeMotion-Framework

Im Rahmen einer Projektarbeit an der Christian-Albrechts-Universiät zu Kiel, entstand 2015 ein EyeMotion genanntes Java-Framework zur Ansteuerung von Eyetrackern. Ziel dieser Projektarbeit war es eine Möglichkeit zu schaffen, verschiedene Eyetracker über einheitliche Schnittstellen anzusteuern und die anfallenden Daten in einer einheitlichen Form zu speichern. Aufgrund des unterschiedlichen Funktionsumfang der Eyetracker-Modelle ist es notwendig auf alle von den Herstellern gebotenen Funktionen zu verzichten und nur die Rohdaten zu verarbeiten. Zur Fixationserkennung arbeitet EyeMotion standardmäßig mit einem Dispersion-Threshold Identification (I-DT) Filter (siehe Abschnitt: 2.2). Es besteht darüber hinaus die Möglichkeit über eine Schnittstelle weitere Fixationsfilter zu implementieren. Neue Eyetracker lassen sich durch Implementierung des ITracker-Interface einbinden, dabei ist alles unterhalb dieser Ebene gerätespezifisch und vom restlichen Teil des Frameworks getrennt. [GJLR15]

Das EyeMotion-Framework eignet sich nicht nur für Versuchsaufbauten, sondern ermöglicht es auch jede Art von Anwendung zu realisieren, die auf die Daten eines Eyetrackers zugreifen soll. Im Rahmen der oben genannten Projektgruppe wurde untersucht, ob Informationen aus Infografiken bei unterschiedlicher Darstellungsweise anders, bzw. in anderer Geschwindigkeit analysiert werden. Um die Probanden in der Mitte des Experiments von den zuvor gestellten Fragen abzulenken, wurde ein Memory-Spiel implementiert, dass sich über die Augenbewegungen steuern ließ. Der Proband kann eine Karte durch Betrachtung markieren (siehe Abbildung 2.1) und durch die Betätigung der Leertaste aufdecken (siehe Abbildung 2.2). Dies ist ein einfaches Beispiel für die Nutzung von Eyetracking in Spielen. [GJLR15]

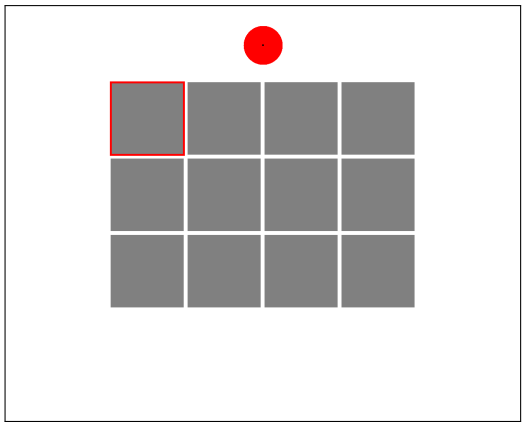

Abbildung 2.1: Memory: Auswahl

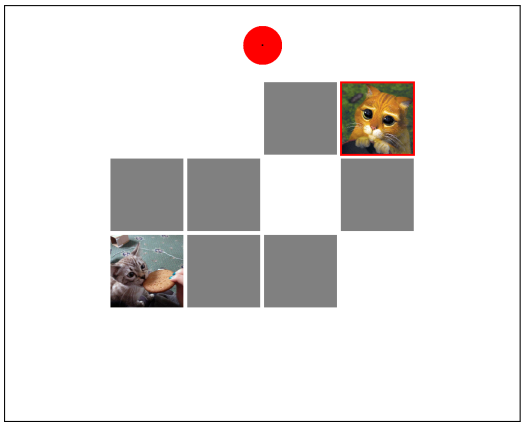

Abbildung 2.2: Memory: Aufgedeckt

2.2 Fixationen und Sakkaden

Im Eyetracking wird bei der Augenbewegung zwischen Fixation und Sakkade unterschieden. Bei einer Fixation ruht das Auge auf einem bestimmten Bereich und nimmt das Bild wahr. Eine Sakkade wiederum ist die Bewegung des Auges zwischen zwei Fixationen. In dieser Zeit werden keine Informationen aufgenommen, daher sind die Sakkaden für die Analyse von geringerer Bedeutung [SG00] [WSS14].

2.2.1 Erkennung von Fixationen

Verfahren zur Bestimmung von Fixationen lassen sich in drei Kategorien einteilen. Die einzelnen Algorithmen sind der Arbeit von Salvucci und Goldberg [SG00] entnommen und werden hier nicht im Detail beschrieben.

Geschwindigkeitsbasierte Algorithmen Diese Algorithmen basieren auf der Tatsache, dass eine Sakkade eine sehr viel höhere Geschwindigkeit als eine Fixation aufweist. Für jeden aufgenommen Punkt wird die Geschwindigkeit zum nächsten Punkt bestimmt. Hier seien die Velocity-Threshold Identification (I-VT) und die Hidden Markov Model Fixation Identification (I-HMM) genannt [SG00].

Dispersionsbasierte Algorithmen Aufgrund ihrer geringen Geschwindigkeit sind Fixationen als Gruppe von Punkten zu sehen, die dicht beieinander liegen. Algorithmen, die diese Tatsache ausnutzen, sind die Dispersion-Threshold Identification (I-DT) und die Minimum Spanning Tree Identification (I-MST) [SG00].

Bereichsbasierte Algorithmen Mittels vorher festgelegte AOI lässt sich untersuchen, ob Punkte in einem definierten Bereich liegen. Damit Sakkaden nicht fälschlicherweise als Fixation erkannt werden, ist es notwendig, eine minimale Dauer für eine Fixation

Tabelle 2.1: Vergleich der einzelnen Algorithmen [SG00]

Methode	Genauigkeit	Geschwindigkeit	Robustheit
I-VT	+	++	x
I-HMM	++	+	++
I-DT	++	+	++
I-MST	+	x	++
I-AOI	x	+	+

++ = sehr gut, + = gut, x = nicht gut

festzulegen. Im Gegensatz zu den oben genannten Möglichkeiten, können Fixation bei bereichsbasierten Algorithmen nur innerhalb der festgelegten AOI erkannt werden. Ein bereichsbasierter Algorithmus ist die Area-of-Interest Identification (I-AOI) [SG00].

EyeMotion-Framework

Das EyeMotion-Framework verwendet zur Erkennung von Fixationen standardmäßig den I-DT Algorithmus [GJLR15]. Er bietet eine hohe Genauigkeit bei der Bestimmung, sowie eine gute Geschwindigkeit und weist eine hohe Robustheit auf. Im Gegensatz zum I-HMM lässt er sich ohne viel Aufwand implementieren [SG00].

2.3 Area of Interest (AOI)

Als Area of Interest (AOI) oder Region of Interest (ROI) werden Bereiche bezeichnet, die vom Benutzer festgelegt werden und einen markanten Abschnitt in einem Stimulus beschreiben. Häufig werden AOI als Rechteck angegeben, andere Formen wie Kreise, Polygone und Pfade sind jedoch auch möglich. Es existieren Verfahren, um die AOI nach der Datenerhebung automatisch zu erkennen. Eines dieser Verfahren wird als Clusteranalyse bezeichnet und im Abschnitt 2.4 ausführlich behandelt. Die AOI-basierte Analyse von Eyetracking-Daten beschäftigt sich mit den Fixationen, die innerhalb der vorher definierten AOI liegen. Es lassen sich die Reihenfolge der betrachteten AOI und die jeweilige Verweildauer untersuchen. Dabei ist die Anzahl der Transitionen zwischen den AOI von großer Bedeutung, denn sie gibt Aufschluss über das Verhalten eines Probanden bei der Analyse eines bestimmen Stimulus.

2.3.1 Transitionsmatrix

Eine Transitionsmatrix ist eine Form der Repräsentation der Anzahl von Wechseln zwischen definierten AOI. Eine ungewöhnlich dichte Matrix, in der die meisten Felder mit einer eins versehen sind, lässt auf eine ausgedehnte Suche schließen und impliziert ein schlechtes Design. Eine spärlich gefüllte Matrix lässt hingegen auf eine effiziente Suche schließen. Die Transitionsmatrix kann durch einen quantitativen Wert beschrieben werden, indem man die Anzahl der Zellen, die mindestens eine Eins aufweisen durch die Anzahl aller

Zellen teilt. Ein großer Wert deutet auf eine große Streuung hin, wobei ein kleiner Wert für eine effiziente Suche steht. In Abbildung 2.3 wurde ein Bereich inhaltsunabhängig in gleichmäßige AOI aufgeteilt und die dazugehörige Transitionsmatrix konstruiert. [GK99]

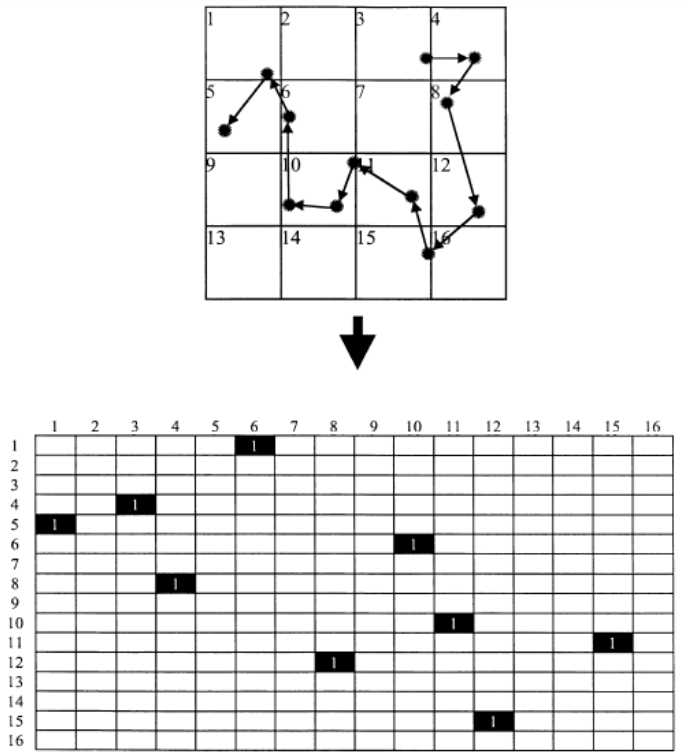

Abbildung 2.3: Beispiel einer Transitionsmatrix (Quelle: [GK99]))

2.3.2 AOI Transitionsdiagramm

Eine Möglichkeit die Daten einer Transitionsmatrix in ein übersichtliches Diagramm zu überführen zeigten Blascheck, Raschke und Ertl [BRE13] mit dem *Circular Heat Map Transition Diagram*. Die einzelnen AOI werden in gleichgroße Bereiche unterteilt und mit

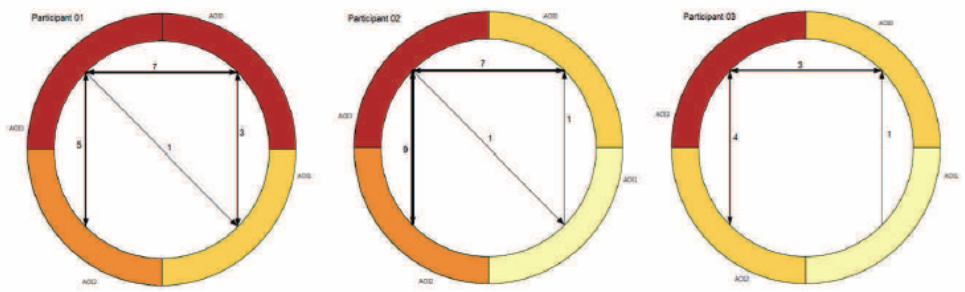

Abbildung 2.4: Beispiel eines Transitionsdiagramms (Quelle: [BRE13]))

Pfeilen die verschiedenen Transitionen von einer zur anderen AOI visualisiert. Dabei wird immer nur ein Pfeil für die Transitionen von AOI x zu AOI y gezeichnet. Die Anzahl der Transitionen kann an den Pfeil geschrieben werden. Die erfassten Fixationen in einem Bereich werden durch einen Farbcode ausgedrückt.

2.4 Erkennung von AOI mittels Clusteranalyse

Durch eine Clusteranalyse lassen sich eine Anzahl von Objekten in homogene Gruppen einteilen. Dazu muss zunächst ein Ähnlichkeitsmaß festgelegt werden, welches die Ähnlichkeit oder Unähnlichkeit zweier Objekte beschreibt. Im Fall der Clusteranalyse zur Erkennung von AOI ist dieses Ähnlichkeitsmaß die Distanz zwischen zwei Fixationen. Eine homogene Gruppe repräsentiert dabei eine AOI.

2.4.1 Partitionierende Clusteranalyse

Für die partitionierende Clusteranalyse ist es zwingend notwendig, die Anzahl der Cluster (k) zu Beginn anzugeben. Dies ist zugleich der größte Nachteil dieser Methode, denn oft ist diese Anzahl gar nicht bekannt. Als erstes werden k Cluster erstellt und deren Centroide festgelegt. Dies kann auf unterschiedliche Weise geschehen. Dann werden die Cluster solange iterativ verschoben, bis eine vorher definierte Zielfunktion minimal ist. [Est13] [EKSX96]

K-Means-Algorithmen

Zu den gängigsten partitionierende Algorithmen gehören die K-Means-Algorithmen. Die initiale Einteilung der Punkte in die k Klassen geschieht dabei im Allgemeinen zufällig. Nach dieser Einteilung werden die Centroide berechnet und jeder Punkt wird dem nächstgelegenen Centroid zugeordnet. Es entstehen neue Cluster, deren Centroide abermals berechnet werden müssen. Die Verschiebung von Punkten wird so oft wiederholt, bis sich keine neuen Cluster bilden. [Est13]

2.4.2 Hierarchische Clusteranalyse

Die hierarchische Clusteranalyse wird in *agglomerative* und *divisive* Verfahren unterteilt und bieten den Vorteil, dass die Anzahl der Cluster, im Gegensatz zu der partitionierenden Clusteranalyse, zu Beginn nicht festgelegt werden muss.

Agglomeratives Verfahren

Beim agglomerativen Verfahren werden zunächst alle Objekte in eigene Cluster eingeteilt. Nun werden die Abstände (Distanz - oder Ähnlichkeitsmaß) zwischen den Clustern berechnet und in eine Distanzmatrix eingetragen. Als Nächstes werden die zwei Cluster mit dem geringsten Abstand zueinander ausgewählt und zu Einem zusammengeführt. Dann müssen die Abstände der Cluster zueinander neu berechnet werden und es werden abermals die

zwei Cluster mit dem geringsten Abstand gesucht und zusammengeführt. Dieses Prozedere wird solange wiederholt, bis nur noch ein Cluster übrig ist. Zur Bestimmung der Abstände zwischen Clustern gibt es verschiedene Möglichkeiten: [WZ01]

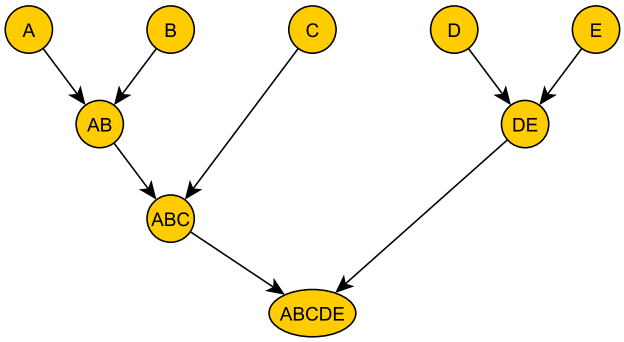

Abbildung 2.5: Beispiel eines agglomerativen Verfahrens

Single-Linkage Bei diesem Verfahren wird der minimale Abstand zweier Objekte aus je einem Cluster als Abstand der beiden Cluster zueinander definiert. Dadurch entstehen längliche Cluster die sich anhand eines Pfades fortsetzen.

Complete-Linkage Der Abstand zweier Cluster zueinander ist als maximaler Abstand zweier Objekte aus je einem Cluster definiert. Dies erzeugt Cluster mit möglichst geringem Durchmesser.

Average-Linkage Der Mittelwert der Abstände aller Objekte aus einem Cluster zu allen Objekten eines andern Cluster ist der Abstand zwischen den beiden Clustern.

Centroid-Method Der Abstand der Zentren zweier Cluster ist der Abstand der Cluster zueinander.

Divisives Verfahren

Divisive Verfahren arbeiten in umgekehrter Form zu den agglomerativen Verfahren. Zunächst bilden alle Objekte einen großen Cluster, der nach und nach in immer Kleinere aufgeteilt wird, bis jeder Cluster nur noch aus einem Objekt besteht.

2.4.3 Dichtebasierte Clusteranalyse

Bei der dichtebasierten Clusteranalyse weisen Cluster eine typische Dichte auf, die wesentlich höher ist als die der Zwischenräume [EKSX96]. Einer der bekanntesten Algorithmen dieser Kategorie ist der Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

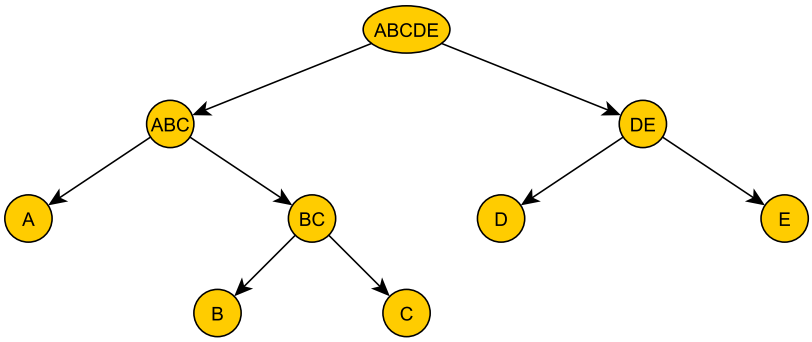

Abbildung 2.6: Beispiel eines divisiven Verfahrens

DBSCAN

DBSCAN wurde 1996 von Martin Ester, Hans-Peter Kriegel, Jörg Sander und Xiaowei Xu von der Universität München vorgestellt und erkennt Cluster und Rauschen in räumlichen Daten. Dazu werden zwei Parameter ***eps*** (maximaler Abstand zwischen zwei Punkten) und ***minPts*** (minimale Anzahl von *eps*-erreichbaren Punkten) festgelegt. Ein Punkt P1 ist von Punkt P2 erreichbar, wenn der Abstand zwischen beiden Punkten kleiner als *eps* ist. Gibt es mindestens *minPts* *eps*-erreichbare Punkte, so gilt der Punkt als dicht. Des Weiteren sind zwei Punkte dichte-verbunden, wenn es eine Kette von dichten Punkten gibt, die diese beiden miteinander verbindet. Ein Cluster besteht aus den Punkten, die durch dieselben dichten Punkte miteinander verbunden sind. Als Rauschen werden die Punkte bezeichnet, die nicht Teil eines Clusters sind. Sie werden getrennt erfasst und ausgegeben. [EKSX96]

2.5 Scalable Vector Graphics

Scalable Vector Graphics (SVG) ist eine 2001 veröffentlichte Spezifikation zur Erstellung von Vektorgrafiken. Sie basiert auf XML und ist somit ohne viel Aufwand von Programmen einlesbar und kann in einem einfachen Texteditor verändert oder erstellt werden. Im Gegensatz zu den gerasterten Grafiken wie GIF, JPG oder PNG verlieren SVG bei Größenveränderungen nicht an Qualität. Sie sind zudem hierarchisch aufgebaut und bietet mit dem Tag *<g />* die Möglichkeit Elemente zu gruppieren und Attribute, sowie Transformationen auf eine gesamte Gruppe anzuwenden. Als Elemente werden Kreise, Ellipsen, Rechtecke, Linen, Polygonzüge, Polygone, Text und Bilder unterstützt. Zusätzlich zu den Attributen der jeweiligen Elemente können Transformationen angegeben werden. Dabei stehen sechs Arten zur Verfügung: *matrix*, *translate*, *scale*, *rotate*, *skewX* und *skewY*. Jede dieser Transformationen kann auch als *matrix* ausgedrückt werden, denn die übrigen Typen sind nur Spezialfälle einer Matrixtransformation. [SVG01] [PMEB01]

2.5.1 Animationen

SVG erlaubt die Animation von Elementen und Gruppen, dazu werden Teile von der *Synchronized Multimedia Integration Language* (SMIL) [SMI08] übernommen. Es stehen zwei Möglichkeiten zur Animation zur Verfügung: Mit dem Attribut *animate* können einzelne Attribute und mit *animateTransform* können Transformationen über eine definierte Zeit verändert werden. Zur Definition des Zeitraums kann mit *begin* der Beginn der Animation und mit *dur* die Dauer festgelegt werden. Alternativ lässt sich auch mit *end* ein genauer Endzeitpunkt angeben. Zeitangaben werden immer relativ zum Zeitpunkt der Anzeige in Form von Millisekunden (ms), Sekunden (s) oder Stunden (h) getätigt. [SMI08]

3 Anforderung an das Analysetool

Die Anforderung an das Analysetool wird in Anlehnung an Sommerville [Som11] in zwei Kategorien eingeteilt: In der Anwenderanforderung wird der Bedarf des Anwenders beschrieben, es wird in einer nicht sehr detailreichen Weise die Funktion des Programms nach Außen hin festgelegt. Dabei wird zum größten Teil auf softwaretechnische Aspekte verzichtet. In der Systemanforderung werden die in der Anwenderanforderung spezifizierten Anforderungen näher beschrieben und durch technische Details ergänzt. Beide Kategorien lassen jeweils in die zwei Unterkategorien *funktionale Anforderung* und *nichtfunktionale Anforderung* aufteilen. Die funktionalen Anforderungen beschreiben das Verhalten eines Systems, sie legen die Art und Weise fest, auf die ein System auf Input reagiert und Daten verarbeitet. Nichtfunktionale Anforderungen oder auch Qualitätsanforderung, beschreiben Eigenschaften eines Systems, die sich nicht direkt in Funktionen ausdrücken lassen, darunter fallen zum Beispiel Zuverlässigkeit, Benutzbarkeit, Effizienz und Sicherheit. Diese Anforderungen können weitere funktionale Anforderungen generieren. Eine Sicherheitsanforderung wird immer weitere Funktionen benötigen, die diese Anforderung umsetzen.

3.1 Anwenderanforderung

Aus Sicht des Anwenders ergeben sich die folgenden Anforderungen an das Analysetool:

3.1.1 Funktionale Anforderungen

Laden Das Programm soll die Möglichkeit bieten, ein Bild, sowie AOI und Fixationen zu laden. Außerdem soll, um eine einheitliche Auswertung zu garantieren, die Konfiguration einer zuvor angelegten Auswertung geladen werden können.

Speichern Die Konfiguration einer Auswertung soll in eine eigene Datei gespeichert werden können, damit sie später weiter verwendet werden kann. Das Ergebnis der Analyse soll sowohl als Grafik, als auch als Textdatei zu speichern sein.

Stapelverarbeitung Eine Stapelverarbeitung für einen Satz von Fixationsdaten soll möglich sein. Das bedeutet, dass die Ergebnisse einer jeden Analyse separat gespeichert werden müssen.

Statische und dynamische AOI Das Programm soll die Möglichkeit bieten, statische und dynamische AOI zu analysieren. Dynamische AOI sollen Bereiche darstellen, die über einen definierten Zeitraum (Startzeitpunkt und Endzeitpunkt, oder Startzeitpunkt und Dauer) ein Attribut oder eine Transformation verändern. Als Transformation sollen Skalierungen, sowie Rotationen und Verschiebungen unterstützt werden.

AOI-Schwellenwert Um der Ungenauigkeit von erhobenen Daten eines Eyetracker entgegenzuwirken, soll ein Schwellenwert definiert werden können, der den maximalen Abstand zu einer AOI angibt, sodass ein Punkt, der noch innerhalb dieser Toleranz liegt, als zur AOI gehörig zu betrachten ist.

Erweiterbarkeit Es muss die Möglichkeit bestehen, weitere Auswertungsmethoden hinzuzufügen.

3.1.2 Nichtfunktionale Anforderungen

Die Bedienung des Programms soll über eine grafische Oberfläche möglich sein. Die Effizienz ist nachrangig, eine Analyse sollte bei großen Datenmenge jedoch nicht allzu viel Zeit in Anspruch nehmen. Es sollen, wenn möglich, existierende Standards für den Austausch von Daten verwendet werden, damit eine höhere Flexibilität garantiert ist. Das Programm soll plattformunabhängig arbeiten.

3.2 Systemanforderung

Die Systemanforderungen konkretisieren die Anwenderanforderung und reichern diese mit technischen Details an.

3.2.1 Funktionale Anforderungen

zu: Laden Das Auswertungstool muss das Laden der Bildformate JPEG, PNG, GIF und TIFF unterstützen. Die Fixationen werden in Form einer CSV-Datei mit dem Aufbau: [Timestamp, X-Koordinate, Y-Koordinate, Fixationsdauer] vorliegen. Als Dateiformat für AOI wird das SVG-Format dienen.

zu: Speichern Für die grafische Auswertung muss das Programm die grafischen Ergebnisse der einzelnen Auswertungsmethoden als separate Schichten vorhalten und diese beim Speichern, in definierter Reihenfolge, übereinander anordnen. Die textuellen Ergebnisse werden untereinander in eine Textdatei geschrieben.

zu: Stapelverarbeitung Die Analyse wird mit vorher festgelegten Parametern für jeden Datensatz durchgeführt. Nach jeder Analyse werden die Ergebnisse in einem, vom Benutzer anzugebenden Ordner, gespeichert. Danach werden die Ergebnisse gelöscht und mit dem nächsten Datensatz fortgefahren.

zu: Statische und dynamische AOI AOI werden in Form einer SVG-Datei erstellt und geladen. Dazu muss ein SVG-Parser geschrieben werden, der die Elemente *rect*, *ellipse*, *circle*, *polygon*, *path* und das Gruppen-Tag *<g>* auslesen und interpretieren kann. Alle weiteren Angaben, mit Ausnahme von *transform*, werden dabei ignoriert. Zu Realisierung von Animationen muss der Parser die von SVG aus SMIL übernommenen Attribute *animate* und *animateTransform* unterstützen.

zu: AOI-Schwellenwert Um den AOI-Schwellenwert zu realisieren, muss der minimalen Abstand eines Punktes zu einer AOI ermittelt werden können. Dazu wird die AOI als Polygon mit einer Abweichung von unter einem Pixel dargestellt und der minimale Abstand aller Linien zu diesem Punkt bestimmt.

zu: Erweiterbarkeit Das Programm wird modular aufgebaut und eine Schnittstelle für Auswertungsmethoden bieten. Diese Schnittstelle umfasst das Laden von Fixationen und AOI, sowie die Konfiguration der Auswertungsparameter. Zusätzlich wird es eine einheitliche Abfragemethode von Auswertungsergebnisse und Auswertungsgrafik geben.

3.2.2 Nichtfunktionale Anforderungen

Die grafische Benutzeroberfläche ist vom Analyseteil des Programms zu trennen. Die Kommunikation zwischen Programm und Benutzeroberfläche geschieht über definierte Methoden. Fehler sind intern abzufangen und auf geeignete Weise zu verarbeiten. Um eine Plattformunabhängigkeit zu erreichen empfiehlt sich die Verwendung der Java-Programmiersprache. Für die Datenhaltung sind geeignete Konstrukte in Hinblick auf die Laufzeit bei üblichen Operationen wie hinzufügen, löschen, abfragen, zu verwenden.

4 Architektur und Entwurf des Analysetools

Die Architektur und der Entwurf des Analysetools geschieht losgelöst von jeglicher Programmiersprache. Es wird das grundsätzliche Konzept des Programmes erläutert.

4.1 Übersicht

Aus der Anforderungsanalyse in Kapitel 3 können vier Phasen identifiziert werden. In **Phase I** werden alle benötigten Daten geladen, dazu gehören die Fixationen, die AOI, das als Stimulus verwendete Bild, sowie eine Konfiguration zur Auswertung. Bis auf die Fixationen sind alle übrigen Dateien optional. **Phase II** beinhaltet die Konfiguration der

Abbildung 4.1: Einteilung des Tools in vier Phasen

Auswertungsmethoden. Wurde zuvor eine Konfigurations-Datei geladen, so kann diese angepasst werden. Die Auswertung geschieht in **Phase III** und teilt sich in die zwei Bereiche grafische und textuelle Auswertung auf. In **Phase IV** werden die Ergebnisse gespeichert, dabei kann zwischen textuellen, grafischen oder beiden Ergebnissen gewählt werden.

4.1.1 Prozessablauf

Der Ablauf des Prozesses lässt sich mit einem Business Process Model and Notation (BPMN) Diagramm (siehe Abbildung 4.4) abbilden. Dies ist zwar hauptsächlich für die Modellierung von Geschäftsprozessen gedacht, eignet sich jedoch auch für jegliche andere Abbildung von Prozessen. Zu Beginn muss der Benutzer die Fixationsdaten laden, dies ist notwendig um die Dauer der Auswertung festzulegen. Als Nächstes können weitere Daten (Bild, AOI, Auswertungs-Konfiguration) geladen werden. Alle Schritte bis hierher gehören zu Phase I der Abbildung 4.1. Im nächsten Schritt werden die Parameter der Auswertung festgelegt oder angepasst (Phase II). Nun startet die Auswertung (Phase III) gefolgt von der Anzeige der Ergebnisse. Als Nächstes können die Ergebnisse entweder

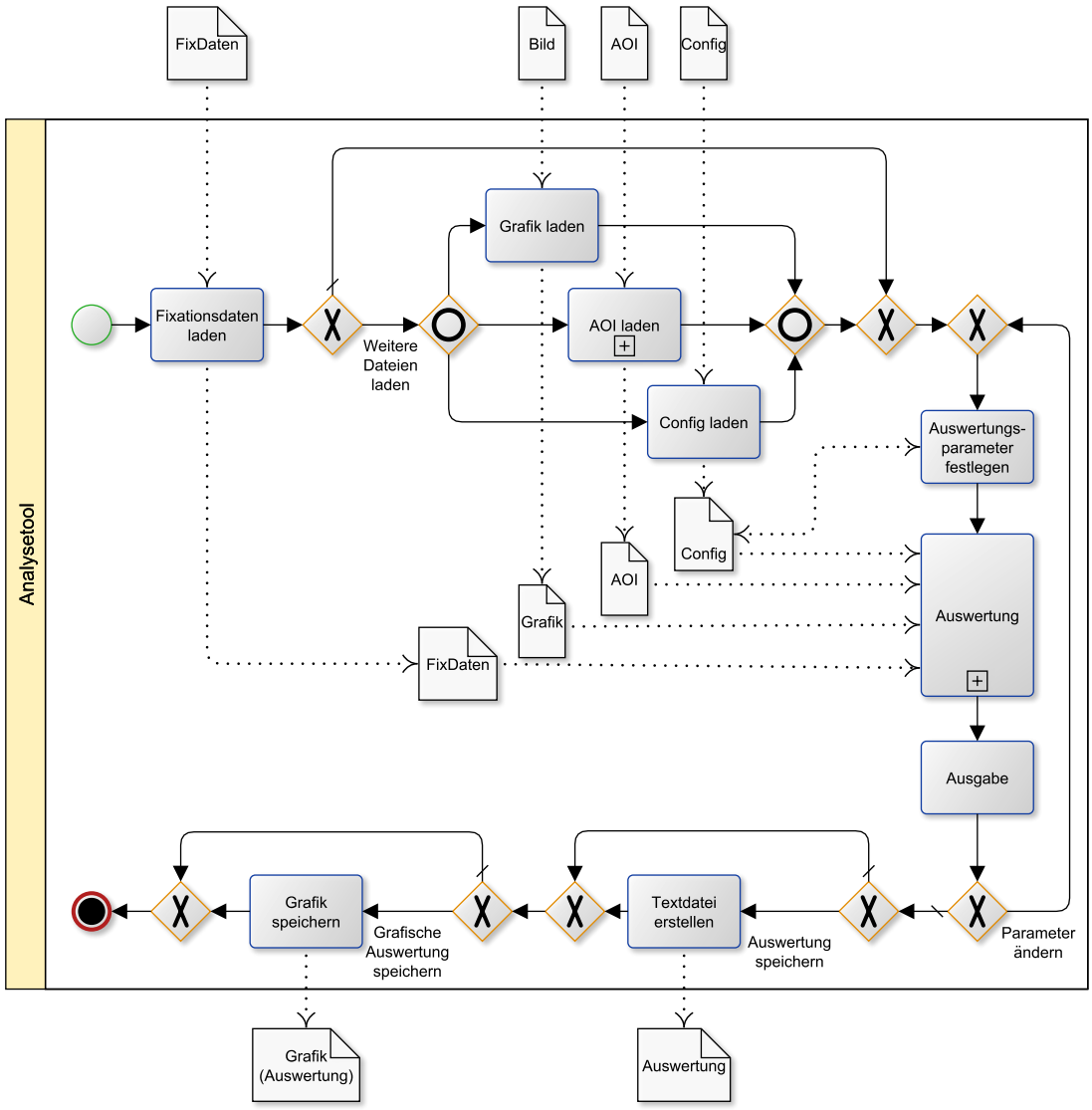

Abbildung 4.2: BPMN des Auswertungsprozesses

gespeichert werden, oder die Konfiguration wird angepasst und die Auswertung beginnt abermals. Angekommen in Phase IV besteht die Möglichkeit die Ergebnisse in Form einer Text- und/oder Bilddatei zu speichern. Damit ist der Auswertungsprozess beendet.

4.2 Komponenten

Das Programm besteht aus drei Hauptkomponenten: Der **Analyzer** beinhaltet die Daten und Auswertungsmethoden, der **AOIParser** liest die SVG-Dateien ein und erstellt daraus die AOI und die **AnalyzerGUI** stellt eine grafische Benutzeroberfläche bereit.

4.2.1 Analyzer

Dies ist die Kernkomponente des Analysetools, sie beinhaltet die Fixationsdaten, sowie die AOI und die Auswertungsmethoden. Über den Analyzer werden Einstellungen vorgenommen und die Auswertung gestartet. Zusätzlich beinhaltet diese Klasse den ImageContainer mit der grafischen Auswertung. Es lassen sich die einzelnen Ebenen der Grafik frei anordnen und mit Transparenz versehen. Analysemethoden werden über das *IAnalysisMethod*-Interface hinzugefügt.

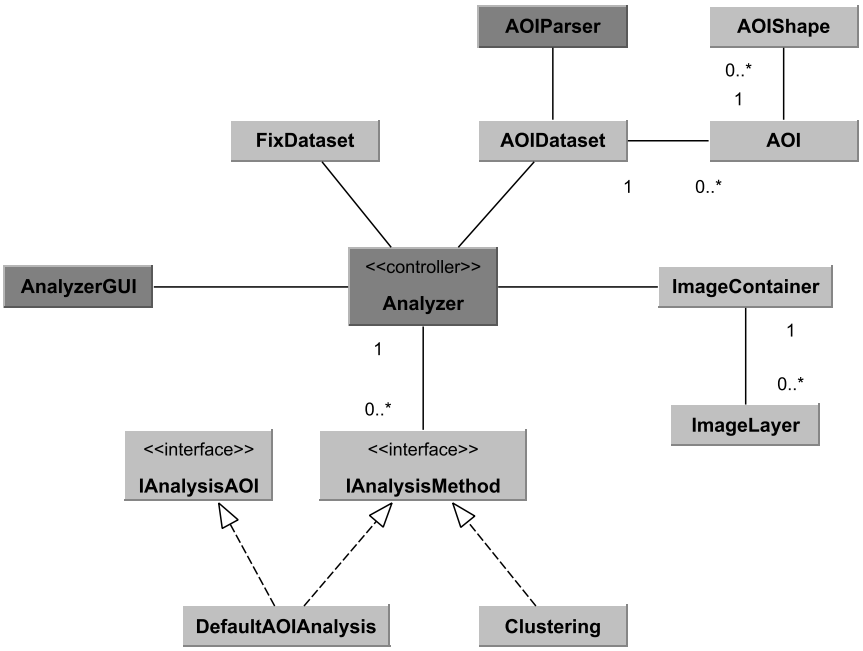

Abbildung 4.3: Übersicht Analyzer

AOI

An die AOI werden zwei wesentliche Anforderungen gestellt: Sie muss eine geometrisch Form (Rechteck, Ellipse, Polygon, Pfad) bereitstellen und angeben können, ob sich ein Punkt zu einem bestimmten Zeitpunkt innerhalb dieses Bereiches befindet. Zudem soll ein Schwellenwert angegeben werden können, der den Bereich der AOI erweitert. Animationen werden in statische AOI transformiert, sodass der Berechnungsaufwand während der Analyse reduziert wird. Die Klasse **AOI** beinhaltet eine Identifikation und eine Liste mit

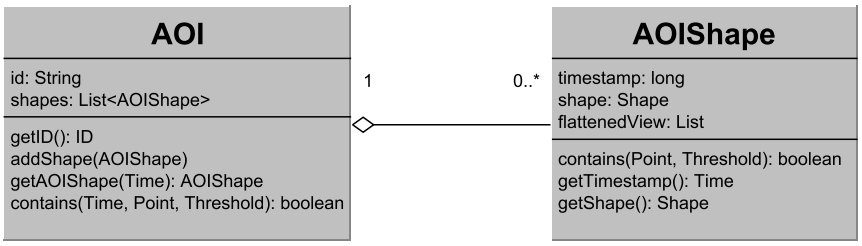

Abbildung 4.4: AOI

AOIShapes. Über die Methode *contains* wird nach der entsprechenden AOIShape gesucht und anschließend geprüft, ob ein Punkt innerhalb liegt. Die Klasse **AOIShape** beinhalten die geometrische Form, sowie den Zeitpunkt, an dem diese Form vorliegt. Beim Erstellen wird zudem eine abgeflachte Repräsentation dieser Form als Liste von Linien gespeichert. Diese wird zur Bestimmung der minimalen Distanz der AOI zu einem Punkt benötigt. Ein **AOIDataset** beinhaltet alle eingelesenen AOI und die Dauer der Animationen. Über die Methode *loadAOIFile* wird eine SVG-Datei geladen, die vom AOIParser analysiert wird.

ImageContainer

Der **ImageConteiner** beinhaltet die grafische Auswertung der ausgewählten Analysemethoden. Er besteht aus einer Liste von ImageLayer, die in ihrer Position verändert werden können. Zusätzlich verfügt diese Klasse über eine Basisebene, in der der Stimulus hinterlegt wird. Der **ImageLayer** ist eine einfache Klasse, die das Bild der Auswertung beinhaltet und zusätzlich über eine Angabe zur Sichtbarkeit (true, false) und Transparenz (0.0 - 1.0) verfügt.

FixDataset

Das **FixDataset** lädt die Fixationen aus einer CSV-Datei und vergibt, auf Basis der ersten Fixation, einen neuen Zeitstempel. Es besteht die Möglichkeit alle Fixationen, oder die eines gewählten Datensatzes als sortierte Liste auszugeben. Des Weiteren wird die Zeitspanne, in der die Daten vorliegen, als Gesamtdauer geliefert. Diese Dauer wird vor allem für die Berechnung der dynamischen AOI benötigt.

IAnalysisMethod

Eine Forderung an das Analysetool ist die Erweiterbarkeit: Neue Auswertungsmethoden sollen auf einfache Weise hinzugefügt werden können. Durch das **IAnalysisMethod**-Interface wird diese Möglichkeit geboten. Es schreibt grundlegende Funktionen vor, die jede Auswertungsmethode bieten muss:

- FixDataset setzen
- Konfiguration setzen/liefern
- Auswertung starten
- Auswertung für bestimmten Datensatz starten
- Ergebnis liefern
- Grafisches Ergebnis liefern

IAnalysisAOI

Zusätzlich zu den Grundfunktionen muss eine Methode, die AOI verwendet, das **IAnalysisAOI**-Interface implementieren. Dieses fordert eine Methode zum setzen eines AOIDataset.

4.2.2 AOIParser

Der AOIParser liest eine SVG-Datei ein und untersucht diese auf die Elemente *rect*, *ellipse*, *circle*, *polygon*, *path*, sowie deren Transformationsanweisung. Die Transformation von Gruppen werden auf die Unterelemente angewandt. Animationen werden in statische AOI übersetzt, indem der AOI ein neues AOIShape hinzugefügt wird, sobald sich ein Attribut um mehr als einen Pixel ändert. Aufgrund der pixelbasierten Auswertungen müssen Änderungen von unter einem Pixel nicht berücksichtigt werden.

AOIAnimation

Eine besondere Herausforderung stellt die Umsetzung der dynamischen AOI dar, diese muss zu einer bestimmten Zeit prüfen können, ob sich eine Fixation in ihrem Bereich befindet. Aufgrund der hohen Anzahl von Fixationen, die bei einer Auswertung anfallen können, kann diese Bestimmung der Zugehörigkeit nicht erst in der Auswertungsphase berechnet werden. Aus diesem Grund wird die animierte SVG-Datei beim Einlesen vom AOIParser auf Animationen untersucht und von der Klasse **AOIAnimation** in eine Liste von Transformationsanweisungen über die Dauer umgewandelt. Diese prüft zunächst die Art der Animation und unterscheidet zwei Fälle: Handelt es sich bei der Animation um eine Skalierung, so wird in 40 ms Schritten der Skalierungsfaktor bestimmt. Dies entspricht einer Framerate von 25 Bildern in der Sekunde. Jede andere Animation wird mit einer Schrittgröße von 1 Pixel (Rotation: 1°) und dem Zeitpunkt, an dem diese Änderung auftritt, zerlegt.

4.2.3 AnalyzerGUI

Die AnalyzerGUI ist der grafische Aufsatz des Analyzer, sie stellt dem Benutzer Methoden zum Laden der einzelnen Dateiformate zur Verfügung. Zudem lassen sich über sie die für die Analyse notwendigen Parameter setzen.

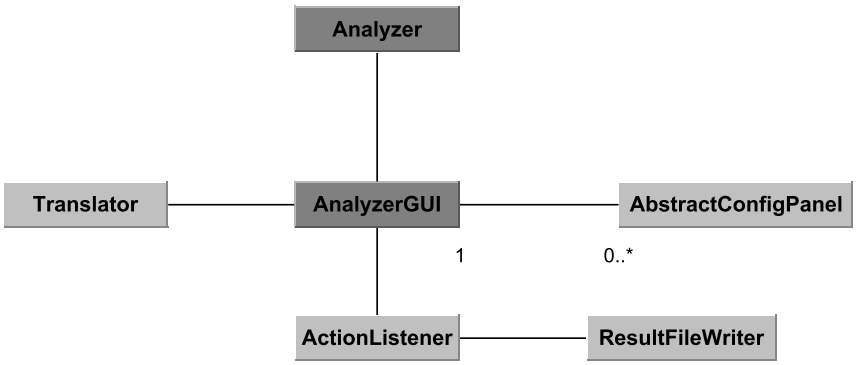

Abbildung 4.5: Übersicht AnalyzerGUI

ActionListener

Benutzeraktionen sollen von der grafischen Oberfläche entkoppelt werden, dadurch wird ein **ActionListener** unverzichtbar. Im Listener werden Eingaben des Benutzers ausgewertet und lösen entsprechende Aktionen im Analyzer und/oder der GUI aus.

ResultFileWriter

Die textuellen und grafischen Ergebnisse der Auswertung müssen abgespeichert werden können. Der **ResultFileWriter** bietet dazu die Möglichkeit entweder den Text, das Bild oder Beides in ein ausgewählten Verzeichnis zu schreiben. Zusätzlich verfügt er über die Möglichkeit in einem Stapelverarbeitungsverfahren die Ergebnisse aller Datensätze im Programm zu speichern. Dazu wird die Auswertung mit einer vorher festgelegten Konfiguration auf einen Datensatz angewendet und das Ergebnis gespeichert. Dies wird so lange wiederholt, bis alle Datensätze abgearbeitet wurden.

AbstractConfigPanel

Eine Methode muss ein ConfigPanel vorhalten, um in der Benutzeroberfläche konfiguriert werden zu können. Das **AbstractConfigPanel** bietet die dazu nötigen Funktionen an, die unter anderem die manuelle Auslösung des Auswertungsprozesses und die Aktualisierung der Werte übernehmen.

Translator

Zur Realisierung von Mehrsprachigkeit müssen Bezeichnungen ausgelagert werden. Der **Translator** hält die einzelnen Sprachdateien vor und liefert über eine Methode die jeweilige Übersetzung zu der gewählten Sprache.

5 Implementierung des Analysetools

Die Implementierung des Analysetools erfolgt in der Programmiersprache Java. Aufgrund der Verwendung von *Strings* in *Switch-Case*-Verzweigungen wird mindestens Java in Version 1.7 vorausgesetzt. Das EyeMotion-Framework wurde nur geringfügig angepasst. Die Änderungen betreffen lediglich die Klasse *Fixation* und umfassen einen neuen Konstruktor, der eine Fixation mit den Parametern: *start, duration, x, y* erstellt und eine neue Methode, die anstatt die X- und Y-Koordinaten einzeln zu liefern, einen *Point* zurückgibt. Weitere Änderungen an dem Framework wurden nicht vorgenommen.

5.1 Verwendete Bibliotheken/Klassen

Bei der Implementierung des Analysetools wurden, neben dem EyeMotion-Framework, zwei externe Bibliotheken und eine externe Klasse verwendet. Diese sind öffentlich zugänglich und erlauben die Verwendung in eigenen Projekten.

ApacheTM Batik SVG Toolkit

Für den AOIParser wurde auf Teile des Batik SVG Toolkit zurückgegriffen. Diese Bibliotheken bieten einen großen Funktionsumfang im Bereich der Scalable Vector Graphics. Die verwendeten Teile umfassen die Parser für *Transform*, *Path*, *Point* und *Number*.

Bart Kiers GrahamScan von 2010

Zur Ermittlung der konvexen Hülle in Punktwolken wurde der 1972 von Ronald L. Graham vorgestellte [Gra72] Graham Scan Algorithmus verwendet. Die verwendete Java-Klasse wurde 2010 von Bart Kiers implementiert.

opencsv

Die Java-Bibliothek opencsv unterliegt der Apache 2.0 Lizenz und erleichtert den Umgang mit Comma-Seperated-Values, sie bietet die Möglichkeit CSV-Dateien zu laden und zu speichern. Dabei kann das als Werte-Trenner verwendete Zeichen frei gewählt werden. Für das Analysetool wurde das Kommata als Trennzeichen festgelegt.

5.2 Auswertungsmethoden

Alle Auswertungsmethoden müssen das *IAnalysisMethod*-Interface (siehe Tabelle 5.1) und falls AOI verwendet werden sollen, das *IAnalysisAOI*-Interface implementieren. Das zweite Interface fordert eine Methode *setAOI(AOIDataset)*, die der Auswertungsmethode die zu verwendenden AOI hinzufügt. Intern wird es somit durch den Aufruf von *instanceof* ermöglicht, die Auswertungsmethoden unterschiedlich zu behandeln. Zusätzlich zu den in

Tabelle 5.1: *IAnalysisMethod*-Interface

Methode	**Erklärung**
setData(FixDataset)	Übergebe der Methode die Fixationen.
setSize(int, int)	Setze die Größe (Breite, Höhe) der Auswertungsgrafik.
setConfig(String, String)	Setze die Konfiguration (Schlüssel, Wert) der Auswertung.
getConfig()	Liefere die Konfiguration als HashMap.
analyse()	Starte die Analyse mit allen Fixationsdaten.
analyse(String)	Starte die Analyse für einen bestimmten Datensatz.
getResult()	Liefere das Ergebnis der Auswertung.
getImageResult()	Liefere das grafische Ergebnis der Auswertung.

den zwei Interface geforderten Methoden, kann jede Auswertungsmethode ihre eigenen Methoden implementieren. Darunter fallen zum Beispiel die Konfiguration der jeweiligen Parameter. Um die geforderten Funktionen (dynamisch und statische AOI, sowie AOI-Erkennung) zu gewährleisten, wurden zwei Auswertungsmethoden *DefaultAOIAnalysis* und *Clustering* implementiert. Zwei weitere Methoden *DrawFixations* und *DrawAOI* dienen der Visualisierung.

5.2.1 DrawFixations

Diese einfache Auswertungsmethode zeichnet die Fixationen in die Grafik. Es lassen sich die Punktgröße und Punktfarbe einstellen. In der privaten Methode *draw* wird die Liste der Fixationen mit einer for-Schleife durchlaufen und der jeweilige Punkt in ein *Graphics2D*-Objekt geschrieben. Es wird nur eine grafische Auswertung zurückgegeben. Die Methode *getResult()* liefert hingegen ein *null*.

Tabelle 5.2: DrawFixations

Parameter	**Standardwert**	**Getter**	**Setter**
Punktgröße	4 Pixel	getDotSize()	setDotSize(int)
Punktfarbe	Rot	getColor()	setColor(Color)

5.2.2 DrawAOI

Diese Klasse dient, genau so wie DrawFixations, lediglich der Visualisierung. Es können die eingelesenen AOI zu einem bestimmten Zeitpunkt angezeigt werden. Dazu benötigt

DrawAOI die Parameter Farbe, Linienstärke und Zeitpunkt. Diese Klasse liefert abermals nur ein grafisches Ergebnis.

Tabelle 5.3: DrawAOI

Parameter	Standardwert	Getter	Setter
Farbe	4 Rot	getColor()	setColor(Color)
Linienstärke	1 Pixel	getStrokeWidth()	setStrokeWidth(int)
Zeitpunkt	0 ms	getTime()	setTime(long)

5.2.3 DefaultAOIAnalysis

Die Standardauswertung von AOI erfolgt mit Hilfe der in Abschnitt 2.3.1 vorgestellten Transitionsmatrix. Dazu wird als erstes eine $n \times n$-Matrix erstellt, wobei n die Anzahl der AOI repräsentiert. Zusätzlich wird eine TreeMap der Form *<Long, Tuple<AOI, Long> >* für die Transitionen vorgehalten. In ihr wird der Zeitpunkt, an dem in die jeweilige AOI gewechselt wurde und die Dauer der Fixationen notiert. Nun wird die Liste der Fixationen

Tabelle 5.4: DefaultAOIAnalysis

Parameter	Standardwert	Getter	Setter
Schwellenwert	0 Pixel	getThreshold()	setThreshold(int)

durchlaufen und für jede AOI geprüft, ob die jeweilige Fixation innerhalb liegt. Ist dies der Fall müssen zwei Fälle unterschieden werden: In **Fall 1** lag die vorherige Fixation bereits in dieser AOI und somit muss die Dauer der Fixationen um die Fixationsdauer der jeweiligen Fixation erhöht werden. Es erfolgt kein Eintrag in die Transitionsmatrix. In **Fall 2** liegt ein Wechsel der Fixation vor, es muss somit ein Eintrag in die Transitionsmatrix erfolgen. Die Dauer der Fixationen beginnt nun bei der Dauer der jeweiligen Fixation.

```
1  long begin = -1;
2  long duration = 0;
3  int aoiIndex = -1;
4  for(Fixation fix : this.fixList) {
5    int currentIndex = -1;
6    for(int i = 0; i < aoiList.size(); i++) {
7      AOI aoi = aoiList.get(i);
8      if(aoi.contains(fix.getTimestamp(), fix.getPoint(), this.threshold)) {
9        currentIndex = i;
10     }
11   }
12   if(currentIndex != -1) {
13     if(aoiIndex != -1) {
14       //switch between two AOI
15       if(aoiIndex != currentIndex) {
16         int x = aoiIndex;
17         int y = currentIndex;
```

```
18
19          matrix[x][y]++;
20
21          AOI aoi = aoiList.get(aoiIndex);
22          Tuple<AOI, Long> value = new Tuple<AOI, Long>(aoi, duration);
23          map.put(begin, value);
24
25          begin = fix.getTimestamp();
26          duration = fix.getDuration();
27        } else {
28        //no switch
29          duration += fix.getDuration();
30        }
31      } else {
32        begin = fix.getTimestamp();
33        duration = fix.getDuration();
34      }
35      aoiIndex = currentIndex;
36    }
37  }
```

Sobald alle Fixationen durchlaufen wurde, wird überprüft, ob die letzte Fixation auf einer AOI lag, ist dies der Fall, so wird die bisherige Dauer in die TreeMap übernommen. Als Nächstes werden die Daten in mehreren Schritten, auf die nicht weiter eingegangen werden soll, aufbereitet und die Ergebnisse in einer HashMap gespeichert. Im letzten Schritt wird aus diesen Daten ein Bild generiert, dass die Transitionen durch Linien darstellt.

5.2.4 Clustering

Die Clusteranalyse ermöglicht die Einordnung von Punkten in Cluster. Dadurch lassen sich AOI bestimmen und eine Heatmap erzeugen. Es wurde eine hierarchische Clusteranalyse mit agglomerativen Verfahren implementiert. Die Parameter *Abweichung* und *Min.# Fixationen* haben nur Auswirkung auf die Bestimmung der AOI. *Abweichung* gibt die

Tabelle 5.5: Clustering

Parameter	**Standardwert**	**Getter**	**Setter**
Distanzmethode	SINGLE LINKAGE	getDistanceMethod()	setDistanceMethod (IClusterDistance)
Zeige Heatmap	true	getShowHeatMap()	setShowHeatMap (boolean)
Zeige AOI	false	getShowAOI()	setShowAOI(boolean)
Abweichung	50 Pixel	getDeviation()	setDeviation(double)
Min.# Fixationen	10	getMinFixNumber()	setMinFixNumber(int)
Heatmap-Werte	{0.70, 0.50, 0.45, 0.40, 0.35, 0.30, 0.25}	getHeatMapValues()	setHeatMapValues (Double[])

maximale Standardabweichung innerhalb eines Clusters an. Damit kleine Cluster aussortiert werden können, muss die Anzahl der minimalen Fixationen in einem Cluster angegeben werden. Zur Distanzberechnung stehen alle in Abschnitt 2.4.2 aufgeführten Methoden zur Verfügung. Zu Beginn werden alle Fixationen in eigene Cluster verschoben. Als Nächstes

Tabelle 5.6: Distanzmatrix

Cluster	**1**	**2**	**3**	**4**	**5**
1	0	x	x	x	x
2		0	x	x	x
3			0	x	x
4				0	x
5					0

wird die Distanzmatrix der Cluster zueinander berechnet, hierbei wird aus Performance-Gründen nur die halbe Matrix berechnet. Die Distanz von (1,2) entspricht der Distanz von (2,1). Die Matrix wird als *LinkedList<LinkedList<Double> >* aufgebaut. Der Vorteil liegt in der guten Laufzeit beim Hinzufügen und Löschen von Elementen. Während der Berechnung werden Cluster aus dieser Matrix herausgenommen und als neues Cluster ans Ende angefügt. Es muss darauf geachtet werden, dass die Listen mit einem Iterator durchlaufen werden, ansonsten wird für jeden Wert am Anfang der Liste begonnen und die Laufzeit verlängert sich erheblich.

```
1  private int[] findMinDistance(LinkedList<LinkedList<Double>>
       distanceMatrix) {
2    double min = Double.MAX_VALUE;
3    int indexA = 0;
4    int indexB = 0;
5
6    Iterator<LinkedList<Double>> xIter = distanceMatrix.iterator();
7    int i = 0;
8    while(xIter.hasNext()) {
9      LinkedList<Double> subList = xIter.next();
10     Iterator<Double> yIter = subList.iterator();
11     int j = 0;
12     while(yIter.hasNext()) {
13       double distance = yIter.next();
14       if(distance < min && i != j) {
15         min = distance;
16         indexA = i;
17         indexB = j;
18       }
19       j++;
20     }
21   i++;
22   }
23
24   int[] index = new int[2];
```

```
25    if(indexA >= indexB) {
26      index[0] = indexA;
27      index[1] = indexB;
28    } else {
29      index[0] = indexB;
30      index[1] = indexA;
31    }
32    return index;
33  }
```

Es werden solange zwei Cluster zusammengeführt, bis nur noch ein Cluster übrig ist. Zur Bestimmung der minimalen Distanz wird die Methode *findMinDistance* aufgerufen, sie liefert die Indizes der beiden Cluster, die den minimalen Abstand zueinander haben. Die entsprechenden Elemente werden nun aus der Distanzmatrix entfernt, zu einem Cluster verschmolzen und dieses ans Ende der Distanzmatrix hinzugefügt. Nun werden die Abstände des neuen Cluster zu den anderen berechnet und die Schleife beginnt von vorne.

5.3 Benutzeroberfläche

Die Benutzeroberfläche gliedert sich in drei Bereiche: **Links** befindet sich die Auswahlbox der Datensätze, sowie die Liste der ausgewählten Auswertungsmethoden. Über dieser Liste liegt ein Regler, der die Transparenz der einzelnen Schichten einstellt. Mit der Maus können diese Schichten in ihrer Reihenfolge verschoben werden. Ein Doppelklick mit der Maus blendet eine Schicht aus. Der **mittlere** Bereich dient der Darstellung der grafischen und textuellen Auswertung. Diese können durch den Reiter am oberen Rand umgeschaltet werden. Die Größe der Grafik passt sich dem verfügbaren Platz an, überschreitet jedoch niemals die angegebene Größe. Im **rechten** Bereich ist die Konfiguration der einzelnen Auswertungsmethoden zu finden. Durch die Auswahl der Methode in der linken Liste wird das entsprechende *ConfigPanel* der Methode eingeblendet. Über den Button „Anwenden“ werden die Einstellungen übernommen und die Auswertung für die jeweilige Methode gestartet. Das Ergebnis ist im Anschluss im mittleren Bereich zu sehen.

Im oberen Bereich befindet sich die *Menubar*. Über sie lassen sich Bilder, Fixationen und AOI laden, sowie die benötigten Auswertungsmethoden auswählen. Zusätzlich kann hier die Konfiguration gespeichert oder geladen, Ergebnisse gesichert und die Sprache umgestellt werden.

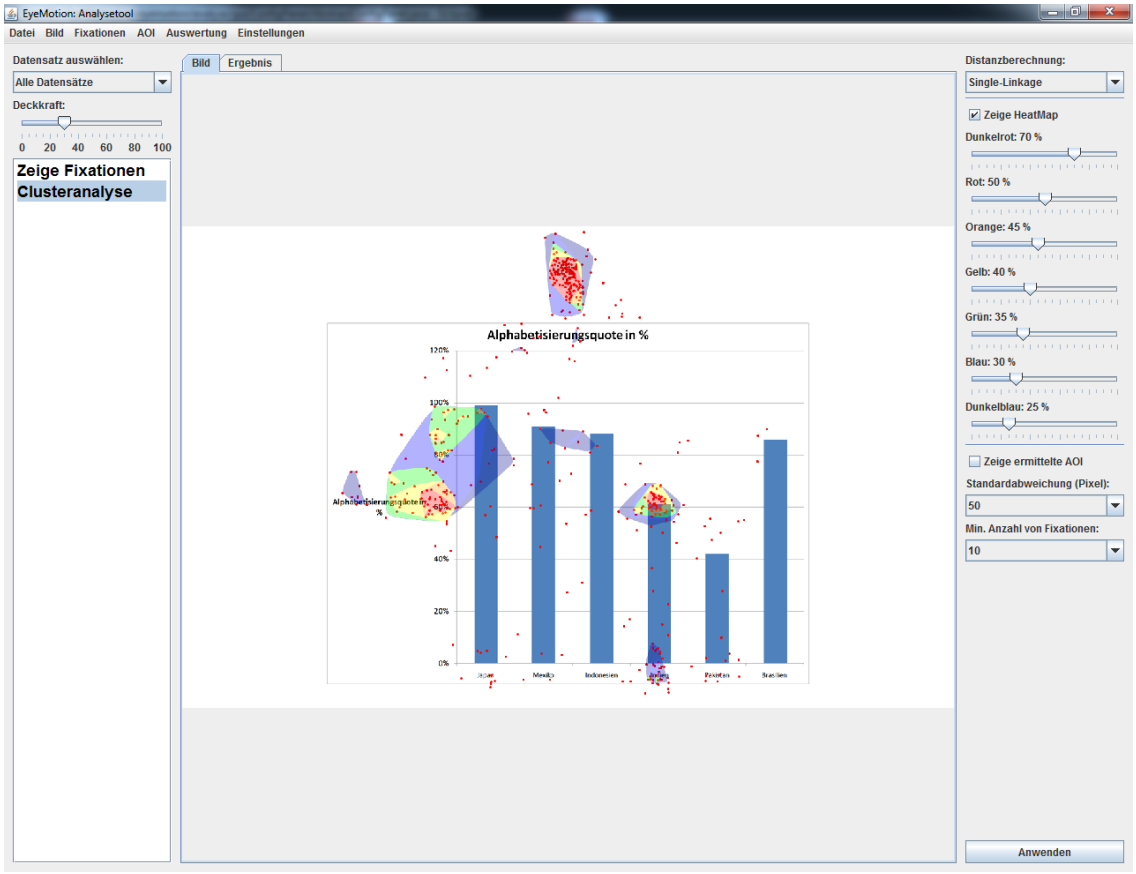

Abbildung 5.1: Benutzeroberfläche

6 Proof of Concept: Anwendung des Analysetools

Diese Kapitel soll die Funktion des Analysetools anhand von drei Auswertungsfällen, die sich im Kapitel 3 ergeben haben, demonstrieren. Dazu werden zuvor festgelegte statische und dynamischen AOI mit der implementieren Methode ausgewertet und durch eine Clusteranalyse relevante AOI bestimmt.

6.1 Auswertung von statischen AOI

Zur Auswertung von statischen AOI wurde auf die Daten der EyeMotion-Projektgruppe zurückgegriffen. Die verwendeten Daten gehören zu dem Datensatz: *Alphabetisierungsquote als Säulendiagramm*. 21 Probanden mussten in einer Infografik die Antwort auf die Frage: *„Welches Land hat eine Alphabetisierungsquote von 61%?“* finden. Der erfasste Bereich betrug 1280x800 Pixel und an der Position (590,30) lag ein Synchronisationspunkt mit einem Radius von 50 Pixel. [GJLR15]

Als Erstes wurden zehn AOI definiert, die den oberen Bereich jeder Säule, sowie den Synchronisationspunkt, den Titel und die Achsbeschriftungen abdeckten. Danach wurden

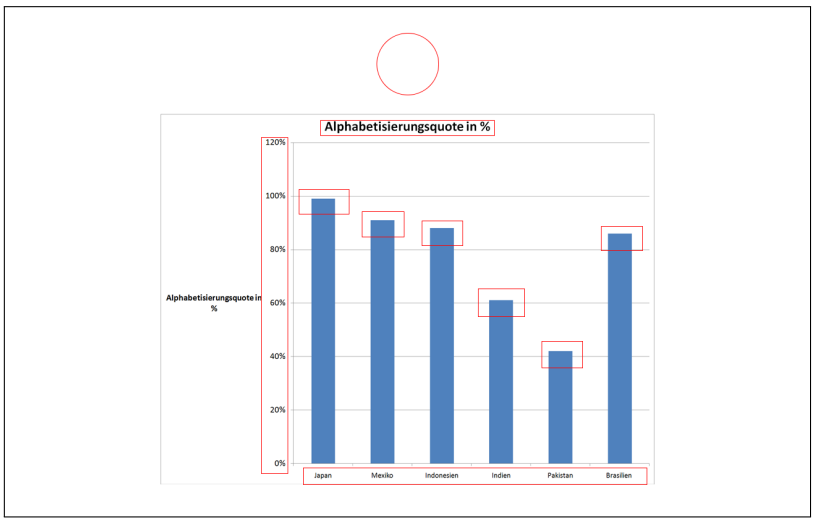

Abbildung 6.1: AOI

die Fixationsdaten in jeweils eine Datei pro Proband kopiert und die Analyse mit einem Threshold von 10 Pixel gestartet.

6.1.1 Ergebnis

Im Folgenden wird das Ergebnis der Auswertung für einen Probanden näher betrachtet. Das Analysetool liefert die Dauer der Fixationen jeder AOI in Millisekunden (siehe Tabelle 6.1), die Reihenfolge der betrachteten AOI (siehe Tabelle 6.2), sowie die Transitionsmatrix (siehe Tabelle 6.3) und die Transitionsdichte. Die grafische Auswertung liefert die Transitionen zwischen den AOI als Linien, zusätzlich wird die Reihenfolge in die AOI geschrieben Anhand der Abbildung 6.2 und der Tabelle 6.2 lässt sich ablesen, dass der Blick vom

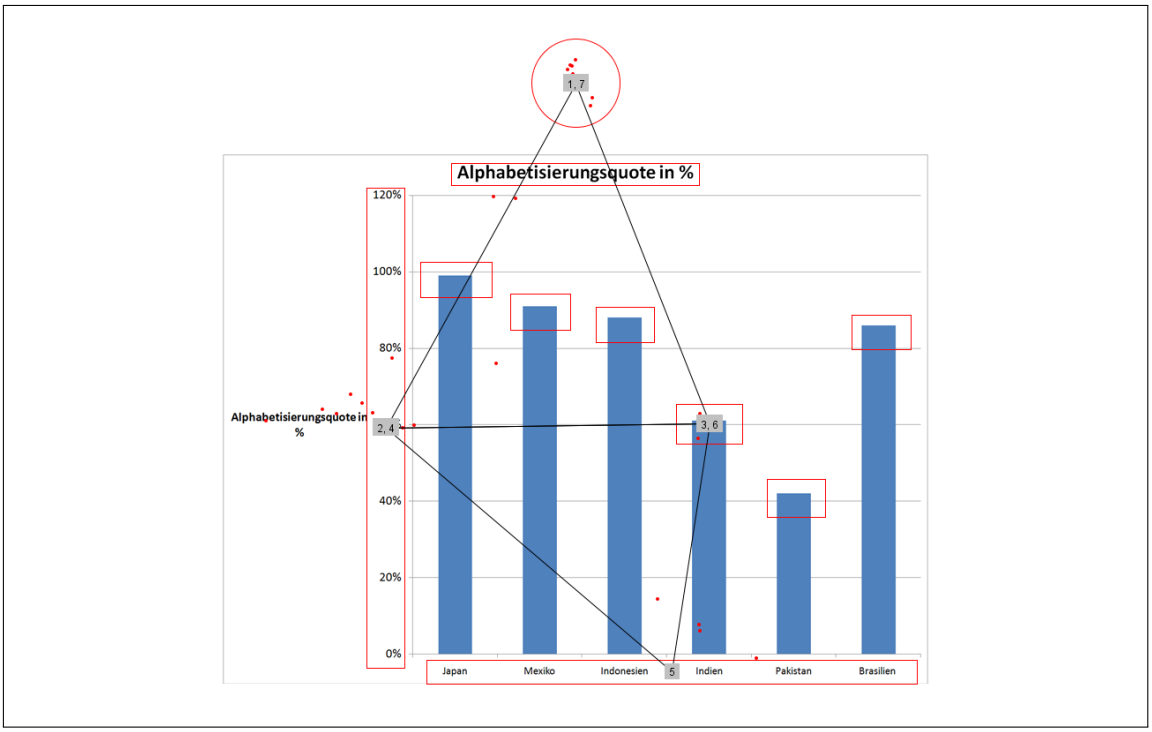

Abbildung 6.2: AOI Analyse

Synchronisationspunkt über die Y-Achsenbeschriftung hin zur richtigen Säule „Indien“ führt, gefolgt von einem Kontrollblick zur Y-Achse und zurück zu Indien. Als Nächstes geht der Blick zur X-Achsenbeschriftung an der die Lösung „Indien“ zu finden ist. Auf dem Weg zurück zum Synchronisationspunkt wird noch einmal die Höhe der Säule „Indien“ kontrolliert. Die Transitionsdichte beträgt 0.06 und lässt auf eine effiziente Suche schließen. Der Tabelle 6.1 ist zu entnehmen, dass die Suche sich auf die für die Antwort relevanten AOI „Indien“, „BeschriftungY“ und „BeschriftungX“ beschränkt. Die Betrachtung der AOI „SyncPoint“ hat mit dem Aufbau des Experiment zu tun und kann in diesem Fall vernachlässigt werden.

Tabelle 6.1: Fixationsdauer

	Name	**Dauer**
AOI0	SyncPoint	2317 ms
AOI1	Indien	566 ms
AOI2	Japan	0 ms
AOI3	Mexiko	0 ms
AOI4	Indonesien	0 ms
AOI5	Pakistan	0 ms
AOI6	Braslien	0 ms
AOI7	BeschfriftungX	60 ms
AOI8	Titel	0 ms
AOI9	BeschriftungY	990 ms

Tabelle 6.2: Reihenfolge

Zeitpunkt	**Dauer**	**AOI**
0 ms	603 ms	SyncPoint
1905 ms	778 ms	BeschriftungY
3814 ms	178 ms	Indien
4050 ms	212 ms	BeschriftungY
4501 ms	60 ms	BeschriftungX
4787 ms	388 ms	Indien
5265 ms	1714 ms	SyncPoint

Tabelle 6.3: Transitionsmatrix

	AOI0	AOI1	AOI2	AOI3	AOI4	AOI5	AOI6	AOI7	AOI8	AOI9
AOI0	0	**1**	0	0	0	0	0	0	0	0
AOI1	0	0	0	0	0	0	0	**1**	0	**1**
AOI2	0	0	0	0	0	0	0	0	0	0
AOI3	0	0	0	0	0	0	0	0	0	0
AOI4	0	0	0	0	0	0	0	0	0	0
AOI5	0	0	0	0	0	0	0	0	0	0
AOI6	0	0	0	0	0	0	0	0	0	0
AOI7	0	0	0	0	0	0	0	0	0	**1**
AOI8	0	0	0	0	0	0	0	0	0	0
AOI9	**1**	**1**	0	0	0	0	0	0	0	0

6.2 Auswertung von dynamischen AOI

Für die Auswertung der dynamischen AOI konnten die aus dem EyeMotion-Projekt erhobenen Daten nicht verwendet werden. Aus diesem Grund wurde ein eigener Versuch entwickelt und ausgewertet. Unter Verwendung des EyeMotion-Framework wurde ein simpler Versuchsaufbau realisiert. Es wurde eine animierte SVG-Datei mit der Größe 800x600 Pixel erstellt, die aus zwei Punkte mit einem Radius von 40 Pixel (SyncPoint, Ablenkung) und einen Punkt mit einem Radius von 50 Pixel (Ball) bestand. Alle Punkte waren rot eingefärbt. Die Animation hatte eine Dauer von 20 Sekunden und verschob in dieser Zeit den Ball vom linken zum rechten Rand der Grafik. Die Start- und Endposition des Balls lag jeweils außerhalb des sichtbaren Bereichs. Nach acht Sekunden wurde der Punkt mit der Bezeichnung *Ablenkung* für genau acht Sekunden eingeblendet und sollte den Blick der Probanden auf sich lenken. Ziel dieses Punktes war es zu Prüfen, ob sich die Probanden ablenken ließen, wenn dies der Fall war, sollte die Dauer der Ablenkung erfasst

werden. Zur Anzeige der Grafik diente ein handelsüblicher Browser im Vollbildmodus, dieser startete, durch Aktualisierung (Taste: F5), die Animation. Die Taste F5 startete gleichzeitig die Aufnahme der Eyetracking-Daten. Zur Erkennung der Fixation diente ein I-DT (siehe Abschnitt 2.2.1) mit einer minimalen Fixationsdauer von 60 ms und einer maximalen Streuung von 25 Pixel. Als Eyetracker wurde ein Tobii EyeX Controller verwendet. Zunächst mussten die Probanden einen Synchronisationspunkt am oberen

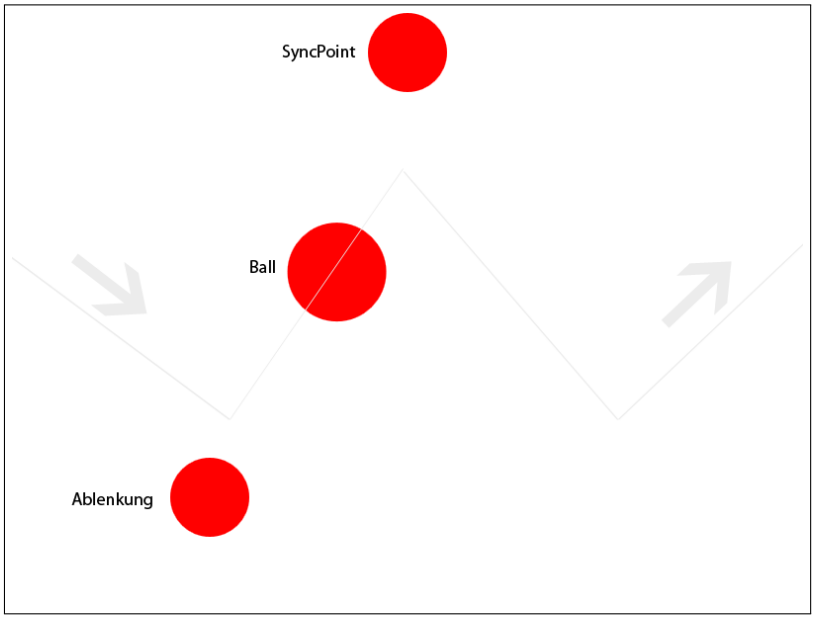

Abbildung 6.3: Ablauf der animierten SVG

Rand der Grafik fixieren. Mit Bestätigung der F5 Taste wurde die Animation gestartet. Die Aufgabe bestand im Fixieren des sich bewegenden Balls.

6.2.1 Ergebnis

Um die Ungenauigkeit des Eyetracker bei der Auswertung der dynamischen AOI auszugleichen, wurde ein Threshold von 30 Pixel gewählt. Proband 1 (siehe Tabelle 6.4) weist einen Anteil der Fixationen in den definierten AOI von ca. 91% auf. Obwohl dieser hohe Wert von den übrigen Probanden, insbesondere Proband 2, nicht erreicht wird, zeigen die Ergebnisse, dass die Auswertung von dynamischen AOI funktioniert. Eine genauere

Tabelle 6.4: Fixationsdauer innerhalb der AOI

AOI	Proband 1	Proband 2	Proband 3	Proband 4
Ball	16707 ms	2019 ms	14506 ms	9078 ms
SyncPoint	780 ms	751 ms	750 ms	150 ms
Ablenkung	702 ms	0 ms	0 ms	1893 ms

Betrachtung der Fixation der schlechtesten Werte (Proband 2: ca. 14% und Proband 4:

ca. 56%) ergeben, dass diese ein wenig zu hoch liegen. Diese Ungenauigkeit könnte durch die verwendeten Sehhilfen (Proband 2: Kontaktlinsen, Proband 4: Brille) entstanden sein. Abbildung 6.4 zeigt die Fixationen des Probanden 1 und die Position der AOI nach neun Sekunden.

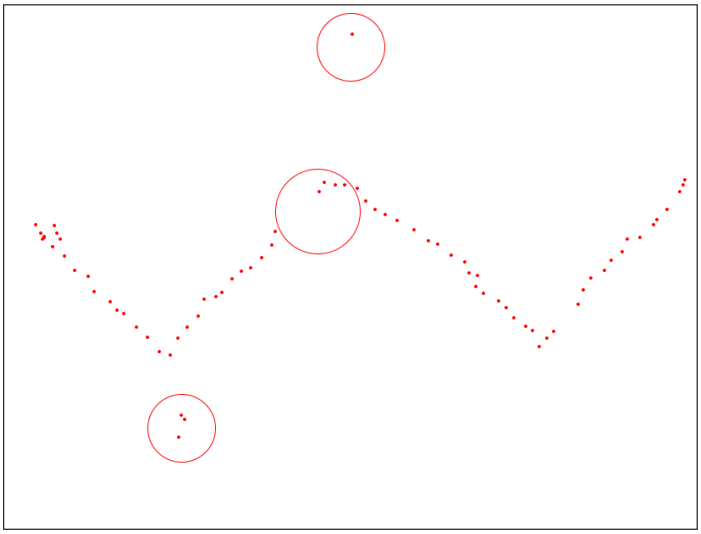

Abbildung 6.4: Fixationen des Probanden 1

In den Tabellen 6.5 bis 6.8 sind die Reihenfolgen der betrachteten AOI mit Zeitpunkt und Dauer aufgeführt. Aus ihnen geht hervor, dass im Falle von Proband 2 und 3 die Ablenkung keinen Einfluss auf das Blickverhalten hatte. Proband 1 und 4 reagierten nach ca. einer halben Sekunde auf den Ablenkungspunkt.

Tabelle 6.5: Proband 1

Zeitpunkt	**Dauer**	**AOI**
0 ms	780 ms	SyncPoint
989 ms	6972 ms	Ball
8564 ms	702 ms	Ablenkung
9341 ms	9735 ms	Ball

Tabelle 6.6: Proband 2

Zeitpunkt	**Dauer**	**AOI**
0 ms	255 ms	SyncPoint
3673 ms	769 ms	Ball
11621 ms	496 ms	SyncPoint
14860 ms	1250 ms	Ball

Tabelle 6.7: Proband 3

Zeitpunkt	**Dauer**	**AOI**
0 ms	750 ms	SyncPoint
1245 ms	6972 ms	Ball

Tabelle 6.8: Proband 4

Zeitpunkt	**Dauer**	**AOI**
495 ms	255 ms	SyncPoint
1683 ms	3882 ms	Ball
8594 ms	1773 ms	Ablenkung
10562 ms	3429 ms	Ball
15152 ms	120 ms	Ablenkung
15647 ms	1767 ms	Ball

6.3 Erkennung von AOI mittels Clusteranalyse

Als Daten dienten auch hier die im Abschnitt 6.1 genannten EyeMotion-Daten. Im Gegensatz zur Auswertung von statischen AOI wurde hier nicht mit einem, sondern mit den kumulierten Datensätzen der 21 Probanden gearbeitet.

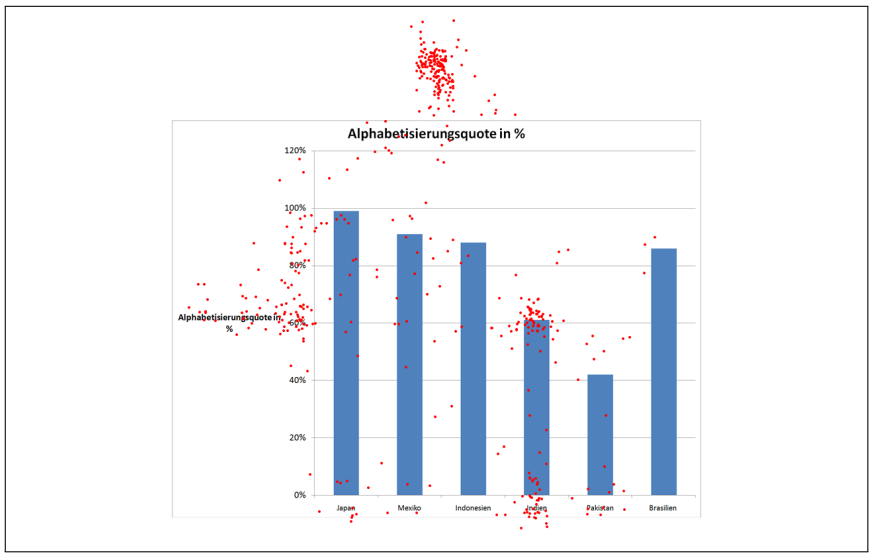

Abbildung 6.5: Kumulierte Fixationen von 21 Probanden

6.3.1 Heatmap

Zur Berechnung der Heatmap wurden als Distanzberechnung die Single-Linkage-Methode (siehe Abschnitt 2.4.2) und der Farbcode (siehe Tabelle 6.9) gewählt, der Faktor hinter einer Farbe gibt die Tiefe im Baum an, ab der ein Cluster eingefärbt wird. Die Heatmap (siehe

Tabelle 6.9: Farbcode der Heatmap

Farbe	**Tiefe im Baum**
Dunkelrot	0.70
Rot	0.50
Orange	0.45
Gelb	0.40
Grün	0.35
Blau	0.30
Dunkelblau	0.25

Abbildung 6.7) zeigt vier relevante Bereiche an den Stellen des Synchronisationspunktes, der Y-Achsbeschriftung „60%“, dem oberen Ende der Säule „Indien“ und der Beschriftung der Säule „Indien“. Des Weiteren sind der Bereich zwischen „100%“ und „80%“, sowie der Leerraum zwischen den Säulen „Mexiko“ und „Indonesien“ häufiger betrachtet worden.

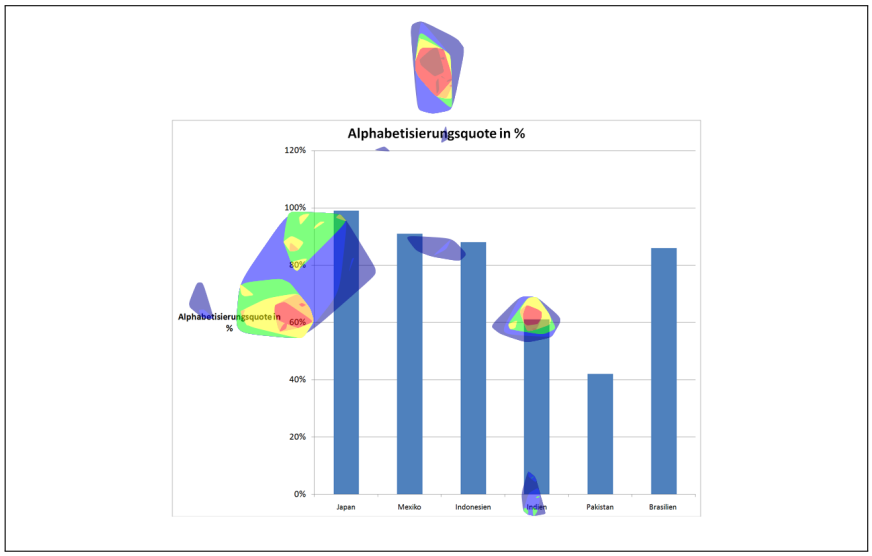

Abbildung 6.6: Heatmap

6.3.2 AOI

Eine weitere Darstellungsart der Clusteranalyse im Tool ist die Anzeige der AOI als Polygone. Dazu werden neben der Distanzberechnung, in diesem Fall erneut Single-Linkage, die Parameter Standardabweichung (30 Pixel) und die minimale Anzahl von Fixationen (15) gesetzt.

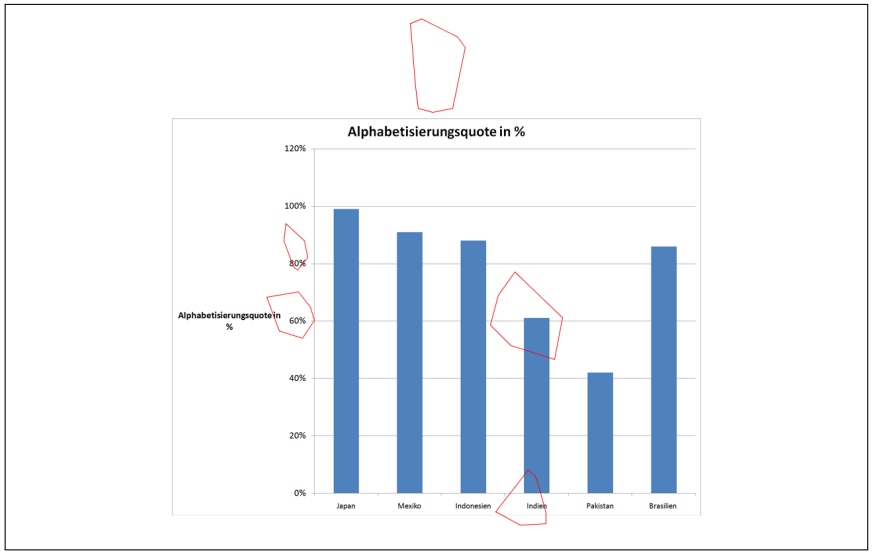

Abbildung 6.7: AOI-Erkennung durch Clusteranalyse

7 Fazit

Die Analyse von statischen Area of Interest liefert die Dauer der Fixationen, sowie die Reihenfolge der betrachteten AOI. Zusätzlich wird die entsprechende Transitionsmatrix ausgegeben. Die grafische Auswertung zeigt durch Angabe der Reihenfolge im Zentrum der AOI, sowie den Linien der Transitionen zwischen den AOI, das Blickverhalten des Probanden. Im Abschnitt 6.2 wurde die Analyse auf dynamischen AOI angewendet und gezeigt, dass das Tool mit dieser Art der Area of Interest umgehen kann. Es gibt jedoch eine Einschränkung bei der grafischen Auswertung. Diese ist statisch und kann die Ergebnisse der Auswertung von dynamischen AOI nicht korrekt anzeigen. Die automatische Ermittlung von Area of Interest durch eine hierarchische Clusteranalyse liefert die relevanten Bereiche an den richtigen Stellen. Diese stimmen mit den vermuteten Bereichen, die zuvor in dem Stimulus definiert wurden, überein.

7.1 Entwicklung

Die Entwicklung des Analysetool ist mit dieser Bachelorarbeit nicht beendet. Es gibt eine Vielzahl von verschiedenen Erweiterungsmöglichkeiten, wie zum Beispiel die Erfassung von externen Ereignissen. Darunter fallen Maus- und Tastatureingaben, sowie diverse Zeitgesteuerte Ereignisse, die einen Einfluss auf die Analyse haben können. Durch die Implementierung von DBSCAN kann die Erkennung von AOI stabiler gegenüber dem Rauschen einzelner Fixationen gemacht werden. Es ist auch eine Erweiterung der grafischen Auswertung um eine dynamische Komponente denkbar, Videos könnten so eine Auswertungsschicht bekommen, die die Auswertung in Echtzeit einblendet.

7.2 Ausblick

Die Analyse von dynamischen AOI erweitert die Anwendungsmöglichkeiten von Eyetracking enorm. Eine Reihe neuer Einsatzgebiete entstehen vor allem im Bereich der Videoanalyse und der Marktforschung. So kann zum Beispiel der Blick eines Kunden, der sich einen Werbefilm zu einem Produkt ansieht, verfolgt werden. Mit Hilfe der dynamischen AOI lässt sich die Zeit messen, die der Blick auf dem jeweiligen Produkt verweilt. Zusätzlich können durch die Analyse Störfaktoren bei der Erfassung ermittelt werden.

Literaturverzeichnis

[BRE13] Blascheck, Tanja ; Raschke, Michael ; Ertl, Thomas: Circular Heat Map Transition Diagram. In: *Proceedings of the 2013 Conference on Eye Tracking South Africa*. New York, NY, USA : ACM, 2013 (ETSA '13). – ISBN 978–1–4503–2110–5, 58–61

[BWGT09] Bulling, Andreas ; Ward, Jamie A. ; Gellersen, Hans ; Troster, Gerhard: Eye Movement Analysis for Activity Recognition. In: *Proceedings of the 11th International Conference on Ubiquitous Computing*. New York, NY, USA : ACM, 2009 (UbiComp '09). – ISBN 978–1–60558–431–7, 41–50

[EKSX96] Ester, Martin ; Kriegel, Hans-Peter ; Sander, Jörg ; Xu, Xiaowei: A density-based algorithm for discovering clusters in large spatial databases with noise. In: *KDD* Bd. 96, AAAI Press, 1996, S. 226–231

[Est13] Ester, Martin: *Knowledge Discovery in Databases: Techniken und Anwendungen*. Springer, 2013

[GJLR15] Goos, Steffen ; Janssen, Anna ; Lemburg, Sven ; Rauh, Lena: *EyeMotion: Erkennung relevanter Komponenten in Infografiken unter Verwendung von Eye-Trackern.* 2015. – Projektgruppe WS14/15, Knowledge Discovery, Christian-Albrechts-Universität zu Kiel

[GK99] Goldberg, Joseph H. ; Kotval, Xerxes P.: Computer interface evaluation using eye movements: methods and constructs. In: *International Journal of Industrial Ergonomics* Bd. 24, 1999, S. 631–645

[Gra72] Graham, Ronald L.: An efficient algorithm for determining the convex hull of a finite planar set. In: *Information Processing Letters*, 1972

[HJ10] Hansen, D.W. ; Ji, Qiang: In the Eye of the Beholder: A Survey of Models for Eyes and Gaze. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32 (2010), March, Nr. 3, S. 478–500. `http://dx.doi.org/10.1109/TPAMI.2009.30`. – DOI 10.1109/TPAMI.2009.30. – ISSN 0162–8828

[Lev91] Leven, Wilfried: *Blickverhalten von Konsumenten*. Physica-Verlag, 1991. – ISBN 978–3–7908–0554–3

[PMEB01] Probets, Steve ; Mong, Julius ; Evans, David ; Brailsford, David: Vector Graphics: From PostScript and Flash to SVG. In: *Proceedings of the 2001 ACM Symposium on Document Engineering*. New York, NY, USA : ACM, 2001 (DocEng '01). – ISBN 1–58113–432–0, 135–143

[PSK06] Panke, Stefanie ; Studer, Petra ; Kohls, Christian: Use & Usability: Portalevaluation mit Eye-Tracking und Logfile-Daten. In: *DELFI 2006. 4te Deutsche e-Learning Fachtagung Informatik*, 2006, S. 267–278

[Sau15] Sauter, Marc: *Sentry Eye Tracker ausprobiert.* http://www.golem.de/news/sentry-eye-tracker-ausprobiert-nur-anfaenger-starren-auf-die-mini-map-1501-111907.html, 2015. – Abruf: 13.09.2015

[SG00] Salvucci, Dario D. ; Goldberg, Joseph H.: Identifying Fixations and Saccades in Eye-tracking Protocols. In: *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications*. New York, NY, USA : ACM, 2000 (ETRA '00). – ISBN 1–58113–280–8, 71–78

[SMI08] *Synchronized Multimedia Integration Language (SMIL 3.0).* http://www.w3.org/TR/REC-smil/, 2008. – Abruf: 15.09.2015

[Som11] Sommerville, Ian: *Software Engineering.* 9. Pearson, 2011. – ISBN 978–0–13–703515–1

[SVG01] *Scalable Vector Graphics (SVG) 1.0 Specification.* http://www.w3.org/TR/2001/REC-SVG-20010904, 2001. – Abruf: 15.09.2015

[WSS14] Walber, Tina ; Scherp, Ansgar ; Staab, Steffen: Benefiting from Users' Gaze: Selection of Image Regions from Eye Tracking Information for Provided Tags. In: *Multimedia Tools Appl.* 71 (2014), Juli, Nr. 1, 363–390. http://dx.doi.org/10.1007/s11042-013-1390-3. – DOI 10.1007/s11042–013–1390–3. – ISSN 1380–7501

[WZ01] Wiedenbeck, Michael ; Züll, Cornelia: *Klassifikation mit Clusteranalyse: Grundlegende Techniken hierarchischer und K-means-Verfahren.* Bd. 10. 2001. – 18 S. http://nbn-resolving.de/urn:nbn:de:0168-ssoar-201428

Abbildungsverzeichnis

2.1 Memory: Auswahl 5
2.2 Memory: Aufgedeckt 5
2.3 Beispiel einer Transitionsmatrix (Quelle: [GK99])) 7
2.4 Beispiel eines Transitionsdiagramms (Quelle: [BRE13])) 7
2.5 Beispiel eines agglomerativen Verfahrens 9
2.6 Beispiel eines divisiven Verfahrens 10

4.1 Einteilung des Tools in vier Phasen 17
4.2 BPMN des Auswertungsprozesses 18
4.3 Übersicht Analyzer 19
4.4 AOI 20
4.5 Übersicht AnalyzerGUI 22

5.1 Benutzeroberfläche 31

6.1 AOI 33
6.2 AOI Analyse 34
6.3 Ablauf der animierten SVG 36
6.4 Fixationen des Probanden 1 37
6.5 Kumulierte Fixationen von 21 Probanden 38
6.6 Heatmap 39
6.7 AOI-Erkennung durch Clusteranalyse 39

Abkürzungsverzeichnis

AOI ..... Area of Interest
BPMN ..... Business Process Model and Notation
I-DT ..... Dispersion-Threshold Identification
I-HMM ..... Hidden Markov Model Fixation Identification
I-MST ..... Minimum Spanning Tree Identification
I-VT ..... Velocity-Threshold Identification
ROI ..... Region of Interest
SMIL ..... Synchronized Multimedia Integration Language

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Die eingereichte schriftliche Fassung der Arbeit entspricht der auf dem elektronischen Speichermedium.

Weiterhin versichere ich, dass diese Arbeit noch nicht als Abschlussarbeit an anderer Stelle vorgelegen hat.

Schleswig, den 28. September 2015

Steffen Goos