Institut für Visualisierung und Interaktive Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

# Visuelle Eye-Tracking-Analyse bei kooperativen Spielszenarien

Noel Schäfer

| | |
|---|---|
| **Studiengang:** | Softwaretechnik |
| **Prüfer/in:** | Prof. Dr. Daniel Weiskopf |
| **Betreuer/in:** | Dr. Tanja Blascheck,<br>Dr. Kuno Kurzhals,<br>Dipl.-Inf. Tanja Munz, M.Sc.,<br>Prof. Eugene Zhang |
| **Beginn am:** | 10. April 2019 |
| **Beendet am:** | 10. Oktober 2019 |

## Kurzfassung

Gegenstand dieser Arbeit ist die Entwicklung eines Werkzeuges zur Unterstützung der Analyse kooperativer und kompetitiver Spielszenarien zwischen mehreren Spielern hinsichtlich ihres Blickverhaltens. Im Vorfeld wurden bei diversen Partien einer virtuellen Version des chinesischen Brettspiels „Go“ zwei Kontrahenten per Eye-Tracking erfasst und das Spielfeld aufgezeichnet. Den Zweck des entwickelten Programmes bildet eine Zusammenführung der Datensätze unterschiedlicher Eye-Tracking-Geräte, die die Erzeugung eines Synchronisations-, Filterungs- und Transformationsverfahrens der Rohdaten sowie eine kombinierte visuelle Darstellung dieser auf der Bildschirmaufzeichnung des gespielten Szenarios umfasst. Abschließend kommt das entwickelte Werkzeug bei einer Analyse der aufgezeichneten Go-Partien zum Einsatz, bei der der Fokus insbesondere auf dem Beleuchten der Unterschiede in der visuellen Wahrnehmung verschiedener Szenarien zwischen den Spielern liegt und welche Erkenntnisse daraus hinsichtlich ihres Spielverhaltens gewonnen werden können. Dabei zeigt das Programm, dass ein Proband allgemein intensiver bestimmte Punkte des Spielfeldes fixiert, während der andere gleichmäßiger das ganze Feld betrachtet. Auch scheint der aggressivere Spieler häufiger die Steine des Gegners im Blick zu haben, während der passivere Spieler mehr dazu geneigt ist, auf Räume zwischen den eigenen Steinen zu achten.

## Abstract

The subject of this thesis is the development of a tool to support the analysis of cooperative and competitive game scenarios between several players with regard to their eye gaze movement. In multiple games of a virtual version of the Chinese board game "Go", the eye movements of two players were recorded and the playing field was screen-recorded. The main task of the tool is to merge raw data records from different eye-tracking devices. This incorporates the development of a synchronization, filtering and transformation process for the data, as well as a combined visual representation of the results projected onto the recording of the playing field. Lastly, the finished tool is used in an analysis of various of the previously recorded matches between the two players with the intent of highlighting differences in their visual perception of different scenarios and how this might have affected their decision-making. The program reveals that one player generally focuses more intensely on specific points on the playing field while the other observes the entire field more evenly. It also seems that the more aggressive player is more likely to look at the opponent’s stones, whereas the passive player spends more time looking for spaces between his own stones.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Verzeichnis der Algorithmen

# Abkürzungsverzeichnis

**AOI** Area of Interest. 31

**EGT** Eye-Gaze-Tracker. 16

**EOG** Elektrookulogramm. 16

**ETR** Eye-Tracker-Recording. 44

**GEP** Gaze-Event-Punkt. 15, 48, 58

**GP** Gaze-Punkt. 15

**GRF** Gaze-Recording-Frequenz. 19, 33, 34, 48, 50, 53

**GRT** Gaze-Recovery-Time. 34

**I-VT** Velocity-Threshold Identification. 19

**OpenCV** Open Source Computer Vision Library. 44

**TSV** Tab-Separated Values. 47

# 1 Einleitung

Alfred L. Yarbus stellte 1967 in seinem berühmten Experiment [Yar67] fest, dass die Augenbewegungen einer Versuchsperson stark abhängig sind von der Aufgabe, mit der diese betraut wurde. Das Blickverhalten eines Menschen beim Sammeln von Informationen über seine Umwelt scheint einzigartig [LNJ11], weist aber auch wiederkehrende Muster auf [BSBE17; Pri06]. Beim Eye-Tracking werden diese Augenbewegungen aufgezeichnet und analysierbar gemacht. Der Vorgang gewährt Einblicke in visuelle und kognitive Prozesse der Versuchsperson [RHTE18; SG00], hält den Aufmerksamkeitsverlauf fest und zeigt, welche Elemente während der Betrachtung scheinbar für besonders interessant befunden wurden [Duc07]. Aus diesen Anhaltspunkten lassen sich Informationen zu den Problemlösungsstrategien einer Person gewinnen [BSBE17] und sogar eine Einschätzung der Expertise im Rahmen des zu lösenden Problems durchführen [LHL+09; RHTE18; SKKH14].

In vielen vorangehenden Arbeiten wurden bereits die Auswirkungen diverser Problemszenarien auf die Augenbewegungen einer Versuchsperson bei der Informationsaufnahme untersucht (siehe Kapitel 3). Das geschaffene Szenario ist dabei häufig spielerischer Natur und enthält oftmals ein Element des Drucks, das den/die Spieler/in etwa durch begrenzte Zeit zum Handeln [LNJ11; SKKH14] oder strategische Komplexität [RHTE18] kognitiv auslastet. Auch ist in den vergangenen Jahren ein steigendes Interesse an der Nützlichkeit von Eye-Tracking-Technologien im Rahmen kollaborativer Arbeit festzustellen. An dieser Stelle möchte diese Arbeit anknüpfen. Im Kern soll es darum gehen, das aufgezeichnete Blickverhalten zweier Probanden/innen in einem kollaborativen oder kompetitiven Kontext bei der Lösung diverser Probleme zu visualisieren und infolge dessen vergleichbar und analysierbar zu machen. Beim gewählten Kontext handelt es sich um ein Strategie-Brettspiel, das sowohl einen Faktor strategischer Komplexität mit sich bringt, als auch in regelmäßigen Abständen Zeitdruck aufbaut.

Diese Arbeit verfolgt das Ziel, ein Konzept zur gebündelten visuellen Analyse des Blickverhaltens der Kontrahenten zu entwickeln und zu implementieren und damit eine Grundlage für zukünftige tiefergreifende Analysen zu schaffen. Dabei sind zunächst einige Probleme zu lösen, die die angestrebte Zusammenführung der Gaze-Daten mit sich bringt. Da Augenbewegungen der Probanden/innen beispielsweise unabhängig voneinander aufgezeichnet werden und dies häufig asynchron geschieht, muss ein zeitlicher Bezug zwischen zwei Aufzeichnungen gefunden werden. Dazu kommen gerätespezifische Merkmale bei Eye-Tracking-Hardware, wie etwa Unterschiede in der Dimensionalität des präsentierten Mediums und variable Abtastfrequenzen, für die ein Lösungsframework geschaffen werden soll, das die nötigen Maßnahmen zur Abstrahierung der Daten implementiert und dadurch identische Voraussetzungen schafft. Sind diese Voraussetzungen geschaffen, soll das Blickverhalten im gemeinsamen Kontext visualisiert werden. Von Interesse sind dabei die Scanpaths der Spieler, also der Verlauf des Fokus auf dem Spielfeld, die Schwerpunkte der Aufmerksamkeit jedes Spielers und das Hervorheben von Gemeinsamkeiten und Unterschieden zwischen deren Blickverhalten.

## Gliederung

Überblick über die Gliederung, in die sich diese Arbeit unterteilt:

**Kapitel 2 – Grundlagen:** In den Grundlagen wird eine Basis für den weiteren Inhalt geschaffen. Notwendige Informationen zum Bereich Eye-Tracking werden hier aufgeführt und verschiedene Eye-Tracking-Ansätze werden vorgestellt. Die unterschiedlichen Filtermethoden für Eye-Tracking-Daten und der für die Implementierung relevante I-VT-Filter werden erklärt. Anschließend werden mit dem Gaze Plot und der Attention Map zwei Visualisierungstechniken vorgestellt, die Bestandteil des zu entwickelnden Programmes werden sollen. Dann werden Histogramme definiert und es wird erklärt, wie diese als Mittel zum Bildvergleich genutzt werden können. Der Lab-Farbraum wird beschrieben und abschließend folgt ein Überblick über die wichtigsten Regeln des Brettspiels »Go«.

**Kapitel 3 – Verwandte Arbeiten:** Hier werden vorangehende Arbeiten beleuchtet, die inhaltlich für die Problematik dieser Arbeit relevant sind und sich mit Ansätzen beschäftigen, die hier wieder aufgegriffen werden.

**Kapitel 4 – Aufgabenstellung und Konzept:** In dem Kapitel „Aufgabenstellung und Konzept“ wird das von dieser Arbeit verfolgte Ziel genauer definiert. Hierzu wird die Aufgabenstellung gegliedert wiedergegeben und entworfene Lösungskonzepte grob umrissen. Außerdem wird die Aufgabenstellung in einen Anwendungsfall überführt, der beschreibt, wie das zu entwickelnde Programm am Ende genutzt werden können soll. Dieser wird im Anschluss an die Analyse erneut aufgegriffen, um die Funktionstüchtigkeit der Entwicklungsergebnisse zu verifizieren.

**Kapitel 5 – Implementierung:** Im Implementierungs-Kapitel geht es um die technischen Details der Programmkomponenten. Implementierungsspezifische Merkmale werden betrachtet und die Oberfläche des Programmes wird vorgestellt. Der Fokus liegt auf der praktischen Umsetzung des Konzepts.

**Kapitel 6 – Analyse und Auswertung:** Zum Analyseteil werden in Kapitel 6 erneut die aufgezeichneten Daten der Go-Partien herangezogen. Das entwickelte Programm kommt in der Praxis zum Einsatz und wird verwendet, um das Spielverhalten der beiden Kontrahenten zu untersuchen. Dabei werden zwei unterschiedliche Herangehensweisen genutzt, um zu ermitteln, wie die implementierten Visualisierungstechniken Spielereignisse sichtbar machen und wie gut das Programm den Anwendungsfall erfüllt. In einer anschließenden Auswertung wird für alle Anforderungen aus dem Anwendungsfall nacheinander gezeigt, dass sie von dem entwickelten Programm erfüllt werden.

**Kapitel 7 – Zusammenfassung und Ausblick:** Die Zusammenfassung rekapituliert die wesentlichen Punkte dieser Arbeit. Die wichtigsten gewonnenen Erkenntnisse werden hier noch einmal knapp aufgeführt. Abschließend werden im Ausblick Möglichkeiten zur Fortführung der behandelten Thematik vorgeschlagen. Dies umfasst sowohl die Motivation hinter der Aufgabenstellung, als auch die dazu gefundenen Lösungen.

# 2 Grundlagen

In diesem Kapitel werden grundlegende Methoden erläutert und Mechanismen erklärt, mit denen sich diese Arbeit im Fortgang befassen wird. Die unterschiedlichen Typen von Eye-Tracking-Ansätzen werden beleuchtet. Anschließend werden Gaze-Filter mit besonderem Fokus auf den I-VT-Filter erläutert. Dann folgt ein Überblick über Visualisierungstechniken, die für die Implementierung relevant sind. Nach einem Abschnitt über Histogramme und deren Nutzen zum Bildvergleich wird der Lab-Farbraum grob umrissen. Zum Schluss werden die wichtigsten Regeln des Go-Spiels aufgeführt um Begrifflichkeiten für den Analyseteil (Kapitel 6) zu klären.

## 2.1 Eye-Tracking

Dieser Abschnitt führt einige grundlegende Begrifflichkeiten des Eye-Trackings sowie verschiedene Eye-Tracking-Ansätze auf.

### 2.1.1 Begrifflichkeiten

Eye-Tracker zeichnen das Blickverhalten eines/r Probanden/in auf. Dabei entstehen Datenpunkte, sogenannte Gaze-Event-Punkte (GEPs) oder Gaze-Punkte (GPs), die im Wesentlichen aus einem Zeitstempel $Z$ und einem positiven Koordinatenpaar $(X, Y)$ bestehen, das die Position des Blickes des/r Probanden/in auf einem Medium zum Zeitpunkt $Z$ beschreibt. Moderne Geräte sind oftmals in der Lage, die genaue Position der Augen des/r Probanden/in im Raum zu tracken, was komplexere und präzisere Filtermechanismen ermöglicht (mehr dazu in Abschnitt 2.2). In diesem Abschnitt werden nun relevante Begrifflichkeiten rund um das Thema Eye-Tracking geklärt.

Bei der Interpretation der Gaze-Punkte unterscheidet man grundsätzlich zwischen zwei Arten, Fixationen und Sakkaden:

**Fixation.** Bei einer Fixation handelt es sich um eine Ansammlung von Gaze-Punkten, die räumlich und zeitlich dicht beieinander liegen. Sie treten dann auf, wenn das Auge auf einem stationären Objekt ruht [Duc07]. Tatsächlich ruht der Blick bei einer Fixation allerdings nicht vollständig. Stattdessen lassen sich Fixationen aus mindestens drei weiteren Arten von Augenbewegungen zusammensetzen: Tremor, Drift und Mikrosakkaden. Mehr Informationen dazu liefern Holmqvist *et. al* [HA17] und Duchowski [Duc07]. Es gibt diverse Filteralgorithmen, nach denen Rohdaten in Fixationen und Sakkaden unterteilt werden können (siehe Abschnitt 2.2).

**Sakkade.** Als Sakkade werden Augenbewegungen zwischen Fixationen bezeichnet. Sie können sowohl bewusst, also auch reflexartig durchgeführt werden, und dauern für gewöhnlich zwischen 10 ms und 100 ms an, während denen praktisch keine visuellen Informationen verarbeitet werden [BKR+17; Duc07].

**Scanpath.** Ein Scanpath oder „Suchpfad“ ist eine Sequenz von Fixationen und Sakkaden (siehe Abbildung 2.1). Sie beschreibt den zeitlichen Verlauf des visuellen Fokus eines Betrachters [Duc07] und kann Aufschluss über dessen Suchverhalten geben [BKR+17].

**Smooth Pursuit.** Beim Smooth Pursuit verfolgt ein Betrachter mit den Augen ein bewegliches Ziel. Dies geschieht unbewusst und mit einer Geschwindigkeit von etwa 10-30 Grad pro Sekunde [BKR+17; HNA+11].

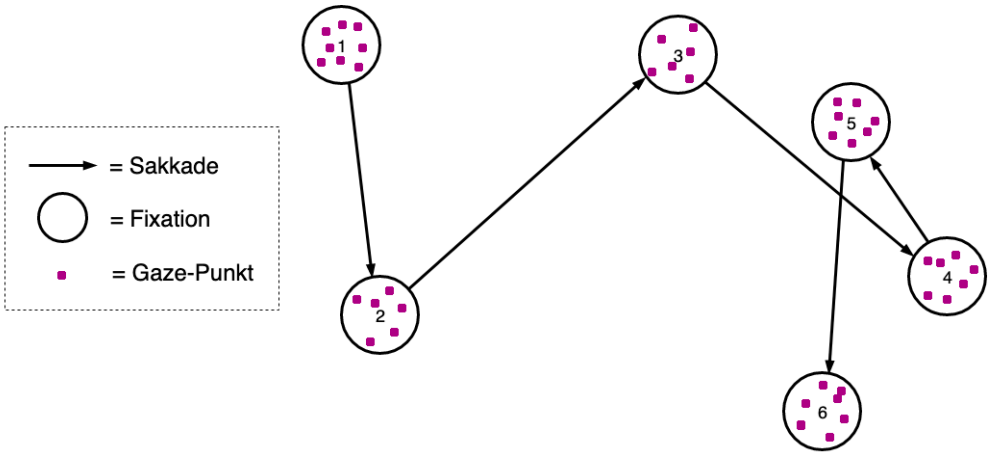



**Abbildung 2.1:** Scanpath, bestehend aus Gaze-Punkten (lila Punkte), aus denen sich Fixationen zusammensetzen (Kreise), die durch Sakkaden (Pfeile) verbunden sind.

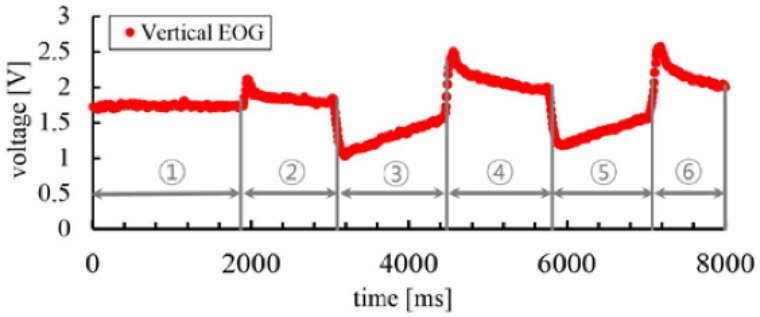
### 2.1.2 Eye-Tracker-Ansätze

Prinzipiell lassen sich Eye-Tracker, auch Eye-Gaze-Tracker (EGT) genannt, in folgende drei Kategorien unterteilen:

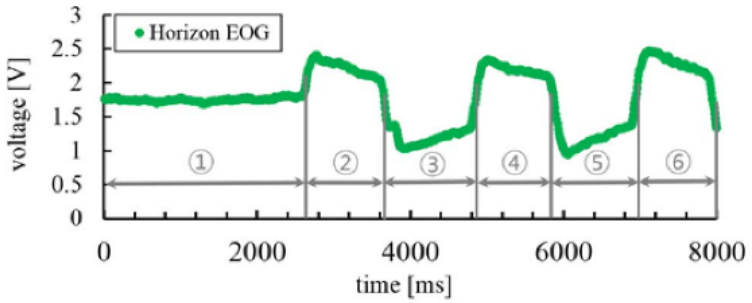
**Kontaktbasierte Tracker** arbeiten mit einer Komponente, die direkt am Augapfel angebracht wird. Hierbei kommt üblicherweise eine Kontaktlinse zum Einsatz, die ein extern verfolgbares Medium enthält. Robinson [Rob63] beschreibt ein solches Verfahren, bei dem ein/e Proband/in Kontaktlinsen trägt, in die jeweils zwei um die Pupille führende Drahtspulen eingebettet sind. Mithilfe zweier 90° versetzter magnetischer Felder, die gemäß Faraday eine Spannung zwischen den Spulen induzieren, lassen sich horizontale, vertikale und kreisende Augenbewegungen messen. An den Spulen der Kontaktlinsen sind hierfür weitere Drähte angeschlossen, die die zwischen den Spulen herrschende Spannung an einen Verstärker führen, von dem aus Phasenveränderungen aufgezeichnet werden können.

**Elektrookulographische Tracker** analysieren ein feines Feld elektrischen Potenzials, das das menschliche Auge zwischen Hornhaut und Netzhaut erzeugt. Bulling *et al.* [BRT09] beschreiben ein Modell, bei dem mittels zwei Elektrodenpaaren, die um das Auge herum auf der Haut angebracht werden, das durch das Feld erzeugte elektrische Signal gemessen wird. Ergebnis dieser Messung ist ein sogenanntes Elektrookulogramm (EOG) (siehe Abbildung 2.2). Ändert sich die Ausrichtung des Auges, ändert sich gleichzeitig die Ausrichtung der Hornhaut und

der Netzhaut und damit des erzeugten Feldes. Während die Hornhaut sich einer Elektrode nähert, nähert sich gleichzeitig die Netzhaut der gegenüberliegenden Elektrode. Dies wirkt sich auf das gemessene EOG-Signal aus, über das sich Augenbewegungen tracken lassen.




(a) Vertical EOG signal




(b) Horizontal EOG signal

**Abbildung 2.2: (a)** vertikale und **(b)** horizontale EOG-Signale. Bei **(a)**: (1): Blick zentral. (2), (4), (6): Blick nach oben. (3), (5): Blick nach unten. Bei **(b)**: (1): Blick zentral. (2), (4), (6): Blick nach rechts. (3), (5): Blick nach links. Abbildungen aus Moon *et. al* [MPYK17], © 2017 IEEE.

**Optische / Videookulographische Tracker** arbeiten hingegen kontaktlos. Üblicherweise wird die Blickrichtung des/r Probanden/in hierfür mit Kameras oder anderen optischen Sensoren aufgezeichnet und Veränderungen im Lichtreflektionsverhalten der Augen gemessen. Ein solcher Ansatz, genannt *Dark Pupil Tracking*, macht sich die Differenzen im Kontrast zwischen Pupille und Iris sowie Iris und Lederhaut (*lat.* Sclera) des Auges zunutze [MR05] (siehe Abbildung 2.4 (a)). Dieser Kontrast kann bei Bedarf mit einer Infrarotlichtquelle verstärkt werden (siehe Abbildung 2.3 A). Wird die Lichtquelle so platziert, dass sie entlang der optischen Achse der Kamera direkt in das Auge des/r Probanden/in scheint, wird das für Menschen unsichtbare Infrarot-Licht im Augeninneren reflektiert und fällt zurück in die Kamera (siehe Abbildung 2.3 B). Die Pupille erscheint in diesem Fall hell erleuchtet (siehe Abbildung 2.4 (b)), man spricht vom *Bright Pupil Tracking*.
Ein weiteres optisches Verfahren von Cornsweet und Crane [CC73] nutzt hingegen Reflektionen von Licht in unterschiedlichen Schichten des Auges, sogenannte Purkinje-Bilder oder Purkinje-Reflektionen, zur Bestimmung der Blickrichtung. Ein optischer Sensor zeichnet helle Reflektionen einer Lichtquelle in Form zweier heller Punkte auf der vorderseitigen

Oberfläche der Hornhaut und der rückseitigen Oberfläche der Linse auf. Mit einer Änderung der Blickrichtung ändert sich der Abstand zwischen diesen zwei Lichtpunkten, wodurch die Blickrichtung präzise gemessen werden kann.
Unter handelsüblicher Eye-Tracking-Hardware sind optische Geräte heutzutage am weitesten verbreitet dank vergleichsweise niedriger Beschaffungskosten und ihrer nichtinvasiven oder minimalinvasiven Natur. Tracking-Geräte dieser Art gibt es sowohl in Form von Brillen [RMK+88], als auch extern montiert.

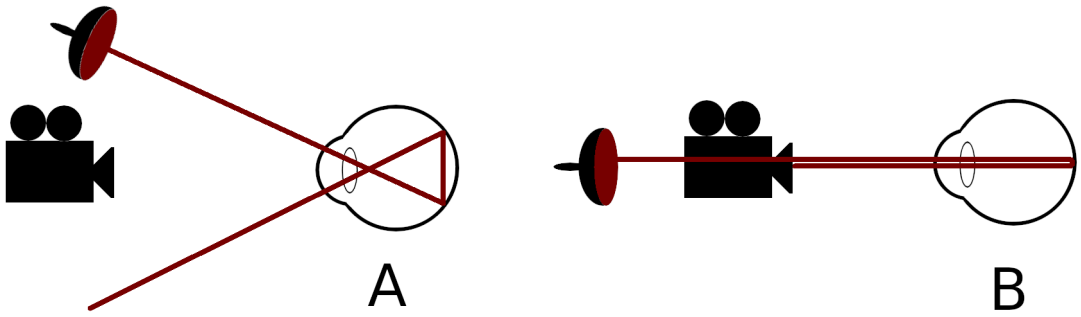



**Abbildung 2.3:** A: Dark Pupil Tracking. Die Kamera trackt eine dunkle Pupille (siehe Abbildung 2.4 (a)), IR-Licht verstärkt Kontrast und Reflektionsverhalten und erleichtert so das Tracking. B: Bright Pupil Tracking. Das IR-Licht scheint entlang der optischen Achse in das Auge und wird zurückreflektiert. Für die Kamera erscheint die Pupille hell erleuchtet (siehe Abbildung 2.4 (b)).

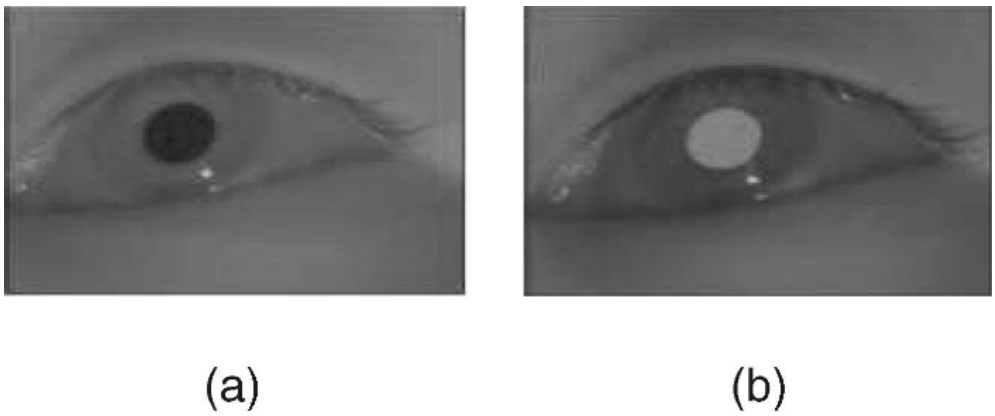



**Abbildung 2.4:** (a): Dunkle Pupille (Dark Pupil), (b): Erleuchtete Pupille durch Reflektion (Bright Pupil). Abbildung aus Hansen *et. al* [HJ10], © 2010 IEEE, Teilabbildung aus Fig. 5.

Bei den zwei im Rahmen dieser Arbeit verwendeten Geräten handelt es sich um optische Tracker, daher wird sich dieses Dokument im weiteren Verlauf ausschließlich auf Anwendungsszenarien im Zusammenhang mit optischem Eye-Tracking beschränken.

## 2.2 Gaze-Filter

Dieser Abschnitt beschäftigt sich mit dem Filtern und Aggregieren von Eye-Tracking-Daten. Dabei werden Rohdaten in Fixationen und Sakkaden aggregiert (siehe Unterabschnitt 2.1.1). Es existieren einige Methoden, mit denen dies erreicht werden kann:

**Geschwindigkeitsbasierte Filtermethoden** unterteilen Gaze-Punkte in Fixationen und Sakkaden basierend auf deren Punkt-zu-Punkt-Geschwindigkeit. Das Verfahren beruht auf der Annahme, dass die Geschwindigkeit zwischen Gaze-Punkten, die Teil einer Fixation sind, niedrig ist (<100 Grad pro Sekunde), und solchen, die Teil einer Sakkade sind, hoch ist (>300 Grad pro Sekunde) [SG00].

**Streuungsbasierte Filtermethoden** nutzen die Tatsache, dass Gaze-Punkte, die Teil einer Fixation sind und daher geringe Punkt-zu-Punkt-Geschwindigkeiten aufweisen, dicht beieinander liegen. Filter dieses Typs arbeiten auf Grundlage einer vordefinierten maximalen Streuung oder eines maximalen Abstandes, um Fixationsgruppen zu erkennen, und kombinieren dies nicht selten mit einem Threshold für die Minimaldauer einer Fixation (z. B. >100 ms) [SG00].

**Flächenbasierte Filtermethoden** schränken die Erkennung von Fixationen auf gewisse vordefinierte Bereiche ein, sogenannten *Areas of Interest*. Darüber hinaus kommen auch hier wie bei den streuungsbasierten Verfahren Minimaldauer-Thresholds zum Einsatz, um Fixationen von Sakkaden zu unterscheiden, die die Area of Interest durchkreuzen [SG00].

### 2.2.1 I-VT-Filter

Der Velocity-Threshold Identification (I-VT)-Filter zählt zu den geschwindigkeitsbasierten Methoden. Typischerweise arbeitet er neben dem Datensatz mit nur einem zusätzlichen Parameter, dem *velocity threshold* [SG00]. Er iteriert wiederholt über den Datensatz mit Gaze-Punkten und aggregiert in mehreren Schritten Fixationen anhand dieser Geschwindigkeitsschranke.
Im ersten Schritt wird für jeden Gaze-Punkt eine Punkt-zu-Punkt-Geschwindigkeit ermittelt. Es bietet sich an, hierfür einfach den Abstand eines Punktes zu seinem Vorgänger oder dem Nachfolger durch die Gaze-Recording-Frequenz (GRF) zu teilen. Dieser Wert is jedoch sehr ungenau, weil er bei Direktvergleich mit einer beliebigen Geschwindigkeitsschranke den Abstand des Betrachters zur Bildfläche nicht berücksichtigt. Einen besseren Ansatz liefert der Blickwinkel zwischen den zwei Punkten unter Berücksichtigung des Abstandes des Betrachters. Ist diese Information im Datensatz enthalten, so lässt sich entweder ein velocity threshold in Grad pro Sekunden definieren oder, unter der Annahme, dass der Abstand des Betrachters zur Bildfläche konstant ist, eine präzisere und angemessenere Schranke für eine einfache Punktabstandsberechnung bestimmen. Liegt diese Information nicht vor, so müssen diesbezüglich Annahmen getroffen werden. Bei monitorbasierten Tracking-Geräten bieten sich etwa 65 Zentimeter als angemessener Näherungswert für den Abstand des Betrachters zur Bildfläche an, da dieser Wert häufig im empfohlenen Bereich liegt (siehe Unterabschnitt 4.1.1).
Im zweiten Schritt wird die berechnete Punkt-zu-Punkt-Geschwindigkeit mit dem velocity threshold verglichen und Punkte entsprechend als Fixationen markiert, falls sie darunter liegen, und als Sakkaden markiert, falls sie darüber liegen.
Im dritten Schritt werden aufeinanderfolgende Fixationspunkte als Fixationsgruppen zusammengefasst und die Sakkaden zwischen den Gruppen entfernt.
Im letzten Schritt werden für jede Fixationsgruppe Mittelwerte der Koordinaten aller enthaltenen

Gaze-Punkte berechnet und diese als Koordinaten der Fixation festgelegt. Außerdem lässt sich die Fixationsdauer jeder Fixationsgruppe über die Zeitstempel des ersten und des letzten enthaltenen Datenpunktes bestimmen.

## 2.3 Visualisierungstechniken

Dieses Kapitel stellt mit den Gaze Plots und den Attention Maps zwei Methoden vor, Eye-Tracking-Daten zur Auswertung zu visualisieren.

### 2.3.1 Gaze Plot (Scanpath)

Unter einem Gaze Plot versteht diese Arbeit prinzipiell eine Visualisierung eines Scanpaths, bei der Fixationen als Kreise oder Ringe und Sakkaden als Verbindungsgeraden dazwischen dargestellt werden (siehe Abbildung 2.5). Diese lässt sich sowohl für das Suchverhalten eines/r Probanden/in auf Standbildern anfertigen, als auch über laufendes Filmmaterial zeichnen. Im Falle eines Standbildes wird je nach Anforderung der gesamte Blickverlauf mit Fixationsreihenfolge dargestellt. Bei Videos hingegen wird in Echtzeit das Blickverhalten eines Betrachters beim Ansehen rekonstruiert, wobei jeder neue Gaze-Punkt zur entsprechenden Zeit an entsprechender Stelle erscheint. In diesem Fall bietet es sich an, einen Threshold für die Anzahl der vergangenen Elemente zu definieren, die für jeden neuen Frame noch dargestellt werden sollen. Tut man dies nicht, kann der Gaze Plot unübersichtlich werden und je nach Anzahl und Größe der Fixationen und Dicke der Sakkaden einen großen Teil des Mediums überdecken. Beim Zeichnen des Gaze Plots besteht zusätzlich die Möglichkeit, weitere Informationen in diverse Parameter zu kodieren. So kann man etwa den Radius eines Fixationskreises von der Dauer der Fixation abhängig machen.

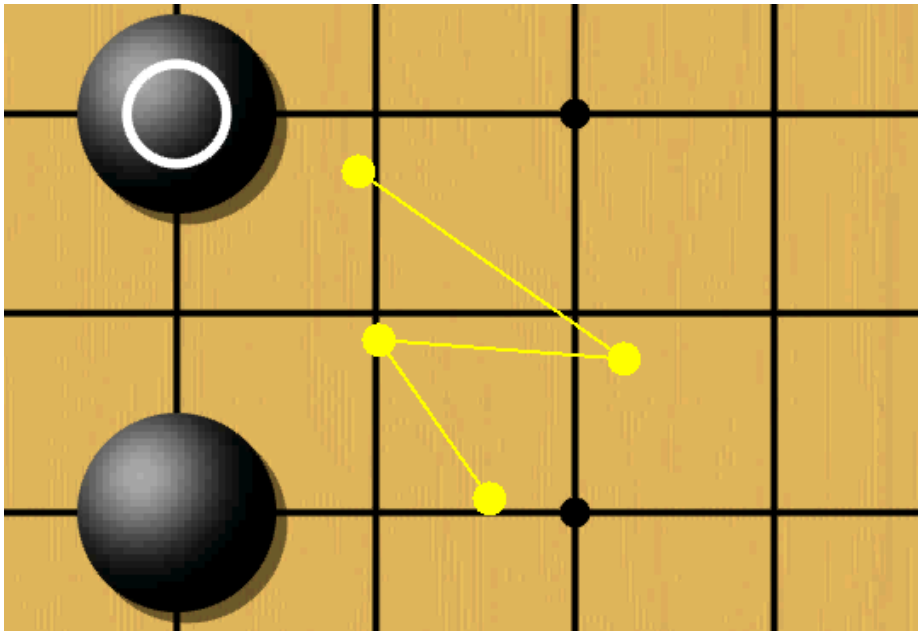


**Abbildung 2.5:** Visualisierung eines Scanpaths, auch Gaze Plot genannt. Die Kreise verkörpern Fixationen, die Linien stehen für Sakkaden.

### 2.3.2 Attention Maps

Attention Maps, oftmals auch allgemeiner »Heatmaps« genannt, eignen sich besonders beim Darstellen einer größeren Menge an Gaze-Punkten besser als Scanpaths. Sie ermöglichen eine übersichtliche Kombination der Gaze-Daten mehrerer Betrachter und stellen ein starkes Verhältnis zwischen der Anzahl der Gaze-Punkte und den Bereichen her, in die diese fallen [DPMO12]. Im Kontext mehrerer Betrachter unterscheiden sich Heatmaps auch dahingehend von Scanpaths, dass sich erzeugte Heatmaps zweier Betrachter über ein Medium wohl eher ähneln als erzeugte Scanpaths, da verschiedene Betrachter häufig dazu neigen, die gleichen Regionen eines Mediums zu betrachten, dies jedoch in unterschiedlicher Reihenfolge tun [DPMO12; Pri06].
Zur Erstellung einer Heatmap wird für jeden Pixel $P$ mit Koordinaten $(x, y)$ eine Intensität $I(x, y)$ ermittelt. Diese Intensität errechnet sich aus der Summe der erfahrenen »Hitze« (= Aufmerksamkeit) durch umliegende Gaze-Punkte oder Fixationen mit Koordinaten $(i, j)$. Mit steigendem Abstand zwischen Pixel $P$ und einem Gaze-Punkt sinkt die erfahrene Hitze durch den Gaze-Punkt exponentiell [DPMO12]. Beim Rendern der Heatmap wird die Intensität jedes Pixels farblich kodiert (siehe Abbildung 2.6).

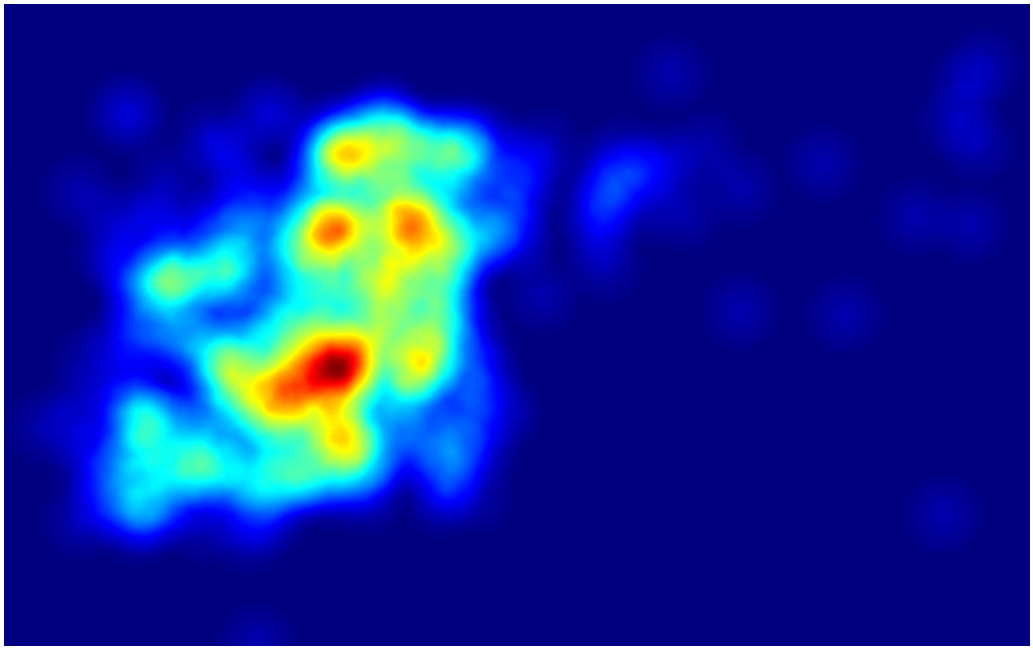


**Abbildung 2.6:** Abbildung einer Attention Map (Heatmap), farblich kodiert von Dunkelblau (keine Aufmerksamkeit, Wert null) bis Rot/Braun (höchste Aufmerksamkeit, Wert maximal).

## 2.4 Histogramme als Mittel zum Bildvergleich

Histogramme sind ein nützliches Werkzeug, um die Ähnlichkeit zweier Bilder zu bestimmen. Ein Histogramm wird jeweils über einen Satz Elemente erzeugt und gibt Auskunft über die Anzahl $y$ an Elementen im Set $Z$, die den Wert $x$ haben. Histogramme bilden also Werte auf Zähler ab. Mittels verschiedener Vergleichsmethoden lassen sich dann zwei Histogramme miteinander vergleichen, um

einen Ähnlichkeitswert zu ermitteln. Für den Bildvergleich lässt sich dieses Verfahren anwenden, indem Histogramme über die einzelnen Farbkanäle der Bilder gebildet werden, die verglichen werden sollen. So lässt sich ein Eindruck darüber gewinnen, wie ähnlich sich die Bilder hinsichtlich Verteilung der Farbkomponenten sind.

#### Definition

Diese Arbeit definiert Histogramme ähnlich Cha *et. al* [CS02] wie folgt:

Sei $X = \{x_0, x_1, ..., x_{n-1}\}$ eine Menge von $n$ vielen möglichen Messwerten (auch *bins* genannt).
Man betrachte ein Set $Z = \{z_1, z_2, ..., z_m\}$ mit $m$ Elementen.
Sei $B = \{b_1, b_2, ..., b_m\}$ eine Messung, wobei $b_i \in X$, $b_i$ der zugehörige Messwert zu Element $z_i \in Z$.
Dann ist $H(x, B)$ das Histogramm der Messung $B$ entlang des Messwertes $x \in X$ und $y$ ein Ergebnis dieser Funktion.

Es handelt sich um einen $n$-dimensionalen geordneten Vektor, der einem Messwert $x$ die Anzahl $y$ der Vorkommen von $x$ in Messung $B$ über Set $Z$ zuordnet.

#### Histogramm-Vergleich

Es gibt diverse Methoden, um die Ähnlichkeit der Histogramme zweier Datensätze zu bewerten. Gemäß dem Histogrammvergleich des OpenCV-Frameworks [OHC] definiert diese Arbeit die Ähnlichkeit zweier Histogramme $H_1$ und $H_2$ als Metrik $d(H_1, H_2)$, die einen Ähnlichkeitswert liefert. OpenCV bietet für den Histogramm-Vergleich die folgenden vier Metriken an: **Correlation**, **Intersection**, **Chi-Square** und **Bhattacharyya**. Da in der Implementierung ausschließlich **Correlation** verwendet wird, beschränkt sich dieser Abschnitt darauf.

**Correlation**

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H_1})(H_2(I) - \bar{H_2})}{\sqrt{\sum_I (H_1(I) - \bar{H_1})^2 \sum_I (H_2(I) - \bar{H_2})^2}}$$

wobei

$$\bar{H_k} = \frac{1}{N} \sum_J H_k(J)$$

und $N$ = Anzahl der möglichen Messwerte (bins).

Hierbei handelt es sich um einen Korrelationskoeffizienten, auch *Pearson's r* genannt. Er misst das Verhältnis zwischen der Summe der mittelwertbereinigten Messwertpaare bzw. der Summe der Abweichungsprodukte (über dem Bruchstrich) und der Wurzel des Produktes der Summen der Varianzen, also quadratischen Abweichungen vom Mittelwert (unter dem Bruchstrich), der Werte zweier Messungen und bewertet dadurch ihre lineare Abhängigkeit voneinander. Einfacher ausgedrückt ist er ein Maß für einen linearen Zusammenhang zwischen Messwertpaaren zweier Messungen [RN88]. Der ermittelte Wert liegt zwischen **0** und **1** (**Fall 1**) bzw. **-1** und **0** (**Fall 2**):

In **Fall 1** liegt der Wert nahe 1, falls scheinbar ein direkter Zusammenhang zwischen der Größe der Messwerte der Messwertpaare besteht, d.h. dass beide Werte generell ähnlich klein oder groß sind. Die Regressionsgerade in einem Streuungsdiagramm gemäß Fall 1 würde in diesem Fall linear nahe Steigung 1 steigen (siehe Abbildung 2.7 A).

In **Fall 2** liegt der Wert nahe -1, falls scheinbar ein inverser Zusammenhang zwischen der Größe der zwei Messwerte der Messwertpaare besteht, d.h. dass der eine Wert aus Messung A generell groß ist, wenn der andere Wert aus Messung B klein ist und umgekehrt. Die Regressionsgerade im Streuungsdiagramm mit den Messwertepaaren der Messungen entlang den zwei Achsen würde in diesem Fall linear nahe Steigung -1 fallen (siehe Abbildung 2.7 B).

In beiden Fällen liegt der Wert nahe 0, falls die Korrelation klein ist, sich also kein Zusammenhang zwischen den Größen der Werte abzubilden scheint. Die Regressionsgerade in einem Streuungsdiagramm würde also nur leicht steigen oder fallen nahe Steigung 0. Die Korrelation zwischen den Messungen ist hoch bei einem Ergebnis nahe -1 bzw. 1 und gering bei einem Ergebnis nahe 0. Identische Messungen produzieren einen Ähnlichkeitswert von 1, grundverschiedene Messungen einen Ähnlichkeitswert von 0.
Im vorliegenden Anwendungsfall handelt es sich bei den zwei Messungen um zwei zu vergleichende Histogramme über die Farbwerte der Bildpunkte zweier Bilder, bei denen jeweils die Werte der Histogramme über die Punkte bzw. Pixel mit den gleichen Koordinaten in den Bildern als Messwertpaare herangezogen werden. Da eine inverser Zusammenhang der Farbwerte für einen direkten Bildvergleich nicht von Nutzen ist, beschränkt sich die Metrik hierbei ausschließlich auf Fall 1 und produziert somit einen Wert zwischen 0 und 1 als Maß der Korrelation.
Die Formel entstammt OpenCV [OHC], der Histogrammvergleich für zwei über Bilder erzeugte Histogramme ist in der OpenCV-Library (siehe Unterabschnitt 5.2.2) implementiert. Einen detaillierteren Überblick über diverse weitere Ähnlichkeitsmaße liefert Cha [Cha07]. Weitere implementierungsspezifische Details zum Bildvergleich mittels Histogrammen finden sich in Unterabschnitt 5.8.2.

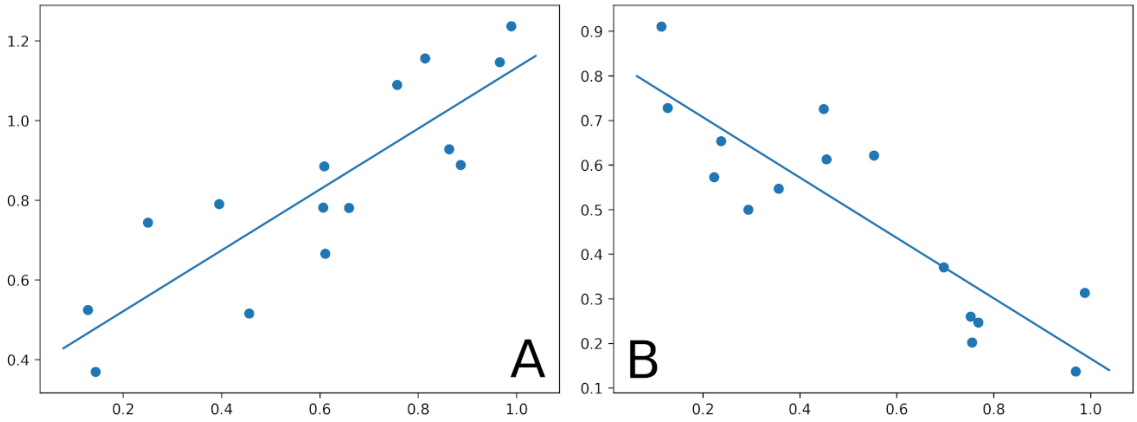



**Abbildung 2.7:** Zwei Scatter Plots mit Regressionsgeraden. A: Regressionsgerade steigt; direkter Zusammenhang zwischen den Messwertpaaren. B: Regressionsgerade fällt; inverser Zusammenhang zwischen den Messwertpaaren.

## 2.5 Lab-Farbraum

Der Lab-Farbraum, auch L*a*b-Farbraum oder CIELAB genannt, ist ein wahrnehmungsbezogener Farbraum und getreu der menschlichen Wahrnehmungsfähigkeit definiert. Dies steht im Kontrast zu nicht wahrnehmungsbezogenen Farbräumen wie etwa dem weit verbreiteten RGB, bei dem

die euklidischen Abstände zwischen den Farben im Farbraum nicht den vom Menschen wahrgenommenen Abständen entsprechen [Pas01]. Lab besteht aus einer Achse *L* für die Helligkeit der Farbe und zwei weiteren Achsen, die jeweils Farbarten und -intensitäten beschreiben–*a* und *b*. Der Farbton von a und b bestimmt sich dadurch, ob der Wert der Koordinate positiv oder negativ ist. Die a-Koordinate beschreibt einen Farbton zwischen Grün (negativ) und Rot (positiv), und die b-Koordinate beschreibt einen Farbton zwischen Blau (negativ) und Gelb (positiv). Die Intensität der Farbe der zwei Kanäle bestimmt sich dadurch, wie groß ein positiver bzw. klein ein negativer Wert ist. Größere positive bzw. kleinere negative Werte erzeugen intensivere Farben. Im Falle von a = b = 0 ist das Resultat ein Farbton ohne jegliche Intensität, also ein farbneutraler Wert auf der Grauskala (Schwarz über alle Graustufen bis Weiß, je nach L). Je größer L, desto heller die Farbe.

Die Implementierung macht sich einen Farbvergleichsalgorithmus zunutze, der mit dem Lab-Farbraum arbeitet. Es handelt sich um den Ciede2000-Algorithmus. Mehr Informationen dazu in Johnson *et. al* [JF03].

## 2.6 Strategie-Brettspiel „Go“

Im Hinblick auf die Aufgabenstellung in Kapitel 4 und den Analyseteil in Kapitel 6 werden in diesem Abschnitt die grundlegenden Spielregeln des Spiels „Go“ erläutert, das exemplarisch als Teil der Aufgabenstellung als kooperatives/kompetitives Strategiespiel gewählt wurde.

Go ist ein altertümliches Brettspiel chinesischen Ursprungs, dessen Alter auf zwischen 2.500 und 4.000 Jahre geschätzt wird [AGA]. Üblicherweise werden Partien zwischen zwei Spielern ausgetragen, es gibt jedoch auch Varianten mit mehr als zwei Spielern (Pair Go, bzw. *Rengo*).
Das Spiel wird auf einem quadratischen Spielbrett mit Gitternetz-Aufdruck ausgetragen (siehe Abbildung 2.8). Steine werden dabei auf Schnittpunkte zwischen horizontalen und vertikalen Gitterlinien platziert. Da die Größe des Spielfeldes flexibel ist, können Partien zwischen wenigen Minuten oder gar einigen Stunden lang andauern, bevor ein Sieger feststeht. Die Standardgröße beträgt 19x19 Felder; 13x13 und 9x9 sind jedoch ebenfalls üblich. Anfängern wird im Allgemeinen dazu geraten, das Spiel auf kleinen Spielfeldern zu erlernen.

Für die Aufzeichnungen wurde eine virtuelle Version dieses Spieles in Form der »KGS Go«-Software [KGS] verwendet.

### 2.6.1 Die Spielregeln

Im Ausgangszustand ist das Spielbrett leer und jede/r der beiden Spieler/innen bekommt alle Steine einer Farbe. Der/die Spieler/in mit den schwarzen Steinen macht den ersten Zug. Die Spieler/innen sind abwechselnd an der Reihe und setzen pro Zug jeweils einen Stein. Alternativ kann sich ein Spieler dazu entscheiden zu passen. In diesem Fall muss der/die passende Spieler/in seinem Gegenspieler einen seiner noch nicht gesetzten Steine als „Gefangenen“ überreichen. Passen beide Spieler/innen nacheinander, ist das Spiel (ggfs. vorzeitig) zu Ende.

Im Wesentlichen besteht ein Spiel aus zwei Vorgängen: dem **Erobern von Gebieten** und dem **Fangen von gegnerischen Steinen**.

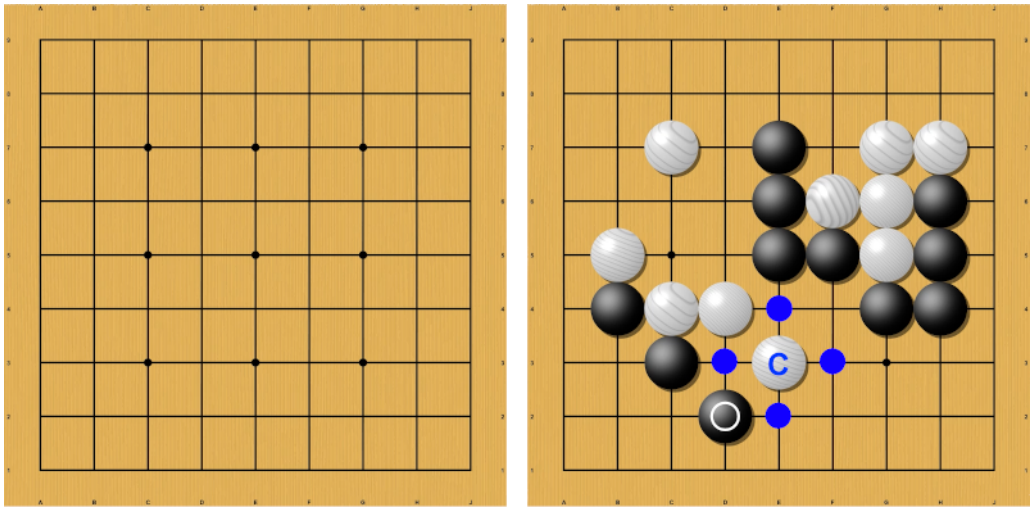



**Abbildung 2.8:** Links: Leeres Go-Spielfeld, 9x9; Rechts: Blaue Punkte markieren die Freiheiten des weißen Steines „C“. Gemäß Unterabschnitt 2.6.1 gilt keiner der bisher gesetzten Steine als "gefangen", und es wurden noch keine Gebiete erobert.

#### Erobern von Gebieten

In der Ausgangssituation besteht das gesamte Spielfeld aus „neutralem Gebiet“. Gebiete gelten als erobert, wenn sie durch eine ununterbrochene Kette von Steinen einer Farbe vollständig vom neutralen Gebiet abgeschnitten sind. Hierbei sind sowohl gerade Verbindungen (horizontal beziehungsweise vertikal nebeneinander) als auch diagonale Verbindungen über alle acht direkt angrenzenden Felder zulässig. Am einfachsten lassen sich Gebiete um die Eckpunkte des Spielfeldes abtrennen. Etwas schwerer zu erobern sind Gebiete am Spielfeldrand abseits der Ecken. Am meisten Steine benötigt das Erobern von Gebieten im Feldinneren, da in diesem Fall zum Erobern ein geschlossener Kreis mit Steinen um das zu erobernde Gebiet gebildet werden muss.

#### Fangen von gegnerischen Steinen

Jede/r Spieler/in kann einzelne Steine oder Gruppen von Steinen des Gegners „fangen“. Dazu müssen sämtliche „Freiheiten“ dieser gegnerischen Steine mit den eigenen Steinen besetzt werden. Der Begriff „Freiheiten“ bezeichnet dabei alle unbesetzten benachbarten Schnittpunkte eines Steines entlang jeweils einer der beiden Gitterlinien (siehe Abbildung 2.8 rechts, gekennzeichnet durch blaue Markierungen). Diagonal liegende Schnittpunkte zählen daher nicht zu den Freiheiten. Um einen einzelnen Stein im Spielfeldinneren zu fangen, sind daher vier Steine notwendig. Um eine Gruppe zu fangen, müssen alle Freiheiten der zur Gruppe gehörenden Steine besetzt werden. Eine Gruppe bilden Steine einer Farbe miteinander, wenn sie jeweils eines der zu den Freiheiten des anderen Steines gehörenden Schnittpunkte besetzen. Dementsprechend zählen auch hier also diagonale Verbindungen nicht dazu, anders als bei der Kettenbildung zum Erobern von Gebieten. Wird ein Stein oder eine Gruppe von Steinen gefangen, werden diese vom Spielfeld genommen und dem/der Spieler/in der anderen Farbe als „Gefangene“ übergeben.

Es ist jedem/r Spieler/in frei, seinen Stein auf jedem beliebigen unbesetzten Schnittpunkt zu platzieren, mit den folgenden Einschränkungen:

- Opfer-Züge sind nicht erlaubt. Es dürfen keine Steine so gesetzt werden, dass sie unmittelbar vom Gegner gefangen werden.
- Sonderfälle (siehe Unterabschnitt 2.6.2) schränken die Zugfreiheit unter Umständen ein.

Der/Die Spieler/in, der am Ende des Spiels mehr Punkte gesammelt hat, gewinnt. Die Punkte werden aus folgenden vier Quellen aufsummiert:

1. Gefangene gegnerische Steine geben jeweils einen Punkt.
2. Eroberte Felder (Schnittpunkte) geben jeweils einen Punkt.
3. „Tote“ Steine auf dem Spielfeld, über deren Zustand sich beide Spieler/innen einig sind, werden jeweils mit einem Punkt dem/r Spieler/in der anderen Farbe zugeschrieben. Als tote Steine werden typischerweise solche Steine betrachtet, die vereinzelt in Gebieten liegen, die vom Gegner kontrolliert werden. Sie haben keine Möglichkeit, eigene Gebiete zu erobern oder noch Gruppen des Gegners zu schlagen.
4. Der/Die Spieler/in mit den weißen Steinen erhält einen Punktebonus von üblicherweise 6,5 oder 7,5 Punkten, da der/die Spieler/in mit den schwarzen Steinen den ersten Zug macht und dadurch einen Vorteil genießt. Diesen Bonus nennt man »Komi«. Der halbe Punkt dient dem Zweck der Verhinderung eines Unentschiedens.

### 2.6.2 Sonderfälle

**„Ko“:**
Während der Partie können sich Anordnungen ergeben, bei denen es beiden Spielern/innen möglich ist, im ewigen Kreislauf durch Wiederholung des gleichen Zuges abwechselnd einen Stein des Gegners zu erobern. Um dies zu verhindern, wurde eine Regel eingeführt, die besagt, dass vor einer Zugwiederholung ein Stein anderswo gesetzt werden muss. Mit anderen Worten: es ist untersagt, durch das Platzieren eines Steines an gleicher Stelle wie im Zug zuvor die gleiche Spielbrettanordnung wiederherzustellen wie im Zug zuvor.

**„Tsumego“ (Leben und Tod):**
Angenommen es ergibt sich eine Situation, in der eine Farbe (A) eine Gruppe bildet, die ringförmig geschlossen ist, und die andere Farbe (B) einen größeren Ring unmittelbar drumherum baut. Schafft es nun Farbe A, das Gebiet im Inneren des eigenen Ringes in 2 „Augen“ zu unterteilen, also 2 voneinander abgetrennte leere Schnittpunkte zu entwickeln, so ist die Gruppe der Farbe A unfangbar. Schafft es Farbe A hingegen nicht, zwei abgetrennte Augen zu bilden, so ist die Gruppe als gefangen zu betrachten. Zwar besagt die Regel gegen Opferzüge, dass Steine nicht so gesetzt werden dürfen, dass sie nach dem Setzen unmittelbar keine Freiheiten haben, jedoch ist es Farbe B möglich, durch das Besetzen des letzten Gebietes im Inneren des Ringes der Gruppe A sämtliche Freiheiten der Gruppe A zu besetzen und damit Gruppe A zu fangen. Da Gruppe A infolge dessen vom Brett genommen wird, hat der frisch gesetzte Stein von B Freiheiten dort, wo vorher Steine von A saßen.

**„Seki“:**

Seki beschreibt eine Pattsituation, in der zwei ungleichfarbige Gruppen so miteinander angeordnet sind, dass sie jeweils nicht als gefangen betrachtet werden können, weil sie noch Freiheiten besitzen. In einem Seki kann keine der beiden Farben die andere Gruppe angreifen, ohne dabei selbst vorher gefangen zu werden. Die Anordnung ist daher im Interesse beider Spieler/innen unbeweglich und nicht eindeutig zugunsten einem/r der beiden Spieler/innen zu werten.

# 3 Verwandte Arbeiten

Zur Betrachtung verwandter Arbeiten lässt sich die von dieser Arbeit behandelte Problematik in drei Themengebiete unterteilen. Zunächst spielt die Analyse von Eye-Tracking-Daten eine Rolle, da die Entwicklung eines Werkzeugs zur Unterstützung ebendieser den zentralen Aspekt dieser Arbeit verkörpert. Dabei geht es insbesondere um eine Analyse über kooperatives und kompetitives Spiel, für diesen Abschnitt etwas verallgemeinert unter dem Begriff „kollaboratives Arbeiten und Lernen“. Ein spezifisches Problem, das für die gepaarte Analyse von in kollaborativem Kontext erhobenen Eye-Tracking-Daten zu lösen ist, stellt letztlich das Mappen von Gaze-Daten dar. Einige der hier aufgelisteten Arbeiten fallen zwischen zwei der nachfolgenden Sektionen. Sie wurden gegebenenfalls in die jeweils prävalente Rubrik aus Perspektive dieser Arbeit eingeordnet und mit einer Notiz versehen.

## 3.1 Analyse von Eye-Tracking-Daten beim Lösen von Problemen

Mallick *et. al* [MST+16] untersuchen Eye-Tracking-Metriken als Indikatoren für kognitive Auslastung bei Videospielen. In ihrer Publikation werden sowohl Eigenschaften aufgezeichneter Fixationen und Sakkaden, darunter Dauer der Fixationen und Blickgeschwindigkeit bei Sakkaden, als auch andere Merkmale wie Pupillengröße und Blinzeldauer auf Zusammenhänge mit kognitiver Auslastung analysiert. Die Autoren stellen eine umgekehrte Relation zwischen Fixations- sowie Blinzeldauer zur kognitiven Auslastung und einen direkten Zusammenhang zwischen sakkadischer Höchstgeschwindigkeit sowie Pupillengröße mit der kognitiven Auslastung fest.

Li *et. al* [LNJ11] erkunden persönliche Aspekte der Probanden/innen durch Analyse des Blickverhaltens bei diversen Tetris-Partien. Ziel ist es zu untersuchen, wie eindeutig und identifizierbar das Blickverhalten jedes/r Probanden/in während der Informationsaufnahme in einer dynamischen Umgebung ist. Zu den Ergebnissen zählt ein Modell, das mehrere Rohcharakteristiken und dynamische Eigenschaften berücksichtigt und darüber eine Identifikationsrate von 82,1% nach etwa 30 Sekunden andauernder Datenerfassung erzielt.

In der Studie von Shimizu *et. al* [SKKH14] vergleichen die Autoren das Blickverhalten von Anfängern/innen und Experten/innen beim Betrachten und Antizipieren des Rückschlags des/der Gegenspielers/in in aufgezeichneten Tennis-Matches. Die Probanden/innen werden vor der Aufzeichnung angewiesen, sich mental auf einen von zwei spezifischen Schlägen vorzubereiten. Aus den erhobenen Daten lässt sich unter anderem eine Tendenz der Anfänger/innen ermitteln, unmittelbar nach dem Schlag kurz unbewusst dem Ball zu folgen. Außerdem stellen die Autoren Indikatoren für ein trainiertes Blickverhalten bei den Experten/innen fest, deren Gaze-Punkte im Schnitt weniger von denen der anderen Experten/innen abweichen als das zwischen den Gaze-Punkten der Anfänger/innen der Fall ist. Diese Publikation ließe sich eventuell auch in Abschnitt 3.2 einordnen, da zwei unterschiedliche Gruppen von Probanden/innen miteinander verglichen werden.

Ein ähnliches Thema behandeln Liu *et. al* [LHL+09]. In kollaborativen Anwendungen soll zwischen jeweils zwei Probanden/innen der/die sachverständigere identifiziert werden. Die Autoren/innen nutzen dafür hidden Markov-Modelle, die über einen Teil der erhobenen Daten trainiert und dann genutzt werden, um die restlichen Daten zu klassifizieren. Es wird untersucht, wie akkurat das Modell je nach Größe des Trainingssatzes Voraussagen treffen kann und wie bald eine Einschätzung der Expertise erfolgen kann. Dabei gelingt bereits nach 5% der Gesamtdauer des Experiments eine akkurate Vorhersage des Levels der Expertise. Diese Publikation ließe sich auch in Abschnitt 3.2 einordnen, da jeweils ein Direktvergleich zwischen zwei Probanden/innen stattfindet.

Frutos-Pascual *et. al* [FGM15] untersuchen das Blickverhalten einer Gruppe jüngerer Probanden/innen (im Schnitt etwa zehn Jahre alt) auf Usability. In vier Stufen sollen die Probanden/innen verschiedene Puzzles lösen, während besonders auf Merkmale wie die Anzahl der Fixationen pro Level und Nutzer/in und die Koordinaten der allerersten Fixation pro Level und Nutzer/in geachtet wurde. Die ermittelten Ergebnisse geben Aufschluss darüber, wo etwa bei der Gestaltung grafischer Nutzeroberflächen am besten eine wichtige Information zu platzieren wäre, und deuten darauf hin, dass Probanden/innen mit steigendem Schwierigkeitsgrad des Puzzles eine größere Anzahl an Fixationen ausüben und länger nachdenken.

Silva Junior *et. al* [RHTE18] analysieren gleichzeitig den Fokus der visuellen Aufmerksamkeit und die Gehirnaktivität (mittels Elektroencephalographie) bei Schach-Zügen zwischen Spielern/innen mit unterschiedlich stark ausgeprägten Spielkenntnissen. Den Probanden/innen wurden eine Reihe von Fragen zum Spiel gestellt und die Leistungen wurden nach Korrektheit und Antwortzeit ausgewertet. Die Publikation stellt signifikante Unterschiede zwischen Experten/innen und Laien fest, darunter eine Tendenz der Laien, verstärkt unwichtige Bereiche des Spielbretts zu fixieren, länger nachzudenken und (gemäß EEG) mehr visuelle Informationen im Gehirn zu verarbeiten, während die Experten/innen häufiger relevante Bereiche fixieren und verstärkt Gehirnaktivität in Arealen mit Bezug zur Entscheidungsfindung und Planung aufweisen.

## 3.2 Kollaboratives Arbeiten und Lernen

Mit *Eye-Write* stellen Kütt *et. al* [KLHP19] ein System zur Erweiterung eines kollaborativen Remote-Echtzeit-Texteditors um den Faktor einer gegenseitigen Gaze-Awareness vor. Über das System können zwei Autoren, die zeitgleich an einem Text arbeiten, in Echtzeit sehen, wo das Gegenüber gerade hinschaut. Um den Effekt eines solchen Add-Ons auf die Qualität der Kollaboration zu messen, wird das System in einer Studie mit 20 Autorenpaaren getestet. Zu den Ergebnissen zählt die Erkenntnis, dass der Grad der Kollaboration zwischen Autoren/innen mit aktivierter Gaze-Visualisierung größer ist als bei Paaren, die den Editor ohne dieses Feature nutzen. Außerdem lassen sich erhöhtes gegenseitiges Einverständnis, erhöhte gemeinsame Aufmerksamkeit, ein besserer Kommunikationsfluss, verbesserte Überwindungsfähigkeit sowie verbessertes Bewusstsein um die Aktivitäten des/der Co-Autors/in feststellen.

Harrer *et. al* [HSSK15] untersuchen, wie stark verschiedene Arten der gegenseitigen Gaze-Awareness zwei Probanden/innen in einem kollaborativen Problemlösungsszenario dabei unterstützen können, zu lernen, voneinander zu lernen. Dies geschieht unter dem pädagogischen Ansatz L2L2–„Learning to Learn together“ mit dem Ziel, die Fähigkeit bei Schülern/innen und Studierenden zu fördern, gemeinsam und voneinander zu lernen. Die Probanden/innen müssen gemeinsam ein Puzzle lösen und haben dabei entweder keinerlei Unterstützung durch Gaze-Awareness, einen Gaze-Cursor, der

visualisiert, wo der Blick des Partners / der Partnerin ruht, oder bekommen das Element des Puzzles, das das Gegenüber gerade betrachtet, hervorgehoben. Dabei zeichnen sich Verhaltensmuster zur gegenseitigen Interaktion und Kommunikation ab, die durch Gaze-Awareness unterstützt werden.

## 3.3 Mapping von Gaze-Daten

Diese Arbeit implementiert Mapping von Gaze-Daten in mehreren Vorgängen in einem spezifischen Anwendungsfall. Dennoch gibt es hier eine Arbeit, deren Thematik etwas entfernter mit der hier behandelten verwandt ist.

Kang *et. al* [KB14] beschreiben ein Konzept zum Mappen von Eye-Tracking-Daten im Kontext beweglicher Ziele auf Computerbildschirmen. Dazu gehören eine Methode, Fixationen auf bewegliche Ziele zu mappen und ein Algorithmus, der „zielbasiertes“ Mapping von Eye-Tracking-Daten unter Berücksichtigung der objektspezifischen Fixationsreihenfolge und -dauer durchführt. Dadurch können die zur Analyse auszuwertenden Areas of Interest (AOIs) im Wesentlichen auf *Objects of Interest* aggregiert werden.

# 4 Aufgabenstellung und Konzept

In diesem Kapitel wird die Aufgabenstellung für diese Arbeit wiedergegeben, basierend darauf ein Anwendungsfall formuliert und das dazu erarbeitete Konzept präsentiert.

## 4.1 Aufgabenstellung

Im Rahmen dieser Arbeit soll ein Konzept zur gebündelten Analyse von Eye-Tracking-Daten und Screen Recordings mehrerer Spieler aus gemeinsamen kooperativen Spielszenarien entwickelt und implementiert werden. Als Teil der Aufgabenstellung werden Datensätze bereitgestellt, die im Vorfeld aufgezeichnet wurden. Diese Datensätze enthalten Eye-Tracker-Daten und Screen Recordings zweier Spieler, die gegeneinander Partien des chinesischen Brettspieles Go an zwei Computern austrugen. Die Spieler sind ausgerüstet gewesen mit jeweils einem stationären Eye-Tracker, der das Blickverhalten des zugehörigen Spielers während der Partien aufzeichnete. Bei den zwei verwendeten Geräten handelt es sich um unterschiedliche Modelle. Mehr Informationen dazu in Unterabschnitt 4.1.1. Die zu entwickelnde Software soll in der Lage sein, Datensätze beider Geräte einzulesen und insoweit zu synchronisieren, dass relevante Informationen aus beiden Perspektiven gemeinsam auf einem einzigen Screen Recording dargestellt und analysiert werden können. Ferner soll das Programm sinnvolle Visualisierungstechniken zur Darstellung der eingelesenen Daten implementieren. Die Aufgabenstellung setzt außerdem das Arbeiten mit den mitgelieferten Datensätzen voraus, das Aufzeichnen weiterer Datensätze ist optional.

Folgende Aufgaben werden gestellt:

- Die **Fusion der Daten** beider Geräte mit speziellem Fokus auf Synchronisation, Behandlung hardwarespezifischer Eigenheiten der zwei verwendeten Geräte und dem Kombinieren mehrerer Aufzeichnungen eines Spiels.
- Eine **Transformation der Koordinaten** der Gaze-Daten zwischen den Screen Recordings beider Eye-Tracker, sodass die Datensätze bi-direktional auf jeweils eines der Screen Recordings gemappt werden können. Eine Notwendigkeit hierfür begründet sich durch die Tatsache, dass die Monitore und somit die Screen Recordings der zwei Geräte unterschiedliche Auflösungen und Seitenverhältnisse aufweisen.
- Das **Visualisieren** der transformierten Daten im Kontext der Bildschirmaufzeichnungen der ausgetragenen Partien. Dies soll durch eine Implementierung einfacher Visualisierungstechniken wie Gaze Plots und Heat Maps gelingen.

### 4.1.1 Verwendete Hardware

Wie bereits in Unterabschnitt 2.1.2 erwähnt, kamen bei der Aufzeichnung der Datensätze für die Analyse zwei optische Eye-Tracker zum Einsatz. Es handelt sich um zwei unterschiedliche monitorbasierte Geräte der Marke Tobii, die sich besonders in den Punkten Bildseitenformat und Gaze-Recording-Frequenz (GRF), also der Aufzeichnungsfrequenz, voneinander unterscheiden:

**Gerät 1** hat eine Bildschirmauflösung von 1920x1080 Pixeln und unterstützt Aufzeichnungen mit mindestens 60 Hz und bis zu 1200 Hz GRF. Gemessene Blickrichtungen des/der Betrachters/in sind bis auf maximal 0,3° Abweichung genau und der maximale unterstützte Blickwinkel[1] beträgt 30°. Der empfohlene Abstand zum Display beträgt 55 bis 75 Zentimeter. Es handelt sich bei diesem Gerät um ein Exemplar des „Tobii Pro Spectrum 1200“[2].

**Gerät 2** hat hingegen eine Bildschirmauflösung von 1920x1200 Pixeln und eine GRF von 60 Hz. Es misst Blickrichtungen des/der Betrachters/in mit einer maximalen Abweichung von 0,5° genau und unterstützt Blickwinkel von bis zu 42° relativ zur Mitte des Sensors. Der empfohlene Abstand zum Display beträgt 50 bis 80 Zentimeter. Bei diesem Gerät handelt es sich um ein Exemplar des Modells „Tobii T60XL“[3].

Beide Geräte unterstützen *Bright* und *Dark Pupil Tracking* wie in Unterabschnitt 2.1.2 beschrieben. Da die Gaze-Recovery-Time (GRT), also die benötigte Zeit zur Wiederfindung des Blickes des/r Probanden/in nach Verlust der Augenposition, bei **Gerät 2** etwas höher ist als bei **Gerät 1** (ca. 300 ms vs. unter 150 ms), kam zum Tracking mit **Gerät 2** zusätzlich eine am Tisch fixierte Kopfstütze zum Einsatz, die einen konstanten Abstand des/r Probanden/in vom Display gewährleisten soll und gleichzeitig hektische und grobe Veränderungen der Position der Augen verhindert.

## 4.2 Anwendungsfall

Der folgende Anwendungsfall für das zu entwickelnde Programm beschreibt die angestrebten Ziele der Entwicklung aus Perspektive eines potenziellen Anwenders. Er dient in der abschließenden Auswertung (siehe Unterabschnitt 6.2.1) dem Zweck der Verifikation, dass das Ziel der Entwicklung erreicht wurde.

Als Nutzer möchte ich mit dem Programm die Datensätze von zwei unterschiedlichen Eye-Trackern kombinieren können. Das Programm soll in der Lage sein, alle nötigen Informationen aus den exportierbaren Dateien der zu den Trackern gehörenden Softwares zu lesen. Außerdem soll das Programm auch die Bildschirmaufzeichnungen darstellen und abspielen können, die während des Trackings mitgeschnitten wurden. Auf diesen Videos soll dargestellt werden, wohin die Probanden/innen zum Zeitpunkt der Aufnahme auf dem Bildschirm geschaut haben. Zum Zwecke des Direktvergleichs sollen jedoch nicht einfach beide Perspektiven nebeneinander dargestellt werden, sondern das Blickverhalten beider Probanden/innen soll auf einem einzigen Video visualisiert werden, das von einem/r der beiden Probanden/innen stammt. Hierfür möchte ich das zu nutzende

[1] maximale Abweichung von der Blickrichtung, die orthogonal zur Bildfläche / zum Bildschirm verläuft

[2] https://www.tobiipro.com/product-listing/tobii-pro-spectrum/#Specifications (Zul. geprüft: 03.09.2019)

[3] https://www.tobiipro.com/siteassets/tobii-pro/product-descriptions/tobii-pro-tx-product-description.pdf/?v=1.0 (Zul. geprüft: 03.09.2019)

Video nach Belieben wählen können. Das heißt, dass der Kombinationsvorgang bidirektional zwischen den Eye-Trackern funktionieren soll. Die Aufzeichnungen, die vom Programm kombiniert und transformiert werden sollen, entstammen einigen Partien des Strategie-Brettspiels Go. Primär soll es also alle nötigen Transformationen auf den Daten ausführen, um das Blickverhalten der beiden Spieler/innen während des Spiels gemeinsam auf einem Spielbrett darstellen zu können und vergleichbar und somit analysierbar zu machen. Neben dem Video mit der Darstellung des Blickverhaltens sollen zu Analysezwecken noch zusätzliche Visualisierungstechniken verfügbar sein wie beispielsweise Heat Maps. Diese sollen mir dabei helfen, die Eye-Tracker-Recordings einfach auf unterschiedliche Eigenschaften hin zu betrachten, darunter etwa ob die Probanden/innen häufiger an die gleiche Stelle auf dem Spielbrett geschaut haben oder wie lange im Verhältnis jede/r Spieler/in wohin geschaut hat.

## 4.3 Konzept

Im Folgenden wird der geplante Workflow zur Nutzung des Programms beschrieben und die nötigen Transformationsschritte in Form einer Pipeline vorgestellt, die von den Eye-Tracker-Datensätzen durchlaufen werden soll.

### 4.3.1 Workflow

Der Einstiegspunkt der grafischen Oberfläche wird verkörpert durch ein Menü, über das der Nutzer ein neues Projekt erstellen oder ein vorhandenes Projekt laden kann. Für die Erstellung eines neuen Projekts wird zunächst das gewählte „Host“-Screen-Recording über einen Dateiauswahldialog erfragt. Wählt der Nutzer eine Datei aus, gelangt er in die Hauptansicht. In dieser Ansicht kann das Screen Recording abgespielt werden. Zudem kann die Geschwindigkeit der Video-Wiedergabe über das Menü auf fünf Stufen reguliert werden. Über das Menü der Hauptansicht kann der Nutzer zunächst den Host-Datensatz zum gewählten Screen Recording laden. Sobald dies geschehen ist, beginnt das Programm damit, den Gaze Plot der „Host“-Daten zu rendern. Nun ist es dem Nutzer möglich, nach Belieben weitere sogenannte „Gast“-Datensätze einzubinden, die nach einer Synchronisation und einer Transformation gleich dem Host-Datensatz dargestellt werden. Dieser Prozess wird ebenfalls über das Menü gestartet und begleitet durch einen Wizard, der vom Nutzer in vier Schritten alle nötigen Informationen erfragt. Sind zwei oder mehr Recordings geladen, wird neben dem Gaze Plot noch ein Scarf Plot zur Visualisierung der Distanzen zwischen den Gaze-Punkten unter der Zeitachse dargestellt. Zu jedem Zeitpunkt kann der Nutzer über das Menü der Hauptansicht das aktuelle Projekt speichern. Gespeicherte Projekte können über das Einstiegsfenster geladen werden. Dieser Vorgang stellt den gespeicherten Zustand inklusive relevanter Parameter der geladenen Gast-Recordings wieder her, solange die ursprünglich geladenen Dateien noch an gleicher Stelle im Dateisystem vorzufinden sind. Ist dies nicht der Fall, so wird der Nutzer für jede betroffene Datei gebeten, über einen Dateiauswahldialog einen neuen Pfad anzugeben. Siehe Abbildung 4.1 für eine grafische Darstellung des Workflows.

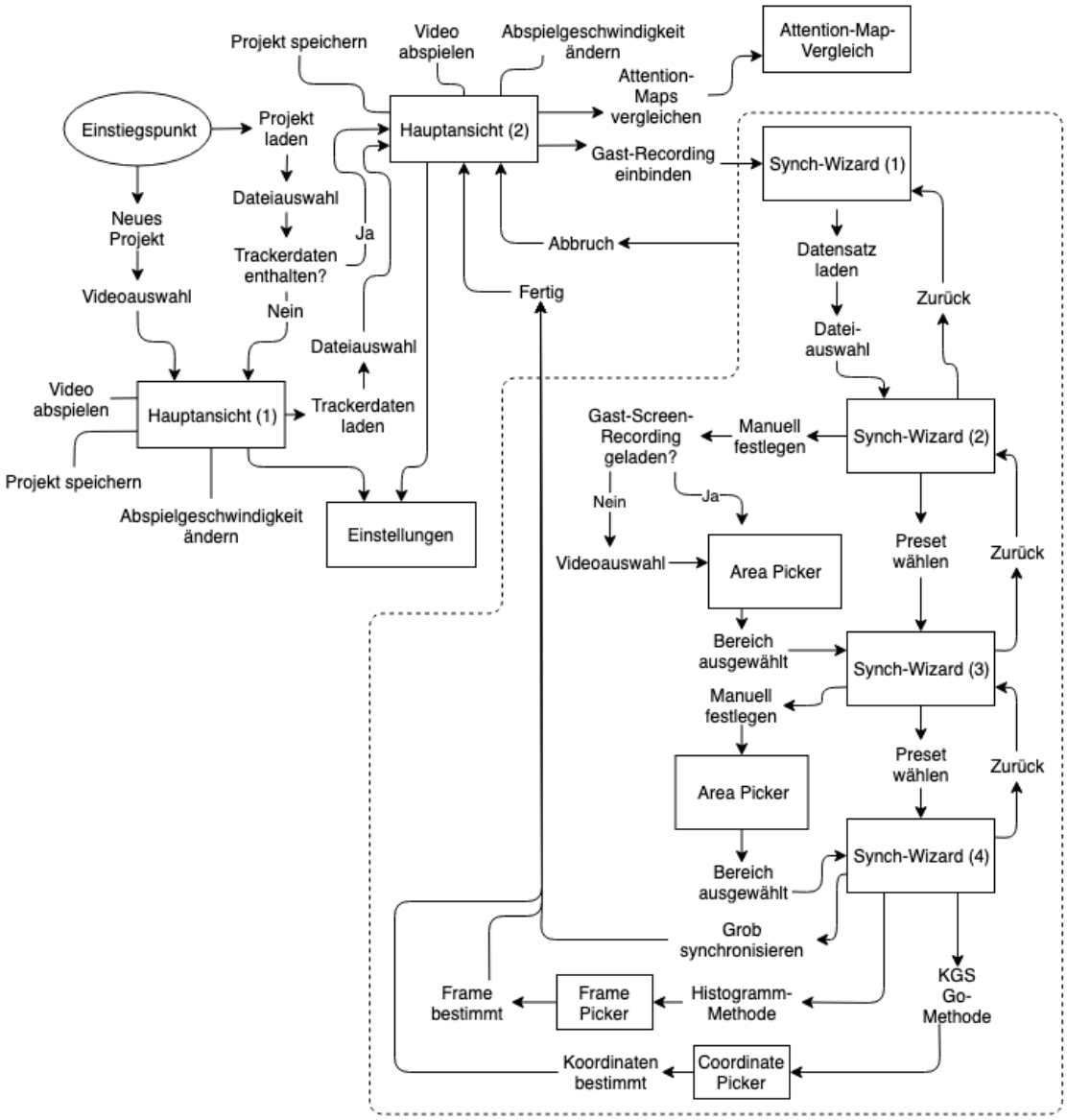



**Abbildung 4.1:** Workflow der Programm-Nutzung. Hauptansicht (1): Kein Host-Datensatz geladen (nur Screen Recording). Hauptansicht (2): Host-Datensatz geladen, Einbinden weiterer Datensätze möglich. Synch-Wizard (1): Gast-Datensatz laden. Synch-Wizard (2): Spielfeld festlegen im Gast-Recording. Synch-Wizard (3): Ziel-Spielfeld festlegen im Host-Recording. Synch-Wizard (4): Methode zur zeitlichen Synchronisation festlegen. Mehr Informationen in Unterabschnitt 4.3.1.

### 4.3.2 Daten-Pipeline

Die Datensätze der Eye-Tracker liegen als Dateien im TSV-Format (Unterabschnitt 5.5.1) vor. Das Ziel ist die Überführung in eine Datenstruktur, deren Inhalte flexibel mittels unterschiedlicher Techniken visualisiert werden können. Außerdem sollen mehrere Eye-Tracker-Recordings, falls vorhanden, nach dem Durchlaufen der Pipeline bereits miteinander kontextuell verknüpft sein,

sodass erzeugte Visualisierungen über dieselben in einem gemeinsamen zeitlichen und räumlichen Kontext stehen. Im ersten Schritt wird die Datei eingelesen. Das Resultat ist ein String-Array über die Inhalte der Datei, nach Zeilen gespalten. Der zweite Schritt ist ein Parsing-Vorgang, bei dem das korrekte Parser-Modul für den Datensatz gewählt (siehe Unterabschnitt 5.5.2) und die Datenstruktur erzeugt wird. Anschließend durchläuft die erzeugte Datenstruktur im dritten Schritt einen Filteralgorithmus, der die eingelesenen Rohdaten der Gaze-Punkte zu Fixationen und Sakkaden aggregiert (siehe Abschnitt 5.6). Im Falle des ersten geladenen Datensatzes (des „Host-Recordings“), zu dem auch die dargestellte Bildschirmaufzeichnung gehört, ist der Vorgang an dieser Stelle abgeschlossen. Fremde Datensätze, wie etwa solche, die die Perspektive eines/r anderen Spielers/in repräsentieren, durchlaufen in einem vierten Schritt eine Koordinatentransformation, bei der die Koordinaten der Gaze-Daten dahingehend angepasst werden, dass sie korrekt auf das Spielfeld in der Bildschirmaufzeichnung des Host-Recordings mappen. Nach Abschluss dieses Schrittes besteht eine räumliche Verknüpfung zwischen dem Host-Recording und dem Gast-Recording. Im letzten Schritt wird die zeitliche Verschiebung zwischen dem Gast-Recording und dem Host-Recording ermittelt, sodass Gaze-Events beider Recordings, die im Kontext des aufgezeichneten Spiels von den Tracker-Geräten zum gleichen Zeitpunkt erfasst wurden, den gleichen Zeitstempel zugewiesen bekommen. Dieser Vorgang ermöglicht das asynchrone Aufzeichnen der verschiedenen Perspektiven bzw. eliminiert die Notwendigkeit eines gleichzeitig erfolgenden Aufnahmestarts zwischen den Spielern/-innen. Hierfür stehen drei Synchronisationsmethoden zur Verfügung (siehe Unterabschnitt 5.8.2). Ist dies erfolgt, besteht eine zeitliche Verknüpfung zwischen dem Host-Recording und dem Gast-Recording und eine gemeinsame Visualisierung kann erfolgen. Für eine grafische Darstellung der Pipeline, siehe Abbildung 4.2.

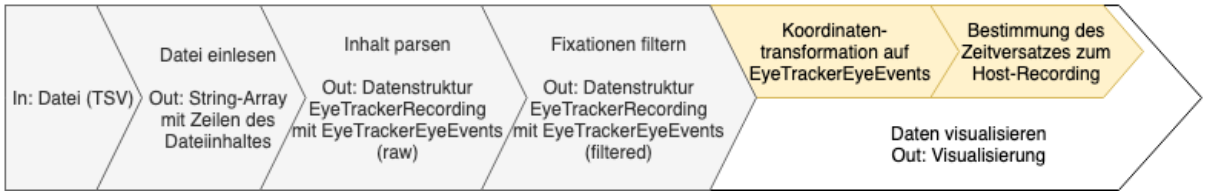



**Abbildung 4.2:** Pipeline der Eye-Tracker-Datensätze. Gelb hinterlegte Schritte betreffen ausschließlich Datensätze, die im Kontext einer fremden Bildschirmaufzeichnung visualisiert werden sollen („Gast-Datensätze“). Beschreibung in Unterabschnitt 4.3.2.

# 5 Implementierung

Dieser Abschnitt schildert den Implementierungsprozess und entwicklungsspezifische Merkmale des im Rahmen dieser Arbeit entstandenen Programms. Dabei wird auch erklärt, wie die zur Zusammenführung zweier oder mehrerer Eye-Tracker-Recordings zu lösenden Probleme behandelt wurden–darunter die Transformation der Koordinaten zwischen den Bildschirmen der Eye-Tracker und die Verfahren zur zeitlichen Synchronisation der Datensätze. Zur Entwicklung wurde die Programmiersprache Java verwendet. Als Projektentwicklungsstrategie wurde ein iteratives Vorgehen angewandt.

## 5.1 Gesamtübersicht der grafischen Bedienelemente

In diesem Abschnitt werden die primären grafischen Bedienelemente des Programms vorgestellt.

### 5.1.1 Hauptansicht

Abbildung 5.1 zeigt die Hauptansicht mit farblich gekennzeichneten Bereichen. Die Hauptansicht setzt sich aus den folgenden vier Bereichen zusammen: dem Video-Panel (A, Grün), der Menüleiste (B, Orange), der Zeitleiste mit Playback-Controls, MinDist-Plot, Frame Counter und Auswahlleiste für den Bereich, über den die Attention Maps generiert werden sollen (C, Blau) sowie einer Schnelleinstellungsleiste mit Informationen zu den Spielerfarben und einer Farblegende zum MinDist-Plot (D, Pink).
Das **Video-Panel** rendert die Frames des geladenen (Host-)Screen-Recordings und visualisiert die geladenen Trackerdaten. Zur Wahrung der Übersicht wurde das Video in der Abbildung herausgeschnitten, damit keine Missverständnisse darüber entstehen, welche sichtbaren GUI-Elemente zum Programm gehören und welche Teil des Screen-Recordings sind. In der **Menüleiste** finden sich einige Untermenüs, über die Datensätze geladen, das Projekt gespeichert, die Abspielgeschwindigkeit des Videos reguliert, ein Tool zum Vergleichen der Attention Maps und die Einstellungen geöffnet werden können. Die **Zeitleiste** ermöglicht das Springen im Video. Der **Play-/Pause-Button** auf der linken Seite pausiert bzw. setzt die Wiedergabe fort, der **Frame Counter** darunter zeigt den Zähler des aktuell angezeigten Frames im Video an und erlaubt das gezielte Springen im Video über ein Dialogmenü, das über einen Mausklick auf den Counter geöffnet werden kann. **Mindestdistanz-Scarf-Plots** (Unterabschnitt 5.7.3) werden unter der Zeitleiste gezeichnet, wenn mehr als ein Datensatz geladen ist. Der zweite **Slider mit zwei Griffen** unter der Zeitleiste bestimmt den Abschnitt des Videos (und des Datensatzes), über den die Attention Maps der Spieler/innen generiert werden soll.

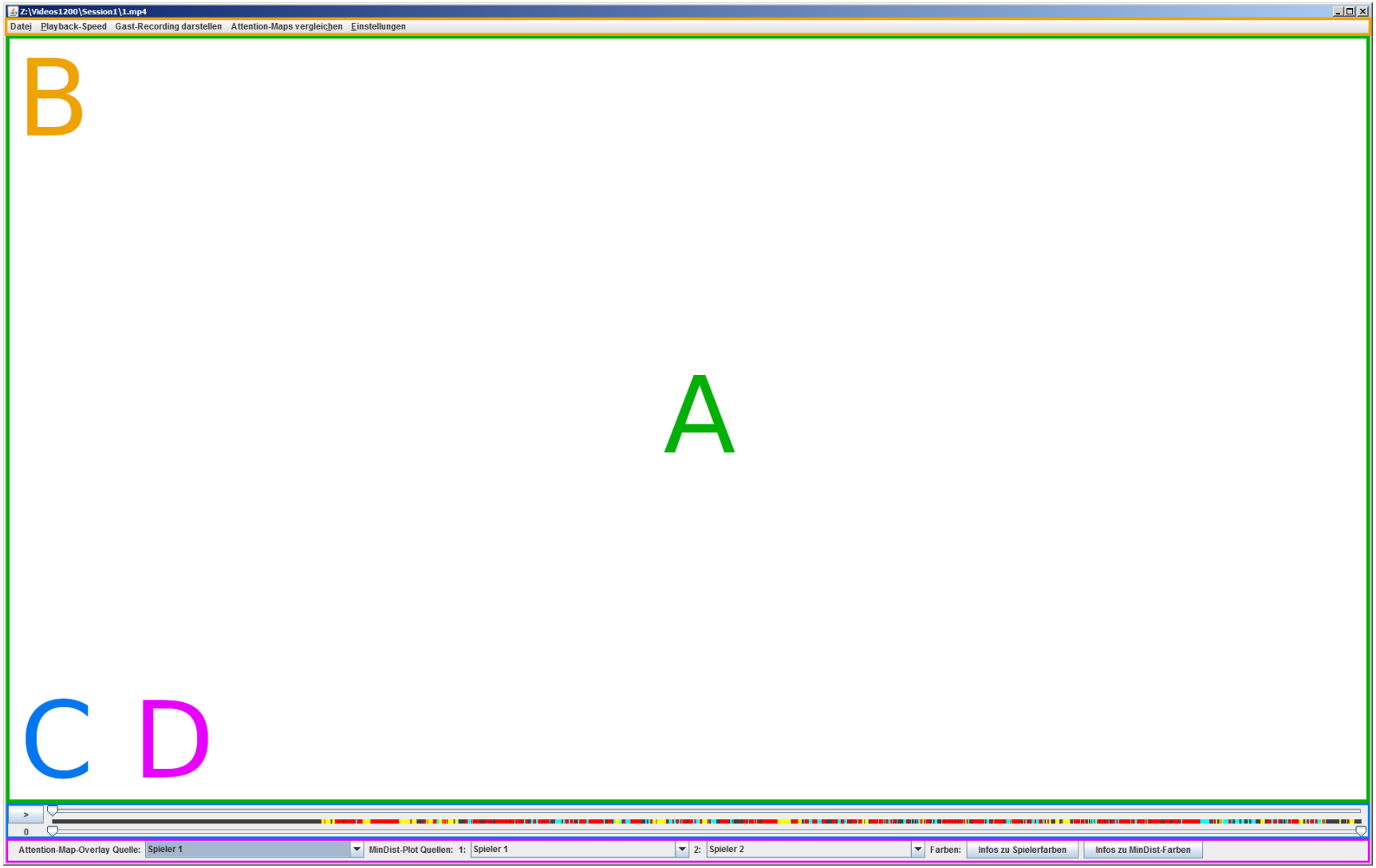



**Abbildung 5.1:** Die Hauptansicht des Programms. Bereiche: A (Grün): Video-Panel (dargestelltes Video wurde zur Wahrung der Übersicht herausgeschnitten); B (Orange): Menüleiste; C (Blau): Play-/Pause-Button, Zeitleiste, MinDist-Plot, Frame Counter, Bereichsauswahl für Attention-Map-Generator; D (Pink): Schnelleinstellungen  Informationen

### 5.1.2 Attention-Map-Vergleich

Abbildung 5.2 zeigt das Werkzeug für den Attention-Map-Vergleich. Nahe der linken und der rechten unteren Ecke des Fensters werden die Spieler/innen bestimmt, deren Attention Maps gemeinsam dargestellt werden sollen. Es stehen Optionen zur Verfügung, nur den Bereich des Spielfeldes darzustellen und die Maps getrennt zu normalisieren (mehr dazu in Unterabschnitt 5.7.2). Außerdem kann die Differenz der zwei Maps angezeigt werden (siehe Abbildung 5.2 B).

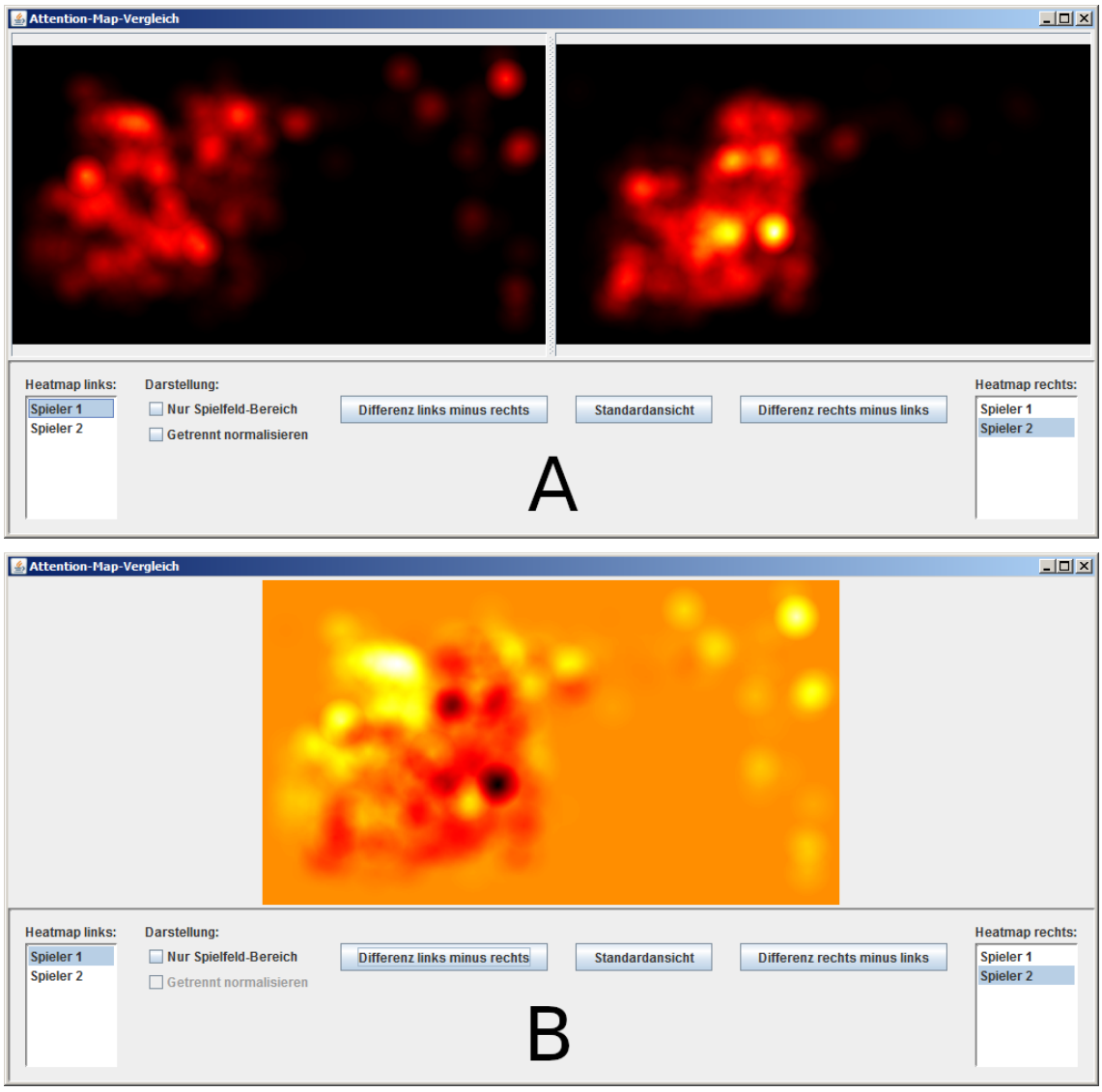



**Abbildung 5.2:** Das Attention-Map-Vergleichswerkzeug des Programms. A: Standardansicht, die Attention Maps der zwei ausgewählten Spieler/innen werden nebeneinander dargestellt. B: Ansicht der Differenz.

### 5.1.3 Einstellungen

In den Einstellungen können verschiedene Overlays für das Video aktiviert werden (siehe Abbildung 5.3 A), die Parameter für den MinDist-Plot (siehe Abbildung 5.3 B) und den Gaze-Filter (siehe Abbildung 5.3 C) angepasst werden, der Heatmap-Generator konfiguriert werden (siehe Abbildung 5.3 D) und die Metrik für den Histogramm-Vergleich in der zeitlichen Synchronisation angepasst werden (siehe Abbildung 5.3 E).

### 5.1.4 Sync-Wizard

Der Sync-Wizard (siehe Abbildung 5.4) unterstützt den/die Nutzer/in dabei, zu einem geladenen Screen-Recording mit Datensatz zusätzliche Gast-Recordings hinzuzufügen. In einem Prozess mit vier Schritten werden alle Angaben getätigt, die das Programm benötigt, um ein Gast-Recording mit dem Host-Recording zu synchronisieren. Mehr Informationen dazu in Abschnitt 5.8.

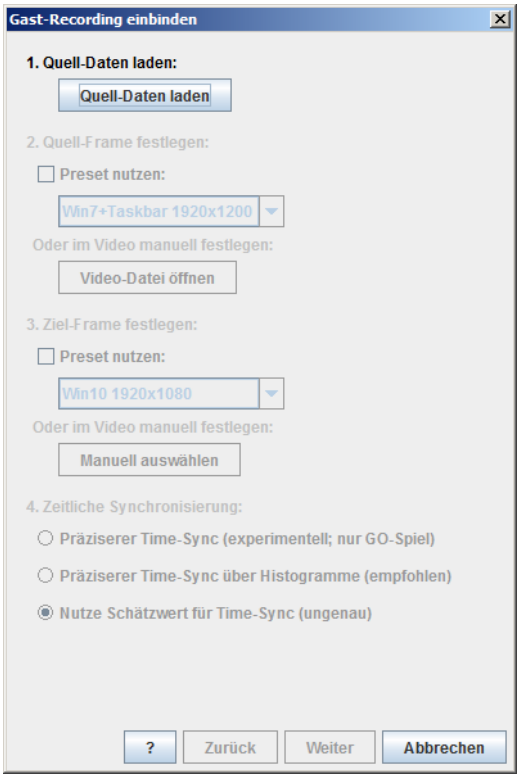



**Abbildung 5.4:** Der Sync-Wizard zur Unterstützung beim Synchronisieren von Recordings.

## 5.2 Wahl des Frameworks

Das Rendern von Videos bzw. Videoinhalten verkörpert einen elementaren Teil des Aufgabenumfangs des zu entwickelnden Programms. Um eine gute Kompatibilität mit diversen Video-Codecs und Formaten zu gewährleisten und zielgerichtet zu agieren, baut diese Arbeit die Lösungen zu den

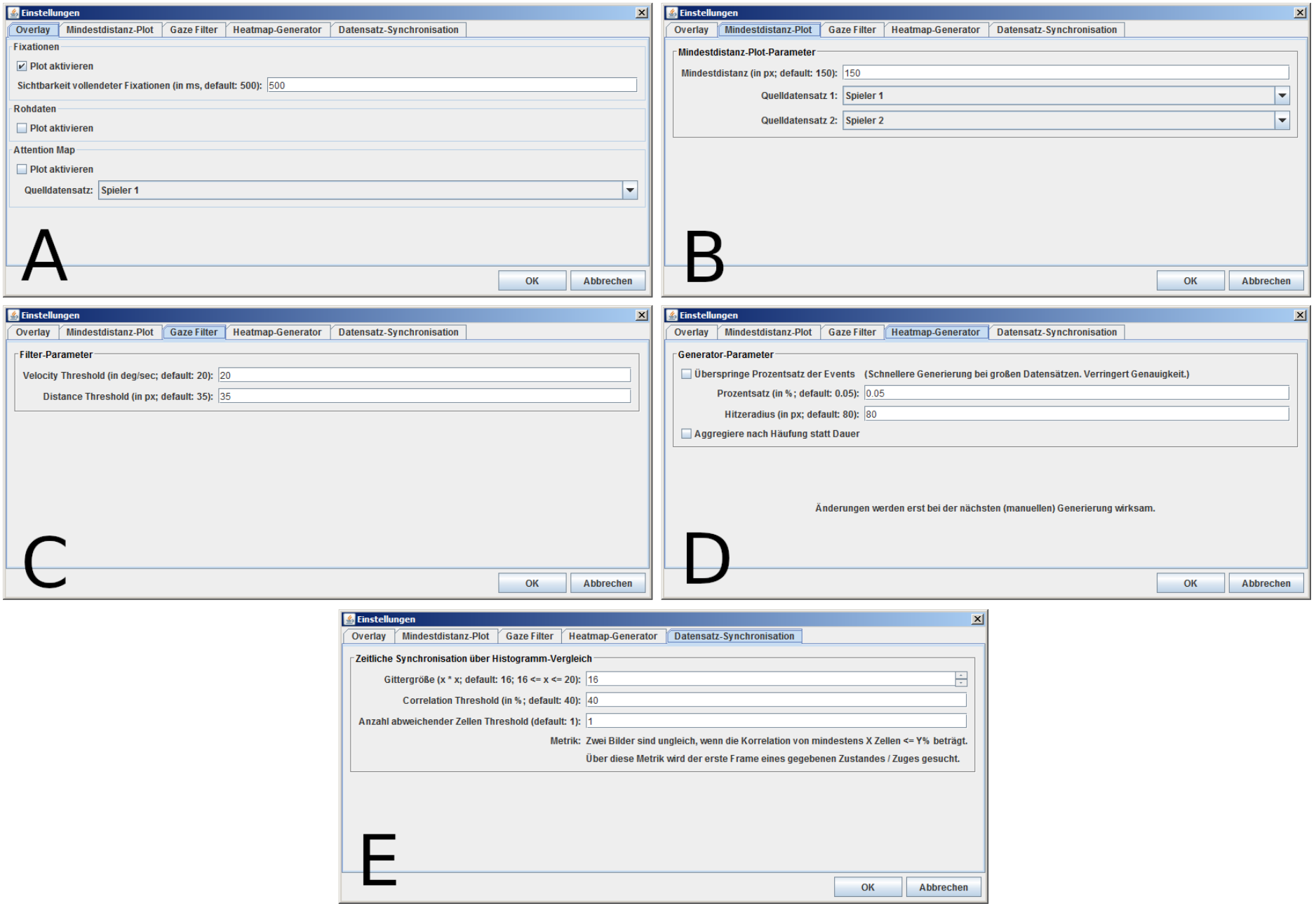



**Abbildung 5.3:** Das Einstellungs-Fenster des Programms. A: Einstellungen zum Video-Overlay. B: Einstellungen zum MinDist-Plot. C: Parameter für den Gaze-Filter. D: Konfiguration des Heatmap-Generators. E: Einstellungen für die zeitliche Synchronisation.

in Kapitel 4 formulierten Problemen um ein bereits existierendes Framework, das jene Aufgaben übernimmt, die außerhalb des Themenbereiches der Arbeit fallen. Zur Wahl standen primär zwei quelloffene Lösungen, **vlcj** und **OpenCV**, die jeweils unterschiedliche Vorteile bieten. Die Entscheidung wurde letztendlich zugunsten von OpenCV getroffen. Zur Dokumentation des Entscheidungsprozesses werden die zwei Optionen in dieser Sektion kurz erläutert.

### 5.2.1 vlcj

Bei *vlcj* handelt es sich um einen Java-Wrapper für die Kern-Bibliotheken des VLC-Mediaplayers[1]. Von Vorteil ist die ausgezeichnete Codec- und Containerformatunterstützung[2], die mit diesem Ansatz einhergeht. Der Wrapper bietet zum Rendern des Videos grundsätzlich zwei Lösungen:

**EmbeddedMediaPlayer** zur Darstellung der Videoinhalte über einen Canvas, praktisch eine fertige Leinwand, auf die die VLC-Bibliotheken zeichnen. Die Einbindung ist in diesem Fall recht einfach, aber die Möglichkeiten zur Einwirkung und Manipulation der dargestellten Bilder sind beschränkt.

**DirectMediaPlayer** zum Empfangen der Bilddaten in einem Puffer, der dann manuell auf eine Oberfläche gezeichnet werden muss. Dies bringt diverse Freiheiten mit sich, das dargestellte Bild zu manipulieren. Im Hinblick auf das Rendern von Gaze-Punkten im späteren Verlauf der Implementierung ist diese Variante daher sehr attraktiv. Auf neue Frames im Puffer wird durch ein Callback hingewiesen, über das eine Aktualisierung des dargestellten Bildes ausgelöst werden kann.

Nachteilhaft an vlcj ist im Wesentlichen nur die Tatsache, dass VLC keine präzisen Informationen zur Position eines Frames im Video liefert, wie etwa einen einzigartigen Zeitstempel oder einen Framecount. Diese Information wird allerdings benötigt, um die Bildschirmaufzeichnungen mit den Events aus den Eye-Tracker-Recordings (ETRs) zeitlich zu synchronisieren.

### 5.2.2 OpenCV

Die Open Source Computer Vision Library (OpenCV) ist eine Programmbibliothek, die eine Vielzahl an Algorithmen rund um das Thema Maschinelles Sehen / Computer Vision bereitstellt[3]. Anders als vlcj ist OpenCV nicht auf den Umgang mit Videos spezialisiert und bietet daher keine vergleichbar umfangreiche Kompatibilität mit diversen Formaten. Von Vorteil ist jedoch, dass OpenCV präzisen Zugriff auf individuelle Frames nach Nummerierung gewährt und umgekehrt auch präzise Angaben zur Position eines Frames im Video macht. Vergleicht man die Implementierung eines Video-Spielers auf OpenCV-Basis mit vlcj-basierten Lösungen, so ähnelt diese dem DirectMediaPlayer-Ansatz. OpenCV arbeitet jedoch nicht über ein Callback, sondern über ein Pull- bzw. Fetch-System, bei dem der nächste Frame als Puffer angefragt wird. Dieser Puffer wird dann wie bei dem DirectMediaPlayer manuell auf eine Oberfläche gezeichnet. Über einen Timer mit einer Tickrate in Abhängigkeit von der Bildfrequenz der Videodatei, die mittels OpenCV leicht ermittelt werden kann, lässt

[1] https://github.com/caprica/vlcj. (Zul. geprüft: 11.08.2019)
[2] https://wiki.videolan.org/VLC_Features_Formats/. (Zul. geprüft: 11.08.2019)
[3] https://docs.opencv.org/3.4.7/d1/dfb/intro.html. (Zul. geprüft: 12.08.2019)

sich eine Schleife realisieren, die in gleichmäßigen Zeitabständen neue Frame-Puffer abfragt und eine Neuzeichnung der Video-Panel-Oberfläche auslöst. So lässt sich ein einfacher Video-Player auf OpenCV-Basis umsetzen. Ein sehr großer Vorteil von OpenCV sind die bereits erwähnten Algorithmen, von denen einige auch an anderer Stelle in dieser Arbeit Verwendung finden. OpenCV ist unter einer BSD-Lizenz mit drei Klauseln lizenziert[4].

### 5.2.3 Sonstige Programmbibliotheken

Neben OpenCV nutzt dieses Programm noch folgende anderen Libraries:
**JIDE-OSS** bzw. JIDE Common Layer[5] für das FormLayout in Java, lizenziert unter GPLv2 mit Classpath-Ausnahme[6].
**JGoodies** Common und Forms[7] für ein Swing-Interface-Element, den Slider mit zwei Griffen. Dieses Paket ist lizenziert unter einer BSD-Lizenz[8].
**FFmpeg** als Teil von OpenCV unter LGPLv2.1[9].
Die **LAB**-Klasse aus **C3** (Categorical Color Components) der Stanford University[10] unter eigener Lizenz[11], insbesondere für den enthaltenen Ciede2000-Algorithmus, dessen Gewichtungsfunktionen dahingehend nachträglich verändert wurden, dass Unterschiede in der Helligkeit wesentlich schwächer gewichtet werden als Unterschiede in den Farbtönen.

## 5.3 Laden einer Bildschirmaufzeichnung

Über den Button mit der Aufschrift „Neues Projekt“ im Einstiegsfenster des Programms (Abbildung 5.5) kann ein Screen Recording geladen und die Hauptansicht geöffnet werden. Klickt man den Button, öffnet sich zunächst ein Dateiauswahldialog, der die Auswahl einer Datei eines gängigen Videoformat-Typs ermöglicht. Wählt der Nutzer eine gültige Datei aus, so wird das Video geladen, das Einstiegsfenster verschwindet und die Hauptansicht (Abbildung 5.1) öffnet sich. Das nun geladene Screen Recording wird im Nachfolgenden ggfs. auch als **Host-Screen-Recording** oder **Host-Bildschirmaufzeichnung** bezeichnet.
Der Button „Projekt laden“ öffnet ebenfalls einen Dateiauswahldialog, in dem eine Datei mit der Endung „.etproj“ ausgewählt werden kann. Diese enthält Informationen zu einem zuvor gespeicherten Projekt, darunter alle Einstellungen, die auf diesem Wege wiederhergestellt werden können. Bei gültiger Auswahl öffnet sich ebenfalls die Hauptansicht und alle vorherigen Datensätze und Einstellungen werden automatisch geladen.

[4] https://opencv.org/license/ (Zul. geprüft: 13.09.2019)
[5] https://github.com/jidesoft/jide-oss (Zul. geprüft: 13.09.2019)
[6] http://openjdk.java.net/legal/gplv2+ce.html (Zul. geprüft: 13.09.2019)
[7] http://www.jgoodies.com/downloads/libraries/ (Zul. geprüft: 13.09.2019)
[8] http://www.opensource.org/licenses/bsd-license.html (Zul. geprüft: 13.09.2019)
[9] http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html (Zul. geprüft: 13.09.2019)
[10] https://github.com/StanfordHCI/c3/blob/master/java/src/edu/stanford/vis/color/LAB.java (Zul. geprüft: 13.09.2019)
[11] https://github.com/StanfordHCI/c3/blob/master/LICENSE (Zul. geprüft: 13.09.2019)

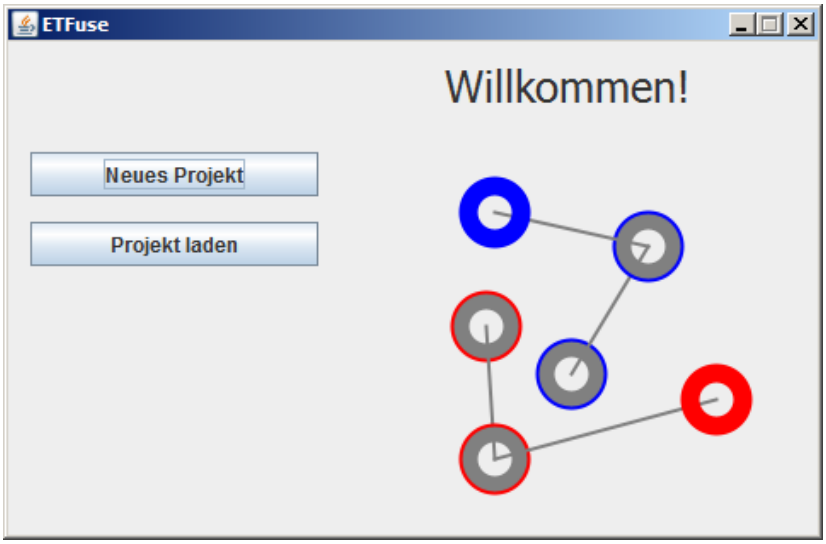



**Abbildung 5.5:** Einstiegspunkt des Programms. Button mit der Aufschrift „Neues Projekt“ öffnet einen Dateiauswahldialog, über den ein Video (Bildschirmaufzeichnung) ausgewählt werden kann. „Projekt laden“ ermöglicht das Wiederherstellen eines zuvor gespeicherten Projekts inklusive aller gespeicherten Einstellungen.

#### Hinweise

Für die Videodateien sind einige Kriterien zu berücksichtigen. Erstens müssen die Screen Recordings dahingehend zugeschnitten sein, dass sämtliche Vorkalibrierungsschritte nicht Teil des Videos sind. Im Verlauf des Daten-Parsings (Abschnitt 5.5) werden gelesene Elemente vor offiziellen „Screen Recording Start“-Markern nicht beachtet und Zeitstempel von Events so angepasst, dass „Zeitpunkt 0“ mit besagtem Marker koinzidiert. Alle enthaltenen Gaze-Events aus der Kalibrierungsphase und vor dem offiziellen Beginn des Screen Recordings würden also einen negativen Zeitstempel erhalten. Damit die Gaze-Events mit den Szenarien im Spiel also zeitlich korrekt korrelieren, ist es erforderlich, dass das zu verwendende Screen Recording entsprechend den Event-Markern zugeschnitten ist. Zweitens ist zu beachten, dass OpenCV gewisse Formate und Codecs nicht unterstützt. Im Zweifelsfall lassen sich Screen Recordings mit einem Konverter wie etwa *ffmpeg*[12] in ein unterstütztes Format konvertieren. Diese Arbeit empfiehlt hierfür das mp4-Format mit H.264-Kompression auf schwächerer Hardware bzw. mit H.265-Kompression auf stärkerer Hardware. Mittels ffmpeg lassen sich Dateien wie folgt konvertieren:

```
ffmpeg -i input.avi -c:v libx264 -an output.mp4
```

**Anmerkungen:** input.avi ist der Platzhalter für den Namen der Eingabedatei, output.mp4 ist der Platzhalter für den Namen der Ausgabedatei.
Hierbei wird eine eventuell enthaltene Audiospur bewusst nicht mitkonvertiert, da eine Tonausgabe nicht in den Aufgabenbereich dieses Programms fällt und in den Anwendungsszenarien nicht weiter

[12] https://ffmpeg.org/. (Zul. geprüft: 11.08.2019)

von Belang ist. Möchte man dennoch eine Audiospur aus der Quelldatei übernehmen, lässt sich dies durch das Entfernen des Parameters » -an « erzielen. Soll mittels H.265 statt H.264 komprimiert werden, um im Ergebnis eine kleinere Ausgabedatei zu erhalten, so gelingt dies durch das Ersetzen des Parameters »libx264« durch »libx265«. Hierbei steigt jedoch der Rechenaufwand und damit die Dauer des Vorgangs merklich.

## 5.4 Einbinden eines Eye-Tracker-Datensatzes

Um zu einem geladenen Host-Screen-Recording (Abschnitt 5.3) den zugehörigen Eye-Tracker-Datensatz zu laden, navigiert man zunächst über die Menüleiste (Abbildung 5.1 unten, C) zu „Datei“ und wählt „Host-Trackerdaten laden...“. Infolge dessen öffnet sich ein Dateiauswahldialog, der die Auswahl von TSV-Dateien (mehr hierzu in Abschnitt 5.5) ermöglicht. Wählt man nun einen gültigen Datensatz aus, schließt sich der Dateiauswahldialog und der Parsing-Prozess findet statt. Nach Abschluss des Daten-Parsing werden die geladenen Daten mit dem Screen Recording verknüpft und können auf dem Video-Panel (Abbildung 5.1 unten, A) visualisiert werden.

## 5.5 Daten-Parsing

In diesem Abschnitt wird der Parsing-Prozess erläutert und einige implementierungsspezifische Details beleuchtet.

### 5.5.1 TSV

Wie bereits erwähnt, befinden sich die Eye-Tracker-Datensätze im Tab-Separated Values (TSV)-Format. Bei diesem Format sind Tabellenspalten durch Tabulator voneinander getrennt. Tabellenzeilen sind durch Zeilenumbruch getrennt. Im Gegensatz zu anderen populären Tabellen(mappen)-Formaten wie XLS (Microsoft Excel File Format) oder ODS (OpenDocument Format Spreadsheet) sind bei TSV Tabellenstrukturen auch in einfachen Texteditoren einsehbar. Der Parsing-Vorgang ist dementsprechend einfach.

### 5.5.2 Parser-Module

Die zwei im Rahmen dieser Arbeit verwendeten Geräte exportieren beide unterschiedlich strukturierte Datensätze in TSV. Dementsprechend bedürfen die Datensätze für beide Modelle beim Import unterschiedlicher Behandlung. Hierfür definiert das Programm ein Parser-Interface, das von allen Parser-Modulen implementiert wird. Die Parser-Module werden für jedes zu unterstützende Gerät getrennt implementiert und können daher individuell Eigenheiten in den Exportstrukturen jedes Gerätes behandeln. Im Fall von **Gerät 1** (Unterabschnitt 4.1.1) beispielsweise können die Zeitstempel zu den Events sowohl in Millisekunden, als auch in Mikrosekunden exportiert werden. Über derartige Eigenschaften gibt der restliche Inhalt des erzeugten Datensatzes keinerlei Auskunft und sie müssen daher beim Import jeweils manuell ermittelt werden. Ebenfalls zu beachten gilt, dass jedes Parser-Modul indirekt verantwortlich ist für die Synchronisierung des Datensatzes mit

dem zugehörigen Screen Recording. Dieser Umstand wurde bereits in der Untersektion *Hinweise* in Abschnitt 5.3 angedeutet. Das Parser-Modul hat die Aufgabe, den Event-Marker im Datensatz zu finden, der kennzeichnet, ab wann das Screen Recording offiziell beginnt. Die zwei im Rahmen dieser Arbeit verwendeten Geräte exportieren die Gaze-Daten aus allen Phasen der Aufzeichnung, darunter auch solche, die für die behandelte Problematik uninteressant sind. Primär werden also die vorangehenden Gaze-Daten aus der Kalibrierungsphase übersprungen und alle Zeitstempel der gelesenen Events entsprechend angepasst. Läuft dieser Vorgang korrekt ab, fällt der Zeitstempel „0“ auf die Event-Markierung „ScreenRecordingStart“ (Name variiert je nach Gerät und zugehöriger Software), und alle nachfolgenden Events erhalten einen Zeitstempel gemäß Zeitabstand von „ScreenRecordingStart“. Infolge dessen ist der importierte Datensatz mit dem Screen Recording zeitlich synchron, insofern Letzteres gemäß den Hinweisen in Abschnitt 5.3 entsprechend korrekt zugeschnitten ist. Als Endergebnis des Parsing-Prozesses streben die Parser-Module das Erzeugen einer Instanz einer universellen Datenstruktur an, einem *Eye-Tracker-Recording* mit diversen *Eye-Tracker-Eye-Event*-Instanzen. Das macht auch das Hinzufügen von neuen Parsern und somit das Erweitern der Kompatibilität einfach.

### 5.5.3 Eye-Tracker-Recording & Eye-Tracker-Eye-Event

Die Strukturklassen *Eye-Tracker-Recording* und *Eye-Tracker-Eye-Event* orientieren sich an in Kapitel 2 etablierten Prinzipien. Ein Eye-Tracker-Recording verfügt über eine Liste aller dazugehörigen Gaze-Event-Punkte (GEPs), verkörpert durch die Eye-Tracker-Eye-Event-Instanzen. Jede Instanz von Eye-Tracker-Eye-Event enthält zwei Koordinaten $(x, y)$ des Typs Integer der Position des Blickes des/r Probanden/in auf dem Medium, und den zugehörigen Zeitpunkt $Z$, den Timestamp, als Long Integer. Außerdem ist jedes Event mit einem Index markiert und enthält Informationen zur Position der Augen des/r Probanden/in im dreidimensionalen Raum falls applikabel. Diese sind ggfs. für einen späteren Filtervorgang von Belang (siehe Abschnitt 2.2).

### 5.5.4 Event-Suche

Der Umfang eines Eye-Tracker-Recordings ist je nach Dauer der Aufzeichnung und Gaze-Recording-Frequenz (GRF) des Geräts relativ hoch. Bei **Gerät 1** umfasst ein Datensatz zu einem vier Minuten und 20 Sekunden langen Recording bei einer Frequenz von 1200 Hz über 300.000 Events. Für einige Vorgänge, wie beispielsweise dem Springen im Video über die Zeitleiste oder dem hochfrequentierten Visualisieren der Gaze-Punkte mittels diverser Modi (Abschnitt 5.7), ist ein performanter Event-Suchalgorithmus unabdingbar. Dieser sollte mit möglichst geringem Rechenaufwand den Index der letzten Eye-Tracker-Eye-Event-Instanz mit Zeitstempel vor einem gegebenen Zeitpunkt $Z$ in einem Eye-Tracker-Recording finden. Im erweiterten Modus sollte es außerdem möglich sein, die Menge aller Eye-Tracker-Eye-Event-Instanzen mit Zeitstempeln zwischen zwei gegebenen Zeitpunkten $Z_1$ und $Z_2$ zu finden. Der implementierte Algorithmus akzeptiert eine Anzahl und einen Zeitstempel als Eingabe und schätzt einen Start-Index für die Suche basierend darauf, wie weit der Zeitstempel vom Zeitstempel des ersten und des letzten Events entfernt ist. Dann wird der Such-Index so lange inkrementiert oder dekrementiert, bis er auf das erste Event mit einem Zeitstempel deutet, der <= der eingegebene Parameter ist. Zum Schluss wird der Index entsprechend der angegebenen Anzahl dekrementiert und jedes zugehörige Event zur Ausgabe hinzugefügt. Für den erweiterten Modus akzeptiert der Algorithmus zwei Zeitstempel, einen Start und ein Ende. Der Ablauf in Schritt eins

ist zunächst identisch und die Event-Zeitstempel werden mit dem größeren Eingabe-Parameter, also dem Ende-Zeitstempel, abgeglichen. Ist das letzte Event des Abschnittes gefunden, läuft die Suche im zweiten Schritt solange rückwärts, wie der Index > 0 und der Zeitstempel des Events mit besagtem Index >= der angegebene Start-Zeitstempel ist. Jedes betrachtete Event in Schritt zwei wird zur Ausgabe hinzugefügt.

Für Pseudocode-Algorithmen, siehe Listing 1 und Listing 2 im Anhang.

## 5.6 Fixationsfilter

Das Programm implementiert einen I-VT-Fixationsfilter gemäß Salvucci *et. al* [SG00] mit einer Geschwindigkeitsschranke, dem *velocity threshold*, und einer zusätzlichen Distanzschranke, dem *distance threshold*. Die Implementierung hält sich im Wesentlichen an den Referenzablauf (siehe Unterabschnitt 2.2.1), muss allerdings zu einem der beiden zu unterstützenden Tracker-Geräte eine Annahme machen. Die aufgezeichneten Datensätze des Tobii T60XL (siehe Unterabschnitt 4.1.1) enthalten nur sehr ungenaue Informationen zur Position des/der Probanden/in relativ zum Monitor. Für jeden Gaze-Punkt ist bei diesem Gerät jeweils nur der Abstand zwischen Proband/in und Monitor angegeben, während der Tobii Pro Spectrum 1200 die X/Y/Z-Koordinaten der präzisen relativen Position im Raum angibt. Da zur Berechnung der Punkt-zu-Punkt-Geschwindigkeiten der Blickwinkel zwischen den zwei Punkten berechnet werden sollte, um ein möglichst genaues Ergebnis zu erhalten, nimmt das Programm an, dass der/die Proband/in im Falle des Tobii T60XL zentral vor dem Bildschirm positioniert war und der angegebene Abstand den Minimalabstand zwischen Proband/in und der Mitte des Bildschirms beziffert. Ferner werden für beide Geräte die Abstandsinformationen bzw. Koordinaten für einzelne Augen zusammengefasst und umgerechnet auf ein einzelnes „Durchschnittsauge“, das räumlich in der Mitte zwischen den jeweiligen Werten liegt. Für die Punkt-zu-Punkt-Geschwindigkeiten wird außerdem nicht nur der Abstand zwischen einem Punkt und dessen Vorgänger *oder* dessen Nachfolger betrachtet, wie Salvucci *et. al* [SG00] beschreiben, sondern der Durchschnitt der Geschwindigkeiten zwischen dem Punkt und dem Vorgänger sowie dem Punkt und dem Nachfolger genutzt. Die Distanzschranke aggregiert darüber hinaus in einem abschließenden Schritt rekursiv solche Fixationen, die aufeinander folgen und deren Zentroide (Schwerpunkte bzw. Mittelpunkte) einen Abstand unterhalb der angegebenen Schranke haben, bis keine aufeinanderfolgenden Fixationen mehr unter die Schranke fallen.

## 5.7 Visualisierungsmodi

Das entwickelte Programm bietet drei unterschiedliche Visualisierungsmodi zur Darstellung importierter Eye-Tracker-Daten. Besagte Modi werden in diesem Abschnitt erörtert.

### 5.7.1 Gaze Plots

Gaze Plots (gemäß Abschnitt 2.3) stellen die importierten Daten in Form eines gerenderten Scanpaths dar. Die Implementierung nutzt die Event-Suche-Algorithmen (siehe Unterabschnitt 5.5.4), um eine Menge an zeitlich unmittelbar vorhergegangenen Eye-Tracker-Events zum gegebenen Zeitstempel im dargestellten Screen Recording zu finden. Das Neuzeichnen des Gaze Plots findet frameweise

und damit analog zur Bildrate des Videos statt. Da die Zeitstempel der Eye-Tracker-Events in Millisekunden geparst werden, geschieht die Umrechnung des Frame-Counters des Videos zum zugehörigen Event-Zeitstempel nach folgender Formel:

$$TimestampMS = \frac{FrameCounter}{fps} * 1000$$

(Mehr Informationen zur Synchronisation zwischen Eye-Tracker-Datensätzen und zugehörigen Screen Recordings finden sich in Unterabschnitt 5.5.2.)
Die Koordinaten der gefundenen Events werden über die Inhalte des jeweiligen Frames gezeichnet (siehe Abbildung 5.6). Für jedes Event wird ein Ring gezeichnet, und die Ringe aufeinanderfolgender Events werden über gerade Linien miteinander verbunden. Bei gefilterten Eye-Tracker-Recordings (siehe Abschnitt 2.2) entsprechen die Kreise den Fixationen und die Linien den Sakkaden. Zusätzlich wird für jedes Recording der jüngste Gaze-Punkt aus den Rohdaten in Form eines farblich etwas helleren Kreises gezeichnet, sofern die Option in den Einstellungen aktiviert ist (siehe Abbildung 5.3). Dadurch wird mitunter die Wirkung des Fixationsfilters (Abschnitt 5.6) sehr anschaulich präsentiert. Sind zusätzliche Gast-Eye-Tracker-Recordings geladen, so werden die Gaze Plots der Recordings in unterschiedlichen Farben dargestellt.

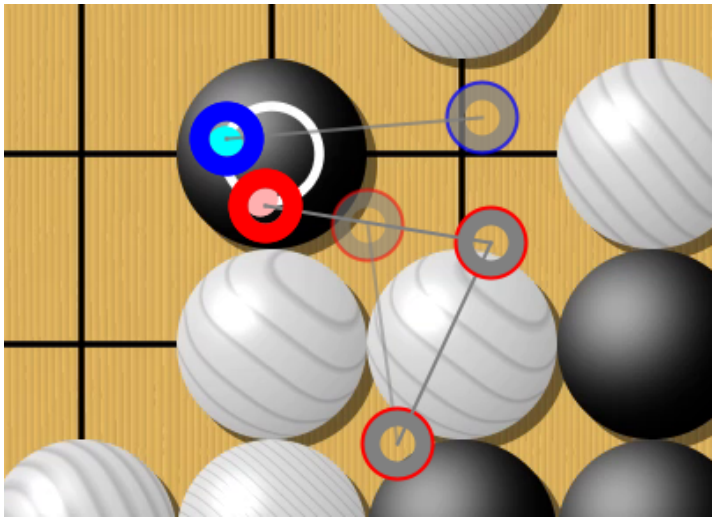


**Abbildung 5.6:** Gaze Plot zweier Spieler in Blau und Rot. Fixationen dargestellt als Ringe. Aktive Fixationen sind farblich komplett ausgefüllt, vergangene Fixationen haben lediglich einen farbigen Rand und sind ansonsten grau. Teiltransparente Fixationen liegen zeitlich etwas länger zurück. Kreise im Inneren der aktiven Fixationen zeigen jeweils den jüngsten Gaze-Punkt aus den Rohdaten zum gegebenen Zeitpunkt.

### 5.7.2 Attention Maps / Heatmaps

Mit den Attention Maps bietet das entwickelte Programm einen zweiten Visualisierungsmodus für die Gaze-Daten. Werden zu einem Screen Recording die Eye-Tracking-Daten geladen, beginnt das Programm automatisch damit, asynchron eine Heatmap über den Datensatz zu berechnen. Der hierfür entwickelte Algorithmus wählt einen Ansatz, der quasi umgekehrt zur Theorie (siehe Unterabschnitt 2.3.2) arbeitet, jedoch vergleichbare Ergebnisse liefert. Die erzeugten Attention Maps

werden an zwei Stellen dargestellt, als Overlay für das Video (siehe Abbildung 5.7) ähnlich dem Gaze Plot (Unterabschnitt 5.7.1) und im dafür vorgesehenen Vergleichsfenster (siehe Unterabschnitt 5.1.2), das zwei Attention Maps einander gegenüberstellt. Außerdem können die Attention Maps auf zwei Weisen aggreggiert werden: nach Dauer (siehe Abbildung 5.8) und nach Häufung (siehe Abbildung 5.9). Beide Maps wurden über den gleichen Zeitraum und den gleichen Datensatz erstellt. Unterschiede werden hier nur bei gefilterten Datensätzen sichtbar, da ungefilterte Datensätze für jeden Gaze-Punkt die gleiche Dauer in Abhängigkeit von der Gaze-Recording-Frequenz (GRF) zugewiesen bekommen. Da die Datensätze unter Umständen recht umfangreich sind und pro Gaze-Punkt einige tausend Rechenoperationen durchgeführt werden müssen, bietet die Implementierung außerdem einen performancesteigernden Trick an, der optional zugeschaltet werden kann und auf den im Nachfolgenden eingegangen wird. Zur Steigerung der Performance lässt sich zunächst allerdings der Hitze-Radius anpassen, der bestimmt, wie weit jeder Punkt seine Hitze auf umliegende Pixel „strahlt“. Verringert man diesen Radius, müssen pro Gaze-Punkt weniger Rechenoperationen durchgeführt werden und die Dauer der Berechnung sinkt. Die Implementierung wählt hier einen Radius von 80 Pixeln als Referenzwert. Einen größeren Performancegewinn erzielt der Algorithmus jedoch durch das Überspringen von Events. Dabei können Events nach einem angegebenen Prozentsatz übersprungen werden. Der Referenzwert der Implementierung liegt bei 0,05%, was zur Folge hat, dass nach jedem Event, das vom Algorithmus beachtet wird, die nächsten 0,05% aller Events übersprungen werden. Dadurch wird die Anzahl der zur Generierung der Heatmap berücksichtigten Events fix und unabhängig von der Größe des Datensatzes. Relative Häufigkeiten von Vorkommen an bestimmten Punkten bleiben so mehr oder weniger gut erhalten und die erzeugten Heatmaps bei Aggregation nach Häufung unterscheiden sich nicht maßgeblich von solchen, die über den gesamten Datensatz erzeugt wurden. Eine etwas unvorhersehbarere Auswirkung hat diese Performanceoptimierung allerdings bei Aggregation nach Aufmerksamkeitsdauer, dem Standardmodus des Generators. Für den Standard-Anwendungsfall einer Heatmap wirkt diese Methode deshalb kontraproduktiv, weil sie künstlich und ungleichmäßig die Aufmerksamkeitsspannen für bestimmte Koordinaten unvorhersehbar reduziert. Unter Umständen könnten Heatmaps dadurch merkbar verfälscht werden. Grundsätzlich ist das Überspringen von Events zur Optimierung der Performance jedoch nicht notwendig, da die Heatmaps über die gefilterten Fixationen generiert werden, und erst übersprungen wird, wenn die Anzahl an Datenpunkten einen gewissen Wert überschreitet. Dieser Wert berechnet sich wie folgt:

$$Schranke = \frac{100}{SkipProzentsatz} * 1,5$$

Im Falle des Referenzwertes von 0,05% liegt die Schranke somit bei 3.000. Bei einem Recording mit einer Länge von drei bis vier Minuten werden jedoch üblicherweise nur etwa 600-700 Fixationen aggregiert, daher wird die Option erst bei verhältnismäßig langen Recordings wirksam und ist standardmäßig deaktviert.
Die Normalisierung für unabhängige Attention Maps, die nicht für den Direktvergleich zwischen zwei Spielern gedacht sind, geschieht nach einem einfachen linearen Stauchungsprinzip, bei dem Werte von $0 \rightarrow X$ (wobei $X$ = höchster aggregierter Wert in der Attention Map) gestaucht werden auf eine Skala von $0 \rightarrow 255$. Für zwei zu vergleichende Attention Maps funktioniert die Normalisierung nach dem gleichen Prinzip, jedoch wird in einem vorangehenden Schritt der Faktor (fortan *Größenfaktor*) ermittelt, um den sich die Maxima der zwei Attention Maps unterscheiden, um bei der Stauchung relative Unterschiede beizubehalten. Für den unwahrscheinlichen Fall, dass bei einer Attention Map tatsächlich jeder Punkt im Bild bestrahlt wird und daher einen aggregierten

Hitzewert > 0 hat, wird zusätzlich die Differenz zwischen dem eigenen Minimum und dem der anderen Attention Map mit dem eigenen Größenfaktor multipliziert und der resultierende Wert als untere Schranke der Normalisierungsskala verwendet. Dadurch bleiben Unterschiede in den globalen Minima der Attention Maps für den Direktvergleich erhalten. Abschließend wird noch ein Farb-Mapping für diese normalisierte Skala je nach Modus der Darstellung durchgeführt. Attention Maps für das Vergleichsfenster durchlaufen ein Mapping, bei dem niedrige Werte auf ein Schwarz oder Dunkelrot mappen und hohe Werte in einem Gelb bzw. Weiß resultieren. Für das Overlay über dem Video wird stattdessen ein Mapping auf einen Regenbogen-Farbverlauf durchgeführt, bei dem niedrige Werte auf einen tiefen Blauton gemappt werden und hohe Werte in einem Dunkelrot bzw. Braun resultieren. Beim Zeichnen des Overlays, im Compositing-Schritt, verhält sich der Alpha-Wert jedes Pixels der Attention Map invers zum Blaugehalt, also zu den „kalten“ Regionen. Dadurch werden praktisch nur die Hotspots sichtbar und Bereiche, die weniger Aufmerksamkeit erfahren haben, bleiben unsichtbar.

Für einen Pseudocode-Algorithmus, siehe Listing 3 im Anhang.

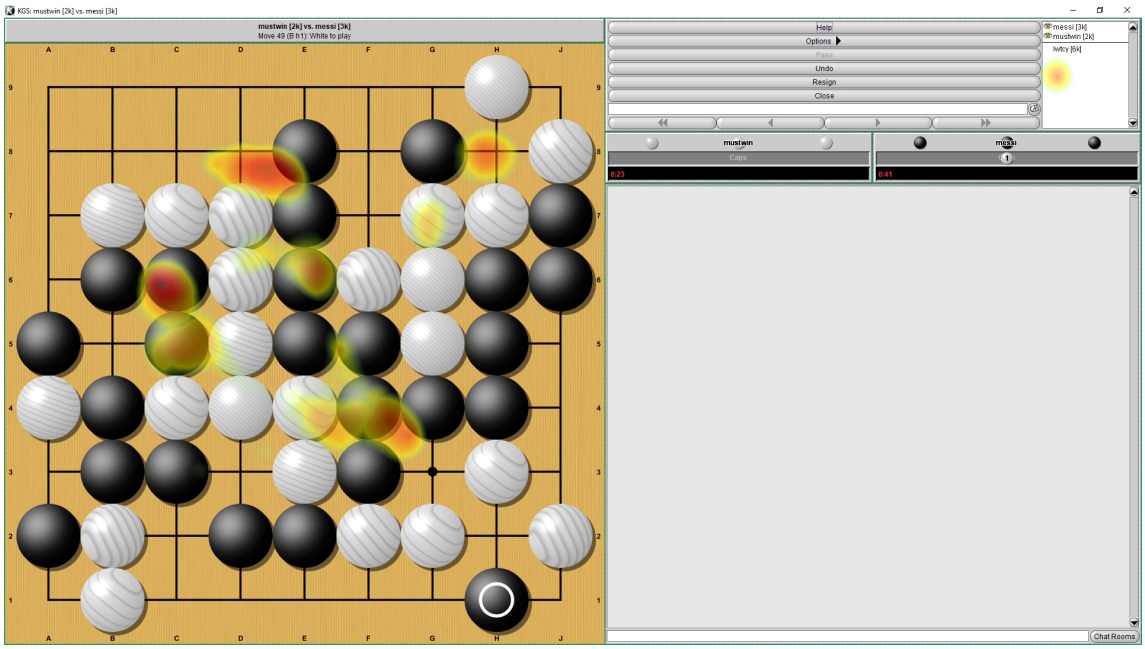



**Abbildung 5.7:** Attention Map (Heatmap) einer Partie, dargestellt als Overlay über dem Video. Nach einem Farb-Mapping auf eine Regenbogen-Farbskala verhält sich der Alpha-Wert jedes Pixels invers zum Blaugehalt. Dadurch werden nur die Hotspots mit der höchsten erfahrenen Aufmerksamkeit sichtbar. Es handelt sich um die gleiche Attention Map wie in Abbildung 5.8.

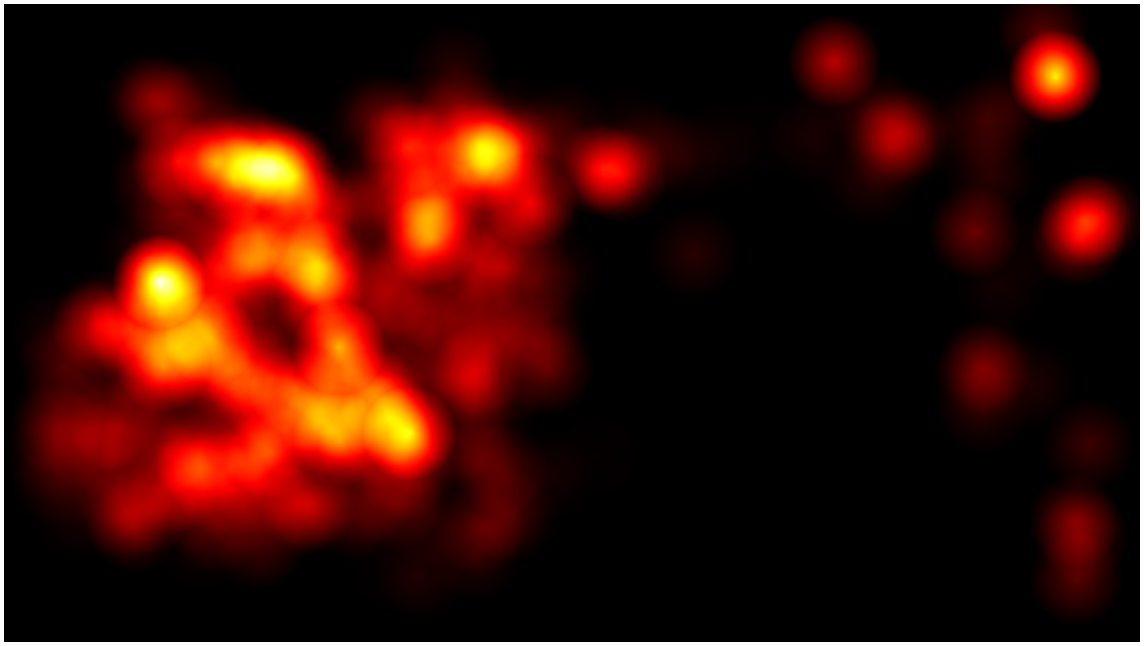


**Abbildung 5.8:** Attention Map (Heatmap) einer Partie, aggregiert nach der Dauer der Fixationen.

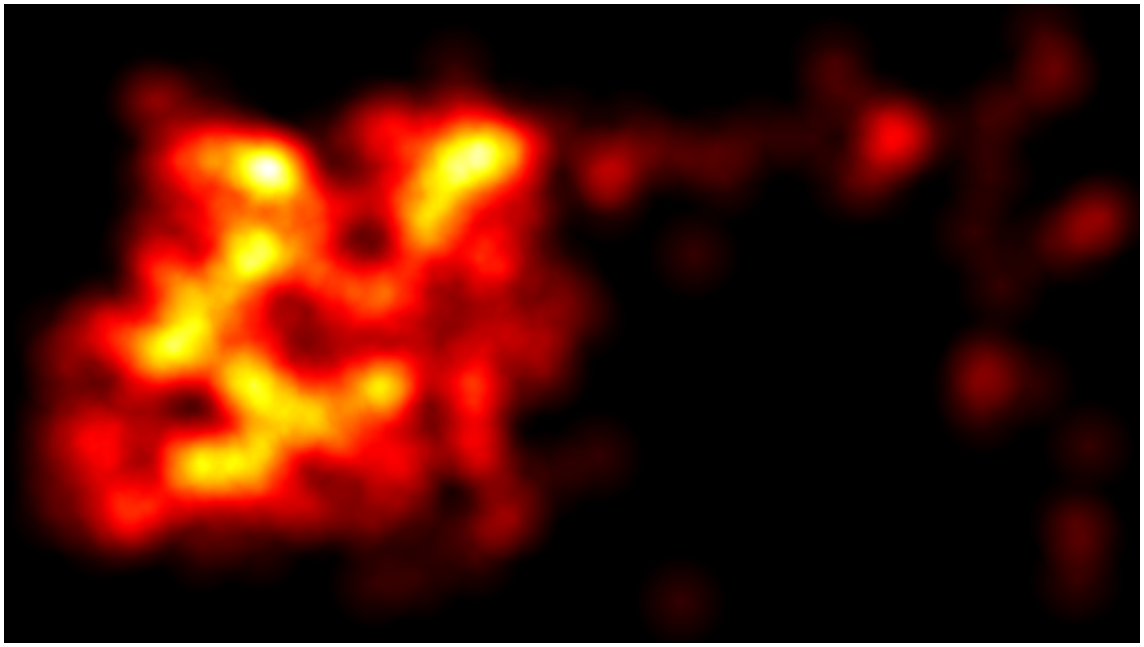


**Abbildung 5.9:** Attention Map (Heatmap) einer Partie, aggregiert nach Häufung der Fixationen.

### 5.7.3 Mindestdistanz-Scarf-Plots

Bei der dritten Visualisierungstechnik handelt es sich um einen Plot des Abstandes der Gaze-Punkte aus den Rohdaten des Host- und eines geladenen Gast-Recordings zu jedem Zeitpunkt über eine Zeitachse. Für jeden Pixel bzw. vertikalen Strich auf der Zeitleiste bestimmt sich die Einfärbung darüber, ob der Abstand zwischen den Punkten mehrheitlich unter einer Schranke liegt (cyan) oder darüber (rot). Fehlen für den gegebenen Zeitpunkt Gaze-Informationen in mindestens einem der Recordings, weil etwa der/die Spieler/in beispielsweise gerade geblinzelt, die Augen geschlossen oder den Blick abgewandt hat, so wird dunkelgrau eingefärbt. Liegen die Gaze-Punkt mindestens eines Recordings mehrheitlich außerhalb des Spielfeldes und kein einziges mal unter der Schranke, so wird dies über eine gelbe Markierung gekennzeichnet. Da je nach Gaze-Recording-Frequenz (GRF) eine größere Anzahl an Events farblich auf einen einzigen Pixel gemappt wird, zählt die Implementierung Vorkommen unter allen Events zwischen denen des vorhergehenden Pixels (bzw. Spalte) und denen des nachfolgenden. Der gewählte Standardwert für die Mindestdistanz-Schranke beträgt 150 Pixel. Entsprechend der größten Gruppe (< Schranke; > Schranke; keine Informationen; außerhalb des Spielfeldes) wird eingefärbt. Der Mindestdistanz-Scarf-Plot setzt die Präsenz eines Gast-Recordings voraus und wird daher erst nach der Einbindung eines weiteren Datensatzes neben dem Host-Datensatz gezeichnet. Die Visualisierung erscheint am unteren Rand der Hauptansicht unterhalb der Zeitleiste (siehe Abbildung 5.10).

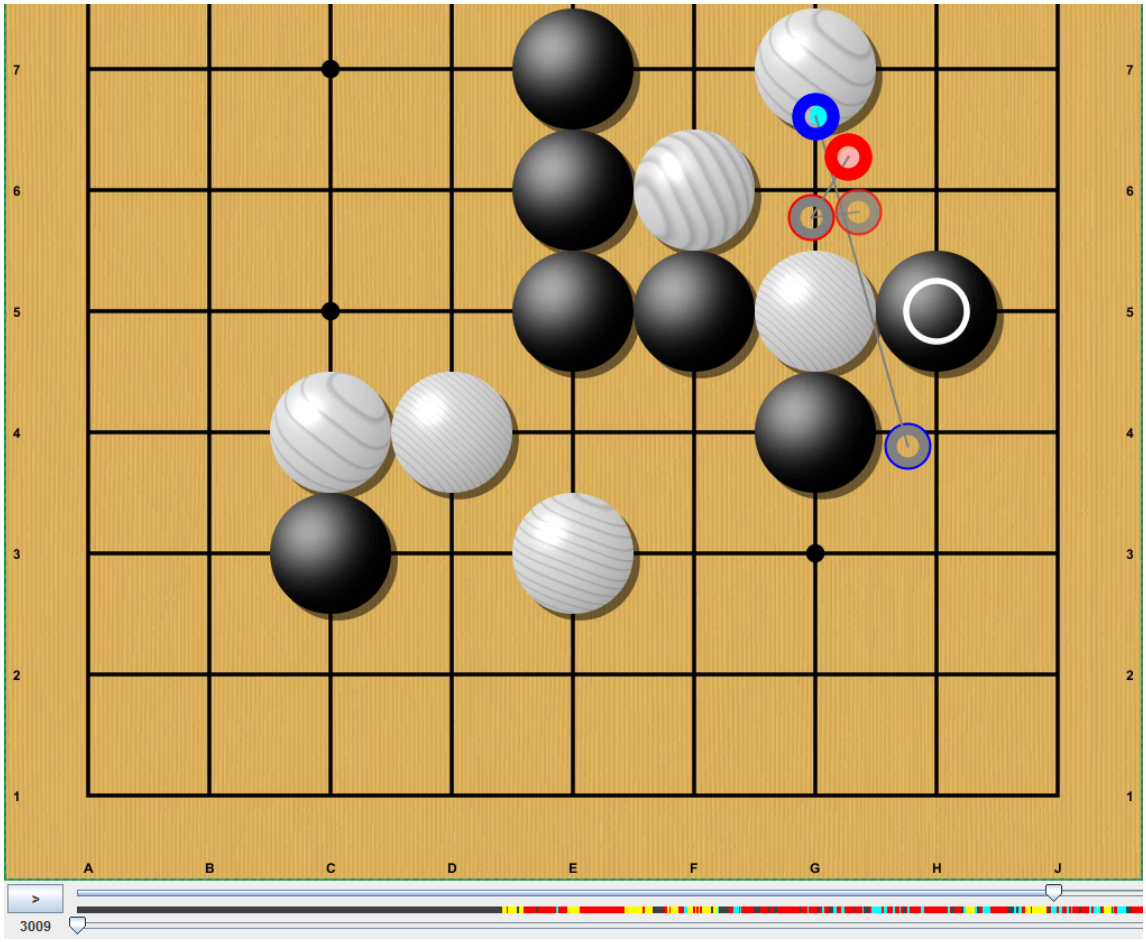



**Abbildung 5.10:** Ausschnitt des Mindestdistanz-Plots (am unteren Rand). Der graue Abschnitt zu Beginn ist Resultat des verzögerten Aufnahmestarts zwischen den zwei Eye-Tracker-Recordings. Cyan markiert Gaze-Abstand < Schranke, Rot markiert Gaze-Abstand > Schranke. Gelb markiert Zeitpunkte, zu denen Blick mindestens eines Spielers nicht innerhalb des Spielfeldes fällt.

## 5.8 Einbinden weiterer Datensätze zur parallelen Darstellung

Die behandelte Kernproblematik des entwickelten Programms ist die Kombination mehrerer Datensätze verschiedener Aufzeichnungen der gleichen Partie aus der Sicht verschiedener Spieler/innen mit dem Ziel, auf einem einzigen Screen Recording Visualisierungen der Gaze-Daten aller Spieler/innen zu sehen. Zum Einbinden und Anpassen weiterer Datensätze auf ein geladenes Host-Screen-Recording (Abschnitt 5.3) stellt das Programm einen Wizard bereit, der es dem Nutzer ermöglicht, in vier Schritten die nötigen Angaben zu machen und Maßnahmen zu tätigen und sogenannte *Gast-Recordings* zu laden (Abbildung 5.11). Um den Wizard aufrufen zu können, muss zunächst ein Host-Screen-Recording mitsamt zugehörigem Datensatz geladen sein (Abschnitt 5.3, Abschnitt 5.4). Über den Eintrag „Synch“ → „Gast-Recording darstellen“ in der Menüleiste lässt sich daraufhin der Wizard öffnen. Der Nutzer durchläuft im Wizard die folgenden vier Schritte:

**Schritt 1 – Laden des Gast-Datensatzes.** Im ersten Schritt (Abbildung 5.11 A) wird der Datensatz des Gast-Recordings geladen. Dieser befindet sich wie der Datensatz des Host-Recordings auch im TSV-Format (Abschnitt 5.5). Nach Laden der Daten geht der Wizard in den nächsten Schritt über.

**Schritt 2 – Festlegen des Spielfeld-Bereiches im Gast-Recording.** Im zweiten Schritt wird der Bereich des Videos im Gast-Recording ausgewählt, der von der Koordinatentransformation (Unterabschnitt 5.8.1) betroffen sein soll. Hierfür stehen dem Nutzer zwei Optionen zur Wahl: Presets oder eine manuelle Bestimmung. Unter den Presets gibt es zwei vorgefertigte Bereiche, die auf die Recordings passen, die im Rahmen dieser Arbeit mit den zwei Geräten getätigt wurden. Alternativ kann der Bereich jedoch auch manuell festgelegt werden. Hierfür ist die Bildschirmaufzeichnung des Gast-Recordings notwendig. Wird diese Option gewählt, öffnet sich ein Dateiauswahldialog, über den das Video geladen werden kann. Wählt der Nutzer eine gültige Datei aus, öffnet sich ein weiteres Fenster mit dem Video des Gast-Recordings, in dem mittels Mausklick die obere linke und die untere rechte Ecke des Spielfeldes markiert werden kann (Abbildung 5.12). Erfolgt dies, geht der Wizard in den nächsten Schritt über.

**Schritt 3 – Festlegen des Spielfeld-Bereiches im Host-Recording.** Analog zu Schritt 2 wird im dritten Schritt der Spielfeld-Bereich im Host-Recording bestimmt. Auch hierfür stehen dieselben Presets wie aus dem vorherigen Schritt bereit. Alternativ kann auch hier der Bereich manuell festgelegt werden (Abbildung 5.12). Ein Laden einer Video-Datei ist in diesem Schritt jedoch nicht nötig, da das Host-Recording bereits geladen ist und vom Wizard berücksichtigt wird. Ist dieser Schritt erfolgt, geht der Wizard in den letzten Schritt über.

**Schritt 4 – Wahl der Synchronisations-Methode.** Im letzten Schritt wählt der Nutzer zwischen drei Optionen zur zeitlichen Synchronisation der Recordings. Zur Auswahl stehen eine ungenaue, schnelle Methode, die sich nur an den System-Zeitstempeln in den Recordings orientiert, und zwei genauere Methoden, die in Unterabschnitt 5.8.2 im Detail erläutert werden. Wird die ungenaue Methode gewählt, ist die Konfiguration abgeschlossen und das Gast-Recording wird hinzugefügt. Wird hingegen eine der anderen beiden Methoden gewählt, sind zusätzliche Angaben notwendig, die dem jeweiligen Verfahren als Richtwerte dienen. Für die Synchronisation via KGS-Go-spezifischer Methode muss die Position des Steines auf dem Spielfeld bestimmt werden, an dem sich das Programm orientieren soll (Abbildung 5.13). Für die Synchronisation via Histogramm-Vergleich wird der Nutzer gebeten, ein Start-Frame aus dem Host-Screen-Recording auszuwählen, das zwischen zwei Zügen im Spiel liegt. Zum Tätigen dieser Angaben öffnet sich jeweils das Host-Screen-Recording in einem separaten Fenster. Sobald die Angaben getätigt wurden, ist der Wizard bereit, das Gast-Recording hinzuzufügen.

Nach Abschluss des Vorgangs steht das importierte Gast-Recording für sämtliche Visualisierungen zur Verfügung.

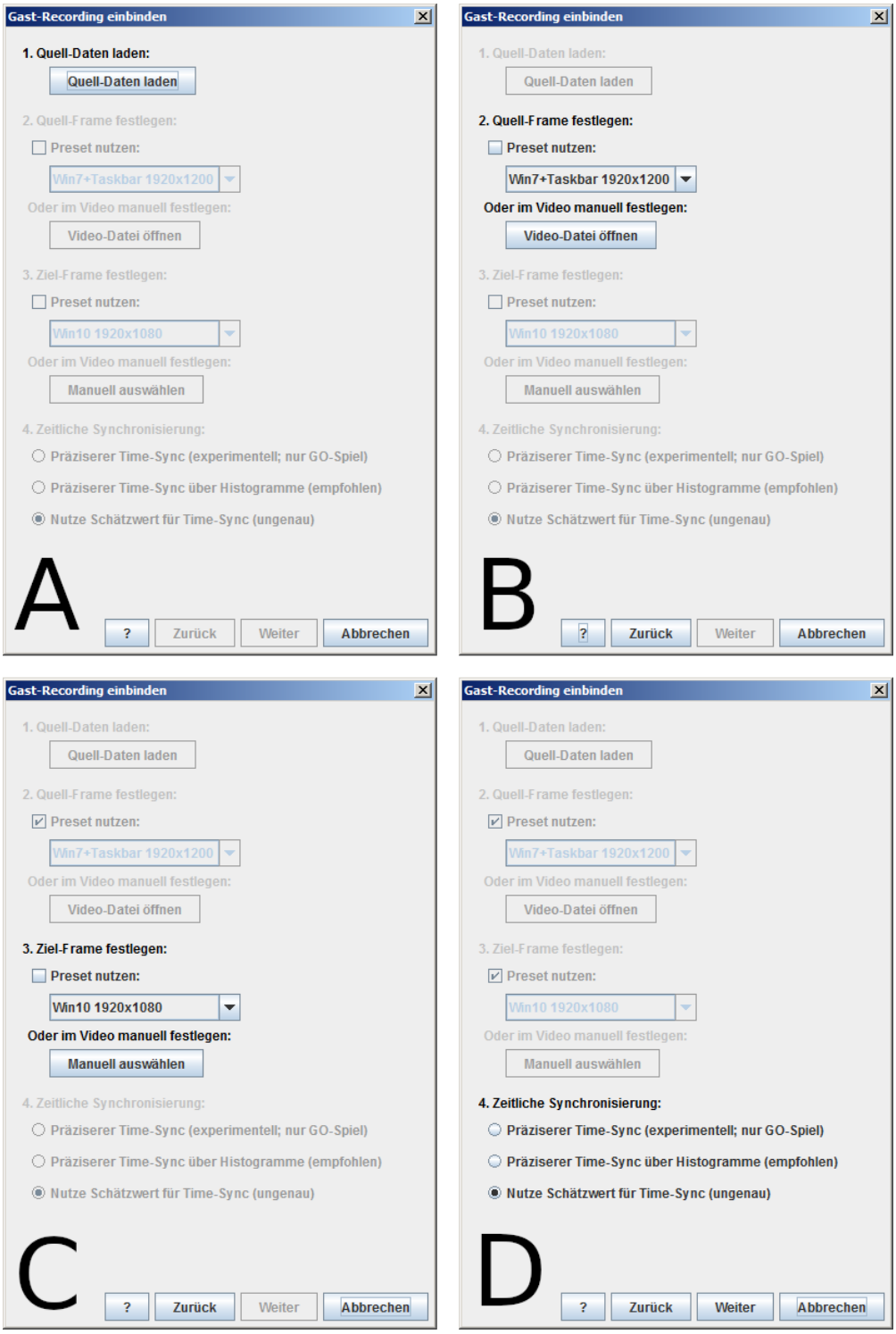



**Abbildung 5.11:** Wizard zum Hinzufügen weiterer Eye-Tracker-Recordings. Schritte 1 (A), 2 (B), 3 (C) und 4 (D). In Schritt 1 wird der Datensatz des Gast-Recordings geladen. In Schritt 2 wird der Spielfeld-Bereich im Gast-Recording bestimmt. In Schritt 3 wird der Spielfeld-Bereich im Host-Recording bestimmt. In Schritt 4 wird die Methode zur zeitlichen Synchronisation bestimmt.

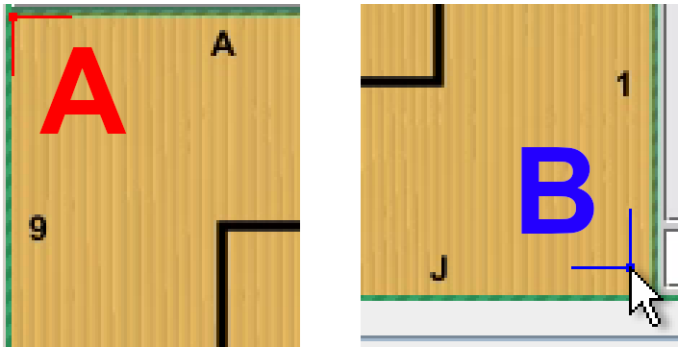



**Abbildung 5.12:** Tool zur manuellen Bestimmung des Spielfeld-Bereiches, der für die Koordinatentransformation beachtet werden soll (Area Picker). A: Obere linke Ecke des Spielfeldes, B: Untere rechte Ecke des Spielfeldes. Auswahl per Mausklick.

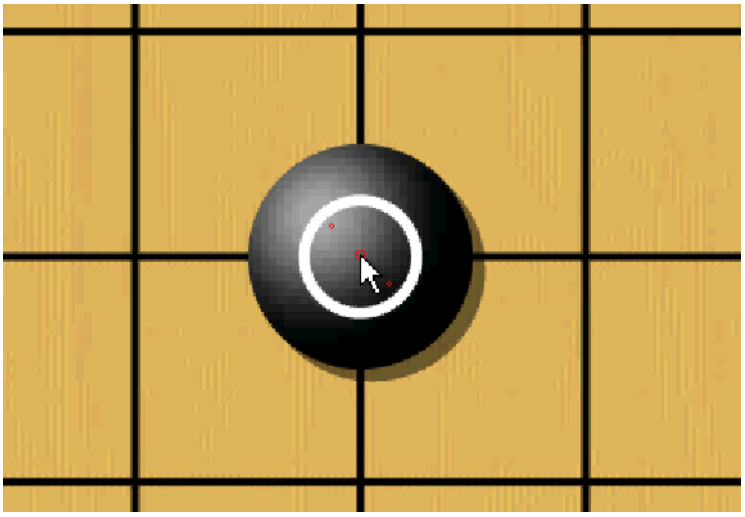


**Abbildung 5.13:** Tool zur Bestimmung des Steines, an dem sich die zeitliche Synchronisation orientieren soll (Coordinate Picker). Vorschau der zwei Punkte, an denen die Farbwerte überprüft werden, in rot (oberhalb links und unterhalb rechts der Mitte des Maus-Cursors). Auswahl per Mausklick.

### 5.8.1 Koordinatentransformation

Die Zusammenführung unterschiedlicher Datenpunkte zweier grundlegend verschiedener Datensätze auf einen gemeinsamen Nenner stellt einen der Kernaspekte dieser Arbeit dar. Eine Teilkomponente davon verkörpert das Koordinatentransformationsverfahren, das die entwickelte Software aus gegebenem Anlass durchführt, um zweidimensionale Koordinaten der Gaze-Event-Punkte (GEPs) auf die zugehörigen Koordinaten der Bildschirmaufzeichnung des jeweils anderen Spielers zu übertragen. Dies ist notwendig, da die Monitore der zwei Trackinggeräte, wie in Unterabschnitt 4.1.1 beschrieben, unterschiedliche Bildseitenverhältnisse aufweisen. Da die Go-Software im Vollbildmodus aufgezeichnet wurde, hat das zur Folge, dass die Software der zwei verwendeten Geräte sich dahingehend unterscheidet, dass das Spielfeld als Resultat des Interface-Layoutverhaltens der

Software in den Aufzeichnungen jeweils ungleiche Größen und Ursprungskoordinaten aufweist (siehe Abbildung 5.14). Daraus lässt sich schließen, dass auch die restlichen Elemente der grafischen Nutzeroberfläche an unterschiedlichen Stellen platziert sind und gegebenenfalls unterschiedlich groß sein können. Da sich kein einfaches lineares Mapping erstellen lässt, das die komplette Nutzeroberfläche beider Geräte abdeckt und aufeinander übertragbar macht, wählt diese Arbeit einen alternativen Ansatz, bei dem ausschließlich der für die Analyse interessante Teil der Bildschirmaufzeichnung betrachtet wird. Üblicherweise handelt es sich dabei um das Spielfeld. Vorteilhaft ist dies auch deshalb, weil das Transformationsverfahren dadurch allgemein anwendbar wird, und in puncto Kompatibilität nicht auf die verwendete Go-Software beschränkt ist.
Das Koordinatentransformationsverfahren dieser Arbeit transformiert also alle Koordinaten eines Datensatzes abhängig vom festgelegten Quellbereich bzw. Quell-Spielfeld so, dass sie, falls sie im Quellbereich liegen, auf den korrespondierenden Punkt des Spielfeldes im Zielbereich fallen. Für Gaze-Punkte, deren Koordinaten außerhalb des Spielfeldes liegen, geht der Kontext zumeist restlos verloren. Damit ist gemeint, dass sie bei Darstellung über dem Host-Screen-Recording nicht mehr zweifelsfrei Aufschluss darüber geben können, was genau der/die andere Spieler/in auf seinem/ihrem Bildschirm gesehen hat, da an der zugehörigen Stelle auf dem anderen Display andere Interfaceelemente platziert gewesen sein könnten. Die transformierten Koordinaten für einen gegebenen Gaze-Punkt berechnen sich folgendermaßen:

$$FaktorX = \frac{AlteKoordinateX - QuellBereichMitteX}{QuellBereichBreite * 0.5}$$

$$FaktorY = \frac{AlteKoordinateY - QuellBereichMitteY}{QuellBereichHoehe * 0.5}$$

$$NeueKoordinateX = ZielBereichMitteX + FaktorX * ZielBereichBreite * 0.5$$

$$NeueKoordinateY = ZielBereichMitteY + FaktorY * ZielBereichHoehe * 0.5$$

### 5.8.2 Zeitliche Synchronisation zweier Datensätze

Die zeitliche Synchronisation zweier Eye-Tracker-Recordings stellt eine nicht zu vernachlässigende Herausforderung dar. Dieser Schritt ist notwendig, damit für ein Spielszenario zu Zeitpunkt $Z$ im Host-Screen-Recording nicht nur die zeitlich dazugehörigen Gaze-Events des Spielers visualisiert werden, zu dem das Host-Screen-Recording gehört, sondern auch die dargestellten Gast-Daten zeitlich zu der im Video gezeigten Stelle passen. Der Ansatz, den diese Arbeit verfolgt, macht sich zunutze, dass der Datensatz eines Recordings bereits mit der zugehörigen Bildschirmaufzeichnung synchron ist. Basierend auf dieser Annahme lassen sich die Events zweier verschiedener Datensätze über deren Screen Recordings synchronisieren. Außerdem gibt es einige Anhaltspunkte, über die sich bei geschicktem Einsatz der benötigte Rechenaufwand zur Bestimmung des Zeitversatzes verringern lässt. Im Folgenden werden die Hilfsmittel und Methoden zum zeitlichen Synchronisieren zweier Datensätze erläutert, die das entwickelte Programm implementiert.

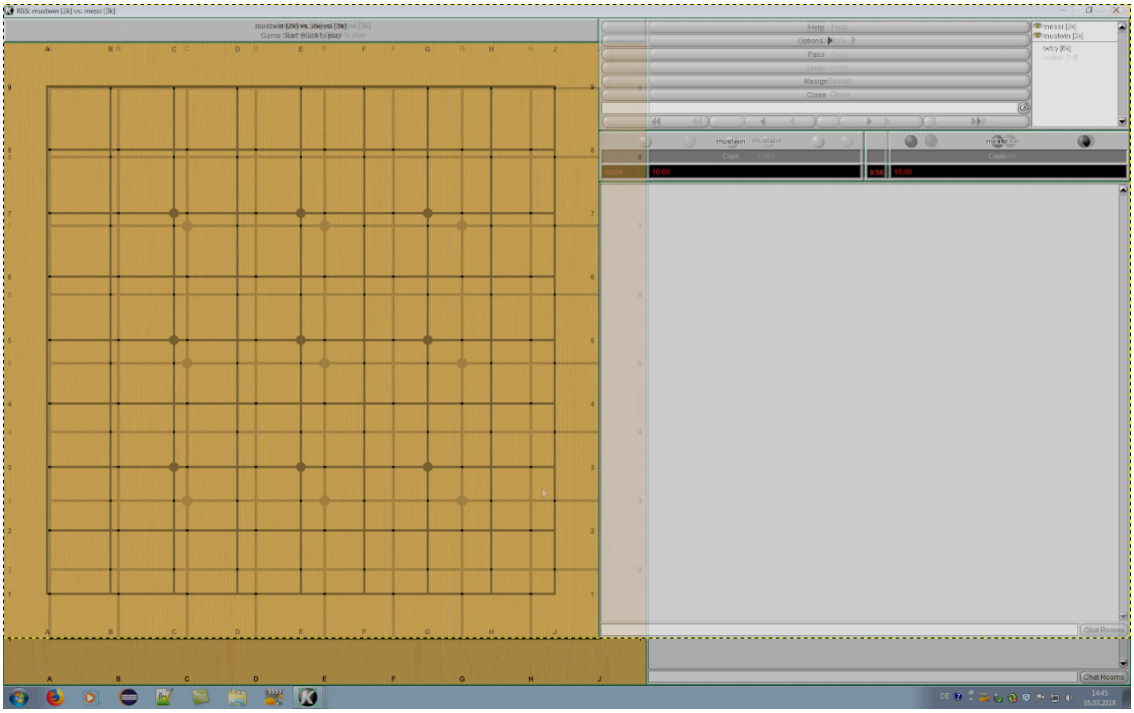


**Abbildung 5.14:** Unterschiede in den Spielfeldmaßen und Koordinaten zwischen den Tracker-Geräten. Die Aufzeichnung des Geräts mit kleinerem Monitor ist von einer dünnen gestrichelten Linie umgeben. Die Koordinaten der Gaze-Punkte eines Datensatzes müssen angepasst werden, damit sie im Spielfeld des anderen Screen-Recordings an die richtige Stelle fallen.

#### Hilfsmittel: System-Zeit

Falls der Datensatz der Recordings die absolute Systemzeit für die Events oder zu Beginn der Aufnahme enthält, und die PCs, auf denen die Aufnahmen getätigt wurden, mit einem Zeitserver synchronisiert waren, lässt sich dieser Umstand ausnutzen. In diesem Fall kann die Differenz der eingebetteten System-Zeit als Richtwert für nachfolgende Methoden genutzt werden, um längere Suchvorgänge zu verkürzen, indem die Suchbereiche durch den Schätzwert verkleinert werden.

#### Hilfsmittel: Eingebettete Mausklicks

Sollten im Datensatz Informationen zu Mausklicks enthalten sein, so können diese genutzt werden, um gezielt in zeitlich eingeschränkten Bereichen nach Indikatoren für einen Zug des Spielers zu suchen.

#### »KGS Go«-spezifische Methode

Die erste der zwei Methoden wurde speziell für den hier behandelten Anwendungsfall konstruiert. Die Grundidee hierfür sieht vor, die Farbe der Pixel an einer definierten Position in Screen Recording *A* über viele Frames hinweg zu überwachen und das Setzen eines Steines durch Farbänderung der

Pixel zu erkennen. Wird eine Farbänderung erkannt, wird zunächst über den Index des erkannten Frames und die Bildfrequenz (fps) der Videos ein Zeitstempel ermittelt. Dann nutzt die Methode, falls applikabel, den System-Zeit-Schätzwert, um den ermittelten Zeitstempel anzupassen und möglichst nah an das zu findende Frame aus Screen Recording *B* heranzukommen. Im vorletzten Schritt sucht die Methode in einem Bereich um den ermittelten Zeitstempel in Screen Recording *B* die durch das Setzen des gleichen Steines erzeugte gleiche Farbänderung wie in Screen Recording *A*. Zum Schluss wird die tatsächliche zeitliche Differenz aus den Indizes der gefundenen Frames und den Bildfrequenzen der Videos ermittelt und der Eye-Tracker-Recording-Instanz des Gastdatensatzes hinzugefügt.

Für einen Pseudocode-Algorithmus, siehe Listing 4 im Anhang.

Diese Methode funktioniert zwar potenziell auch mit Aufzeichnungen anderer Spiele, jedoch verbaut die Implementierung des Verfahrens ein paar Besonderheiten, die der universellen Kompatibilität entgegenwirken. So prüft das Programm beispielsweise die Farbe der Pixel an zwei verschiedenen Positionen, um auszuschließen, dass durch einen Mauszeiger im Screen Recording an einer Position eine farbliche Veränderung fälschlicherweise als Setzen eines Steines gewertet wird. Über zwei Positionen, die weit genug voneinander entfernt sind aber dennoch komplett in die Fläche eines Steines fallen, lassen sich solche *false positives* vermeiden. Somit besteht eine gewisse Abhängigkeit von der Form der Spielfiguren bzw. Spielsteine. Zusätzlich basiert der Erkennungsmechanismus für eine ausreichend große Farbänderung an den gegebenen Positionen auf einer Farbabstandsberechnung. Hierfür wird die neue Farbe mit einem Beigeton der Holztextur des Spielbretthintergrundes von KGS Go verglichen. Erst, wenn der ermittelte Farbabstand eine untere Schranke übersteigt, erkennt das Programm die Farbänderung als Setzen eines Steines an. Dies ist notwendig, da die KGS Go-Software eine clientseitige Vorschau für das Setzen eines Steines implementiert, bei der an der Mausposition auf dem Spielbrett ein halbtransparenter Stein angezeigt wird. Ist der farbliche Mindestabstand nicht groß genug definiert, wird dieser Vorschau-Effekt irrtümlicherweise als Setzen eines Steines erkannt. Ist er hingegen zu groß, wird das tatsächliche Setzen des Steines nicht erkannt. Zur Berechnung des Abstandes zweier Farben wird eine leicht abgeänderte Version des Ciede2000-Algorithmus[13] genutzt, bei dem die Helligkeit lediglich mit 10% gewichtet wird und somit Unterschiede im Farbton eine größere Auswirkung auf das Ergebnis haben. Dadurch wird die vom halbtransparenten weißen Stein aufgehellte bzw. vom halbtransparenten schwarzen Stein abgedunkelte Spielbrettfarbe als relativ nah am Referenzfarbwert eingestuft und nicht als endgültiges Setzen des Steins interpretiert. Die Bild-Puffer werden vor dem Vergleich in den Lab-Farbraum konvertiert.

#### Histogramm-Methode

Die zweite Methode wurde mit dem Ziel entworfen, auch andere Software neben »KGS Go« unterstützen zu können. An dieser Stelle kommen Histogramm-Vergleiche (siehe Abschnitt 2.4) zum Einsatz. Der Algorithmus beginnt damit, das erste Frame *A* einer neuen Spielbrettkonfiguration bzw. eines Zuges zu finden. Hierfür benötigt der Algorithmus einen initialen Index eines Frames, das zwischen zwei Spielzügen liegt. Über mehrere Histogramm-Vergleiche nach *Correlation* (siehe Abschnitt 2.4) mit unmittelbar vorhergehenden Frames wird der Index so lange verringert,

[13] Mehr Informationen zum Ciede2000-Algorithmus in Johnson *et. al* [JF03]

bis eine vordefinierte Mindestabweichung überschritten wird. Dies wird als letztes Frame der vorhergehenden Spielbrettkonfiguration interpretiert. Analog zur »KGS Go«-spezifischen Methode wird basierend auf dem ermittelten Index, der Bildfrequenz des Screen-Recordings und der System-Zeit ein Zeitstempel berechnet, der als Orientierungswert zur Suche im Gast-Screen-Recording dient. Dann wird über einen Histogramm-Vergleich mit Frames aus dem Gast-Screen-Recording der zu *A* ähnlichste Frame *B* nach *Correlation* in einem Suchraum von ca. 150 Frames vor und nach dem ermittelten Orientierungswert gesucht. Im letzten Schritt wird wie im ersten Schritt rückwärts das erste Frame des Zuges gesucht, der in *B* zu sehen ist. Auch hier werden wieder Histogramm-Vergleiche genutzt.

Da die Bildschirmaufzeichnungen eine recht hohe Auflösung aufweisen, werden die Histogramme nicht über ganze Frames in einem Stück gebildet. Mit steigender Auflösung sinkt die Verlässlichkeit des Histogramm-Vergleiches, da die Menge an möglichen ungleichen Bildern, die ähnliche Histogramme produzieren, ebenfalls steigt. Um dieses Problem zu umgehen, unterteilt der Algorithmus die Frames erst in ein Gitter aus Zellen, die dann jeweils mit dem Gegenüber aus einem anderen Frame verglichen werden. Je nach gewählter Gittergröße (16x16 bis 20x20) werden bei einem Vergleichsvorgang zweier Frames tatsächlich zwischen 256 und 400 Histogramm-Vergleiche durchgeführt. Die zwei Metriken, nach denen dieses Verfahren Unterscheidungen trifft, sind *höchster Ähnlichkeitsgrad nach Correlation* für die Findung des ähnlichsten Frames *B* aus dem Gast-Screen-Recording zu Frame *A* aus dem Host-Screen-Recording, und *mindestens 60% Abweichung in mindestens einer Gitterzelle* zur Findung des ersten Frames des vorhergehenden Zuges. Für letzteren Unterscheidungsfall kann je nach Auflösung des Screen-Recordings und gewählter Gittergröße auch eine Metrik mit (evtl. geringerer) Mindestabweichung in mehr als einer Gitterzelle sinnvoll sein. Dadurch kann beispielsweise durch einen mitaufgezeichneten Maus-Zeiger erzeugten *false positives* vorgebeugt werden–insbesondere dann, wenn der Mauszeiger im Verhältnis zur Größe der Zellen groß und im Verhältnis zur Größe der Spielsteine klein ist.

Für einen Pseudocode-Algorithmus, siehe Listing 5 im Anhang.

#### Mögliche Probleme

Es ist denkbar, dass die ermittelte Zeitverschiebung durch unvorhersehbare Faktoren von einem hypothetischen Optimalwert abweicht. Dies kann zum Beispiel dann passieren, wenn die aufgezeichneten Spielszenarien von einer Netzwerkverbindung und Kommunikation über das Internet abhängen. Kommt es in diesem Fall zu Verbindungsproblemen, kann dies unter anderem dazu führen, dass zeitliche Abstände zwischen dem „Setzen“ zweier gleicher Steine zwischen zwei verschiedenen Bildschirmaufzeichnungen derselben Partie inkonsistent sind. Da sich beide Methoden aus Performancegründen weitestgehend an einem einzigen Zug orientieren, sind geringfügige Abweichungen von einem hypothetischen Optimalwert einer zeitlichen Verschiebung nicht komplett auszuschließen.

## 5.9 Projekt-IO

Über den Menüpunkt „Datei“ → „Projekt speichern unter...“ in der Hauptansicht kann eine erzeugte Konfiguration mitsamt veränderten Einstellungen gespeichert werden. Diese kann dann, wie in Abschnitt 5.3 beschrieben, zu einem späteren Zeitpunkt wiederhergestellt werden. Wenn der/die

Nutzer/in ein Projekt speichert, wird ein Objekt der Klasse „Project“ serialisiert und über einen ObjectOutputStream in eine Datei geschrieben. Dieses Objekt enthält die Pfade zu allen ursprünglich geladenen Datensätzen, Screen-Recordings sowie festgelegte Spielfeld-Bereiche und ermittelte zeitliche Verschiebungen zwischen den Datensätzen. Außerdem besitzt das Project-Objekt ein ebenfalls serialisiertes Objekt des Typs „Preferences“, einer Struktur- bzw. Behälterklasse, die sämtliche Einstellungswerte enthält. Die Daten werden in eine vom Nutzer bestimmte Datei mit der Endung „.etproj“ geschrieben. Die Inhalte und gefilterten/transformierten Daten aus den Eye-Tracking-Datensätzen selbst sind nicht in der resultierenden Datei enthalten, da die dabei erzeugte Datenmenge unter Umständen sehr groß werden könnte und Redundanz vermieden werden soll. Stattdessen wird nur festgehalten, wo ein geladener Datensatz im Dateisystem lag, und sämtliche Parsing-, Transformations- und Filtervorgänge werden beim Laden des Projekts erneut durchgeführt. Tritt der Fall ein, dass das Programm eine Datei nicht am ursprünglichen Ort auffinden kann, wird der/die Nutzer/in gebeten, den aktualisierten Pfad anzugeben. Beim Laden führt das Programm einige Integritätsprüfungen durch die sicherstellen, dass die Version der Datei mit der Version des Programmes kompatibel ist und dass alle nötigen Informationen zur Wiederherstellung des Projekts vorhanden sind. Nach erfolgreich abgeschlossenem Wiederherstellungsprozess findet der/die Nutzer/in sich in der Hauptansicht wieder. Gespeichert werden kann ein Projekt zu jedem Zeitpunkt über die Hauptansicht, sobald mindestens ein Eye-Tracking-Datensatz geladen ist. Da die Hauptansicht erst verfügbar ist, sobald ein Host-Screen-Recording geladen ist, und in jedem Fall als erstes der Host-Eye-Tracking-Datensatz importiert wird, enthält der Mindestumfang eines gespeicherten Projekts also den Pfad zum gewählten Host-Screen-Recording und den Pfad des Host-Eye-Tracking-Datensatzes inklusive aller Standardwerte der Einstellungen.

# 6 Analyse und Auswertung

In diesem Kapitel wird das entwickelte Programm dazu genutzt, drei aufgezeichnete Partien des Strategiebrettspiels Go zu analysieren. Abschließend folgt eine Auswertung der Ergebnisse und der Leistungsfähigkeit des Programms im Hinblick auf die Anforderungen.

## 6.1 Analyse

Im Folgenden werden nacheinander die drei Partien analysiert. Ausgetragen wurden diese an zwei Terminen zwischen den gleichen zwei Spielern. Bildmaterial von Spielszenen entstammt immer der Perspektive von Spieler 1. Die Kennzeichnung „Gerät 1“ bzw. „Gerät 2“ bezeichnet den Sitzplatz bzw. den Eye-Tracker, wohingegen „Proband 1“ bzw. „Proband 2“ die jeweilige Person identifiziert. Da die Kontrahenten zwischen den Terminen die Sitzplätze wechselten, ist diese Unterscheidung notwendig. Aus einer Befragung der Probanden geht hervor, dass Proband 2 das Spiel bereits länger spielt als Proband 1 und daher etwas erfahrener ist. Da in Go immer der/die Spieler/in mit den schwarzen Steinen anfängt und Proband 1 in den drei analysierten Partien immer die schwarzen Steine spielt, ist Proband 1 in allen drei Partien als erster am Zug. Für die Analyse kommen zwei verschiedene Ansätze zum Einsatz. Partie 1 wird mittels MinDist-Plot als Leitfaden analysiert, während sich die Analyse von Partie 2 und Partie 3 nah an das Spielgeschehen hält.

### 6.1.1 Partie 1

Die erste Partie wird hauptsächlich anhand des MinDist-Plots (Abbildung 6.1) analysiert. Nacheinander werden alle größeren Bereiche betrachtet, in denen die Fixationen der Probanden nah beieinander liegen (<150 px). Diese werden dann in einen Kontext eingeordnet. Die Attention Maps (Abbildung 6.2, Abbildung 6.3) zeigen, dass beide Spieler dem nordwestlichen Quadranten in dieser Partie kaum lange Beachtung geschenkt haben.

#### Aufbau / Identifikation

**Gerät 1**: Proband 1 (Schwarze Steine, Blaue Fixationen)
**Gerät 2**: Proband 2 (Weiße Steine, Rote Fixationen)
**Länge der Partie**: 3 Minuten und 2 Sekunden
**Sieger**: Proband 1

### MinDist-Plot

**Abbildung 6.1:** MinDist-Plot der ersten Partie. Türkis: Distanz der Gaze-Punkte der Probanden <150 px. Rot: Distanz der Gaze-Punkte der Probanden >=150 px. Gelb: Mind. ein Proband schaut nicht auf das Spielfeld. Grau: Keine Daten vorhanden für mind. einen Probanden. Bei mehreren Werten pro Spalte (bzw. Pixel) entscheidet die Mehrheit. In der Nachbearbeitung vertikal gestreckt, sodass Farben etwas besser zu erkennen sind. Anfänglicher dunkelgrauer Abschnitt verbildlicht den Zeitversatz zwischen den zwei Recordings. Zeitraum: 4m 21s.

### Attention Maps

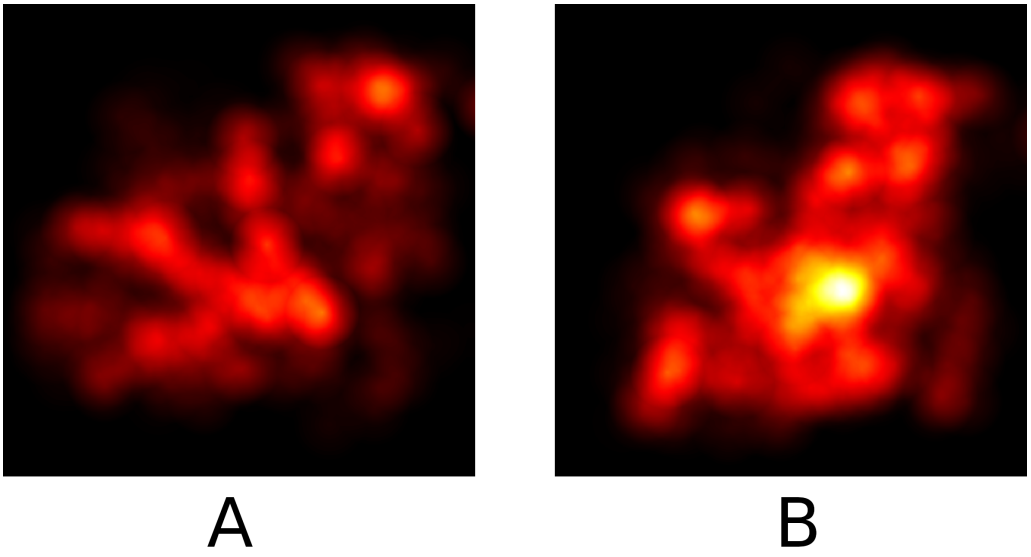



**Abbildung 6.2:** Attention Maps der ersten Partie, erstellt über die Gesamtdauer der Partie. Dunkel/Schwarz: Wenig bzw. keine Aufmerksamkeit. Hell/Gelb: Viel bzw. hohe Aufmerksamkeit. **A**: Proband 1, **B**: Proband 2

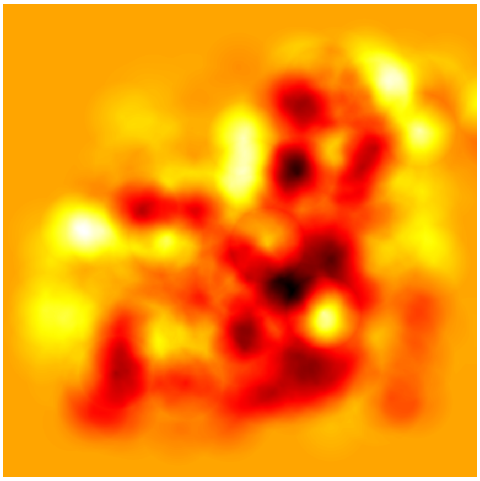


**Abbildung 6.3:** Differenz der Attention Maps der ersten Partie, erstellt über die Gesamtdauer der Partie. Orange (siehe Ecken und Randbereiche): Gleich viel Aufmerksamkeit. (Hell)Gelb: Mehr Aufmerksamkeit bei Proband 1. Schwarz/Rot: Mehr Aufmerksamkeit bei Proband 2. Map von Proband 2 subtrahiert von Map von Proband 1: Abbildung 6.2.A − Abbildung 6.2.B

#### Bereich 1

**Abbildung 6.4:** Erster interessanter Bereich in der ersten Partie. Schranke: 150px.

Der erste Bereich (Abbildung 6.4), in dem die Fixationen ein paarmal nah beieinander liegen, findet sich wenige Züge nach Spielbeginn. Die Blicke wandern um die Mitte des Spielfeldes, einem taktisch wertvollen Bereich, in dem zu diesem Zeitpunkt noch sehr viel möglich ist (siehe Abbildung 6.5).

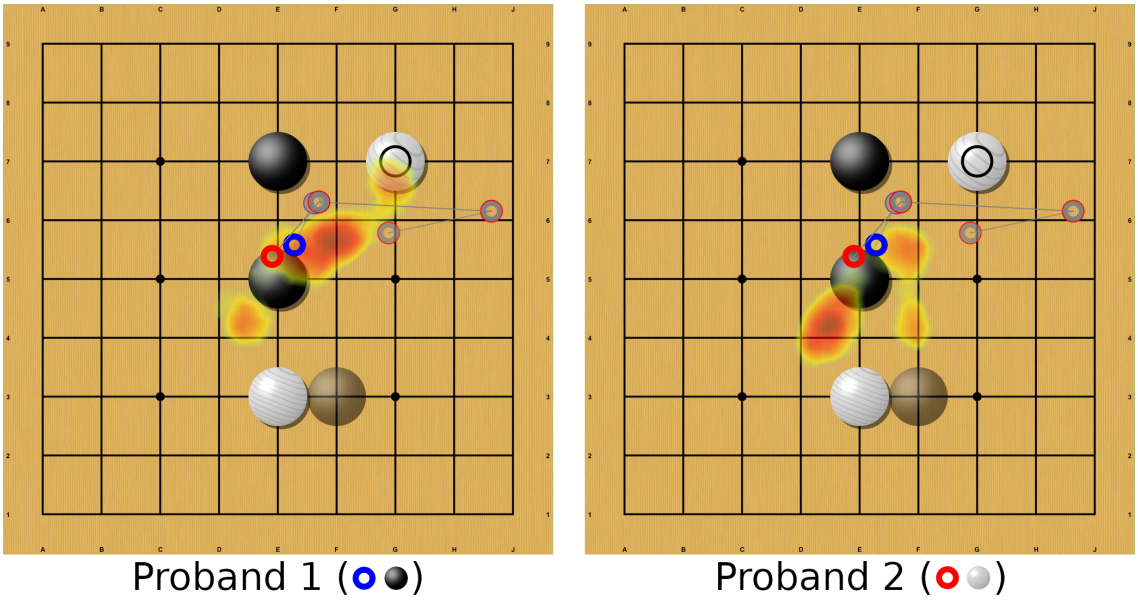



**Abbildung 6.5:** Attention-Map-Overlay des ersten interessanten Bereiches in der ersten Partie, aggregiert nach Häufung. Die Aufmerksamkeit beider Probanden konzentriert sich auf die Spielfeldmitte, während sich die Fixationen treffen.

#### Bereich 2

**Abbildung 6.6:** Zweiter interessanter Bereich in der ersten Partie. Schranke: 150px.

Im zweiten interessanten Bereich (Abbildung 6.6) ringen die Spieler um die rechte Spielfeldhälfte. Dabei wandern die Blicke hauptsächlich zwischen zwei Schlüsselpositionen hin und her (Abbildung 6.7). Tatsächlich wird der nächste Stein nach diesem zeitlichen Abschnitt auf der anderen Seite des Spielfeldes gesetzt, wodurch das Gefecht ein Ende findet.

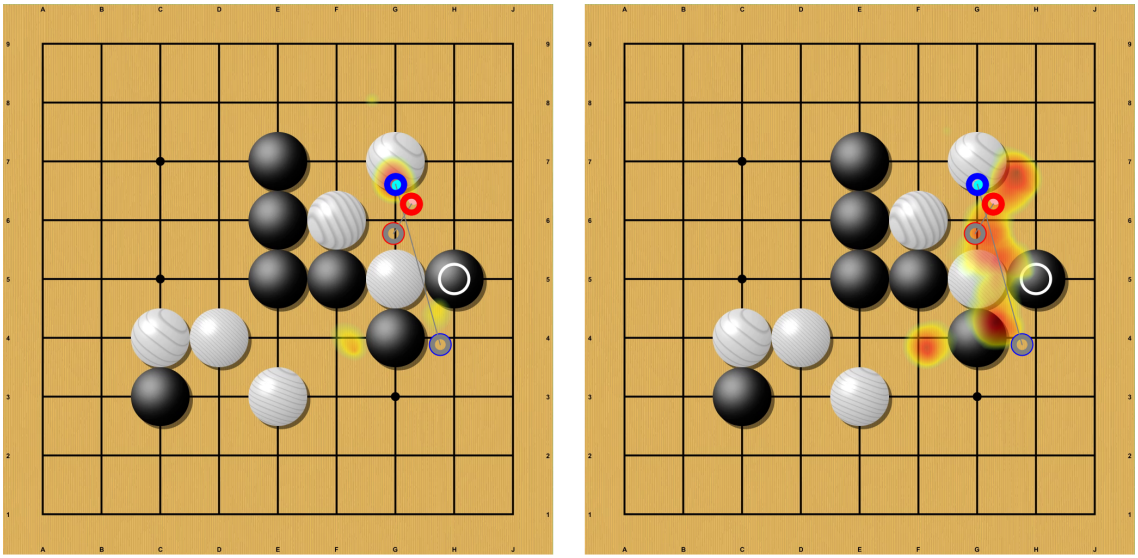



**Abbildung 6.7:** Attention-Map-Overlay des zweiten interessanten Bereiches in der ersten Partie, aggregiert nach Dauer. Beide Probanden achten verstärkt auf den Bereich rechts neben der Spielfeldmitte. Die Fixationen liegen sowohl nahe der Mitte des nordöstlichen Quadranten, als auch unterhalb rechts der Spielfeldmitte jeweils zum gleichen Zeitpunkt nah beieinander.

#### Bereich 3

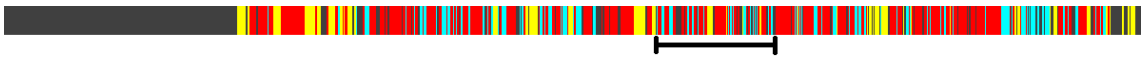


**Abbildung 6.8:** Dritter interessanter Bereich in der ersten Partie. Schranke: 150px.

Während in Bereich 3 (Abbildung 6.8) ein Gefecht um den unteren linken Quadranten wütet, begegnen sich die Fixationen der Probanden kurz in der Mitte der linken Spielfeldhälfte, wo noch einige Felder frei sind. Kurz darauf schauen beide Spieler zum gleichen Zeitpunkt auf ein Feld nahe der Spielfeldecke, bis Proband 2 dort einen Stein platziert (siehe Abbildung 6.9).

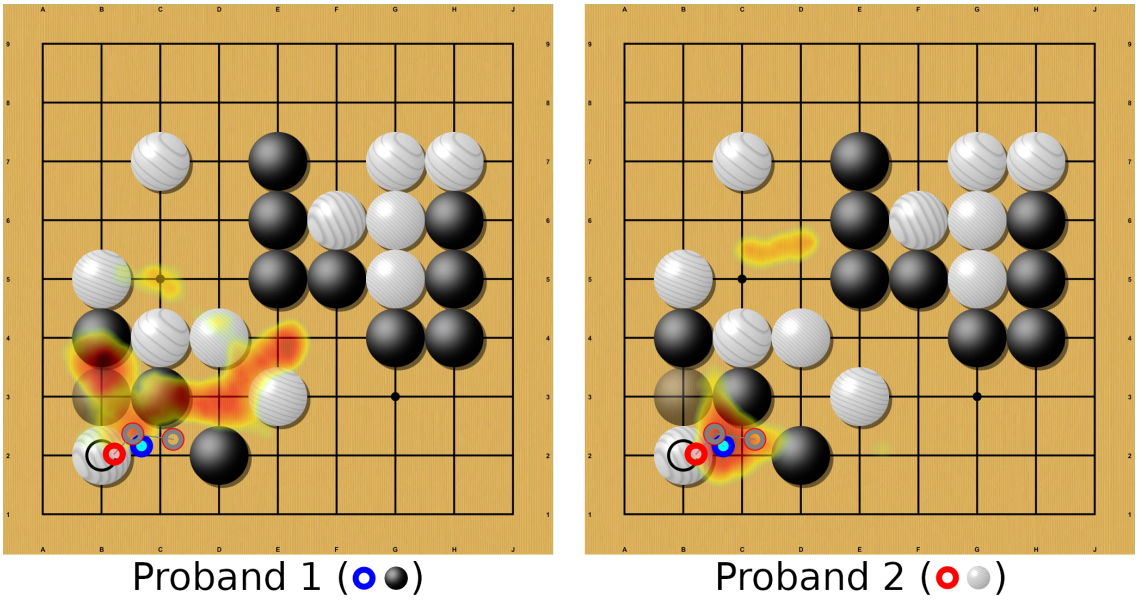



**Abbildung 6.9:** Attention-Map-Overlay des dritten interessanten Bereiches in der ersten Partie, aggregiert nach Häufung. Ein Gefecht um den südwestlichen Quadranten findet statt.

### Bereich 4

**Abbildung 6.10:** Vierter interessanter Bereich in der ersten Partie. Schranke: 150px.

Im vierten und letzten interessanten Bereich (Abbildung 6.10) kämpfen die Spieler hauptsächlich um den nordöstlichen Quadranten. Die Fixationen häufen sich nur etwa ein Feld voneinander entfernt (siehe Abbildung 6.11) und begegnen sich zwischen beiden Hotspots mehrfach. Proband 2 versucht hier offenbar, Schwarz eine Falle zu stellen, doch Proband 1 fällt nicht darauf herein.

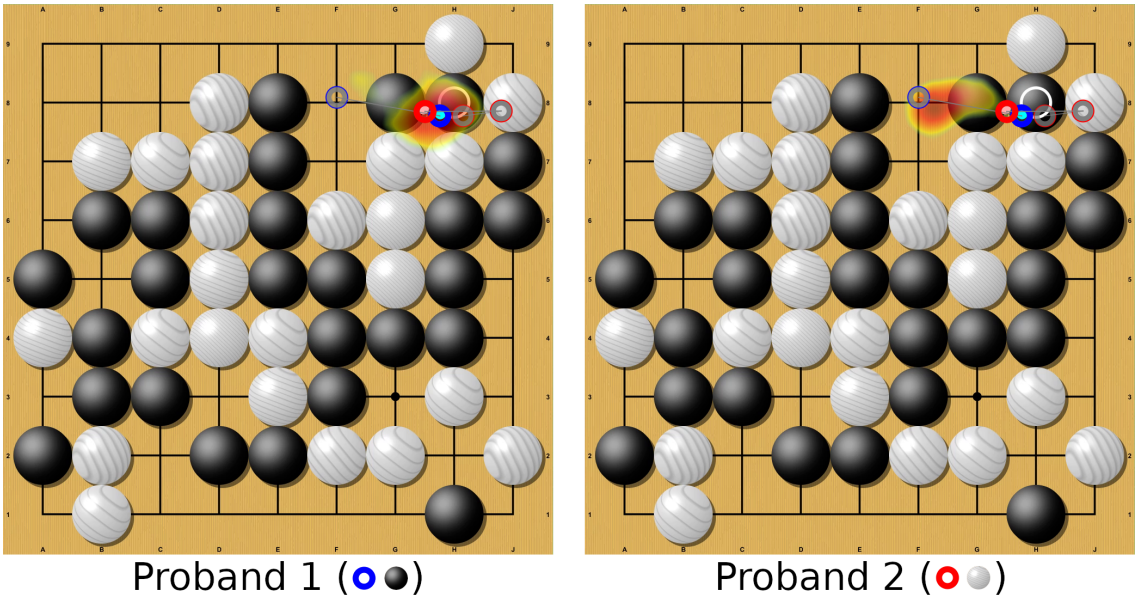



**Abbildung 6.11:** Attention-Map-Overlay des vierten interessanten Bereiches in der ersten Partie, aggregiert nach Häufung. Ein Gefecht um den nordöstlichen Quadranten findet statt.

### 6.1.2 Partie 2

In diesem Abschnitt wird die zweite Partie analysiert. Bei dieser Analyse liegt der Fokus nah am Spielgeschehen. Diese Partie ist interessant, weil sich Proband 1 recht früh einen Vorteil erspielt und diesen bis zum Ende der Partie behaupten und sogar ausbauen kann. In den Attention Maps (Abbildung 6.13, Abbildung 6.14) wird sichtbar, dass sich die Aufmerksamkeit von Proband 1 recht gleichmäßig über das gesamte Spielfeld verteilt, während sich bei Proband 2 zwei klare Hotspots und kaum nennenswerte Berücksichtigung des oberen linken Quadranten abzeichnen. Der MinDist-Plot (siehe Abbildung 6.12) zeugt von einer hohen Datenqualität. Es gibt kaum Bereiche, in denen die Geräte die Blickrichtung der Probanden nicht aufzeichnen konnten.

#### Aufbau / Identifikation

**Gerät 1**: Proband 1 (Schwarze Steine, Blaue Fixationen)
**Gerät 2**: Proband 2 (Weiße Steine, Rote Fixationen)
**Länge der Partie**: 2 Minuten und 58 Sekunden
**Sieger**: Proband 1

### MinDist-Plot

**Abbildung 6.12:** MinDist-Plot der zweiten Partie. Türkis: Distanz der Gaze-Punkte der Probanden <150 px. Rot: Distanz der Gaze-Punkte der Probanden >=150 px. Gelb: Mind. ein Proband schaut nicht auf das Spielfeld. Grau: Keine Daten vorhanden für mind. einen Probanden. Bei mehreren Werten pro Spalte (bzw. Pixel) entscheidet die Mehrheit. In der Nachbearbeitung vertikal gestreckt, sodass Farben etwas besser zu erkennen sind. Zeitraum: 3m 17s.

### Attention Maps

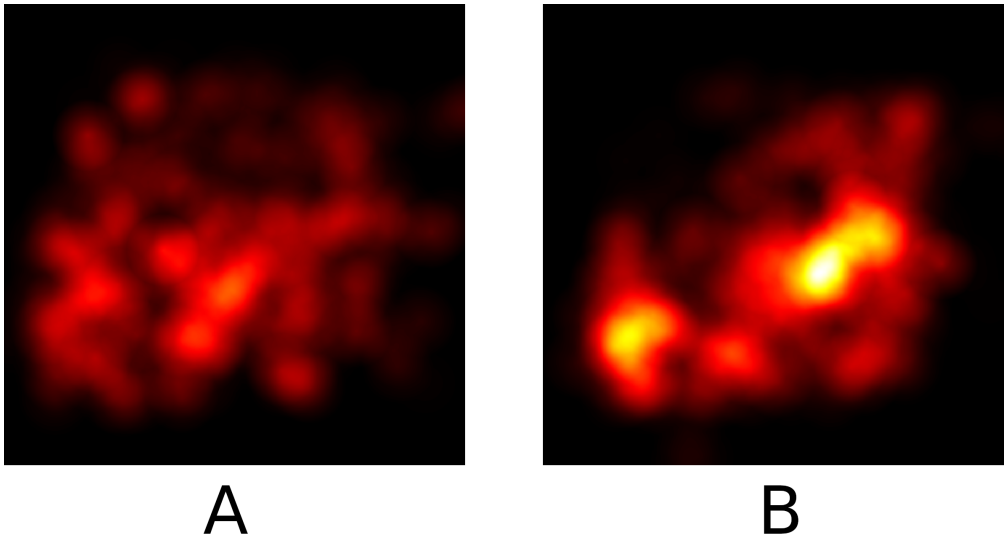



**Abbildung 6.13:** Attention Maps der zweiten Partie, erstellt über die Gesamtdauer der Partie. Dunkel/Schwarz: Wenig bzw. keine Aufmerksamkeit. Hell/Gelb: Viel bzw. hohe Aufmerksamkeit. **A**: Proband 1, **B**: Proband 2

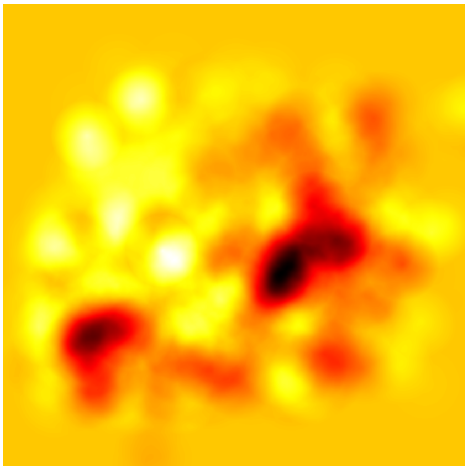


**Abbildung 6.14:** Differenz der Attention Maps der zweiten Partie, erstellt über die Gesamtdauer der Partie. Orange (siehe Ecken und Randbereiche): Gleich viel Aufmerksamkeit. (Hell)Gelb: Mehr Aufmerksamkeit bei Proband 1. Schwarz/Rot: Mehr Aufmerksamkeit bei Proband 2. Map von Proband 2 subtrahiert von Map von Proband 1: Abbildung 6.13.A − Abbildung 6.13.B

#### Spielverlauf, Beginn der Partie

Beide Spieler verteilen ihre Steine in den ersten vier Zügen lose über das Spielfeld und bilden eine lockere Anordnung mit diversen Freiheiten und großer Raumabdeckung (siehe Abbildung 6.15). Schwarz besetzt im ersten Zug die Mitte.

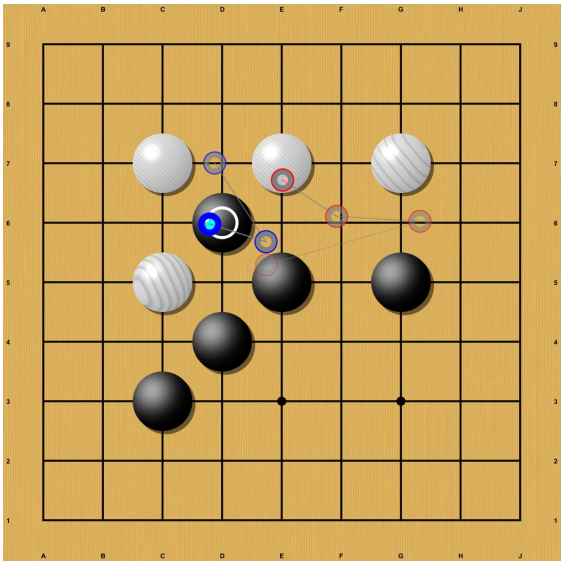


**Abbildung 6.15:** Die Probanden verteilen ihre Steine so über das Spielfeld, dass mit wenigen Zügen Einfluss auf ein möglichst großes Gebiet genommen werden kann. Dazu lassen sie jeweils eine Freiheit zwischen zwei Steinen frei.

### Spielverlauf

Nach anfänglichem Aufbau kommt es zum ersten Kontakt, als Weiß ab dem fünften Zug die Fronten im Nordwesten neben einem schwarzen Stein verdichtet. Proband 2 unternimmt einige Züge lang Versuche, eine bedrohliche Situation im oberen linken Quadranten zu erschaffen, doch Schwarz weicht lieber aus und plant eine Expansion (siehe Abbildung 6.16).

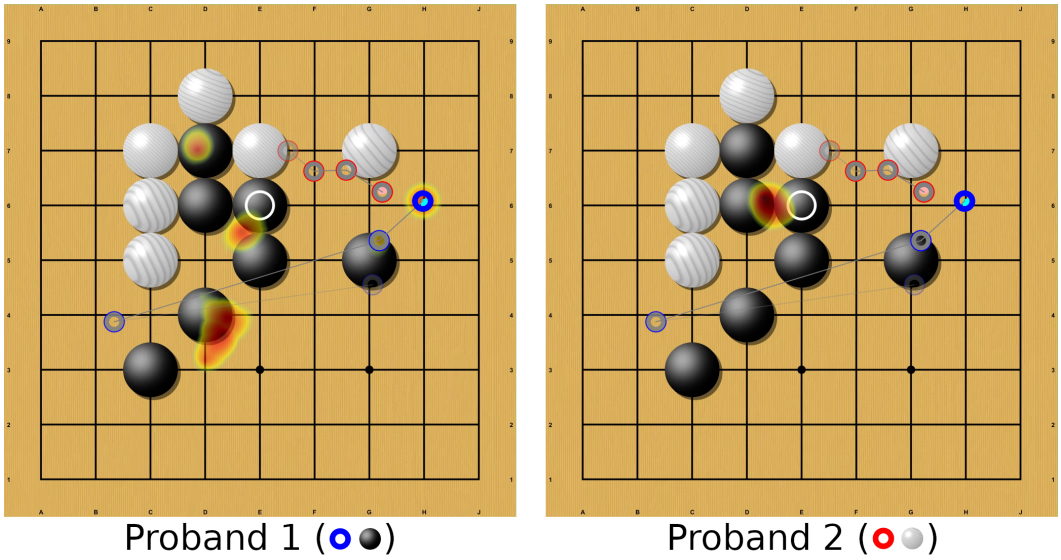



**Abbildung 6.16:** Erste Auseinandersetzung in der zweiten Partie. Die Initiative des Angriffs geht von Weiß aus. Schwarz scheint währenddessen mit dem Gedanken zu spielen, die Präsenz im Osten zu stärken. Proband 1 weicht aus und achtet verstärkt auf die eigenen Steine. Proband 2 achtet mehr auf die Steine des Gegners.

Nachdem Proband 1 wie geplant einen Stein nahe des rechten Spielfeldrandes platziert und Proband 2 die Lücke zum Stein in der Mitte des nordöstlichen Quadranten schließt, wandert der Fokus für zwei Züge durch den noch relativ leeren südöstlichen Quadranten, in dem beide Spieler eine Zweiergruppe bilden. Anschließend attackiert Proband 2 erneut einen losen schwarzen Stein, diesmal im unteren linken Quadranten. Proband 1 überlegt kurz, defensiv zu spielen und den Stein an die zentrale Gruppe anzuschließen, doch geht dann zum Gegenangriff über und nötigt Proband 2 so zur Vorsicht. Nach reichlicher Bedenkzeit wählt Weiß die sichere Variante (siehe Abbildung 6.17). Dies ermöglicht Schwarz das Schließen der Lücke, die zuvor lange betrachtet wurde (Hotspot in Abbildung 6.17 A).

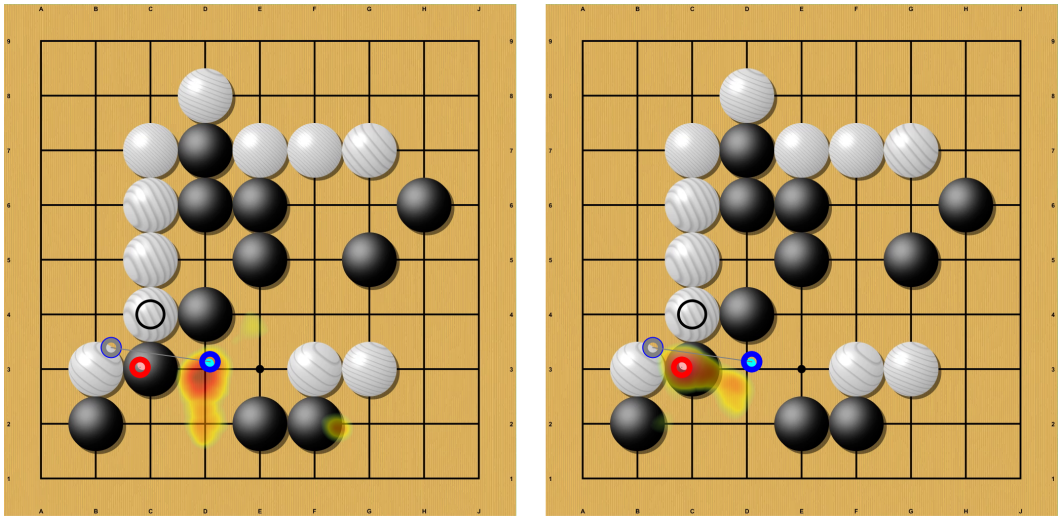



**Abbildung 6.17:** Zweite Auseinandersetzung in der zweiten Partie. Nach einem Angriff von Weiß geht Proband 1 zum Gegenangriff über. Dies zwingt Proband 2 zu einer etwas defensiveren Maßnahme. Proband 1 achtet nach wie vor mehr auf die Räume zwischen den eigenen Steinen. Proband 2 plant sehr offensiv und hat nur den gegnerischen Stein im Blick.

Die Szene scheint sich daraufhin zu wiederholen, denn Proband 2 holt erneut weit aus und greift den noch ungeschützten schwarzen Stein ganz unten links an. Dank starker Präsenz im Quadranten und kürzlichem Erfolg fühlt sich Proband 1 ermutigt, wieder einen Gegenangriff zu starten. Die Anordnung der schwarzen Steine liefert dabei einen taktischen Vorteil und garantiert nun mindestens eine Gefangennahme, die Weiß nicht mehr verhindern kann. Stattdessen setzt Proband 2 nun auf einen längerfristigen Punkteaustausch am linken Spielfeldrand (siehe Abbildung 6.18).

Der weiße Stein wird wie erwartet gefangen. Da die Ko-Regel eine unmittelbare Rückeroberung verbietet, ist Proband 2 gezwungen, einen Zug lang woanders hin zu spielen und damit den Zustand des Spielbretts zu verändern. Das Zentrum des Spielfeldes wird komplett von Schwarz dominiert, daher spielt Weiß erneut abseits der eigenen Gruppe rechts neben die Mitte. Im Genuss des Momentums reagiert Proband 1 direkt und beginnt mit der Isolation des eben gesetzten einsamen weißen Steines. Proband 2 hat indes nur das verlorene Feld im Blick (siehe Abbildung 6.19) und erobert es zurück.

Erneut greift die Ko-Regelung und Proband 1 nutzt den Zug, um dem schwarzen Stein ein Feld darüber mehr Freiheiten zu schaffen. Einen Zug später erobert Schwarz den Spielfeldrand erneut zurück. Es steht zwei zu eins und Schwarz kontrolliert den größeren Teil des Spielfeldes. Nach einem Versuch von Weiß, die Gruppe im südöstlichen Quadranten zu stärken und von dort aus Druck auf die Mitte auszuüben, fängt Schwarz schließlich auch den einsamen weißen Stein nahe der Mitte. Ein weiteres Mal fängt Weiß den schwarzen Stein am Spielfeldrand, kann dadurch aber nicht mehr ausgleichen. Der Ko-Regel folgend setzt Proband 1 in den Quadranten oben rechts, um dort etwas Druck auf den Gegner aufzubauen. Nun bietet sich die Gelegenheit für Weiß, in Führung

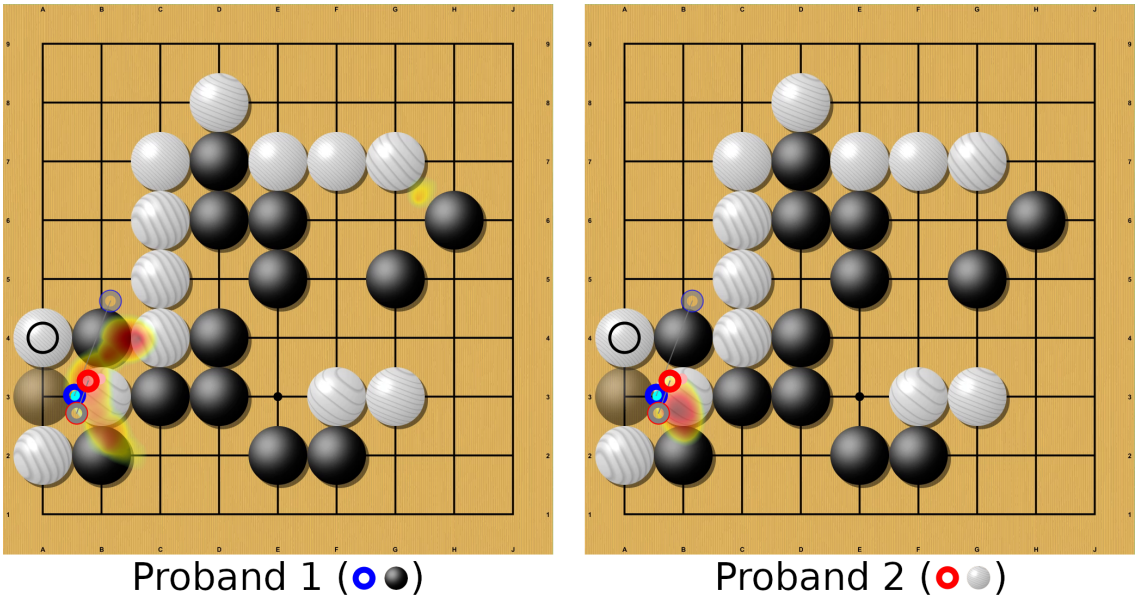



**Abbildung 6.18:** Dritte Auseinandersetzung in der zweiten Partie. Weiß übernimmt sich erneut etwas und zieht die Linien dünn. Proband 1 unternimmt erneut einen Gegenangriff, der mindestens eine Gefangennahme garantiert. Das Attention-Map-Overlay von Proband 1 zeugt vom Plan der Einkreisung. Proband 2 erkennt die Situation, in der sich der eigene Stein befindet, erst spät.

zu gehen. Am linken Rand sitzen zwei beinahe komplett umstellte schwarze Steine, durch deren Gefangennahme Proband 2 auch den ständigen Austausch im unteren linken Quadranten mit einem Unentschieden beenden könnte, doch die Gelegenheit bleibt ungenutzt (siehe Abbildung 6.20).

Zwei Züge später tritt diese Situation erneut auf, doch auch diesmal ist Proband 2 nicht mehr so aggressiv wie zu Beginn und stärkt lieber die Gruppe im Nordosten. Nachdem Schwarz zum vierten Mal an den linken Spielfeldrand setzt und so den fünften Punkt ergattert, spielt Weiß einen augenscheinlichen Opferzug zwischen drei schwarze Steine direkt unter der Mitte. Schwarz fängt diesen umgehend und Weiß wiederholt zum vierten Mal den Zug unten links. Es steht sechs zu vier. In dieser Schlussphase des Spiels suchen beide Spieler auf dem gesamten Spielfeld noch nach Gelegenheiten, Vorteile zu erspielen (siehe Abbildung 6.21). Nach wie vor scheint Proband 2 den zwei fangbaren schwarzen Steinen auf der linken Seite kaum Beachtung zu schenken. Im Norden, etwas abseits von der durch Proband 1 kontrollierten Spielfeldmitte, lässt sich noch ein einzelner schwarzer Stein fangen. Nachdem Schwarz zum fünften Mal an den linken Rand spielt, wieder das Ko erzeugt, und den Vorsprung von zwei Punkten behauptet, kapituliert Proband 1.

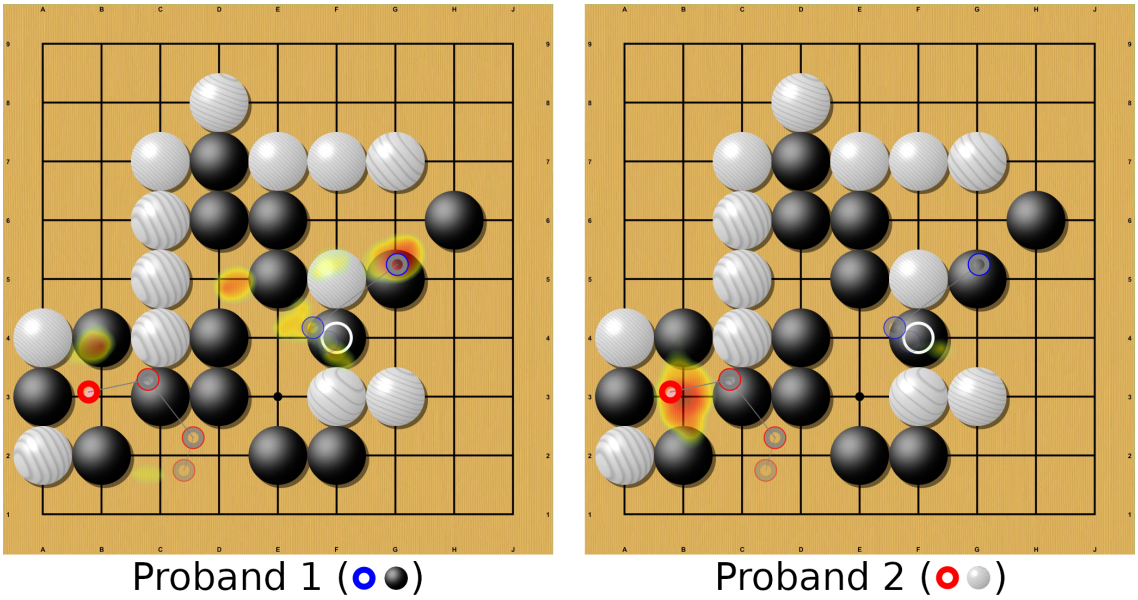



**Abbildung 6.19:** Vierte Auseinandersetzung in der zweiten Partie. Weiß möchte Anteile an der Spielfeldmitte. Proband 1 ist wachsam und möchte die Vorherrschaft in der Spielfeldmitte behaupten. Das Attention-Map-Overlay von Proband 2 zeigt weiterhin einen Konzentrationsschwerpunkt am Spielfeldrand.

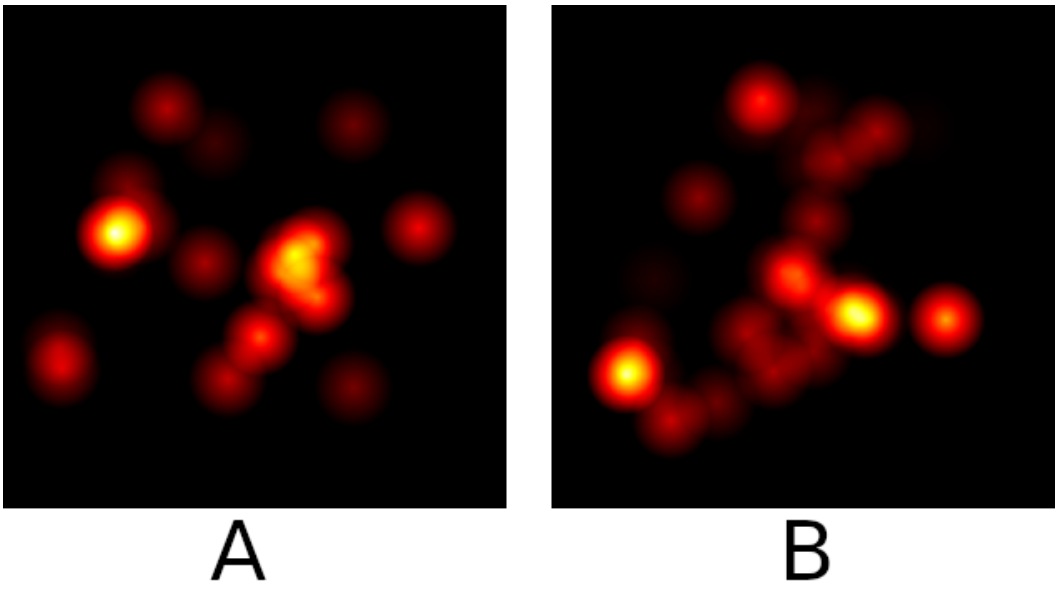



**Abbildung 6.21:** Attention Maps der Schlussphase in der zweiten Partie. Beide Spieler suchen noch nach Gelegenheiten, Punkte zu ergattern. **A**: Proband 1; **B**: Proband 2

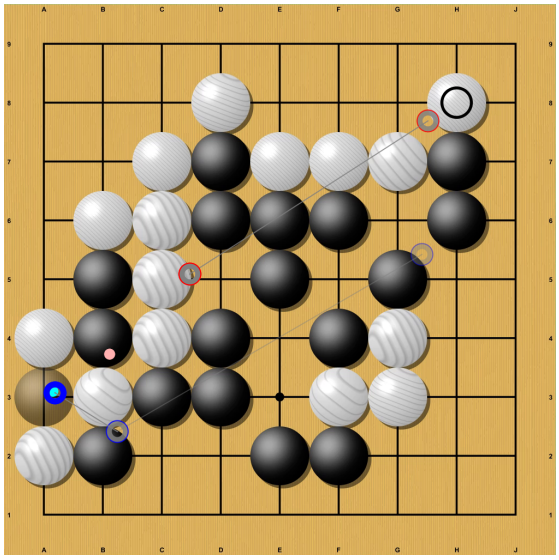


**Abbildung 6.20:** Verpasste Gelegenheit von Weiß in der zweiten Partie. Proband 2 (Weiß/Rot) hätte einen großen Teil der linken Spielfeldhälfte sichern, zwei Steine erobern und das Hin und Her am Rand mit einem Unentschieden beenden können. Stattdessen spielt Weiß oben rechts und Schwarz fängt weiter Steine.

### 6.1.3 Partie 3

In dieser Sektion folgt die Analyse der dritten Partie. Der Leitfaden ist erneut das Spielgeschehen, diesmal jedoch aufgrund der Länge der Partie unterteilt in mehrere Abschnitte (siehe Abbildung 6.22). Diese Partie wurde aufgrund der Spielfeldgröße und des MinDist-Plots (Abbildung 6.22) gewählt, der vergleichsweise viele türkise Abschnitte aufweist, während denen der Abstand der Blicke der Spieler unter der definierten Schranke lag. Gegen Ende der Partie lassen Qualität und Quantität der erfassten Daten jedoch leider etwas nach. In den Attention Maps (siehe Abbildung 6.23) ist zu erkennen, dass der Schwerpunkt der Partie wahrscheinlich im unteren rechten Quadranten bestritten wurde. Die Differenz (Abbildung 6.24) enthüllt, dass die Schwerpunkte der Aufmerksamkeit von Proband 1 etwas tiefer liegen als die Schwerpunkte der Aufmerksamkeit von Proband 2.

#### Aufbau / Identifikation

**Gerät 1**: Proband 2 (Weiße Steine, Blaue Fixationen)
**Gerät 2**: Proband 1 (Schwarze Steine, Rote Fixationen)
**Länge der Partie**: 8 Minuten und 45 Sekunden
**Sieger**: Proband 2

#### MinDist-Plot


**Abbildung 6.22:** MinDist-Plot der dritten Partie. Türkis: Distanz der Gaze-Punkte der Probanden <150 px. Rot: Distanz der Gaze-Punkte der Probanden >=150 px. Gelb: Mind. ein Proband schaut nicht auf das Spielfeld. Grau: Keine Daten vorhanden für mind. einen Probanden. Bei mehreren Werten pro Spalte (bzw. Pixel) entscheidet die Mehrheit. In der Nachbearbeitung vertikal gestreckt, sodass Farben etwas besser zu erkennen sind. Relativ viele dunkelgraue Bereiche, weil Eye-Tracker häufiger Gaze der Probanden verlieren als in den anderen Partien. Viele Markierungen in türkis in der ersten Hälfte der Partie, während der um Punkte im unteren rechten Quadranten des Spielfeldes gefochten wurde. Zeitraum: 9m 30s. **A**: Beginn der Partie; **B**: Gefecht im unteren rechten Quadranten; **C**: Gefecht im oberen rechten Quadranten, das bis in den Süden wandert und die Frontlinien vereint; **D**: Gefecht im oberen linken Quadranten; **E**: Parallele vertikale Erschließung der Spielfeldmitte mit Abzweigungen Richtung Südosten und Nordwesten; **F**: Gefecht um den unteren linke Quadranten, ausgehend von der Spielfeldmitte; **G**: Entscheidender Kampf um die Spielfeldmitte im oberen rechten Quadranten; **H**: Ende der Partie.

#### Attention Maps

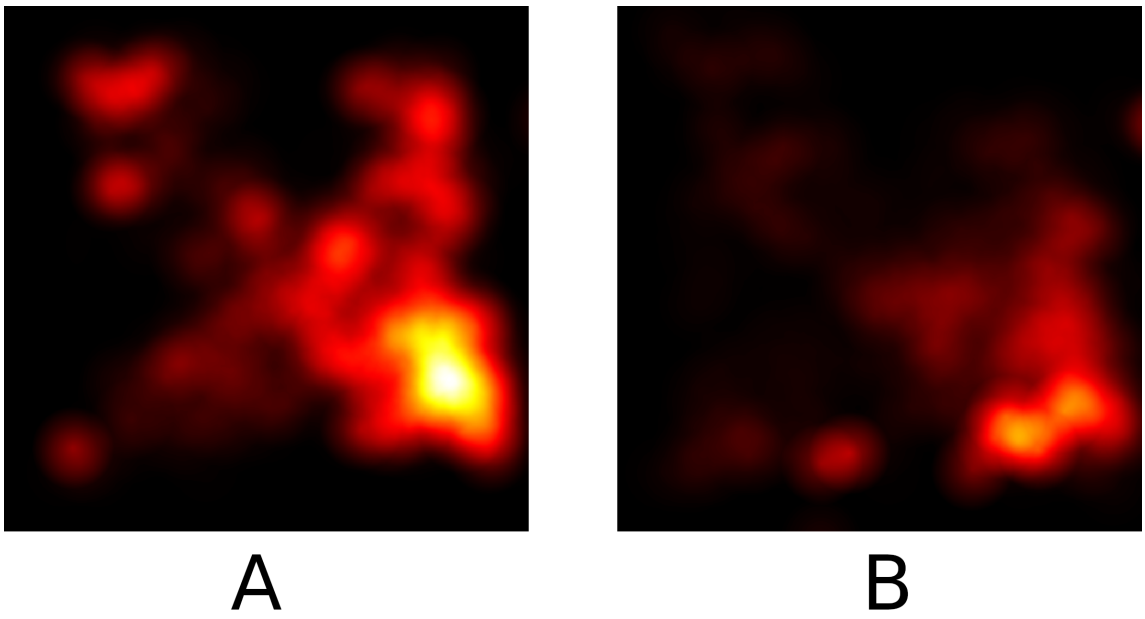



**Abbildung 6.23:** Attention Maps der dritten Partie, erstellt über die Gesamtdauer der Partie. Dunkel/Schwarz: Wenig bzw. keine Aufmerksamkeit. Hell/Gelb: Viel bzw. hohe Aufmerksamkeit. **A**: Proband 2, **B**: Proband 1

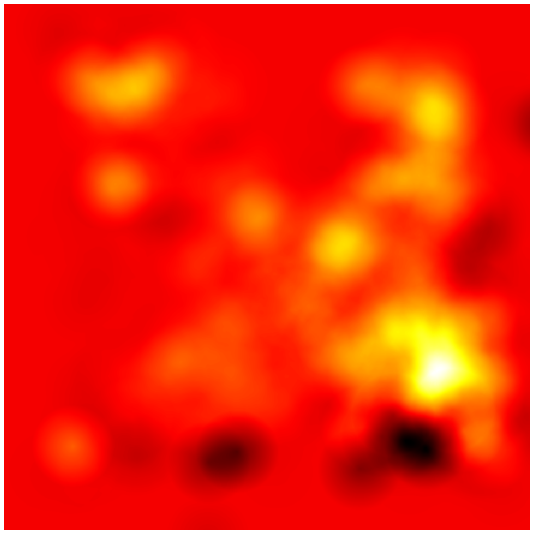


**Abbildung 6.24:** Differenz der Attention Maps der dritten Partie, erstellt über die Gesamtdauer der Partie. Rot (siehe Ecken und Randbereiche): Gleich viel Aufmerksamkeit. Orange/Gelb: Mehr Aufmerksamkeit bei Proband 2. Dunkelrot/Schwarz: Mehr Aufmerksamkeit bei Proband 1. Map von Proband 1 subtrahiert von Map von Proband 2:
Abbildung 6.23.A − Abbildung 6.23.B

#### Spielverlauf, Beginn der Partie

Die dritte Partie wurde auf einem Spielbrett der Größe 19x19 bestritten und dauerte daher länger an. Auch waren die Spielsteine in der Konsequenz kleiner als bei Partie 1 und 2, wodurch die Blicke der Spieler in umkämpften Szenarien im Schnitt etwas näher beieinander liegen dürften als in den anderen beiden Partien. Der MinDist-Plot (Abbildung 6.22) spiegelt dies wieder, da wesentlich häufiger und länger türkise Markierungen auftreten als in den MinDist-Plots der ersten (Abbildung 6.1) und der zweiten (Abbildung 6.12) Partie. Ebenfalls auffällig sind die zuhauf vorkommenden, teilweise etwas längeren dunkelgrauen Bereiche. Diese sind die Folge verstärkter Kopfbewegungen der Probanden und des dadurch resultierenden häufigeren Gaze-Verlustes der Eye-Tracker. Ursache hierfür ist eventuell die Tatsache, dass es sich bei der Partie um eine der letzten aufgezeichneten Partien handelt und die Probanden bereits unmittelbar zuvor ein etwas längeres und einige kürzere Spiele gegeneinander ausgetragen hatten. Dennoch ist diese Partie interessant für die Analyse, da sich ein deutlicher Schwerpunkt der Aufmerksamkeit beider Spieler im unteren rechten Quadranten des Spielfeldes abzeichnet, wie sowohl den Attention Maps (siehe Abbildung 6.23), als auch bis zu einem gewissen Grad dem MinDist-Plot (Abbildung 6.22) zu entnehmen ist.

Zunächst beginnen beide Probanden damit, Präsenz in jeweils zwei Quadranten des Feldes zu aufzubauen. Im dritten Zug greift Proband 1 (Schwarz/Rot) Proband 2 im unteren linken Quadranten an. Die Aufmerksamkeit von Proband 2 lag unmittelbar zuvor im oberen Bereich des Spielfeldes, scheinbar war zunächst ein defensiveres Spiel mit Gebietserschließungen beabsichtigt (siehe Abbildung 6.25).

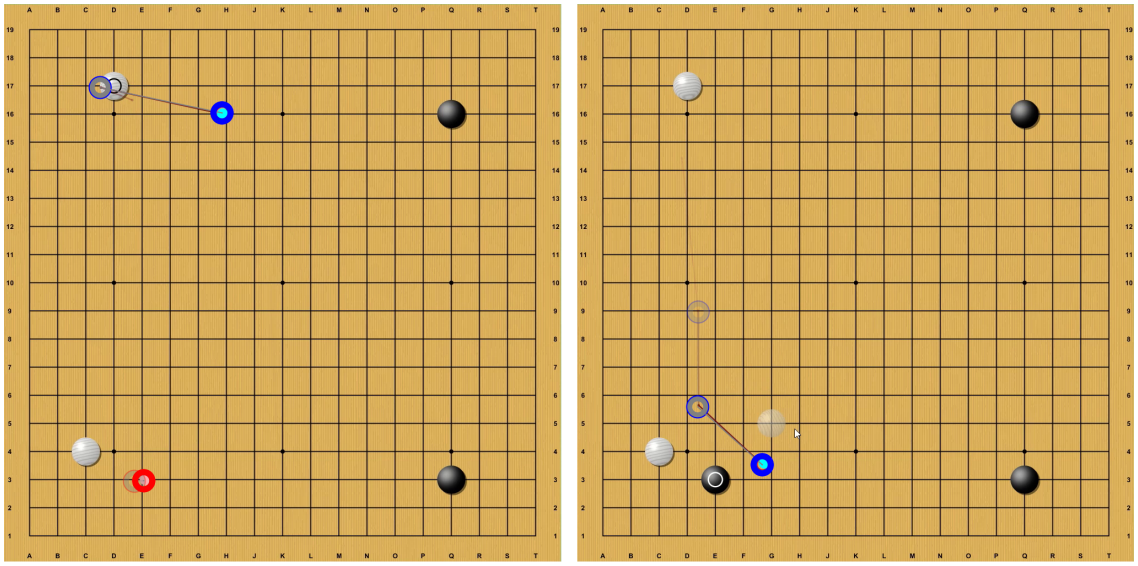



**Abbildung 6.25:** Dritter Zug der dritten Partie. **A**: Bevor Proband 1 (Schwarz/Rot) den Stein setzt. Weiß scheint zu beabsichtigen, weiteres Gebiet in der oberen linken Ecke zu sichern und damit defensiv zu spielen. **B**: Schwarz greift weiß in der unteren linken Ecke an. Weiß reagiert darauf und platziert den Stein ebenfalls im unteren linken Quadranten, um die Vorherrschaft zu sichern.

#### Spielverlauf, Abschnitt B

Nach Stärkung der Präsenz durch Weiß legt Schwarz im vierten Zug nach, doch weiß dreht den Spieß um und spielt offensiv in den unteren rechten Quadranten. Schwarz wird dadurch in die Defensive gedrängt (siehe Abbildung 6.26). Dies löst ein Geplänkel aus, bei dem beide Spieler einige Züge lang abwechselnd Steine in den unteren rechten Quadranten setzen. Auf dem MinDist-Plot (Abbildung 6.22, Abschnitt B) liegen die Blicke der Kontrahenten in diesem Abschnitt häufiger nah beieinander. Das Attention-Map-Overlay für diesen Abschnitt (siehe Abbildung 6.27) zeigt, dass Proband 2 (Weiß/Blau) etwas mehr auf die Position der eigenen Steine schaut, wohingegen Proband 1 etwas mehr auf die Räume zwischen den eigenen Steinen und denen des Gegners zu achten scheint. Letztendlich beendet Proband 1 den Kampf durch Setzen eines Steines in den oberen linken Quadranten. Die Fixationen von Proband 2 liegen bis zu diesem Zeitpunkt im umkämpften Gebiet, es ist also wahrscheinlich, dass Proband 2 das Momentum genutzt und weiter in diesen Bereich gespielt hätte.

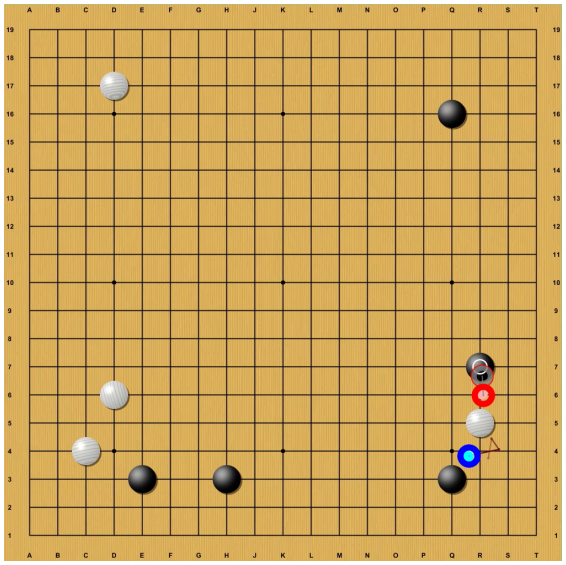


**Abbildung 6.26:** Fünfter Zug der dritten Partie, Weiß ist am Zug. Im vierten Zug hatte Schwarz im unteren linken Quadranten nachgelegt, doch nun geht weiß in die Offensive über. Schwarz fühlt sich bedroht und stärkt im fünften Zug die Präsenz unten rechts.

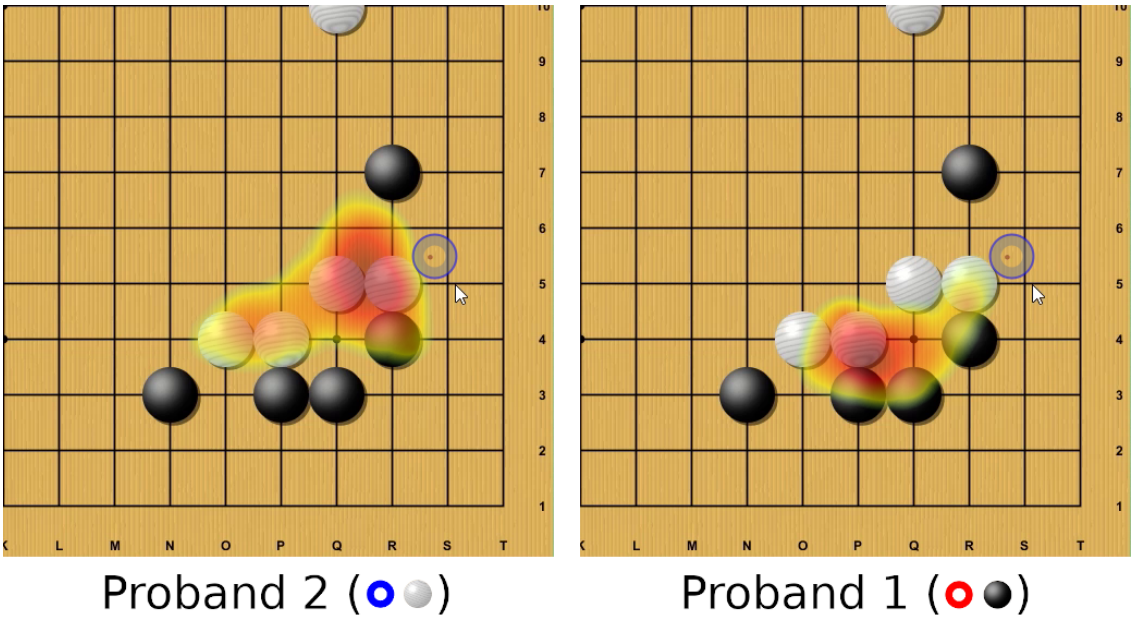



**Abbildung 6.27:** Geplänkel im unteren rechten Quadranten in der dritten Partie. Die erste größere Auseinandersetzung des Spiels. Weiß agiert offensiv und versucht, die Präsenz von Schwarz etwas einzudämmen und in die Ecke zu drängen. Proband 2 achtet etwas mehr auf die eigene Position, während Proband 1 etwas mehr auf den Raum dazwischen blickt.

### Spielverlauf, Abschnitt C

Im neunten Zug, direkt nach dem Befreiungsschlag von Schwarz, startet Weiß ohne zu zögern einen weiteren Angriff gegen Schwarz im oberen rechten Quadranten. Ein zweiter, etwas längerer Kampf entfacht. Proband 2 (Weiß/Blau) setzt dabei zunächst einige Steine ein paar Felder auseinander, während Proband 1 immer wieder versucht, umgehend die Räume dazwischen zu besetzen und somit die Steine des Gegners zu isolieren und einfacher fangbar zu machen (siehe Abbildung 6.28 A). Weiß geht dazu über, Verbindungen zwischen den gesetzten Ecksteinen zu ziehen und somit die Front zu verdichten. Die Kompaktheit der schwarzen Gruppe ergibt einen leichten Vorteil in dieser Anordnung für Proband 1, weshalb Schwarz sich nicht unmittelbar in ein direktes Kopf-an-Kopf-Rennen mit Kettenbildung einlässt, sondern zunächst einen schwarzen Stein etwas weiter unten auf dem Spielfeld platziert, um den leichten Vorteil zu erhalten, sollte sich der Kampf in die angedeutete Richtung weiterentwickeln (siehe Abbildung 6.28 B und C). Proband 2 scheint diesen Zug zu erwarten, direkt vor dem Erscheinen des schwarzen Steines ruht der Blick des Probanden etwa zwei Sekunden genau an dieser Stelle. Im nach Häufung aggregierten Attention-Map-Overlay wird sichtbar, dass Proband 1 eine Entwicklung des Kampfes in die südliche Spielfeldhälfte erwartet (siehe Abbildung 6.28 C).

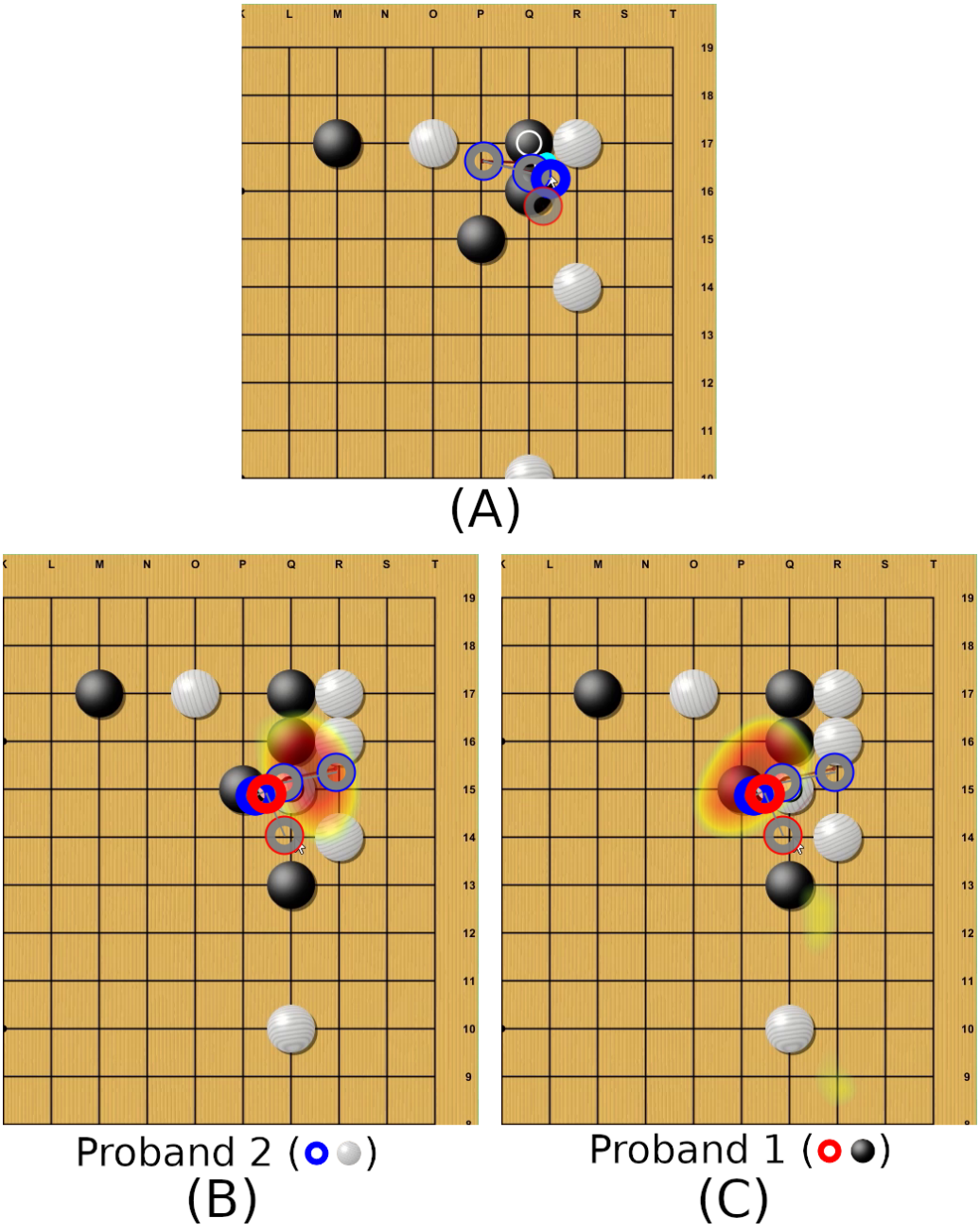



**Abbildung 6.28:** Geplänkel im oberen rechten Quadranten in der dritten Partie. Der Beginn eines langen Kampfes auf der rechten Spielfeldhälfte. **A**: Erneut eingeleitet durch Weiß versucht Schwarz, die Räume zwischen den gegnerischen Steinen zu erschließen und so die sporadisch verteilten weißen Steine zu isolieren. **B/C**: Schwarz hat einen leichten Vorteil durch Kompaktheit und versucht, diesen durch taktisches Vorgreifen zu erhalten. Weiß schließt indes die Lücken in den eigenen Reihen und lässt sich auf die Gegenwehr ein. **B**: Aufmerksamkeit von Proband 2 (Weiß/Blau) liegt auf den Lücken in den eigenen Reihen. **C**: Proband 1 scheint eine Entwicklung Richtung untere Spielfeldhälfte zu erwarten. Da der Abschnitt recht kurz ist, wird dieser Umstand durch Aggregation nach Häufung im Attention-Map-Overlay sichtbar.

Dies tritt dann auch ein, während beide Spieler zunächst die Lücken in den eigenen Reihen verdichten. Die Initiative gen Süden und dann Westen geht von Proband 2 aus, während Proband 1 etwas reaktiver spielt und bemüht ist, die Kontrolle über die Gruppe des Gegners zu behalten und ein Vordringen in die Spielfeldmitte zu verhindern (siehe Abbildung 6.29).

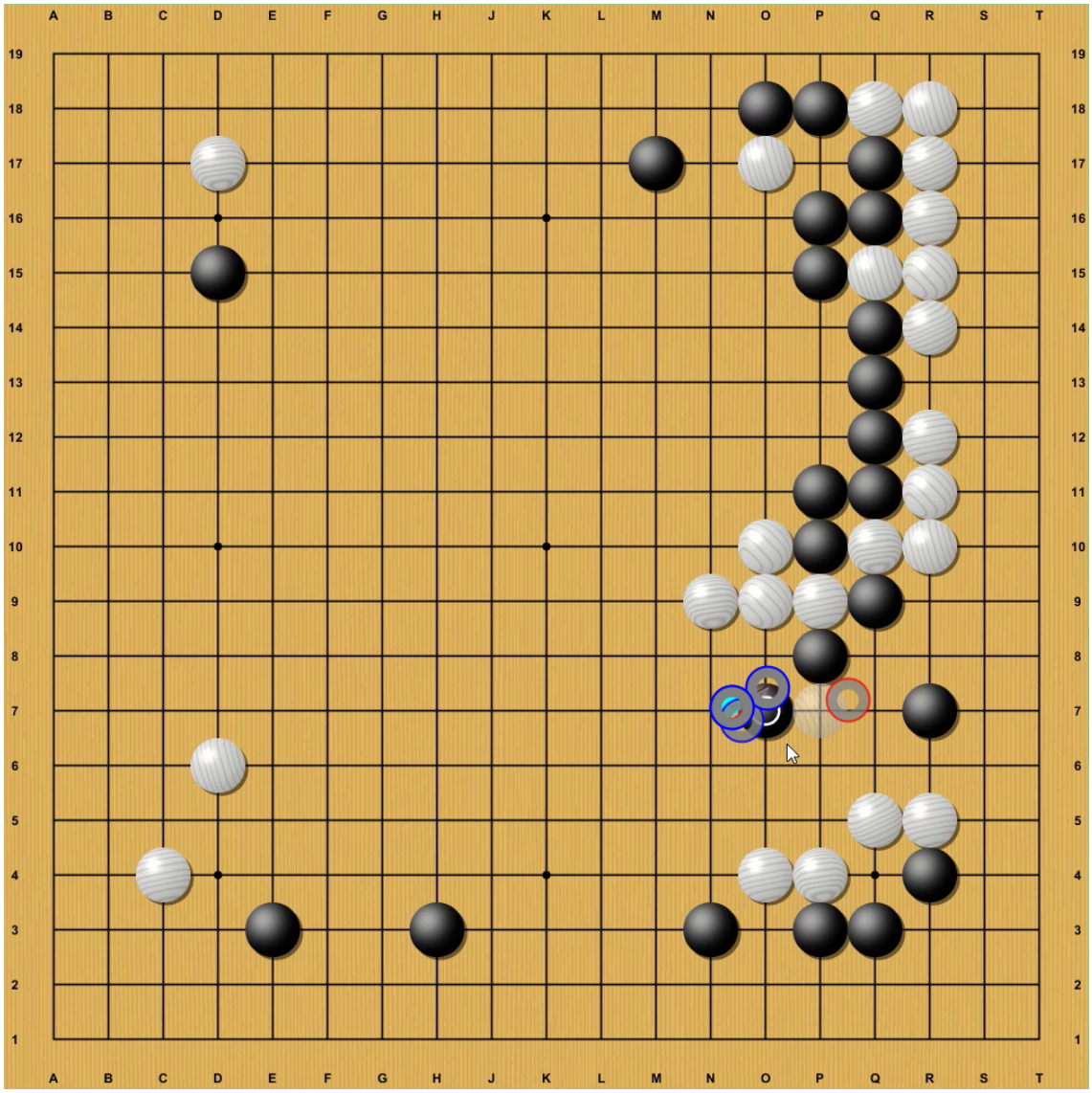



**Abbildung 6.29:** Kampf um die rechte Spielfeldhälfte in der dritten Partie. Wie erwartet wandert das Gefecht in die untere Spielfeldhälfte, während beide Spieler die Lücken in den eigenen Reihen schließen. Proband 2 (Weiß/Blau) versucht, ins Innere des Spielfeldes vorzudringen und im späteren Spielverlauf eventuell die Front mit den Steinen im Süden zu schließen.

Dies gelingt jedoch nicht (siehe Abbildung 6.30) und Proband 2 erringt eine vorteilhafte Konstellation im südöstlichen Bereich des Spielfeldes. Das Attention-Map-Overlay macht die Absicht deutlich, ins Spielfeldinnere vorzudringen (siehe Abbildung 6.30 links), während Proband 1 scheinbar der

gelungene Ausbruch beschäftigt (Abbildung 6.30 rechts). Dieser Umstand scheint eine Taktikänderung bei Proband 1 auszulösen. Im Nachfolgenden Abschnitt versucht dieser stattdessen, die weiße Gruppe im Südosten des Spielfeldes komplett zu isolieren und einzukreisen.

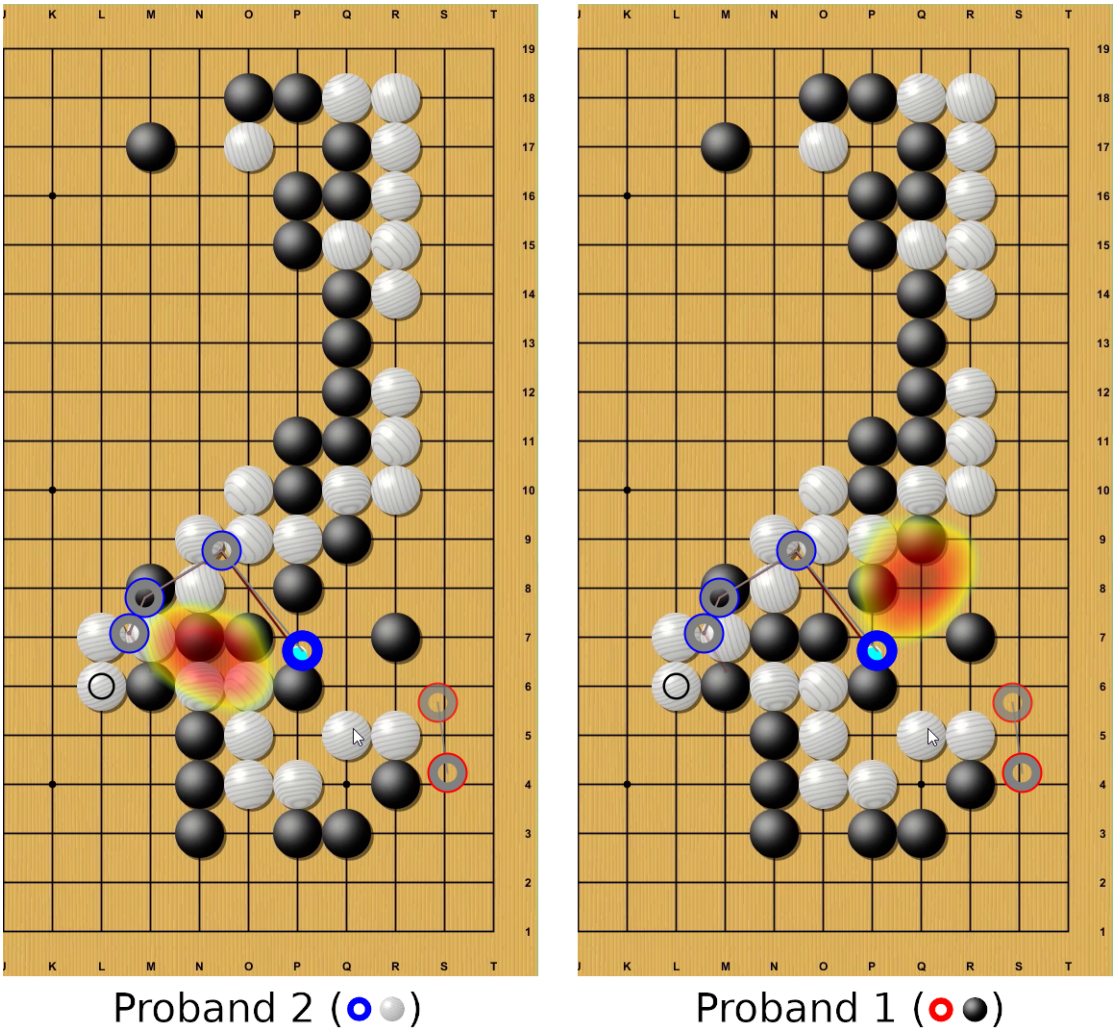



**Abbildung 6.30:** Fortgeschrittener Kampf um die rechte Spielfeldhälfte in der dritten Partie. Proband 2 gelingt der Ausbruch Richtung Spielfeldmitte. Das Attention-Map-Overlay lässt klar die Absichten von Proband 2 erkennen. Die Aufmerksamkeit von Proband 1 indes zeugt von der Absicht, die weißen Steine weiter an den Rand zu drängen. Die Stelle des Ausbruchs der gegnerischen Steine erfuhr die größte Aufmerksamkeit.

Einige Züge später ergibt sich eine Ko-Situation, als Proband 1 einen einzelnen weißen Stein umkreist und damit fängt (siehe Abbildung 6.31). Proband 2 (Weiß/Blau) schien eine Verlagerung des Gefechts vom Norden des Quadranten in den Süden zu erwarten, wenige Sekunden zuvor wanderte der Blick nach unten über die Gruppe, deren Einkreisung scheinbar bevorsteht.

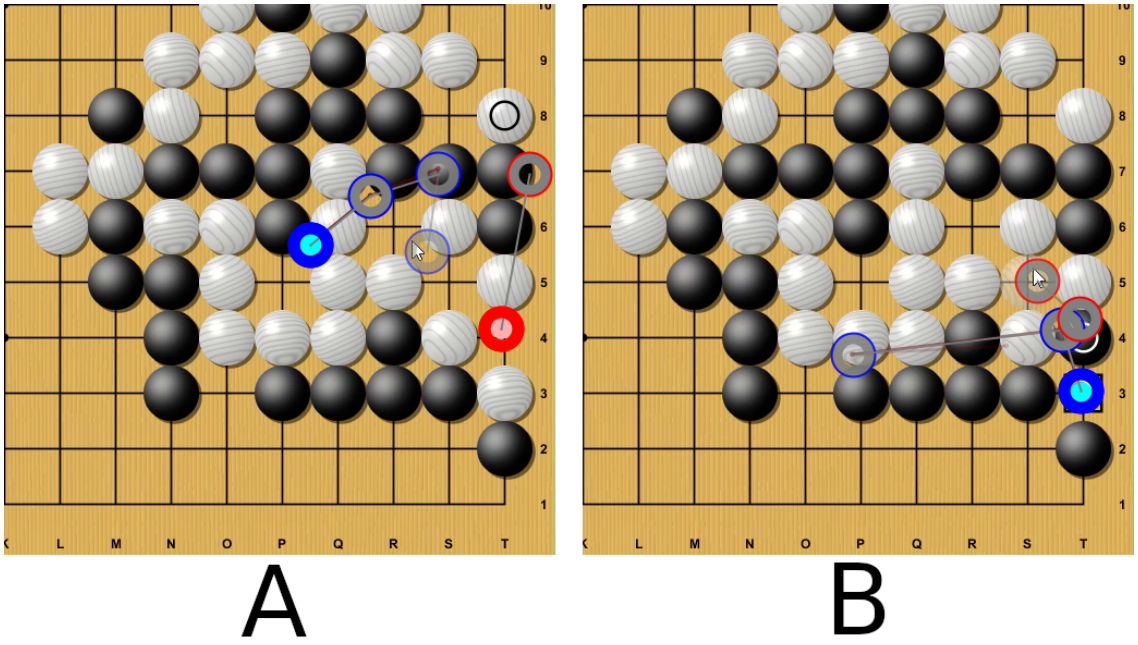



**Abbildung 6.31:** Kampf um den Südosten des Spielfeldes in der dritten Partie. **A**: Proband 1 (Schwarz/Rot) erkennt die Gelegenheit, einen gegnerischen Stein zu fangen. **B**: Eine Ko-Situation entsteht. Proband 2 (Weiß/Blau) könnte den alten Zustand wiederherstellen, die Spielregeln lassen dies jedoch nicht zu.

Stattdessen fährt Proband 2 damit fort, eine vergleichbare doppelte Gefangennahme zwei Felder weiter oben zu verhindern, während Proband 1 die Lücke in der Kette im Süden schließt. Im nächsten Zug dann erwidert Proband 2 tatsächlich das Ko im Süden und fängt damit einen schwarzen Stein. Proband 1 schließt daraufhin den freien Stein nahe Spielfeldecke an die Kette an, und während Proband 2 die Gruppe im Nordosten des Quadranten verdichtet, damit durch Erhöhung des Drucks auf den oberen schwarzen Halbkreis der beinahe umstellten südlichen weißen Gruppe den Rücken stärkt und anschließend weiter in die Spielfeldmitte vordringt, erobert Proband 1 erneut das Ko zurück und beendet die Szene (siehe Abbildung 6.32). Da sich eine Möglichkeit zum Seki und damit zum Ende des Gefechts im unteren rechten Quadranten zwischen den zwei überlappenden Kreiskonstellationen anzubahnen scheint, konzentrieren die Kontrahenten ihre Bemühungen lieber auf andere Bereiche des Spielfeldes, nachdem Proband 2 die Möglichkeit ergreift und eine Pattsituation schafft (siehe Abbildung 6.32).

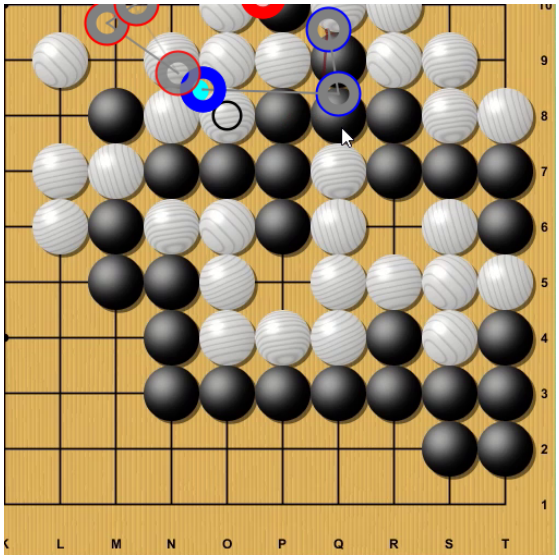



**Abbildung 6.32:** Ende der aufeinanderfolgenden Ko-Szenarien in der dritten Partie. Proband 1 besetzt alle Felder und beendet damit die Szene. Proband 2 erzwingt indes das Seki, da zwischen den zwei inneren Gruppen nun nur noch zwei geteilte Freiheiten bestehen.

#### Spielverlauf, Abschnitt D

Nach sehr kurzem Hin und Her gibt Proband 2 wieder den Ton vor und erwirkt eine Verlagerung der Partie in den oberen linken Quadranten. Wie im MinDist-Plot (Abbildung 6.22) zu erkennen ist, lässt die Qualität der aufgezeichneten Tracker-Daten in diesem Abschnitt stark nach. Es fehlen hauptsächlich die Gaze-Daten von Proband 2, eventuell ist dies einem Konzentrationsverlust und damit einhergehendem erhöhtem Bewegungsdrang des Probanden zu verschulden, da der Tracker scheinbar über einen etwas längeren Zeitraum Schwierigkeiten hatte, die Augen zu erfassen. Auch im weiteren Verlauf der Partie nimmt die Qualität des Recordings nur noch einmal kurz marginal zu. Beide Spieler bilden in diesem Abschnitt einige kleine Gruppen, jedoch entsteht keine interessante Spielszene. Lediglich ein einziger Zug von Proband 2 macht noch auf sich aufmerksam. Der Proband platziert im Verlauf des Geplänkels einen schwarzen Stein etwas weiter im Süden nahe der vertikalen Mitte des Spielfeldes, einen vergleichbaren Zug hatte es zuvor auf der rechten Spielfeldhälfte gegeben. Ein paar Züge später weicht Proband 1 aus und konzentriert die Bemühungen erneut auf die südliche Spielfeldmitte (Abbildung 6.33).

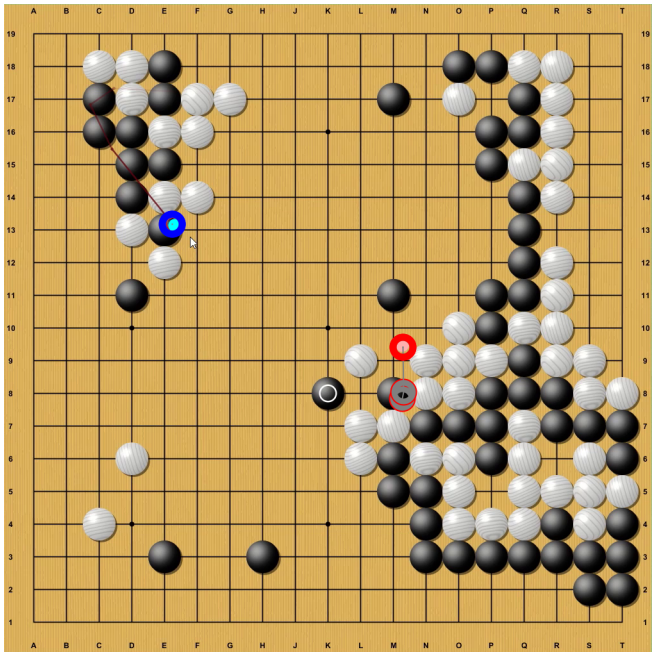


**Abbildung 6.33:** Gefecht um den nordwestlichen Quadranten in der dritten Partie. Die Konfrontation wird kurzzeitig durch Proband 1 unterbrochen.

Proband 2 folgt dem Beispiel und sichert die Überlegenheit in dem Gebiet, jedoch handelt es sich hierbei nur um einen kurzen Ausreißer und der Fokus verweilt weiter im oberen linken Quadranten. Im weiteren Verlauf des Gefechts im Nordwesten wird es nur einmal kurz spannend. Das Attention-Map-Overlay verrät, dass die Aufmerksamkeit von Proband 2 während dieses Abschnittes eher auf Gebieten nahe des Spielfeldrandes lastete, während sich Proband 1 scheinbar noch Hoffnungen um die Spielfeldmitte zu machen schien (siehe Abbildung 6.34). Der kleinere südliche Hotspot in Abbildung 6.34 A und die blaue Fixation zeugen von der von Proband 2 erkannten und ergriffenen Gelegenheit, einen einzelnen schwarzen Stein zu fangen, ermöglicht durch den Fehler von Proband 1, auf der linken Seite statt drei Felder weiter rechts zu setzen. Es steht zwei zu zwei.

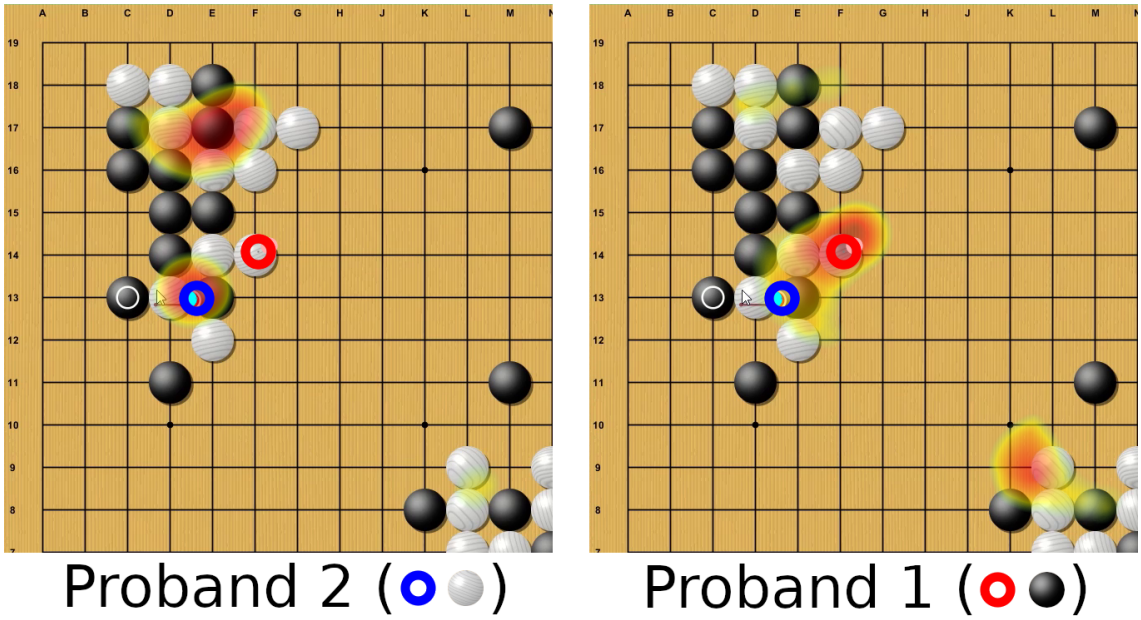



**Abbildung 6.34:** Aufmerksamkeit der Spieler während des Geplänkels im Nordwesten in der dritten Partie. Proband 2 achtet in diesem Abschnitt mehr auf das Gefecht und konzentriert sich auf Randgebiete. Proband 1 scheint währenddessen noch Chancen nahe der Spielfeldmitte zu wittern.

#### Spielverlauf, Abschnitt E

Zwischen Abschnitt D und E spielen beide Spieler drei Züge lang in den Quadranten unten links, jedoch scheint keiner der beiden an diesem Ort nach einer längeren Auseinandersetzung zu suchen. Proband 2 löst schließlich eine vertikale Erschließung der Spielfeldmitte zwischen den zwei Spielern aus (siehe Abbildung 6.35).

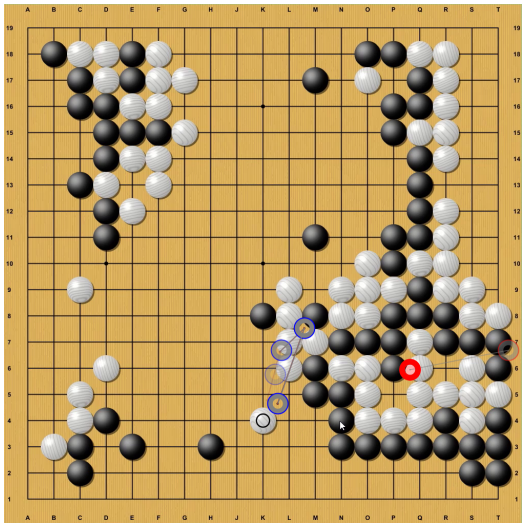


**Abbildung 6.35:** Proband 2 löst die parallele Eroberung der Spielfeldmitte in der dritten Partie aus.

In diesem Abschnitt setzen die Probanden ihre Steine meist ohne große Verzögerung und arbeiten sich langsam vom Süden aus durch die Spielfeldmitte Richtung Norden vor. Wie in den Attention Maps über den Abschnitt (siehe Abbildung 6.36) erkennbar ist, verliert Proband 1 einige Male den Fokus auf das Spiel, während Proband 2 weiter konzentriert vorgeht. Dieser Umstand wird auch im MinDist-Plot (Abbildung 6.22) sichtbar.

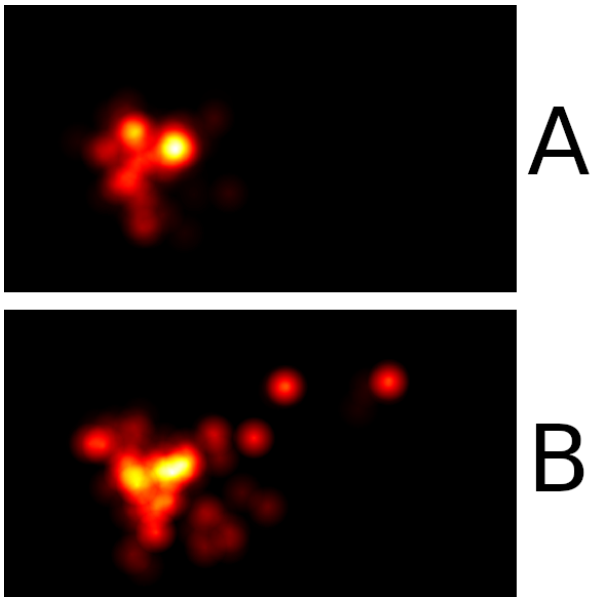



**Abbildung 6.36:** Attention Maps der parallelen Eroberung der Spielfeldmitte in der dritten Partie über die Grenzen des Spielfeldes hinaus. **A**: Proband 2 ist ausschließlich auf die Spielfeldmitte fokussiert. **B**: Die Konzentration von Proband 1 scheint etwas nachzulassen und die Aufmerksamkeit liegt teilweise außerhalb des Spielfeldes. Da für Proband 1 in diesem Abschnitt häufiger Daten fehlen, wurden die Attention Maps getrennt normalisiert, damit die Informationen in B gut erkennbar sind. Dadurch sind die Maps jedoch nicht mehr direkt auf relative Aufmerksamkeitsdauer vergleichbar.

Im ersten Viertel des Abschnittes versucht Proband 2 offenbar, die schwarzen Steine möglichst von der Spielfeldmitte fernzuhalten und zwischen den eigenen Reihen einzuengen (siehe Abbildung 6.37). Dabei wird außerdem ein weiterer schwarzer Stein gefangen und ein Auge entsteht. Da Proband 1 im Osten noch eine recht lange Gruppe besitzt, ist Proband 2 gleichzeitig bemüht, eine Einkreisung der sehr großen weißen Gruppe südöstlich der Spielfeldmitte zu verhindern. Dies gelingt geschickt durch Erzeugung eines weiteren Auges und dadurch eines Tsumego (siehe Abbildung 6.38).

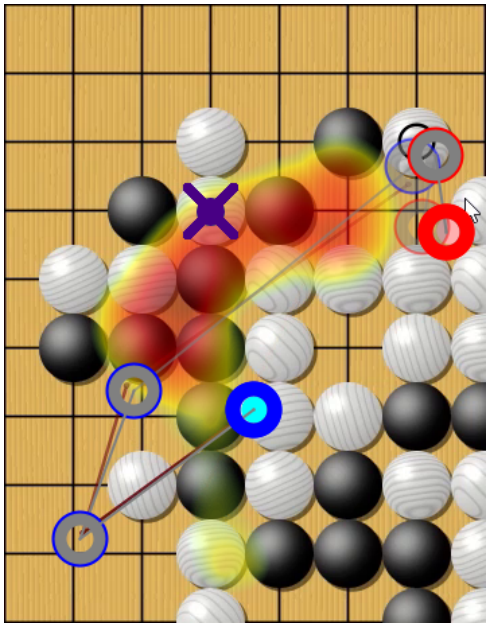


**Abbildung 6.37:** Attention-Map-Overlay zeigt den Plan von Proband 2, die schwarzen Steine von der Mitte des Spielfeldes abzudrängen. X in Lila markiert die Spielfeldmitte.

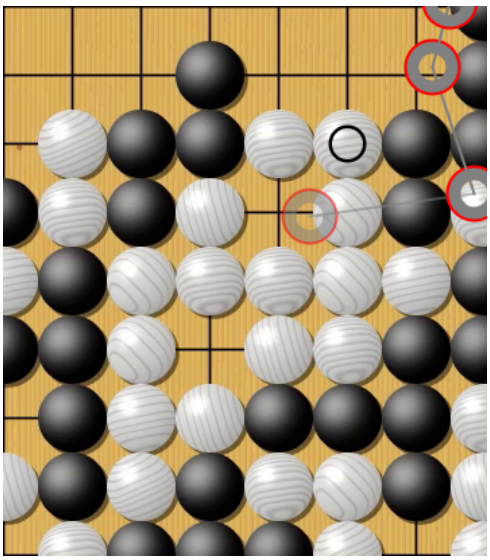


**Abbildung 6.38:** Proband 2 vollendet das zweite Auge eines Tsumego, um die bedrohte weiße Gruppe zu schützen. Der Anreiz für Schwarz, Richtung Osten zu expandieren, geht verloren.

Anschließend arbeiten sich die Spieler nach oben links vor. Weiß gelingt es im letzten Viertel des Abschnittes, eine Gruppe schwarzer Steine in der Mitte zu isolieren (siehe Abbildung 6.39), während Proband 1 damit beschäftigt ist, durch offensives Spiel eine Verbindung zwischen zwei weißen Gruppen und ein paar vereinzelten Steinen zu unterbinden. Die Qualität der Daten von Proband 1 ist in dieser Szene wieder etwas durchwachsen, weshalb leider nicht daraus hervorgeht, ob dem

Spieler bewusst ist, dass eine Gruppe der eigenen Steine unmittelbar vor der Gefangennahme steht, jedoch vergrößert er bereitwillig besagte Gruppe um zwei weitere Steine, während Proband 2 die Einkreisung sichert. Als Schwarz daraufhin die Aussichtslosigkeit der Lage erkennt, entscheidet der Proband sich dazu, stattdessen um Gebiete im Süden zu kämpfen. Weiß folgt unmittelbar.

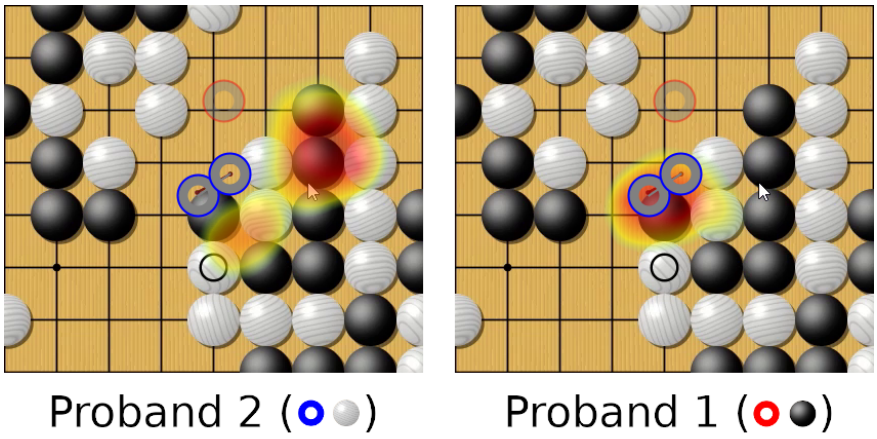



**Abbildung 6.39:** Proband 2 schafft es hier, eine Gruppe der Größe fünf entscheidend zu isolieren, sodass eine Gefangennahme sehr wahrscheinlich wird. Das Attention-Map-Overlay von Proband 2 zeigt, dass der Proband das Ziel klar im Blick hat. Proband 1 scheint währenddessen eine Verbindung von Weiß zwischen Spielfeldmitte und oberem linken Quadranten unterbinden zu wollen.

#### Spielverlauf, Abschnitt F

In Abschnitt F wurden kaum Gaze-Daten von Proband 1 durch den Tracker aufgezeichnet, und die Daten von Proband 2 sind ebenfalls etwas lückenhaft. Dennoch macht Weiß im letzten lokalen Gefecht drei weitere Punkte durch eine erfolgreiche weitere Isolation (siehe Abbildung 6.40). Schwarz bleibt nichts anderes übrig als die Gruppe aufzugeben und setzt den nächsten Stein etwas abwärts. Proband 2 fängt die drei Steine und einen einzelnen weiteren, während Proband 1 die Kette verlängert, die nun keine Bedrohung mehr für Weiß darstellt. Spätestens an dieser Stelle dürfte den Kontrahenten der bevorstehende Ausgang der Partie bekannt sein, jedoch folgt eine letzte Phase mit einigen Bemühungen um die Lücken im oberen rechten Quadranten. Es steht inzwischen sieben zu zwei für Weiß und der strategische Vorteil liegt ebenfalls bei Proband 2.

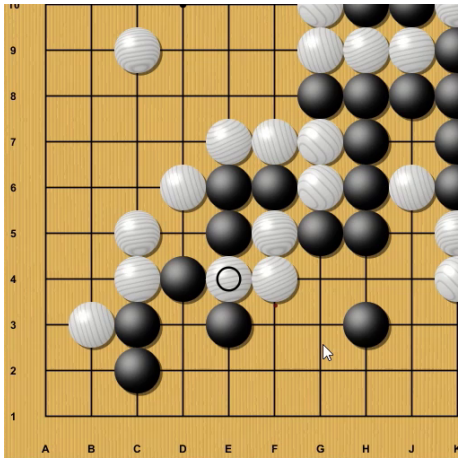



**Abbildung 6.40:** Proband 2 (Weiß/Blau) schafft es erneut, eine Gruppe der Größe drei zu isolieren.

#### Spielverlauf, Abschnitt G

Im letzten Abschnitt setzen beide Spieler vereinzelt Steine in den oberen rechten Quadranten. Proband 1 startet einen letzten Versuch, die dort verweilenden Gruppen mit der Spielfeldmitte zu verbinden, doch gestattet dadurch Weiß nur das Fangen eines weiteren Steines. Das Spiel ist entschieden.

#### Ende der Partie

Aufgrund der deutlichen Punkteführung von Proband 2 dank gefangener Steine und dem Mangel an Möglichkeiten, nachzuziehen oder die Punktedifferenz durch Gebietseroberungen noch auszugleichen, kapituliert Proband 1. Somit ist Proband 2 der Sieger der dritten Partie.

### 6.1.4 Partieübergreifende Merkmale

Über die drei analysierten Partien hinweg durfte Proband 1 immer den ersten Zug machen. In zwei von drei Partien, nämlich genau denen, bei denen auf einem kleinen Spielfeld gespielt wurde und der durch das Recht des ersten Zuges erlangte Vorteil am größten ist, gewann Proband 1. Die Aufmerksamkeit von Proband 2 lag gemäß Attention Maps in allen drei Partien konzentrierter und länger auf bestimmten Punkten, während die Aufmerksamkeit von Proband 1 generell etwas gleichmäßiger verteilt über größere Bereiche des Spielfeldes wanderte. Darüber hinaus scheint Proband 1 häufig auf die eigenen Steine und die Räume dazwischen zu schauen, während Proband 2 öfter die Steine des Gegners im Blick hat. Außerdem fällt auf, dass die Aufmerksamkeit von Proband 2 generell in allen drei Partien am längsten im unteren rechten Quadranten des Spielfeldes weilte. Zum Spielstil der Probanden bleibt zu sagen, dass Proband 1 im Allgemeinen etwas vorsichtiger und defensiver in eine Partie geht, während Proband 2 lieber mit mehr Risiko spielt und meist die frühe Initiative zum Angriff ergreift.

## 6.2 Auswertung

Es folgt eine Auswertung der Programmfunktionalität anhand der Details der Implementierung (siehe Kapitel 5) und der Ergebnisse der Analyse im Hinblick auf die in Abschnitt 4.2 formulierten Anforderungen.

### 6.2.1 Erfüllung des Anwendungsfalls

Zunächst ist das Programm grundsätzlich in der Lage, die Datensätze der Eye-Tracker einzulesen und zu kombinieren. Die exportierten TSV-Dateien der zwei verschiedenen Geräte sind importierbar (siehe Unterabschnitt 5.5.2). Auch die Screen Recordings können geladen und abgespielt werden, und verkörpern den wesentlichen Bestandteil der Hauptansicht (siehe Abschnitt 5.3). Die Gaze-Daten jedes eingebundenen Datensatzes werden nach dem Import über dem Video dargestellt (Abschnitt 5.4, Abschnitt 5.8). Der Vorgang zum Einbinden weiterer Datensätze in Abschnitt 5.8 ist beliebig oft wiederholbar. Ein Projekt kann auf Grundlage der Screen Recordings beider Tracker erstellt werden, ein bidirektionales Mapping ist dank Behandlung gerätespezifischer Merkmale im Parsing-Vorgang (siehe Abschnitt 5.5) möglich. Beim Durchlaufen der Daten-Pipeline (siehe Unterabschnitt 4.3.2) jedes Datensatzes werden mehrere Datensätze einer Partie durch einen Filtervorgang (siehe Abschnitt 5.6), eine Koordinatentransformation auf das geladene Screen Recording (siehe Unterabschnitt 5.8.1) und eine zeitliche Synchronisation (siehe Unterabschnitt 5.8.2) vergleichbar gemacht. Die verschiedenen Visualisierungsmodi (siehe Abschnitt 5.7) ermöglichen eine Analyse der Daten. Darunter finden sich neben dem bereits angedeuteten Gaze Plot (siehe Unterabschnitt 5.7.1) auch die Attention Maps (siehe Unterabschnitt 5.7.2), die Auskunft darüber geben können, wie lange zwei Spieler/innen im Verhältnis zueinander an welche Stelle auf dem Spielfeld geschaut haben. In der Analyse kam dieses Werkzeug häufig direkt (Bsp.: Abbildung 6.13, Abbildung 6.21, Abbildung 6.36), aber auch zur Visualisierung der Differenz (Bsp: Abbildung 6.24) und als Overlay über dem Screen Recording (Bsp: Abbildung 6.27) zum Einsatz. Die Anwendungsmöglichkeiten für diesen Modus sind, wie sich im praktischen Einsatz zeigt, sehr flexibel. Auch die Abstände der Gaze-Daten beider Spieler werden im MinDist-Plot (siehe Unterabschnitt 5.7.3) verarbeitet, aus dem hervorgeht, zu welchen Zeitpunkten der Abstand unter oder über einer definierbaren Schranke liegt, wann die Blicke der Spieler außerhalb des Spielfeldes liegen oder gar komplett fehlen. Dieser Plot kann auch dazu genutzt werden, einen groben Eindruck über die Qualität eines Eye-Tracker-Recordings zu gewinnen (siehe Unterabschnitt 6.1.2, Unterabschnitt 6.1.3 Spielverlauf, Abschnitt F).

# 7 Zusammenfassung und Ausblick

Es folgt eine kurze Zusammenfassung, die die Inhalte dieser Arbeit knapp reflektiert. Abschließend werden in einem Ausblick Möglichkeiten zur Fortführung der dargelegten Ansätze, Konzepte, Ideen und Lösungen vorgeschlagen.

## Zusammenfassung

Diese Arbeit hat das Ziel verfolgt und umgesetzt, ein Konzept für die gekoppelte Analyse der Eye-Tracking-Daten zweier Geräte zu entwerfen und zu implementieren. Gerätespezifische Probleme wurden in maßgeschneiderten Parsermodulen behandelt, die die Datensätze in eine gemeinsame Form bringen. Neue Parsermodule für weitere Eye-Tracking-Geräte können mit überschaubarem Aufwand zum Programm hinzugefügt werden. Importierte Daten durchlaufen mehrere Schritte, die dazu dienen, den abgelegenen geteilten Kontext der gleichen Partie aus zwei Perspektiven in einen lokalen gemeinsamen Kontext auf Basis einer einzigen Bildschirmaufzeichnung zu überführen. Dazu gehört ein Filtervorgang bei dem ein I-VT-Filter die Rohdaten zu Fixationen und Sakkaden aggregiert, ein Mapping bzw. eine Koordinatentransformation auf ein anderes GUI-Layout bei unterschiedlichen Bildseitenverhältnissen sowie die Ermittlung der zeitlichen Verschiebung zwischen den Zeitstempeln der Datensätze. Mit OpenCV als Grundlage wurde ein einfacher Video-Player gebaut der in der Lage ist, die Bildschirmaufzeichnungen zu den Datensätzen abzuspielen. Da die Frames der Videos einzeln nacheinander in einen Puffer geladen und dann manuell gezeichnet werden, sind Modifikationen am Bildmaterial während des Zeichenvorgangs sehr einfach umzusetzen. Dieser Umstand wird genutzt, um verschiedene Visualisierungsmodi als Overlay über dem Video darzustellen. Dazu zählt ein Gaze Plot, der den Scanpath, also den Verlauf der Augenbewegungen jedes/r dargestellten Spielers/Spielerin visualisiert. Eine begrenzte Visualisierung der Rohdaten neben den Fixationen ist ebenfalls möglich. Das dritte Overlay basiert auf Attention Maps, die über variable, vom Nutzer auswählbare Abschnitte der dargestellten Partie erzeugt werden. Es zeigt für jede/n Spieler/in einzeln die Bereiche auf dem Spielfeld an, die im gewählten Abschnitt die meiste Aufmerksamkeit erfahren haben. Die generierten Attention Maps können außerdem unabhängig vom Video betrachtet und miteinander verglichen werden. Hierfür steht ein Werkzeug bereit, das zudem Differenzen zwischen den Attention Maps zweier Spieler darstellen kann. Beim letzten Visualisierungsmodus handelt es sich um einen Scarf-Plot, der einige Informationen zur Position der Gaze-Daten der beiden Spieler farblich kodiert, allem voran den Abstand der Gaze-Punkte im Bezug auf eine Mindestabstandsschranke. Die nötigen Informationen zu den geladenen Datensätzen und ermittelten Transformationen einer Sitzung können serialisiert und in eine Datei exportiert werden, die dann zu einem späteren Zeitpunkt eingelesen und genutzt werden kann, um die Sitzung wiederherzustellen ohne die notwendigen Schritte des Setups erneut einzeln durchlaufen zu müssen. In einem abschließenden Analysekapitel wurde die Funktionstüchtigkeit des entwickelten Programmes demonstriert. Dabei wurden Unterschiede im Blickverhalten der

Spieler sichtbar und ein Zusammenhang zu den Unterschieden im Spielstil scheint wahrscheinlich. Der aggressivere Spieler fokussiert länger bestimmte Bereiche des Spielfeldes und hat häufiger die Steine des Gegners im Blick, während der passivere Spieler ein gleichmäßigeres Blickverhalten aufweist und mehr damit beschäftigt ist, auf Räume zwischen den Steinen zu achten.

## Ausblick

In diesem Abschnitt werden einige Denkanstöße zur Aufgreifung und Weiterführung der Thematik dieser Arbeit geboten.

### Vorschläge zur Implementierung

Was die implementierten Features betrifft, hat der MinDist-Plot noch etwas Luft nach oben. Man könnte zum Beispiel die Entwicklung der Abstände der Fixationen farblich auf einer Skala kodieren anstatt mit einer Schranke zu arbeiten. Eine Kodierung allgemeinerer Informationen zu den Positionen wäre auch denkbar. Die aktuelle Fassung tut dies bereits in Ansätzen dadurch, dass Blicke außerhalb des Spielfeldes gelb und fehlende Gaze-Daten grau kodiert werden, aber eine umfangreichere Kodierung beispielsweise der Quadranten, in die die Blicke fallen, könnte interessant sein. Vielleicht macht es in diesem Fall auch Sinn, die Informationen in mehrere Scarf-Plots zu unterteilen: einen, der Auskunft über die relativen Abstände zwischen den Gaze-Punkten der Spieler gibt, und einen, der die Positionen der Gaze-Punkte relativ zum Spielfeld beschreibt.

Beim Attention-Map-Vergleich bieten sich noch Gelegenheiten, mehr Modi für den direkten oder den indirekten Vergleich zu implementieren. Die aktuell enthaltene Berechnung der Differenz ist sehr rudimentär. Es gibt sicherlich noch bessere Vergleichsmethoden, die die Unterschiede zwischen den Schwerpunkten der Aufmerksamkeit zweier Spieler auf eine bessere Weise hervorheben und so bislang verborgene Umstände sichtbar machen.

### Vorschläge zur Analyse

Für eine tiefergreifendere Analyse sollten zunächst weitere Daten über eine größere Anzahl an Probanden/innen erhoben werden. Es könnte interessant sein, zu untersuchen, unter welchen Umständen zwei Kontrahenten häufig in die gleichen Bereiche schauen, und wann die Blicke häufig weiter auseinander liegen. Außerdem könnte umfangreicher untersucht werden, welcher Zusammenhang zwischen dem Spielstil und dem Hauptfokus der Aufmerksamkeit besteht. Hierfür böte sich zum Beispiel die Leitfrage an, ob verstärktes Fixieren der gegnerischen Steine tatsächlich mit einem allgemein aggressiveren und riskanteren Spielstil zusammenhängt.

Im Geiste vorangehender Arbeiten, die sich mit dem Klassifizieren von Eigenschaften des Blickverhaltens von Experten und von Laien beschäftigen, wäre auch eine Analyse hinsichtlich typischen Blickverhaltens bei Experten im Kontrast zum Blickverhalten unerfahrenerer Spieler denkbar. Dabei könnte etwa ein Fokus auf der Anzahl und der Dauer der Fixationen zwischen zwei Zügen liegen. Zu erwarten wäre, dass in Partien zwischen einem/r Experten/in und einem Laien der/die Experte/in allgemein weniger und kürzere Fixationen aufweist, da vergleichbare Szenarien bereits zum Repertoire der gesammelten Erfahrungen gehören, während der Laie durch erhöhte kognitive

Auslastung und verstärkte visuelle Informationsaufnahme länger nachdenkt und somit mehr und länger andauernde Fixationen vollzieht. Man könnte auch die Unterschiede zwischen Partien zwischen zwei Experten/innen und Partien zwischen zwei Laien beleuchten und dabei möglicherweise interessante Entdeckungen machen.

# Anhang

Im nachfolgenden Abschnitt finden sich sämtliche Anhänge.

## Algorithmen

Diese Sektion enthält alle Algorithmen, auf die im Text verwiesen wurde.

### Event-Suche

```
procedure findeLetzteEventsVorZeitpunkt(anzahl, zeitpunkt) {
    gefundeneEvents = {};
    anfangZeitstempel = alleEvents.erstes().zeitstempel;
    endeZeitstempel = alleEvents.letztes().zeitstempel;

    if (zeitpunkt < anfangZeitstempel || zeitpunkt > endeZeitstempel)
        return gefundeneEvents;

    // schaetze einen start-index basierend auf dem zeitpunkt
    anteil = zeitpunkt / endeZeitstempel;
    index = floor(anteil * alleEvents.anzahl());

    if (index >= alleEvents.anzahl())
        index = alleEvents.anzahl() - 1;

    while (true) {
        if (alleEvents.get(index).zeitstempel == zeitpunkt)
            break;

        if (alleEvents.get(index).zeitstempel < zeitpunkt) {
            index++;
            if (alleEvents.get(index).zeitstempel > zeitpunkt) {
                index--;
                break;
            }
            else if (alleEvents.get(index).zeitstempel == zeitpunkt)
                break;
        }
    }

    if (index < 0 || index >= alleEvents.anzahl())
        return gefundeneEvents;

    zaehler = 0;
    while (index > 0 && zaehler < anzahl) {
        gefundeneEvents.add(alleEvents.get(index);
```

```
37          zaehler++;
38          index--;
39      }
40
41      return gefundeneEvents;
42  }
```

**Listing 1:** Event-Suche: Finde letztes Event vor Zeitpunkt

Für den erweiterten Modus wird dieses Prinzip etwas abgeändert wie folgt:

```
1   procedure findeEventsZwischenZeitpunkten(zeitpunktAnfang, zeitpunktEnde) {
2
3   /*
4   Zeilen 2-32 wie in "findeLetzteEventsVorZeitpunkt(anzahl, zeitpunkt)"
5   jedoch werden alle Vorkommen der Variable "zeitpunkt"
6   durch "zeitpunktEnde" ersetzt
7   */
8
9   while (index > 0) {
10      if (alleEvents.get(index).zeitstempel < zeitpunktAnfang)
11          break;
12
13      gefundeneEvents.add(alleEvents.get(index);
14      index--;
15  }
16
17  return gefundeneEvents;
```

**Listing 2:** Event-Suche: Finde alle Events zwischen zwei Zeitpunkten

### Heatmap-Generierung

```
1   procedure generiereHeatmap(recording) {
2       final int hitzeRadius = 80;
3       final Boolean ignoriereProzentsatz = true;
4       final double prozentsatz = 0.05;
5
6       heatMap = leere Matrix;
7       alpha = leere Matrix;
8
9       // Hitzekreis (alpha) erstellen
10      for (r = 0; r < alpha.zeilen(); r++) {
11          for (c = 0; c < alpha.spalten(); c++) {
12              x = hitzeRadius - r;
13              y = hitzeRadius - c;
14              radius = Math.hypotenuse(x, y);
15              pixelFarbe = alpha.get(r, c);
16
17              if (radius > hitzeRadius)
18                  pixelFarbe[0] = 0; // transparent
19              else
20                  pixelFarbe[0] = Math.round((1.0 - (radius / hitzeRadius)) * 255); // partial
21
```

```
            alpha.put(r, c, pixelFarbe);
        }
    }

    zaehler = 0;

    numEvents = recording.getEvents().size();
    anzahlZuIgnorieren = Math.round(numEvents / (100 / prozentsatz));

    for (e : recording.getEvents()) {
        zaehler++;

        if (ignoriereProzentsatz && anzahlZuIgnorieren > 1)
            if (zaehler % anzahlZuIgnorieren != 0)
                continue;

        // Hitze aufaddieren
        xOrigin = e.koordinaten().x - hitzeRadius;
        yOrigin = e.koordinaten().y - hitzeRadius;

        for (j = 0; j < alpha.zeilen(); j++) {
            if (j + yOrigin < 0 || j + yOrigin >= heatMap.zeilen())
                continue;
            for (i = 0; i < alpha.spalten(); i++) {
                if (i + xOrigin < 0 || i + xOrigin >= heatMap.spalten())
                    continue;

                bildPixelFarbe = heatMap.get(j + yOrigin, i + xOrigin);
                alphaFarbe = alpha.get(j, i);
                // aggregiere nach Dauer der Fixation; fuer Haeufung nur alphaFarbe * 1 draufaddieren
                bildPixelFarbe += (alphaFarbe * e.fixationsDauer());
                heatMap.put(j + yOrigin, i + xOrigin, bildPixelFarbe);
            }
        }
    }

    // groesster wert -> 255, kleinster wert -> 0
    heatMap = normalisiereHeatmap(heatMap, 255, 0, Normalisierungstyp.MIN_MAX);
    // Farbverlauf Hot: Schwarz -> Dunkelrot -> Hellrot -> Orange -> Gelb ->  Wei ; (wenig -> viel)
    bild = mappeHeatmapFarben(heatMap, Colormap.HOT);

    return bild;
}
```

**Listing 3:** Heatmap-Generierung: Erzeuge Attention Map über gegebenes Eye-Tracker-Recording.

## Zeitliche Synchronisation

### »KGS Go«-spezifische Methode

```
procedure findeVerschiebung(steinKoordinaten) {
    hostVideo.geheZu(0);
    gastVideo.geheZu(0);
    zeitVerschiebung = schaetzWert;
```

```
    xHR = 15 + hostAufzeichnung.spielfeldBereich().mitte();
    yHR = 15 + hostAufzeichnung.spielfeldBereich().mitte();
    xHL = -15 + hostAufzeichnung.spielfeldBereich().mitte();
    yHL = -15 + hostAufzeichnung.spielfeldBereich().mitte();
    if (steinKoordinaten != null) {
        xHR = 15 + steinKoordinaten.x;
        yHR = 15 + steinKoordinaten.y;
        xHL = -15 + steinKoordinaten.x;
        yHL = -15 + steinKoordinaten.y;
    }
    transformedCoordR = transformiereKoordinaten(Punkt(xHr, yHr), hostAufzeichnung,
         gastAufzeichnung);
    xGR = transformedCoordR.x;
    yGR = transformedCoordR.y;
    transformedCoordL = transformiereKoordinaten(Punkt(xHl, yHl), hostAufzeichnung,
         gastAufzeichnung);
    xGL = transformedCoordL.x;
    yGL = transformedCoordL.y;

    bildHost = leere Matrix;
    bildGast = leere Matrix;
    pufferBildGast = pufferBildHost = null;
    ersterMarkerMitStein = null;
    aktiverSpieler = passiverSpieler = null;
    xPR = yPR = xPL = yPL = 0;
    beige = Color(0xb0, 0x8b, 0x31);

    ersterMarkerMitStein = findeSetzenDesSteines(hostAufzeichnung, hostVideo);
    if (ersterMarkerMitStein != null) {
        aktiverSpieler = hostVideo;
        passiverSpieler = gastVideo;
        xPR = xGr;
        yPR = yGr;
        xPL = xGl;
        yPL = yGl;
    } else {
        ersterMarkerMitStein = findeSetzenDesSteines(gastAufzeichnung, gastVideo);
        if (ersterMarkerMitStein != null) {
                aktiverSpieler = gastVideo;
                passiverSpieler = hostVideo;
                xPR = xHr;
                yPR = yHr;
                xPL = xHl;
                yPL = yHl;
                schaetzWert *= -1;
        }
    }
    if (ersterMarkerMitStein == null) {
        return zeitVerschiebung;
    }

    timeStamp = ersterMarkerMitStein.frame / aktiverSpieler.fps() * 1000;
    startFrameSchaetzwert = (timeStamp + schaetzWert) / 1000 * passiverSpieler.fps() - 10;
    endFrameSchaetzwert = startFrameSchaetzwert + 20;
    passiverSpieler.geheZu(startFrameSchaetzwert);

    bildPassiv = leere Matrix;
```

```
60      pufferBildPassiv = null;
61      markerPassivR = {};
62      markerPassivL = {};
63
64      while (passiverSpieler.read(bildPassiv)) {
65          pufferBildPassiv = bildZuPuffer(bildPassiv);
66          neueFarbeR = pufferBildPassiv.getRGB(xPr, yPr);
67          neuerMarkerR = Marker(neueFarbeR, passiverSpieler.aktuellerFrameCounter() - 1.0);
68          markerPassivR.add(neuerMarkerr);
69          neueFarbeL = pufferBildPassiv.getRGB(xPl, yPl);
70          neuerMarkerL = Marker(neueFarbeL, passiverSpieler.aktuellerFrameCounter() - 1.0);
71          markerPassivL.add(neuerMarkerl);
72
73          if (neuerMarkerL.frame >= endFrameSchaetzwert)
74              break;
75      }
76
77      s = markerPassivR.groesse() - 1;
78      while (s >= 0) {
79          colorR = markerPassivR.get(s).color;
80          colorL = markerPassivL.get(s).color;
81
82          // finde beige von hinten
83          if (14.0 > cielabCiede2000(beige.getRGB(), colorR) || 14.0 > cielabCiede2000(beige.getRGB(),
                colorL))
84              break;
85
86          s--;
87      }
88
89      if (s < 0)
90          s = 0;
91
92      passiverSpieler.geheZu(markerPassivR.get(s).frame);
93      ersteUnterschiedlicheFarbePassiv = 0;
94
95      while (passiverSpieler.read(bildPassiv)) {
96          pufferBildPassiv = bildZuPuffer(bildPassiv);
97          neueFarbeR = pufferBildPassiv.getRGB(xPr, yPr);
98          neueFarbeL = pufferBildPassiv.getRGB(xPl, yPl);
99
100         if (14.0 <= cielabCiede2000(beige.getRGB(), neueFarbeR) && 14.0 <=
                cielabCiede2000(beige.getRGB(), neueFarbeL)) {
101             ersteUnterschiedlicheFarbePassiv = neueFarbeR;
102             break;
103         }
104     }
105
106     ersterFrameMitSteinPassiv = (passiverSpieler.aktuellerFrameCounter() - 1.0);
107     zeitstempelAktiv = ersterMarkerMitStein.frame / aktiverSpieler.fps() * 1000;
108     zeitstempelPassiv = ersterFrameMitSteinPassiv / passiverSpieler.fps() * 1000;
109     zeitVerschiebung = zeitstempelPassiv - zeitstempelAktiv;
110
111     if (aktiverSpieler == gastVideo) { // zurueckdrehen
112         schaetzWert *= -1;
113         zeitVerschiebung *= -1;
114     }
```

```
115
116     return zeitVerschiebung;
117 }
118
119 procedure findeSetzenDesSteines(aufzeichnung, video) {
120     bild = leere Matrix;
121     markerListeR = {{}};
122     markerListeL = {{}};
123     beige = Color(0xb0, 0x8b, 0x31);
124
125     for (zeitstempelKlick : aufzeichnung.klicks()) {
126         startPunkt = floor((zeitstempelKlick / 1000.0) * video.fps()) - 8;
127         video.geheZu(startPunkt);
128         farbenUmKlickR = {};
129         farbenUmKlickL = {};
130
131         // 17 proben zeitlich um jeden klick herum sammeln
132         for (i = 0; i < 17; i++) {
133             if (video.read(bild)) {
134                 pufferBild = bildZuPuffer(bild);
135                 neueFarbeR = pufferBild.getRGB(xHr, yHr);
136                 farbenUmKlickR.add(Marker(neueFarbeR, video.aktuellerFrameCounter() - 1.0));
137                 neueFarbeL = pufferBild.getRGB(xHl, yHl);
138                 farbenUmKlickL.add(Marker(neueFarbeL, video.aktuellerFrameCounter() - 1.0));
139             }
140         }
141
142         if (farbenUmKlickR.groesse() > 0)
143             markerListeR.add(farbenUmKlickR);
144         if (farbenUmKlickL.groesse() > 0)
145             markerListeL.add(farbenUmKlickL);
146     }
147
148     if (markerListeR.groesse() != markerListeL.groesse()) {
149         return null;
150     }
151
152     // schauen, ob spieler den stein setzt (bzw der stein in der naehe eines klicks auftaucht)
153     for (i = 0; i < markerListeR.groesse(); i++) {
154         markerR = markerListeR.get(i);
155         markerL = markerListeL.get(i);
156         ersterMarkerR = markerR.get(0);
157         ersterMarkerL = markerL.get(0);
158
159         if (14.0 <= cielabCiede2000(ersterMarkerR.color, beige.getRGB()) && 14.0 <=
                 cielabCiede2000(ersterMarkerL.color, beige.getRGB()))
160             continue; // vor dem klick sind beide nicht beige -> uninteressant, stein wird nicht
                     gesetzt
161
162         // wir haben (mind.) eine beige farbe
163         for (j = 1; j < markerR.groesse(); j++) {
164             if (14.0 <= cielabCiede2000(markerR.get(j).color, beige.getRGB()) && 14.0 <=
                     cielabCiede2000(markerL.get(j).color, beige.getRGB())) {
165
166                 return markerR.get(j);
167             }
168         }
```

```
    }

    return null;
}
```

**Listing 4:** Zeitliche Synchronisation: Finde in beiden Recordings den Zeitpunkt, zu dem ein gegebener Stein gesetzt wird und berechne Zeitverschiebung

### Histogramm-Methode

```
procedure findeFrameUndBerechneVerschiebung(hostFrameStartIndex) {
    // Gittergroesse: 16x16
    kernelSize = 16;
    // 40% als Schranke fuer Correlation
    correlationThreshold = 0.4;
    // mindestens 1 Zelle  muss Correlation-Schranke (<40% Aehnlichkeit) erfuellen
    minAnzAbweichendeZellen = 1;
    bildHost, bildHostSpielfeld, bildGast, bildGastSpielfeld = leere Matrix;
    histHost, histGast = {};
    // Parameter fuer Histogramm-Generierung
    histogramRanges = {0, 180, 0, 256};
    histogramSize = {50, 60};
    histogramChannels = {0, 1};

    hostVideo.geheZu(hostFrameStartIndex);

    if (hostVideo.leseBild(bildHost)) {
        bildHostSpielfeld = bildHost.beschneide(gastAufzeichnung.spielfeldBereich().groesse());
        zerteiltesSpielfeld = zerteileMatrixInZellenArray(bildHostSpielfeld, kernelSize);

        for (i = 0; i < zerteiltesSpielfeld.length; i++)
            histHost[i] = berechneHistogramm(zerteiltesSpielfeld[i], histogramChannels,
                histogramSize, histogramRanges);
    }
    else
        return 0;

    // finde erstes ueber x% abweichendes frame
    histHostAktuell = {};
    erstesNeuesFrameHost = 0;
    anzahlTreffer = 0;

    while (hostVideo.leseBild(bildHost)) {
        bildHostSpielfeld = bildHost.beschneide(gastAufzeichnung.spielfeldBereich().groesse());
        zerteiltesSpielfeld = zerteileMatrixInZellenArray(bildHostSpielfeld, kernelSize);

        for (i = 0; i < zerteiltesSpielfeld.length; i++)
            histHostAktuell[i] = berechneHistogramm(zerteiltesSpielfeld[i], histogramChannels,
                histogramSize, histogramRanges);

        correlation = {};

        for (i = 0; i < correlation.length; i++) {
            correlation[i] = vergleicheHistogramme(histHostAktuell[i], histHost[i],
                Vergleichsmethode.CORRELATION);
```

```
43
44              if (correlation[i] <= correlationThreshold) {
45                  erstesNeuesFrameHost = hostVideo.aktuellerFrameCounter() - 1;
46                  histHost = histHostAktuell;
47                  anzahlTreffer++;
48              }
49          }
50
51          if (anzahlTreffer >= minAnzAbweichendeZellen)
52              break;
53      }
54
55      hostFrameZeitstempelMillisekunden = erstesNeuesFrameHost / hostVideo.fps() * 1000;
56
57      // Nutze Systemzeit-Abweichung als groben Orientierungswert
58      systemzeitAbweichung = berechneSystemzeitAbweichung(host, gast);
59      startingFrameGuess = ((hostFrameZeitstempelMillisekunden + systemzeitAbweichung) / 1000 *
            gastVideo.fps()) - 150;
60      if (startingFrameGuess < 0) startingFrameGuess = 0;
61      gastVideo.geheZu(startingFrameGuess);
62
63      groessteCorrelation = 0.0;
64      groessteCorrelationFrame = 0;
65      iterationen = 0;
66      bildGroessteCorrelation = leere Matrix;
67
68      histGast = {};
69      histGastGroessteCorrelation = {};
70
71      while (gastVideo.leseBild(bildGast)) {
72          bildGastSpielfeld = bildGast.beschneide(gastAufzeichnung.spielfeldBereich().groesse());
73          zerteiltesSpielfeld = zerteileMatrixInZellenArray(bildGastSpielfeld, kernelSize);
74
75          for (i = 0; i < zerteiltesSpielfeld.length; i++)
76              histGast[i] = berechneHistogramm(zerteiltesSpielfeld[i], histogramChannels,
                    histogramSize, histogramRanges);
77
78          correlation = {};
79
80          for (i = 0; i < correlation.length; i++)
81              correlation[i] = vergleicheHistogramme(histGast[i], histHost[i],
                    Vergleichsmethode.CORRELATION);
82
83          summe = berechneArraySumme(correlation);
84          if (summe > groessteCorrelation) {
85              groessteCorrelation = summe;
86              groessteCorrelationFrame = gastVideo.aktuellerFrameCounter() - 1;
87              bildGroessteCorrelation = bildGastSpielfeld;
88              histGastGroessteCorrelation = histGast;
89          }
90
91          iterationen++;
92
93          if (iterationen > 299)
94              break;
95      }
96
```

```
97      // finde erstes vorhergehendes  ber  x% abweichendes frame
98
99      histGastAktuell = {};
100     erstesNeuesFrameGast = 0;
101     anzahlTreffer = 0;
102
103     for (minus = 1; minus < 300; minus++) {
104         gastVideo.geheZu(groessteCorrelationFrame - minus);
105
106         if (minus > groessteCorrelationFrame)
107             break;
108
109         if (!gastVideo.leseBild(bildGast))
110             break;
111
112         bildGastSpielfeld = bildGast.beschneide(gastAufzeichnung.spielfeldBereich().groesse());
113         zerteiltesSpielfeld = Utils.splitMatrixIntoCells(bildGastSpielfeld, kernelSize);
114
115         for (i = 0; i < zerteiltesSpielfeld.length; i++)
116             histGastAktuell[i] = berechneHistogramm(zerteiltesSpielfeld[i], histogramChannels,
                    histogramSize, histogramRanges);
117
118         correlation = {}
119
120         for (i = 0; i < correlation.length; i++) {
121             correlation[i] = vergleicheHistogramme(histGastAktuell[i],
                        histGastGroessteCorrelation[i], Vergleichsmethode.CORRELATION);
122
123             if (correlation[i] <= correlationThreshold) {
124                 erstesNeuesFrameGast = gastVideo.aktuellerFrameCounter();
125                 anzahlTreffer++;
126             }
127         }
128
129         if (anzahlTreffer >= minAnzAbweichendeZellen)
130             break;
131     }
132
133     gastFrameZeitstempelMillisekunden = erstesNeuesFrameGast / gastVideo.fps() * 1000;
134     zeitVerschiebung = gastFrameZeitstempelMillisekunden - hostFrameZeitstempelMillisekunden;
135     return zeitVerschiebung;
136 }
```

**Listing 5:** Zeitliche Synchronisation: Finde zu gegebenem Frame in Host-Recording ähnlichstes Frame in Gast-Recording und berechne Zeitverschiebung

# Literaturverzeichnis


[AGA] *A Brief History Of Go*. https://www.usgo.org/brief-history-go. Zul. geprüft: 02.08.2019 (zitiert auf S. 24).

[BKR+17] T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, T. Ertl. „Visualization of Eye Tracking Data: A Taxonomy and Survey: Visualization of Eye Tracking Data“. In: *Computer Graphics Forum* (Feb. 2017). DOI: 10.1111/cgf.13079 (zitiert auf S. 15, 16).

[BRT09] A. Bulling, D. Roggen, G. Tröster. „Wearable EOG goggles: Seamless sensing and context-awareness in everyday environments“. In: *JAISE* 1 (Jan. 2009), S. 157–171. DOI: 10.3233/AIS-2009-0020 (zitiert auf S. 16).

[BSBE17] T. Blascheck, M. Schweizer, F. Beck, T. Ertl. „Visual Comparison of Eye Movement Patterns“. In: *Comput. Graph. Forum* 36 (2017), S. 87–97. DOI: 10.1111/cgf.13170 (zitiert auf S. 13).

[CC73] T. N. Cornsweet, H. D. Crane. „Accurate two-dimensional eye tracker using first and fourth Purkinje images“. In: *J. Opt. Soc. Am.* 63.8 (Aug. 1973), S. 921–928. DOI: 10.1364/JOSA.63.000921. URL: http://www.osapublishing.org/abstract.cfm?URI=josa-63-8-921 (zitiert auf S. 17).

[Cha07] S.-H. Cha. „Comprehensive survey on distance/similarity measures between probability density functions“. In: *City* 1.2 (2007), S. 1 (zitiert auf S. 23).

[CS02] S.-H. Cha, S. Srihari. „On measuring the distance between histograms“. In: *Pattern Recognition* 35 (Juni 2002), S. 1355–1370. DOI: 10.1016/S0031-3203(01)00118-2 (zitiert auf S. 22).

[DPMO12] A. T. Duchowski, M. M. Price, M. Meyer, P. Orero. „Aggregate Gaze Visualization with Real-time Heatmaps“. In: *Proceedings of the Symposium on Eye Tracking Research and Applications*. ETRA '12. Santa Barbara, California: ACM, 2012, S. 13–20. ISBN: 978-1-4503-1221-9. DOI: 10.1145/2168556.2168558. URL: http://doi.acm.org/10.1145/2168556.2168558 (zitiert auf S. 21).

[Duc07] A. T. Duchowski. „Eye tracking methodology“. In: *Theory and practice* 328.614 (2007), S. 2–3. DOI: 10.1007/978-3-319-57883-5 (zitiert auf S. 13, 15, 16).

[FGM15] M. Frutos-Pascual, B. García-Zapirain, Q. H. Mehdi. „Where do they look at? Analysis of gaze interaction in children while playing a puzzle game“. In: *2015 Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES)*. Juli 2015, S. 103–106. DOI: 10.1109/CGames.2015.7272954 (zitiert auf S. 30).

[HA17] K. Holmqvist, R. Andersson. *Eye-tracking: A comprehensive guide to methods, paradigms and measures*. Nov. 2017. ISBN: ISBN-13: 978-1979484893 (zitiert auf S. 15).

[HJ10] D. W. Hansen, Q. Ji. „In the Eye of the Beholder: A Survey of Models for Eyes and Gaze“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.3 (März 2010), S. 478–500. DOI: 10.1109/TPAMI.2009.30 (zitiert auf S. 18).

[HNA+11] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, J. Van de Weijer. *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford, 2011 (zitiert auf S. 16).

[HSSK15] A. Harrer, C. Schlösser, P. Schlieker-Steens, A. Kienle. „Here's Looking at you, Kid – Can Gaze Awareness Help to Learn to Learn Together in Collaborative Problem Solving?“ In: *2015 IEEE 15th International Conference on Advanced Learning Technologies*. Juli 2015, S. 190–194. DOI: 10.1109/ICALT.2015.135 (zitiert auf S. 30).

[JF03] G. M. Johnson, M. D. Fairchild. „A top down description of S-CIELAB and CIE-DE2000“. In: *Color Research & Application* 28.6 (2003), S. 425–435. DOI: 10.1002/col.10195. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/col.10195 (zitiert auf S. 24, 61).

[KB14] Z. Kang, E. J. Bass. „Supporting the eye tracking analysis of multiple moving targets: Design concept and algorithm“. In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Okt. 2014, S. 3184–3189. DOI: 10.1109/SMC.2014.6974418 (zitiert auf S. 31).

[KGS] https://www.gokgs.com/. Zul. geprüft: 02.08.2019 (zitiert auf S. 24).

[KLHP19] G. Kütt, K. Lee, E. Hardacre, A. Papoutsaki. „Eye-Write: Gaze Sharing for Collaborative Writing“. In: Mai 2019, S. 1–12. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300727 (zitiert auf S. 30).

[LHL+09] Y. Liu, P. Hsueh, J. Lai, M. Sangin, M. Nussli, P. Dillenbourg. „Who is the expert? Analyzing gaze data to predict expertise level in collaborative applications“. In: *2009 IEEE International Conference on Multimedia and Expo*. Juni 2009, S. 898–901. DOI: 10.1109/ICME.2009.5202640 (zitiert auf S. 13, 30).

[LNJ11] W. Li, M. Nüssli, P. Jermann. „Exploring personal aspects using eye-tracking modality in Tetris-playing“. In: *2011 IEEE 13th International Workshop on Multimedia Signal Processing*. Okt. 2011, S. 1–4. DOI: 10.1109/MMSP.2011.6093841 (zitiert auf S. 13, 29).

[MPYK17] E. Moon, H. Park, J. Yura, D. Kim. „Novel Design of Artificial Eye Using EOG (Electrooculography)“. In: *2017 First IEEE International Conference on Robotic Computing (IRC)*. Apr. 2017, S. 404–407. DOI: 10.1109/IRC.2017.38 (zitiert auf S. 17).

[MR05] C. Morimoto, M. R.M. Mimica. „Eye gaze tracking techniques for interactive applications“. In: *Computer Vision and Image Understanding* 98 (Apr. 2005), S. 4–24. DOI: 10.1016/j.cviu.2004.07.010 (zitiert auf S. 17).

[MST+16] R. Mallick, D. Slayback, J. Touryan, A. J. Ries, B. J. Lance. „The use of eye metrics to index cognitive workload in video games“. In: *2016 IEEE Second Workshop on Eye Tracking and Visualization (ETVIS)*. Okt. 2016, S. 60–64. DOI: 10.1109/ETVIS.2016.7851168 (zitiert auf S. 29).

[OHC] *Histogram Comparison*. https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html. Zul. geprüft: 06.08.2019 (zitiert auf S. 22, 23).

[Pas01] G. Paschos. „Perceptually uniform color spaces for color texture analysis: an empirical evaluation“. In: *IEEE Transactions on Image Processing* 10.6 (Juni 2001), S. 932–937. DOI: 10.1109/83.923289 (zitiert auf S. 24).

[Pri06] C. Privitera. „The Scanpath Theory: its definition and later developments - art. no. 60570A“. In: *Proceedings of SPIE - The International Society for Optical Engineering* 6057 (Feb. 2006). DOI: 10.1117/12.674146 (zitiert auf S. 13, 21).

[RHTE18] L. Ribeiro da Silva Junior, F. Henrique Goncalves Cesar, F. Theoto Rocha, C. Eduardo Thomaz. „A Combined Eye-tracking and EEG Analysis on Chess Moves“. In: *IEEE Latin America Transactions* 16.5 (Mai 2018), S. 1288–1297. DOI: 10.1109/TLA.2018.8407099 (zitiert auf S. 13, 30).

[RMK+88] J. P. H. Reulen, J. T. Marcus, D. Koops, F. R. de Vries, G. Tiesinga, K. Boshuizen, J. E. Bos. „Precise recording of eye movement: the IRIS technique Part 1“. In: *Medical and Biological Engineering and Computing* 26.1 (Jan. 1988), S. 20–26. DOI: 10.1007/BF02441823. URL: https://doi.org/10.1007/BF02441823 (zitiert auf S. 18).

[RN88] J. L. Rodgers, W. A. Nicewander. „Thirteen Ways to Look at the Correlation Coefficient“. In: *The American Statistician* 42.1 (1988), S. 59–66. DOI: 10.1080/00031305.1988.10475524. eprint: https://doi.org/10.1080/00031305.1988.10475524. URL: https://doi.org/10.1080/00031305.1988.10475524 (zitiert auf S. 22).

[Rob63] D. A. Robinson. „A Method of Measuring Eye Movemnent Using a Scieral Search Coil in a Magnetic Field“. In: *IEEE Transactions on Bio-medical Electronics* 10.4 (Okt. 1963), S. 137–145. ISSN: 0096-0616. DOI: 10.1109/TBMEL.1963.4322822 (zitiert auf S. 16).

[SG00] D. Salvucci, J. Goldberg. „Identifying fixations and saccades in eye-tracking protocols“. In: Jan. 2000, S. 71–78. DOI: 10.1145/355017.355028 (zitiert auf S. 13, 19, 49).

[SKKH14] S. Shimizu, T. Kadogawa, S. Kikuchi, T. Hashizume. „Quantitative analysis of tennis experts' eye movement skill“. In: *2014 IEEE 13th International Workshop on Advanced Motion Control (AMC)*. März 2014, S. 203–207. DOI: 10.1109/AMC.2014.6823282 (zitiert auf S. 13, 29).

[Yar67] A. L. Yarbus. *Eye Movements and Vision*. New York: Plenum Press, 1967 (zitiert auf S. 13).

Nicht explizit gekennzeichnete URLs wurden zuletzt am 01. 08. 2019 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