Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Multiskalenvisualisierung von Trajektorien

Joachim Spalink

Studiengang:	Medieninformatik
Prüfer/in:	Prof. Dr. Daniel Weiskopf
Betreuer/in:	Dipl.-Inf. Rudolf Netzel
Beginn am:	5. Juni 2017
Beendet am:	12. Dezember 2017
CR-Nummer:	H.3.1, I.3.3, I.5.3

Kurzfassung

In der vorliegenden Bachelorarbeit wird die Multiskalenvisualisierung von Trajektorien behandelt. Durch multiple Skalen werden unterschiedliche Grade an Details visualisiert. Zur Messung der Ähnlichkeit von Pfaden, werden unterschiedliche Metriken vorgestellt. Für ein System zur Multiskalenvisulierung dieser Trajektorien, wurden Anforderungen aufgestellt. In diesem System sind unterschiedliche Ansichten, für einen erdachten Arbeitsablauf, vorhanden. In einem Prototyp, für dieses System, basierend auf Webtechnologien, wurde eine Visualisierung mit unterschiedlichen Skalen erstellt. Es gibt drei Ansichten für den erdachten Workflow. Die erste Ansicht ist der Trajectory View, in welcher eine visuelle Analyse bietet. In der zweiten Ansicht, Cluster View, steht eine maschinelle Analyse zur Verfügung. Für die maschinellen Analyse wurde MulitMatch als Ähnlichkeitsmaß implementiert, dieser wird für ein Clustering verwendet sowie im Detailvergleich. Die dritte Ansicht, genannt Compare View, bietet einen Detailvergleich. Dieser bietet die Möglichkeit, zwei Trajektorien genau auf ihre Ähnlichkeit zu untersuchen. An einem Beispiel unter Verwendung des Prototyps, wurde der in den Anforderungen aufgezeigte Workflow erläutert.

Abstract

This bachelor thesis deals with multiscale visualization of trajectories. Multiple scales visualize varying degrees of detail. To measure the similarity of paths, different metrics are presented. For a system for multi-scale visualization of these trajectories, requirements have been established. In this system different views exist for a thought-out workflow. In a prototype, for this system, based on web technologies, a visualization was created with different scales. There are three views for the conceived workflow. The first view is the Trajectory View, which offers a visual analysis. In the second view, Cluster View, a machine analysis is available. For machine analysis, MulitMatch was implemented as a measure for similarity, which is used for clustering and in detailed comparison. The third view, called Compare View, provides the detailed comparison. This offers the possibility to examine two trajectories exactly for their similarity. As an example using the prototype, the workflow shown in the requirements was explained.

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Listings

Verzeichnis der Algorithmen

Abkürzungsverzeichnis

AOI Area of Interest. 23

DTW Dynamic Time Warping [BC94]. 26

ED Euclidean Distance. 27

GPS Global Positioning System. 15

LCSS Lowest Common Subsequence [VGK02]. 26

MASSVIS Massachusetts (Massive) Visualization Dataset. 23

1 Einleitung

Die Positionen von Objekten werden seit langer Zeit erfasst und festgehalten. Früher geschah dies durch Beobachten der Umgebung, bis dann Kompass und Sextant die Bestimmung verbesserten. Heute wird die Position meistens mithilfe von Global Positioning System (GPS) bestimmt. Positionen in zeitlicher Abfolge beschreiben einen Pfad, dieser bildet den Weg eines Objektes in einem Koordinatensystem ab. Diesen Pfad nennt man Trajektorie. Durch die Analyse von Trajektorien können Bewegungsmuster erkannt werden. Zu untersuchendes Verhalten kann die Bewegungen von Fahrzeugen und Personen oder die Bewegung von Tieren in bestimmten Phasen dokumentieren, wenn für die Datenerfassung nur GPS verwendet wird. Es können aber auch beispielsweise Positionen in eingeschränkteren Bereichen aufgezeichnet werden, wie Bewegungen in Räumen oder die Augenbewegungen, während vorgegebene Aufgaben erledigt werden. Dabei kann das Verhalten der unterschiedlichen Personen verglichen werden, um festzustellen ob es Gemeinsamkeiten im Verhalten gibt, mit welchem das Ziel der Aufgabe erreicht wurde. Die Augenbewegungen werden in diesem Fall mit einem Eye-Tracker aufgenommen. Diese Daten werden dann auf einen Stimulus abgebildet und können daraufhin mit unterschiedlichen Methoden analysiert werden.

Diese Arbeit behandelt die Multiskalenvisualisierung von Trajektorien. Dabei sollen Trajektorien auf unterschiedlichen Skalen visualisiert und Gemeinsamkeiten beziehungsweise Unterschiede sichtbar gemacht werden. Durch unterschiedliche Skalen wird eine Abstraktion der Daten erstellt, durch welche zusammenhängende Punkte eines festen Bereichs gruppiert werden. Des Weiteren stehen unterschiedliche Metriken zur Verfügung, mit denen diese Unterschiede gruppiert und als Zahlen erfasst werden, wie gleich oder unterschiedlich die Trajektorien sind. Die Daten für dieses Projekt stammen aus einer Studie auf Metrokarten unterschiedlicher Städte. Die Teilnehmer sollten eine Route zwischen zwei Punkten finden. Während dieser Aufgabe wurden deren Augenbewegungen aufgenommen. Es sind 40 unterschiedliche Teilnehmer und 96 Stimuli in den Daten enthalten. Aufgrund der Menge an Daten ist ein automatisches Verfahren für eine erste grobe Analyse vorzuziehen, bevor Details manuell analysiert werden.

Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Grundlagen: Hier werden die Grundlagen dieser Arbeit beschrieben.

Kapitel 3 – Verwandte Arbeiten: Behandelt kurz verwandte Arbeiten.

Kapitel 4 – Anforderungen: Diese Kapitel führt die Anforderungen auf.

Kapitel 5 – System: Hier wird die Funktion des Systems vorgestellt.

Kapitel 6 – Use-Case: Beschreibt den Ablauf einer Analyse.

Kapitel 7 – Zusammenfassung und Ausblick: Fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte für zukünftige Erweiterungen vor.

2 Grundlagen

In diesem Kapitel werden Algorithmen und Technologien vorgestellt, die genannt und teilweise auch in dem entstandenen System verwendet werden. Durch diese Grundlagen kann ein Überblick über die vorhandenen Möglichkeiten geschaffen werden.

2.1 Algorithmen

Folgender Abschnitt beschreibt mögliche Algorithmen, welche für diese Arbeit relevant sind. Dazu zählen auch Methoden die gut in verwandten Arbeiten (Kapitel 3) funktioniert oder häufig vorkommen. Die Grundlagen für die Algorithmen bieten die Möglichkeit die genannten Verfahren besser zu verstehen.

2.1.1 Levenshtein-Distanz

Mit der Levenshtein-Distanz [Lev66] kann die Ähnlichkeit von Zeichenketten berechnet werden. Das Verfahren berechnet die minimale Anzahl an Operationen um eine Zeichenkette in die andere zu überführen. Ein simpler Algorithmus der dies Implementiert ist in Algorithmus 2.1 aufgezeigt. Die Matrix, welche durch den Algorithmus ausgefüllt wird, ist in Abbildung 2.1 gezeigt. Der initiale Schritt füllt die obere Zeile und linke Reihe mit den Zahlen von 0 bis Zeichenkettenlänge. Danach wird für jede Zelle der Wert errechnet und eingetragen. Die minimale Anzahl an Operationen ist in der untersten rechten Zelle der Matrix, siehe Abbildung 2.1, enthalten, nach Ausführung des Algorithmus.

2.1.2 Fréchet-Distanz

Die Fréchet-Distanz [EGH+02] ist vergleichbar mit einer Hundeleine wie in Abbildung 2.2 beispielhaft dargestellt. Sie wird errechnet indem man auf den Pfaden entlang läuft und die kürzeste Entfernung zur anderen zu Erhalten. So haben zwei parallele Pfade ihren Abstand zueinander als Fréchet-Distanz. Die maximale Distanz zwischen zwei Pfaden entsteht, falls der zweite Pfad, die Umkehrung des ersten Pfades (rotiert um 180 Grad) ist.

		H	A	U	S
	0	1	2	3	4
B	1	1	2	3	4
A	2	2	1	2	3
U	3	3	2	1	2
M	4	4	3	2	2

Abbildung 2.1: Levenshtein als Matrix, diese entsteht wenn der Algorithmus 2.1 ausgeführt wird. Die minimale Anzahl an Operationen zur Umwandlung von Haus in Baum steht unten rechts der Matrix.

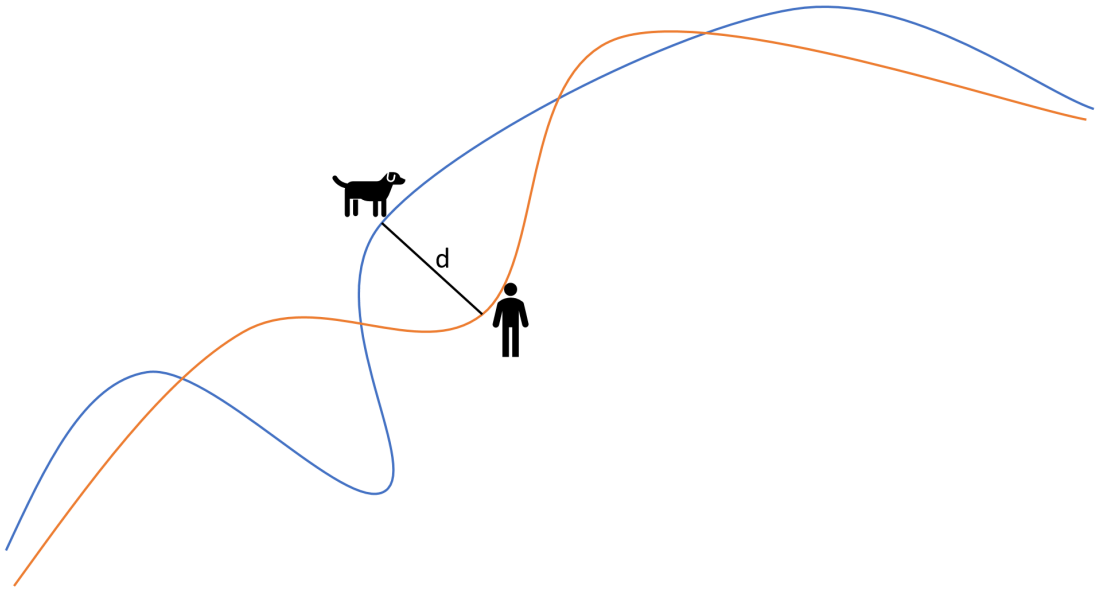


Abbildung 2.2: Fréchet-Distanz, es wird auf dem Pfad entlang gegangen und die kürzest mögliche Distanz d bestimmt.

Algorithmus 2.1 Levenshtein

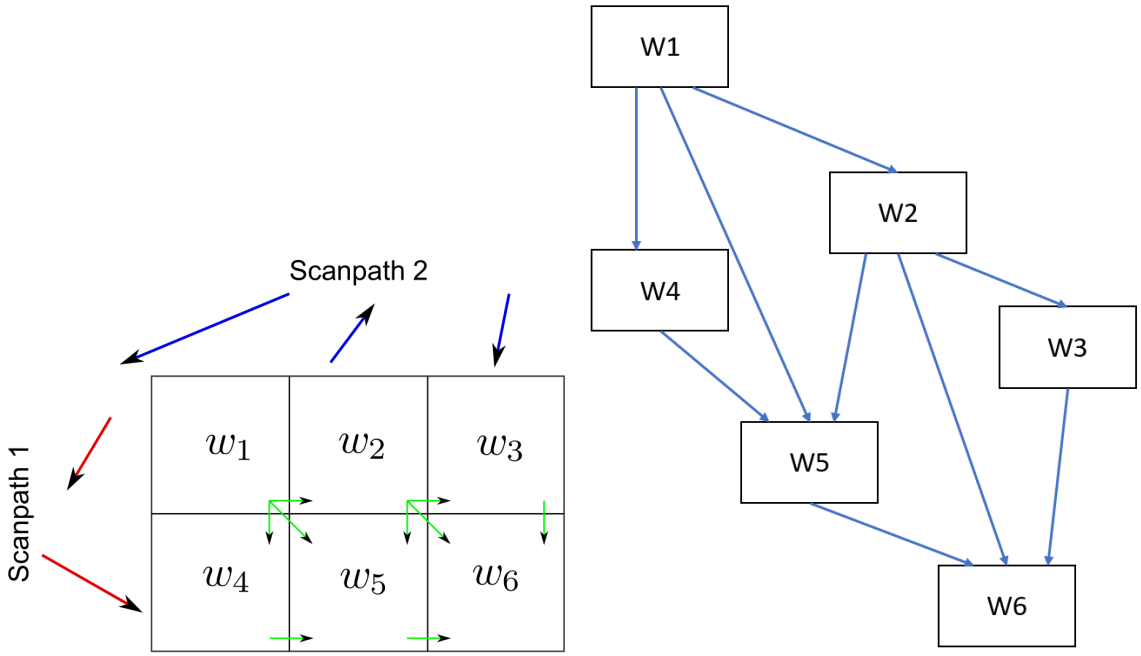
```
procedure LEVENSHTEIN(s1,s2)
    dist[][] ← initresultMatrix    // Obere Zeile und linke Reihe mit 0 bis Zeichenkettenlänge füllen
    for i = 0 to s1.length; i++ do
        for j = 0 to s2.length; j++ do
            if s1[i]=s2[j] then
                dist[i][j] ← MIN(dist[i − 1][j] + 1, dist[i][j − 1] + 1, dist[i − 1][j − 1])
            else
                dist[i][j] ← MIN(dist[i − 1][j] + 1, dist[i][j − 1] + 1, dist[i − 1][j − 1] + 1)
            end if
        end for
    end for
    return dist[s1.length][s2.length]
end procedure
```

Algorithmus 2.2 MultiMatch

```
procedure COMPARE(traj1, traj2, diagonal)
    vectors1 ← CREATEVECTORS(traj1)
    vectors2 ← CREATEVECOTRS(traj2)
    simMatrix ← COMPUTESIMILARITYMATRIX()
    path ← GETPATHTHROUGHSIMMATRIX(simMatrix)
    singleScores ← []
    for all p in path do
        tempScore ← COMPUTESCORES(p)
        SINGLESCORES.PUSH(tempScore)
    end for
    score ← AVERAGE(singleScores)
    return score
end procedure
```

2.1.3 MultiMatch

Ein weiteres Verfahren um die Ähnlichkeit von Trajektorien zu Messen ist von Jarodzka et al. [JHN10] und Dewhurst et al. [DNJ+12] beschrieben, genannt MultiMatch. Es vergleicht zwei Trajektorien miteinander basierend auf ihren Vektoren. Der erste Schritt ist das alignieren der Trajektorien. Dafür wird am Anfang eine Ähnlichkeitmatrix der einzelnen Vektoren erstellt. Durch diese Matrix wird ein Graph aufgebaut nach dem Schema in Abbildung 2.3a, die grünen Pfeile kennzeichnen die Kanten des Graphen. So kann dann mit Dijkstra [Dij59] der kürzeste Pfad durch den Graphen (Algorithmus 2.3 findet den kürzesten Pfad durch einen Graph gegeben als Adjazenzliste) ermittelt und jeder Vektor der zwei Trajektorien basieren auf dem Pfad verglichen werden, dieser Vergleichswert wird über die fünf Einzelwerte Form, Länge Richtung, Position, Dauer bewerkstelligt. Das Vorgehen von MultiMatch als Pseudocode ist in Algorithmus 2.2 kompakt zu sehen.



(a) MutliMatch Ähnlichkeitsmatrix **(b)** Graph zu Abbildung 2.3a

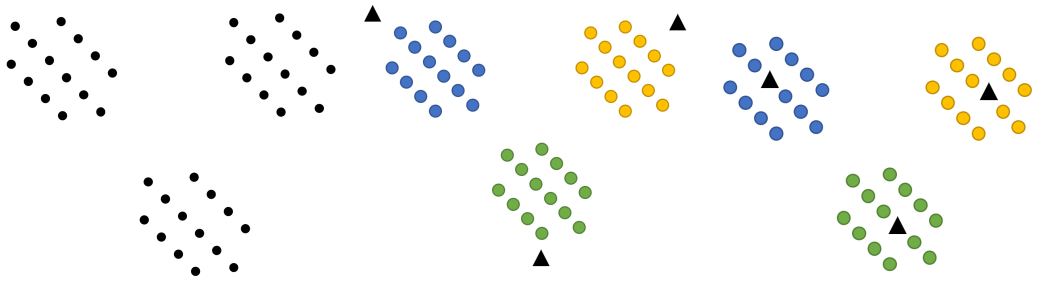
Abbildung 2.3: In 2.3a ist die Ähnlichkeitsmatrix der einzelnen Vektoren des MultiMatch mit möglichen Pfaden (grün) durch die Matrix. Entnommen aus [JHN10] Seite 4

Algorithmus 2.3 Kürzester Pfad

```
procedure GETSHORTESTPATH(target,list)
    path ← [target]
    u ← target
    while PREV.GET(u) do
        u ← PREV.GET(u).key
        PATH.PUSH(u)
    end while
    return PATH.REVERSE()
end procedure
```

2.1.4 k-means

k-means [Llo82] kann zur Bildung von Clustern verwendet werden. Hierbei gibt k die Anzahl der zu erstellenden Cluster an. Jeder Cluster sammelt um sein Zentrum Punkte, die eine möglichst geringe Distanz zu ihm haben. Die Daten können beispielsweise wie in Abbildung 2.4a vorhanden sein. Der erste Schritt bei diesem Verfahren entspricht der Wahl von k Startpunkten, dies sind die initialen Zentren für die k Cluster. Daraufhin wird für alle Punkte die Distanz zu den Startpunkten ausgerechnet und dem mit der niedrigsten Entfernung zugewiesen wie in Abbildung 2.4b. Nachdem alle Punkte den Clustern zugeordnet wurden, werden die Clusterzentren anhand der Punkte im Cluster neu berechnet. Diese beiden Schritte werden entweder solange wiederholt bis



(a) Anfangsdaten **(b)** Zwischenschritt des k-means **(c)** Ende des Clusterings

Abbildung 2.4: Ablauf von k-means. Zu Beginn in Abbildung 2.4a des k-means sind die Daten ungruppiert. Nach dem wählen von Clusterzentren hier dargestellt durch die leeren Punkte. Können mehrere Iteration des Verfahrens durchgeführt werden. Ein Zwischenschritt mit den Zentren am Rand der Cluster in Abbildung 2.4b dabei sind Farben nach den entstehenden Clustern. Am Ende des Verfahrens in Abbildung 2.4c sind die Zentren der Cluster in die Mitte dieser gewandert.

die Veränderung minimal ist oder eine festgelegte Anzahl an Wiederholungen erreicht ist siehe Abbildung 2.4c.

2.2 Technologien

Um möglichst unabhängig von einem Betriebssystem zu sein, wurde der Fokus auf Webtechniken gelegt, da nur ein Browser mit Javascript vorhanden sein muss. Zur Implementierung des System wurden die folgenden Technologien verwendet: Typescript [Mic17], Node.js [Nod17], D3.js [BOH11], Angular [Ang17] und Bootstrap [OTc17]. Diese Technologien werden aktuell häufig für Webseiten eingesetzt und sind kostenlos verfügbar.

2.2.1 Javascript/Typescript

Als Sprache wird zum Großteil Typescript [Mic17] verwendet. Sie wird durch Microsoft entwickelt, basierend auf dem ECMAScript-6-Standard [Int15]. Es bietet Sprachkonstrukte wie Klassen, Interfaces, Vererbung und weitere häufig in Objektorientierten vorkommenden Konstrukte. Javascriptcode ist auch gültiger Typescriptcode. Der Typesciptcompiler kompiliert den Code zu ECMAScript (minimum ECMAScript 3).

2.2.2 Node.js

Die technische Komponente wird mithilfe der Plattform Node.js [Nod17] entwickelt. Mit dieser lässt sich einfach und schnell ein Webserver realisieren. Die Plattform läuft in der Laufzeitumgebung „V8“, welche auch im Webbrowser Google Chrome zum Einsatz kommt. In Node.js sind die einzelnen Funktionen in Module aufgeteilt, für einen einfachen Webserver würde man das Modul „http“ verwenden, für Operationen auf dem Dateisystem das Modul „fs“. Für die Verwaltung der

Module gibt es den Paketmanager NPM. Dort gibt es über 400.000 Module, die bekanntesten sind vermutlich JQuery und Bootstrap.

2.2.3 D3.js

D3.js abgekürzt für „Data-Driven Documents“[BOH11] ist eine Javascript Bibliothek, veröffentlicht im Jahr 2011 von Bostock et al. mit Ursprung in Stanford, welche zur Darstellung von Daten verwendet werden kann. Sie baut auf die Erfahrungen, welche die Ersteller bei Protovis [BH09] gesammelt haben auf. Die Bibliothek kombiniert HTML, SVG und CSS zur Visualisierung. Es gibt Methoden um Elemente zu selektieren und transformieren, wodurch auch Animationen möglich gemacht werden. Als Grundformen gibt es unter anderem Flächen, Linien, Kuchendiagramme und viele weitere, des Weiteren stehen wiederverwendbare Funktionen und Fabrikfunktionen zur Verfügung.

2.2.4 Angular

Angular [Ang17] ist ein Framework zur Erstellung von wiederverwendbaren Komponenten. Die erstellten Anwendungen laufen auf nahezu jeder Plattform. Eine Angular Anwendung besteht aus einer Wurzelkomponente in welche die Anderen eingefügt werden, daraus entsteht eine Komponentenhierarchie.

2.2.5 Bootstrap

Bootstrap [OTc17] ist ein CSS-Framework, was anfangs nur von Twitter entwickelt und eingesetzt wurde, aber durch das gute Konzept wurde es als Open-Source veröffentlicht. Bootstrap besteht aus mehreren Modulen die konfiguriert werden können. Die Grundlage hinter dem Bootstrap Design bildet ein Gitter mit zwölf Spalten. Das dazugehörige Stylesheet bietet ein einheitliches, modernes Aussehen auf unterschiedlichen Geräten. Es beinhaltet häufig genutzte Oberflächenelemente für Webseiten, wie einige Navigationselemente oder Buttons welche auch erweiterte Funktionen haben.

3 Verwandte Arbeiten

Der folgende Abschnitt stellt verwandte Arbeiten untergliedert in die Bereiche vor, die auf Zeichenketten operieren oder geometrische Ansätze verfolgen. Diese Unterteilung basiert darauf, in welcher Form die Daten in der Analyse vorliegen. So wandeln die auf Zeichenketten basierenden Verfahren die Daten als erstes in Sequenzen von Zeichen um, bevor die Analyse startet. Dort werden beispielsweise einige Techniken aus der Biologie oder maschinellen Sprachverarbeitung verwendet. Der andere Teil nutzt geometrische Maße um Trajektorien zu analysieren, zum Beispiel Distanz zwischen zwei Pfaden oder basierend auf den Attributen der Vektoren des Pfades, wie Länge, Form oder Richtung.

3.1 Zeichenketten

Um Augenbewegungen analysierbar zu machen werden Metriken benötigt. Bylinskii und Borkin [BBK+17] stellen mehrere auf Fixationen aufbauende Metriken vor, um Blickmuster zu untersuchen. Als Daten wurde dort der Massachusetts (Massive) Visualization Dataset (MASSVIS) Datensatz [BBK+16] verwendet. Eine konstruierte Datenbank aus unterschiedlichen Visualisierungen, um tieferes Verständnis über die Elemente einer Visualisierung zu bekommen. Zu den Metriken gehören: die Anzahl Fixationen und Refixationen, die Gesamtdauer der Fixationen, die Dauer der Fixation in Teilbereichen und Anzahl an unterschiedlichen fixierten Elementen.

Der Artikel von Andrienko et al. [AABW12] beschreibt Methoden für die Arbeit mit Eye-Trackern und Richtlinien für die Auswahl der passenden Methode für die Analyse. Eine wichtige Prozedur in der Arbeit mit Bewegungsdaten sind Datentransformationen. Um die Analyse der Traktorien auf den, für das Experiment, wichtigen Teil zu beschränken müssen die Daten gefiltert werden. Die Autoren beschreiben, dass eine Anpassung des Startzeitpunktes immer Vernünftig ist. Eine Anpassung des Endzeitpunkt ist sinnvoll, wenn der Fokus des Experiments auf der Strategie der Teilnehmer liegt. Um die Anzahl der Datenpunkte zu reduzieren kann eine Räumliche Generalisierung als weitere Datentransformation angewendet werden, in dieser werden Punkte zu Gebieten zusammengefasst. Beispielweise dort wo dichte Gruppen an Punkten liegen, sozusagen dichte basiertes Clustering. Die letzte genannte Methode ist das Darstellen der Trajektorie durch eine endliche Menge an Orten, wie vordefinierte Orte von Interesse (Area of Interest (AOI)) siehe Abbildung 3.1.

BLAST[SWW+90] steht für „Basic Local Alignment Search Tool“ und misst Veränderungen basierend auf definierten Mutationswerten. Es ist ein simpler Algorithmus wie in Abbildung 3.2 zu sehen bestehend aus einfachen Schritten und kann für unterschiedliche Arten von Aufgaben basierend auf Zeichenketten verwendet werden. Als erstes wird die Sequenz in Teile zerlegt. Im Anschluss wird mit einer Datenbanksuche eine Liste von Paaren mit hoher Punktzahl in dieser Datenbank zusammengestellt. Daraufhin wird nach weiteren Treffern dieser Paare gesucht und abschließen die Treffer erweitert.

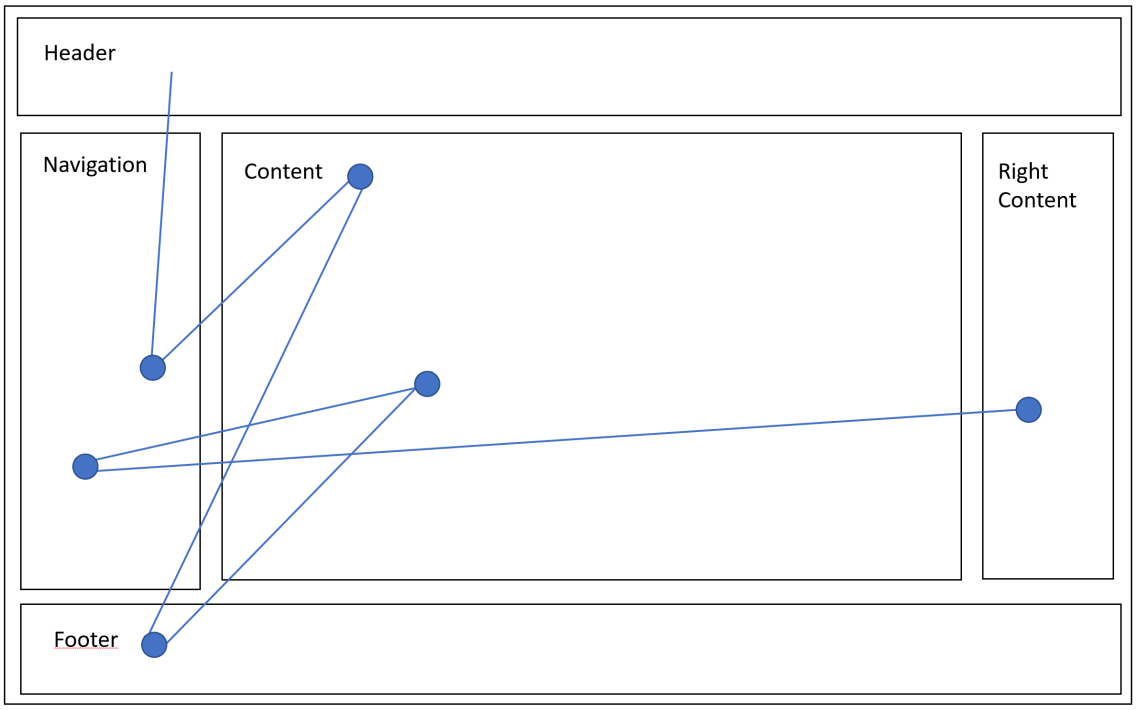


Abbildung 3.1: Ein Stimulus mit mehreren Bereichen (AOI). Aus der Reihenfolge, in welcher die einzelnen Bereiche besucht worden sind, kann eine Zeichenkette erstellte werden. Zum Beispiel basierend auf den Anfangsbuchstaben der einzelnen Bereiche „HNCFCNR“.

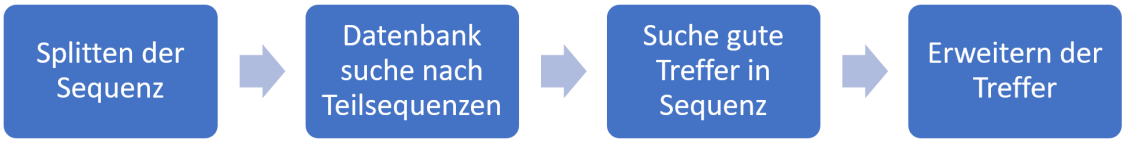


Abbildung 3.2: Einzelschritte von BLAST.

Der Needleman-Wunsch Algorithmus [NW70] wird hauptsächlich für den Vergleich von Aminosäurepaaren verwendet. Dabei werden die Proteine in eine Sequenz von Aminosäuren aufgeteilt. Diese zwei Ketten können dann verglichen Werten und die Ergebnisse in einer Matrix gespeichert werden. Eine Beispielsequenz ist in Abbildung 3.3 dargestellt. Dadurch wurden bei gleichen Zeichen eine 1 eingetragen sonst leer gelassen siehe Abbildung 3.3a. Anschließend wird angefangen die leeren Felder nach dem Algorithmus aufzufüllen. In dieser Matrix stehen dann alle möglichen Kombinationen dieser Proteine. Jeder Pfad durch diese Matrix stellt eine mögliche Kombination dar. Die beste Alignierung liegt beim dem Pfad mit den maximalen Werten (Abbildung 3.3b). Es können auch mehrere Möglichkeiten vorhanden sein.

ScanMatch [CMTG10] verwendet den Needleman-Wunsch Algorithmus um die Ähnlichkeit zu berechnen siehe Abbildung 3.4. Dazu wird die Trajektorie zuerst in eine Zeichenkette umgewandelt. Dies kann basierend auf AOI oder eines einfachen Gitters bewerkstelligt werden. Um die beiden Sequenzen auszurichten wird der Needleman-Wunsch Algorithmus verwendet. Aus dem Alignment bekommt man einen Wert siehe Abbildung 3.3, dieser kann über eine Ersetzungsmatrix

	A	B	C	N	J	R	Q
A	1						
J					1		
C			1				
J					1		
N				1	1	0	0
R	1	1	0	0	0	1	0
C	0	0	1	0	0	0	0

(a) Matrix während der Ausführung

	A	B	C	N	J	R	Q
A	4	3	2	2	1	0	0
J	3	3	2	1	2	0	0
C	3	3	3	1	1	0	0
J	2	2	3	1	1	0	0
N	1	1	1	2	1	0	0
R	1	1	0	0	0	1	0
C	0	0	1	0	0	0	0

(b) Endmatrix

Abbildung 3.3: Ablauf des Needleman-Wunsch Algorithmus. Erst wurde die Matrix mit 1 bei passenden Zeichen gefüllt. Daraufhin wurde in der unteren rechten Ecke mit dem auffüllen der Felder begonnen. Der Wert im Grün markierten Feld siehe 3.3a wird aus dem Maximum der gelben Felder addiert mit dem Wert im grünen Feld gebildet. Am Ende 3.3b steht in der oberen linken Ecke das Maximum. Das beste Alignment ist dort, wo ein Pfad mit maximalen Werten liegt.

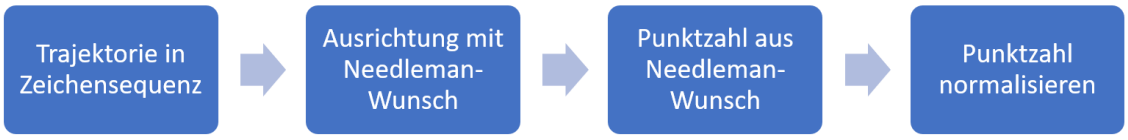


Abbildung 3.4: Ablauf von ScanMatch. Needleman-Wunsch wird in Abbildung 3.3 gezeigt.

beeinflusst werden. Diese Punktzahl wird als letzten Schritt mit 3.1 normalisiert, wobei 1 für den bestmöglichen Wert steht. Die vorgestellte Methode wurde auch mit drei Experimenten demonstriert. Im ersten Experiment mit synthethischen Daten war ScanMatch deutlich besser. In den anderen Beiden Experimenten wurden echte Daten eines Eye-Trackers verwendet um die Robustheit zu zeigen.

$$normalisierterWert = \frac{Wert}{Max(Ersetzungsmatrix) * LängederlängstenSequenz} \tag{3.1}$$

Eine weitere Methode, welche mit diskreten Werten wie Zeichen arbeitet, wird von McGuire und Chakraborty [JM16] vorgestellt. Bei dieser wird jeder Richtung ein Zeichen zugewiesen beispielsweise für Norden a, Osten b, Süden c und Westen d, siehe Abbildung 3.5. Die durch das Ersetzen der Richtung durch die Zeichen entstehende Zeichenkette kann für das Finden von Ähnlichkeiten verwendet werden. Auch wird ein Algorithmus angegeben um häufige Sequenzen zu finden.

Eine Software welche Ähnlichkeiten und Muster in Fixationssequenzen finden kann ist eyePatterns [WHRK06]. Sie kann Blicksequenzen, welche aus Zeichen bestehen, analysieren. Für die Analyse

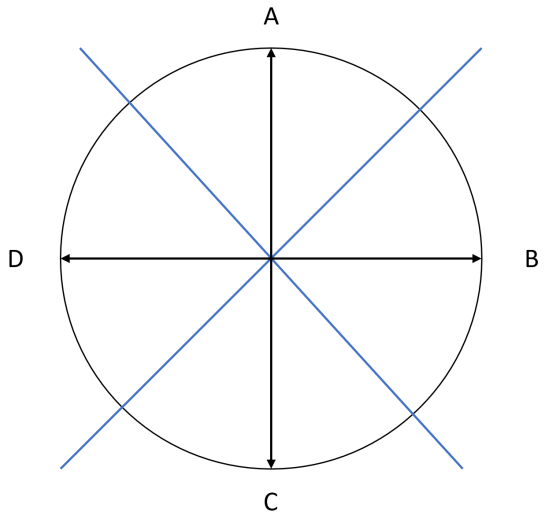


Abbildung 3.5: Zuweisen eines Zeichens basierend auf der Richtung.

der Sequenzähnlichkeit stehen der Levenshtein-Algorithmus und der Needleman-Wunsch aus einer Bibliothek zur Verfügung. Für die Visualisierung der Daten wird Prefuse [HCL05], eine Sammlung an Werkzeugen für Visualisierungen in Java, verwendet. EyePatterns bietet mehrere Methoden zum finden von Ähnlichkeiten. Eine davon ist, das lokale ausrichten von Sequenzen.

Eine weitere Methode ist FuncSim [SUF13]. Bei dieser wird der Pfad in einzelne Unterpfade geteilt basierend auf einzelnen Teilen einer Aufgabe. In dem Artikel wurden als Beispiel die Augenbewegungen während dem Speedstacking aufgezeichnet und dann mit einzelnen Teilaufgaben, die bei diesem Sport auftreten, annotiert. Der Algorithmus erstellt einen Basis-Pfad durch zufälliges Mischen der Teilaufgaben. Daraufhin werden die Pfade aligniert. Danach werden zwei Punktzahlen berechnet die mit statischen Werkzeugen weiter untersucht werden können.

3.2 Geometrisch

Larios et al. [LMKG15] vergleichen drei Entfernungsmessungsalgorithmen für Trajektorien. Die Alogrithmen sind Dynamic Time Warping [BC94] (DTW), Lowest Common Subsequence [VGK02] (LCSS) und die Fréchet-Distanz [EGH+02]. In Kapitel 2 wurde die Fréchet-Distanz kurz beschrieben. In dem Vergleich hatte die implementierte Version der Fréchet-Distanz eine ähnliche Ausführungszeit wie LCSS, wohingegen die Fréchet-Distanz Ähnlichkeit aufgrund der Form bestimmt. Als Ergebnis hatten die anderen Algorithmen schlechtere Resultate als die Fréchet-Distanz und diese ist deswegen ein gutes Maß für die Ähnlichkeit von Trajektorien.

Eine andere Möglichkeit sind auf Vektoren aufbauende Methoden, welche die Sequenz erhalten. In Jarodzka et al. [JHN10] wird eine Methode vorgestellt die eine Trajektorie mit geometrischen Vektoren darstellt. Dadurch bleiben die Form, Länge, Richtung, Positionen und Dauer erhalten. Vor dem Trajektorienvergleich werden, wie in dem Artikel beschrieben, die Pfade temporär basierend auf ihrer Form aligniert. Danach werden folgende fünf Maße berechnet:

- Differenz der Form

- Distanz zwischen Fixationen
- Unterschied der Winkel einzelner Vektoren
- Differenz der Dauer zwischen Fixationen
- Unterschied im Ausschlag der Vektoren.

Die einzelnen Werte zeigen eine Ähnlichkeit in den Unterschiedlichen Dimensionen. Das Gewichten der einzelnen Maße sollte je nach Anwendungsfall vorgenommen werden.

Eine Studie, die sich mit der Effektivität von Trajektorienähnlichkeitsmaßen beschäftigt ist von Wang et al. [WSZ+13] durchgeführt worden. Es wurden unterschiedliche Arten von Methoden verglichen bezüglich ihrem Verhalten auf Störung, Unterschiede in der Samplingrate und Verschiebungen. LCSS und PDTW haben am besten in diesem Vergleich abgeschnitten. Euclidean Distance (ED) und DTW waren sensitive zu dem Großteil der Einflüsse.

Von Buchin et al. [BBvL11] wird ein Algorithmus beschrieben der die meisten ähnlichen Subtrajektorien einer Trajektorie findet. Dabei stellen sie für gleiche Startzeiten einen Algorithmus in Linearzeit vor. Für nicht gleiche Startzeiten wird eine Approximation angegeben, wodurch der Algorithmus auch einen Zeitversatz in den Daten erlaubt, um eine Ähnlichkeit zu finden. Die Autoren schreiben, um ein gutes Clustern zu erreichen, ist es sinnvoll in die Ähnlichkeit mehrere Eigenschaften einfließen zu lassen. Auch wird angemerkt, dass bei vielen Trajektorien ein Vergleich von Paaren ineffizient ist und ein Algorithmus oder Datenstrukturen gefunden werden müssen, welche dieses Problem lösen.

Dewhurst et al. [DNJ+12] beschreiben Scanpfadvergleichsmethoden: AOI basierten Levenshtein und ScanMatch, sowie ohne AOI die Mannan Distanz und Aufmerksamkeitskarten oder Hitzekarten. Jedoch berücksichtigen diese Verfahren einen Teil der Eigenschaften der Daten nicht. Dazu zählen je nach Verfahren die Reihenfolge, Position, Form oder Dauer. Mit dem vorgeschlagenen MultiMatch werden, im Gegensatz zu ScanMatch und Levenshtein-Distanz, alle diese Eigenschaften aufgenommen. Als erster Schritt hat MultiMatch eine Vereinfachungsphase, die je nach Aufgabe unterschiedlich ist und auf die Daten angewandt werden kann. Daraufhin wird der Pfad basierend auf der Form ausgerichtet. Der nächste Schritt ist dann der Vergleich, welcher aus den fünf Dimensionen Form, Länge, Richtung, Position und Dauer besteht. Durch die einzelnen Dimensionen ist es gut möglich zu sagen in welchem Aspekt der Trajektorie der Unterschied liegt. In dem Vergleich wurde der Fokus auf die Stärken von MultiMatch gelegt und die Einstellungen von ScanMatch wurden nicht an die Aufgabe angepasst.

Zur Verwendung in dieser Arbeit scheint der MultiMatch am besten geeignet, da in diesem Trajektorien in mehreren Kategorien bewertet werden. In den anderen vorgestellten Verfahren wurde meist nur eine Eigenschaft untersucht. Gemeinsam bei mehreren ist das Ausrichten der Trajektorien zueinander, dies ist auch in MutliMatch der Fall.

3.3 Visualisierung

In den Materialwissenschaften sind Multiskalenvisualisierungen nützlich, um die Eigenschaften und Zusammensetzungen verschiedener Materialien zu untersuchen. In Nikolov et al. [NPL+10] wird dies Anhand der Struktur der Hummer-Kutikula gezeigt, siehe Abbildung 3.6. Eine Kutikula

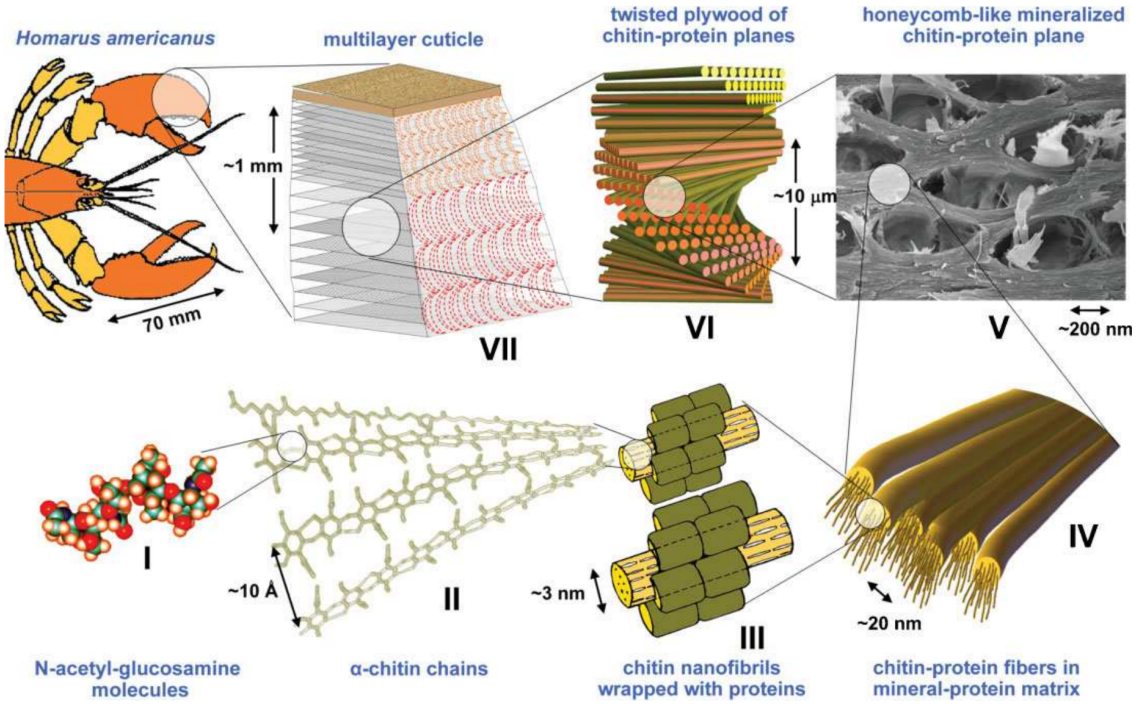


Abbildung 3.6: Die unterschiedlichen Schichten der Hummer-Kutikula, in mehreren Skalen visualisiert. Startend bei I mit Molekülen bis VII in der die äußere Sicht aufgeteilt in drei unterschiedliche Schichten dargestellt ist. Entnommen aus [NPL+10]

ist in der Biologie eine Schicht, welche je nach Spezies auf unterschiedliche Weise zum Schutz dient. Der Artikel untersucht die Zusammensetzung dieser Schicht bei Hummern, in unterschiedlichen Skalen. So besteht diese im detailreichsten Level aus Molekülen, wenn man ein Abstraktionslevel höher geht sieht man die Schicht der Chitinketten. Durch die unterschiedlichen Abstraktionen, sind unterschiedliche Details des Materials sichtbar. So besteht die Zusammensetzung aus mehreren Lagen und diese wiederum aus Chitin-Protein Fasern.

Eine weitere Visualisierung wäre Edge Bundling von Holten und van Wijk [HW09]. Hier werden Knoten-Kanten Diagramme untersucht, welche aus vielen Knoten und Kanten bestehen, wodurch eine visuelle Unordnung entsteht. Dies kann durch Edge Bundling verringert werden. In dem Artikel sind die Kanten als flexible Federn, welche sich anziehen können modelliert. Das Edge Bundling ist in Abbildung 3.7 im Vergleich zu einen normalen Knoten-Kanten Diagramm zu sehen.

Bei Visualisierungen mit vielen Daten werden diese meist schnell unübersichtlich. Dieses Problem wird von Furnas und Bederson in [FB95] untersucht. Dabei werden für die gewünschten Zoomstufen eines Bildes Kopien angelegt und die Daten darauf abgebildet in der gewünschten Auflösung. Es entsteht eine Pyramide mit den einzelnen Abstraktionen wie in Abbildung 3.8.

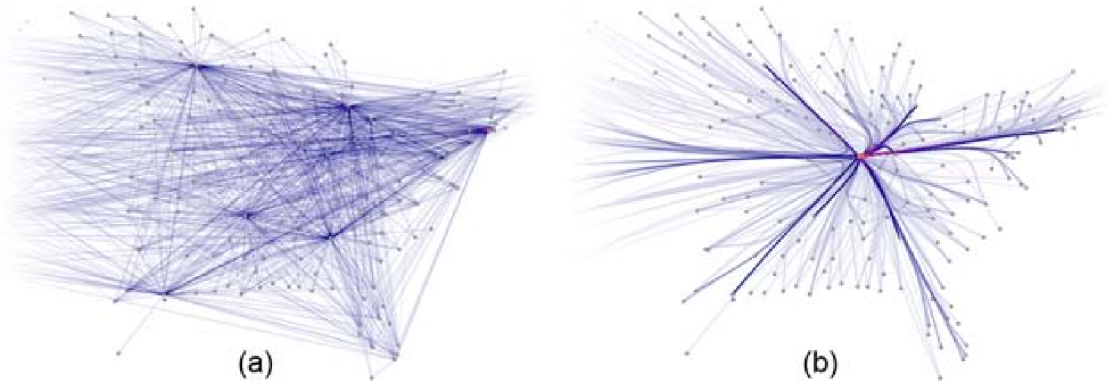


Abbildung 3.7: a) zeigt ein Knoten-Kanten Diagramm, in b) wurden die Kanten mit Edge Bundling zusammengefasst. Es ist deutlich weniger visuelle Unordnung vorhanden. Entnommen aus [HW09] Seite 4

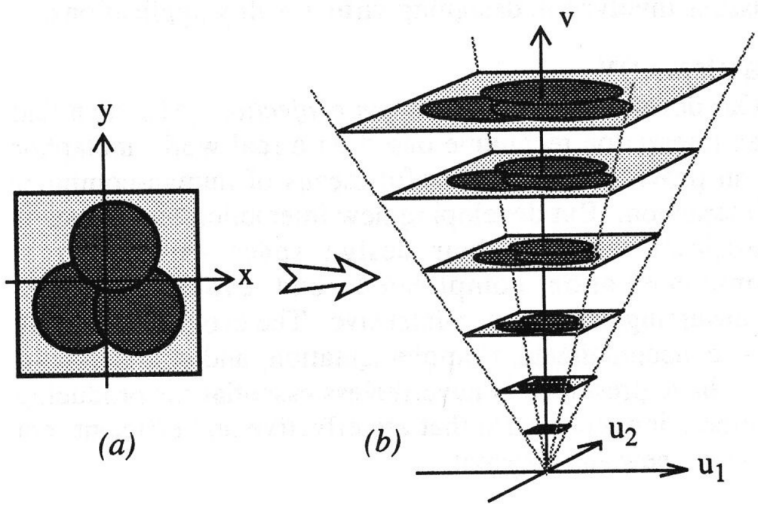


Abbildung 3.8: Aus einem Bild a) werden mehrere Ebenen an Details erstellt, diese können in einer Pyramide b) gestapelt werden. Entnommen aus [FB95]

4 Anforderungen

In Kapitel 3 wurden unterschiedliche Arbeiten betrachtet, welche sich mit der Analyse von Trajektorien beschäftigen. Da ein Großteil der Systeme Trajektorien basierend auf Zeichenketten vergleicht, soll hier ein anderer Ansatz verwendet werden. Als Visualisierung wurde in keinem der Artikel eine Abstraktion der Pfade verwendet. Eine Anforderungen an ein neues Werkzeug zur Visualisierung von Trajektorien war die Möglichkeit unterschiedliche Grade der Abstraktion auf Basis eines Gitters zu haben. Diese Art der Abstraktion ist eine Kombination der Technik aus der Literatur Nikolov et al. [NPL+10] und Furnas und Bederson [FB95], mit denen Strukturen auf unterschiedlichen Skalen hervorgehoben werden können. Edge Bundling 3.3 hätte auch dazu verwendet werden können, ist jedoch rechenintensiv und wird daher in dieser Arbeit nicht verwendet. Meistens werden Trajektorien auf unterschiedliche Arten analysiert: durch Abstraktion, (semi-) automatisch durch Clustering oder eine tiefer gehende Analyse. In der tiefer gehenden Analyse werden Daten manuell, im Detail, analysiert. Diese Arbeitsschritte wurden in einen Workflow abgeleitet und in Abbildung 4.1 dargestellt. Für diesen Workflow werden unterschiedliche Funktionen, wie Ansichten und Interaktionen benötigt.

Es werden insgesamt drei Ansichten erstellt, für jeden Schritt eine. In den Ansichten werden mehrere Gui-Elemente benötigt. Jede Ansicht sollte ein Element zur Darstellung der Trajektorien in unterschiedlichen Abstraktionen beinhalten. Zur Interaktionen mit dem Programm existieren mehrere Auswahllisten, in welchen Elemente an- und abgewählt werden können. Dadurch wird ein Filter für die Daten erstellt, welcher die Daten im Workflow einschränkt. Zur Eingabe von Parametern für Strukturen und Algorithmen die angewandt werden, werden Eingabefelder zur Verfügung gestellt. Des Weiteren sollte das System die Möglichkeit haben mit größeren Datenmengen umzugehen. Eine weitere Funktion ist eine Art Glätten im manuellen Detailvergleich, durch Entfernen von Fixationen aus der Trajektorie, um eine höhere Ähnlichkeit zu erhalten, basierend auf einer Referenztrajektorie. Für die manuelle Analyse zeigt ein Histogramm die Ähnlichkeit, falls die Fixation an diesem Index entfernt wird. Es bietet Möglichkeit zum Auswählen einer oder mehrerer Fixationen, für das Entferne aus der Trajektorie.

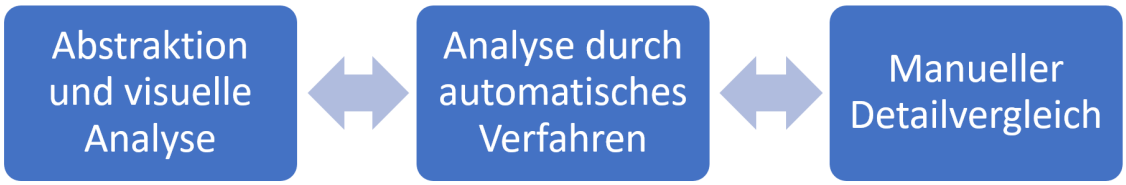


Abbildung 4.1: Analyse Workflow, angefangen mit Abstraktion und visueller Analyse. Gefolgt von einer Analyse durch ein automatisches Verfahren. Am Schluss ein manueller Detailvergleich an zwei Trajektorien.

So ist folgender Analyseworkflow sinnvoll. Als erstes können die kompletten Daten in unterschiedlichen Granulalitäten angezeigt werden. Anschließend werden ausgewählte Daten maschinell analysiert werden, um vorher nicht sichtbare Erkenntnisse zu erlangen. Als dritten Schritt können zwei Trajektorien genau miteinander verglichen werden. In diesem Schritt soll es auch möglich sein aus der Trajektorie einzelne Fixationen zu entfernen um Beobachten zu können wie die Werte sich im Vergleich verändern. Ausreißer sollen durch diese Möglichkeiten weniger Einfluss auf die Analyse nehmen können.

5 System

In diesem Kapitel wird das aus den erfassten Anforderungen heraus entstandene System beschrieben. Während der Planungsphase haben sich zusätzliche Anforderungen hinsichtlich Modularität und Wiederverwendbarkeit ergeben. Die Implementierung erfolgte unter Zuhilfenahme der in Kapitel 2 beschriebenen Technologien. Das webbasierte System, auf Basis von Angular 4, beinhaltet drei Ansichten um den, in den Anforderungen erstellten Workflow umzusetzen. Die einzelnen Ansichten, sowie die verwendeten Datenstrukturen werden nun im Detail erläutert.

5.1 Datenstruktur

Als Datentrukturen kommt eine Klasse Trajectory, gezeigt in Listing 5.1, zum Einsatz, in die jede einzelne Trajektorie eingefügt wird.

Die Klasse spiegelt die Attribute aus den Eingabedaten wieder. Diese hatten die Spalten:

- Timestamp
- Stimuliname
- Fixationindex
- FixationDuration
- Mappedfixationpointx
- Mappedfixationpointy
- User
- Description

Listing 5.1 Trajektorien Datenstruktur

```
1  class Trajectory {
2  stimulus: string;
3  participant: string;
4  color: string;
5  points: Array<PointType>;
6  qTree: QTree;
7  constructor() {}
8  genQtree(maxHeight: number, maxWidth: number, minHalfDimension) {}
9  }
```

Listing 5.2 Datenstruktur für Punkte

```
class PointType {
x: number;
y: number;
duration?: number;
timestamp?: number;
index?: number;
}
```

Listing 5.3 Datenstruktur für Quadtree

```
class QTree {
boundary: AABB;
minHalfDimension: number;
currentLevel: number;
// Points in this quad tree node
points: Array<PointType> = [];

// Children
northWest: QTree;
northEast: QTree;
southEast: QTree;
southWest: QTree;
getDepth() {}
getPoints(): PointType[] {}
getCleanPointsForLevel(level: number) {}
queryRangeTrajectory(range: AABB) {}
queryRange(range: AABB) {}
}
```

Auf diese Weise importiert es Daten eines Eye-Trackers. Die Auflösung, der einzelnen Stimuli, wurde aus einer Datei mit den Spalten city, width und height eingelesen. Beim Einlesen werden die Punkte in ein Array mit dem Datentyp PointType, wie in Listing 5.2 gezeigt, eingefügt, welches in der Klasse Trajektorie vorhanden ist.

Um die Multiskalenvisualisierung zu erreichen, wird ein Quadtree für jede Trajektorie gespeichert. Er hat die in Listing 5.3 gezeigte Struktur.

Es gibt eine Methode um die maximale Tiefe des Baumes, getDepth(), zu erhalten und eine Methode, getPoints(), um die Punkte des aktuellen Knotens zu erhalten. Die Methode getPoints(), führt eine Bereichsanfrage mit den Grenzen des Knotens aus, in welchen sie aufgerufen wird. Um Zugriff auf die einzelnen Detailgrade zu bekommen, steht die Methode getCleanPointsForLevel(level:number) bereit. Diese kombiniert mehrere Methoden. Falls das angefragten Level erreicht ist, werden mit der Methode getPoints() die Punkte zurückgeben. Falls das angefragte Level nicht erreicht ist, werden die vier Kindknoten rekursiv aufgerufen. Bevor die Punkte zurück gegeben werden, wird die Reihenfolge überprüft, um zu erkennen ob ein Wechsel des Quadranten stattgefunden hat und somit eine Teilsequenz erstellt werden muss. Der Wechsel ist in Abbildung 5.1 dargestellt und die betroffenen Fixationen markiert.

Die benötigten Daten werden im Programm von Services bereitgestellt. Ein Datenservice, welcher die Trajektorien bereitstellt und ein Gui-Service, um die ausgewählten Optionen der Oberfläche über die einzelnen Sichtweisen zu speichern, wurden implementiert. Die Trajektorien werden aktuell aus einer Datei, welche mit Tabs als Trennzeichen arbeitet gelesen. Solange das Format

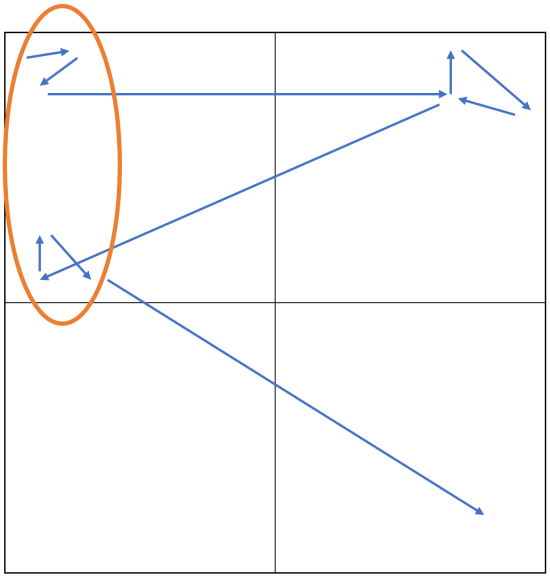


Abbildung 5.1: Erstellung einer Teilsequenz bei Wechsel des Quadranten bei Anfrage eines Abstraktionslevels. Die betroffenen Fixationen sind durch die Ellipse markiert.

eingehalten wird können die Daten beliebig ausgetauscht werden. Die Daten für die Auflösung der Stimuli war auch, wie die Trajektorien, in einer Datei mit Tabs als Trennzeichen vorhanden. Über das Utils Modul werden Hilfsfunktionen zur Farbauswahl, das Berechnen eines Zentrum einer Trajektorie und weitere Funktionen, als auch der MultiMatch Algorithmus bereitgestellt.

5.2 Trajectory View

Der Trajectory View in Abbildung 5.2, besteht aus den Gui-Elementen MultiSelect, in der Abbildung gekennzeichnet mit C und D, der Visualisierung für Trajektorien markiert mit E, sowie einer Checkbox in Verbindung mit einem Slider markiert mit B um den Quadtree anzuzeigen und das Level der Abstraktion einzustellen. Über das Element markiert mit A kann der Quadtree, mit einer anderen minimalen Gittergröße, neu erstellt werden. Die Visualisierung der Trajektorien wird, wie auch das Balkendiagramm, mit D3.js gemacht, was ein SVG-Element produziert. Der Start der Trajektorie wird durch einen schwarzen Punkt gekennzeichnet, die Farbe anhand des Namens oder Clusters gewählt. Das Element zur Visualisierung ist ein eigenes Element und ihm kann in den anderen Ansichten einfach Daten zur Anzeige übergeben werden. Die initiale minimale Größe pro Quad ist 20 Pixel. Dabei ist Null das oberste Level mit der größten Abstraktion in welchem die Trajektorien nur als Punkt Sichtbar sind. Je Tiefer in den Quadtree vorgedrungen wird, desto mehr Details der Trajektorie werden sichtbar. Durch Eingabe einer minimalen Quadtreegröße in Pixel und anschließendem Klick auf „Gen Tree“ wird der Quadtree basierend auf den Einstellungen neu aufgebaut. Über die Auswahl eines Stimuli wird ein generelles Muster sichtbar, welches die Teilnehmer verwendet haben, um die Aufgabe zu lösen (siehe Abbildung 5.3a). Wenn einzelne Teilnehmer selektiert werden, wird der Filter weiter eingeschränkt. Falls keiner ausgewählt wurde, werden alle Teilnehmer der gewählten Stimuli angezeigt. Auch ist es möglich alle Trajektorien

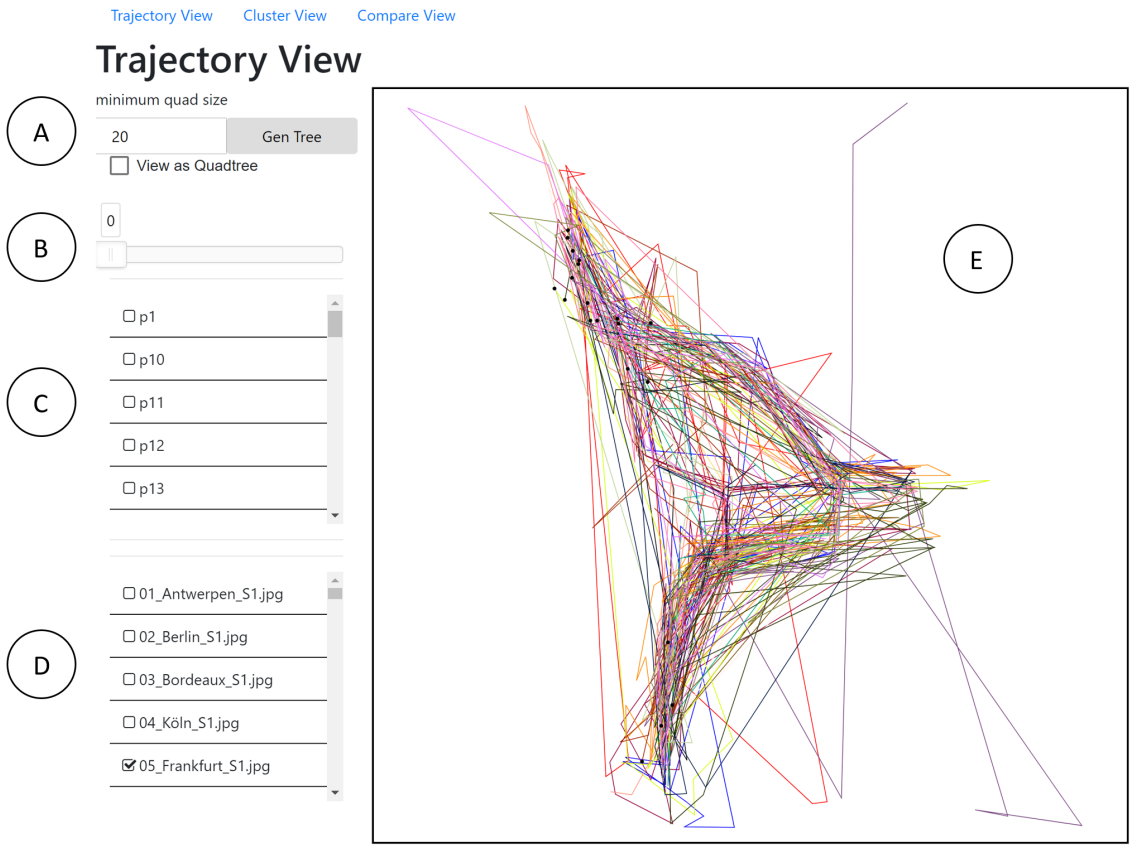
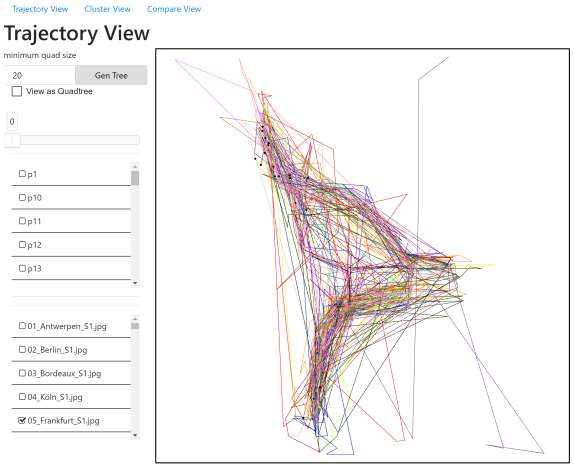


Abbildung 5.2: Trajectory View, A: Einstellung für Quadtree, B: Slider für Level des Quadtree, C: Auswählbare Teilnehmer, D: Auswählbare Stimuli, E: Visualisierung der Auswahl

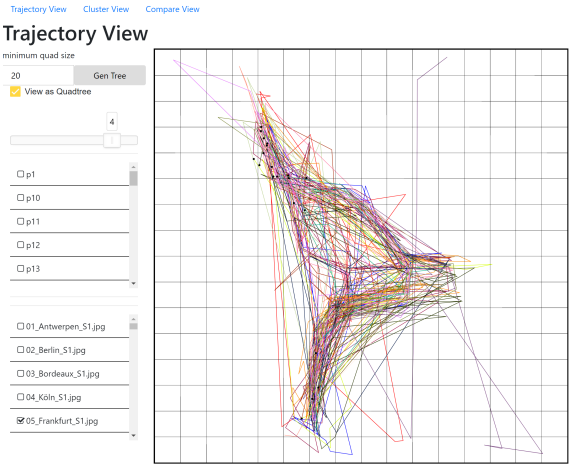
eines Teilnehmers zu sehen, falls kein Stimulus ausgewählt wird. Wenn durch den Slider in Abbildung 5.2, markiert mit B, ein Level der Abstraktion ausgewählt und der Haken für den Quadtree gesetzt ist werden die Daten auf dem Level abstrahiert angezeigt. Wenn das Level 0 ausgewählt wurde, entsteht eine Ansicht wie in Abbildung 5.3f in der die ausgewählten Trajektorien auf ein Punktezentrum reduziert wurden. Für Level 1 wird die zweidimensionale Fläche in 4 Quadranten unterteilt (Abbildung 5.3e) und die Trajektorien auf diese vier Bereiche reduziert. Weitere feinere Gitter sind in den Abbildungen Abbildung 5.3d, Abbildung 5.3c und Abbildung 5.3b zu sehen, je feiner das Gitter desto besser ist die ursprüngliche Form (Abbildung 5.3a) erkennbar.

5.3 Cluster View

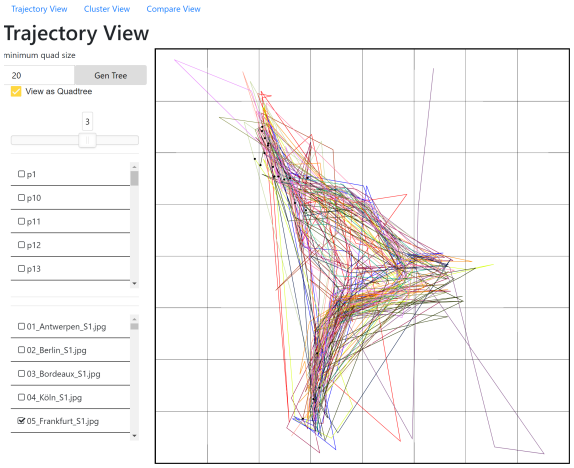
Im Cluster View, siehe Abbildung 5.4, sind die Gui-Elemente MultiSelect, die Visualisierung für Trajektorien, Elemente für die Einstellungen des Quadtree, sowie Eingabefelder für die Anzahl der Custer und Anzahl der Durchläufe vorhanden. Die oberen zwei Multiselektionen enthalten einerseits eine Liste der Teilnehmer und andererseits die einzelnen Stimuli. Durch die Auswahl von Teilnehmern und Stimuli in der Selektion und anschließendem drücken des Clusterknopfes wird das Clustering durchgeführt. Sobald das Clustern fertig ist, werden diese in einer Liste angezeigt. Durch Selektion eines Clusters werden die Trajektorien darin visualisiert und in der



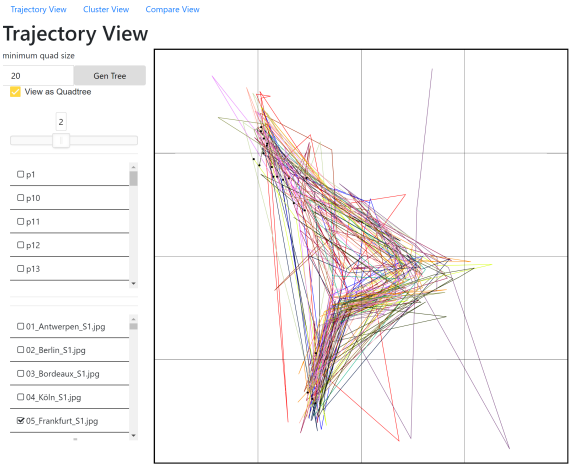
(a) Anfangsdaten



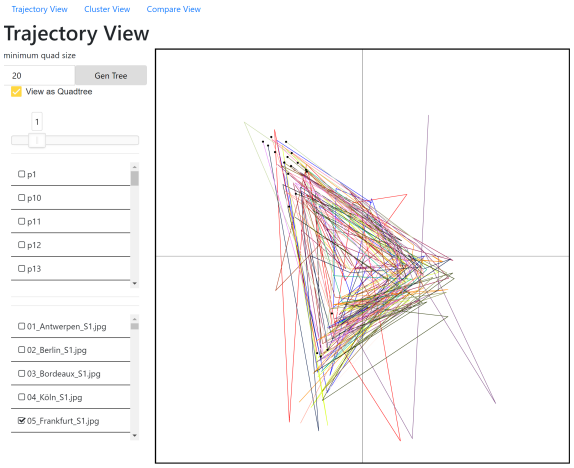
(b) Level 4: Abstrahiert in 16x16 Grid



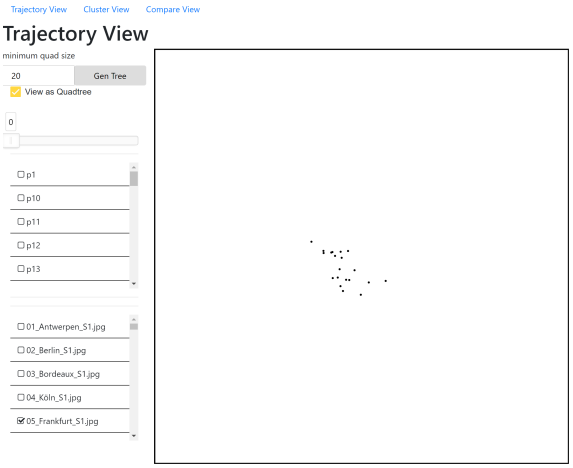
(c) Level 3: Abstrahiert in 8x8 Grid



(d) Level 2: Abstrahiert in 4x4 Grid



(e) Level 1: Abstrahiert in 2x2 Grid



(f) Level 0: Abstrahiert auf Punkte

Abbildung 5.3: Einzelne Abstraktionslevel der Trajektorien

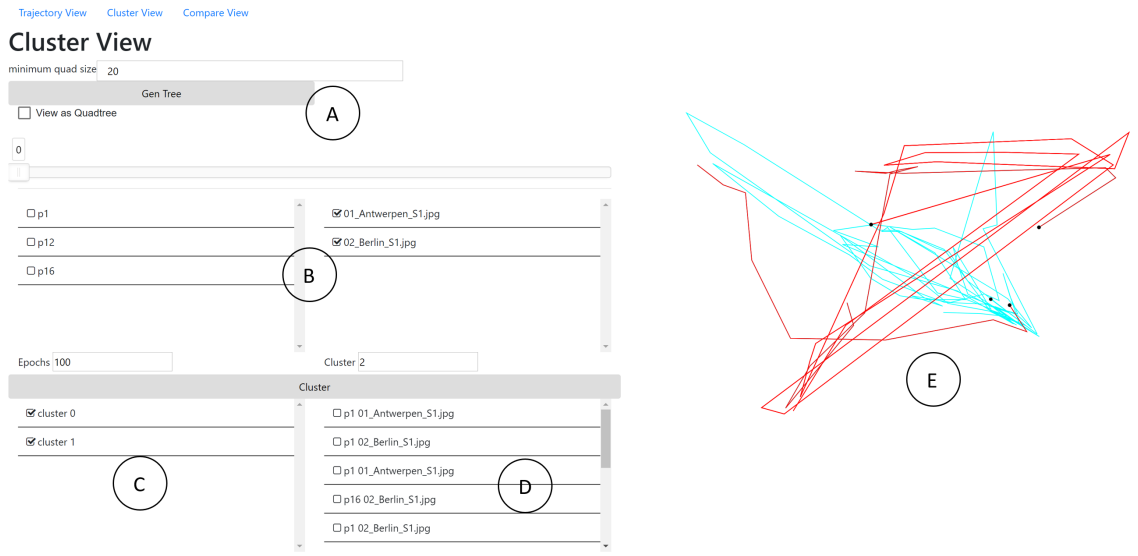


Abbildung 5.4: Cluster View, A: Einstellung für Quadtree, B: Slider für Level des Quadtree, C: Auswählbare Teilnehmer und Stimuli, D: Enthaltene Trajektorien in den Ausgewählten Clustern E: Visualisierung der Auswahl.

Liste daneben angezeigt. Durch auswählen einer Trajektorie wird nur diese visualisiert. Die Werte für das Clustering werden mit MultiMatch für Paare von Trajektorien berechnet. Das Clustering bildet aus diesen fünf Dimensionen, was die einzelnen Kategorien von MultiMatch sind, die Cluster. Diese Cluster sind so aufgebaut, dass ähnliche Paare in einem Cluster landen. So entstehen Cluster sortiert nach der Ähnlichkeit. Beispielsweise beinhaltet Cluster eins sehr ähnliche Trajektorienpaare und in weiteren Clustern nimmt die Ähnlichkeit ab. Dadurch, dass die Cluster anhand der Paare gebildet werden, herrscht keine Transitivität zwischen den Paaren des Clusterings. Dabei ist MultiMatch auf diese Weise keine perfekte Metrik um die Trajektorien zu Analysieren.

5.4 Compare View

Im Compare View (Abbildung 5.5) stehen die ausgewählten Trajektorien aus den anderen beiden Ansichten zur Verfügung, um zwei Trajektorien direkt miteinander zu vergleichen. In dieser Ansicht kommen die Gui-Elemente Histogramm-Slider (Abbildung 5.6) und Visualisierung für Trajektorien zum Einsatz.

Der Histogramm-Slider besteht aus einem Histogramm und einem Schieberegler (Abbildung 5.6), hier kann über den Regler der Index eines Balken oder ein Bereich an Indizes ausgewählt werden. Dieses Element bekommt, wie auch schon die Visualisierung, die Daten zur Anzeige übergeben. Es existieren in dieser Ansicht zwei Instanzen des Histogramm-Sliders. Der Vergleich ist ein iterativer Prozess, indem eine oder mehrere Fixationen aus einer Trajektorie entfernt werden und mit einer Referenztrajektorie verglichen wird. Um einen Vergleichswert zu erhalten wird MulitMatch [DNJ+12] verwendet. Diese Werte werden als Histogramm, für beide Trajektorien, visualisiert, wie es in Abbildung 5.5 dargestellt ist. Die x-Achse entspricht dem Index der Fixation

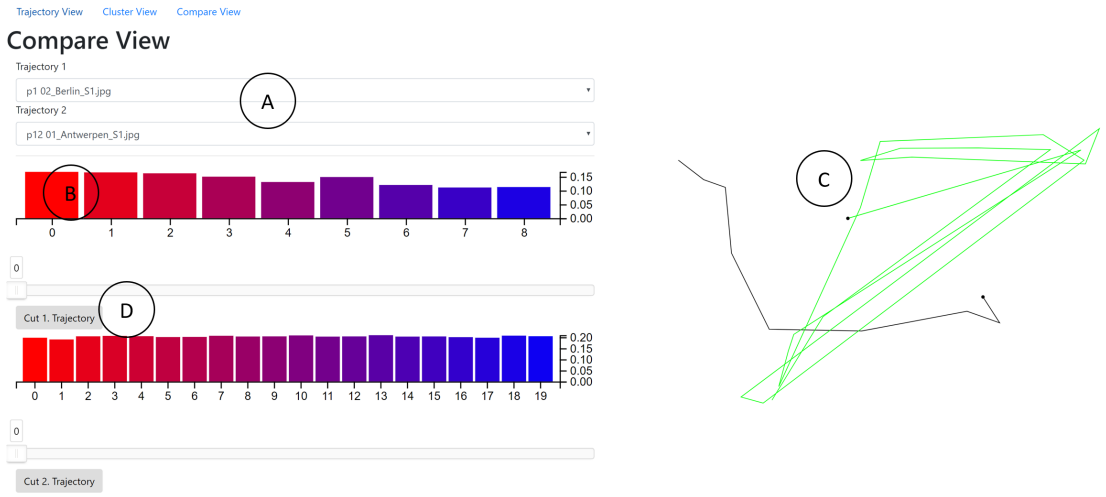


Abbildung 5.5: Compare View, A: Auswahl der 2 zu vergleichenden Trajektorien, B: HIstogramm-Slider mit Score für jede entfernte Fixation, C: Visualisierung der Trajektorie, D: Slider zum Auswählen was entfernt werden soll.

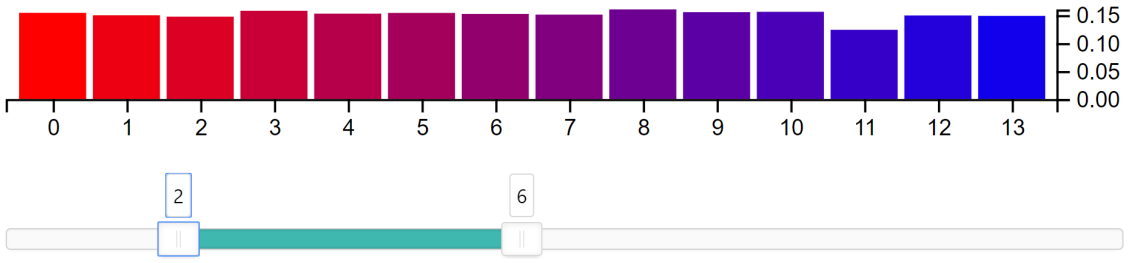


Abbildung 5.6: Histogramm-Slider

der Trajektorie, die y-Achse stellt den MulitMatch-Wert dar. Je niedriger der Balken ist desto ähnlicher, ist laut MultiMatch, die Trajektorie zu der Referenztrajektorie. Das Ergebnis ist zwischen null und eins für jede entfernte Fixation. Ein Nutzer kann über den Slider unter dem Histogramm eine Fixation auswählen und über den „Cut Trajectory Schaltfläche“ diese heraus schneiden. Dieser Vorgang kann quasi beliebig of wiederholt werden. Auf diese Weise ist es möglich in mehreren Iterationen zu versuchen beide Trajectorien zu einer höheren Übereinstimmung anzugleichen.

6 Use-Case

Dieser Abschnitt wendet den in Kapitel 4 vorgestellten Workflow erst theoretisch und nachfolgend an einem Beispiel mit den Städten Berlin und Antwerpen an. Ein Use-Case, der die implementierten Ansichten verwendet, sieht folgendermaßen aus. Als erstes verschafft sich der Nutzer im Trajektory View, eine Darstellung wie in Abbildung 5.2, einen Überlick über die einzelnen Trajektorien und Stimuli, indem er unterschiedlich viele Stimuli und Teilnehmer auswählt. Die Daten können abstrahiert werden durch den in Kapitel 5 beschriebenen Quadtree. Dieser wird über die Checkbox aktiviert. Über den zugehörigem Slider kann der Detailgrad verändert werden. Wenn man einem Überblick über die Trajektorien hat wechselt man in den Cluster View, siehe Abbildung 5.4. Hier kann man die Auswahl aus dem Trajectory View beibehalten oder ändern um dann ein Clustering zu starten. Dieses gibt einem nach Beendigung Paare an Trajektorien zugeordnet in die gewünschte Anzahl an Cluster. Die Cluster können durch Auswählen näher betrachtet werden. Die Liste der Trajektorien, welche in dem ausgewählten Cluster vorhanden sind, wird neben der Liste der Cluster angezeigt. Im Compare View können, nach der maschinellen Analyse, zwei einzelne Trajektorien genau miteinander verglichen werden. Dabei wählt man diese aus und es wird ein Histogramm generiert, welches die Ähnlichkeit anzeigt wenn eine Fixation an der entsprechenden Stelle entfernt wird. Über den Slider können dann ausgewählte zusammenhängende Fixationen durch den Knopf (Abbildung 5.5 gekennzeichnet mit D) darunter entfernt werden, woraufhin die Visualisierung aktualisiert wird.

Als Beispiel, wird dieser beschriebene Use-Case, mit Daten der Stimuli Antwerpen und Berlin durchgeführt, dies wird in Abbildung 6.1 gezeigt. Es wurden nur die beiden genannten Stimuli ausgewählt, um keine Einschränkung auf die einzelnen Teilnehmer zu erhalten. In der Visualisierung sind zwei generelle Muster zu erkennen. Durch eine Reduzierung der Details wie in Abbildung 6.2 zu vier Quadranten, tritt dieses Muster besser hervor. Wahrgenommen werden können zwei Dreiecke mit unterschiedlicher Ausrichtung. Es gibt leichte Abweichen in Fixationen zwischen den Teilnehmern. Wenn die Details um einen Schritt erhöht werden, sodass ein Gitter von 4x4 die Grundlage bildet, ist die Ursprüngliche Struktur der Trajektorie wieder besser zu erkennen. Dies ist in Abbildung 6.3 dargestellt.

Als nächsten Schritt wird in den Cluster View gewechselt. Hier können die Daten aus der vorherigen Ansicht beibehalten oder durch den Filter die Auswahl bearbeitet werden. Für ein schnelles Beispiel wurden die Teilnehmer p1 und p12, sowie die Stimuli „01_Antwerpen_S1.jpg“ und „02_Berlin_S1.jpg“ ausgewählt. Dies ist in Abbildung 6.4, durch in den Listen markierte Elemente, zu sehen. Durch Betätigen der Clusterschaltfläche werden die Daten, falls die Standarteinstellungen beibehalten wurden, in zwei Cluster aufgeteilt. Der erste Cluster, der in Abbildung 6.4 zu sehen ist, zeigt die Trajektorien für die Paare, welche in diesem enthalten sind. Im zweiten Cluster sind die Trajektorien der Paare zu sehen welche in diesen eingeordnet wurden, dies zeigt Abbildung 6.5. Die zwei Trajektorien im ersten Cluster lagen schon sehr gut übereinander, diese können im Compare View ausgewählt werden. Niedrige Werte im Histogramm (Abbildung 6.6) bedeuten

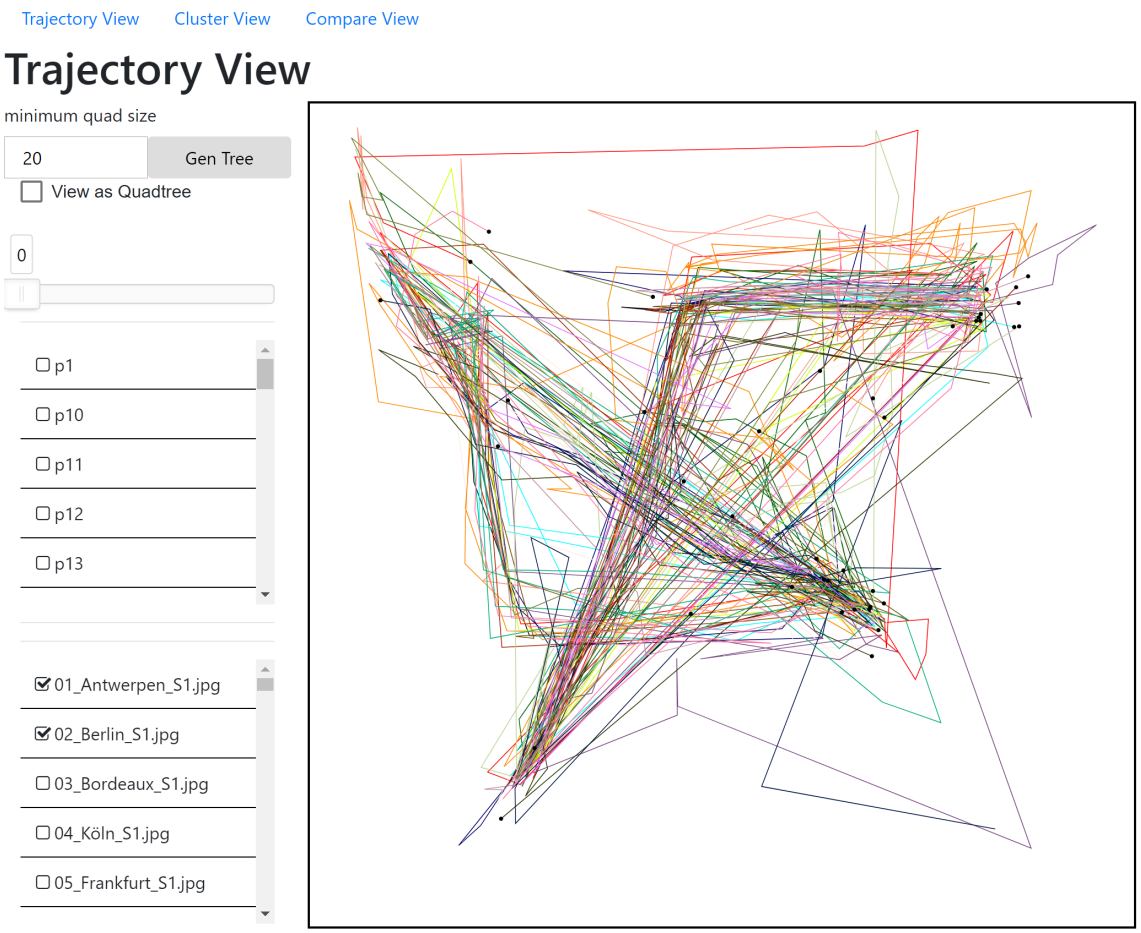


Abbildung 6.1: Im Ersten Schritt wurden die Daten von Berlin und Antwerpen ausgewählt um sich einen Überblick zu verschaffen.

eine bessere Ähnlichkeit, falls diese Fixation entfernt wird. Wir entfernen aus der zweiten Trajektorie die Fixation mit dem sechsten Index und aus der ersten Trajectorie die erste Fixation (Abbildung 6.7). Dies kann so lange wiederholt werden, bis keine Fixationen mehr übrig sind. Die Veränderung, dass die visualisierten Trajektorien sich angeglichen haben, ist nach dem Entfernen von zwei Fixationen sichtbar (in Abbildung 6.6, Abbildung 6.7 markiert durch Ellipse).

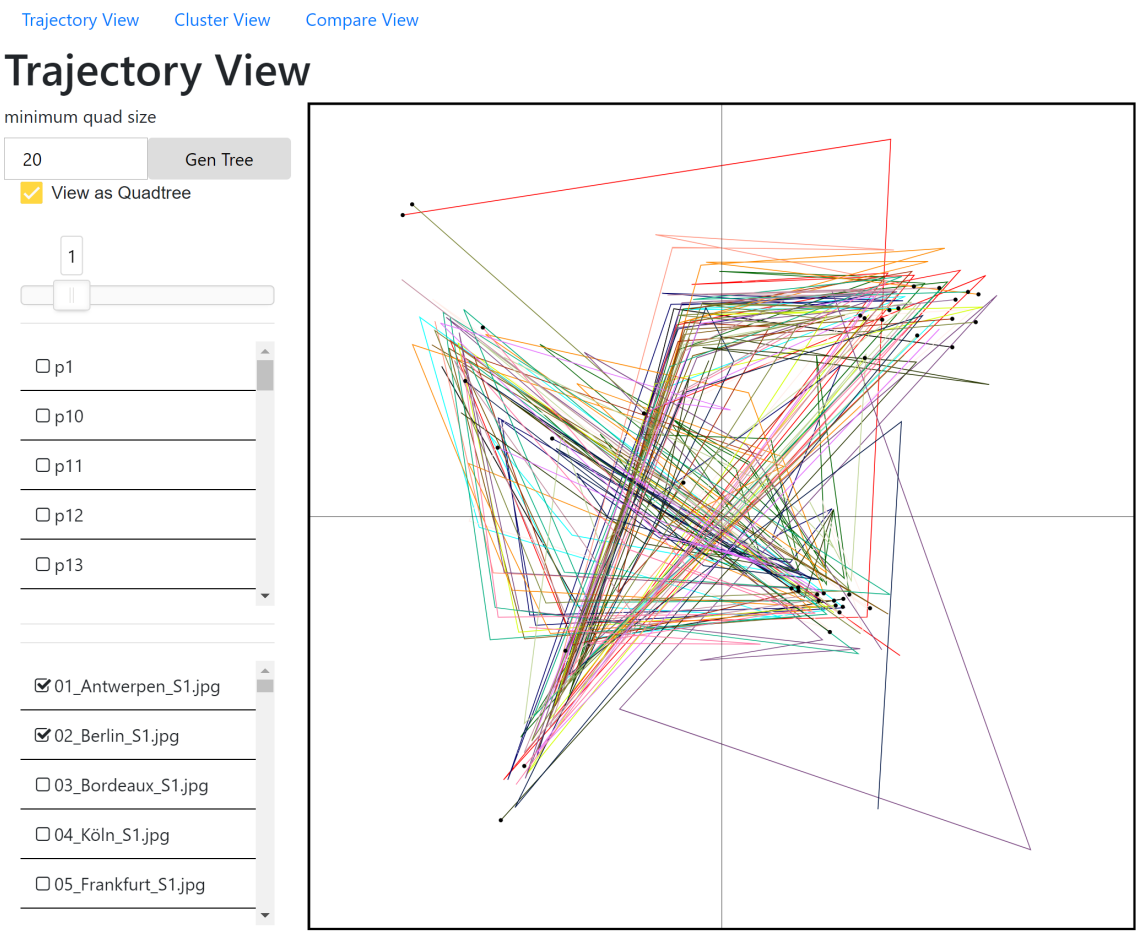


Abbildung 6.2: Als zweites wurde der Detailgrad auf das Level 1 reduziert, dadurch ist besser erkennbar in welchen Bereichen sich die einzelnen Teilnehmer sich bewegt haben

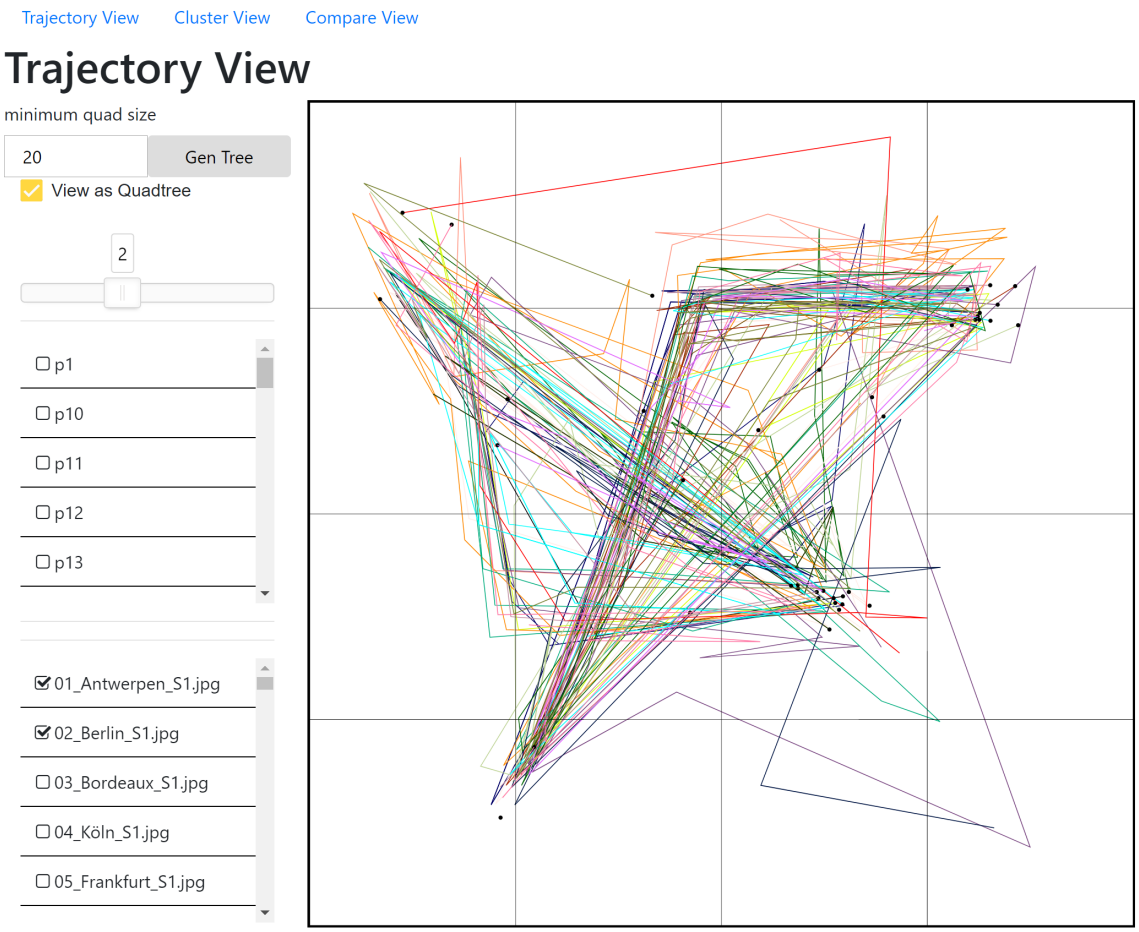


Abbildung 6.3: Der Detailgrad wurde auf das Level 2 erhöht. Hier sind die Bewegungen schon deutlich detaillierter.

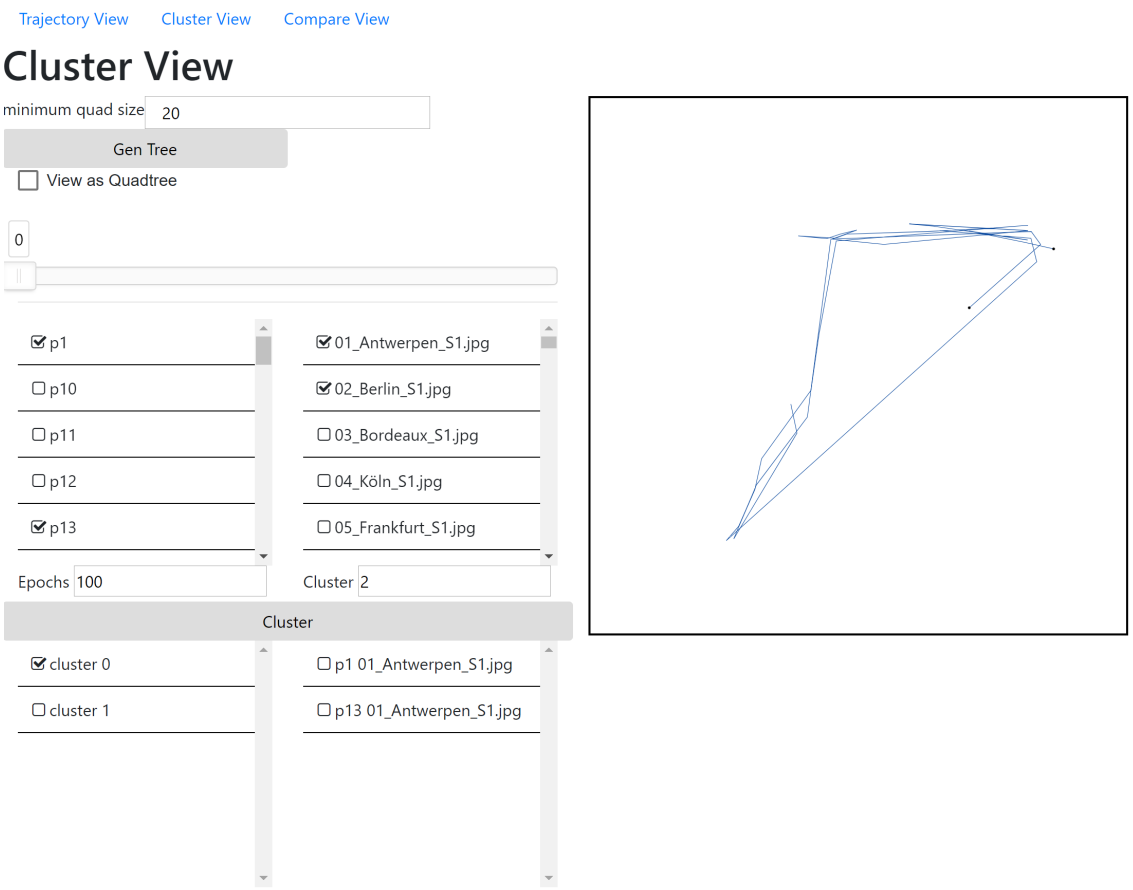


Abbildung 6.4: Die Auswahl wurde zusätzlich reduziert auf zwei Teilnehmer sodass nur vier Trajektorien für das Clustering bereit stehen. Der erste Cluster enthält ein paar aus zwei Trajektorien die visualisiert wurden.

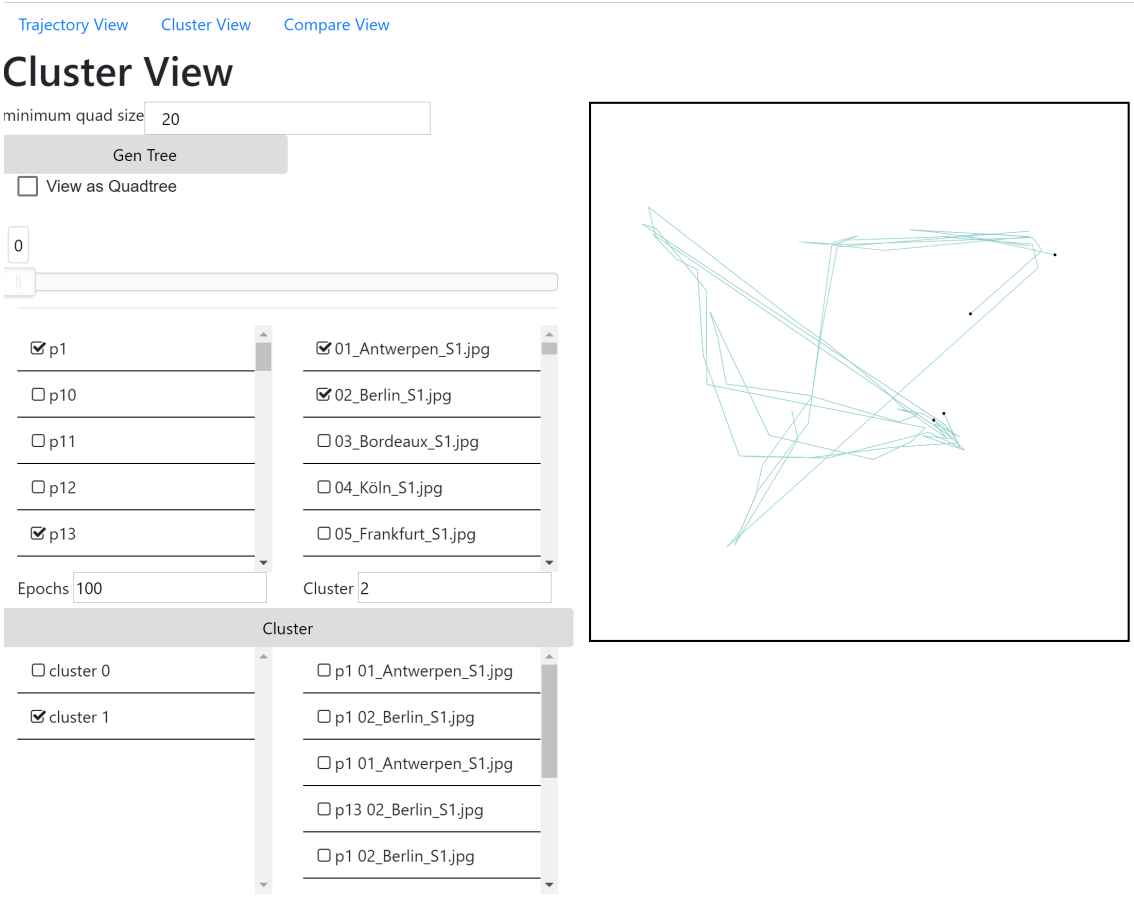


Abbildung 6.5: Selbe Auswahl wie in 6.4, der zweite Cluster wurde visualisiert. Das doppelte vorkommen der Trajektorien ist dem Vergleich von Paaren geschuldet.

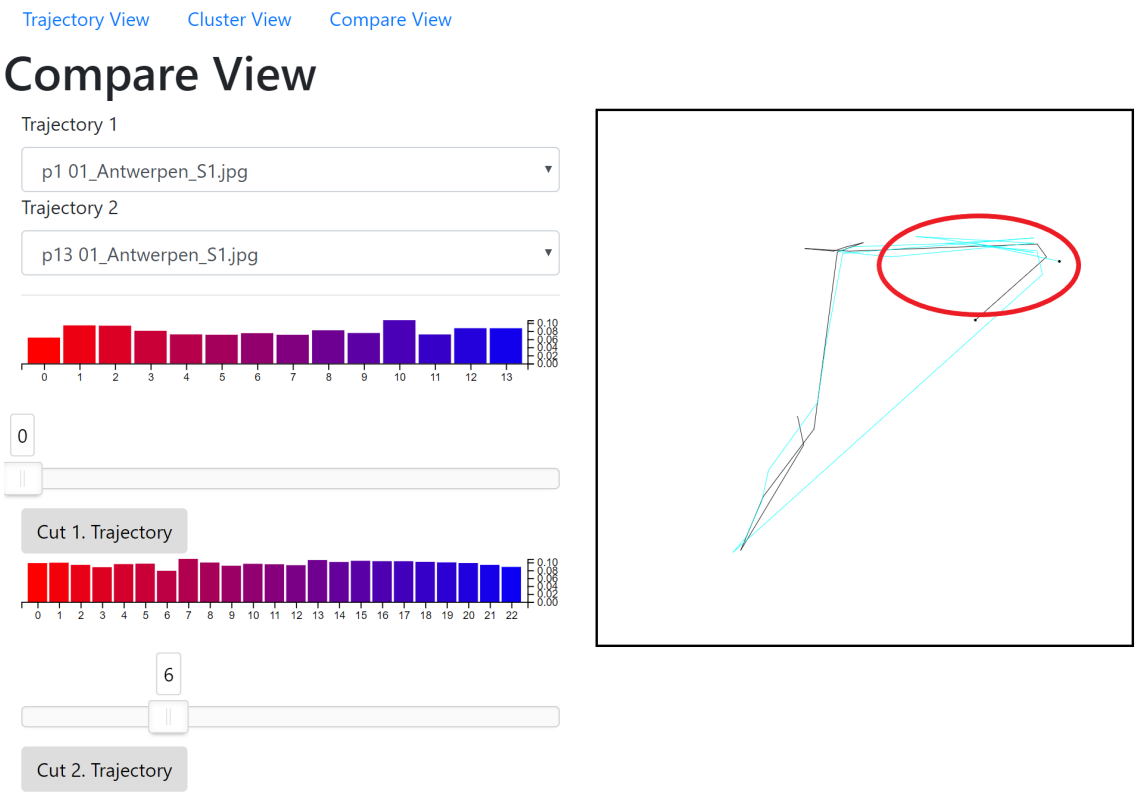


Abbildung 6.6: Es wurden zwei Trajektorien für den direkten Vergleich ausgewählt. Das obere Histogramm zeigt die Ähnlichkeit nach entfernen der Fixation aus der ersten Trajektorie (schwarz) mit dem Index des Balkens, genauso wie für das untere Histogramm wo aus der zweiten Trajektorie (türkis) die Fixationen entfernt wurden. Eine Veränderung im Vergleich zu Abbildung 6.7 ist durch die rote Ellipse markiert.

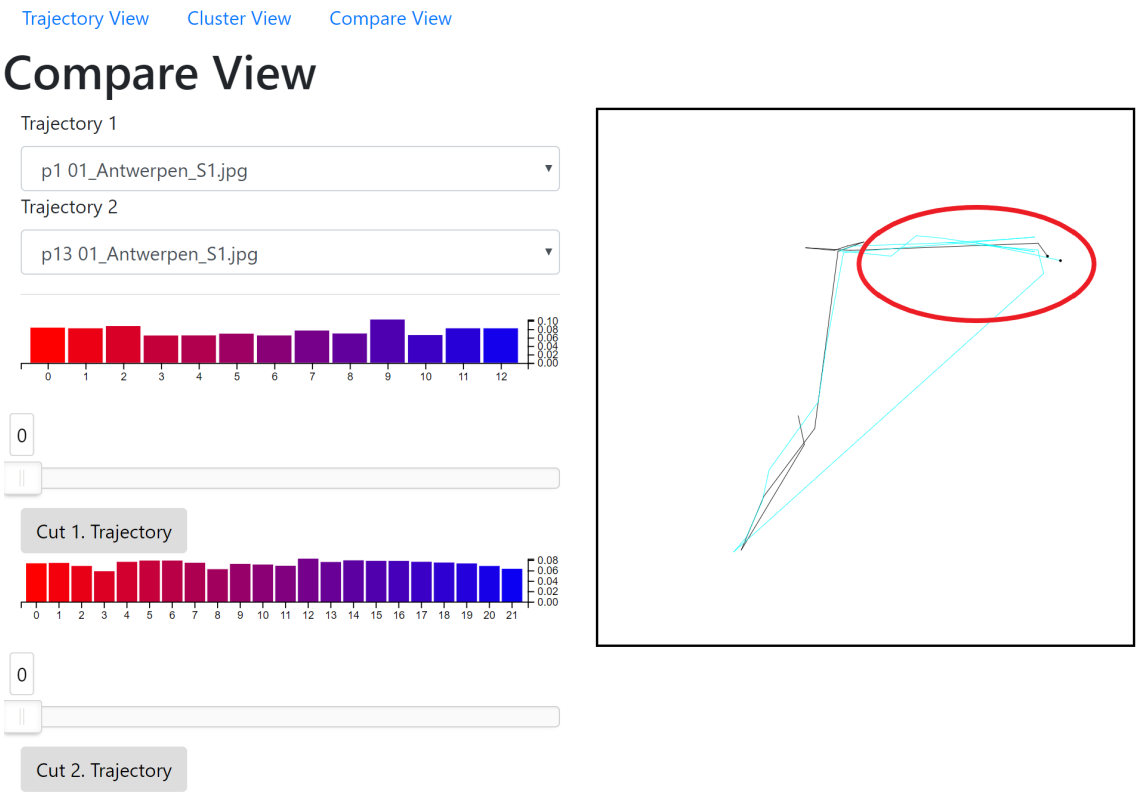


Abbildung 6.7: Aus der ersten Trajektorie (schwarz) wurde die erste Fixation entfernt und die Startpunkte liegen nah beieinander. Durch entfernen der siebten Fixation (Index 6) hat sich der waagerechte Verlauf der türkisfarbenen Trajektorie im Abbildung 6.6 verändert (rote Ellipse).

7 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Prototyp für die Multiskalenvisualisierung von Trajektorien entwickelt. Als Grundlage wurden aktuelle Webtechnologien wie Angular und D3.js zur Visualisierung verwendet. Zur Analyse wurde ein Clustering in Verbindung mit MultiMatch getestet, um einen Wert für das Clustering zu errechnen. Es wurden mit Levenshtein-Distanz und Fréchet-Distanz zwei weitere Verfahren kurz angesprochen, welche nicht zur Verwendung kamen. Danach wurden die Anforderungen, an ein Werkzeug, zur Multiskalenvisualisierung aufgestellt. Darauf folgt die allgemeine Beschreibung des Systems und den einzelnen Komponenten. Sowie im Detail die einzelnen Ansichten (Trajectory View, Cluster View, Compare View) beschrieben. Im Anschluss an diesen Teil wurde ein Use-Case erstellt und dieser an einem Beispiel mit den Stimmuli Antwerpen und Berlin durchgeführt. Der Prototyp ist in einem privaten Gitlab Verzeichnis, bei Interesse an diesem kann Zugriff auf dieses gewährt werden.

Ausblick

In Zukunft könnten einfache Features implementiert werden. Es fehlt ein Einlesen von allgemeinen Daten über einen Dateipicker, mit dem in Kapitel 5 definierten Datenformat, sodass diese auf einem kartesischen Koordinatensystem angezeigt werden.

Auch sollte evaluiert werden, wie sinnvoll der Workflow (Abbildung 4.1) ist oder ob es sinnvolle Verbesserung oder Erweiterungen gibt.

Die Visualisierung einer großen Anzahl von Trajektorien kann in WebGl realisiert werden, da ab einer gewissen Menge an Punkten die generierte SVG-Grafik für den Browser schwerer zu Verarbeiten wird. Mit der Verwendung von WebGl kann die Grafikkarte das Rendern der Elemente direkt übernehmen.

Im Cluster View kann die Visualisierung optimiert werden, um besser sichtbar zu machen, welche Trajektorien zu welchem Paar und Cluster gehören. Für eine große Anzahl an Daten, wäre auch die Verteilung der Analyse auf mehrere Threads ein wichtiger Punkt. Javascript verwendet bei Ausführung im Browser einen Thread, mit Webworkern können solche rechenintensiven Aufgaben ausgelagert und parallelisiert werden. Als Verbesserung für den Compare View kann die Historie der entfernten Fixationen visualisiert werden, beispielsweise als Punkte oder Trajektorien in einer besonderen Farbe oder Struktur. Auch kann in den Ähnlichkeitswert, die Anzahl an Veränderungen einfließen, um so eine wenig zu verändernde Trajektorie besser zu bewerten.

Ein automatisches Entfernen von Fixationen im Compare View, die einen gewissen Grenzwert erreichen, wäre ein weiteres Feature, um die Arbeit mit dem Werkzeug weiter zu vereinfachen und zu automatisieren.

Literaturverzeichnis

[AABW12] G. Andrienko, N. Andrienko, M. Burch, D. Weiskopf. „Visual Analytics Methodology for Eye Movement Studies“. In: *IEEE transactions on visualization and computer graphics* 18.12 (2012), S. 2889–2898. ISSN: 1941-0506. DOI: 10.1109/TVCG.2012.276 (zitiert auf S. 23).

[Ang17] Angular Contributors. *Angular*. 2017. URL: https://angular.io/ (zitiert auf S. 21, 22).

[BBK+16] M. A. Borkin, Z. Bylinskii, N. W. Kim, C. M. Bainbridge, C. S. Yeh, D. Borkin, H. Pfister, A. Oliva. „Beyond Memorability: Visualization Recognition and Recall“. In: *IEEE transactions on visualization and computer graphics* 22.1 (2016), S. 519–528. ISSN: 1941-0506. DOI: 10.1109/TVCG.2015.2467732 (zitiert auf S. 23).

[BBK+17] Z. Bylinskii, M. A. Borkin, N. W. Kim, H. Pfister, A. Oliva. „Eye Fixation Metrics for Large Scale Evaluation and Comparison of Information Visualizations“. In: *Eye Tracking and Visualization: Foundations, Techniques, and Applications. ETVIS 2015*. Hrsg. von M. Burch, L. Chuang, B. Fisher, A. Schmidt, D. Weiskopf. Cham: Springer International Publishing, 2017, S. 235–255. ISBN: 978-3-319-47024-5. DOI: 10.1007/978-3-319-47024-5{\textunderscore}14. URL: https://doi.org/10.1007/978-3-319-47024-5_14 (zitiert auf S. 23).

[BBvL11] K. Buchin, M. Buchin, M. van Kreveld, J. Luo. „Finding long and similar parts of trajectories“. In: *Computational Geometry* 44.9 (2011), S. 465–476. ISSN: 09257721. DOI: 10.1016/j.comgeo.2011.05.004 (zitiert auf S. 27).

[BC94] D. J. Berndt, J. Clifford. „Using Dynamic Time Warping to Find Patterns in Time Series“. In: *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*. AAAIWS'94. Seattle, WA: AAAI Press, 1994, S. 359–370 (zitiert auf S. 13, 26).

[BH09] M. Bostock, J. Heer. „Protovis: A Graphical Toolkit for Visualization“. In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (Nov. 2009), S. 1121–1128. ISSN: 1077-2626. DOI: 10.1109/TVCG.2009.174. URL: http://dx.doi.org/10.1109/TVCG.2009.174 (zitiert auf S. 22).

[BOH11] M. Bostock, V. Ogievetsky, J. Heer. „D^3: Data-Driven Documents“. In: *IEEE transactions on visualization and computer graphics* 17.12 (2011), S. 2301–2309. ISSN: 1941-0506. DOI: 10.1109/TVCG.2011.185 (zitiert auf S. 21, 22).

[CMTG10] F. Cristino, S. Mathôt, J. Theeuwes, I. D. Gilchrist. „ScanMatch: A novel method for comparing fixation sequences“. In: *Behavior Research Methods* 42.3 (2010), S. 692–700. ISSN: 1554-3528. DOI: 10.3758/BRM.42.3.692. URL: https://doi.org/10.3758/BRM.42.3.692 (zitiert auf S. 24).

[Dij59] E. W. Dijkstra. „A note on two problems in connexion with graphs“. In: *Numerische Mathematik* 1.1 (Dez. 1959), S. 269–271. ISSN: 0945-3245. DOI: 10.1007/BF01386390. URL: https://doi.org/10.1007/BF01386390 (zitiert auf S. 19).

[DNJ+12] R. Dewhurst, M. Nyström, H. Jarodzka, T. Foulsham, R. Johansson, K. Holmqvist. „It depends on how you look at it: Scanpath comparison in multiple dimensions with MultiMatch, a vector-based approach“. In: *Behavior Research Methods* 44.4 (2012), S. 1079–1100. ISSN: 1554-3528. DOI: 10.3758/s13428-012-0212-2 (zitiert auf S. 19, 27, 38).

[EGH+02] Efrat, Guibas, S. Har-Peled, Mitchell, Murali. „New Similarity Measures between Polylines with Applications to Morphing and Polygon Sweeping“. In: *Discrete & Computational Geometry* 28.4 (Nov. 2002), S. 535–569. DOI: 10.1007/s00454-002-2886-1. URL: https://doi.org/10.1007/s00454-002-2886-1 (zitiert auf S. 17, 26).

[FB95] G. W. Furnas, B. B. Bederson. „Space-scale Diagrams: Understanding Multiscale Interfaces“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., 1995, S. 234–241. ISBN: 0-201-84705-1. DOI: 10.1145/223904.223934. URL: http://dx.doi.org/10.1145/223904.223934 (zitiert auf S. 28, 29, 31).

[HCL05] J. Heer, S. K. Card, J. A. Landay. „prefuse“. In: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '05*. Hrsg. von G. van der Veer, C. Gale. New York, New York, USA: ACM Press, 2005, S. 421. ISBN: 1581139985. DOI: 10.1145/1054972.1055031 (zitiert auf S. 26).

[HW09] D. Holten, J. J. van Wijk. „Force-directed Edge Bundling for Graph Visualization“. In: *Proceedings of the 11th Eurographics / IEEE - VGTC Conference on Visualization*. EuroVis'09. Berlin, Germany: The Eurographs Association & John Wiley & Sons, Ltd., 2009, S. 983–998. DOI: 10.1111/j.1467-8659.2009.01450.x. URL: http://dx.doi.org/10.1111/j.1467-8659.2009.01450.x (zitiert auf S. 28, 29).

[Int15] E. International. *ECMAScript® 2015 Language Specification*. 2015. URL: http://www.ecma-international.org/ecma-262/6.0/index.html (zitiert auf S. 21).

[JHN10] H. Jarodzka, K. Holmqvist, M. Nyström. „A Vector-based, Multidimensional Scanpath Similarity Measure“. In: *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*. ETRA '10. New York, NY, USA: ACM, 2010, S. 211–218. ISBN: 978-1-60558-994-7. DOI: 10.1145/1743666.1743718. URL: http://doi.acm.org/10.1145/1743666.1743718 (zitiert auf S. 19, 20, 26).

[JM16] C. J., P. M. M. „Directional scan path characterization of eye tracking sequences: A multi-scale approach“. In: *2016 Future Technologies Conference (FTC)*. 2016, S. 51–61. DOI: 10.1109/FTC.2016.7821589 (zitiert auf S. 25).

[Lev66] V. Levenshtein. „Binary Codes Capable of Correcting Deletions, Insertions and Reversals“. In: *Soviet Physics Doklady* 10 (1966), S. 707 (zitiert auf S. 17).

[Llo82] S. Lloyd. „Least squares quantization in PCM“. In: *IEEE Transactions on Information Theory* 28.2 (1982), S. 129–137. ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489 (zitiert auf S. 20).

[LMKG15] N. Larios, C. Mitatakis, V. Kalogeraki, D. Gunopulos. „Evaluating Distance Measures for Trajectories in the Mobile Setting“. In: *Proceedings of the 2Nd International Conference on Mining Urban Data - Volume 1392*. MUD'15. Lille, France: CEUR-WS.org, 2015, S. 97–105 (zitiert auf S. 26).

[Mic17] Microsoft. *Typescript*. 2017. URL: https://www.typescriptlang.org/ (zitiert auf S. 21).

[Nod17] Node.js Foundation. *Node.js*. 2017. URL: https://nodejs.org/en/ (zitiert auf S. 21).

[NPL+10] S. Nikolov, M. Petrov, L. Lymperakis, M. Friák, C. Sachs, H.-O. Fabritius, D. Raabe, J. Neugebauer. „Revealing the design principles of high-performance biological composites using ab initio and multiscale simulations: The example of lobster cuticle“. In: *Advanced materials (Deerfield Beach, Fla.)* 22.4 (2010), S. 519–526. ISSN: 1521-4095. DOI: 10.1002/adma.200902019 (zitiert auf S. 27, 28, 31).

[NW70] S. B. Needleman, C. D. Wunsch. „A general method applicable to the search for similarities in the amino acid sequence of two proteins“. In: *Journal of Molecular Biology* 48.3 (1970), S. 443–453. ISSN: 0022-2836 (zitiert auf S. 24).

[OTc17] M. Otto, J. Thornton, B. contributors. *Bootstrap – The most popular HTML, CSS, and JS library in the world.* 2017. URL: https://getbootstrap.com/ (zitiert auf S. 21, 22).

[SUF13] W. X. Schneider, University of Bielefeld, R. M. Foerster. *Functionally sequenced scanpath similarity method (FuncSim): Comparing and evaluating scanpath similarity based on a task's inherent sequence of functional (action) units.* 2013. DOI: 10.16910/jemr.6.5.4 (zitiert auf S. 26).

[SWW+90] F. A. Stephen, G. Warren, M. Webb, W. M. Eugene, J. L. David. „Basic local alignment search tool“. In: *Journal of Molecular Biology* 215.3 (1990), S. 403–410. ISSN: 0022-2836. DOI: 10.1016/S0022-2836(05)80360-2. URL: http://www.sciencedirect.com/science/article/pii/S0022283605803602 (zitiert auf S. 23).

[VGK02] M. Vlachos, D. Gunopoulos, G. Kollios. „Discovering Similar Multidimensional Trajectories“. In: *Proceedings of the 18th International Conference on Data Engineering*. ICDE '02. Washington, DC, USA: IEEE Computer Society, 2002, S. 673– (zitiert auf S. 13, 26).

[WHRK06] J. M. West, A. R. Haake, E. P. Rozanski, K. S. Karn. „eyePatterns“. In: *Proceedings of the 2006 symposium on Eye tracking research & applications - ETRA '06*. Hrsg. von K.-J. Räihä, A. T. Duchowski. New York, New York, USA: ACM Press, 2006, S. 149. ISBN: 1595933050. DOI: 10.1145/1117309.1117360 (zitiert auf S. 25).

[WSZ+13] H. Wang, H. Su, K. Zheng, S. Sadiq, X. Zhou. „An Effectiveness Study on Trajectory Similarity Measures“. In: *Proceedings of the Twenty-Fourth Australasian Database Conference - Volume 137*. ADC '13. Adelaide, Australia: Australian Computer Society, Inc., 2013, S. 13–22. ISBN: 978-1-921770-22-7 (zitiert auf S. 27).

Alle URLs wurden zuletzt am 11. 12. 2017 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