der Bundeswehr
Universität München

Hierarchisches Clustering von Kinect-Bewegungsdaten mittels Dynamic Time Warping

Bachelorarbeit von

Laurenz Fuchs

1196307

Erstprüfer:	Prof. Dr. Michael Koch
Zweitprüfer:	Prof. Dr. Gunnar Teege
Betreuer:	M. Sc. Julian Fietkau
Abgabetermin:	31. Mai 2022

Universität der Bundeswehr München

Fakultät für Informatik

Kurzfassung

Im Rahmen des *HoPE-Projekts* wird das Verhalten von Menschen bei der Nutzung von *Ambient Displays* analysiert. Tiefenkameras werden als eine mögliche Sensorik genutzt, um Bewegungsabläufe aufzunehmen. Konkret werden dazu Skelettdaten betrachtet. Eine manuelle Analyse der Daten ist schwer umsetzbar. Daher muss ein methodisches Rahmenwerk entwickelt werden, welches eine Evaluation von Sensordaten ermöglicht. Diese Bachelorarbeit ist ein Beitrag zu diesem Rahmenwerk. Ziel ist die Entwicklung eines Java-Tools, welches derartige Daten ohne Vorwissen, mithilfe deterministischer Algorithmen clustern kann. Es kommt *hierarchisches Clustering* mittels des *Dynamic Time Warping* Algorithmus zum Einsatz. Es soll geklärt werden, ob diese Algorithmen einen sinnvollen Beitrag zur Analyse von Bewegungsdaten leisten können. Nach der Implementierung erfolgt ein Clustering von *Kinect-Bewegungsdaten*. Mithilfe einer Ground-Truth-Analyse und deskriptiven Statistiken wird die Qualität der gefundenen Cluster überprüft. Insbesondere hat hat sich dabei herausgestellt, dass ein geeigneter Threshold von großer Bedeutung ist. Eine Evaluation bestätigt, dass die verwendeten Algorithmen zum Clustering von Time-Series Daten genutzt werden können. Für die Interpretation dieser Daten ist allerdings weitere Forschung notwendig.

Inhaltsverzeichnis

1 Einleitung

Von September 2021 bis August 2024 forschen die Hochschule für angewandte Wissenschaften Hamburg und die Universität der Bundeswehr München im Rahmen des *HoPE-Projekts* am Verhalten von Menschen bei der Nutzung von großen, interaktiven Wandbildschirmen. Im Zuge des Projekt wird unter anderem der *Honeypot-Effekt* erforscht (UniBw, 2021). Dieser beschreibt im Bereich der Mensch-Computer-Interaktion (HCI) wie Menschen, die mit einem System interagieren, weitere Passanten anregen die Interaktion zu beobachten oder sogar an ihr teilzuhaben (Wouters et al., 2016). Der Honeypot-Effekt soll bei der Nutzung von *Ambient Displays* im öffentlichen und halb-öffentlichen Raum in Langzeit-Feldstudien untersucht werden. Dabei muss auch der Aspekt der Datenerhebung und -analyse weiter ausgebaut werden. Hierzu muss ein methodisches Rahmenwerk entwickelt werden, welches eine auf Sensordaten-basierende, automatische und zeitlich uneingeschränkte Evaluation von Ambient Displays ermöglicht (UniBw, 2021). Die Sensordaten werden von Body-Tracking-Kameras bereitgestellt. Es liegt ein Datensatz vor, für den Wandbildschirme mit Microsoft Kinect v2 Kameras ausgestattet wurden. Diese zeichnen die Interaktion von Nutzern mit den Displays auf, wodurch das Nutzerverhalten zu einem späteren Zeitpunkt ausgewertet werden kann. Die Daten liegen als *Time-Series Data (TSD)* vor. Dabei handelt es sich um geordnete Sequenzen von Datenpunkten, die über eine gewisse Zeit hinweg aufgenommen werden (Ali et al., 2019). Da sich die Attributwerte mit der Zeit verändern, spricht man von dynamischen Daten. TSD enthalten oft wichtige Informationen, die durch eine Analyse entdeckt werden können (Ali et al., 2019). Mit ihrer Hilfe können die Daten gruppiert werden. Dabei stellt sich die Frage, ob Kategorien identifizierbar sind, die etwas über das Verhalten von Menschen vor Wandbildschirmen aussagen. Wiederkehrende Bewegungsabläufe treten teilweise zu unterschiedlichen Zeitpunkten in den Aufnahmen auf. Zudem kann sich die Ausführung bei verschiedenen Personen unterscheiden. Dies erschwert eine manuelle Kategorisierung. Bei großen Datensätzen ist ein manuelles Vorgehen im Allgemeinen ohnehin nicht möglich. Daher muss eine automatisierte Sortierung angeboten werden. In dieser Bachelorarbeit wird eine mögliche Herangehensweise zur Lösung des Problems mittels deterministischer Algorithmen

betrachtet. Wesentliches Ziel ist die Implementierung eines Systems zur Kategorisierung der vorliegenden *Kinect-Bewegungsdaten*. Eine gängige Methode zur Analyse sind dabei *Clustering*-Verfahren (Aghabozorgi et al., 2015). Konkret soll *hierarchisches Clustering* mithilfe des *Dynamic Time Warping (DTW)*-Algorithmus eingesetzt werden.

Die Arbeit ist wie folgt aufgebaut: Zunächst wird in Kapitel 2 auf wichtige Grundlagen der Thematik eingegangen. Es werden mögliche Modelle der Interaktion mit Wandbildschirmen vorgestellt. Außerdem werden die Grundzüge der Kinect Kamera und des vorliegenden Datensatzes dargestellt. Abschließend erfolgt eine tiefergehende Erläuterung der Problemstellung. Kapitel 3 widmet sich dem Hierarchischen Clustering und dem DTW-Algorithmus. Hier werden Gründe für die Wahl dieser Algorithmen erwähnt. Nach einer ausführlichen Beschreibung der Funktionsweise wird das Vorgehen an einem Beispiel veranschaulicht. Am Ende wird auf verwandte Literatur und den Einsatz im Kontext von Kinect-Bewegungsdaten eingegangen. Kapitel 4 behandelt die Konzeption des Software-Tools. Dafür werden die Anforderungen betrachtet, bevor im Rest des Kapitels auf den Programmablauf, sowie nötige Teilsysteme eingegangen wird. Anschließend erfolgt die Implementierung der Anwendung. In Kapitel 5 wird deren Aufbau beleuchtet. Außerdem werden zentrale Ausschnitte des Codes und Abweichungen zur Konzeption beschrieben. Abschließend erfolgt eine Evaluation (Kapitel 6). Hier wird zunächst auf Limitierungen der Anwendung und die zur Evaluation verwendeten Daten eingegangen. Teile des Datensatzes werden mithilfe des Tools geclustert. Die Qualität der Cluster wird mithilfe von Ground-Truth-Daten und durch deskriptive Statistiken überprüft. Ein abschließendes Fazit erfolgt in Kapitel 7.

2 Grundlagen und Problembeschreibung

Als Basis für das weitere Vorgehen werden nun grundlegende Aspekte herausgearbeitet. Dabei werden neben einigen Interaktionsmodellen vor Wandbildschirmen auch die eingesetzte Kinect-Sensorik und die Struktur des vorliegenden Datensatzes beschrieben. Zudem erfolgt abschließend eine ausführliche Problembeschreibung.

2.1 Modelle der Interaktion mit Wandbildschirmen

Wouters et al. (2016) verweisen darauf, dass interaktive digitale Medien in der Öffentlichkeit immer präsenter sind. Das anfängliche Interesse an der Technologie ist gesunken. Für Wandbildschirme, die versuchen, die Aufmerksamkeit von Passanten zu erregen und sie zur Interaktion zu animieren, wird dies immer schwieriger (Wouters et al., 2016). Diese Herausforderungen können nicht allein durch verbesserte Hardware oder attraktiv gestaltete Hardware gelöst werden. Stattdessen muss ein besseres Verständnis von Menschen und deren Technologienutzung geschaffen werden, um diese Erkenntnisse bei der Gestaltung zukünftiger Wandbildschirme berücksichtigen zu können (Wouters et al., 2016). Ambient Displays sind große, interaktive Bildschirme im (halb-) öffentlichen Raum, mit denen Nutzer interagieren können. Es handelt sich meist um ansprechende Displays die Personen mit Informationen versorgen (Mankoff et al., 2003). Eine Kategorisierung von Interaktionen mit solchen Bildschirmen kann das Verständnis des Nutzungsverhaltens verbessern. Dazu existieren verschiedenste *Audience Behaviour-Interaktionsmodelle*. Das *Audience Funnel Modell* ist ein weit verbreitetes Modell und der *Honeypot-Effekt* ist im Kontext des HoPE-Projekts relevant. Daher sollen diese beiden Modelle im Folgenden näher beschrieben werden.

Das Audience Funnel Modell (Mai & Hußmann, 2018; Wouters et al., 2016) beschreibt, wie Menschen sich um große halb-öffentliche oder öffentliche Displays versammeln und von Beobachtern zu Interagierenden mit dem System, und anschließend wieder zu Beobachtern werden. Menschen neigen dazu verschiedene Phasen der Interaktivität zu durchlaufen, bevor sie direkt mit dem System interagieren (Mai & Hußmann, 2018; Wouters et al., 2016). Die

einzelnen Phasen des Audience Funnel werden in Abbildung 2.1 gezeigt. Eine der Aufgaben eines Wandbildschirms kann es sein, Aufmerksamkeit auf sich zu ziehen und den Nutzer zu motivieren mit dem System zu interagieren (Mai & Hußmann, 2018). Mai und Hußmann (2018) verweisen aber darauf, dass Ambient Displays in der Öffentlichkeit nicht unbedingt der zentrale Punkt der Aufmerksamkeit sind, da vorbeigehende Personen eigene intrinsische Ziele verfolgen. Die Herausforderung für Entwickler ist es, die Inhalte auf den Systemen so relevant zu gestalten, dass sie Aufmerksamkeit erregen, sich aber gleichzeitig nicht aufdrängend in den Mittelpunkt stellen (Mai & Hußmann, 2018).

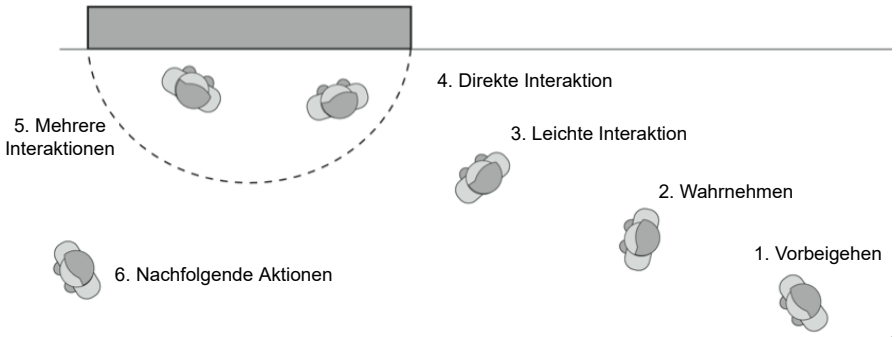

Abbildung 2.1: Audience Funnel Framework. Abbildung aus Mai und Hußmann (2018).

Ein zweites Modell wird durch den bereits erwähnten Honeypot-Effekt beschrieben. Er zeigt, dass Individuen unabhängig von Belohnung, Bestrafung oder sozialem Wettbewerb von der reinen Präsenz oder den Aktivitäten anderer beeinflusst werden (Wouters et al., 2016). In der HCI wird dies meist erkennbar, indem Passanten sich einem System nähern und überlegen, ob sie mit ihm interagieren sollen, nachdem sie anderen Menschen dabei zugesehen haben (Wouters et al., 2016).

Eine Einordnung von Bewegungsdaten in die Kategorien solcher Modelle und eine weiterführende Analyse der Bewegungen kann Aufschluss über das Verhalten von Menschen vor interaktiven Bildschirmen geben. Wie bereits erwähnt, soll eine derartige Kategorisierung automatisiert werden.

2.2 Spezifikation der Microsoft Kinect

Ursprünglich war der Xbox 360 Kinect-Sensor ausschließlich für die Videospiel-Industrie gedacht. Schon bald wurde dieser aber auch für wissenschaftliche Experimente genutzt

(Tölgyessy et al., 2021). So kam auch im vorliegenden Datensatz des HoPE-Projekts die Kinect v2 für Xbox One zum Einsatz. Die zugrunde liegende Sensorik stellt Farbbilder einer Rot-Grün-Blau (RGB) Kamera, Tiefendaten mithilfe einer Tiefenkamera und Audiodateien verschiedener Mikrofone zur Verfügung (Microsoft, 2014). Tiefenkameras ordnen jedem Punkt eines Bildes eine Entfernung zum Sensor zu. Dies hilft zuverlässige Ergebnisse bei der Erkennung von Menschen vor Ambient Displays zu erzielen. Li et al. (2014) stellen fest: „Die kompakte Größe, die Benutzungsfreundlichkeit, die stark vereinfachte Hintergrund-Subtraktion im Vergleich zu anderer Sensorik, sowie die hohe Genauigkeit und die hohe Bildrate machen Tiefenkameras zu einer attraktiven Lösung für ein breites Spektrum an Anwendungen.“ Die Kinect v2 verwendet dabei den Ansatz der kontinuierlichen Wellenintensitätsmodulation, der häufig bei Time-of-Flight (ToF)-Tiefenkameras zum Einsatz kommt. Dabei wird das Licht einer Lichtquelle von anderen Objekten im Sichtfeld der Kamera zurückgestreut und die Zeit zwischen der Emittierung und dem Wiedereintreffen des reflektierten Lichtes gemessen. Diese „Flugzeit“ wird für jedes Pixel im Bild in einen Entfernungswert umgerechnet (Tölgyessy et al., 2021). Der Sensor kann Tiefenbilder mit einer Auflösung von 512 x 424 Pixeln und Farbbilder mit 1920 x 1080 Pixeln aufnehmen (Marin et al., 2019). Bei der Kinect v2 können bis zu sechs Personen erfasst werden. Dabei wird die Lage von 25 Skelettpunkten, sowie verschiedene Gesichtsattribute erfasst (Microsoft, 2014). Abbildung 2.2 zeigt eine Übersicht dieser Punkte.

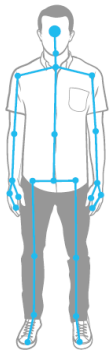

Abbildung 2.2: Skelettpunkte der Kinect v2. Abbildung von Microsoft (2014).

2.3 Struktur des vorliegenden Datensatzes

Im Rahmen eines Feldeinsatzes wurden, im agilen Umfeld eines Software-Unternehmens, Ambient Displays installiert. Mitarbeiter konnten dort auf Hilfsanwendungen zur Softwareentwicklung, wie *Jira* oder *Confluence*, zugreifen (Schwarzer et al., 2021). Die Bildschirme wurden mit Kinect v2 Systemen ausgestattet, um die Bewegungen der Vorbeigehenden aufzuzeichnen (Schwarzer et al., 2021). Der dabei entstandene Datensatz dient zur späteren Evaluation der zu implementierenden Anwendung.

Eine Betrachtung des Datensatzes ergab, dass dieser 97.626 Records beinhaltet (Temiz, 2022). Dabei sind jedem Record mehrere sogenannte Frames zugeordnet, welche der Momentaufnahme eines Menschen entsprechen. Jede Person erhält eine eindeutige Identifikationsnummer. Falls sich mehr als eine Person gleichzeitig im Sensorbereich befindet, erfolgt eine weitere Unterteilung des Records in Frames (Temiz, 2022). Zu jedem Record liegen verschiedene Dateien vor. Sie tragen jeweils den Zeitstempel in Kombination mit einem geeigneten Postfix als Dateinamen. Tabelle 2.1 zeigt die Attribute der Datei *timestamp.txt*. Diese Attribute können im Kontext der Kategorisierung von Bewegungsdaten genutzt werden. Dazu kann eine Teilmenge der Werte verschiedener Records miteinander verglichen werden. So können sie auf Gemeinsamkeiten und Unterschiede hin untersucht werden. Die Attribute werden in der Textdatei durch das Trennzeichen '###' voneinander abgegrenzt. Ein exemplarischer Frame aus dem Datensatz kann Abbildung 2.3 entnommen werden. Da die Anordnung und das Vorhandensein dieser Attribute je nach Version des Datensatzes abweichen kann, sollen diese in der Implementierung manuell konfigurierbar sein. Selbes gilt für das Trennzeichen. Das Tool kann deshalb auch eingesetzt werden, falls verschiedene Versionen des Datensatzes ausgeliefert wurden. Die Datei *timestamp_bodies.txt* enthält für jeden Frame die Position aller 25 Skelettpunkte. *timestamp.xef* kann genutzt werden, um die Aufnahme in der Anwendung Kinect Studio zu visualisieren. Letztlich sind die Einträge sogenannte TSD. Bei diesem Datentyp handelt es sich um geordnete Sequenzen von Datenpunkten, die, oft in regelmäßigen Abständen, über eine gewisse Zeit hinweg aufgenommen wurden (Ali et al., 2019). Insgesamt befinden sich im Datensatz 34.687.630 Frames (Temiz, 2022). Diese Anzahl verdeutlicht die Notwendigkeit eines Software-Tools zur Auswertung. Für diese Bachelorarbeit wurde eine vorselektierte Version des Datensatzes bereitgestellt, welche auch in Kapitel 6 zum Einsatz kommt.

Dieser Datensatz kam durch folgende Regeln zustande:

1. Nur Aufzeichnungen von Werktagen werden beachtet.
2. Nur Aufzeichnungen zwischen 07:00 Uhr und 17:59 Uhr werden beachtet.
3. Nur Aufzeichnungen mit mindestens 63 Frames pro Aufzeichnung werden beachtet.
4. Nur Aufzeichnungen mit Personen, die Mindestmaß an Aufmerksamkeit zeigen, werden beachtet.
5. Aufzeichnungen mit 100 Prozent Engagement werden nicht beachtet, da manche dieser Fälle durch die Kinect falsch erkannt wurden.
6. Aufzeichnungen mit großen Sprüngen auf den Achsen werden nicht beachtet.

Hier ist anzumerken, dass das zu entwickelnde Tool auch mit anderen Datensätzen funktionieren soll. So sollen bei Bedarf auch Daten anderer Tiefenkameras geclustert werden können.

2017-04-10 07:41:53.943 +02:00 ### 72057594038063128 ### 1341053376 ### 1 ### 1 ### No ### Maybe ### Unknown ### No ### No ### No ### Yes ### Maybe ### Unknown ### Closed ### 0,1074784 ### 0,2451882 ### 4,18441 ### 4,19296454679429

Abbildung 2.3: Frame aus dem Datensatz.

Attributname	Beschreibung
Zeitstempel	Zeitpunkt der Aufnahme.
KinectId	Skelett-Identifikationsnummer des Frameworks.
RecordId	Identifikationsnummer des Records.
BodyIndex	Identifikationsnummer der Person.
BodyCount	Anzahl der erfassten Personen.
Happy	Person ist glücklich.
Engaged	Person zeigt Interesse.
WearingGlasses	Person trägt eine Brille.
LeftEyeClosed	Person hat das linke Auge geschlossen.
RightEyeClosed	Person hat das rechte Auge geschlossen.
MouthOpen	Person hat den Mund geöffnet.
MouthMoved	Person bewegt den Mund.
LookingAway	Person schaut nicht zur Kinect.
Body.HandLeftState	Zustand der linken Hand.
Body.HandRightState	Zustand der rechten Hand.
x	x-Koordinate des Skelettpunkts SpineShoulder.
y	y-Koordinate des Skelettpunkts SpineShoulder.
z	z-Koordinate des Skelettpunkts SpineShoulder.
Distance	Distanz zwischen Kinect und SpineShoulder (in Metern).

Tabelle 2.1: Attribute des Datensatzes.

2.4 Problembeschreibung

Da die Bewegungen im Datensatz von verschiedenen Personen ausgeführt werden, kann es sein, dass gegebenenfalls auch gleiche Bewegungshandlungen voneinander abweichen. Zudem kann es vorkommen, dass die Bewegungen zu unterschiedlichen Zeitpunkten in den Records auftreten. Diese Tatsachen machen eine manuelle Kategorisierung oft schwierig. Hinzu kommt das bereits beschriebene Problem der großen Datenmenge. Im Folgenden werden mögliche Lösungsansätze der beschriebenen Problematik diskutiert.

Es gibt verschiedene Möglichkeiten sich der Kategorisierung von Bewegungsdaten zu nähern (Aghabozorgi et al., 2015). So kann beispielsweise der Ansatz des *Maschine Learnings* eingesetzt werden, welcher derzeit ebenfalls im Rahmen des HoPE-Projekts erforscht (Plischke, 2022) wird. In dieser Bachelorarbeit werden hingegen *deterministische Algorithmen* genutzt. Aghabozorgi et al. (2015) verweisen darauf, dass die Verwendung von Methoden wie Maschine Learning (*supervised*), im Falle von großen Datensätzen, zu Problemen führen kann. Es ist nicht möglich Trainingsdaten, die alle möglichen Szenarien abdecken, zur Verfügung zu stellen. Dies ist auch nicht Ziel der Untersuchungen. Stattdessen stellt sich die Frage, ob mithilfe der Anwendung automatisiert Bewegungsabläufe gefunden werden können. Deterministische Cluster-Algorithmen sind hier geeignet, da diese keine zuvor definierten Klassen benötigen (Aghabozorgi et al., 2015).

Ziel der Verarbeitung ist es, möglichst sinnvolle *Cluster* zu erhalten. Um die Gemeinsamkeiten zweier Aufnahmen zu berechnen, wird eine geeignete Vergleichsfunktion benötigt. Die Wahl dieser Funktion ist ausschlaggebend für den Erfolg des *Clusterings* (Warren Liao, 2005). Zu beachten ist zudem, dass die Bewegungsaufnahmen durch TSD beschrieben werden. Bei der bloßen Anwendung herkömmlicher Distanzmetriken, wie der Euklidische-Distanz, ist die Aussagekraft des Vergleichs gering (Warren Liao, 2005). Wie bereits erwähnt kann es vorkommen, dass verschiedene Personen die gleiche Bewegung unterschiedlich schnell, oder zeitlich versetzt ausführen. Gegebenenfalls weisen die Datenreihen gleicher Bewegungen daher sogar eine unterschiedliche Anzahl an Frames auf. Dies wird in Abbildung 2.4 veranschaulicht. Die Kurven weisen ähnliche Segmente auf. Diese Ähnlichkeiten treten allerdings zu unterschiedlichen Zeitpunkten auf. Zudem sind die Datenreihen unterschiedlich lang. Der letzte Punkt der Reihe kann daher gar keinem anderen Punkt zugordnet werden. Bei der bloßen Anwendung der bekannten euklidischen Distanz, ohne Zuordnung der Punkte, ist das Ergebnis hier nicht aussagekräftig. Um dieses Problem zu lösen ist eine *elastische Metrik* nötig, die besser mit zeitlichen Verschiebungen umgehen kann (Aghabozorgi et al., 2015).

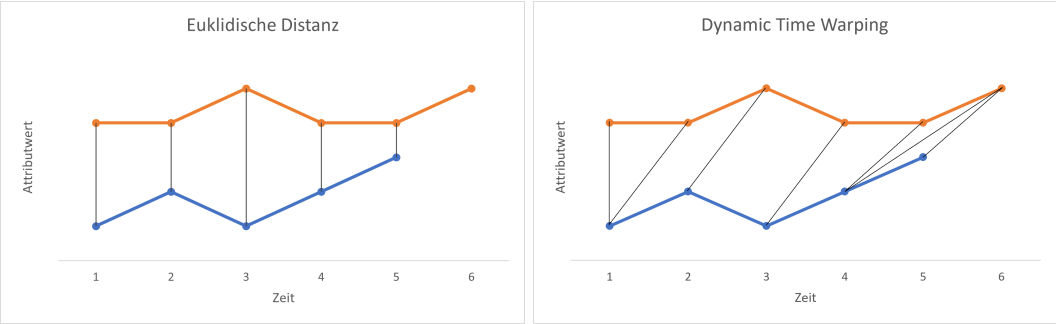

Abbildung 2.4: Zuordnung der Messpunkte zweier Datenreihen.

Dazu wird DTW verwendet (Abschnitt 3.2). Hier ist eine Mehrfachzuordnung von Punkten möglich. Dies erlaubt eine Zuordnung ähnlicher Muster in den Daten, auch wenn sie zeitlich verschoben sind (Mohammadzade et al., 2021). Diese können so verbunden werden, dass die Kombinationskosten minimal sind. Es gibt also keine andere Zuordnung der Punkte, sodass die Summe der Differenzen zwischen ihnen geringer wäre. Das Clustering soll mithilfe von hierarchischem Clustering, welches in Abschnitt 3.1 beschrieben wird, durchgeführt werden. Das Verfahren ist für die vorliegenden Daten geeignet, da es erlaubt TSD unterschiedlicher Länge zu clustern. Zudem muss die Anzahl der zu bildenden Cluster nicht im Voraus definiert werden, wie dies etwa bei *K-Means* der Fall ist (Aghabozorgi et al., 2015). Es ist in diesem Kontext nicht zielführend die Clusteranzahl zuvor zu definieren, da die Arbeit darauf abzielt, mehr Erkenntnis über mögliche Cluster zu gewinnen. Statt die Anzahl vorzugeben, wird ein *Threshold* definiert, der angibt, ab wann zwei Cluster nicht mehr zusammengeführt werden sollen, weil die Differenz zwischen ihnen zu groß ist. Hierarchisches Clustering mit DTW ist also zur Lösung des beschriebenen Problems geeignet.

3 Clustering von Bewegungsdaten

Nach den einführenden Bemerkungen in Kapitel 1 und Kapitel 2 werden nun die eingesetzten deterministischen Algorithmen näher beschrieben. Die Grundlagen der Verfahren werden erläutert und an einem Beispiel veranschaulicht. Ein Einblick in verwandte Arbeiten (Abschnitt 3.4) vertieft das Verständnis weiter. Abschließend wird auf zu beachtende Besonderheiten des Kinect-Datensatzes eingegangen.

3.1 Hierarchisches Time-Series Clustering

Clustering kann genutzt werden, um große Datensätze zu kategorisieren, wenn vorab keine Informationen über die Kategorien vorliegen (Aghabozorgi et al., 2015). Ziel ist es, eine Struktur in den vorliegenden Daten zu finden, indem Elemente anhand eines festgelegten Vorgehens in homogene Gruppen einsortiert werden. Dabei sollen die Unterschiede bei Elementen der gleichen Gruppe möglichst klein und bei Elementen verschiedener Gruppen möglichst groß sein (Aghabozorgi et al., 2015; Warren Liao, 2005). Bei n Datenpunkten erzeugt ein Clustering k Partitionen des Datensatzes. Jede Partition entspricht dabei einem Cluster, das mindestens ein Objekt enthält (Warren Liao, 2005). Beim *Clustering* sind besonders die Wahl des Clustering-Algorithmus und der Methode zur Bestimmung der Ähnlichkeit von zwei Datenreihen von Bedeutung (Warren Liao, 2005). Letzteres wird durch DTW gelöst. Das Verfahren wird in Abschnitt 3.2 beschrieben. Als Clustering-Vorgehen kann das sogenannte hierarchische Clustering genutzt werden. Dabei werden Datenobjekte in einem Cluster-Baum eingeordnet. Es wird ein *bottom-up* Ansatz verfolgt (Warren Liao, 2005). Jedes Objekt ist daher zunächst ein eigenes Cluster. Dann werden schrittweise Cluster zusammengeführt, bis alle Objekte im selben Cluster sind, oder eine zuvor definierte Terminierungsbedingung erfüllt ist (Warren Liao, 2005). In jedem Iterationsschritt werden dabei die beiden Cluster zusammengeführt, welche die geringste Differenz zueinander aufweisen. Die Bestimmung der Differenz wird in Abschnitt 3.2 erläutert. Die Terminierungsbedingung entspricht in diesem Kontext dem Überschreiten des vordefinierten Thresholds.

Das Clustering von TSD findet in vielen wissenschaftlichen Gebieten Anwendung, um Muster zu finden (Aghabozorgi et al., 2015). Das Vorgehen hilft dabei, wichtige Informationen aus großen Datensätzen zu extrahieren. Das Kategorisieren von TSD-Objekten kann bei der Detektion relevanter Muster helfen (Aghabozorgi et al., 2015). Die meisten bestehenden Cluster-Algorithmen können auf diesen Datentypen zugeschnitten werden. So auch das hierarchische Clustering. Wie bereits erwähnt, ist der größte Unterschied, dass die Distanz-/ Ähnlichkeitsmetrik angepasst werden muss, um sinnvoll bei TSD genutzt werden zu können.

3.2 Dynamic Time Warping Algorithmus

Der DTW Algorithmus kann als Distanz-Metrik beim hierarchischen Clustern genutzt werden. Hier sind *one-to-many* oder *one-to-none* Beziehungen zwischen Datenpunkten möglich. Das Vorgehen wird daher auch als eine *elastische Metrik* bezeichnet. Diese kann mit zeitlichen Verschiebungen und unterschiedlich langen Datenreihen umgehen (Aghabozorgi et al., 2015).

In der Literatur gibt es viele Erläuterungen des DTW-Algorithmus. Beispielsweise in Mohammadzade et al. (2021), Warren Liao (2005), Aghabozorgi et al. (2015) oder Yu und Xiong (2019). Diese Arbeiten helfen, den Algorithmus im Folgenden zu beschreiben. Dabei werden im Wesentlichen die Erläuterungen aus Mohammadzade et al. (2021) und Warren Liao (2005) genutzt. Das Ziel ist die Zuordnung einzelner Signale zu Gruppen von Signalen, die im Verlauf der Zeit ähnliche Muster aufweisen. Diese Muster treten zu unterschiedlichen Zeiten oder in unterschiedlichen Raten auf. Um die Zuweisung zueinander zu erreichen, muss ein Signal gegebenenfalls lokal gestreckt oder gestaucht werden. Es kann eine Korrespondenz zwischen den Zeitindizes zweier Signale gefunden werden, sodass die Summe der Distanzen zwischen den beiden Signalen minimal ist (Mohammadzade et al., 2021). Mit Distanz ist hierbei der Abstand zwischen zwei Punkten gemeint, der mithilfe einer Distanzfunktion berechnet wird.

Seien $Q = q_1, q_2, ..., q_i, ..., q_n$ und $R = r_1, r_2, ..., r_j, ..., r_m$ zwei Signale, die durch DTW ausgerichtet werden sollen, sodass die Summe der Differenzen zwischen ihren Punkten minimal ist. Dafür wird eine n *x* m Matrix M gebildet, wobei ein Element *(i, j)* die Distanz $d(q_i, r_j)$ zwischen zwei Punkten q_i und r_j enthält. Hierfür wird im Normalfall die euklidische Distanz genutzt (Warren Liao, 2005). Allerdings sind je nach Problemstellung auch andere Funktionen denkbar. In der Zelle *(1, 1)* enthält die Matrix den Wert 0. Für die übrigen Zellen der ersten Spalte und Zeile wird jeweils der Wert „unendlich“ angenommen. Das Verfahren kann auf einfache Weise auf mehrdimensionale TSD angepasst werden. In diesem Fall

sind die zu minimierenden Kosten die Summe der Distanzen zwischen den korrespondierenden Werten der beiden Datenreihen (Mohammadzade et al., 2021). Ein *Warping Path* $W = w_1, w_2, ..., w_k, ..., w_K$, ist eine Reihe von Matrixzellen, die drei Bedingungen erfüllen:

- Grenzbedingungen
- Kontinuität
- Monotonie

Die Grenzbedingungen fordern, dass die Startzelle des Warping Pfads in der Matrix eine andere ist als die Zielzelle. Dabei gilt: $w_1 = (m, n)$ und $w_K = (1, 1)$. Aufgrund der Kontinuität sind nur Schritte in angrenzende Zellen erlaubt. Zuletzt führt die Monotoniebedingung dazu, dass der Warping Path sich monoton in der Zeit bewegt. Rücksprünge in vorherige Zellen sind daher nicht möglich.

Die Matrixeinträge können von $M_{(1,1)}$ ausgehend berechnet werden, indem wiederholt die folgende Gleichung ausgewertet wird: $M_{(i,j)} = d(q_i, r_j) + min\ \{M_{(i-1,j-1)}, M_{(i-1,j)}, M_{(i,j-1)}\}$. Sie definiert den kumulativen Abstand als die Summe aus dem Abstand des aktuellen Elementes und dem Minimum der kumulativen Abstände der benachbarten Elemente (Warren Liao, 2005). Der Pfad mit der minimalen Distanz ist von Bedeutung, da er Informationen zur optimalen Zuordnung der Punkte liefert. Er kann mit $d_{DTW} = min(\frac{\sum_{k=1}^{K} w_k}{K})$ berechnet werden. Wobei es sich bei der Division durch K um eine Normalisierung handelt.

3.3 Beispiel zum hierarchischem Clustering mittels Dynamic Time Warping

Zur Veranschaulichung soll nun das Vorgehen mit vier beliebigen Datenreihen durchgeführt werden. Seien $R_1 = (0, 1, 1, 2, 2, 1, 1)$, $R_2 = (0, 1, 1, 1, 2, 2, 1, 1)$, $R_3 = (1, 1, 0, 1, 2, 2, 1, 1)$ und $R_4 = (2, 2, 1, 2, 2, 2)$ gegeben. Da beim hierarchischen Clustering zunächst jede Datenreihe ein eigenes Cluster darstellt, handelt es sich hierbei um die initialen Cluster. Die Kosten können berechnet werden, indem die Kostenmatrix aufgestellt und der Warping Pfad berechnet wird (Abschnitt 3.2). Für die Berechnung der Kosten zwischen R_1 und R_2 ergibt sich beispielsweise die in Tabelle 3.1 gezeigte Matrix mit hervorgehobenem Warping Pfad. Tabelle 3.2 zeigt die gerundeten Kombinationskosten der einzelnen Reihen. Da R_1 und R_2 zusammen die geringsten Kosten haben bilden sie das erste Cluster C_1. Um die Wertereihe des neuen Clusters zu erhalten, werden die Attributwerte der beiden ursprünglichen Cluster

paarweise addiert. Anhand des Warping Paths kann abgelesen werden, welche Werte dabei zusammengehören. Die berechneten Summen werden anschließend jeweils halbiert, damit die Werte des neuen Clusters mit den anderen Reihen vergleichbar bleiben. So ergibt sich C_1 = *(0, 1, 1, 1, 2, 2, 1, 1)*. Falls mehrere Attribute berücksichtigt werden müssen, wird für jedes Attribut eine eigene Matrix erstellt und der Warping Path berechnet. Die Kosten der Pfade werden aufsummiert und durch die Anzahl der Attribute dividiert. Damit kann auch im mehrdimensionalen Fall ein Vergleich erfolgen. Die mittlere Wertereihe wird in diesem Fall für jedes Attribut neu berechnet. Es sollte ein Threshold t definiert werden, der angibt, ab wann die Kosten zu groß sind, um zwei Datenreihen zu einem neuen Cluster zusammenzuführen. Ein sinnvoller Wert hängt dabei vom vorliegenden Datensatz ab. Sei dieser Threshold im Folgenden als $t = 2$ definiert. Nun werden die Schritte wiederholt, wobei C_1 als eine einzelne neue Datenreihe betrachtet wird. Dabei ergeben sich die Kosten in Tabelle 3.3. Hier haben C_1 und R_3 die geringsten Kombinationskosten. Da die Kosten geringer sind als der Threshold kann ein neues Cluster gebildet werden. Es ergibt sich C_2 = *(0.5, 1, 0.5, 1, 2, 2, 1, 1)*. Damit können die neuen Kosten berechnet werden, welche in Tabelle 3.4 dargestellt sind. Da die berechneten Kosten den zuvor definierten Threshold übersteigen, wird kein neues Cluster mehr gebildet und die Berechnung terminiert. Abbildung 3.1 veranschaulicht die gebildeten Cluster.

0	∞	∞	∞	∞	∞	∞	∞	∞
∞	**0**	1	2	3	5	7	8	9
∞	1	**0**	**0**	0	1	2	2	2
∞	2	0	0	**0**	1	2	2	2
∞	4	1	1	1	**0**	0	1	2
∞	6	2	2	2	0	**0**	1	2
∞	7	2	2	2	1	1	**0**	0
∞	8	2	2	2	2	2	0	**0**

Tabelle 3.1: Kostenmatrix mit hervorgehobenem Warping Path.

Reihen	Kosten
R_1, R_2	**0**
R_1, R_3	1.56
R_1, R_4	2.88
R_2, R_3	2.89
R_3, R_4	2.67

Tabelle 3.2: Erster Durchlauf der Kostenberechnung.

Reihen	Kosten
C_1, R_3	**1.56**
C_1, R_4	2.89
R_3, R_4	2.67

Tabelle 3.3: Zweiter Durchlauf der Kostenberechnung.

Reihen	Kosten
C_2, R_4	**2.78**

Tabelle 3.4: Dritter Durchlauf der Kostenberechnung.

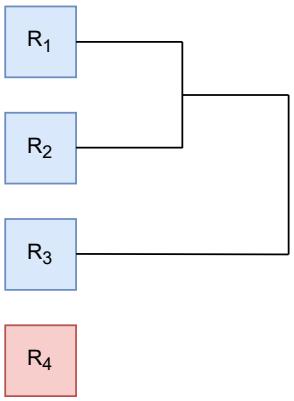

Abbildung 3.1: Graphische Veranschaulichung der gebildeten Cluster.

3.4 Related Work Analyse

DTW kommt in vielen Arbeiten als Distanz-Metrik zum Einsatz. Im Folgenden wird eine Auswahl dieser Veröffentlichungen und deren Einfluss auf diese Bachelorarbeit beschrieben.

Wahyuni et al. (2021) entwickeln ein Erkennungssystem für Gesten. Dabei wird eine Kinect-Kamera genutzt, um die Bewegungen aufzuzeichnen. Es kommt eine Form des DTW-Algorithmus zum Einsatz, um die Gesten zu erkennen. Für den Vergleich von Bewegungsabläufen werden die x- und y-Koordinaten von sechs Körperpunkten genutzt. Dabei liegen zu jeder bekannten Bewegung jeweils Referenzwerte für den Vergleich mit den aktuellen Werten vor. Für die Entscheidung, ob eine betrachtete Bewegung zu einer Kategorie gehört oder nicht, sind die Kosten entscheidend. Für das Ende des Clusterings wird ein Threshold definiert. Liegt der berechnete Wert über diesem, wird die Bewegung verworfen. Falls sie geringer sind, wird der Ablauf in die Kategorie einsortiert. Die Definition von geeigneten Threshold-Werten für die Differenz zwischen Datenreihen wird auch im späteren Verlauf dieser Arbeit relevant. So kann entschieden werden, ob Cluster weiter zusammengeführt werden sollten oder nicht. Yu und Xiong (2019) nutzten DTW, um physische Rehabilitationsübungen mithilfe eines Kinect-Sensors zu bewerten. Dabei wird zunächst die Ähnlichkeit zwischen den Bewegungsabläufen des Nutzers und des Trainers berechnet. Als Eingabeattribute werden hier acht sogenannte Knochenvektoren des menschlichen Skeletts und dessen Körperorientierung verwendet. Ein Knochen, der zwischen zwei Gelenkpunkten liegt, wird als 3D-Vektor im Raum interpretiert. Die Summe der Winkeldifferenzen zwischen allen korrespondierenden Knochenvektoren der beiden Bewegungsreihen dient als Distanz-Metrik. Die Ergebnisse zeigen, dass DTW effektiv eingesetzt werden kann, um automatisiert Bewegungsabläufe mit Vorlagen zu vergleichen und diese somit zu bewerten. Im Kontext der Interaktion mit Wandbildschirmen können unterschiedliche Eingabewerte interessant sein. In dieser Bachelorarbeit sollen die Attribute für den Vergleich, daher nicht bei der Implementierung festgelegt werden. Stattdessen sollen sie frei konfigurierbar sein, damit der Anwender das Clustering, je nach Fragestellung, mit verschiedenen Attributen durchführen kann. Keines der in der Literatur gezeigten Systeme bietet eine derartige Funktionalität. Mohammadzade et al. (2021) nutzen TSD-Informationen für eine Überwachung von alltäglichen menschlichen Aktivitäten. Jede Aktion wird durch eine mehrdimensionale Time-Series repräsentiert, wobei eine Dimension jeweils der Position eines Skelettpunkts im Verlauf der Zeit entspricht. Im vorliegenden Ansatz wird jeder Bewegungsklasse genau eine Referenzbewegung zugeordnet. Die Daten werden entsprechend mit diesen Referenzen verglichen. Dadurch sind viel weniger

Vergleiche nötig als bei anderen DTW-Methoden, wodurch die Rechenkomplexität abnimmt. Allerdings sind dafür vordefinierte Klassen mit teils künstlich erzeugten Beispielbewegungen notwendig. Wesentliches Ziel dieser Bachelorarbeit ist es allerdings dazu beizutragen, neue Informationen über mögliche Interaktionsverhalten mit interaktiven Wandbildschirmen zu gewinnen. Eine Definition im Voraus ist für dieses Szenario nicht zielführend. Daher soll ein Clustering-Verfahren implementiert werden, welches ohne vordefinierte Kategorien funktioniert. Hautamaki et al. (2008) vergleichen experimentell einige TSD Clustering-Verfahren mit DTW als Distanzmetrik. Unter anderem auch hierarchisches Clustering. Hier werden zunächst alle Records als eigenes Cluster initialisiert. Durch iterierte Anwendung von Merge-Operationen werden schrittweise zwei Cluster zusammengeführt, die eine geringe Differenz aufweisen. Hautamaki et al. (2008) prüfen die Zuverlässigkeit anhand mehrerer Experimente. Unter anderem verwenden sie Sprachdaten, die ebenfalls zu den TSD gehören. Die Evaluation zeigt, dass hierarchisches Clustering in Kombination mit DTW erfolgreich eingesetzt werden kann. Ein Versuch im Kontext von Bewegungsdaten erfolgt in Hautamaki et al. (2008) allerdings nicht.

3.5 Einsatz im Kontext von Kinect-Bewegungsdaten

Warren Liao (2005) und Aghabozorgi et al. (2015) weisen darauf hin, dass besonders die verwendete Distanzmetrik entscheidend ist, um TSD sinnvoll in Kategorien einteilen zu können. Die vorgestellten Arbeiten in Abschnitt 3.4 zeigen, dass hierarchisches Clustering in Kombination mit DTW ein vielversprechender Ansatz ist. Da die Daten in Textdateien vorliegen, ist eine Komponente im System nötig, die diese Informationen ausliest und in geeigneten Objekten abspeichert. Von besonderer Bedeutung ist zudem die Wahl geeigneter Attribute. Die Daten enthalten viele Werte, die mehr oder weniger interessant für die Kategorisierung sein können. Welche Werte für eine Analyse relevant sind, muss im Einzelfall überprüft werden. Die Wahl der genutzten Attribute muss also vom Nutzer definiert werden können. So kann das Tool effektiv genutzt werden, um nach wiederkehrenden Abläufen in den Daten zu suchen. Der zu definierende Threshold kann je nach Daten unterschiedlich ausfallen und sollte daher ebenfalls von Nutzer konfiguriert werden können. Abschließend ist die große Menge der Daten zu erwähnen. Aghabozorgi et al. (2015) weisen darauf hin, dass DTW weniger performant ist als herkömmliche Metriken. Gegebenenfalls müssen die Daten vor der Nutzung geeignet gefiltert werden. Diese Filterung ist allerdings nicht Bestandteil dieser Bachelorarbeit. Für die Evaluation in Kapitel 6 wurden bereits gefilterte Daten bereitgestellt.

4 Konzeption

Vor der eigentlichen Implementierung erfolgt nun die Konzeption. Damit wird elaboriert, welche Anforderungen an die Software gestellt werden und wie die Implementierung erfolgen muss. Etwaige Ungereimtheiten und Probleme können so frühzeitig erkannt werden.

4.1 Anforderungsanalyse

Es soll ein System implementiert werden, welches Kinect-Bewegungsdaten mithilfe von hierarchischem Clustering und DTW gruppiert. Es gibt sogenannte *funktionale Anforderungen* und *nicht-funktionale Anforderungen*, die das System erfüllen muss. Letztere beinhalten beispielsweise Anforderungen zu Benutzbarkeit, Effizienz oder Portierbarkeit. Funktionale Anforderungen beschreiben hingegen die primären Funktionen des Systems. Die folgenden Anforderungen sind im Wesentlichen durch Diskussionen im Projektkontext entstanden.

4.1.1 Nicht-funktionale Anforderungen

Das Tool soll ohne großen Installationsaufwand nutzbar sein und daher, nach Möglichkeit, eine stark verbreitete Programmiersprache nutzen. Bei der Implementierung fällt deshalb die Wahl auf Java in der Version 14. Um weitere Installationen zu vermeiden, sollen auch keine externen Bibliotheken genutzt werden. An die Effizienz gibt es keine besonderen Anforderungen, da die Auswertung der Daten nur einmalig erfolgt. Eine Echtzeitauswertung ist also nicht geplant. Zu lange Wartezeiten, schränken allerdings die Nutzbarkeit des Tools ein. Daher sollen berechnete Kosten in einer geeigneten Datenstruktur zwischengespeichert werden. So können im darauffolgenden Iterationsschritt Berechnungsschritte eingespart werden. Des Weiteren soll das Tool durch wenige Anpassungen auch auf Datensätze anderer Tiefenkameras angewandt werden können. Um dieses Ziel zu erreichen, wird die Software von Beginn an generisch entwickelt. Zu allen Bestandteilen des Tools existiert daher ein Interface. Bei Bedarf können so mit wenig Aufwand neue Implementierungen, die den Anforderungen anderer

Datensätze entsprechen, ergänzt werden. Zudem werden die Informationen zur Struktur des Datensatzes, durch eine konfigurierbare Datei an das Tool herangetragen.

4.1.2 Funktionale Anforderungen

Die Anwendung sollte die folgenden Funktionalitäten anbieten. Unter anderem ist sie für den Input zuständig. Ein Datensatz kann aus Dateien eingelesen und in passenden Datenobjekten abgespeichert werden. Als Input-Dateiformat werden Textdateien unterstützt. Die Unterstützung weiterer Dateiformate, wie CSV, muss bei Bedarf ergänzt werden. Das Tool soll ein Clustering mittels hierarchischem Clustering wie in Abschnitt 3.1 beschrieben anbieten. Dabei ist der Schwellwert für das Zusammenführen von zwei Clustern vom Nutzer konfigurierbar. Als Distanzmetrik wird DTW (Abschnitt 3.2) genutzt. Des Weiteren ist konfigurierbar, welche Attribute des Datensatzes für den Vergleich genutzt werden. Beispielsweise können Laufwege betrachtet werden. Zur Berechnung der Distanz zwischen zwei Punkten wird die euklidische Distanz verwendet. Allerdings können in der DTW-Implementierung weitere Funktionen ergänzt werden. Mithilfe eines Startparameters kann entschieden werden, welche dieser Funktionen für die Berechnung verwendet wird. Zudem wird der Output durch das Tool umgesetzt. Die gefundenen Cluster werden in einer Textdatei aufgelistet. Jedes Cluster erhält eine eindeutige Identifikationsnummer und eine Liste aller Records, die zusammengeführt wurden. Für ein erleichtertes Verständnis der Cluster wird auch eine primitive Visualisierung der Cluster angeboten. Die Konfigurierbarkeit wird durch eine *config-file* erreicht, die der Nutzer beim Start der Anwendung übergeben kann. Tabelle 4.1 zeigt deren Attribute.

4.2 Teilsysteme

Abbildung 4.1 kann das Paketdiagramm des Tools entnommen werden. Das Paket *model* enthält sowohl die Interfaces für Cluster, Records und Frames, als auch jeweils geeignete Implementierungen für den Kinect-Datensatz. Bei Bedarf können hier weitere Implementierungen für andere Datensätze ergänzt werden. Im *utility*-Paket befinden sich Interfaces für Reader und Writer. Auch hier sind geeignete Implementierungen bereitgestellt. Ein Reader für die Config-Datei, ein Reader für die Kinect-Daten und ein Writer für die gefundenen Cluster. Auch das Interface und die Implementierung für die Visualisierung befinden sich hier. Im Paket *calculating* befinden sich alle Klassen, die für das Clustern der Daten benötigt werden.

Attributname	Beschreibung
inputPath	Pfad zum Datensatz.
outputPath	Pfad zum Ablegen der Output-Dateien.
separator	Trennzeichen zwischen den Attributen.
datasetType	Art des vorliegenden Datensatzes.
threshold	Grenzwert für das Zusammenführen von Clustern.
attributes	Vorhandene Attribute.
usedAttributes	Attribute, die zum Vergleich genutzt werden.
distanceFunction	Bezeichnung der genutzten Distanzfunktion.
flipVisualization	Spiegelung der Visualisierung.
attributeForBodyIdentification	Bezeichnung des Körperidentifikationsparameters.
skipFrames	Performanzsteigerung durch ignorieren jedes 3. Frames.

Tabelle 4.1: Attribute der Configuration-Datei.

Es ist weiter untergliedert in *clustering* und *metric*. Letzteres liefert die Vergleichsmetrik für unser Clusterverfahren. Wieder liegen geeignete Interfaces vor, sodass die Anwendung bei Bedarf um weitere Metriken oder Clustering-Ansätze erweitert werden kann. Bereitsgestellt werden initial eine Implementierung mit hierarchischem Clustering und DTW als Metrik. Das Paket *processor* ist von zentraler Bedeutung, denn hier wird die Funktionalität des Tools zusammengeführt. Zunächst werden die benötigten Reader Instanzen initialisiert. Mit ihnen werden die Daten eingelesen. Anschließend wird das Clustering gestartet. Die Ergebnisse werden in der Ausgabedatei gespeichert und zusätzlich visualisiert.

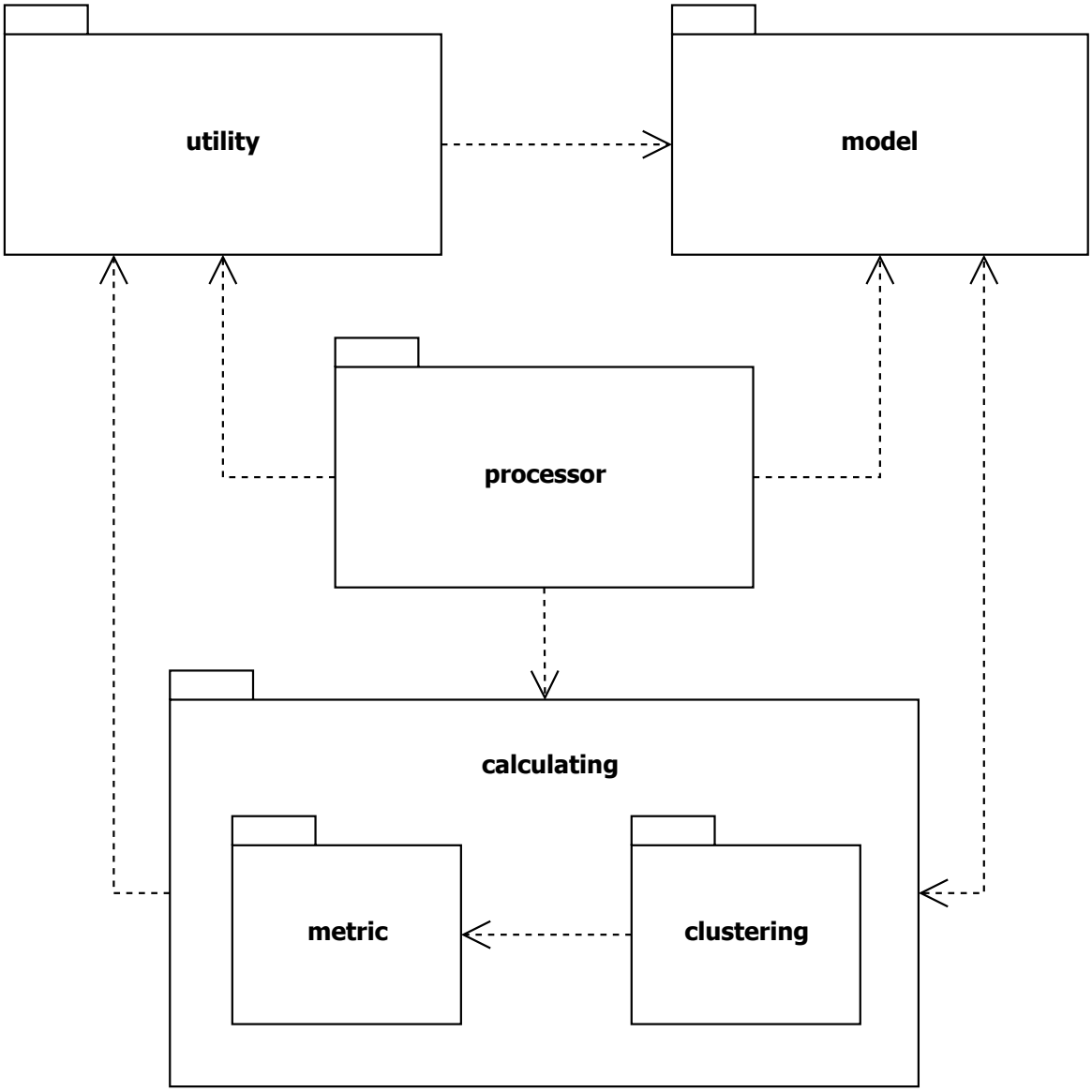

Abbildung 4.1: Paketdiagramm des Tools.

4.3 Programmablauf

Im Folgenden wird der Programmablauf beschrieben.

1. Der Nutzer startet den *Processor* und gibt als Startparameter den Pfad der *config-Datei* an.
2. Der Processor nutzt den *Config-Reader*, um die Werte der Konfigurationdatei auszulesen und speichert diese ab.
3. Der Processor nutzt den *DataReader*, um den Datensatz einzulesen und legt die Informationen in geeigneten Objekten ab. Für den Kinect-Datensatz sind die Implementierungen *RecordImpl* und *FrameImpl* bereitgestellt.
4. Der Processor startet das Clustering durch den Aufruf der cluster-Methode eines *HierarchicalClustering-Objekts*. Zu beginn entspricht jeder Record einem eigenen Cluster. Die recordToCluster-Methode ermöglicht eine Transformation von Records zu Clustern.
5. Das Clustering nutzt ein *DTW-Objekt*, um paarweise die geringsten Kombinationskosten zweier Cluster zu berechnen.
6. Die beiden Cluster mit den geringsten Kombinationskosten werden zusammengeführt. Konkret wird eines der Cluster in das andere integriert. Durch diese Kombination entsteht ein neues Cluster. Für alle Attribute werden die Wertereihen für das neue Cluster, aus denen der alten Cluster berechnet und abgespeichert. Zudem werden alle Bestandteile des neuen Clusters abgespeichert. Bei allen Komponenten wird ein boolescher Wert auf *false* gesetzt, damit sie bei zukünftigen Vergleichen nicht mehr berücksichtigt werden.
7. Schritte 5 und 6 werden so lange wiederholt, bis die geringsten Kosten den vom Nutzer definierten Threshold übersteigen oder nur noch ein Cluster übrig ist.
8. Der Processor nutzt den *Cluster-Writer*, um alle Cluster der clusters-Liste in eine Ausgabedatei zu schreiben.
9. Der Processor nutzt die *VisualizerImpl*, um alle Records der gefundenen Cluster zu visualisieren.
10. Die Anwendung terminiert.

5 Implementierung

Wie bereits erwähnt erfolgt die Implementierung mit Java. Damit für die Nutzung keine weiteren Installationen nötig sind, werden nur die Java-Standardbibliotheken zur Umsetzung genutzt. Der gesamte Source Code kann unter github.com/lzfs/tsprocessor abgerufen werden. In diesem Kapitel wird vertiefend auf die Struktur des Tools eingegangen. Zudem werden wichtige Aspekte der Implementierung betrachtet.

5.1 Aufbau

Der detailierte Aufbau des Tools kann dem Klassendiagramm in Abbildung 5.1 entnommen werden. Dabei wurde aus Gründen der Übersichtlichkeit auf Konstruktoren, Getter- und Setter-Methoden verzichtet. Die wichtigsten Aspekte der Software werden im Folgenden beschrieben. Die grobe Struktur entspricht der aus Abschnitt 4.2. Zum Start der Anwendung ist ein Aufruf der *main-Methode* des *Processor* nötig. Dieser initialisiert zunächst geeignete *Reader*. Das *Reader-Interface* wird von den Klassen *ConfigReader* und *DataReader* implementiert. Der *ConfigReader* dient lediglich dazu, die Werte der Config-Datei einzulesen. Dabei wird ein *Properties-Objekt* zurückgeliefert, mit dem auf die jeweiligen Werte zugegriffen werden kann. Der DataReader liest die Kinect-Daten ein. Ihm wird zunächst übermittelt, welche Attribute vorhanden sind, durch welches Zeichen sie voneinander abgetrennt sind und ob Frames übersprungen werden sollen, um die Performanz zu verbessern. Mit diesen Informationen können die Daten geeignet eingelesen und eine *RecordImpl-Liste* zurückgeliefert werden. Jede *RecordImpl* enthält dabei eine Liste von *FrameImpls*. Diese Datentypen sind die entsprechenden Implementierungen der Interfaces *Record* und *Frame*. Sie können für Kinect-Daten genutzt werden. Eine FrameImpl enthält jeweils eine Map mit ihren Attributen, welche mithilfe der *getValue-Methode* abgerufen werden können. Der letzte Datentyp ist das Cluster und die zugehörige *ClusterImpl*. Cluster bestehen aus beliebig vielen, aber mindestens einer RecordImpl. Die Bestandteile werden in einer Liste gespeichert. Hier werden die Methoden *mergeWithCluster* und *mergeWithRecord* angeboten. Mit ihnen wird jeweils

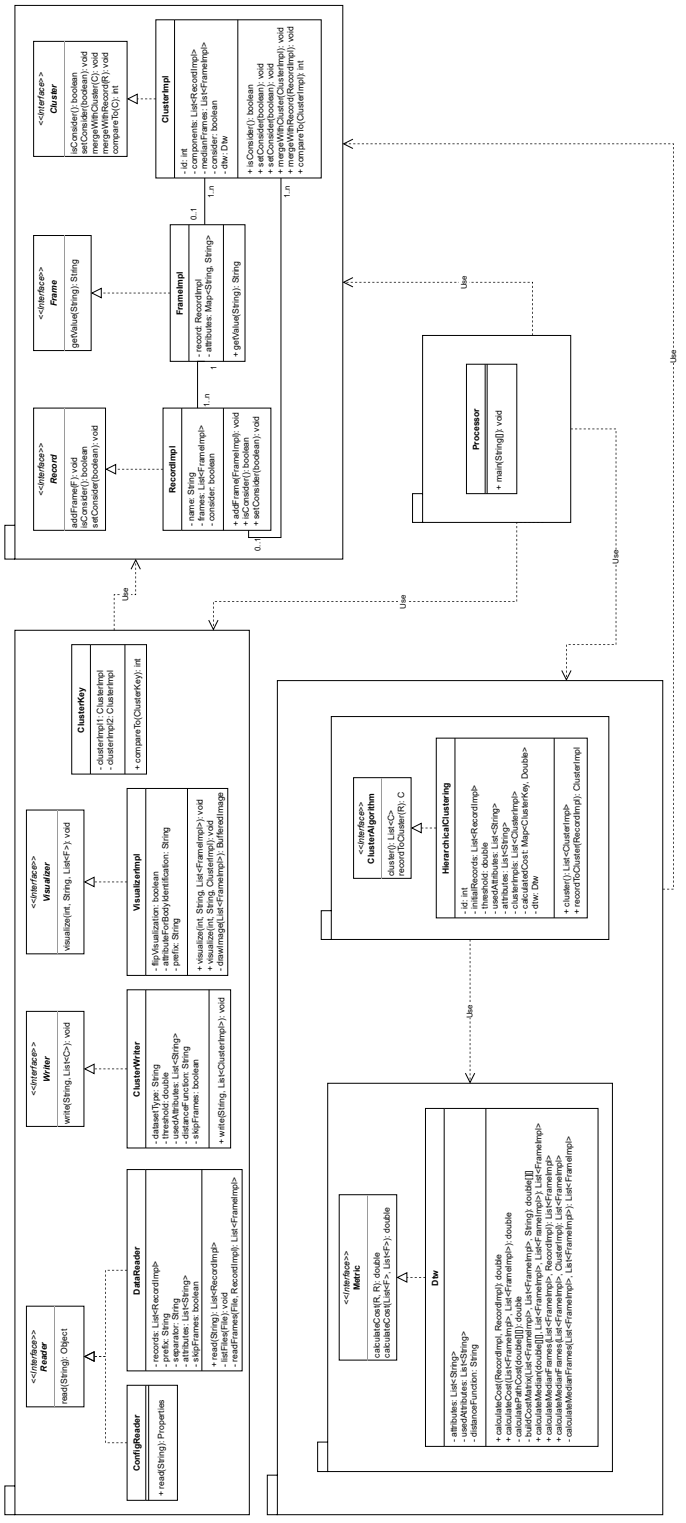

Abbildung 5.1: Klassendiagramm des Tools.

eine weitere Komponente in die ClusterImpl integriert. ClusterImpls haben außerdem eine eindeutige Identifikationsnummer und eine Liste der berechneten *FrameImpl-Mittelwerte* aller Komponenten. Zudem besitzen sie, genau wie die RecordImpls, einen Wahrheitswert, der angibt, ob die ClusterImpl in zukünftigen Kombinationsschritten weiterhin beachtet werden soll. Alle genannten Interfaces können bei Bedarf geeignet für andere TSD-Datensätze implementiert werden. Nach dem Einlesen und Abspeichern der Daten startet der Processor das Clustering. Dazu wird eine Instanz von *HierarchicalClustering*, einer Implementierung von *ClusterAlgorithm*, erstellt. Das Objekt erhält dafür alle wichtigen Informationen, wie die initialen RecordImpls, den Threshold und die Attribute vom Processor. Da jede ClusterImpl am Anfang aus einer einzigen RecordImpl besteht, wird die Methode *recordToCluster* angeboten. Mit ihr kann jeder der initialen RecordImpls transformiert und anschließend in der *clusterImpl-Liste* abgelegt werden. Die *cluster-Methode* kümmert sich um den eigentlichen Cluster-Vorgang. Dabei nutzt sie zur Berechnung von Kosten eine Instanz der *Dtw-Klasse*. Sie implementiert das *Metric-Interface* und bietet daher die nötigen Methoden, um die Kosten zwischen zwei RecordImpls, beziehungsweise deren *FrameImpl-Listen* zu berechnen. Dabei kommt der in Abschnitt 3.2 beschriebene DTW-Algorithmus zum Einsatz. Um auch eine ClusterImpl mit einer RecordImpls vergleichen zu können bietet die Klasse zudem entsprechende *calculateMedianFrames-Methoden*, die aus mehreren *FrameImpl-Listen* eine neue erzeugt, indem die Werte mithilfe von DTW kombiniert werden. Das genaue Verhalten der Klassen HierarchicalClustering und *Dtw* wird in Abschnitt 5.2 näher betrachtet. Nach dem Ende der Berechnungen liegen die gefundenen ClusterImpls in der *clusterImpl-Liste* vor. Sie werden mit dem *ClusterWriter* geeignet in einer Textdatei abgespeichert. Zudem erfolgt eine Visualiserung mithilfe der *VisualizerImpl*. Die Klasse nutzt dazu Java-AWT-Komponenten. Der Wahrheitswert *flipVisualization* gibt dabei an, ob die Werte in x-Richtung gespiegelt werden sollen. Dies ist nötig, da bei der Kinect das Koordinatensystem aus Nutzersicht erstellt wird. In der Visualiserung führt dies dazu, dass die aufgezeichneten Bewegungen spiegelverkehrt erscheinen. Der String *attributeForBodyIdentification* dient dazu, die unterschiedlichen Personen im Record korrekt darzustellen.

5.2 Codebeschreibung

Nach der Beschreibung des Ablaufs und der groben Struktur sollen nun wichtige Codeausschnitte des hierarchischen Clusterings und des DTW Algorithmus beschrieben werden. Objekte der Klassen ClusterImpl, RecordImpl und FrameImpl werden in diesem Abschnitt

vereinfacht als Cluster, Record und Frame bezeichnet. Bei der Ausführung des folgenden Codes wurden die Daten bereits eingelesen. Zudem wurde ein HierarchicalClustering-Objekt mit den nötigen Parametern erzeugt. Damit wird die cluster-Methode gestartet. Die Klasse HierarchicalClustering verfügt über eine clusterImpl-Liste, welche zunächst leer ist. Sie wird mit den initialen Clustern belegt, indem alle Records zunächst zu trivialen Clustern transformiert werden und anschließend der Liste hinzugefügt werden. Des Weiteren werden die nötigen Variablen für die Berechnung deklariert. Listing 5.1 zeigt diese Schritte. Der eigentliche Berechnungsablauf wird in Listing 5.2 gezeigt. Er befindet sich in einer Schleife, die mindestens einmal ausgeführt wird. Sie wird so lange fortgesetzt, bis die geringsten gefundenen Kosten den Threshold übersteigen, oder nur noch ein einziges Cluster übrig ist. Die Merge-Kandidaten geben die Indizes der beiden Cluster mit den geringsten Kombinationskosten an. In jedem Durchlauf der while-Schleife werden die Werte der Kosten und die gewählten Merge-Kandidaten zurückgesetzt. Dazu werden standardmäßig das erste und zweite Cluster in der Liste und deren Kosten gewählt. Im Anschluss werden mittels der geschachtelten Schleifen paarweise die Cluster miteinander verglichen. Dabei werden nur Cluster, deren *consider-Wert* noch *true* ist, in Betracht gezogen. Außerdem darf es sich beim Vergleich nicht zweimal um das gleiche Cluster handeln. Mithilfe der *calculatedCost-Map* wird geprüft, ob die Kosten für diese Kombination schon berechnet wurden. Falls nicht, werden sie neu berechnet. Als Schlüssel dient ein Objekt der *ClusterKey-Klasse*. Deren *equal-Methode* ist so definiert, dass *new ClusterKey(clusterImpl1, clusterImpl2).equals(new ClusterKey(clusterImpl2, clusterImpl1)) == true* gilt. Somit wird auch der symmetrische Fall beachtet. Immer wenn eine Kombination gefunden wird, deren Kosten geringer sind als die bisherigen, wird die aktuelle Belegung der Minimalkosten und der Merge-Kandidaten angepasst. Nach dem Vergleich aller Cluster, erfolgt die Kombination der beiden mit den geringsten Kombinationskosten. Dies ist in Listing 5.3 zu sehen. Die Indizes der beiden Cluster mit den geringsten Kombinationskosten sind in *mergeCandidate1* und *mergeCandidate2* abgelegt. Die zugehörigen Kosten sind in der Variablen *currentMinimumCost* abgelegt. Sind diese immer noch geringer als der definierte Threshold, können die Kandidaten zusammengeführt werden. Dazu wird *mergeCandidate2* mit der Methode *mergeWithCluster* in *mergeCandidate1* integriert. Dabei wird der *consider*-Wert des *mergeCandidate2*-Clusters auf *false* gesetzt, da es in künftigen Schritten nicht mehr als eigenständige Einheit betrachtet werden soll. Abschließend müssen alle berechneten Werte mit *cluster1* aus der *calculateCost-Map* entfernt werden, da die aktuellen Werte aufgrund der neuen Komponente nicht mehr gültig sind. Die Kosten mit *cluster2*

können ignoriert werden, da diese Cluster ohnehin nicht mehr betrachtet werden. In den oben gezeigten Clustering-Schritten wurden die Kosten mithilfe der *calculateCost-Methode* berechnet. Listing 5.4 zeigt diese. Für jedes der Attribute, das für die Berechnung verwendet werden soll, wird gemäß des DTW-Algorithmus eine Kostenmatrix aufgestellt (Listing 5.5). Für sie werden die minimalen Pfadkosten berechnet und zu den Gesamtkosten für diesen Attributvergleich addiert. Das Ergebnis wird am Ende durch die Anzahl der betrachteten Attribute dividiert, um auch vergleichbare Werte zu erhalten, wenn die Anzahl von Attributen variiert. Die Methode *buildCostMatrix* liefert die passende Kostenmatrix für ein gemeinsames Attribut zweier Frame-Listen gemäß des DTW-Algorithmus. Zunächst werden die beiden Attributreihen in Arrays kopiert. Anschließend wird die initiale Matrix erstellt. Die Zellen der Reihe und Spalte mit dem Index *0* werden mit dem Wert „unendlich“ belegt. Die Zelle *(0, 0)* und alle übrigen Zellen erhalten den Wert 0. Nun werden die Attributwerte paarweise verglichen. Hier wird standardmäßig die absolute Distanz verwendet. Bei Bedarf können an dieser Stelle weitere Distanzmetriken ergänzt werden. Zur berechneten Differenz wird jeweils das Minimum der angrenzenden, bereits berechneten Zellen addiert. Diese Summe wird in die entsprechende Matrixzelle eingetragen. Dadurch füllt sich die Matrix und kann nach Abschluss der Berechnungen zurückgegeben werden. Die Kosten des Warping Pfads der entsprechenden Matrix können mithilfe der Methode *calculatePathCost* berechnet werden. Das Verfahren startet in der letzten Zelle der Matrix und sucht den Pfad mit den geringsten Kosten, um die Zelle *(0, 0)* zu erreichen. Dabei werden die in Abschnitt 3.2 beschriebenen Vorgaben eingehalten. Die Kosten des Pfades werden aufsummiert und anschließend durch die Anzahl der betretenen Zellen geteilt. Diese Division ist nötig, um, im Falle von unterschiedlich großen Matrizen, vergleichbare Ergebnisse zu erhalten. Listing 5.6 veranschaulicht dieses Vorgehen.

```
1 // Initialize each record as a cluster and add it to the clusters list.
2 for (RecordImpl record : this.initialRecords) {
3     this.clusterImpls.add(this.recordToCluster(record));
4 }
5 double currentMinimumCost;
6 double cost;
7 int mergeCandidate1;
8 int mergeCandidate2;
```

Listing 5.1: Cluster-Methode: Initialisierung.

```
1   do {
2     // Reset for next loop iteration.
3     cost = this.dtw.calculateCost(
4         this.clusterImpls.get(0).getMedianFrames(),
5         this.clusterImpls.get(1).getMedianFrames());
6     currentMinimumCost = cost;
7     mergeCandidate1 = 0;
8     mergeCandidate2 = 1;
9     for (ClusterImpl clusterImpl1 : this.clusterImpls) {
10        for (ClusterImpl clusterImpl2 : this.clusterImpls) {
11            if (
12                clusterImpl1 != clusterImpl2
13                && clusterImpl1.isConsider()
14                && clusterImpl2.isConsider()) {
15                if (this.calculatedCost.containsKey(
16                    new ClusterKey(clusterImpl1, clusterImpl2))) {
17                    cost = this.calculatedCost.get(
18                        new ClusterKey(clusterImpl1, clusterImpl2));
19                }
20                else {
21                    cost = this.dtw.calculateCost(
22                        clusterImpl1.getMedianFrames(),
23                        clusterImpl2.getMedianFrames());
24                    this.calculatedCost.put(
25                        new ClusterKey(clusterImpl1, clusterImpl2), cost);
26                }
27                if (cost < currentMinimumCost) {
28                    /* If a cost smaller than the current minimum cost
29                    has been found it will be the new minimum. */
30                    currentMinimumCost = cost;
31                    mergeCandidate1 = this.clusterImpls.indexOf(clusterImpl1);
32                    mergeCandidate2 = this.clusterImpls.indexOf(clusterImpl2);
33                }
34            }
35        }
36    }
37    // Merging. Showed in next Listing.
38  } while (currentMinimumCost < threshold && this.clusterImpls.size() > 1);
```

Listing 5.2: Cluster-Methode: Berechnungsvorgang.

```
if (currentMinimumCost < threshold) {
    /* MergeCandidate1 and mergeCandidate2 should be merged together.
    Merge cluster2 into cluster1 and update the cluster list.
    Consider of cluster2 is set to false. */
    this.clusterImpls.get(mergeCandidate1).mergeWithCluster(
        this.clusterImpls.get(mergeCandidate2));
    this.clusterImpls.remove(this.clusterImpls.get(mergeCandidate2));
    // Create a copy of the map to avoid an exception.
    Map<ClusterKey, Double> calculatedCostCopy = new HashMap<>(calculatedCost);
    /* Remove all calculated costs from the map that contain
    cluster1 because it changed and therefore all cost values
    with this cluster have to be calculated again. */
    for (Map.Entry<ClusterKey, Double> entry : calculatedCostCopy.entrySet()) {
        /* Ignore mergeCandidate2 because it got removed from the list and
        the consider value of cluster2 is set to false. */
        if (entry.getKey().getCluster1().getId() == mergeCandidate1
              || entry.getKey().getCluster2().getId() == mergeCandidate1) {
            for (ClusterImpl clusterImpl : clusterImpls) {
                calculatedCost.remove(
                    new ClusterKey(
                        this.clusterImpls.get(mergeCandidate1), clusterImpl));
            }
        }
    }
}
```

Listing 5.3: Cluster-Methode: Mergevorgang.

```
public double calculateCost(
    List<FrameImpl> frames1,
    List<FrameImpl> frames2) {
    // Add the calculated cost of each attribute to it.
    double cost = 0;
    // Calculate the cost for all attributes.
    for (String attribute : this.usedAttributes) {
        // Build the cost matrix for this attribute.
        double[][] dtwMatrix = buildCostMatrix(frames1, frames2, attribute);
        // Calculate and add the cost of this attribute.
        cost += calculatePathCost(dtwMatrix);
    }
    return cost / this.usedAttributes.size();
}
```

Listing 5.4: DTW: Berechnung der Kosten.

```java
private double[][] buildCostMatrix(
    List<FrameImpl> frames1,
    List<FrameImpl> frames2,
    String attribute) {
    double[] s = new double[frames1.size()];
    double[] t = new double[frames2.size()];
    int counter = 0;
    for (FrameImpl frame : frames1) {
        s[counter] = Double.parseDouble(frame.getValue(attribute));
        counter += 1;
    }
    counter = 0;
    for (FrameImpl frame : frames2) {
        t[counter] = Double.parseDouble(frame.getValue(attribute));
        counter += 1;
    }
    int n = s.length;
    int m = t.length;
    // Filled with zeros by default.
    double[][] dtwMatrix = new double[n + 1][m + 1];
    for (int i = 0; i < dtwMatrix.length; i++) {
        for (int j = 0; j < dtwMatrix[0].length; j++) {
            // Filled with infinity by definition of dtw.
            dtwMatrix[i][j] = Double.POSITIVE_INFINITY;
        }
    }
    // Filled with zero by definition of dtw.
    dtwMatrix[0][0] = 0;
    for (int i = 1; i < dtwMatrix.length; i++) {
        for (int j = 1; j < dtwMatrix[0].length; j++) {
            double cost = Math.abs(s[i - 1] - t[j - 1]);
            double minTmp = Math.min(dtwMatrix[i - 1][j], dtwMatrix[i][j - 1]);
            double lastMin = Math.min(minTmp, dtwMatrix[i - 1][j - 1]);
            dtwMatrix[i][j] = cost + lastMin;
        }
    }
    return dtwMatrix;
}
```

Listing 5.5: DTW: Kostenmatrix.

```
public double calculatePathCost(double[][] dtwMatrix) {
    int n = dtwMatrix.length - 1;
    int m = dtwMatrix[n - 1].length - 1;
    int pathCounter = 1;
    double cost = dtwMatrix[n][m];
    while (n != 0 && m != 0) {
        double minTmp = Math.min(dtwMatrix[n - 1][m], dtwMatrix[n][m - 1]);
        double lastMin = Math.min(minTmp, dtwMatrix[n - 1][m - 1]);
        cost += lastMin;
        if (lastMin == dtwMatrix[n - 1][m - 1]) {
            n = n - 1;
            m = m - 1;
        }
        else if (lastMin == dtwMatrix[n - 1][m]) {
            n = n - 1;
        }
        else if (lastMin == dtwMatrix[n][m - 1]) {
            m = m - 1;
        }
        pathCounter += 1;
    }
    return cost / pathCounter;
}
```

Listing 5.6: DTW: Warping Path berechnen.

5.3 Abweichungen zur Konzeption

Alle im folgenden genannten Punkte waren in der ursprünglichen Konzeption nicht vorgesehen. Die Notwendigkeit der Anpassungen wird im Folgenden begründet.

An die Performanz der Anwendung wurden keine besonderen Anforderungen gestellt (Unterabschnitt 4.1.1). Es zeigte sich aber schnell, dass bereits Datensätze mit einigen Hundert Records zu langen Bearbeitungszeiten führen. Daher wurden zwei Konzepte umgesetzt, die sich diesem Problem annehmen. Zum einen werden die berechneten Kosten in einer Map abgespeichert, um wiederholte Berechnungen in darauffolgenden Schritten zu vermeiden. Als Key der jeweiligen Map-Einträge dient ein Objekt der Klasse ClusterKey. Sie implementiert das Comparable-Interface und sorgt dafür, dass zwei Cluster zusammen als Key einer Map verwendet werden können. Dies spart viele Berechnungsschritte, da beinahe alle Werte im Vergleich zum vorherigen Iterationsschritt gleichbleiben. Lediglich jene Kosten, an denen die neu zusammengeführten Cluster beteiligt waren, sowie die Kosten mit dem neuen Cluster müssen neu berechnet werden. Zum anderen gibt es die Möglichkeit jeden dritten Frame der Records zu ignorieren. Dies geschieht, indem der Parameter *skipFrames* in der config-Datei auf *true* gesetzt wird. Damit ist in der Regel weiterhin ein sinnvolles Clustering möglich, wobei die Anzahl der Berechnungsschritte reduziert werden kann. Es ist zu erwähnen, dass es dabei zu einem Informationsverlust kommt. Diese Option sollte daher nur bei Bedarf genutzt, wenn ansonsten beispielsweise der Zeitaufwand für die Berechnung zu hoch ist.

Zudem wurde abweichend zur Konzeption ein primitiver Visualizer ergänzt, da sonst die Analyse der Cluster manuell über Kinect Studio erfolgen muss. Er stellt die Laufwege der gefundenen Cluster aus der Vogelperspektive dar. Für jedes Cluster wird ein Verzeichnis erstellt. Darin werden die einzelnen Komponenten visualisiert.

6 Evaluation

Um die Praxistauglichkeit zu überprüfen, werden nun ausgewählte Daten mithilfe des entwickelten Tools geclustert. Die Ergebnisse werden anschließend bewertet. Zur Bewertung des Clusterings werden *externe* und *interne* Indizes unterschieden (Aghabozorgi et al., 2015; Warren Liao, 2005). Bei einem *externen Index* werden *Ground-Truth-Daten* extern bereitgestellt. Nach dem Clustering kann der Grad der Übereinstimmung zwischen gefundenen Clustern und den vorgegebenen Clustern überprüft werden (Aghabozorgi et al., 2015; Warren Liao, 2005). Die Daten werden mithilfe einer Visualisierung manuell geclustert. In Abschnitt 6.2 werden die gefundenen Cluster mit den erwarteten verglichen. Bei *internen Indizes* muss die Qualität der Cluster hingegen ohne zusätzliche Informationen überprüft werden (Aghabozorgi et al., 2015; Warren Liao, 2005). Dazu sollen in Abschnitt 6.3 *deskriptive Statistiken* genutzt werden. Die Standardabweichung, der Mittelwert, das Maximum und das Minimum liefern ein Maß für die Ähnlichkeit von Cluster-Komponenten. Damit wird überprüft, ob zusammengefasste Records tatsächlich einen hohen Grad an Übereinstimmung aufweisen.

6.1 Einführende Bemerkungen

Statt den gesamten, in Abschnitt 2.3 beschriebenen, Datensatz zu verwenden, sollen geeignete Teilmengen genutzt werden. Es wurde eine gefilterte Version des Datensatzes bereitgestellt. Zu kurze oder fehlerhafte Records wurden aussortiert (Abschnitt 2.3). Außerdem erfolgte eine Sortierung nach Personenzahl. Das Filtern der Daten gehört nicht zu den Anforderungen an das entwickelte Tool und wird daher nicht in dieser Arbeit betrachtet. Dennoch sollte dem Nutzer bewusst sein, dass mit gefilterten Daten gegebenenfalls aussagekräftigere Ergebnisse erzielbar sind. Gerade die separate Betrachtung von Records mit unterschiedlicher Personenzahl kann helfen, wiederkehrende Bewegungsmuster durch Clustering zu erkennen. In der Evaluation werden nur Records mit ein bis drei Personen betrachtet.

Ebenfalls zu erwähnen ist die Bedeutung eines geeigneten Threshold-Wertes für ein erfolgreiches Clustering. Dieser kann je nach vorliegenden Daten, verwendeten Attributen und

Zielstellung variieren. Häufig ist daher ein „Herantasten“ nötig. In den später beschriebenen Szenarien war beispielsweise auffällig, dass der Threshold bei Records mit einer Person deutlich niedriger war als bei Records mit mehreren Personen. Abhängig davon wie viele Cluster am Ende erwünscht sind und wie stark die Cluster-Bestandteile voneinander abweichen dürfen, muss ein anderer Wert gewählt werden.

Die Ground-Truth-Daten wurden manuell mithilfe der Visualisierung erstellt. Zunächst werden drei Minimalbeispiele betrachtet. Dafür wurden für die Fälle, eine bis drei Personen, jeweils beliebig 25 Records gewählt. Diese wurden mithilfe des Visualizers und Kinect Studio visualisiert und manuell in Cluster eingeteilt. Abbildung 6.1 zeigt jeweils die Visualisierung eines Repräsentanten der Cluster. Die Ergebnisse und eine sprachliche Beschreibung des Inhalts können im Evaluation-Verzeichnis des Code-Repositories eingesehen werden. Das hierarchische Clustering wurde mit den x- und z-Werten als Vergleichsattribute gestartet. Der Threshold wurde dabei jeweils so gewählt, dass genau so viele Cluster entstehen wie auch manuell erkannt wurden. Die Auswertung zeigt, dass die vom Tool berechneten Cluster genau mit den Ground-Truth-Daten übereinstimmen. Diese Minimalbeispiele sind aber nur bedingt aussagekräftig, da lediglich eine kleine Teilmenge der tatsächlich vorhandenen Daten genutzt wurde. Manche Muster treten dadurch gegebenenfalls überhaupt nicht auf. In Abschnitt 6.2 soll die Analyse daher auf größere Mengen gefilterter Daten, für ein bis drei Personen, ausgeweitet werden. Aufgrund der Vielzahl der Daten ist das manuelle Clustering allerdings nicht so zuverlässig möglich wie in den oben beschriebenen Minimalbeispielen. Die Daten wurden aber nach bestem Wissen, auf die wichtigsten Bewegungen hin, untersucht. Spezielle Bewegungen, die nur einmalig auftreten, wurden ignoriert. Ziel ist es zu überprüfen, ob häufig wiederkehrende Muster korrekt vom Tool erkannt werden. Abschließend bleibt zu erwähnen, dass beim Clustering von Bewegungsdaten, mithilfe von Laufwegen, der Zweck des Gehens offen bleibt (Monastero & McGookin, 2018). Dieser kann durch manuelle Interpretation bestimmt werden, wobei die gefundenen Cluster unterstützend eingesetzt werden können. Zur Automatisierung derartiger Interpretationen ist weitere Forschung notwendig.

6.2 Ground-Truth-Analyse

Im gefilterten Datensatz mit drei Personen befinden sich insgesamt 103 Records. Die manuelle Einteilung ergab, dass es sich bei vielen darin vorkommenden Bewegungsabläufen um spezielle Bewegungen, die nicht häufiger auftreten, handelt. Sie werden im Folgenden nicht aufgeführt. Allerdings wurde ein eindeutiges Cluster identifiziert. Abbildung 6.2 zeigt einen

Repräsentanten dieses Clusters. Es handelt sich dabei um drei Personen, die nebeneinander durch den Sensorbereich laufen. Dieselben Erkenntnisse lieferte auch das Minimalbeispiel aus Abschnitt 6.1. Grund hierfür kann sein, dass die Datenmenge mit circa 100 Records immer noch gering ist. Anschließend wurde überprüft, ob auch das Tool diese häufig auftretende Bewegung korrekt erkannt hat. Die Ergebnisse der Berechnungen stimmen mit dem manuell gefundenen Cluster überein. Im Wesentlichen wird nur ein nicht-triviales Cluster gefunden, und zwar das oben beschriebene.

Die Teilmenge aller Records mit zwei Personen enthält 513 Records. Bei der manuellen Durchsicht lassen sich wieder spezielle, nicht wiederkehrende Bewegungen erkennen. Diese werden erneut ignoriert. Im Gegensatz zum Fall mit drei Personen sind hier aber mehrere häufig auftretende Bewegungen identifizierbar. Abbildung 6.3 zeigt jeweils einen Repräsentanten dieser manuell erkannten Muster. Zum einen handelt es sich um den Fall, dass zwei Personen durch den Aufnahmebereich laufen. Dies ist die analoge Bewegung zum gefundenen Cluster bei drei Personen. Der zweite Fall zeigt, wie sich zwei Personen dem Sensor nähern. In Szenario drei bewegt sich eine Person zur Kamera, während eine andere lediglich durch das Bild läuft. Im letzten Fall laufen zwei Personen gemeinsam zum oberen Bildrand. Alle Szenarien werden durch das Tool gefunden. Bei dieser großen Menge an Records fällt auf, dass bei niedrigen Thresholds ähnliche Records teilweise noch in unterschiedlichen Clustern liegen und erst in späteren Schritten kombiniert werden, da andere Szenarien noch eine höhere Ähnlichkeit aufweisen.

Das Tool kann für Datensätze mit Ein- bis Zweitausend Records effizient eingesetzt werden. Auf einem heute typischen Mittelklasse-Heimrechner werden innerhalb weniger Minuten Ergebnisse geliefert. Ab einigen Tausend Records nimmt die Bearbeitungszeit zu, sodass, unter Umständen, mit mehreren Stunden Wartezeit zu rechnen ist. Die meisten hierarchischen Clustering-Verfahren werden den Komplexitätsklassen $O(n^2)$ oder $O(n^3)$ zugeordnet. Das implementierte Tool fällt in die Klasse $O(n^2)$, wie die Zeitmessungen aus Tabelle 6.1 belegen. Die Verdoppelung der Recordanzahl führt nicht zu einer Verachtfachung der Bearbeitungszeit, wie dies bei $O(n^3)$ der Fall wäre. Die Zeit zum Einlesen der Daten und zum Schreiben der Ergebnisse wurde hier ignoriert. Diese Limitierung muss beachtet werden, falls in Zukunft noch umfangreichere Datenmengen geclustert werden müssen. In diesem Fall muss entweder ein leistungsstärkerer Rechner eingesetzt oder das Tool weiter optimiert werden. Mögliche Optionen sind Parallelisierung oder die Implementierung einer effizienteren Version des Clustering-Verfahrens (Patel et al., 2015).

Recordanzahl	Bearbeitungszeit (Sekunden)
100	9
200	30
400	133
800	637

Tabelle 6.1: Bearbeitungszeiten des Clusteringvorgangs nach Recordanzahl.

Insgesamt befinden sich 3523 Records mit einer Person im gefilterten Datensatz. Allerdings wiederholen sich die Bewegungsmuster oft. Deshalb soll die Analyse im Folgenden auf eine Teilmenge mit 1000 beliebigen Records eingeschränkt werden. Erneut wird überprüft, ob wiederkehrende Bewegungen korrekt erkannt werden. Abbildung 6.4 zeigt jeweils einen Repräsentanten der manuell erkannten Muster. Im ersten Fall bewegt sich eine Person durch den Aufnahmebereich, während sie sich im zweiten Szenario unmittelbar vor der Kamera befindet. Bild drei zeigt ein direktes Zugehen auf den Bildschirm. Im vierten und sechsten Fall steht die Person an verschiedenen Positionen beinahe still. Szenario fünf zeigt, wie eine Person nach vorne läuft, sich umdreht und wieder zurückläuft. All diese Fälle können durch das Tool erkannt werden. Zu erwähnen ist allerdings, dass manche Cluster bei hohen Thresholds bereits in andere Gruppen integriert werden. So kann etwa das fünfte Szenario nur bei niedrigeren Thresholds erkannt werden, während es bei höheren Werten unter Umständen bereits mit Cluster eins zusammengeführt wurde. Dies verdeutlicht erneut, dass eine sinnvolle Suche nach Clustern möglich ist, diese aber dennoch manuell betrachtet werden müssen. Dabei sollte auch mit verschiedenen Thresholds experimentiert werden.

Die Auswertung des gesamten Datensatzes bestätigt demnach die Vermutung der Minimalbeispiele. Das Tool kann genutzt werden, um sinnvolle Cluster in den Kinect-Bewegungsdaten zu finden. Im folgenden Abschnitt soll abschließend die Güte der gefundenen Cluster an einigen Beispielen überprüft werden.

6.3 Statistische Analyse

Abschließend soll mithilfe von deskriptiven Statistiken überprüft werden, ob ein qualitativer Zusammenhang zwischen den Record-Bestandteilen eines Clusters besteht. Der Grad der Übereinstimmung liefert Informationen zur Ähnlichkeit. Konkret werden nun für die

Komponenten einiger Cluster die Standardabweichung, der Mittelwert, das Maximum und das Minimum für die x-Laufwege berechnet und tabellarisch dargestellt. In der letzten Zeile ist jeweils die Standardabweichung aller darüberstehenden Werte aufgeführt. Hier werden teilweise die Minimalbeispiele aus Abschnitt 6.2, mit veränderten Thresholds, wieder aufgegriffen. Auch diese Daten befinden sich im Evaluation-Verzeichnis des Code-Repositories. Zunächst wird das erste Cluster mit drei Personen betrachtet (Abbildung 6.1). Die Ergebnisse werden in Tabelle 6.2 dargestellt und sind auf zwei Nachkommastellen gerundet. Anschließend werden die Werte des ersten Clusters mit zwei Personen aus Abbildung 6.1 berechnet. Sie sind in Tabelle 6.3 zu sehen. Wird im selben Beispiel der Threshold noch höher gesetzt, sodass die gefundenen Cluster augenscheinlich nicht mehr sinnvoll sind, werden die Werte in Tabelle 6.4 geliefert. Die Mittelwerte und Standardabweichungen in den ersten beiden Beispielen bewegen sich jeweils in einem ähnlichen Rahmen. Auch die Standardabweichung dieser Werte ist mit *0.16* und *0.09* im ersten, sowie *0.16* und *0.08* im zweiten überprüften Cluster gering. Dadurch wird bestätigt, dass die Records einen inhaltlichen Zusammenhang aufweisen. Beim dritten Fall hingegen schwanken diese Werte stärker. Die ermittelte Gesamtstandardabweichungen sind mit *0.31* und *0.12* um circa 52 Prozent, bzw. 39 Prozent höher als bei den zuerst betrachteten Clustern. Einige Records scheinen also nicht in die Gruppe zu passen. So weicht beispielsweise die Zeile *2017-04-04 14.55.11.727* stark von den anderen ab. Ein Blick in die Visualisierung bestätigt, dass dieser Laufweg nicht zu den anderen passt. Dieser Fall wurde lediglich aufgrund des zu hohen Thresholds einsortiert. Dies betont einerseits erneut die bedeutende Rolle eines geeigneten Thresholds und andererseits, dass die Bestimmung der Güte mittels deskriptiver Statistiken möglich ist. Die Records der Cluster der beiden ersten Beispiele weisen eine hohe Ähnlichkeit auf. Mithilfe des entwickelten Tools ist folglich ein sinnvolles Clustering möglich. Beim letzten Cluster sind durch einen hohen Threshold hingegen zu viele Records zusammengeführt worden.

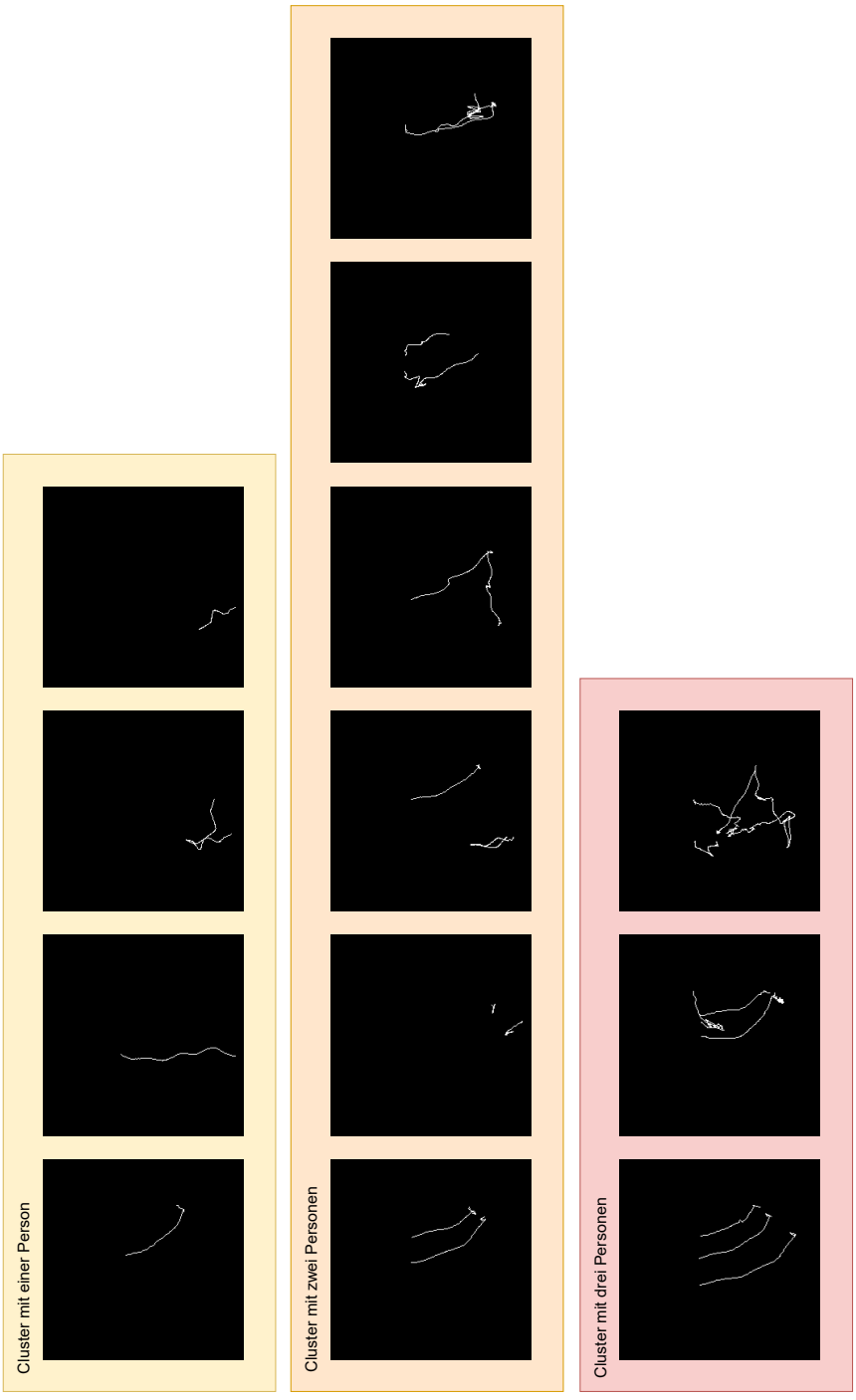

Abbildung 6.1: Repräsentanten der manuell erstellten Cluster.

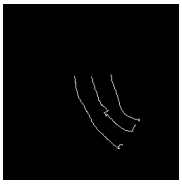

Abbildung 6.2: Repräsentant des nicht trivialen Clusters mit drei Personen.

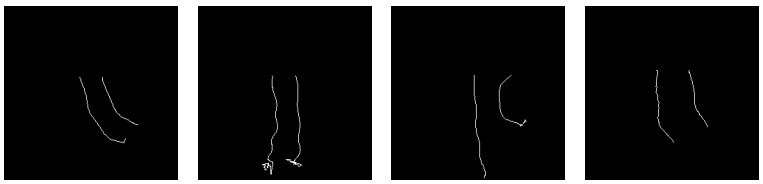

Abbildung 6.3: Repräsentanten der größten Cluster mit zwei Personen.

Abbildung 6.4: Repräsentanten der größten Cluster mit einer Person.

Record	Minimum	Maximum	Mittelwert	Standardabw.
2017-05-04 16.35.15.463	-2.12	0.63	-0.59	0,72
2017-04-13 12.34.38.535	-1.72	0.46	-0.65	0,58
2017-05-08 12.30.56.954	-2.09	0.12	-1.01	0.56
2017-04-13 12.36.06.669	-1.83	0.43	-0.70	0.57
2017-04-20 12.35.56.578	-2.06	0.50	-0.75	0.67
2017-04-10 11.53.16.991	-2.09	-0.61	-1.15	0.36
2017-05-08 12.45.55.652	-1.84	0.51	-0.64	0.65
2017-04-06 09.47.43.189	-1.96	0.44	-0.66	0.64
2017-04-05 13.00.04.745	-1.79	0.17	-0.79	0.50
2017-04-11 13.29.55.196	-2.06	0.72	-0.47	0.72
2017-04-07 13.19.05.395	-2.01	0.32	-0.81	0.56
2017-04-21 14.17.23.301	-2.03	0.30	-0.91	0.59
2017-04-26 11.59.37.413	-1.74	0.31	-0.76	0.55
2017-04-21 12.58.27.894	-1.84	0.20	-0.81	0.51
2017-04-19 12.54.19.390	-1.81	0.66	-0.54	0.69
2017-04-27 12.36.45.995	-1.85	-0.04	-0.92	0.50
2017-05-04 13.13.17.884	-2.07	-0.03	-0.85	0.52
2017-04-05 09.28.25.562	-1.78	0.36	-0.66	0.58
2017-05-04 13.24.36.117	-2.00	0.38	-0.66	0.65
2017-04-05 13.01.44.147	-2.20	0.41	-0.73	0.63
2017-04-07 11.05.51.351	-1.89	0.32	-0.63	0.65
Standardabweichung	0.14	0.29	0.16	0.09

Tabelle 6.2: Deskriptive Statistiken für das Cluster mit drei Personen.

Record	Minimum	Maximum	Mittelwert	Standardabw.
2017-04-06 12.26.06.199	-1.90	-0.34	-0.98	0,49
2017-04-05 09.13.53.631	-1.73	0.01	-0.88	0,41
2017-04-04 16.54.30.868	-1.67	0.45	-0.61	0.61
2017-04-05 12.32.59.075	-1.71	0.30	-0.62	0.54
2017-04-05 12.53.33.743	-1.80	0.28	-0.67	0.55
2017-04-05 10.03.28.598	-1.89	-0.02	-1.00	0.51
2017-04-05 08.16.53.128	-1.77	-0.03	-0.88	0.48
2017-04-05 14.44.27.088	-1.87	0.31	-0.72	0.60
2017-04-06 12.31.41.933	-1.77	0.28	-0.70	0.61
2017-04-05 15.01.29.988	-1.95	0.58	-0.50	0.68
2017-04-04 17.27.46.240	-1.93	-0.32	-1.10	0.38
2017-04-05 15.02.00.252	-2.13	0.26	-0.83	0.65
2017-04-05 17.08.48.758	-1.76	0.30	-0.73	0.64
2017-04-05 16.59.22.830	-1.92	0.38	-0.74	0.57
2017-04-05 11.40.51.237	-1.32	0.17	-0.49	0.46
2017-04-06 11.38.42.632	-1.67	0.12	-0.81	0.50
2017-04-05 10.56.51.141	-1.77	0.31	-0.68	0.58
2017-04-05 12.29.05.007	-1.85	0.16	-0.80	0.55
2017-04-05 13.33.56.412	-1.97	0.31	-0.79	0.63
2017-04-05 10.59.42.469	-1.77	-0.17	-0.92	0.49
Standardabweichung	0.16	0.24	0.16	0.08

Tabelle 6.3: Deskriptive Statistiken für das Cluster mit zwei Personen.

Record	Minimum	Maximum	Mittelwert	Standardabw.
2017-04-06 12.26.06.199	-1.90	-0.34	-0.98	0,49
2017-04-05 09.13.53.631	-1.73	0.01	-0.88	0,41
2017-04-04 16.54.30.868	-1.67	0.45	-0.61	0.61
2017-04-05 12.32.59.075	-1.71	0.30	-0.62	0.54
2017-04-04 18.04.06.507	-1.23	1.07	-0.14	0.54
2017-04-05 12.53.33.743	-1.80	0.28	-0.67	0.55
2017-04-05 10.03.28.598	-1.89	-0.02	-1.00	0.51
2017-04-05 08.16.53.128	-1.77	-0.03	-0.88	0.48
2017-04-05 12.40.57.142	-1.52	-0.23	-1.00	0.22
2017-04-05 14.44.27.088	-1.87	0.31	-0.72	0.60
2017-04-06 12.31.41.933	-1.77	0.28	-0.70	0.61
2017-04-05 15.01.29.988	-1.95	0.58	-0.50	0.68
2017-04-04 17.27.46.240	-1.93	-0.32	-1.10	0.38
2017-04-04 14.55.11.727	**-1.57**	**1.02**	**0.36**	**0.84**
2017-04-05 15.02.00.252	-2.13	0.26	-0.83	0.65
2017-04-05 17.08.48.758	-1.76	0.30	-0.73	0.64
2017-04-05 16.59.22.830	-1.92	0.38	-0.74	0.57
2017-04-05 11.40.51.237	-1.32	0.17	-0.49	0.46
2017-04-05 11.41.21.472	-1.08	-0.14	-0.53	0.36
2017-04-06 11.38.42.632	-1.67	0.12	-0.81	0.50
2017-04-05 10.56.51.141	-1.77	0.31	-0.68	0.58
2017-04-05 12.29.05.007	-1.85	0.16	-0.80	0.55
2017-04-05 13.33.56.412	-1.97	0.31	-0.79	0.63
2017-04-05 10.59.42.469	-1.77	-0.17	-0.92	0.49
Standardabweichung	0.24	0.35	0.31	0.12

Tabelle 6.4: Deskriptive Statistiken für das Cluster mit zwei Pers. (hoher Threshold).

7 Fazit

Im Rahmen des HoPE-Projekts wird das Verhalten von Menschen bei der Nutzung von interaktiven Wandbildschirmen erforscht. Dazu muss ein methodisches Rahmenwerk entwickelt werden, welches eine auf Sensordaten-basierende, automatische und zeitlich uneingeschränkte Evaluation von Ambient Displays ermöglicht (UniBw, 2021). Bewegungsdaten von Tiefenkameras können genutzt werden, um das Verhalten von Menschen vor Wandbildschirmen zu analysieren. Es stellt sich die Frage, ob sich Kategorien identifizieren lassen, die etwas über das Verhalten von Menschen vor Wandbildschirmen aussagen. Bei großen Datensätzen ist eine manuelle Identifikation nicht möglich. Daher ist eine Automatisierung notwendig. Diese Bachelorarbeit ist ein Beitrag zum Rahmenwerk für die Evaluation. Wesentliches Ziel war die Implementierung eines Systems zur Kategorisierung von vorliegenden *Kinect-Bewegungsdaten*. Das Problem wurde mithilfe von deterministischen Algorithmen gelöst. Konkret kam *hierarchisches Clustering* mithilfe des *DTW*-Algorithmus zum Einsatz. Diese Algorithmen sind für den Umgang mit TSD geeignet. In der Evaluation wurde die Praxistauglichkeit des Tools überprüft. Dazu wurden gefilterte Teilmengen des vorliegenden Datensatzes geclustert. Es hat sich gezeigt, dass ein geeigneter Threshold von Bedeutung ist. Die durchgeführte Ground-Truth-Analyse bestätigt, dass ein sinnvolles Clustering möglich ist. Die deskriptiven Statistiken belegen zudem, dass die Qualität der gefunden Cluster hoch ist. Mit den genutzten Algorithmen lassen sich Bewegungsdaten also ohne Vorwissen clustern. Zu erwähnen ist allerdings, dass beim Clustering von Bewegungsdaten der Zweck der Bewegung offen bleibt (Monastero & McGookin, 2018). Dieser kann durch manuelle Interpretation bestimmt werden. Zur Automatisierung dieses Schritts ist weitere Forschung notwendig. In zukünftigen Arbeiten kann außerdem versucht werden die Effizienz des Tools weiter zu erhöhen, um die Nutzbarkeit bei großen Datensätzen zu steigern. Zudem ist der Einsatz anderer Distanzfunktionen denkbar. Abhängig von den Daten können mit anderen Funktionen gegebenenfalls die Ergebnisse verbessert werden. Zuletzt können in künftigen Forschungsarbeiten, mithilfe der Anwendung, weitere Datenanalysen stattfinden.

Abbildungsverzeichnis

Abkürzungsverzeichnis

HCI	Mensch-Computer-Interaktion
RGB	Rot-Grün-Blau
ToF	Time-of-Flight
DTW	Dynamic Time Warping
TSD	Time-Series Data

Listingsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Aghabozorgi, S., Seyed Shirkhorshidi, A., & Ying Wah, T. (2015). Time-series clustering – A decade review. *Information Systems*, 16–38. https://doi.org/10.1016/j.is.2015.04.007

Ali, M., Alqahtani, A., Jones, M. W., & Xie, X. (2019). Clustering and Classification for Time Series Data in Visual Analytics: A Survey. *IEEE Access*, 181314–181338. https://doi.org/10.1109/ACCESS.2019.2958551

Hautamaki, V., Nykanen, P., & Franti, P. (2008). Time-series clustering by approximate prototypes. *2008 19th International Conference on Pattern Recognition*, 1–4. https://doi.org/10.1109/ICPR.2008.4761105

Li, L., et al. (2014). Time-of-flight camera–an introduction [SLOA190B]. *Technical white paper*. Verfügbar 10. März 2022 unter https://www.ti.com/lit/wp/sloa190b/sloa190b.pdf

Mai, C., & Hußmann, H. (2018). The Audience Funnel for Head-Mounted Displays in Public Environments. *2018 IEEE 4th Workshop on Everyday Virtual Reality (WEVR)*, 5.

Mankoff, J., Dey, A. K., Hsieh, G., Kientz, J., Lederer, S., & Ames, M. (2003). Heuristic Evaluation of Ambient Displays. *NEW HORIZONS*, 169–176. https://doi.org/10.1145/642611.642642

Marin, G., Agresti, G., Minto, L., & Zanuttigh, P. (2019). A multi-camera dataset for depth estimation in an indoor scenario. *Data in Brief*, 104619. https://doi.org/10.1016/j.dib.2019.104619

Microsoft. (2014). Human Interface Guidelines v2.0 [Windows-Developer-Center Microsoft Corporation]. Verfügbar 1. März 2022 unter https://download.microsoft.com/download/6/7/6/676611B4-1982-47A4-A42E-4CF84E1095A8/KinectHIG.2.0.pdf

Mohammadzade, H., Hosseini, S., Rezaei-Dastjerdehei, M. R., & Tabejamaat, M. (2021). Dynamic Time Warping-Based Features With Class-Specific Joint Importance Maps for Action Recognition Using Kinect Depth Sensor. *IEEE Sensors Journal*, 9300–9313. https://doi.org/10.1109/JSEN.2021.3051497

Monastero, B., & McGookin, D. K. (2018). Traces: Studying a Public Reactive Floor-Projection of Walking Trajectories to Support Social Awareness. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–13. https://doi.org/10.1145/3173574.3174061

Patel, S., Sihmar, S., & Jatain, A. (2015). A study of hierarchical clustering algorithms. *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 537–541.

Plischke, T. (2022). *Noch nicht abgeschlossen* (Masterarbeit). Universität der Bundeswehr München.

Schwarzer, J., Draheim, S., von Luck, K., Wang, Q., & Grecos, C. (2021). Spontaneous Utilization: A Classic Grounded Theory of Utilizing Ambient Displays in Professional, Large-Scale Agile Software Development Environments. *International Journal of Human–Computer Interaction*, 1–24. https://doi.org/10.1080/10447318.2021.2002051

Temiz, J. (2022). *Konzeption und Implementierung eines Datenanalyse-Werkzeugs für Body-Tracking-Kameras* (Bachelorarbeit). Universität der Bundeswehr München.

Tölgyessy, M., Dekan, M., Chovanec, L., & Hubinský, P. (2021). Evaluation of the Azure Kinect and Its Comparison to Kinect V1 and Kinect V2. *Sensors*, 413. https://doi.org/10.3390/s21020413

UniBw. (2021). Honeypot-Effekt an interaktiven Ambient Displays (HoPE) — Inf2. Verfügbar 28. Februar 2022 unter https://www.unibw.de/inf2/forschung/projekte/hope

Wahyuni, S., Rismayani, Intan, I., Ahyuna, A., & Suryani, S. (2021). Motion Recognition System with Dynamic Time Warping Method using Kinect Camera Sensor. *2021 3rd International Conference on Cybernetics and Intelligent System (ICORIS)*, 1–4. https://doi.org/10.1109/ICORIS52787.2021.9649520

Warren Liao, T. (2005). Clustering of time series data—a survey. *Pattern Recognition*, 1857–1874. https://doi.org/10.1016/j.patcog.2005.01.025

Wouters, N., Downs, J., Harrop, M., Cox, T., Oliveira, E., Webber, S., Vetere, F., & Vande Moere, A. (2016). Uncovering the Honeypot Effect: How Audiences Engage with Public Interactive Systems. *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, 5–16. https://doi.org/10.1145/2901790.2901796

Yu, X., & Xiong, S. (2019). A Dynamic Time Warping Based Algorithm to Evaluate Kinect-Enabled Home-Based Physical Rehabilitation Exercises for Older People. *Sensors*, 2882. https://doi.org/10.3390/s19132882

Hiermit versichere ich, die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, die Zitate ordnungsgemäß gekennzeichnet und keine anderen, als die im Literatur/Schriftenverzeichnis angegebenen Quellen und Hilfsmittel benutzt zu haben.

Ferner habe ich vom Merkblatt über die Verwendung von studentischen Abschlussarbeiten Kenntnis genommen und räume das einfache Nutzungsrecht an meiner Bachelorarbeit der Universität der Bundeswehr München ein.

München, den 31. Mai 2022

.....

(Unterschrift)