Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. -143

Ein interaktiver Ansatz zur visuellen Analyse von Trajektorien basierend auf einem Partition-and-Group Clustering Framework

Philipp Göttlich

Studiengang:	Informatik
Prüfer:	Prof. Dr. Thomas Ertl, Prof. Dr. Johannes Maucher
Betreuer:	M.Sc. Robert Krueger
begonnen am:	17. Juni 2014
beendet am:	17. Dezember 2014
CR-Klassifikation:	D.2.2, D.2.6, H.2.8, H.3.3, I.4.6, I.5.2, I.5.3, I.5.5

Inhaltsverzeichnis

Abbildungsverzeichnis

Kurzfassung

Seit Bewegungen über GPS-Sensoren besser erfasst werden können, steigt die Menge an verfügbaren Bewegungsdaten rapide an. Diese können potentiell viele Verhaltensmuster aufweisen, welche wertvolle Erkenntnisse zur Erforschung oder zum Schutz der sich bewegenden Objekte liefern. Um diese Muster zu erkennen, müssen die unstrukturierten Daten verarbeitet und anschließend passend visualisiert werden. Das ermöglicht hinterher die Analyse der Muster. Die Transparenz des Prozesses und die Unterstützung des Nutzers bei der Analyse sollen dabei gewährleistet werden, weshalb die Möglichkeiten der Interaktion eine wichtige Rolle spielen. Bisher beschränken sich diese interaktiven Möglichkeiten bei der Auswertung von Bewegungsdaten auf ein Minimum und werden meist nur zur Exploration von Verarbeitungsergebnissen genutzt. In dieser Arbeit wird deshalb ein System entwickelt, welches automatische Datenverarbeitung, durch interaktive Methoden, fest mit der Visualisierung verknüpft, um die Interaktions- und Analysemöglichkeiten zu verbessern. Dafür werden eine Partitionierungsmethode, ein Cluster-Algorithmus und ein einstellbares Distanzmaß über eine durchgängige, interaktive Visualisierung vereint. Dadurch soll Analysten das Verstehen und Steuern der Musterextraktion ermöglicht und die abschließende Analyse der Muster erleichtert werden. Um aufzuzeigen, was diesen Ansatz von bisherigen unterscheidet, werden die genutzten Komponenten im Detail erklärt, Neuerungen vorgestellt und auf Vor- und Nachteile eingegangen.

Abstract

Since movements can be tracked easier by the help of GPS-sensors, the quantity of movement data rises rapidly. These can potentially exhibit many behavioural patterns which provide valuable insights for investigation or protection of moving objects. In order to find such patterns, the unstructered data must be processed and visualised. This enables their following analysis. To increase the transparency of this process and to support the user in the analysis, the opportunities of interaction are an important issue. So far, interactive actions for the visual analysis of movement data are pared down to a minimum and are used only for the exploration of the results of the processing. That is why, in this thesis, we develop a framework which firmly connects automatic data processing through interactive methods with visualisation in order to improve the capabilities of analysis and interaction. Therefore, a partition method, a cluster algorithm, and an adjustable distance function are combined by a continuous interactive visualisation in order to support analysts towards the understanding and directing of the extraction of patterns and the concluding analysis. To show what makes this approach distinguishable from existing approaches, the utilised components will be explained in detail, innovations will be illustrated, and advantages and disadvantages will be explained.

1 Einleitung

Durch die Analyse von aufgezeichneten Bewegungsdaten, sogenannten Trajektorien, können eine Vielzahl von Informationen gewonnen werden, welche sich für Optimierungen nutzen lassen. Zum Beispiel durch das Analysieren von Fahrzeug-Trajektorien können Schwachstellen in Straßennetzen entdeckt werden. Nimmt man dagegen Tierbewegungen, so kann das Verhalten von Tieren untersucht werden. Anhand von Laufwegen von Messebesuchern können Routen und Standorte optimiert werden und durch Hurrikan-Pfad-Analysen kann man Vorhersagen treffen um Schutzmaßnahmen einzuleiten. Dies sind nur wenige Beispiele, denn Trajektorienanalysen lassen sich für alle Objekte durchführen, deren Bewegungen aufzeichenbar sind. Die Erkenntnisse, welche durch solche Analysen gewonnen werden, sind demnach von großer Wichtigkeit, wenn man den Nutzen bedenkt, den die Erkenntnisse liefern können. Für diese Arten von Analysen werden bisher meistens Data-Mining Techniken verwendet, welche bestimmte Muster in den Daten lokalisieren und ausgeben. Eine dieser Techniken ist das „Clustering“ von Daten. Dabei handelt es sich um einen Prozess, bei dem Objekte anhand ihrer Ähnlichkeit in Klassen gruppiert werden [15]. Ein Nachteil dabei ist, dass die Nutzer[1] keine Übersicht über die Vorgänge der Analyse erhalten. Es besteht die Gefahr, dass sie den Ergebnissen blind vertrauen. Für viele Aufgaben sind solche vollautomatischen Lösungen völlig ausreichend, wie zum Beispiel für das Auffinden von Ballungsräumen oder um Texte nach ihrer Ähnlichkeit zu gruppieren.
Sobald eine Aufgabe aber die kognitiven Fähigkeiten eines Menschen erfordert, insbesondere in heiklen Situationen in denen es um Menschenleben geht, reichen vollautomatische Lösungen nicht mehr aus. Bei der Verbreitung von Waldbränden beispielsweise benötigt man das Detailwissen von Brandschutzexperten. Das liegt daran, dass die Ergebnisse interpretiert werden müssen. Manche Gebiete sind bei einem Brand gefährdeter als andere. Diese müssen sofort evakuiert werden. Ein Experte ist sich dessen bewusst und kann melden, wie und wo der Brand zu bekämpfen ist. Einer Maschine dies beizubringen ist ein langer komplexer Prozess des maschinellen Lernens, welcher trotzdem fehlerbehaftet sein kann. In einer solchen Situation hätte das fatale Folgen. Die menschlichen Erfahrungen einzubringen erleichtert deshalb die Analyse und sorgt für verlässlichere Ergebnisse. Vor allem bei Bewegungen sind die kognitiven Fähigkeiten des Menschen sehr wichtig. Maschinen wissen von Grund auf nicht, was Bewegungen sind und wie diese zu verstehen oder zu deuten sind, weshalb sie diese nicht interpretieren können.

Die Visualisierung ist eine andere Möglichkeit um Bewegungsdaten zu analysieren. Ein Analyst kann die Darstellung der Daten auswerten. Möchte man einfach Bewegungen zeichnen, um sie nachzuvollziehen, oder Wege darin zu finden, ist eine Visualisierung der automatischen Verarbeitung vorzuziehen. Jedoch fallen bei der Analyse von Trajektorien oft große Datenmengen an, welche ohne maschinelle Hilfe schwer zu überblicken sind. Die visuelle Darstellung von Bewegungen, oder Daten im Allgemeinen, appelliert an die visuellen

[1] Im weiteren Textverlauf wird nur der Begriff „Nutzer“ verwendet, um einen besseren Lesefluss zu gewährleisten. „Nutzerinnen“ sind darin mit inbegriffen. Dies gilt auch für alle weiteren Beschreibungen von Anwendern oder Analysten in dieser Arbeit.

Fähigkeiten des Menschen. Der visuelle Sinneskanal ist bei Menschen besonders gut ausgeprägt, was ihnen dabei hilft, Vorgänge und Daten besser zu verstehen, zu verarbeiten und daraus zu lernen [25]. Eine Visualisierung kann also bei der Analyse von Bewegungsdaten helfen, jedoch ist es schwer für Nutzer, große Datenbestände rein visuell zu analysieren, da sie kaum Anhaltspunkte haben, worauf sie ihre Aufmerksamkeit lenken sollen.

Um die Vorteile beider Methoden zu vereinen, gibt es seit einigen Jahren Ansätze, welche beide Methoden interaktiv miteinander verknüpfen. Diese interdisziplinären Verfahren fallen in den Bereich der Visual Analytics[2] [30]. Sie bestehen aus drei Komponenten, wovon eine für die automatische Datenverarbeitung und eine andere für die Visualisierung zuständig ist. Die dritte Komponente ist die Modellbildung, welche das Data-Mining und das maschinelle Lernen umsetzt [17]. In Abbildung 1.1 ist das VA-Modell von Keim et al. dargestellt, in welchem diese Komponenten eingezeichnet sind.

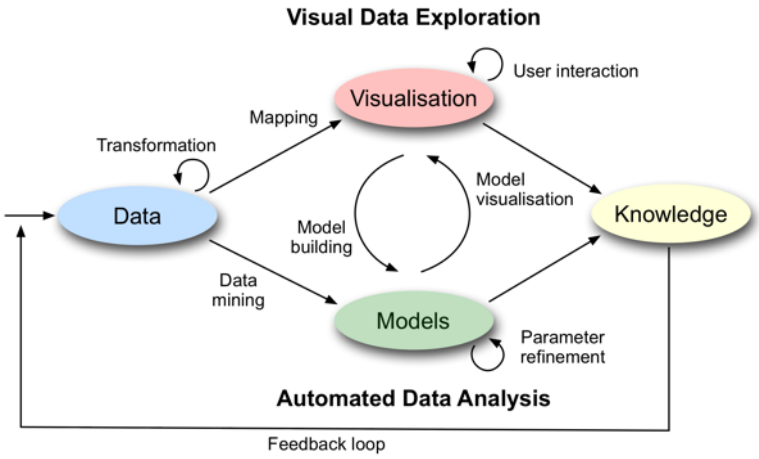


Abbildung 1.1: VA-Modell von Keim et al. [18]

Die Visualisierung fungiert dabei, zusammen mit der Interaktion, als verbindendes Element zwischen Datenverarbeitung und Wissensgewinnung. Viele der bisherigen Ansätze in diesem Gebiet trennen die Komponenten aber strikt voneinander ab, sodass die visuelle Komponente nur zur nachträglichen Exploration der Ergebnisse verwendet wird. Dies bedeutet, dass der Nutzer während der gesamten Dauer der automatischen Datenverarbeitung kein visuelles Feedback und keine Interaktionsmöglichkeiten bekommt, bis die Ergebnisse feststehen. Damit ist das Potential solcher VA-Ansätze also noch lange nicht ausgeschöpft.
Durch die interaktive Einbindung von Benutzern in den automatischen Prozess werden viel mehr Anwendungsmöglichkeiten und feinere Analysen möglich. Außerdem haben Nutzer stets die Übersicht über die laufenden Prozesse. Wenn bei einer automatischen Datenanalyse ein Parameter nicht genau genug ist, so wird dies erst am Ende der Analyse, wenn überhaupt, festgestellt. Die Dauer der fehlerhaften Berechnung geht dadurch verloren.
Bei einem interaktiven VA-Ansatz dagegen könnte man die Idee umsetzen, dass Benutzer schon während der Berechnung Teilergebnisse zu sehen bekommen. Sind diese unrealistisch kann die Berechnung gestoppt werden. Führt man diese Idee fort, kann man dem Nutzer sogar

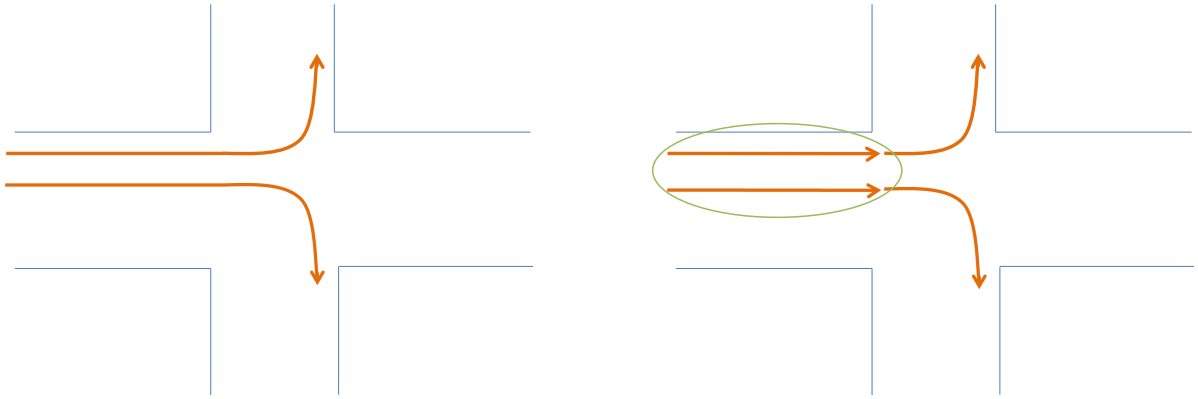
[2] Im weiteren Textverlauf wird „Visual Analytics“, der Einfachheit halber, mit „VA“ abgekürzt.

erlauben, anhand von visuellem Feedback, interaktiv auf Berechnungen zuzugreifen und den Prozess dadurch zu steuern. Nutzer können in jeden Schritt eingebunden werden und sehen jede Veränderung. Dadurch können sie besser verstehen, wie die Ergebnisse sich zusammensetzen und was sie bedeuten. Dem Nutzer kann so ein Gefühl dafür vermittelt werden, wie die automatischen Verfahren arbeiten. Das erleichtert das Interpretieren und Lernen aus den Ergebnissen. Ein noch wichtigerer Vorteil von interaktiven VA-Ansätzen ist, dass Domänenwissen der Nutzer in die Analyse, oder sogar in die Berechnung mit eingebracht werden kann. Dadurch wird ein effizienteres Arbeiten und das Erzeugen relevanter Ergebnisse ermöglicht. Die Ergebnisse können ebenfalls detaillierter analysiert werden. Die Methoden der interaktiven VA eröffnen viele neue Methoden und können die Effizienz verbessern, was das Arbeiten mit Bewegungsdaten angeht.

1.1 Intention dieses Ansatzes

Ziel dieser Arbeit ist es, die Mustererkennung in Bewegungsdatensätzen und die interaktive visuelle Analyse der Muster mit einem VA-System zu ermöglichen. Dabei sollen Nutzer durch hochinteraktive visuelle Oberflächen miteinbezogen werden. Anwender sollen die Möglichkeit erhalten, anhand von Teilergebnissen in den Prozess einzugreifen. Ihr Fachwissen kann so jederzeit in das Geschehen, oder in die Analysen mit eingebracht werden. Die bestehende Lücke zwischen Visualisierung und automatischer Verarbeitung soll überbrückt werden, um die Vorteile beider Methoden stärker zu vereinen.
Im Rahmen dieser Arbeit wurde ein VA-System [30] entwickelt, welches einen dichtebasierten Bewegungsdaten-Clustering-Algorithmus implementiert und interaktiv mit einer visuellen Komponente verknüpft. Die Intention dabei ist, dass Anwender jederzeit in den Clusteringprozess eingreifen und anhand des aktuellen Standes, durch Parameterangaben, Einfluss auf den Prozess nehmen können.
Dieses System baut auf der Arbeit von Jae-Gil Lee, Jiawei Han und Kyu-Young Whang auf, welche in ihrer Ausarbeitung „Trajectory clustering: a partition-and-group framework“ einen Algorithmus zum Clustern von Trajektorien entworfen haben. Der Algorithmus arbeitet mit einer Partitionierung, die dem eigentlichen Clustering vorausgeht [21]. Er läuft demnach in zwei Phasen ab, welche für den hier vorgestellten Ansatz übernommen wurden.
In der ersten Phase erfolgt eine Partitionierung der Trajektorien in einzelne Liniensegmente. Da Trajektorien viele Messpunkten enthalten können, wäre es nicht immer sinnvoll, ganze Trajektorien nach ihrer Ähnlichkeit zu gruppieren. Folgen zum Beispiel mehrere Bewegungen dem gleichen Weg und gehen ab einem bestimmten Punkt, zum Beispiel einer Kreuzung, in verschiedene Richtungen, so sind sich diese Trajektorien im Ganzen nicht ähnlich und würden nicht zusammen geclustert werden (siehe Abbildung 1.2 a). Durch die Aufteilung in die einzelnen Linien, würde aber der gemeinsame Weg als Ähnlichkeit erkannt werden und die entsprechenden Linien könnten in einem Cluster zusammengefasst werden (siehe Abbildung 1.2 b).

Oft beinhalten Trajektorien auch Messfehler. Wenn zwei Bewegungen gleich verlaufen, eine davon aber einen Messfehler enthält, weichen sie, aufgrund dieses Fehlers, stark voneinander ab und werden nicht zusammen gruppiert. Durch eine Partitionierung werden aber die gleich verlaufenden Teile der Bewegung, vor und nach dem Messfehler, als Teilbewegungen



(a) Kein Cluster gefunden, da die Trajektorien sich als Ganzes nicht genug ähneln.

(b) Cluster gefunden (grün), aufgrund der vorherigen Partitionierung.

Abbildung 1.2: Einfluss der Partitionierung auf das Clustering von Trajektorien.

in das gleiche Cluster gruppiert. Die Partitionierung hat dabei noch mehr wesentliche Vorteile. Durch eine starke Partitionierung kann die Anzahl an zu clusternden Elementen erheblich reduziert werden, wodurch die Zeit, die das Clustern in Anspruch nimmt, deutlich gesenkt wird. Performanz-Verbesserungen sind demnach eine mögliche Folge. Außerdem kann eine Partitionierung Stördaten filtern. Diese werden als einzelne Bewegungen betrachtet und beeinflussen somit nicht das Clustering der restlichen Bewegung. Dies geht aber mit dem Nachteil einher, dass der Clusterprozess an Genauigkeit verliert. Umso stärker partitioniert wird, desto schneller und ungenauer wird der Prozess des Clusterns. Auf dieses Problem wird in vorliegendem Ansatz eingegangen und eine mögliche Lösung präsentiert.
Nach der Partitionierung kann das Clustering, die zweite Phase, gestartet werden. Der Clusteralgorithmus arbeitet dichtebasiert, das bedeutet, in der Nähe eines initialen Elementes muss eine festgelegte Anzahl von anderen Elementen vorkommen, damit diese als Cluster erkannt werden. Die Nähe wird anhand einer Distanzfunktion festgelegt. Auch die Distanzfunktion soll der Nutzer selbst optimieren können. Das komplette Clustering soll dabei interaktiv bleiben und ständig visuelles Feedback liefern. Dazu gehört ebenfalls die Benachrichtigung des Nutzers über den Status des Clusterings. Falls dieses in eine ungewollte Richtung läuft, soll die Möglichkeit eines sofortigen Neustarts mit anderen Parametern gewährleistet werden, um die volle Kontrolle über den Prozess zu ermöglichen.
Insgesamt zielt dieser Ansatz darauf ab, Nutzer, beziehungsweise Analysten, interaktiv in einen Clusteringprozess von Bewegungsdaten einzubinden, indem für visuelles Feedback und Interaktionsmöglichkeiten gesorgt wird. Abbildung 1.3 veranschaulicht diesen Vorgang. Vergleicht man den Ablauf mit der Data-Mining Pipeline von Fayyad et al. [9] (Abbildung 1.4), fällt auf, dass Nutzer zu fast jedem Zeitpunkt eine Visualisierung und Möglichkeiten der Interaktion zur Verfügung haben, in der Data-Mining Pipeline aber höchstens im letzten Schritt, der Interpretation.

Viele VA-Ansätze beschränken die Visualisierung nur auf diesen Interpretationsschritt. Dadurch entsteht eine Lücke zwischen Visualisierung und Datenverarbeitung. Diese Lücke soll nun gefüllt werden, indem der gesamte Prozess interaktiv gestaltet und visualisiert wird. Der angestrebte Programmablauf lässt sich auch nicht ganz mit VA-Pipelines beschreiben, wie zum

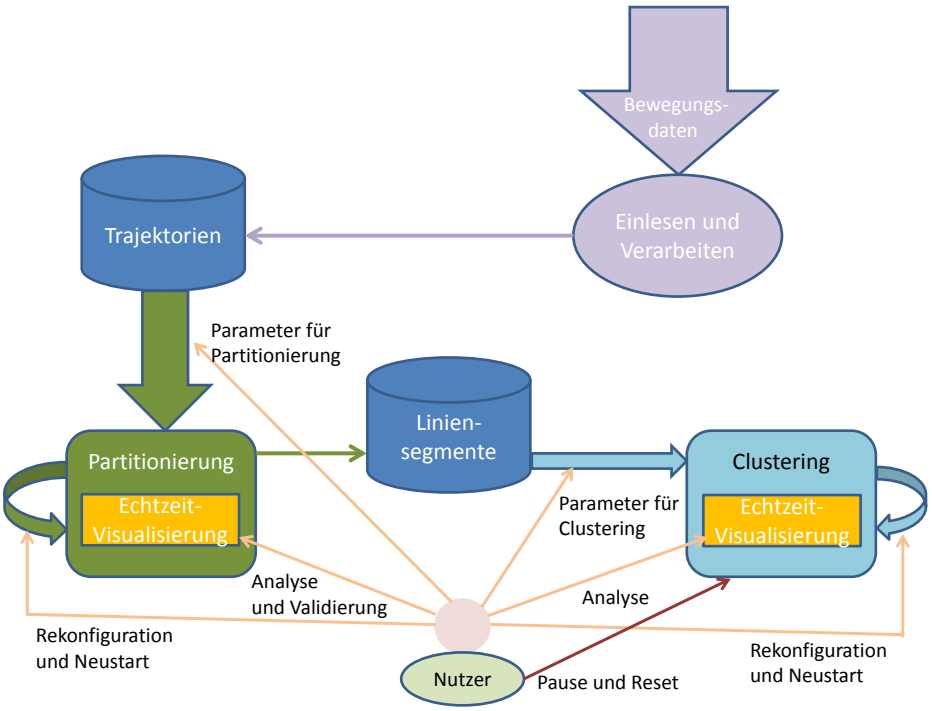


Abbildung 1.3: Schematischer Ablauf des Programmes mit den möglichen Interaktionen des Benutzers. Die Partitionierung und das Clustering können jederzeit mit veränderten Parametern neu gestartet werden.

Beispiel in Abbildung 1.5.

Benutzer haben hier nämlich keinen Einfluss auf die Mining-Algorithmen, wie die Grafik zeigt. Trotzdem kommt diese Pipeline dem hier entworfenen Ansatz näher, da Nutzer jederzeit Einfluss auf die Daten haben. Im Prinzip wird in dieser Arbeit versucht, beide Pipelines zu kombinieren. Statt außerhalb der Data-Mining Vorgänge sollen die Möglichkeiten der Interaktion und Analyse in jedem Zwischenschritt gewährleistet werden.

1.2 Struktur dieser Arbeit

In der weiteren Arbeit werden detailliert das gewählte Vorgehen, die getroffenen Entscheidungen und die gewonnen Erkenntnisse beschrieben. Sie ist wie folgt aufgebaut:
Im nächsten Kapitel werden verwandte Arbeiten beschrieben. Diese werden nach Themengebieten gruppiert und diskutiert. Dabei wird der aktuelle Stand der Forschungen gezeigt und auf Probleme eingegangen. Am Ende des Kapitels steht fest, wo genau diese Arbeit angesiedelt ist und auf welche bestehenden Arbeiten sie sich stützt. Kapitel 3 beschäftigt sich mit dem verfahrenstechnischen Hintergrund dieser Arbeit. Der implementierte Algorithmus wird dazu in seine Bestandteile, nämlich Partitionierung, Distanzmaß und Clustering, zerlegt und erläutert. Die Anpassungen, welche an dem Algorithmus vorgenommen wurden, werden in Kapitel 4 erklärt. Das Kapitel 5 beschreibt die visuellen Komponenten, wobei vor allem auf die Schnittstellen zwischen Nutzer und Algorithmus eingegangen wird. Dabei wird gezeigt, wie

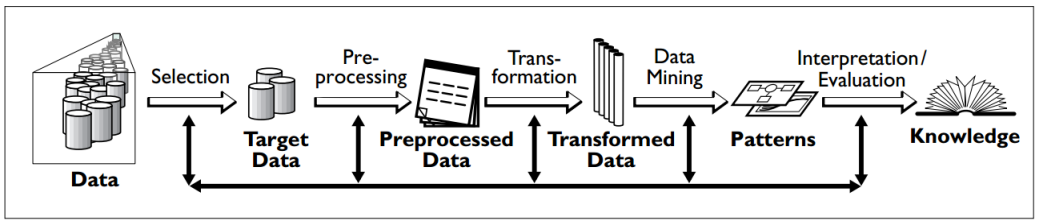


Abbildung 1.4: Data-Mining Pipeline nach Fayyad et al. [9]

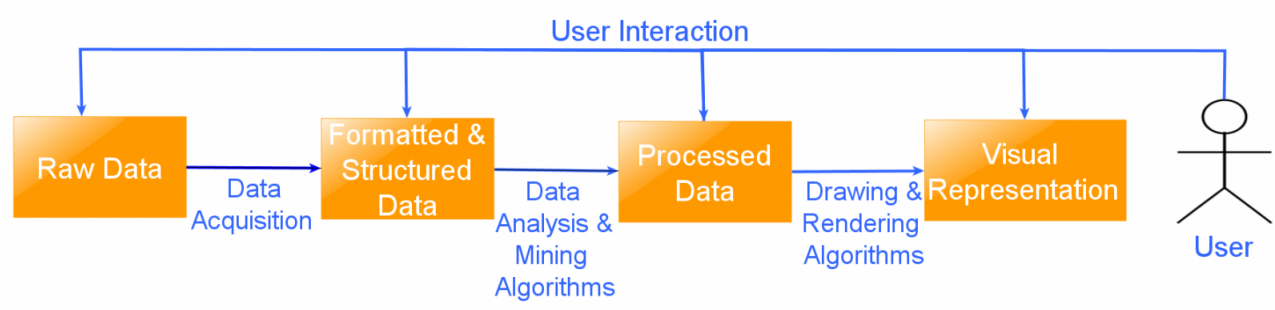


Abbildung 1.5: Eine VA-Pipeline mit den schematischen Abläufen.[3]

die interaktiv visuelle und die berechnende Komponente miteinander kombiniert wurden. In Kapitel 6 wird auf Implementierungsdetails und Systemeigenschaften eingegangen. Kapitel 7 beschreibt anschließend die Anwendungsfälle. Dazu wird der Einfluss von Parametern auf die Ergebnisse untersucht, mögliche Problemfälle aufgezeigt und erläutert, für welche Anwendungsfälle sich dieser Ansatz eignet. Des Weiteren werden die Datensätze vorgestellt, die zum Testen genutzt wurden. Das darauf folgende Kapitel diskutiert die Vor- und Nachteile dieses Ansatzes. In den letzten beiden Kapiteln wird die Arbeit zusammengefasst und ein Ausblick gegeben, welche weiteren Forschungen oder Verbesserungen noch sinnvoll wären und welche Vorteile sie bringen könnten.

[3]Bildquelle: `http://dm.gsse.pafkiet.edu.pk/`

2 Verwandte Arbeiten

Der vorgestellte Ansatz ist in mehreren wissenschaftlichen Bereichen angesiedelt. Aus diesem Grund muss nun geklärt werden, wo genau er zugeordnet werden kann. Dazu werden die zwei Komponenten des Ansatzes betrachtet: Clustering von Trajektorien und „Visualisierung und Interaktion“. Arbeiten, welche nur entfernt etwas mit den Themen gemeinsam haben, werden in einem weiteren Unterkapitel erwähnt. Am Ende soll klar sein, auf welchen Arbeiten dieser Ansatz aufbaut und welche Lücke er schließt.

2.1 Clustering von Trajektorien

Der Clustering-Algorithmus ist ein Hauptbestandteil dieser Arbeit, da er ermöglicht, Muster aus den Daten zu extrahieren. Deshalb folgt nun eine Übersicht über Cluster-Verfahren aus anderen Arbeiten, um den hier implementierten Algorithmus richtig einordnen zu können. 1996 stellten Martin Ester, Hans-Peter Kriegel, Jörg Sander und Xiaowei Xu ihren Ansatz zum dichtebasierten Clustering von räumlichen Daten vor [8]. Dabei handelt es sich um DBSCAN, welcher sich mittlerweile als ein Standardalgorithmus zum Clustern räumlicher Daten etabliert hat. Wichtige Vorteile des Algorithmus sind, dass er auf beliebige Distanzfunktionen und Elemente angepasst werden kann und mit einer geeigneten Indexstruktur eine logarithmische Laufzeitkomplexität aufweist. Diese Vorteile machen den Algorithmus sehr flexibel, weshalb er als Grundlage für viele weitere Ansätze dient. Eine Schwäche von DBSCAN ist aber, dass er, aufgrund eines festen Distanz-Parameters, nur Cluster gleicher Dichte erkennt. Um dieses Problem zu lösen entwickelten Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel und Jörg Sander den Algorithmus OPTICS [5]. Wenn beispielsweise zwei Cluster unterschiedlicher Dichte in einem Datensatz existieren, würde DBSCAN nur eines finden. OPTICS hingegen ordnet die Elemente linear nach ihrer räumlichen Distanz an und wählt den Distanz-Parameter variabel anhand von Kernpunkten. So können auch Cluster unterschiedlicher Dichte gefunden werden mit der gleichen Zeitkomplexität wie bei DBSCAN. Interessanter für diese Arbeit sind aber Clustering-Algorithmen, welche sich direkt mit dem Clustern von Bewegungsdaten befassen. Dazu werden oft bestehende Algorithmen, wie DBSCAN, umfunktioniert und auf das Arbeiten mit Bewegungsdaten angepasst. Da solche Daten komplexer sind als einzelne Punkte, müssen viele Bestandteile der Algorithmen überarbeitet werden. Palma et al. zum Beispiel [23] nutzen eine für Bewegungsdaten angepasste Version von DBSCAN um prägnante Gebiete bei Trajektorien zu ermitteln. Auch TRACLUS (Trajectory Clustering) von Jae-Gil Lee, Jiawei Han und Kyu-Young Whang basiert, ähnlich wie OPTICS oder der Algorithmus von Palma, auf DBSCAN [21]. Ein Unterschied bei TRACLUS ist aber, dass nicht ganze Trajektorien, sondern nur Teile der Bewegungen gruppiert werden, da er die Trajektorien partitioniert. Auf diese Weise können spezifischere Ergebnisse erreicht werden. Der TRACLUS-Algorithmus dient als Grundlage für diese Arbeit. Er unterscheidet sich, abgesehen von der Partitionierung, in einem weiteren Detail von seiner Vorlage. In DBSCAN können alle dichteverbundenen Gruppen von Elementen als Cluster erkannt werden. Würde man also DBSCAN mit einer Partitionierung auf Bewegungsdaten anwenden, könnten Cluster entstehen, welche nur Linien aus einer Trajektorie beinhalten. In TRACLUS dagegen werden nur solche Gruppen

als Cluster erkannt, welche Elemente aus einer definierten Anzahl unterschiedlicher Trajektorien enthalten. Dadurch wird sichergestellt, dass Cluster nicht nur aus einzelnen Trajektorien extrahiert werden, sondern immer aus mehreren. Die entstehenden Cluster sind deshalb von allgemeiner Bedeutung. Das Auffinden ähnlicher Teilbewegungen ist eine große Stärke von TRACLUS. Das Problem dabei ist, dass Nutzer keine Kontrolle über die Partitionierung haben, da TRACLUS den Grad der Partitionierung immer gleich wählt. Auch hierarchische Clusteringalgorithmen, wie zum Beispiel der Algorithmus von Zhouyu Fu, Weiming Hu und Tieniu Tan aus ihrer Arbeit „Similarity based vehicle trajectory clustering and anomaly detection" können mit einer vorausgehenden Partitionierung verwendet werden [11]. Hierarchische Verfahren gruppieren Elemente in Baumstrukturen. Ein Vorteil dabei ist, dass man die Granularität der Ergebnisse beeinflussen kann, indem man die Ebene der Baumstruktur wechselt. Die hohe Laufzeitkomplexität, beim Berechnen des Baumes, macht solche Verfahren aber eher ungeeignet für interaktive Ansätze. TRACLUS arbeitet dagegen dichtebasiert und benötigt keine Baumstruktur. Verwendet wird dabei ein kombiniertes Distanzmaß, welches aus drei Komponenten aufgebaut ist: Eine parallele Distanz (parallel distance), eine Winkeldistanz (angle distance) und eine senkrechte Distanz (perpendicular distance)[1]. Die Arbeit [16] verwendet ein ähnliches kombiniertes Distanzmaß. Nutzer können die einzelnen Parameter für die Gewichtungen der Komponenten dabei selbst einstellen. Manche Algorithmen, wie zum Beispiel DENTRAC (Density based Trajectory Clustering), verzichten auf feste Parameter [7]. Die Erfinder von DENTRAC gehen so weit, eigene Dichtefunktionen durch Abschätzung zu finden, auf welchen sie nach dem Hill-climbing Prinzip arbeiten, um lokale Maxima zu finden. Durch das Fehlen von expliziten Parametern würden aber mögliche Schnittstellen zu dem Benutzer verloren gehen, was nicht in der Intention dieser Arbeit liegt.
Mittlerweile sind viele Abwandlungen entstanden, wie zum Beispiel zeitbasiertes Clustering auf der Grundlage von OPTICS [22], oder Clustering über Vektorfelder [10], welche ebenfalls sehr gute Ergebnisse erzielen. In den letzten Jahren wurden auch die Laufzeiten solcher Algorithmen optimiert. Randomisierte Versionen von DBSCAN wie zum Beispiel DBRS liefern gute Ergebnisse bei geringerer Laufzeit [2] und auch parallele Ansätze, wie zum Beispiel von Pratik Khatavkar [28], haben sich etabliert. Das Clustering von Trajektorien ist bereits fortgeschritten und bietet einige gute Ansätze. Pavel Berkhin fasst in seiner Arbeit einige Clustering-Algorithmen, nach ihren Stärken und Schwächen, zusammen [6]. Für die vorliegende Arbeit wurde, unter Einbeziehung der Vergleiche von Berkhin, TRACLUS als Algorithmus ausgewählt, da er dank der Partitionierung ähnliche Teilbewegungen gruppieren kann, ein kombiniertes Distanzmaß verwendet und vielseitig einsetzbar ist.

2.2 Visualisierung und Interaktion

Der Trend bei Trajektorien-Analyse-Algorithmen geht in die Richtung visueller Analysen, da oft die kognitiven Fähigkeiten eines Menschen erforderlich sind. Anhand vieler aktueller Ansätze, zum Beispiel von Gennady Andrienko und Natalia Andrienko [1][26], ist dieser Trend zu beobachten. Deshalb muss nun der Stand der Visualisierung und Interaktion solcher Ansätze betrachtet werden.
Visualisierung und Interaktion ermöglichen die aktive Einbindung von Nutzern. Diese wer-

[1] Die genaue Beschreibung der einzelnen Distanzen erfolgt in Kapitel 3.

ten visualisierte Ergebnisse aus, oder können Berechnungen, wie etwa das Clustering, in die gewünschte Richtung lenken. Die Vorgehensweise ist also anders, als bei gewöhnlichen Data-Mining Techniken. Nutzer haben dabei eine viel bessere Kontrolle über die Vorgänge. In diesem Ansatz wird ein Clustering-Verfahren verwendet, um die Daten zu aggregieren und so einfacher Muster zu erkennen. Die Einsicht in diesen Prozess wird durch eine Visualisierung ebenfalls verbessert und die Verständlichkeit der Prozessschritte wird gefördert. Die Arbeit „Interactive Visual Clustering of Large Collections of Trajectories“ [3] erläutert die Notwendigkeit von Visualisierungen bei Ansammlungen von Bewegungsdaten. In großen Datensätzen werden mit Clustering-Algorithmen oft viele Cluster gefunden. Nicht alle davon sind aussagekräftig und die Erkenntnisse, die sie liefern, sind oft nicht auf den ersten Blick ersichtlich. Die Visualisierung dient dabei als Bindeglied zwischen Nutzer und Algorithmus. Sie ermöglicht die Interpretation der Ergebnisse. Im eben genannten Ansatz werden Nutzern interaktive Werkzeuge zur Verfügung gestellt, die sie beim Auswerten der Ergebnisse unterstützen. Die weiter oben genannten Ansätze [1][26] verwenden ein ähnliches Prinzip, in welchem Nutzer aus mehreren Ansichten, sowie Anzeigeoptionen, wählen können. Auch diese sollen ihnen dabei helfen, die großen Daten- und Ergebnismengen zu überblicken. Eine ältere Arbeit von 2007 beschäftigt sich ebenfalls mit der Visualisierung großer Datenmengen und der Unterstützung der Nutzer bei der Analyse [4]. Der Nutzer kann Cluster ausblenden, nach Zeit filtern, Diagramme einblenden, die zum Beispiel die Aufenthaltszeit an einem Ort anzeigen, oder die Richtung von Bewegungen visualisieren lassen. Die Möglichkeiten, dem Nutzer die Analyse zu erleichtern, sind demnach vielseitig. Das Buch „Illuminating the Path - The Research and Development Agenda for Visual Analytics“ von James J. Thomas und Kristin A. Cook beschreibt die Relevanz von menschlichem Entscheidungsvermögen welches, laut ihren Feststellungen, unerlässlich für Analysevorgänge ist [30]. Aus diesem Grund beschreiben sie die Prinzipien von visueller Analytik und legen Richtlinien fest, um visuelle Analyseprogramme übersichtlich und einfach zu gestalten. Die Arbeit „Interactive Schematic Summaries for Faceted Exploration of Surveillance Video“ [16] von Höferlin et al., zur Analyse von Bewegungen in Videodaten, hat eine sehr detaillierte Visualisierung, die aber durch ihren zeichentrickartigen Stil etwas herausragt. Nutzer können hier zwischen verschiedenen Darstellungsformen wechseln und sich Repräsentanten von Clustern anzeigen lassen, um eine bessere Übersicht zu bekommen. Auch die ursprünglichen Videodaten können beispielsweise eingeblendet werden. Legenden ermöglichen Nutzern das einfache Zurechtfinden und das Verstehen der Daten und Ergebnisse. Bestimmte Objektgruppen können, falls die Objekte kategorisiert wurden, ein- oder ausgeblendet werden, um zielorientiertes Arbeiten zu ermöglichen. Außerdem verfügt dieser Ansatz über ein System von Facetten, wie etwa Zeit oder Geschwindigkeit von Bewegungen, die Nutzer interaktiv einstellen können. Diese Form der Visualisierung ist sehr benutzerfreundlich und ermöglicht eine gute Exploration der Daten.
Allgemein gibt es schon erprobte Methoden der Visualisierung von Trajektorien, die Nutzern eine gute Übersicht und somit eine Grundlage zum zielorientierten Arbeiten schaffen. Diese werden auch passend mit vorhandenen Clusteralgorithmen verbunden, wie die hier erwähnten Arbeiten zeigen.
Das eigentliche Problem ist, dass sich die bisherigen Interaktionen und Visualisierungen fast ausschließlich auf die Exploration der Ergebnisse beziehen. Erst nach dem Clustering werden die Daten visualisiert und können analysiert werden. Die Werkzeuge und visuellen Darstellungen der erwähnten Arbeiten sind alle hilfreich und erleichtern Nutzern die Interpretation von Ergebnissen, lassen sich aber erst ganz am Schluss der Programmausführung nutzen. Der

Cluster-Vorgang ist ein langer Prozess und während dieser in Gang ist können Nutzer nicht mit dem Programm arbeiten, sondern müssen warten. Sind die Ergebnisse fehlerhaft, muss der gesamte Vorgang wiederholt werden. Um die Zeit während dem Clustervorgang sinnvoll zu nutzen und dadurch die Lücke bisheriger Verfahren zu schließen, bedient sich dieser Ansatz einer fortlaufenden Visualisierung mit zusätzlichen Möglichkeiten der Interaktion.

2.3 Zusätzliche relevante Arbeiten

Ein bekannteres Buch, welches sich mit Konzepten und Techniken von Data-Mining beschäftigt [15], gibt auch viele Anregungen zum Thema Visualisierung solcher Daten. Einige kleinere Informationen dieser Quelle sind in diese Arbeit eingeflossen. Dazu gehören zum Beispiel eine Zoomfunktion, oder die klare farbliche Abgrenzung verschiedener Cluster. Manche Arbeiten, wie zum Beispiel [31], beschäftigen sich ausführlich mit der Darstellung von räumlichen Daten in geografischen Informationssystemen. Solche Arbeiten sind meist gut belegt und helfen dabei, passende Darstellungs- und Umrechnungsmethoden zu finden, wenn sich die räumlichen Daten über große Distanzen erstrecken. Die dort vorgeschlagene Methode, um Bewegungen zu unterschiedlichen Zeitpunkten vergleichend darzustellen, könnte eine gute Erweiterung für den hier vorgestellten Ansatz sein. Ein weiteres relevantes Thema ist die Indexierung von Daten. Durch eine geeignete Indexierung können die Laufzeiten vieler Clusteralgorithmen stark reduziert werden, wie zum Beispiel bei DBSCAN. Mit einer geeigneten Indexstruktur lässt sich die Laufzeit von DBSCAN von einer quadratischen auf eine logarithmische reduzieren. Das spart viel Zeit, welche dafür in Analysen gesteckt werden kann. Einige Indexstrukturen wie beispielsweise die R-Bäume [14] oder VP-Bäume (Vantage-point trees) [33][2] haben sich auch im Gebiet des Arbeitens mit räumlichen Daten etabliert und werden häufig verwendet. Im Wesentlichen werden Strukturen erzeugt, welche dafür sorgen, dass man bei der Suche nach Nachbarn eines Elementes nicht mehr alle anderen Elemente absuchen muss, sondern nur noch einen viel kleineren Teil der Daten. Der Rechenaufwand wird dadurch stark reduziert. Für diese Arbeit ist eine Indexierung aber nur von optionaler Relevanz, da sie lediglich der Beschleunigung dienen würde. Trotzdem wurden erste Versuche mit einer Indexierung durchgeführt, die bereits vielversprechend ausgefallen sind.

[2] Die Indexstruktur des VP-Baumes wurde etwas früher schon von Jeffrey Uhlmann entdeckt. Diese Arbeit von Peter Yianilos wurde aber deshalb als Referenz gewählt, da Yianilos erstmals den Begriff „VP-Tree“ einführte, unter welchem diese Indexstruktur bekannter ist.

3 Berechnendes Verfahren

Dieses Kapitel erläutert die angewandten datenverarbeitenden Verfahren für die vorliegende Arbeit. Dazu wird das jeweilige Verfahren zuerst vorgestellt und die Funktionsweise erklärt. Dabei wird diskutiert, warum das Verfahren genutzt wird und welchen Vorteil es in Bezug auf diese Arbeit liefert. Die Funktionen und das Zusammenspiel der einzelnen Komponenten sollen so für die Leser ersichtlich werden. Das Kapitel ist in drei Unterkapitel aufgeteilt, die jeweils im Detail die drei wichtigsten Komponenten des Algorithmus behandeln. Zuerst wird das Distanzmaß beschrieben. Dieses wird sowohl bei der Partitionierung, als auch beim Clustering verwendet. Das zweite Kapitel beschreibt die Partitionierung. Anschließend wird das eigentliche Clustering erklärt. Die letzten beiden Kapitel sind aufgeteilt in das Prinzip des Verfahrens und die Details der Realisierung, um sie verständlicher zu gestalten. Trajektorien und Liniensegmente folgen stets den Definitionen 2 und 3 und stützen sich auf die Definition 1 der Datenpunkte. Diese Definitionen wurden aus [24] abgeleitet.

Definition 1 (Punkt) Ein Punkt beschreibt eine Position einer Bewegung und ist als Tupel definiert:
$p_i = (time, pt, \delta)$
Dabei symbolisiert δ verschiedene zusätzliche Angaben, wie zum Beispiel Geschwindigkeit oder Höhe der Bewegung an der aktuellen Position.
Der Zeitpunkt eines Punktes wird mit *time* beschrieben und *pt* ist ein Tupel, welches die Koordinaten eines Punktes in Längen- und Breitengraden enthält.

Definition 2 (Trajektorie) Eine Trajektorie wird aus unverarbeiteten Bewegungsdaten erzeugt und ist als Tupel definiert:
$TR_i = (TraID_i, ObjID,$ LISTE von Punkten$)$
Eine Trajektorie der Länge n enthält die Punkte p_1, p_2, ..., p_n.
$TraID$ bezieht sich auf die Nummer der Trajektorie und $ObjID$ auf die Nummer des bewegenden Objektes.

Definition 3 (Liniensegment) Eine Liniensegment wird durch eine Partitionierung aus einer Trajektorie erzeugt und ist als Tupel definiert:
$L_i = (LnID, TraID_i, ObjID,$ Punkt $s_i,$ Punkt $e_i)$
Im Gegensatz zu einer Trajektorie besitzt ein Liniensegment nur einen Startpunkt s_i und einen Endpunkt e_i.
Für den Index dieser Punkte gilt $s_i < e_i$. Ausgehend von den Punkten p_1, p_2, ..., p_n der originalen Trajektorie gilt zusätzlich $p_1 \leq s_i < p_n$ und $p_1 < e_i \leq p_n$.
$TraID$ bezieht sich auf die Nummer der partitionierten Trajektorie und $ObjID$ auf die Nummer des bewegenden Objektes.
$LnID$ ist die Nummer des Liniensegments.

$TraID_i$ und $LnID$ ermöglichen eindeutige Identifizierungen der jeweiligen Trajektorien, beziehungsweise Liniensegmente. Genauso identifiziert $ObjID$ das bewegende Objekte, zum Beispiel ein Mensch oder ein Fahrzeug, eindeutig. Dadurch können mehrere Trajektorien oder Liniensegmente einem Objekt zugeordnet werden.

3.1 Distanzfunktion

Das Distanzmaß ist eine äußert wichtige Komponente bei jedem Clustering. Es definiert nämlich, durch was die Distanz zwischen zwei Objekten beschrieben wird. Oft stellt man sich eine Distanz einfach als eine räumliche Distanz vor, doch ein Distanzmaß kann auch Richtung, Form, Farben, Zeit und viele weitere Faktoren bei der Berechnung einer Distanz miteinbeziehen. Ein Distanzmaß ist austauschbar und manchmal auf spezifische Datensätze zugeschnitten. Für die vorliegende Arbeit wird das Distanzmaß von [21] verwendet. Laut den Autoren liefert es gute Ergebnisse für Liniensegmente, weshalb es auch in einer späteren Arbeit [20] wiederverwendet wurde. Um den implementierten Algorithmus im Ganzen zu verstehen, wird nun die Definition und die Funktion des Distanzmaßes erklärt. Anschließend folgt ein kurzer Abschnitt, der erklärt, warum dieses Distanzmaß ausgewählt wird.

3.1.1 Definition und Funktion

Das Distanzmaß besteht aus drei Komponenten, welche hier nacheinander erläutert werden. Dafür werden die Definitionen von [21] und [20] genutzt. Die Komponenten können alle getrennt berechnet werden, was auch nötig ist, da zwei der Komponenten bei der Partitionierung (Formel 3.6) separat verwendet werden.
Die drei Komponenten sind die „senkrechte Distanz“ $d_\perp$, die „parallele Distanz“ $d_\parallel$ und die „Winkeldistanz“ d_θ, welche in dieser Reihenfolge nun beschrieben werden. Die erste Komponente ist demnach die „senkrechte Distanz“ $d_\perp$, welche in Formel 3.1 definiert ist.

$$d_\perp(L_i, L_j) = \frac{l_{\perp 1}^2 + l_{\perp 2}^2}{l_{\perp 1} + l_{\perp 2}} \tag{3.1}$$

In der Definition kommt mehrfach $l_{\perp x}$ vor. Dabei beschreibt $l_{\perp x}$ die euklidische Distanz, oder eine gleichwertige Distanz für geografische Daten[1], zwischen zwei Punkten. Anzunehmen ist, dass durch die Projektionen des Start- und des Endpunktes s_j und e_j der Linie L_j, auf die Linie L_i, die Punkte p_s und p_e entstehen. Dabei wird stets das längere Liniensegment als L_i und das kürzere als L_j gewählt, was wichtig für die Symmetrie des Distanzmaßes ist. Davon ausgehend beschreibt $l_{\perp 1}$ die euklidische Distanz zwischen s_j und p_s und $l_{\perp 2}$ die euklidische Distanz zwischen e_j und p_e, wie in Abbildung 3.1 zu sehen ist [20].
Im Prinzip beschreibt die senkrechte Distanz also die senkrechte Abweichung zweier Linien, wobei auch mit einfließt, wie parallel diese Linien zueinander sind. Wenn nämlich e_j viel weiter von L_i entfernt ist, als s_j, so hat L_j einen anderen Winkel als L_i und $l_{\perp 2}$ wäre demnach

[1] Wenn in diesem Abschnitt über die euklidische Distanz gesprochen wird, so wird impliziert, dass dies nicht die beste Lösung für bestimmte Datensätze ist. Wenn die Datensätze sich über große Gebiete erstrecken und nur Längen- und Breitengrade zur Verfügung stehen, kann es zu Abweichungen kommen, welche die Ergebnisse verfälschen.

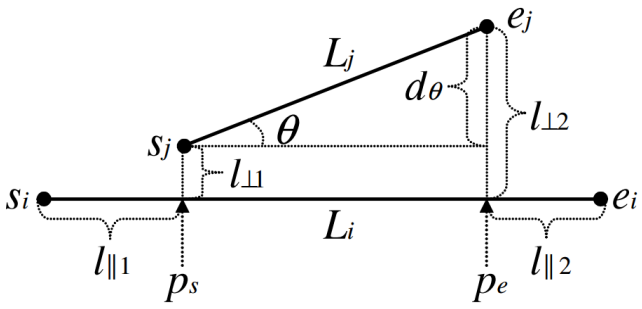


Abbildung 3.1: Die drei Komponenten der Distanzfunktion aus [20].

größer.
Diese Komponente kann also als räumliche Abweichung zweier Linien genutzt werden und sorgt somit dafür, dass benachbarte Linien nicht zu weit auseinander liegen dürfen, oder gar orthogonal zueinander sind. Trotzdem können die Linien noch waagrecht voneinander abdriften, weshalb die parallele Distanz benötigt wird.

Die „parallele Distanz“ $d_\parallel$ in Formel 3.2 ist demnach die zweite Komponente.

$$d_\parallel(L_i, L_j) = MIN(l_{\parallel 1}, l_{\parallel 2}) \tag{3.2}$$

Über p_s und p_e werden die gleichen Annahmen wie zuvor getroffen. In der Formel steht $l_{\parallel 1}$ für das Minimum der euklidischen Distanzen von p_s zu s_i und e_i und $l_{\parallel 2}$ für das Minimum der euklidischen Distanzen von p_e zu s_i und e_i [20]. Abbildung 3.1 veranschaulicht $l_\parallel$ sehr gut. Von diesen beiden Distanzen $l_\parallel$ wird wiederum das Minimum genommen, da L_i zum Beispiel länger sein kann, als L_j, wobei trotzdem keine waagrechte Abdriftung vorliegt. Kombiniert man die senkrechte und die parallele Distanz, kann man also räumlich dichte Cluster finden, welche weder horizontal, noch vertikal, zu stark voneinander abweichen. Dabei wird aber noch nicht betrachtet, in welche Richtung die Linien verlaufen. Zwei Linien, die sich in entgegengesetzte Richtungen bewegen, könnten also in das gleiche Cluster geraten.

Die dritte Komponente für das Distanzmaß ist demnach die „Winkeldistanz“ d_θ.

$$d_\theta(L_i, L_j) = \begin{cases} \|L_j\| \times sin(\theta) & \text{, falls } 0° \leq \theta < 90° \\ \|L_j\| & \text{, falls } 90° \leq \theta \leq 180° \end{cases} \tag{3.3}$$

Es handelt sich hier um eine Überprüfung des kleineren Winkels θ, zu sehen in Abbildung 3.1, zwischen L_i und L_j. $\|L_j\|$ gibt die Länge von L_j an. Auf diese Weise lassen sich Liniensegmente anhand ihrer Richtung vergleichen. Umso kleiner der Winkel zwischen zwei Liniensegmenten ist, desto kleiner ist auch die Winkeldistanz. Damit lassen sich Richtungsverläufe entdecken. Durch eine Kombination mit den anderen beiden Komponenten lassen sich Strecken clustern, da sowohl die Richtung, als auch die räumliche Abweichung miteinbezogen werden.
Die Gesamtdistanz $dist(L_i, L_j)$, welche zur Berechnung der Nachbarn im Clustering verwendet wird, ist eine Kombination aus den drei Komponenten:

$$dist(L_i, L_j) = w_\perp * d_\perp(L_i, L_j) + w_\parallel * d_\parallel(L_i, L_j) + w_\theta * d_\theta(L_i, L_j) \tag{3.4}$$

Das Zusammenspiel dieser Komponenten wird anhand von Gewichtungen w reguliert. Dadurch kann das Distanzmaß für verschiedene Datensätze und Analysevorgänge angepasst werden.

3.1.2 Relevanz des Distanzmaßes

Es gibt eine handvoll Gründe, warum sich das vorgestellte Distanzmaß für die vorliegende Arbeit eignet. Als Erstes ist anzumerken, dass die gewählte Partitionierung bereits zwei der Komponenten des Distanzmaßes erfordert, was aber nur ein beiläufiger Grund ist. Ein wichtiger Grund ist, dass die gewählte Distanzfunktion symmetrisch ist, also $dist(L_i, L_j) = dist(L_j, L_i)$ immer zutrifft. Das längere Liniensegment wird als L_i und das kürzere als L_j festgelegt, wodurch die Symmetrie erreicht wird. Wäre das Distanzmaß asymmetrisch, so wären die Ergebnisse abhängig von der Reihenfolge der Verarbeitung der Linien [21]. Der wichtigste Grund ist aber, dass das Distanzmaß, durch die drei Komponenten, sehr variabel gestaltet ist. Es kann für viele Analysevorgänge und Datensätze individuell, über die Gewichtungen, angepasst werden. Dadurch sind spezifischere Ergebnisse möglich. Des Weiteren hat dieses Distanzmaß bereits in zwei Arbeiten [21][20] gute Ergebnisse geliefert.

3.2 Partitionierung

Die verwendete Partitionierung wurde als Teil des „Partition-and-Group Framework“ von Jae-Gil Lee, Jiawei Han und Kyu-Young Whang [21] speziell zum Clustern von Trajektorien entwickelt. Außerdem ist sie sehr ähnlich zu derjenigen, welche Jae-Gil Lee, Jiawei Han und Xiaolei Li ein Jahr später in ihrem „Partition-and-Detect Framework“ vorstellten [20]. Dieses hat die Intention Ausreißer in Trajektorien zu ermitteln. Die Partitionierung ist jeweils die erste Phase der Frameworks. Sie reduziert ganze Trajektorien auf einzelne Liniensegmente und erfüllt dadurch mehrere Zwecke: Zum Einen reduziert sie die Anzahl an zu verarbeitenden Elementen, was die Laufzeit des Clusterings reduziert. Weiterhin hilft sie dabei, die prägnanteren Teile der Trajektorien zu finden und von spezielleren Passagen abzugrenzen. Dadurch kann später Rauschen besser detektiert werden und es können Teilwege gefunden werden, die ohne Partitionierung nicht entdeckt werden. Im Folgenden wird zuerst das Prinzip der Partitionierung erklärt und daraufhin auf den verfahrenstechnischen Hintergrund eingegangen.

3.2.1 Prinzip der Partitionierung

Trajektorien sind oft komplexe Gebilde, die aus unterschiedlich vielen Datenpunkten bestehen können. Viele solcher Trajektorien haben gemeinsame Streckenabschnitte und verlaufen danach in andere Richtungen. Wenn man beispielsweise annimmt, mehrere Menschen fahren morgens über eine Autobahn in eine Großstadt zu ihrem Arbeitsplatz, so ist der Weg auf der Autobahn für alle Fahrten gleich, danach verlaufen die Fahrten aber alle in unterschiedliche Richtungen. Dadurch werden die Strecken bei einem konventionellen Clustering nicht zusammengefasst. Die Partitionierung kann aber diese Unter-Trajektorie, also die Autobahnfahrt in diesem Fall, isolieren, wodurch sie beim späteren Clustering gefunden wird. Dieses Prinzip lässt sich an Abbildung 3.2 gut nachvollziehen. Die Trajektorien in der Abbildung

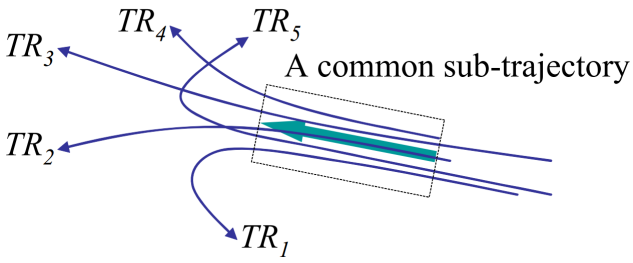


Abbildung 3.2: Fünf Trajektorien bilden eine gemeinsame Unter-Trajektorie, zu sehen an dem gemeinsamen Verlauf im Rechteck [21].

werden also an der linken Kante des Rechtecks in mindestens zwei Liniensegmente aufgeteilt und die gemeinsam verlaufenden Liniensegmente rechts können als Cluster erkannt werden. Besonders bei langen Trajektorien mit vielen Datenpunkten liefert ein Clustering mit Partitionierung genauere Ergebnisse als herkömmliche Clusterings. Ein weiterer Vorteil ist natürlich, dass Rauschen detektiert werden kann und somit sehr spezielle Teile einer Trajektorie, welche durch Messfehler entstanden sein können, gefunden und ignoriert werden können. Somit beeinflussen sie die Clusterergebnisse kaum noch [20]. Dies fließt passiv in diese Arbeit mit ein, da die Wahrscheinlichkeit, dass ungewöhnliche Trajektoriensegmente mitgeclustert werden, reduziert wird.
Um eine Partitionierung vornehmen zu können, müssen die charakteristischen Punkte identifiziert werden, ab welchen sich der Verlauf einer Trajektorie stark ändert. Zur Ermittlung dieser Punkte wird das Prinzip der minimalen Beschreibungslänge (minimum description lenght[2]) [13] genutzt. Die Punkte werden dann dadurch ermittelt, dass zwei Punkte der Trajektorie durch ein fiktives Liniensegment verbunden und die MDL-Kosten errechnet werden. Die MDL-Kosten geben dabei die Abweichung des fiktiven Liniensegmentes von der Trajektorie an. Sie werden bei der Realisierung im Detail erklärt. Ist diese Abweichung nicht zu groß, wird der Startpunkt des Liniensegments mit dem nächsten Punkt der Trajektorie verbunden. Es entsteht ein neues fiktives Segment, welches länger als das vorherige ist. Dieser Vorgang wird wiederholt, bis die Abweichung von der Trajektorie zu groß ist. Dann sind der vorletzte Punkt, sowie der Startpunkt der fiktiven Linie, die gesuchten charakteristischen Punkte. Um den nächsten Punkt zu finden wird der Prozess, ausgehend vom Endpunkt des letzten Segmentes, wiederholt, bis der Endpunkt der Trajektorie erreicht wurde. Der Start- und der Endpunkt jeder Trajektorie sind immer charakteristische Punkte. Betrachtet man Abbildung 3.3 sieht man, dass die ursprüngliche Trajektorie, welche aus acht Punkten besteht, durch drei Liniensegmente mit insgesamt vier charakteristischen Punkten ersetzt werden kann.

3.2.2 Realisierung

Die Partitionierung arbeitet auf einzelnen Trajektorien TR_i mit beliebig vielen Punkten p_1, p_2, ... , p_{len_i}. Als Eingabe erwartet die Partitionierung eine Trajektorie nach der Definition 2. Als Ausgabe liefert der Algorithmus eine Menge CP_i von charakteristischen Punkten p_{c1}, .. , p_{cn}. „n“ ist die Größe der Menge CP_i. Es handelt sich um einen schätzenden Algorithmus, also ist

[2]Im weiteren Textverlauf wird dieser Begriff mit „MDL“ abgekürzt.

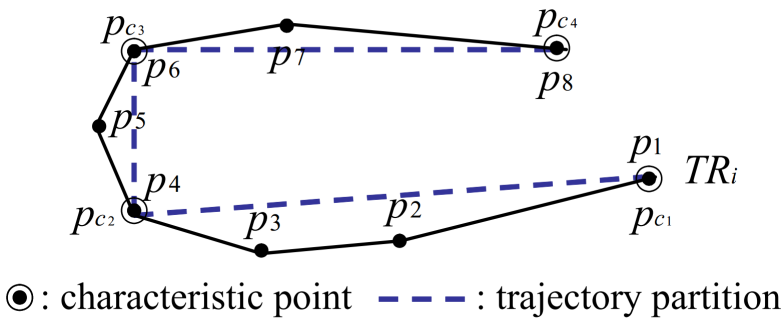


Abbildung 3.3: Eine mögliche Partitionierung mit den zugehörigen charakteristischen Punkten [21].

nicht gewährleistet, dass er die optimale Partitionierung jeder Trajektorie findet.
Um die Realisierung besser verständlich zu machen, wird nun zuerst der abschätzende Faktor der minimalen Beschreibungslänge MDL erläutert.
Das Prinzip von MDL basiert darauf, dass für eine bestimmte Abschätzung die MDL-Kosten berechnet werden, welche sich aus der Summe der Komponenten $L(H)$ und $L(D|H)$ zusammensetzen. Von ihren Autoren werden die Komponenten wie folgt beschrieben [13]: „$L(H)$ ist die Länge der Beschreibung der Hypothese in Bit; und $L(D|H)$ ist die Länge der Beschreibung der Daten, ebenfalls in Bit, wenn sie mit Hilfe der Hypothese kodiert wurden.“[3] H beschreibt dabei die Hypothese und D die Daten. Beide Bestandteile sind also von der Hypothese abhängig, deshalb gilt es, die Hypothese zu finden, welche $L(H)$ und $L(D|H)$ minimiert, wodurch die MDL-Kosten ebenfalls minimiert werden. In dem hier umgesetzten Ansatz entspricht H einer Menge von groben Partitionen, folglich misst $L(H)$ die Länge der Partitionierung einer Trajektorie[4]. Die Trajektorie selbst entspricht D. $L(D|H)$ misst die Abweichung einer Partitionierung von der Trajektorie [20]. Um $L(H)$ und $L(D|H)$ auszurechnen, werden in dieser Arbeit die Formeln von Lee et al. verwendet [21][20]:

$$L(H) = \sum_{j=1}^{par_i - 1} log_2(len(p_{c_j} p_{c_{j+1}})) \tag{3.5}$$

$$L(D|H) = \sum_{j=1}^{par_i - 1} \sum_{k=c_j}^{c_{j+1}-1} \{log_2(d_\perp(p_{c_j} p_{c_{j+1}}, p_k p_{k+1})) + log_2(d_\theta(p_{c_j} p_{c_{j+1}}, p_k p_{k+1}))\} \tag{3.6}$$

par_i ist der letzte (charakteristische) Punkt der Trajektorie, entsprechend gibt $par_i - 1$ den letzten charakteristischen Startpunkt eines Liniensegmentes an. $d_\perp$ ist dabei die senkrechte Distanz zweier Liniensegmente und d_θ ist die Winkeldistanz. In Abbildung 3.4 ist an einem allgemeinen Beispiel zu sehen, wie $L(H)$ und $L(D|H)$ berechnet werden.

[3] Original ist in Englisch verfasst, die Übersetzung wurde selbst erstellt.
[4] „len“ bedeutet in der Formel schlicht „Länge“ und kann abhängig von den Maßeinheiten der Koordinaten berechnet werden. Oft reicht die euklidische Distanz, bei Längen- und Breitengraden kann auf andere Methoden ausgewichen werden.

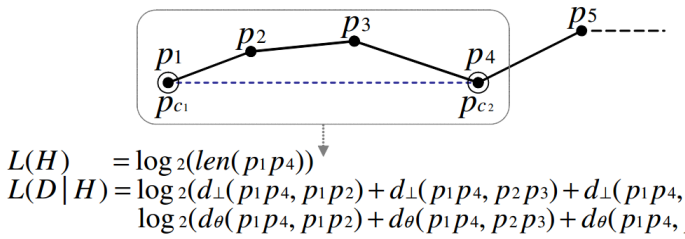


Abbildung 3.4: Berechnung der MDL-Kosten [20]

Die Gesamtkosten sind $MDL = L(H) + L(D|H)$. Im Verlauf der Partitionierung berechnet man mehrmals zwei weitere Kosten mit dieser Funktion. Zum Einen die Kosten, wenn man an entsprechender Stelle partitionieren würde „$MDL_{par}(p_i, p_j)$“ und zum Anderen die Kosten, wenn man an selbiger Stelle nicht partitionieren würde „$MDL_{nopar}(p_i, p_j)$“. p_i und p_j sind dabei zwei Punkte, zwischen welchen man ein Liniensegment erzeugen möchte. Anzumerken ist dabei, dass $L(D|H)$ bei MDL_{nopar} immer Null ist, da ohne eine Partitionierung auch keine Abweichung von der Trajektorie existiert. Als Nächstes muss ein lokales Optimum gefunden werden, welches der längsten Trajektorien-Partition p_ip_j entspricht, die die Gleichung 3.7 für jedes k mit $i < k \leq j$ erfüllt [21].

$$MDL_{par}(p_i, p_k) \leq MDL_{nopar}(p_i, p_k) \tag{3.7}$$

Am Anfang wird der Startpunkt der Trajektorie in die Menge charakteristischer Punkte aufgenommen. Danach geht man einen Punkt weiter und berechnet beide MDL-Kosten. Ist die Gleichung 3.7 weiterhin erfüllt, geht man zum nächsten Punkt über. Sobald diese Gleichung nicht mehr erfüllt ist, die Kosten bei einer Partitionierung also höher wären, als ohne Partitionierung, steht fest, dass das lokale Optimum am vorherigen Punkt gefunden wurde. Dieser Punkt wird in die Menge charakteristischer Punkte aufgenommen, da sich bei Hinzunahme weiterer Punkte die Partitionierung verschlechtern würde. Der Punkt wird als neuer Startpunkt festgelegt und der Vorgang wird wiederholt, bis das Ende der Trajektorie erreicht wird. Der Endpunkt der Trajektorie wird dann ebenfalls als charakteristischen Punkt in die Menge aufgenommen. In Algorithmus 1 sieht man den Pseudoalgorithmus von Lee et al. [21] in welchem die einzelnen Schritte gut erkenntlich sind. Es handelt sich bei diesem Ansatz, wie schon erwähnt, aber um ein approximierendes Verfahren. In Abbildung 3.5 ist ein Fall dargestellt, in welchem der Algorithmus nicht die optimale Lösung findet. Die geringe Abweichung von der optimalen Lösung ist aber durch die gute Laufzeitkomplexität von $O(n)$ zu rechtfertigen, wobei n der Anzahl der Punkte der Trajektorie entspricht. Die Laufzeit ergibt sich dadurch, dass jeder Punkt nur einmal betrachtet werden muss. Am Schluss erhält man eine Menge charakteristischer Punkte, welche noch zu Liniensegmenten verbunden werden müssen.

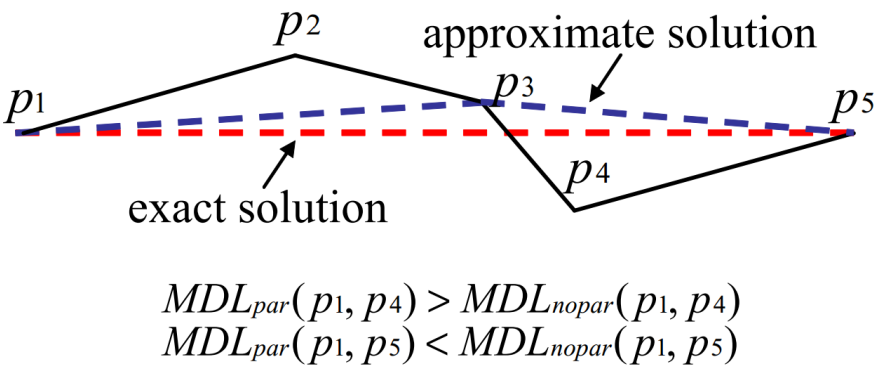


Abbildung 3.5: Ein Beispiel, bei welchem der Algorithmus nicht die optimale Lösung findet [21].

Algorithmus 1: Approximate Trajectory Partitioning

input : A trajectory $TR_i = p_1 p_2 p_3 \dots p_j \dots p_{len_i}$
output : A set CP_i of characteristic points

Algorithm:
Add p_1 into the set CP_i ; /* the starting point */
1 $startIndex := 1, length := 1$;
2 **while** $startIndex + length \leq len_i$ **do**
3 $currIndex := startIndex + length$;
4 $cost_{par} := MDL_{par}(p_{startIndex}, p_{currIndex})$;
5 $cost_{nopar} := MDL_{nopar}(p_{startIndex}, p_{currIndex})$;
/* check if partitioning at the current point makes the MDL cost larger than not partitioning */
6 **if** $cost_{par} > cost_{nopar}$ **then**
/* partition at the previous point */
7 Add $p_{currIndex-1}$ into the set CP_i;
8 $startIndex := currIndex - 1, length := 1$;
9 **else**
10 $length := length + 1$;
11 **end**
12 **end**
13 Add p_{len_i} into set CP_i; /* the ending point */

3.3 Clustering

Auch der Algorithmus zum Clustering der Liniensegmente ist Teil des Partition-and-Group Framework von lee et al. [21]. Es handelt sich dabei um den zweiten Teil des Frameworks, also die „Group“ Phase. Das Clustering beruht auf dem Prinzip von DBSCAN [8], hat aber einige kleine Abweichungen. Das Prinzip von DBSCAN ist, Ketten von Elementen zu finden, indem von jedem Element ausgehend in einem bestimmten Radius (der Parameter dafür heißt „Epsilon“ und beschreibt die Distanz) nach weiteren Elementen gesucht wird. Falls ein solcher Punkt genügend andere Punkte in seiner Umgebung hat (der Parameter dafür heißt meist „minPts“), wird er der Kette hinzugefügt. Alle so verbundenen Elemente werden dann zu einem Cluster zusammengefasst. Andere Elemente werden als Rauschen erkannt und ignoriert, oder gesondert zurückgegeben. Auf diese Weise können mehrere dichteverbundene Cluster

gefunden werden. Bei dem Ansatz von Lee et al. werden in der Clustering-Phase die Liniensegmente nach ihrer Ähnlichkeit gruppiert und in einzelnen Clustern zusammengefasst. Auf diese Weise ist es möglich, interessante Muster in den Daten zu finden, die bei der Analyse wichtige Erkenntnisse liefern können. Außerdem wird im Verlauf des Algorithmus Rauschen erkannt. Somit ist gewährleistet, dass die Ergebnisse nicht von Datenfehlern oder ungewöhnlichen Teilabschnitten der Trajektorien beeinflusst werden. Um das Clustering zu verstehen, wird zuerst das Prinzip des Clusterings dargestellt. Dadurch wird der nachfolgende Abschnitt über die verfahrenstechnische Realisierung übersichtlicher, welcher die Realisierung des Clusterings adressiert.

3.3.1 Prinzip des Clusterings

Die vorliegende Arbeit nutzt ein dichtebasiertes Verfahren. Die Liniensegmente, welche durch die Partitionierung entstanden sind, werden anhand ihrer Dichte gruppiert. Wie nahe zwei Elemente zueinander sind wird über die Distanzfunktion errechnet, welche in Kapitel 3.1 erklärt wurde. Die Vorgehensweise bei diesem Verfahren ist wie folgt: Man startet bei einem beliebigen Liniensegment und misst die Distanz, gemäß der Distanzfunktion, zwischen diesem und jedem anderen Liniensegment. Ist die Distanz zu einem anderen Liniensegment kleiner als ein Parameter Epsilon (ε), so wird das Element in die Epsilon-Nachbarschaft des initialen Segments aufgenommen. Wenn alle anderen Elemente abgearbeitet worden sind, wird geprüft, wie viele Elemente sich in der Epsilon-Nachbarschaft von dem Startsegment befinden. Sind mehr als ein Minimum (später MinLns genannt) an Elementen enthalten, so wurde ein Cluster gefunden, welches aus dem Startsegment mit allen seinen Nachbarn besteht. Sind weniger als ein Minimum an Segmenten in der Nachbarschaft, so werden alle Elemente als Rauschen markiert. Die Elemente können aber im Verlauf des Clusterings noch anderen Gruppen als Randelemente zugeordnet werden. Wurde ein Cluster mit einer minimalen Anzahl an Elementen gefunden, wird das initiale Liniensegment zu einem sogenannten Kernelement. Ausgehend von den Nachbarn des Kernelements wird der Vorgang wiederholt. Auf diese Weise bilden sich dichteverbundene Ketten, oder Bereiche, welche dem Cluster des Kernelements hinzugefügt werden. Wenn keine weiteren Nachbarn mehr auffindbar sind, ist das Cluster maximal expandiert. Dann wird der gesamte Vorgang auf einem anderen, ungeclusterten Liniensegment neu gestartet. Dieses Schema wird so lange durchgeführt, bis alle Segmente als Clusterelement oder Rauschen deklariert wurden. In einem letzten Schritt wird überprüft, aus wie vielen unterschiedlichen Trajektorien die Liniensegmente jedes Clusters stammen. Dadurch lassen sich Cluster herausfiltern, welche aus zu wenigen Trajektorien extrahiert wurden und deshalb eventuell nicht aussagekräftig genug sind.

3.3.2 Realisierung

Um die technische Realisierung zu erklären, dient der Pseudoalgorithmus von Lee et al. [21], welcher in Algorithmus 2 zu sehen ist, als Orientierung.

Algorithmus 2: Line Segment Clustering

input : (1) A set of line segments $\mathcal{D} = \{L_1, \ldots, L_{num_{in}}\}$
(2) Two parameters ϵ and $MinLns$
output : A set of clusters $\mathcal{O} = \{C_1, \ldots, C_{num_{clus}}\}$

Algorithm:

```
/* STEP 1 */
```
Set $clusterId$ to be 0; `/* an initial id */`
Mark all the line segments in $\mathcal{D}$ as *unclassfied*;
$startIndex := 1, length := 1$;
foreach $L \in \mathcal{D}$ **do**
  if L *is unclassfied* **then**
    Compute $N_\epsilon(L)$;
    if $|N_\epsilon(L)| \geq MinLns$ **then**
      Assign $clusterId$ to $\forall X \in N_\epsilon(L)$;
      Insert $N_{\epsilon(L)} - \{L\}$ into the queue $\mathcal{Q}$;
      `/* STEP 2 */`
      ExpandCluster($\mathcal{Q}, clusterId, \epsilon, MinLns$);
      Increase $clusterId$ by 1; `/* a new id */`
    else
      Mark L as *noise*;
    end
  end
  `/* STEP 3 */`
  Allcoate $\forall L \in \mathcal{D}$ to its cluster $C_{clusterId}$;
  `/* check trajectory cardinality */`
  foreach $C \in \mathcal{O}$ **do**
    `/* a threshold other than` $MinLns$ `can be used */`
    if $(|PTR(C)| < MinLns)$ **then**
      Remove C from the set $\mathcal{O}$ of clusters;
    end
  end
end
`/* STEP 2: compute a density-connected set */`
ExpandCluster $(\mathcal{Q}, clusterId, \epsilon, MinLns)$
  while $\mathcal{Q} \neq \varnothing$ **do**
    Let M be the first line segment in $\mathcal{Q}$;
    Compute $N_\epsilon(M)$;
    if $|N_\epsilon(M)| \geq MinLns$ **then**
      foreach $X \in N_\epsilon(M)$ **do**
        if X *is unclassified or noise* **then**
          Assign $clusterId$ to X;
        end
        if X *is unclassified* **then**
          Insert X into the queue $\mathcal{Q}$;
        end
      end
      Remove M from the queue $\mathcal{Q}$
    end
  end

Als Eingabe erwartet der Algorithmus zuerst alle Liniensegmente, sowie die zwei Parameter ε für die maximale Entfernung benachbarter Liniensegmente und $MinLns$ für die minimale Anzahl an Nachbarn. Bevor der Algorithmus startet, wird eine initiale Cluster-ID gewählt, damit später alle Cluster anhand ihrer ID unterschieden werden können. Anschließend muss allen Liniensegmenten L_x der Status „unclassified“ zugewiesen werden, was aber auch direkt

nach der Partitionierung, bei der Erstellung der Liniensegmente, getan werden kann. Jedes Liniensegment kann drei mögliche Status annehmen: „unklassifiziert“, „klassifiziert“ und „Rauschen“. Der Status kann für jedes Element in einer Integer-Variablen festgehalten werden: 0 für „unklassifiziert“, 1 für „klassifiziert“ und 2 für „Rauschen“. Sind diese Vorbereitungen getroffen, so beginnt der eigentliche Clustervorgang.
Dabei wird ein Liniensegment L_i aus der Menge der Liniensegmente genommen. Ist dieses unklassifiziert wird seine ε-Nachbarschaft berechnet. Diese ist definiert als $N_\varepsilon(L_i) = \{L_j \in D | dist(L_i, L_j) \leq \varepsilon\}$, wobei D der Menge aller Liniensegmente angibt. $dist(L_i, L_j)$ entspricht dem Abstand anhand der Distanzfunktion. Die ε-Nachbarschaft eines Liniensegments L_i lässt sich am Einfachsten mit einer Schleife berechnen, in welcher die Distanz zwischen L_i und allen anderen Elementen berechnet wird. Die Elemente, für welche die Distanz kleiner als ε ist, werden in die ε-Nachbarschaft $N_\varepsilon(L_i)$ aufgenommen. Diese Methode ist nicht sehr effizient, da für jedes Liniensegment die Distanz zu allen anderen Liniensegmenten berechnen werden muss. Das führt zu einer quadratischen Laufzeit $O(n^2)$ (n ist die Anzahl der Liniensegmente). Denkbar wäre hier eine Indexierung zu verwenden, so dass pro Liniensegment nur eine kleinere Anzahl anderer Liniensegmente betrachtet werden muss. Ist die ε-Nachbarschaft für das anfängliche Liniensegment berechnet, so wird überprüft, ob diese mindestens $MinLns$ Elemente enthält. Ist dies der Fall, hat der Algorithmus das erste Cluster gefunden und weist allen Elementen in $N_\varepsilon(L_i)$ die aktuelle Cluster-ID zu. Ansonsten wird L_i vorerst als Rauschen markiert.
Wurde aber ein Cluster gefunden, wird dieses erweitert, indem für alle Clusterelemente, außer dem Kernelement L_i, sprich $N_\varepsilon(L_i) - \mid L_i \mid$, die Unterfunktion „ExpandCluster“ aufgerufen wird. Diese ist Teil des Clusteralgorithmus, kann aber außerhalb definiert werden. Davor wird eine Warteschlange Q erstellt, in welche $N_\varepsilon(L_i) - \mid L_i \mid$ eingefügt wird. Beim Aufruf von „ExpandCluster“ werden die Elemente der Schlange Q, sowie die aktuelle Cluster-ID, ε und $MinLns$ an die Funktion übergeben. Sie berechnet dann für jedes Element in Q wieder N_ε und überprüft ob $|N_\varepsilon| \geq MinLns$ ist. Trifft das zu, werden die neu gefundenen Elemente aus der ε-Nachbarschaft dem aktuellen Cluster hinzugefügt, falls sie nicht bereits einem anderen Cluster angehören, und der Warteschlange Q angehängt, falls sie bisher nicht als Rauschen deklariert wurden. Das aktuelle Element, mit welchem N_ε berechnet wurde, wird aus der Schlange entfernt.

Dieser Vorgang wird für alle Elemente in Q wiederholt, bis Q leer ist. So wird eine dichteverbundene Menge an Liniensegmenten gefunden und das Cluster wird vergrößert, bis keine weiteren dichteverbundenen Elemente mehr gefunden werden. Das Cluster hat dann, für die aktuellen Parameter, seine maximale Größe erreicht. Anschließend wird die Cluster-ID erhöht und der gesamte Algorithmus für alle weiteren Elemente aus D erneut durchgeführt, um weitere Cluster zu finden. Wurden alle Elemente aus D abgearbeitet, werden die Cluster mit der jeweiligen ID erstellt und alle Liniensegmente anhand ihrer zugeteilten Cluster-ID in die Cluster eingefügt. Dann folgt der letzte Schritte, der die eigentliche Veränderung zu DBSCAN [8] darstellt. Der Algorithmus prüft aus wie vielen verschiedenen Trajektorien die Liniensegmente eines Clusters entnommen wurden. Ist diese Zahl kleiner als ein gewisser Schwellenwert (In Algorithmus 2, Zeile 18 ist dieser Wert MinLns), so wird das Cluster entfernt. Dies hat zum Ziel, allgemeinere Erkenntnisse durch die Cluster zu gewinnen, da solche Cluster entfernt werden, welche spezifisch nur auf wenige Trajektorien zutreffen.

4 Anpassungen des Verfahrens

Um diesen Ansatz interaktiv zu gestalten, müssen einige Anpassungen an den Komponenten vorgenommen werden. In Abbildung 4.1 ist dargestellt, an welchen Stellen dieser Ansatz Interaktionsmöglichkeiten und Visualisierung gewährleistet. In jedem Schritt lassen sich Erkenntnisse gewinnen und auf andere Schritte übertragen.

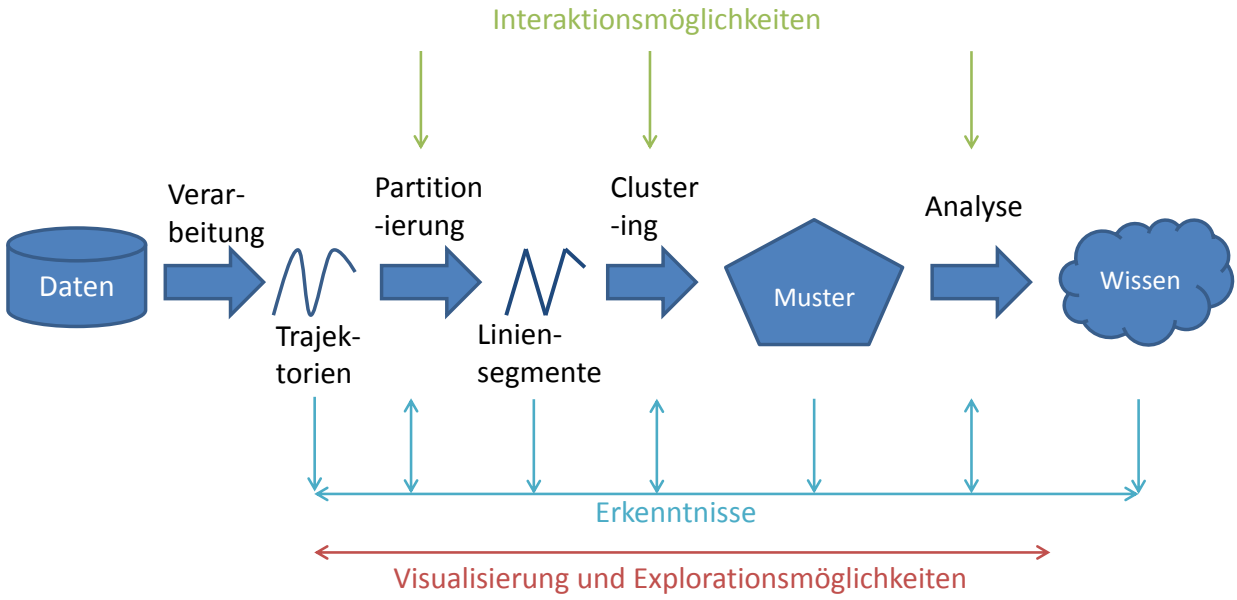


Abbildung 4.1: Ablauf, Aktionsmöglichkeiten, Erkenntnisübertragung und Visualisierung des vorgeschlagenen Ansatzes.

Damit Nutzer interaktiv Einfluss auf die Partitionierung nehmen können, ist eine Modifikation dieser nötig. Dazu wird die Grenze für zulässige Abweichungen variabel gestaltet und nicht statisch wie in [21]. Zu diesem Zweck wird ein weiteren Parameter namens „PartitionParam“ in den Algorithmus 1 in Zeile 6 eingefügt.

$$if(cost_{par} > cost_{nopar} / PartitionParam) then \tag{4.1}$$

Dieser Parameter wird später bei der Initialisierung einer Partitionierung direkt von der Benutzeroberfläche abgegriffen. Durch das Einstellen des Parameters kann der Nutzer Einfluss auf die Granularität der Partitionierung nehmen. Die späteren Ergebnisse der Mustererkennung sind davon abhängig. Wird der Wert von PartitionParam größer als eins gewählt, so wird $cost_{nopar}$ kleiner und es entstehen mehr und genauere Partitionen. Ist er dagegen kleiner als eins, entstehen weniger und ungenauere Partitionen. Der Nutzer kann folglich selbst entscheiden, wie stark die Partitionierung ausfallen soll. Wird eine Trajektorie besonders stark partitioniert, bleiben nur noch der Start- und der Endpunkt vorhanden und es entsteht nur ein Liniensegment. Bei einer besonders schwachen Partitionierung folgen die entstehenden Linien exakt der Trajektorie, da alle Punkte als charakteristische Punkte markiert werden. Dies sind keine Nachteile, denn in manchen Anwendungsfällen können solche extremen Partitionierungen durchaus Sinn machen.

Zur Realisierung von Interaktionen mit dem Clustering sind ebenfalls Anpassungen notwendig. Dazu können aber bereits implementierte Parameter verwendet werden. Das Abgreifen der Parameter ε und $MinLns$ von der Nutzeroberfläche ermöglicht Interaktionen, welche mit festen Werten nicht durchführbar sind. Der Nutzer erhält wesentlichen Einfluss auf den Prozess und kann diesen zu einem gewissen Grad anpassen und steuern. Eine weitere Anpassung betrifft den Schwellenwert für die Cluster-Filterung. Der Schwellenwert in Algorithmus 2 Zeile 18, für den noch $MinLns$ verwendet wird, lässt sich durch einen anderen Parameter ersetzen, welcher ebenfalls später von der Nutzeroberfläche abgegriffen wird. Somit kann der Nutzer diesen letzten Filterschritt beliebig einstellen. Auch das Eintragen des Wertes 0 ist für diesen Parameter erlaubt. Durch diesen Wert wird die Filterung deaktiviert.

Auch die Berechnung der Distanz zweier Linien beim Clustering kann angepasst werden. Die Gewichtungen der drei einzelnen Distanzkomponenten lassen sich ebenfalls über die Benutzeroberfläche einstellen. Das wirft aber das Problem auf, dass bei jedem Clustervorgang mit neuen Gewichtungen andere Distanzen zwischen den Elementen gefunden werden. Jedes Mal müsste ε erst auf die neuen Distanzen abgestimmt werden. Um dieses Problem zu umgehen, wird die Gesamtdistanz $dist(L_i, L_j)$ in dem hier entwickelten Ansatz, anders als bei der Vorlage [21], noch durch die Summe aller Gewichtungen geteilt (Formel 4.2).

$$dist(L_i, L_j) = (w_\perp * d_\perp(L_i, L_j) + w_\parallel * d_\parallel(L_i, L_j) + w_\theta * d_\theta(L_i, L_j))/(w_\perp + w_\parallel + w_\theta) \qquad (4.2)$$

Dadurch wird vermieden, dass der ε-Wert für verschiedene Gewichtungen neu eingestellt werden muss.

Eine weitere wichtige Anpassung ist die strikte Trennung von Partitionierung und Clustering. Im Ansatz von Lee et al. wird das Clustering direkt nach der Partitionierung ausgeführt. Im vorliegenden Ansatz wurden die Methoden aber so implementiert, dass sie einzeln aufgerufen werden können. Das ermöglicht Nutzern mehrere Partitionierung hintereinander durchzuführen. Sie können die Ergebnisse überprüfen, ohne direkt danach ein Clustering durchführen zu müssen. Ebenfalls können mehrere Clusterings mit den Daten der letzten Partitionierung ausgeführt werden, ohne dabei jedes Mal neue Partitionen erstellen zu müssen.
Dazu werden die Liniensegmente der letzten Partitionierung und des letzten Clusterings

in einer Zwischenstruktur festgehalten. Diese Zwischenstruktur kann jederzeit ausgelesen werden und bildet damit auch die Grundlage für die kontinuierliche Visualisierung. Die letzte durchgeführte Anpassung des datenverarbeitenden Verfahrens ist demnach die Einbindung in die visuelle und interaktive Umgebung.

5 Visuelle und interaktive Komponenten

Eine gute Visualisierung ist unerlässlich für visuelle Analysen. Ist die Visualisierung nicht ausreichend gut, kann die Analyse auch keine guten Ergebnisse hervorbringen. Eine Visualisierung muss Klarheit schaffen. Der Nutzer muss in der Lage sein, die dargestellten abstrakten Daten schnell zu begreifen und intuitiv zu verarbeiten. Die Arbeit [29] beschäftigt sich mit solchen Anforderungen, welche an die Visualisierung gestellt werden. Sie dient als Schnittstelle zwischen dem Benutzer und dem Programm und ermöglicht die Kommunikation zwischen den beiden. Diese Kommunikation bildet die Grundlage zur Integration interaktiver Methoden. Ohne ein Feedback eines Programmes ist der Nutzer immer im Ungewissen, was seine Interaktion ausgelöst hat. Aus diesem Grund sind die Interaktion und die Visualisierung stark aneinander gebunden und ermöglichen zusammen eine beidseitige Kommunikation zwischen Nutzer und Programm. Die Schnittstellen, welche durch die Visualisierung realisiert werden können, machen den vorliegenden Ansatz erst interaktiv. Aus diesem Grund erläutern die folgenden Unterkapiteln jede Schnittstelle, beziehungsweise Komponente gesondert. Es wird auf die Relevanz jeder Schnittstelle eingegangen, erklärt, wie sie in den Programmfluss eingreift und gezeigt, wie sie dargestellt wird. Das visuelle System gliedert sich dabei in eine interaktive Karte, ein Konfigurationsmenü und ein kleines Ausgabefeld. Diese bilden zusammen die Benutzeroberfläche, welche in Abbildung 5.1 dargestellt ist. Zur Realisierung der hier genannten Komponenten wird Java genutzt mit der Programmierschnittstelle Java Swing[1].

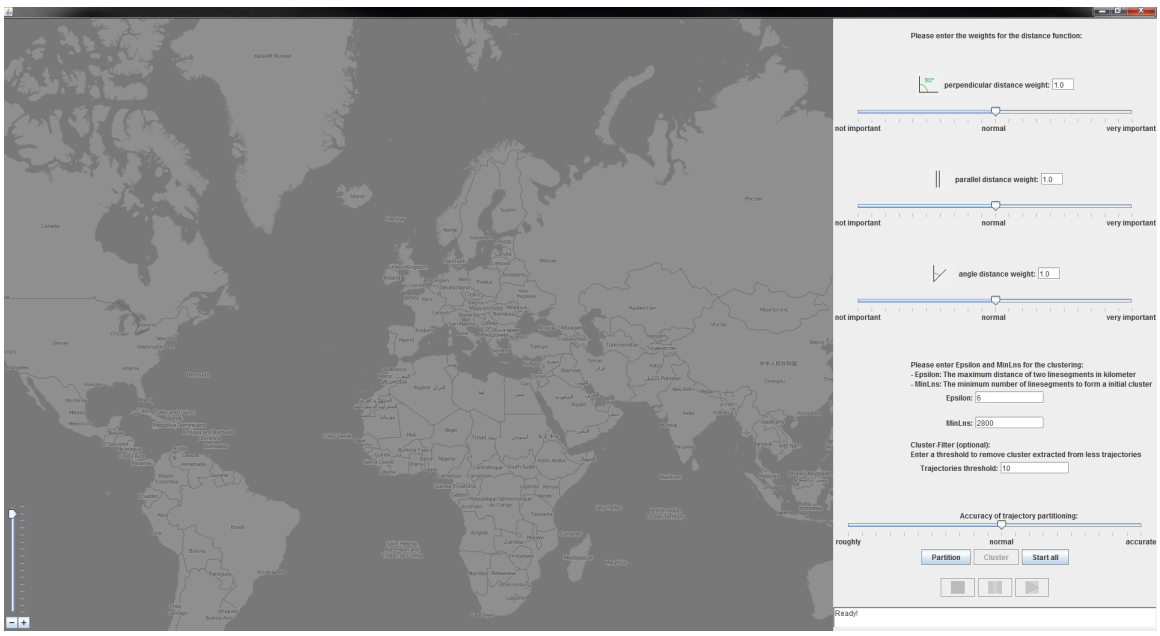


Abbildung 5.1: Die Benutzeroberfläche des vorgestellten Ansatzes.

[1] Für diesen Ansatz wurde SwingX 1.6.2 verwendet.

5.1 Interaktive Karte

Um die Ergebnisse der berechnenden Vorgänge genau analysieren zu können, nimmt eine Karte den größten Teil der Benutzeroberfläche ein. Diese dient als geeignete Umgebung, um die Bewegungen darauf zu visualisieren. Die hier verwendete Karte ist „OpenStreetMap“[2]. Sie enthält eine komplette Weltkarte in verschieden stark hineingezoomten Ansichten. JXMapkit von Java Swing liefert die zugehörigen Methoden, um auf der Karte zu zeichnen oder mit ihr zu interagieren. Ein Beispiel ist die Zoomfunktion. Durch das Zoomen auf bestimmte Bereiche lassen diese sich besser analysieren. In Abbildung 5.1 kann man unten links die Zoomstufe sehen, welche der Nutzer nach Belieben einstellen kann. Zum Zoomen kommt natürlich die Möglichkeit, die Karte via „Drag and Drop“ auf die gewünschten Bereiche einzustellen. Dies sind Standardwerkzeuge, welche man beim visuellen Analysieren von Bewegungsdaten stets zur Verfügung haben sollte. Trotz ihrer Einfachheit gewährleisten sie einen guten Überblick über die Daten. Trajektorien und Liniensegmente werden auf der Karte visualisiert, indem Linien zwischen Koordinatenpunkten gezogen werden. Um diese Punkte zu zeichnen, müssen lediglich die Punktkoordinaten in Pixel-Koordinaten umwandelt werden, wofür JXMapkit die passenden Strukturen zur Verfügung stellt. Durch diese Art der Visualisierung gehen keine Informationen verloren, da alle Daten des Datensatzes verwendet werden.

Die gewählten Farben bei der visuellen Darstellung von Objekten spielen ebenfalls eine wichtige Rolle. Die Karte ist grau gehalten, ohne viele Details. Das ermöglicht die Nutzung einer große Palette an Farben zum Darstellen der Elemente. Direkt nach der Partitionierung werden alle Liniensegmente in weiß gezeichnet. Das gibt zum Einen einen guten Kontrast zur Karte, wodurch der Nutzer genau die Formen der Liniensegmente verfolgen kann. Zum Anderen ist Weiß eine neutrale Farbe, was verdeutlicht, dass die Elemente noch nicht fertig verarbeitet wurden. In [29] wird vorgeschlagen für solche unfertig verarbeiteten Daten eine neutrale Farbe zu verwenden. Rauschen wird ebenfalls weiß gezeichnet. Das liegt daran, dass es keinem Cluster angehört und nicht so einen hohen Stellenwert besitzt wie die Cluster. Jedes Cluster bekommt eine eigene Farbe, in welcher alle Elemente des Cluster gezeichnet werden. Diese Farben müssen möglichst gegensätzlich zueinander sein, damit man die Cluster bei einer Überlagerung noch auseinander halten kann [29]. Bei zu vielen Clustern ist das aber nicht mehr möglich, da die Farben, ab einer gewissen Anzahl, zu ähnlich sind. Mit Hilfe von Colorbrewer 2.0[3] und [29] wurden deshalb Farben festgelegt, welche gut voneinander zu unterscheiden sind.

Beim Zeichnen auf der Karte wird eine Methode verwendet, um Überlappungen deutlich zu machen und die Aufmerksamkeit der Nutzer zu lenken. Dazu werden die einzelnen Linien nur mit etwa 30% Opazität gezeichnet. Bei der Überlagerung mehrere Linien entsteht dabei ein visueller Effekt, der die Farben an Linienschnittpunkten intensiver erscheinen lässt. So lassen sich dichte Stellen in den Clustern besonders gut detektieren. Andrienko et al. verwenden das gleiche Prinzip in der Arbeit [1].

[2] `http://orm.openstreetmap.org`
[3] `http://colorbrewer2.org/`

Wie bereits anfangs angedeutet, ist die Kontinuität der Visualisierung ein wichtiges Anliegen bei diesem Ansatz. Um diese realisieren zu können, müssen die berechnenden Methoden, also Partitionierung und Clustering, parallelisiert werden. Ansonsten würden die Berechnung die Reaktionszeit der Benutzeroberfläche mitsamt Karte zu stark beeinflussen. Die Parallelisierung wird anhand von Threads realisiert und später im Detail erklärt. Um die Kontinuität der Visualisierung zu gewährleisten muss die Zeichenroutine jederzeit Zugriff auf alle entstehenden Elemente besitzen. Dafür werden alle Liniensegmente in einer separaten Klasse aufbewahrt. Nach der Partitionierung einer Trajektorie werden die dadurch entstandenen Liniensegmente an diese Klasse übergeben. Die Zeichenroutine liest die Linien der Klasse aus und führt alle 100ms Neuzeichnungen durch, während die Partitionierung noch im Gange ist. Linien welche sich nicht verändert haben, werden auch nicht nochmal gezeichnet, sondern bleiben bestehen. Dadurch sieht der Nutzer in Echtzeit, welche Linien neu dazu kommen. Ist die Partitionierung abgeschlossen, wird auch die regelmäßige Neuzeichnung deaktiviert. Wird der Kartenausschnitt durch Zoomen oder Verschieben geändert, wird eine Neuzeichnung durchgeführt. Der Nutzer kann also während und nach dem Vorgang die Ergebnisse frei explorieren. Die Zeichenroutine zeichnet aus Performanz-Gründen immer nur die Linien und Kartenstücke, welche gerade im Kartenfenster der Benutzeroberfläche sind. Der momentane Kartenausschnitt wird dabei als Rechteck definiert. Die minimalen und maximalen Koordinaten des aktuellen Kartenausschnitts können dann an den Ecken des Rechtecks abgelesen werden. Liegen Linien innerhalb dieses Rechtecks, werden sie gezeichnet, ansonsten nicht. Es muss demnach nur das gezeichnet werden, was gerade vom Nutzer gesehen werden kann. Auch der Clusterprozess wird, wie die Partitionierung, durch regelmäßige Neuzeichnungen realisiert. Am Anfang des Clusterings werden aber alle Linien bereits in weiß eingezeichnet, da diese schon in der Zwischenklasse hinterlegt wurden. Wird eine Linie einem Cluster zugewiesen, so wird sie bei der nächsten Neuzeichnung entsprechend eingefärbt. Der Nutzer kann also wieder jede Veränderung im Bild sofort erkennen und schon mit seiner Analyse starten bevor der Algorithmus die Cluster fertig berechnet hat.

5.2 Konfigurationsoberfläche

Betrachtet man die Benutzeroberfläche in Abbildung 5.1 kann man im rechten Bildbereich das Konfigurationsmenü erkennen. Dieses enthält Schnittstellen zur Kontrolle des Programmes und zum Einstellen der Berechnungen.

5.2.1 Kontrollschnittstellen

Die Kontrolle über das Programm erlangt der Nutzer über mehrere Schnittstellen. Dadurch kann er den Verlauf des Programmes steuern. In Abbildung 5.2 sieht man die Kontrollmechanismen, welche auf der Benutzeroberfläche über dem Ausgabetextfeld liegen.

Die oberen drei Knöpfe steuern den Kontrollfluss des Programmes. Ein Druck auf den Knopf „Partition“ leitet die Partitionierung in einem neuen Thread ein. Der komplette Vorgang der Partitionierung wird dabei visualisiert, wie bereits oben erwähnt. Der Knopf „Cluster“ ist in Abbildung 5.2 noch deaktiviert, weil keine Partitionierung durchgeführt wurde. Nach einer Partitionierung wird der Knopf automatisch aktiviert. Der Partition-Knopf wird aber nach

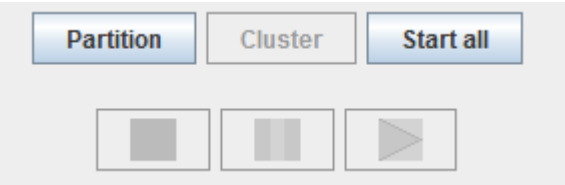


Abbildung 5.2: Kontrollmechanismen im Konfigurationsmenü des Systems.

einer Partitionierung noch nicht deaktiviert. Wie beschrieben, kann der Nutzer die Ergebnisse der Partitionierung visuell explorieren. Ist er mit diesen nicht zufrieden, kann er den Grad der Partitionierung regulieren und erneut den Partition-Knopf betätigen. Dabei werden die alten Liniensegmente gelöscht und es entstehen neue Partitionen. Dies kann beliebig oft durchgeführt werden, bis eine passende Partitionierung gefunden wurde.
Das Clustering wird erst eingeleitet, wenn der Cluster-Knopf gedrückt wurde. Dann wird auch der Partition-Knopf deaktiviert. Das Drücken des Knopfes startet einen neuen Thread in welchem das Clustering ausgeführt wird. So kann die Visualisierung während der Berechnung durchgeführt werden. Während dem Clustering sind alle der obigen drei Knöpfe deaktiviert. Nach dem Clustering werden sie reaktiviert. Der Nutzer kann dann zum Beispiel mit neuen Werten nochmals clustern, ohne davor partitionieren zu müssen. Alternativ könnte er auch eine neue Partitionierung durchführen.
Der dritte Knopf „Start all“ startet die Partitionierung gefolgt vom Clustering mit den aktuellen Parametern ohne eine Visualisierung. Nur die finalen Cluster werden am Ende der Berechnung gezeichnet. Diese Funktion eignet sich für Nutzern, welche schon Ergebnisse zu einer anderen Zeit erzeugt haben und diese schlicht reproduzieren wollen, um sie erneut zu explorieren. Alle Knöpfe werden dabei deaktiviert. Der einzige Vorteile dabei ist, dass die Zeit für die Visualisierung gespart wird. Diese Funktion sollte aber nur dann gewählt werden, wenn die Ergebnisse des Prozesses bereits bekannt sind, da ansonsten die Zeit verloren geht, die der Durchlauf benötigt.

Wurde das Clustering über den Cluster-Knopf gestartet, so werden zwei der drei darunterliegenden Knöpfe aktiv. Zur leichteren Verständlichkeit nutzen diese bekannte Formen und Farben wie man in Abbildung 5.3 sehen kann [29].

Abbildung 5.3: Steuerungsknöpfe für den Clustervorgang. Hier sind alle Knöpfe gleichzeitig aktiviert, was in einem regulären Programmablauf niemals vorkommt, sondern nur der Darstellung dient.

Der linke Knopf stoppt den Thread, in welchem das Clustering ausgeführt wird. Dadurch lässt sich der Clustervorgang frühzeitig beenden, falls er eine falsche Tendenz aufweist. Danach kann der Vorgang mit anderen Parametern neu gestartet werden. Diese Funktion hat sich bei den durchgeführten Versuchen als äußerst nützlich erwiesen. Viele Parameterwerte und Einstellungen lassen sich dadurch in kurzer Zeit testen.
Auch der zweite Knopf ermöglicht eine Steuerung während dem Clustering. Er pausiert den

aktuellen Thread des Clusterings. Die bisherigen Ergebnisse lassen sich weiterhin explorieren. Während der Pause wird der dritte Knopf aktiviert. Dieser setzt einfach den pausierten Thread fort. Durch das Pausieren und Fortsetzen kann jeder einzelne Schritt und jede Bewegung genau betrachtet werden. Die Möglichkeit, das Clustering zu beenden, besteht dabei weiterhin. Um den Nutzer nicht zu überfordern, oder gar zu verwirren, werden stets alle Knöpfe ausgeblendet, die zu gegebenem Zeitpunkt keine Funktion erfüllen.
Durch diese Möglichkeiten der Kontrolle können Nutzer die Prozesse individuell steuern und haben zu jedem Zeitpunkt die Optionen einer Analyse oder Interpretation, noch bevor die Prozesse beendet sind.

5.2.2 Einstellungen

Parameter sind ebenfalls Schnittstelle zwischen Nutzer und Programm. In der bisherigen Ausarbeitung wurden die hier verwendeten Parameter bereits eingeführt. Der Erste davon der wichtig wird, ist der Grad der Partitionierung. Er wird vor der Partitionierung über einen Schieberegler eingestellt, welcher sich direkt über den Kontrollknöpfen befindet. Der Regler ist in Abbildung 5.4 dargestellt.

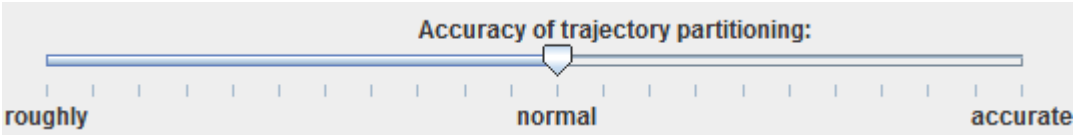


Abbildung 5.4: Schieberegler zur Festlegung des Grades der Partitionierung.

Durch das Verstellen dieses Reglers wird der Wert des Parameter „PartitionParam“ verändert, welcher in Kapitel 4 eingeführt wurde.
Der Regler steht anfangs auf „normal“ und hat somit den Wert 1. Dadurch wird die Partitionierung nicht beeinflusst und nutzt die selbe Einstellung, welche in [21] als Standard verwendet wird. Bewegt man den Regler mehr nach links, wird PartitionParam immer kleiner, bis er ganz links den Wert $0,01$ annimmt. Dadurch werden weniger und gröbere Liniensegmente erzeugt, weshalb die linke Seite des Reglers mit „roughly“ markiert wurde. Schiebt man den Regler in die andere Richtung bis zu „accurate“ wird der Parameter immer größer bis er schließlich den Wert 2 erreicht. Der Wert ist so groß, dass die entstehenden Liniensegmente der Trajektorie ziemlich genau folgen. Dafür sind sie aber sehr vielzählig. Der Nutzer partitioniert so lange mit verschiedenen Einstellungen, bis er eine Partitionierung gefunden hat, welche sich für seinen Anwendungsfall eignet.

Nicht nur die Genauigkeit der Partitionierung, sondern auch andere Parameter sind im Verlauf des Programmes wichtig. Die weiteren Parameter beziehen sich aber alle auf den Clustervorgang. Die Gewichtungen für die Distanzmaße lassen sich über Schieberegler individuell einstellen. In Abbildung 5.5 sind diese Regler dargestellt.

Kleine Piktogramme sollen dem Nutzer dabei helfen, die einzelnen Distanzkomponenten zu identifizieren. Die Wertebereiche der Regler sind mit „not important“ bis „very important“ beschriftet. Durch das Verschieben der Regler können die Nutzer selbst bestimmen, wie wichtig ihnen die einzelnen Komponenten sind. Werden die Regler nach links bewegt, wird der

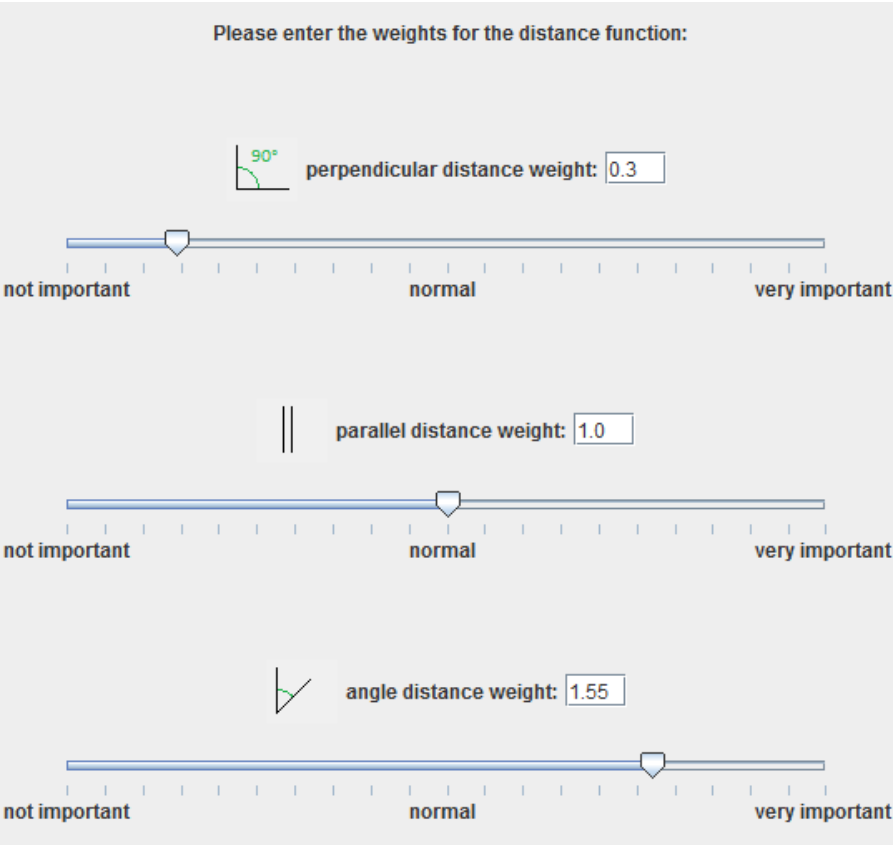


Abbildung 5.5: Schieberegler zur Festlegung der Gewichtungen für das Distanzmaß.

Wert der Gewichtung kleiner und die Komponente fällt bei der Distanzberechnung schwächer ins Gewicht. Bei einer Verschiebung nach rechts fällt die betroffene Komponente entsprechend stärker ins Gewicht. Für fortgeschrittene Nutzer, oder zum feineren Einstellen von Gewichtungen, sind zusätzlich Textboxen angebracht, in welche Werte direkt eingetragen werden können. Dabei sind auch Werte außerhalb des Reglerbereichs zulässig. Die Einstellung der Gewichtungen ermöglicht den Nutzern verschiedene visuelle Analysen durchzuführen, je nachdem worauf sie ihren Fokus legen.
Das Clustering benötigt aber noch weitere Parameter. Zwei davon sind ε und $MinLns$, welche bereits beim Verfahren erläutert wurden. Auch diese lassen sich vom Nutzer einstellen. Dafür stellt das Konfigurationsmenü Textboxen bereit, wie in Abbildung 5.6 zu sehen.

Kurze Beschreibungen sollen den Nutzer dabei unterstützen, die Werte richtig einzustellen. Die Entscheidung, an dieser Stelle keine Schieberegler zu benutzen, wurde deshalb getroffen, weil die sinnvollen Wertebereiche für ε und $MinLns$ stark vom Datensatz abhängen. Die Werte eines Schiebereglers müssten also für jeden Datensatz separat eingestellt werden. Die Textboxen sind einfacher zu realisieren und man kann sie schon mit vorgegebenen Werten initialisieren, um Nutzern eine Orientierungshilfe zu geben. Bei der Initialisierung des Clustervorgangs werden die Werte aus den Textboxen ausgelesen und als Parameter an den Algorithmus übergeben. Dadurch wird verhindert, dass Wertveränderungen während des Clustervorgangs die laufende Berechnung beeinflussen. Eine weitere Textbox ist direkt darunter

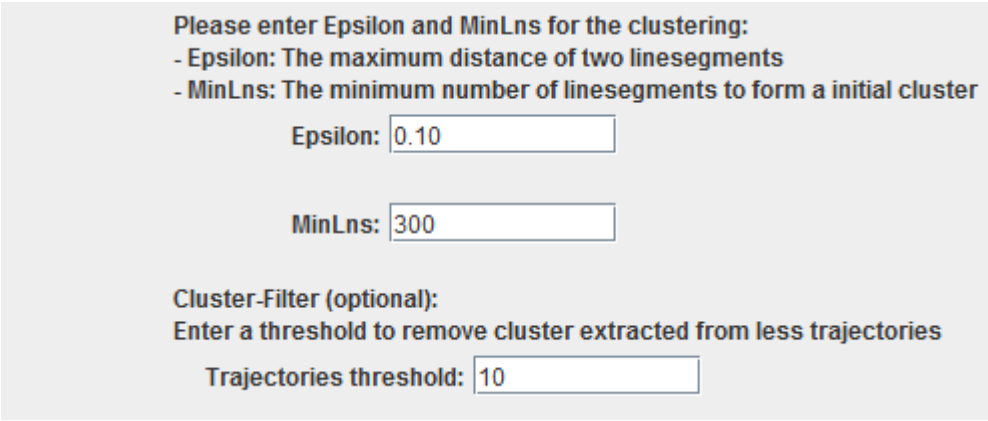


Abbildung 5.6: Textboxen für die Parameter des Clustering.

angebracht, in welcher der Schwellenwert für die Cluster-Filterung eingetragen werden kann. Diese Funktion wurde bereits eingeführt. Ihre Nutzung wird dem Nutzer frei gestellt, weshalb sie mit „optional“ gekennzeichnet wurde und zu Beginn den Wert 0 hat. Die ununterbrochene Visualisierung des Clusterings bringt auch bei der Filterung einen Vorteil. Während der Berechnung sieht der Nutzer alle Cluster. Erst am Schluss des Clusterings werden diejenigen Cluster entfernt, welche den Schwellenwert nicht einhalten. Die Informationen dieser Cluster gehen dadurch nicht sofort verloren, sondern können ebenfalls analysiert werden. Wenn ein Nutzer sich also für ein solches Cluster interessiert, kann er den Schwellenwert heruntersetzen und das Clustering wiederholen. Will ein Nutzer alle Cluster sehen, unabhängig davon, aus wie vielen verschiedenen Trajektorien-Teilen sie bestehen, kann er einfach den Schwellenwert auf 0 lassen.
Das Einstellen des Programmes ist dem Nutzer zu großen Teilen freigestellt. Die Ergebnisse werden stark von diesen Einstellungen beeinflusst. Deshalb wird später diskutiert, welche Vor- und Nachteile diese große Freiheit des Nutzers mit sich bringt.

5.3 Ausgabe

Das Programm kommuniziert nicht nur über die ständige Visualisierung der Prozesse mit dem Nutzer, sondern hat auch eine Textausgabe unterhalb der Konfigurationsoberfläche. Diese dient der Orientierung des Nutzers, indem immer der aktuelle Stand des Programmes ausgegeben wird. Das Textfeld teilt dem Nutzer anfangs mit, wann der Datensatz fertig eingelesen ist und das Programm bereit ist. Außerdem wird im Falle einer Partitionierung ausgegeben, welche Trajektorie aktuell verarbeitet wird und wie viele insgesamt zu verarbeiten sind. Das lässt sich mit Hilfe der IDs der Objekte durchsetzen. Nach der Partitionierung wird die Anzahl der entstanden Liniensegmente ausgegeben. Dies ist eine wichtige Information für den Nutzer, denn sie erleichtert ihm die Beurteilung der Partitionierung. Wenn der Nutzer das Clustering einleitet, wird ihm ebenfalls angezeigt, welche Linie gerade verarbeitet wird und wie viele insgesamt noch zu verarbeiten sind. Dadurch kann abgeschätzt werden, wie viel Zeit das Clustering noch in Anspruch nimmt und ob sich noch viel in den bisherigen Ergebnissen ändern wird. Somit hat man immer den aktuellen Status der Programmausführung vor Augen

und kann sich daran orientieren.

5.4 Analyse-Werkzeuge

Um Nutzern die Analyse zu erleichtern gibt es bereits eine Vielzahl an Analyse-Werkzeugen. Die meisten davon sind aber nur nach der Berechnung der Ergebnisse anwendbar. Für das hier vorgestellten Programm wurde ein solches Werkzeug implementiert, welches auch während dem Clustering funktionsfähig ist. Es zeigt zu jedem Cluster die ID, Farbe und Größe und bietet die Möglichkeit, die Visualisierung gewählter Cluster auszuschalten. Auch ungeclusterte Linien und Rauschen lassen sich dadurch ausblenden. Ein ähnliches Werkzeug wurde auch in der Arbeit „Visual Analytics Tools for Analysis of Movement Data“ [4] zur Exploration von Ergebnissen vorgestellt. In Abbildung 5.7 sieht man die Oberfläche zum Bedienen des hier implementierten Werkzeugs.

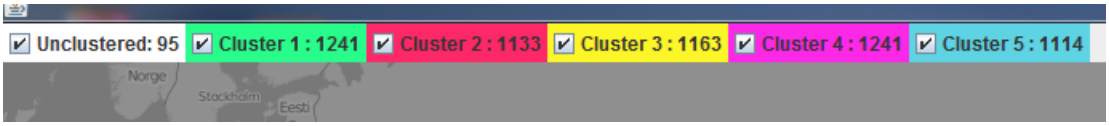


Abbildung 5.7: Werkzeug zum Ausblenden der Cluster.

Zu Beginn des Clusterings wird nur das erste Feld mit den ungeclusterten Elementen angezeigt. Sobald ein neues Cluster gefunden wurde, kommt das entsprechende Feld, mit der Cluster-ID und der Anzahl an Elementen des Clusters dazu. Die Felder werden in der Farbe des jeweiligen Clusters eingefärbt, damit Nutzer identifizieren können, welches Feld sich auf welches Cluster bezieht. Ist eine Zeile im Fenster des Programmes voll, so wird eine neue Zeile für weitere Felder reserviert. Die Anzahl der Elemente eines Clusters wird immer dann aktualisiert, wenn dem Cluster neue Elemente hinzugefügt werden. Wird ein Cluster im Laufe der Filterung entfernt, so verschwindet auch das entsprechende Feld. Jedes Feld verfügt über ein Auswahlkästchen. Damit kann die Visualisierung des entsprechenden Clusters deaktiviert beziehungsweise aktiviert werden. Das funktioniert dank der Neuzeichnungen der implementierten fortlaufenden Visualisierung. Die Zeichenmethode prüft stets, ob die zu zeichnenden Linien einem deaktivierten Cluster angehören. Ist dies der Fall, werden sie nicht gezeichnet. Wenn das Clustering beendet wurde, also keine automatischen Neuzeichnungen mehr durchgeführt werden, dann verursacht die Auswahl eines der Felder eine einmalige Neuzeichnung, damit die Methode weiterhin verwendbar ist. Rauschen wird mit den ungeclusterten Elementen im ersten Feld „unclustered“ zusammengefasst, da beide für die Analyse weniger Relevanz haben, als die Cluster. Deaktiviert man das erste Feld über das Auswahlkästchen, werden folglich nur noch Cluster angezeigt und alle weißen Elemente fallen weg. Das hilft beim Auffinden kleiner Cluster, welche von vielen ungeclusterten Elementen umschlossen sind. Die Übersicht des Nutzers wird dadurch verbessert. Wenn man ein einzelnes Cluster betrachten will, lohnt es sich deshalb oft, alle anderen Cluster auszublenden, um einen besseren Überblick zu bekommen. In den durchgeführten Versuchen mit dem Programm fand diese Funktion häufig Anwendung, da sie die Analyse vereinfacht.
Im letzten Kapitel wird ein Ausblick gegeben, welche anderen Werkzeuge noch sinnvolle Erweiterungen für diese Arbeit wären.

6 Systemeigenschaften

In diesem Kapitel soll das System des implementierten Programmes näher gebracht werden. Dafür wird zuerst auf die Architektur eingegangen. Die Beziehung der Komponenten, sowie Implementierungsdetails werden dabei erläutert. Anschließend wird die Parallelisierung des Programmes dargestellt und danach erklärt, welche Versuche zur Implementierung einer Indexstruktur unternommen wurden.

6.1 Architektur

Das vorgestellte Programm wurde in Java entwickelt und nutzt viele standardisierte Java-Methoden aus Java JRE 7 und Java Swing 1.6.2. In Abbildung 6.1 kann man die Architektur des Systems ablesen.

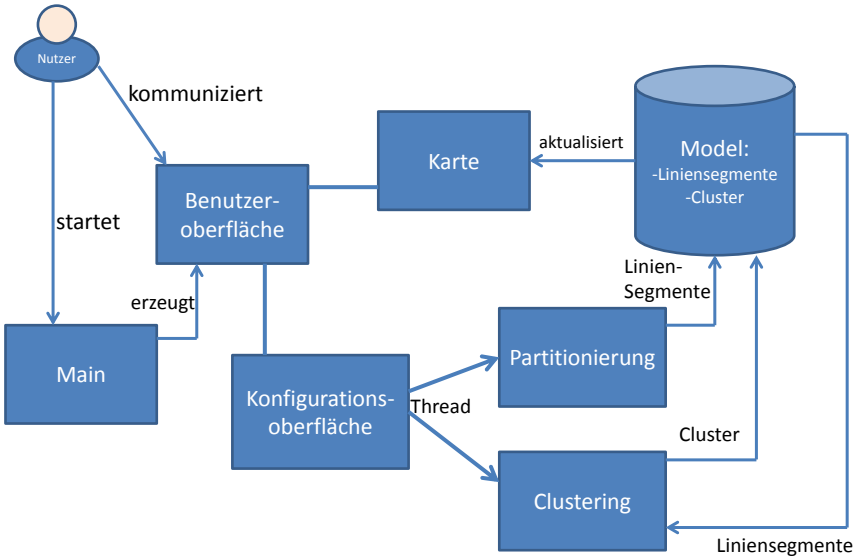


Abbildung 6.1: Architektur des Systems.

Durch Starten des Programmes wird zuerst die Main-Methode der Main-Klasse aufgerufen. Diese erstellt als Erstes eine Benutzeroberfläche, in welcher bereits die Karte und die Konfigurationsoberfläche integriert sind. Des Weiteren enthält Main ein File-Objekt „dir“, welches das Oberverzeichnis des Ordners mit den Bewegungsdaten enthält. Main führt eine Methode scan() aus, welche das Oberverzeichnis des Datensatzes nach Dateien durchsucht und diese in einem File-Array auflistet. Danach wird scanDirectory(fileList) mit diesem Array ausgeführt,

um alle Unterordner rekursiv nach Dateien zu durchsuchen. Auf diese Weise können alle möglichen Datensätze eingelesen werden, unabhängig davon, ob sie als einzelne Datei, oder in Form von mehreren Dateien vorliegen. Zusätzlich werden die gefunden Dateien nach Endungen gefiltert. So wird verhindert, dass Label-Dateien, oder Nutzerhandbücher in die Berechnung geraten. Für jede gefundene Datei wird das Ausgabefeld auf der Benutzeroberfläche mit der aktuellen Anzahl der Dateien aktualisiert.

Wurden keine weiteren Dateien mehr gefunden, wird der Array mit den Pfaden an eine weitere Methode übergeben, welche mit Hilfe eines BufferedReaders und Parsern die einzelnen Zeilen jeder Datei ausliest. Die Daten jeder Zeile werden als PositionPoint-Objekte gespeichert. Der Aufbau der PositionPoints folgt der Definition der Punkte in Kapitel 3. Ändert sich die TrajektorienID in einer Dateizeile, werden alle bisherigen Punkte zu einem Trajektory-Objekt zusammengefasst, welches ebenfalls wie in Kapitel 3 aufgebaut ist. Sind alle Trajektorien eingelesen, ist die Vorverarbeitung beendet, die Main-Klasse bleibt aber geöffnet, da die Benutzeroberfläche noch darin ausgeführt wird.

Alle Trajektorien liegen anschließend in einer Liste in der Main-Klasse vor und das Ausgabefeld im Menü teilt dem Nutzer mit, dass das Programm bereit ist. Die Karte der Benutzeroberfläche ist mit JXMapKit realisiert und greift online die Kartendateien von OpenStreetMap ab. Für das Zeichnen auf der Karte ist eine Overlay-Klasse zuständig, welche die Punktkoordinaten in Pixelkoordinaten umwandelt. Die Farben der Punkte werden, je nach Clusterzugehörigkeit, über eine Funktion in der Overlay-Klasse auswählt. Die Punkte sind als Bestandteile von LineSegment-Objekten in einer Model-Klasse hinterlegt, welche jederzeit von dem Overlay abgefragt werden kann. Weiterhin verfügt die Benutzeroberfläche über Eingabefelder (JTextField) und Knöpfe (JButton). Die Eingabefelder sind public, damit die Partitionierung und das Clustering darauf zugreifen können. Die Knöpfe sind über ActionListener implementiert. Wird der erste Knopf (Partition) gedrückt, startet ein Thread, welcher für alle eingelesenen Trajektorien der Main-Klasse die Partitionierungsroutine „Approximate Trajektory Partitioning“ (kurz „ATP“) ausführt. In Algorithmus 3 ist dieser Aufruf beschrieben. ATP selbst folgt dem Algorithmus 1 aus Kapitel 3 mit den beschriebenen Änderungen aus Kapitel 4.

Algorithmus 3: Aufruf der Partitionierung

```
for Trajectory TR : Main.trajectories do
    aktualisiere Fortschritt in Ausgabe;
    ArrayList<PositionPoint> CharPunkte = Traclus.atp(TR);
    for int i = 0; i < CharPunkte.size() - 1; i++ do
        LineSegment neueLinie = new LineSegment(TR.getNumber(), CharPunkte.get(i),
        CharPunkte.get(i + 1));
        Model.getInstance().getLineseg().add(neueLinie);
```

Die Routine befindet sich zusammen mit der Clusterfunktion in der Klasse Traclus. Traclus greift beim Start der Partitionierung oder des Clusterings auf die Eingabefelder der Benutzeroberfläche zu und übernimmt die angegebenen Parameter. Für jede Trajektorie und jedes Liniensegment wird zuerst der Fortschritt im Ausgabefeld des Menüs aktualisiert.

Bei der Partitionierung entstehen charakteristische Punkte, welche wiederum zu Linien zusammengefasst werden (siehe Algorithmus 3). Anschließend werden die Linien an die Model-Klasse übergeben (Algorithmus 3, letzte Zeile) und können von der Overlay-Klasse ausgelesen werden. Auf diese Weise kann jede einzelne neue Linie sofort gezeichnet werden und es werden nur Punkte miteinander verbunden, welche der selben Trajektorie angehören.

Ist die Partitionierung beendet, so wird der Cluster-Knopf aktiviert, da alle Liniensegmente nun zur Verfügung stehen. Dieser startet durch Drücken ebenfalls einen neuen Thread, welcher die Liniensegmente aus Model nach dem Algorithmus 2 aus Kapitel 3 clustert. Die nötigen Parameter werden bei der Initialisierung wieder direkt von der Benutzeroberfläche abgegriffen. Es ist wichtig, die Parameter beim Starten der Routinen auszulesen, da sich ansonsten Änderungen der Werte während der Berechnung auf die laufende Ergebnisserzeugung auswirken oder das Programm abstürzt. Da die Liniensegmente weiterhin in Model gespeichert sind, müssen sie während dem Clustering nicht neu an Model übergeben werden. Wird ein Liniensegment einem Cluster zugeordnet, weist das Overlay dem Liniensegment die entsprechende Farbe zu. Bei der nächsten Neuzeichnung wird es in der neuen Farbe gezeichnet. Somit sind alle Zuweisungen und Veränderungen in der Karte sofort sichtbar.
Immer wenn ein neues Cluster gefunden wird, erstellt die Clusterroutine eine neue Checkbox, mit welcher alle Liniensegmente des entsprechenden Clusters ein- oder ausgeblendet werden können (siehe Kapitel 5.4). Am Ende des Clusterings erfolgt die Filterung. Um diese darzustellen, werden die gefilterten Cluster an Model übergeben. Die Zeichenroutine verwendet anschließend diese gefilterten Daten anstatt der ursprünglichen Clusterergebnisse. Das automatische Neuzeichnen fällt dadurch ebenfalls weg und die Karte wird nur bei Interaktionen, wie Zoomen, Ein- und Ausblenden oder durch Verschieben neu gezeichnet. Die automatische Neuzeichnung ist nur während der Partitionierung oder des Clusterings aktiv.

Für das Clustering wird das Distanzmaß zur Berechnung der Nachbarn benötigt. Dieses befindet sich in der Klasse TraclusDistanceFunction. Die Parameter werden bei der Initialisierung der Distanzmaß-Klasse ebenfalls direkt aus der Nutzeroberfläche ausgelesen. Die zwei Distanzfunktionen, welche von der Partitionierung benötigt werden, sind separat und public in TraclusDistanceFunction definiert, damit diese einzeln berechnet werden können. In Kapitel 5.2.1 wurde auch der dritte Knopf zum Starten des Programmes vorgestellt. Dieser führt eine Routine in der Traclus-Klasse aus, welche das Partitionieren und das Clustern sequentiell abarbeitet und dabei wie die anderen Routinen auf die Parameter und Trajektorien der Benutzeroberfläche und der Main-Klasse zugreift. Der einzige Unterschied ist, dass hierbei kein Thread genutzt wird und nicht jedes Element an Model übergeben wird. Das bedeutet, die Karte wird nicht ständig aktualisiert. Am Ende werden die Cluster in Model gespeichert und gezeichnet. Auch die drei Steuerknöpfe, Reset, Pause und Fortsetzen, sind über ActionListener implementiert und sorgen dafür, dass der aktuelle Thread, in welchem das Clustering ausgeführt wird, beendet, unterbrochen oder fortgesetzt wird. Die Slider der Benutzeroberfläche sind durch ChangeListener realisiert, welche bei Veränderungen den Wert in der zugehörigen Textbox aktualisieren. Der Aufbau des Menüs der Nutzeroberfläche, sowie die Positionierung der einzelnen Elemente, ist durch mehrere JPanel umgesetzt.

6.2 Parallelisierung

Damit der vorliegende Ansatz interaktiv bleibt wird eine Parallelisierung mit Hilfe von Threads genutzt. Ohne eine Parallelisierung können Nutzer die Karte und das Menü nicht mehr bedienen, während das Programm Partitionen oder Cluster berechnet. Dadurch wäre die Analyse oder die Exploration der Daten erst am Ende der Berechnungen möglich. Aus diesem Grund ist ein gewisser Grad an Parallelität zwingend notwendig. In Abbildung 6.2 werden die parallelen Abläufe dieses Ansatzes in einem Sequenzdiagramm dargestellt.

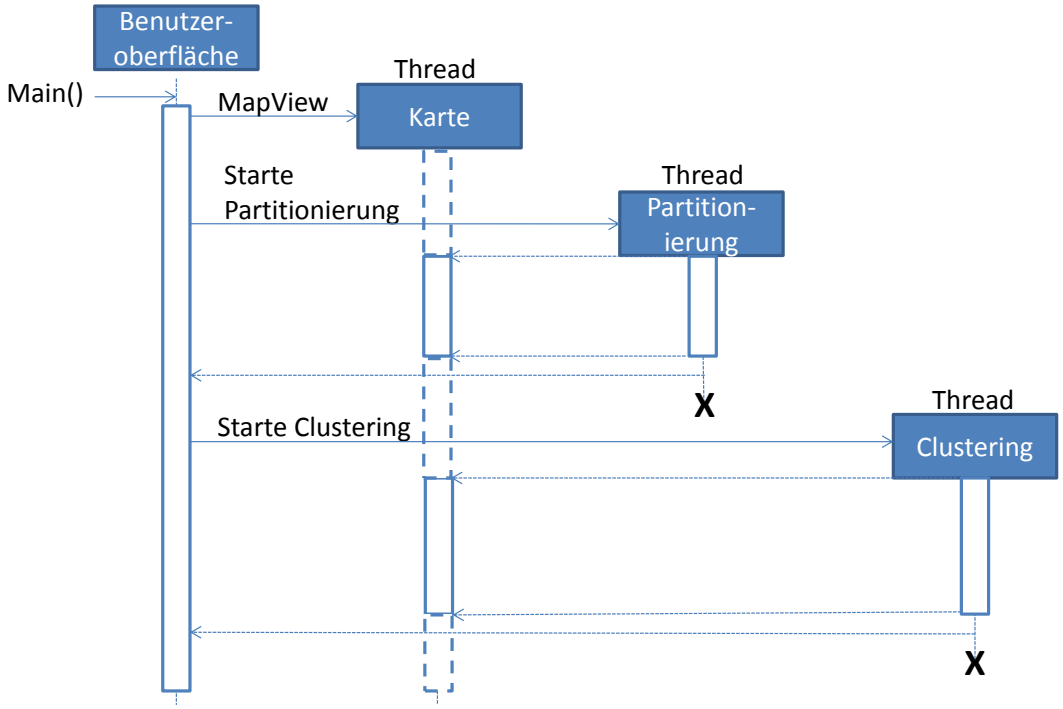


Abbildung 6.2: Sequenzdiagramm der parallelen Abläufe des Systems.

Die Benutzeroberfläche wird durch die Main-Klasse initialisiert und bildet den Hauptprozess des Programmes. Alle folgenden Berechnungen werden über sie gestartet. Beim Initialisieren der Benutzeroberfläche wird die Karte aufgerufen, welche in der Benutzeroberfläche eingebettet ist. Das Zeichnen auf der Karte wird in einem Thread ausgeführt, welcher, während der Partitionierung oder dem Clustering, alle 100ms Neuzeichnungen der Karte durchführt. Ist keine Berechnung aktiv, so wird die Karte nur dann aktualisiert, wenn Aktionen auf ihr ausgeführt werden. Auf diese Weise kann die Benutzeroberfläche ohne Unterbrechungen bedient werden und hat keine Verzögerung bei Eingaben. Wird eine Partitionierung oder ein Clustervorgang über die Konfigurationsoberfläche eingeleitet, starten diese ebenfalls in einem neuen Thread, um die weitere Nutzung der Karte und der Oberfläche nicht zu beeinflussen. Dabei wird auch

die automatische Neuzeichnung in ihrem Thread aktiviert. Ist eine Berechnung beendet, wird dies der Benutzeroberfläche gemeldet und weitere Optionen werden aktiviert. Beispielsweise der Clustering-Vorgang steht erst dann zur Verfügung, wenn eine Partitionierung in einem Thread abgeschlossen wurde. Die Threads der Partitionierung und des Clusterings werden nach einem Durchlauf der Methoden wieder beendet. Die Prozesse der Oberfläche und der Karte dagegen bleiben erhalten, bis das gesamte Programm beendet wird.

6.3 Indexierung

Um die Berechnungen des Programmes zu beschleunigen eignet sich grundsätzlich eine Indexierung. Durch eine Indexierung muss bei der Berechnung der nächsten Nachbarn eines Elementes eine viel kleinere Anzahl von anderen Elementen betrachtet werden. In [21] wird ebenfalls die Verwendung einer Indexstruktur vorgeschlagen, da die Laufzeit des Clusterings, ohne eine solche Struktur, quadratisch ist. Die Autoren von [21] schlagen zur Indexierung R-Bäume [14] vor. Solche Baumverfahren zur Indexierung von Daten gehen immer nach einem ähnlichen Schema vor. Die Daten werden in einer Baumstruktur so angeordnet, dass, abhängig von einem Element und den zugehörigen Parametern, nur noch ein wesentlich kleinerer Unterbaum der Struktur nach den Nachbarn des Elementes durchsucht werden muss. R-Bäume sind aber als Indexstruktur für Datenpunkte entwickelt worden und müssten erst auf Liniensegmente angepasst werden. Weiterhin arbeiten R-Bäume auf räumlichen Distanzmaßen im euklidischen Raum. In dem vorliegenden Ansatz wurde aber nur eine Metrik verwendet. Deshalb eignet sich ein VP-Baum (Vantage-Point Tree) [33] besser, da dieser damit umgehen kann. Ein VP-Baum geht nach einem ähnlichen Verfahren wie ein R-Baum vor und lässt sich auf Liniensegmente anwenden.

Die Laufzeit des entwickelten Ansatzes wird dadurch um ein Vielfaches verbessert und Versuche können schneller durchgeführt werden. Ein Problem dabei ist aber, dass der VP-Baum die Nachbarn anhand von Aussichtspunkten (Vantage-Points) findet und dabei nicht immer exakt alle Nachbarn zurückgibt. Die gewählte Implementierung des VP-Baumes ist zusätzlich randomisiert. Abhängig vom zufälligen Startpunkt der Baumerzeugung werden bei jedem Durchlauf andere Ergebnisse für die Nachbarn jedes Elementes ausgegeben. Dieses Verhalten ist nicht erstrebenswert für den hier diskutierten Ansatz, da Ergebnisse reproduzierbar sein sollten. Ansonsten wird es Nutzern erschwert, gute Parameter für ihre Berechnungen zu finden. Aus diesem Grund wurde die Indexierung wieder aus dem Programm entfernt. Festzuhalten ist an dieser Stelle aber, dass die Beschleunigung durch eine Indexstruktur in diesem Ansatz vorteilhaft ist. Eine nicht randomisierte Variante einer passenden Indexstruktur wäre demnach eine wertvolle Ergänzung für eine Implementierung dieses Ansatzes. Eventuell würde sich auch ein TP-Baum [32] als eine Indexstruktur für diesen Ansatz eignen, da dieser auf Grundlage von VP-Bäumen und anderen Indexbäumen entwickelt wurde und bessere Eigenschaften als diese besitzen soll, was aber hier nicht getestet wurde. Die im Rahmen dieser Arbeit durchgeführten Tests haben auf jeden Fall gezeigt, dass eine Indexstruktur erstrebenswert für eine Implementierung des zugehörigen Programmes ist. Es sollte jedoch darauf geachtet werden, dass keine randomisierten oder approximierenden Faktoren in der Implementierung einer solchen Struktur vorkommen.

7 Anwendungsfälle

Der entwickelte Ansatz kann für mehrere Anwendungsfälle genutzt werden. Um diese zu erklären werden zuerst zwei Datensätze vorgestellt, mit welchen die Tests für das implementierten Verfahren durchgeführt wurden. Anschließend wird anhand dieser Datensätze gezeigt, welche Analysen mit welchen Einstellungen möglich werden. Daraufhin werden mögliche Problemfälle betrachtet, die während der Tests gefunden wurden. Am Ende wird beispielhaft ein möglicher Ablauf einer Analyse mit dem entwickelten Programm geschildert.

7.1 Datensätze

Bei der Entwicklung des Programmes wurden zwei Datensätze verwendet. Dabei handelt es sich um den „Geolife Trajectories 1.3“ und den „North Atlantic Hurricane“ Datensatz. Beide Datensätze werden nun im Detail beschreiben, da diese in die Anwendungsfällen miteinbezogen werden.

7.1.1 Geolife Trajectories 1.3

Dieser Datensatz wurde während dem Microsoft Research Asia Geolife Projekt über fünf Jahre (April 2007 bis August 2012) zusammengetragen und enthält die Bewegungen von 182 Menschen [34]. Er besteht aus 17621 Trajektorien, welche alle aus mehreren Koordinaten-Punkten aufgebaut sind. Die Punkte enthalten einen Zeitstempel und werden durch Längen-, Breiten- und Höhengrade eindeutig beschrieben. Die Trajektorien von 73 Teilnehmern wurden zusätzlich noch mit der zugehörigen Transportmethode gelabelt, zum Beispiel Auto oder Fahrrad. Insgesamt beträgt die zurückgelegte Distanz der Bewegungsdaten 1292951 Kilometer, wobei die Distanz einzelner Trajektorien von unter 5 Km (36% der Daten) bis über 100 Km (5% der Daten) variiert. Die Zeit der Bewegungen beträgt insgesamt 50176 Stunden, die Zeit einzelner Trajektorien ist aber auch hier weit gefächert, von unter einer Stunde (58% der Daten) bis zu über 12 Stunden (5% der Daten). Die Daten wurden von verschiedenen GPS-Loggern und GPS-Chips in Mobiltelefonen aufgezeichnet. Die meisten der Trajektorien-Datenpunkte (91,5%) sind räumlich sehr nah, da die Punkte nur 1-5 Sekunden oder 5-10 Meter auseinander liegen [34]. Die Daten entstammen vielen verschiedenen Bewegungen und Bewegungstypen und sind breit gefächert, von China bis nach Europa und in die USA, wobei die meisten Daten aus China, aus dem Raum Beijing stammen. Es handelt sich hierbei um einen sehr detaillierten Datensatz, welcher in seinem Zentrum (Beijing) viele Trajektorien aufweist und weiter entfernt vom Zentrum eher weniger Trajektorien besitzt. Weitere Informationen zum Datensatz können in der Benutzerbeschreibung [34] gefunden werden.

7.1.2 North Atlantic Hurricane

Der hier verwendete Datensatz ist nur ein kleiner Teil eines größeren Datensatzes von Wirbelstürmen. Dieser wurde vom NOAA's National Climatic Data Center[1] im Verlauf des Projektes „IBTrACS" („International Best Track Archive for Climate Stewardship") aus vielen verschiedenen Daten, von meteorologischen Einrichtungen auf der ganzen Welt, über die Jahre 1848 bis 2014, zusammengetragen. Verwendet wird davon der Teil, welcher die Hurrikane im Nord-Atlantik in den Jahren 1851 bis 2013 zusammenfasst[2]. Dieser Ausschnitt enthält 1768 Trajektorien, mit je 10 bis 100 Datenpunkten. Während dem IBTrACS Projekt wurde für die Hurrikane aus allen verfügbaren Daten meteorologischer Einrichtungen der beste Pfad ermittelt, woraus die Datenpunkte entnommen wurden [19]. Die einzelnen Datenpunkte enthalten Längen- und Breitengrade und wurden mit einem Zeitstempel versehen. Auch weitere Informationen wie die Namen der Hurrikans, die Nummer, das Datum und meteorologische Daten wurden zu jedem Punkt festgehalten. Der Datensatz ist, im Gegensatz zum Geolife Datensatz, grob aufgelöst, da die einzelnen Punkte etwa 6 Stunden auseinanderliegen. Die Arbeit [19] beschäftigt sich mit der Erzeugung der Daten und enthält weitere Informationen zu den Daten selbst.

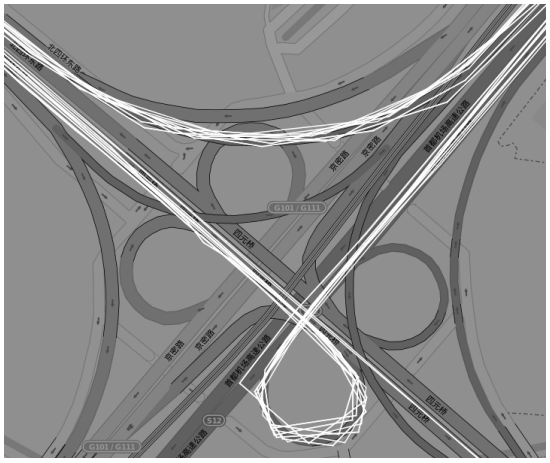
7.2 Partitionierung

Die Partitionierung an sich stellt schon eine mögliche Anwendung des hier beschriebenen Ansatzes dar. Durch die Reduzierung der Bewegungen auf wenige Linien können visuelle Analysen auch ohne ein Clustering durchgeführt werden. Über den Schieberegler lässt sich dafür zuerst der Grad der Partitionierung einstellen und über den entsprechenden Knopf wird die Partitionierung gestartet. In Abbildung 7.1 sieht man die Partitionierung eines Autobahnkreuzes in Beijing aus dem Geolife-Datensatz. In (a) ist der Grad der Partitionierung sehr fein, weshalb die Linien den Straßen sehr gut folgen. In (b) dagegen ist der Grad grob eingestellt. Vergleicht man die Abbildungen, fällt auf, dass die geraden Strecken sich kaum unterscheiden und nur in den Kurven Unterschiede zu erkennen sind.

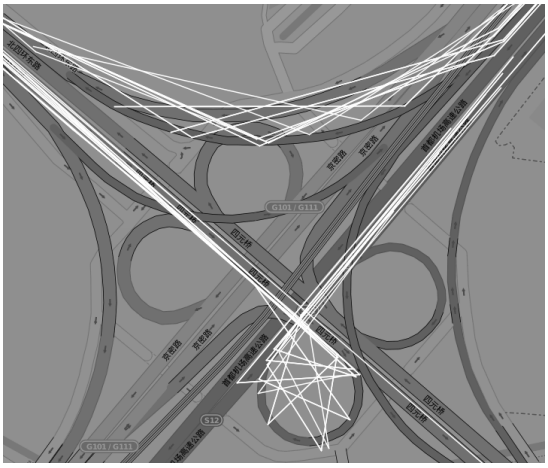
Betrachtet man das vorgestellte Verfahren zur Partitionierung macht das Sinn. Wird nämlich eine gerade Strecke durch zwei Punkte definiert, so ist die resultierende Linie gleich zu einer Linie, deren Strecke ebenfalls gerade verläuft, aber durch mehrere Punkte beschrieben wird. In Kurven ist dies natürlich anders, denn umso mehr Punkte verbunden werden, desto genauer wird die Kurvenbewegung durch Linien beschrieben. Durch eine gröbere Einstellung der Partitionierung werden weniger charakteristische Punkte gefunden und Kurven dadurch schlechter dargestellt. Die Anzahl an Liniensegmenten variiert dabei stark. Bei der groben Partitionierung erhält man in diesem Beispiel nur rund 30% so viele Liniensegmente, wie bei der feinen Partitionierung, was sich natürlich auf die Laufzeit des Programmes auswirkt. Die Hauptverläufe der Bewegungen sind hier aber trotzdem noch gut zu erkennen. Durch diese weniger komplexe Darstellung der Verläufe können sie einfacher überblickt und visuell analysiert werden. Bei der Partitionierung ist es wichtig, einen Kompromiss zwischen Genauigkeit und Anzahl der Elemente zu finden. Nutzer sollten also abhängig von Daten und ihrer In-

[1] http://www.ncdc.noaa.gov/ibtracs/index.php?name=wmo-data

[2] Quelle des Datensatzes: `ftp://eclipse.ncdc.noaa.gov/pub/ibtracs/v03r06/wmo/csv/basin` - Datei: Basin.NA.ibtracs_wmo.v03r06.csv



(a) Feine Partitionierung: Die Linien folgen den Fahrbahnen sehr genau.



(b) Grobe Partitionierung: Die Linien folgen den Fahrbahnen eher grob.

Abbildung 7.1: Partitionierungen eines Autobahnkreuzes in Beijing. Die Linien wurden für diese Beispiele ohne Transparenz gezeichnet.

(a) Schritt 1

(b) Schritt 2

(c) Schritt 3

Abbildung 7.2: Schritte einer Partitionierung der ersten 1000 Trajektorien des Geolife-Datensatzes.

tuition den Grad der Partitionierung wählen. Soll zum Beispiel mit einem kurvenreichen Datensatz gearbeitet werden, eignet sich eine feinere Partitionierung besser, als eine grobe. Um die ideale Partitionierung zu finden, kann der Nutzer die Karte auf den Bereich zentrieren, der ihn besonders interessiert und mehrere Partitionierungen durchführen, bis er eine passende Einstellung gefunden hat. „Passend“ bedeutet dabei zum Einen, dass sie genau genug ist, um alle Verläufe zu erkennen, zum Anderen aber, dass sie nicht zu genau ist und sehr viele Linien entstehen. Die Berechnung der Liniensegmente wird dem Nutzer kontinuierlich dargestellt, so dass er schon während der Berechnung prüfen kann, ob die Partitionierung seinen Ansprüchen genügt. In Abbildung 7.2 sind drei Zeitpunkte eines Partitionierungsvorgangs dargestellt.

Während der Partitionierung sind die Daten also schon explorierbar und man kann dadurch nach besonderen Stellen in den Daten suchen. Möchte man beispielsweise nur herausfinden, welche Wege Radfahrer für ihre Touren verwenden, so bekommt man an dieser Stelle schon die gewünschten Ergebnisse. Sollen allerdings komplexere Analysen mit den Daten durch-

geführt werden, ist es einfacher, wenn ein Clustering der Partitionierung folgt. Die Partitionierung ermöglicht dabei das Auffinden gemeinsamer Teilstrecken beim Clustering. Die Anwendung einer solchen Partitionierung kann auch in anderen Programmen sinnvoll sein, zum Beispiel für die Glättung räumlicher Daten mit unerwünschten Kurven. Ist die Partitionierung abgeschlossen, erhält man die Liniensegmente, welche bereits visuell analysiert werden können, oder sie dienen als Grundlage für ein folgendes Clustering.

7.3 Analysen

Der vorliegende Ansatz ist auf Clusteranalysen ausgelegt. Ein wesentlicher Unterschied zu bisherigen Arbeiten ist, dass die gefundenen Cluster je nach Einstellung variieren. Die meisten anderen Verfahren führen ein statisches Clustering durch und die Ergebnisse können anschließend nach verschiedenen Aspekten exploriert werden. Die Ergebnisse des Clusterings selbst sind dabei, wenn überhaupt, minimal verschieden, da die Werte für die Algorithmen schon bei der Implementierung festgelegt werden. Die möglichen Analysen sind bei der vorliegenden Arbeit also vom Clustering und damit von den Parametern direkt abhängig. Zuerst muss geklärt werden, welchen Einfluss die Parameter auf die Analysemöglichkeiten haben und erst danach kann darauf eingegangen werden, welche Informationen die einzelnen Analysen gewähren.

7.3.1 Einfluss der Parameter auf die Analysen

Wesentlichen Einfluss auf die Ergebnisse haben in erster Linie die Parameter ε und $MinLns$. Wird ein sehr kleiner Epsilonwert, oder ein sehr großer Wert für $MinLns$ genutzt, so findet der Algorithmus viele kleine Cluster. Das kann eine bessere Analysen von zentrierten Daten ermöglichen, bei welchen die Gefahr besteht, dass ein großes Cluster im Zentrum und ein paar kleine Cluster am Rand der Daten entstehen. Solch ein großes Cluster wird durch die richtige Wahl der Parameter in kleinere Cluster aufgeteilt, die detailliertere Analysen ermöglichen. Dabei bleiben dann aber auch mehr ungeclusterte Elemente, vor allem am Rand der Daten, zurück. Wird dagegen ein sehr großer Epsilonwert, oder ein sehr kleiner Wert für $MinLns$ verwendet, bilden sich wenige große Cluster. Dadurch fällt die Analyse von Daten leichter, welche sehr zerstreut angeordnet sind und bei denen die Gefahr besteht, dass viele kleine Cluster entstehen, die sich nicht mehr überblicken lassen. Durch die Hilfe der Parameter kann der Algorithmus folglich auf verschiedene Datensätze angepasst werden und liefert eine Grundlage für die Analyse dieser Daten. Durch den Parameter der Partitionierung können auch komplexe Datensätze auf die wichtigsten Linien reduziert werden, wie bereits gezeigt wurde. Dadurch werden Analysen großer Datensätze erleichtert.

Die wichtigsten Parameter für die Ergebnisse der Berechnungen und damit auch die Analysen, sind die Parameter des Distanzmaßes. Sie beeinflussen die möglichen Analysen der Ergebnisse maßgeblich. Wird hauptsächlich die Winkeldistanz verwendet, so sind die Cluster gerichtet. Dadurch lassen sich zum Beispiel Richtungsanalysen durchführen. Benutzt man hauptsächlich die senkrechte Distanz, erhält man zum Großteil Cluster, die aus Linien der selben Trajektorien bestehen. Daraus ergibt sich die Möglichkeit, einzelne prägnante Trajekto-

rien zu analysieren. Im Gegensatz dazu, wenn hauptsächlich die parallele Distanz verwendet wird, finden sich Cluster, die hauptsächlich aus Linien verschiedener Trajektorien bestehen. Auf diese Weise können häufig verwendete Wegpunkte gefunden und analysiert werden. Verwendet man die senkrechte und parallele Distanz gleichzeitig, so lassen sich räumlich dichte Cluster finden. Damit lassen sich Ortsanalysen vornehmen. Durch eine Kombination aller drei Distanzmaße werden Streckenanalysen ermöglicht, da Cluster gefunden werden, welche räumlich dicht sind und eine bestimmte Richtung verfolgen. Orts-, Strecken- und Richtungsanalysen werden nun im Detail behandelt.

7.3.2 Ortsanalyse

Ortsanalysen können durch die Kombination der senkrechten und der parallele Distanz durchgeführt werden. Die dabei gefundenen Cluster beschreiben Stellen, an denen viele Bewegungen in einem kleinen Raum stattgefunden haben. Durch die Analyse dieser Orte können viele wertvolle Informationen gewonnen. Dies soll beispielhaft an einem Tierbewegungsdatensatz veranschaulicht werden. Durch die geclusterten Daten lassen sich Orte entdecken, an welchen sich häufig viele Tiere aufhalten. Durch eine Analyse dieser Plätze können dann Informationen gewonnen werden, welche Aufschluss darüber geben, ob diese Plätze gefährdet sind. Die zu geringe Distanz zu einer Autobahn kann zum Beispiel so eine Information sein, welche sich durch die Analyse gewinnen lässt. Dadurch könnte veranlasst werden, die betroffenen Plätze einzuzäunen, um den Schutz der Tiere zu gewährleisten. Bezieht man zusätzlich die Uhrzeit der Daten ein, lassen sich auch Schlafplätze oder Jagdreviere von Tiere ermitteln. Dies würde die Erforschung der Gewohnheiten und der Verhalten der Tiere ermöglichen. Ein anderes Beispiel betrifft Messen. Analysiert man hier die Orts-Cluster, kann man folgern, für welche Messestände sich Besucher am meisten interessiert haben. Der Analyst kann auch herausfinden, wo besonders viel Bewegung stattgefunden hat und so die Platzierung eines Messestandes an einem solchen Ort vorschlagen. Dies würde die Wahrscheinlichkeit erhöhen, dass der Stand bei der nächsten Messe oft besucht wird. Interessante Orte kann man so in fast allen Datensätzen finden. Um wertvolle Informationen daraus zu gewinnen, müssen diese Orte aber visuell von einem Fachmann analysiert werden.

7.3.3 Richtungsanalyse

Wird nur die Winkeldistanz zum Clustern von Daten verwendet, so lassen sich Richtungsanalysen durchführen. Die entstehenden Cluster bestehen dabei aus Liniensegmenten, welche in die selbe Richtung gehen, unabhängig von ihrer räumlichen Nähe. Durch eine Analyse solcher Richtungsströme lassen sich wichtige Erkenntnisse über verschiedene Bewegungsströmungen gewinnen. Am besten eignet sich diese Form der Analyse aber für Bewegungen, welche strömungsartig in großen Gebieten verlaufen, wie zum Beispiel Wirbelstürme, oder Meeresströmungen. Es können Erkenntnisse gewonnen werden, die zum Verständnis oder zur Prävention, gegen die Auswirkungen solcher Strömungen, beitragen können. Kennt man beispielsweise die typischen Richtungsverläufe von Hurrikanen, so kann man bei einem neuen Hurrikan Vorhersagen treffen, um frühzeitige Warnungen rauszugeben. Außerdem können in betroffenen Gebieten Schutzbefestigungen errichtet werden, oder Evakuierungen eingeleitet werden. Ein anderes Beispiel wäre zum Beispiel die Ausbreitung von Öl, oder anderen schädlichen

Substanzen im Meer. Durch die Analyse von Meeresströmungen kann man typische Ausbreitungsmuster erkennen und vorhersagen, welche Gebiete die Substanzen erreichen werden. Abbildung 7.3 stellt ein beispielhaftes Clustering des hier vorgestellten Programmes mit der Winkeldistanz dar. Dafür wurde der North Atlantic Hurricane Datensatz mit einer groben Partitionierung verwendet. Ähnlich wie bei Lee et al. in [21] entstehen sieben Cluster, welche die verschiedenen Richtungsverläufe der Hurrikane darstellen.

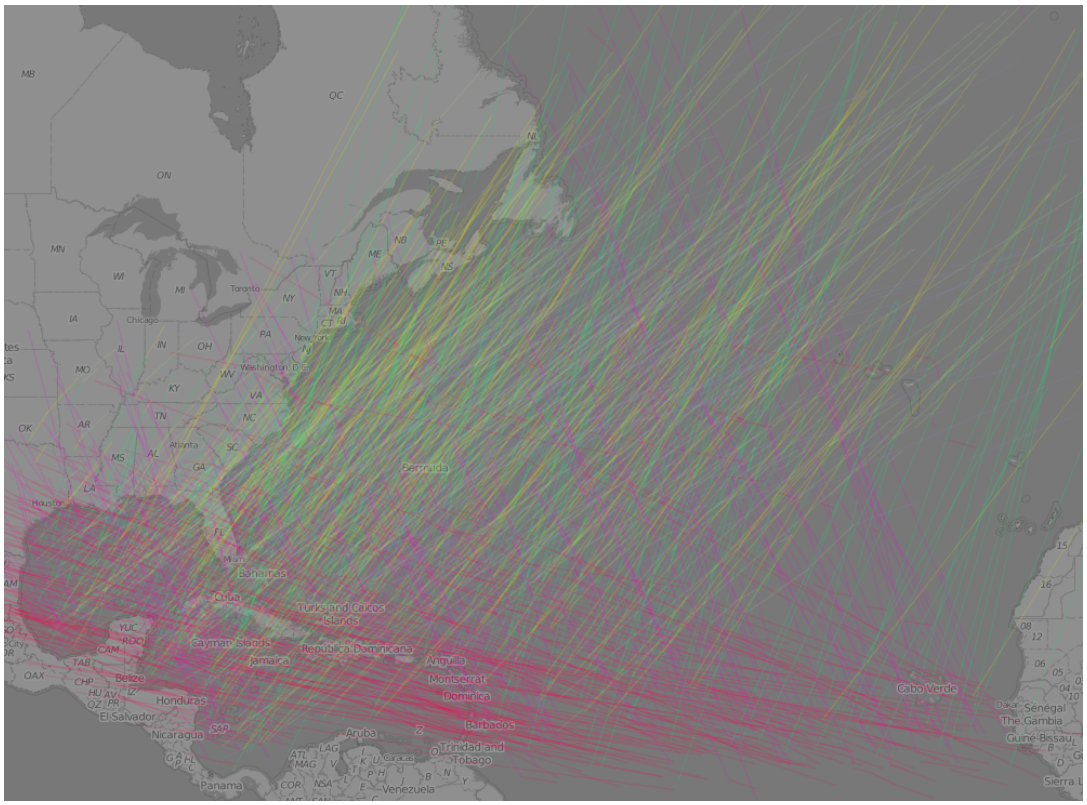


Abbildung 7.3: Clustering mit grober Partitionierung anhand der Winkeldistanz, wobei ungeclusterte Linien und Rauschen ausgeblendet sind.

Die Richtungsverläufe sind klar zu unterscheiden, die räumliche Distanz der Linien ist dabei egal, wie man in der Abbildung erkennen kann. Ein Analyst kann nun die Ergebnisse auswerten und sehen, dass viele der Strömungen aus dem Südosten kommen. Das Gebiet im Südosten könnte, aufgrund dieser Ergebnisse, stärker überwacht werden, um Hurrikane frühzeitig zu entdecken. Natürlich können Analysten auch weitere wichtige Erkenntnisse aus Richtungsanalysen gewinnen, zum Beispiel wie sich die Richtung von Strömungen verändern, wenn sie auf die Küste treffen. Zur Erforschung von Hurrikanverhaltensmuster kann man die häufigsten Richtungsverläufe, anhand der einzelnen Cluster, zusammenfassen und festhalten. Interessiert man sich für eine bestimmte Richtungsbewegung, können alle Cluster bis auf eines ausgeblendet werden, um dieses genauer zu analysieren. Ohne eine Partitionierung wäre es schwieriger Richtungsanalysen durchzuführen, da die Trajektorien im Ganzen geclustert werden. In diesem Ansatz werden nur ähnliche Teile der verschiedenen Trajektorien geclustert, weshalb auch die wirklichen Richtungsverläufe entdeckt werden können, statt nur richtungsähnliche Trajektorien.

7.3.4 Streckenanalysen

Kombiniert man alle drei Distanzmaße, erhält man Cluster, deren Liniensegmente in die selbe Richtung gehen und nicht soweit voneinander entfernt sind, wie bei der Richtungsanalyse. Es handelt sich also um Strecken und nicht um grobe Richtungsströmungen, da die Verläufe auf einem viel engeren Raum sind. Die Clusterelemente haben demnach nicht nur eine gemeinsame Richtung, sondern auch ein gemeinsames Gebiet. So lassen sich häufig verwendete Routen in den Daten finden. Die Analyse solcher Strecken kann viele Erkenntnisse liefern. Zum Beispiel lassen sich stark genutzte Abschnitte von Infrastrukturen in Verkehrsdatensätzen finden. Deren Analyse und Interpretation kann Erkenntnisse liefern, welche zur Entlastung oder zum Ausbau bestimmter Teilabschnitte beitragen. Auch Vorhersagen lassen sich dadurch treffen. Findet man typische Strecken von Hurrikanen, so kann man bei einem weiteren Hurrikan ähnlicher Ausmaße und Position darauf schließen, wie er sich fortbewegen wird. Dadurch lassen sich frühzeitig Präventionsmaßnahmen einleiten, um betroffene Regionen zu schützen. Durch das Anwenden des Clusterings auf den North Atlantic Hurricane Datensatz mit allen Distanzmaßen und einer groben Partitionierung erhält man eine Visualisierung wie in Abbildung 7.4.

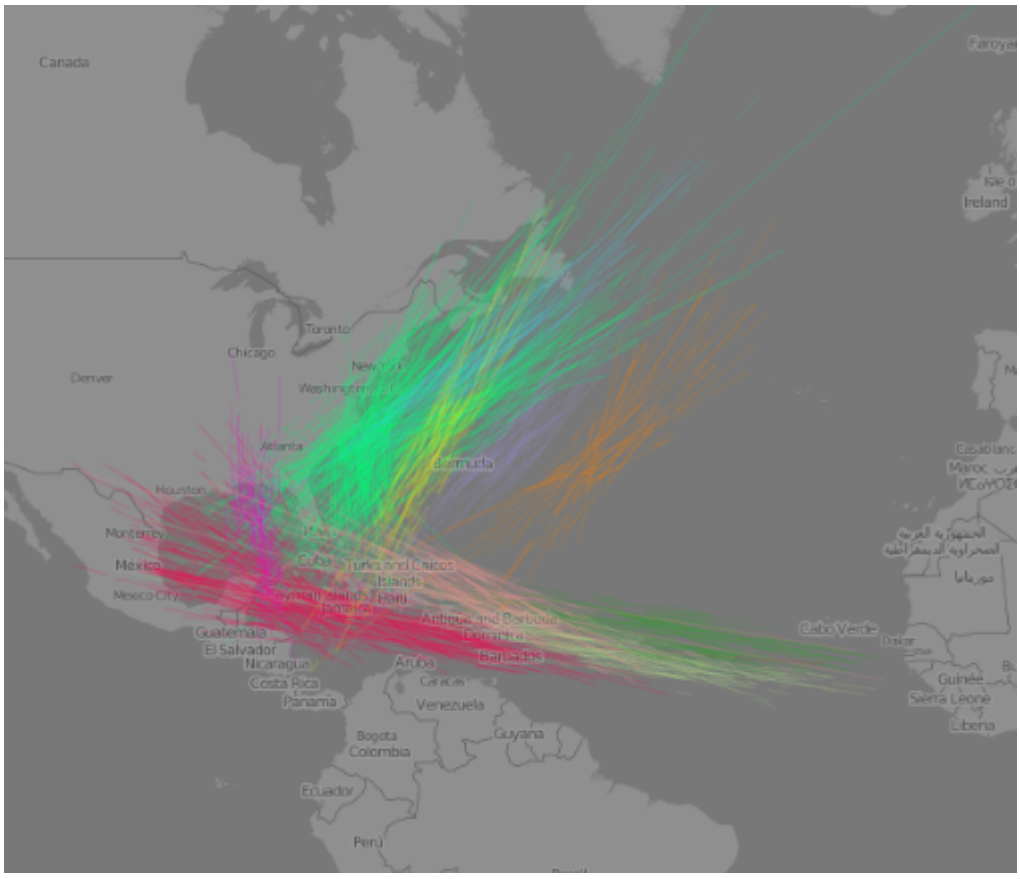


Abbildung 7.4: Clustering mit grober Partitionierung anhand aller Distanzen, wobei ungeclusterte Linien und Rauschen ausgeblendet sind.

Als Erstes fällt auf, dass die Linien eines einzelnen Clusters auf einem viel engeren Raum lie-

gen, als beim Clustering mit der Winkeldistanz alleine. Die Linien verlaufen in die selbe Richtung und erzeugen dadurch eine Strecke. Eine Analyse kann nun verschiedene Erkenntnisse liefern. Weiß man beispielsweise, welche Charakteristika die Hurrikane des orangen Clusters gemeinsam haben, kann man bei einem neuen Hurrikan der gleichen Ausmaße vorhersagen, dass er keine Gefahr für die Küste darstellt. Auch hier bietet die Partitionierung einen Vorteil. In obigen Beispiel kann man erkennen, dass viele Hurrikane eine Kurve beschreiben und vom rosafarbenen Cluster in das neongrüne Cluster übergehen. Der Übergang findet genau in dieser Kurve statt und man kann an dieser Stelle den Grund für das kurvenartige Verhalten durch eine geeignete Analyse feststellen. Daraus können Informationen für das Verhalten von Hurrikanen in diesem Gebiet gewonnen werden. Würden man stattdessen ganze Trajektorien clustern, müsste man erst selbstständig feststellen, wo dieser Streckenwechsel stattfindet. Die in diesem Beispiel gefundenen Strecken der Cluster sind ähnlich zu den Repräsentanten einzelner Cluster, welche Lee et al. in [21] mit dem gleichen Datensatz gefunden haben.

7.4 Problemfälle

Während einigen Tests mit dem implementierten Programm wurden ein Paar Fälle gefunden, in welchen dieses nicht optimal funktioniert. Während des Clusterings mit dem Geolife Datensatz wurde festgestellt, dass im Zentrum des Datensatzes, insbesondere in Beijing, viele Cluster gefunden werden. Entfernt man sich weiter vom Zentrum, werden aber zunehmend weniger Cluster gefunden. Die Ursache lässt sich darauf zurückführen, dass die räumliche Dichte der Trajektorien immer weiter abnimmt, je weiter man sich vom Zentrum entfernt. Im weiteren Umfeld von Beijing findet man also kaum Cluster, obwohl die Trajektorien an diesen Stellen ebenfalls wichtige Informationen enthalten könnten. Das liegt daran, dass am Anfang des Clusteringalgorithmus einmal feste Parameter für $MinLns$ und ε gewählt werden, welche für die weiter entfernten Linien zu groß, beziehungsweise zu klein sind. Werden kleinere Werte für $MinLns$ oder größere Werte für ε gewählt, so lassen sich zwar Cluster im weiteren Umfeld von Beijing finden, das Zentrum wird aber, aufgrund der toleranteren Parameter, zu einem einzelnen großen Cluster. Dieses lässt sich kaum überblicken und liefert dementsprechend auch nicht mehr Informationen, als ohne ein Clustering. Dieses Problem entsteht dadurch, dass der implementierte Clusteralgorithmus auf DBSCAN [8] basiert. DBSCAN weist genau das gleiche Problem auf, da er ebenfalls mit einmal festgelegten Parametern arbeitet. Der hier beschriebene Ansatz kann folglich nicht mit Datensätzen umgehen, in welchen die räumliche Dichte der Trajektorien in verschiedenen Teilgebieten stark variiert.

Ein weiteres Problem wurde beim Einstellen des Parameters der Partitionierung über den Schieberegler festgestellt. Der Wertebereich des Reglers ist fest vordefiniert. Will man nun mit zwei verschiedenen Datensätzen arbeiten, welche stark in der Auflösung der Trajektorien variieren, so kann es sein, dass dieser Wertebereich für den einen Datensatz geeignet ist, für den anderen aber nicht. Das bedeutet, die möglichen Parametereinstellungen machen kaum einen Unterschied was die Anzahl der entstehenden Elemente betrifft. Der Wertebereich müsste manuell im Programmcode verändert werden. Der Standard-Parameter 1.0 funktioniert jedoch bei jedem Datensatz ausreichend gut.

7.5 Beispiel eines konkreten Ablaufs eines Szenarios

Ein Schifffahrtsunternehmen möchte die Sicherheit ihrer Schiffe im Nordatlantik bei Hurrikanen erhöhen, ohne dabei den laufenden Betrieb stark zu reduzieren. Dazu sollen Sperrzonen und Gefahrenzonen festgelegt werden, welche bei einem drohenden Hurrikan zu meiden sind. Die Ergebnisse sollen so bald wie möglich zur Verfügung stehen, denn ein weiterer Hurrikan wurde bereits angesagt. Darum wird anfangs der passende Datensatz ausgewählt, der North Atlantic Hurricane Datensatz in diesem Fall.

Als Erstes wird die Partitionierung mit dem „normalen“ Parameter (1.0) ausgeführt. Es entstehen ungefähr 30000 Liniensegmente. Ein Analyst exploriert die partitionierten Linien und stellt fest, dass sie sehr genau sind und auch weniger Linien ausreichen. Er führt also eine erneute Partitionierung mit einem kleineren Wert durch. Nun sind nur noch etwa 18000 Linien enthalten. Der Analyst exploriert diese erneut und erkennt, dass die Unterschiede zur vorherigen Partitionierung visuell kaum feststellbar sind, jedoch die Anzahl der Elemente um 40% reduziert wurde. Er ist zufrieden mit dem Ergebnis der Partitionierung und geht dazu über, die Parameter für das Clustering auszuwählen. Zuerst überlegt der Analyst, welche Distanzmaße er verwenden soll. Eine Streckenanalyse scheint ihm am sinnvollsten, da er die typischen Wege der Hurrikane herausfinden möchte. Da diese breit gefächert sein können, wählt er kleinere Gewichtungen für die senkrechte und parallele Distanz und eine große Gewichtung für die Winkeldistanz über die Schieberegler. Für Epsilon wählt er zum Testen erstmal 1.0. Er schätzt ab, dass bei 18000 Liniensegmenten mindestens ein dreistelliger Wert für *MinLns* genutzt werden sollte und wählt dementsprechend erst einmal 100. Für den Cluster-Filter wählt er den Wert 0, da er nicht möchte, dass eventuell Informationen übersehen werden, die Schiffe gefährden könnten.

Während des Clusterings merkt der Analyst, dass langsam alle Liniensegmente in das erste gefundene Cluster übernommen werden. Er stoppt also den Prozess und wählt ein kleineres Epsilon von 0.1, danach startet er das Clustering erneut. Diesmal sieht der Anwender, dass zwar mehrere Cluster gefunden werden, aber einige davon sehr klein sind und nur etwa 100 Liniensegmente beinhalten. Bevor der Algorithmus zu Ende ist, bricht er die Berechnung ab und wählt einen neuen Wert von 500 für *MinLns*. Der Analyst startet wieder das Clustering und sieht während dem Prozess, dass sich mehrere Cluster bilden, welche ausreichend groß sind. Er analysiert die ersten Cluster, noch während das Clustering im Gange ist. Nachdem der Algorithmus terminiert, blendet der Analyst alle anderen Cluster, bis auf eines aus, um sich dessen Verlauf und Position genau anzuschauen. Dies tut er für alle Cluster und legt am Ende die Strecken der einzelnen Cluster als Sperrgebiet fest, da hier besonders häufig Hurrikane entlang ziehen. Anschließend blendet er nur die ungecusterten Linien und Rauschen ein und legt diese Gebiete als Gefahrenzonen fest, in welchen sich Schiffe während einer Hurrikan-Meldung zwar aufhalten können, aber vorsichtig verkehren sollten, da eine kleine Möglichkeit besteht, dass doch ein Hurrikan diese Gebiete passiert, oder sie zu Nahe an Sperrgebiete kommen. Der Analyst führt noch ein weiteres Clustering durch, in welchem er die Winkeldistanz auf 0 setzt, also eine zusätzliche Ortsanalyse.

Die Cluster die dabei gefunden werden, nimmt er ebenfalls als Sperrgebiete auf, welche

bei einer Hurrikan-Meldung auf jeden Fall zu meiden sind. Die Zonen können noch weiter analysiert werden, um sichere Routen durch das Meer zu finden, bei denen die Wahrscheinlichkeit einer Hurrikan-Begegnung sehr gering ist. Anschließend können sie an die Schiffe weitergegeben werden, damit diese, während eines Hurrikans, trotzdem sicher verkehren können.

8 Diskussion

An dieser Stelle ist es wichtig, zu diskutieren, welche Vorteile und Nachteile der vorgestellte Ansatz mit sich bringt. Die Zusammenfassung der Vor- und Nachteile beschreibt gleichzeitig die Freiheiten und Limitationen des Ansatzes.

8.1 Vorteile

Durch die Verknüpfung von automatisierten Verfahren mit Visualisierung können die Vorteile beider Methoden genutzt werden. Zum einen werden automatisch Cluster aus den Daten extrahiert, andererseits können diese auch, durch die Visualisierung, von Menschen analysiert werden, welche ihre kognitiven Fähigkeiten mit einbringen. Dadurch lassen sich neue Erkenntnisse gewinnen, welche bei rein automatischen oder rein visuellen Ansätzen nicht gefunden werden können. Ohne den automatischen Teil der Datenverarbeitung würde es einem Analysten viel schwerer fallen, seine Aufmerksamkeit bei einer Analyse in die richtige Richtung zu lenken, da sich große Datensätze ohne Anhaltspunkte schwer überblicken lassen. Weiterhin bietet der entwickelte Ansatz große Freiheitsgrade, was die Interaktion zwischen Nutzer und Programm angeht. Nutzer können zu jeder Zeit in den Prozess eingreifen und abhängig von Zwischen- und Ergebnisvisualisierungen Parameter verändern, um die Prozesse in eine gewünschte Richtung zu lenken. Das bringt den Vorteil mit sich, dass dieser Ansatz auf viele verschiedene Datensätze und Analysen optimiert werden kann und nicht auf spezifische Daten und Analysen beschränkt ist. Über die Partitionierung kann außerdem gesteuert werden, wie viele Elemente zu clustern sind, wodurch sich die Dauer und die Genauigkeit eines Clustering-Durchlaufs regulieren lässt. Die Partitionierung senkt ebenfalls die Auswirkungen von Rauschen und Messfehlern in Trajektorien auf die Ergebnisse, da diese einzeln verarbeitet werden und nicht als Teil der ganzen Trajektorie. Die großen Freiheitsgrade, beim Clustering von Trajektorien, machen diesen Ansatz aus und binden den Analysten stark in den gesamten Prozess ein und nicht nur in die Exploration der Ergebnisse. Durch weitere visuelle und interaktive Komponenten, wie die Funktionen zum Pausieren des Clusterings, oder zum Ausblenden von Clustern und Rauschen, werden Nutzer bei der Analyse und beim Verstehen der Ergebnisse unterstützt. So kann ein Überblick über große Datenmengen geschaffen werden. Zuletzt ist die ständige Aktualisierung der Visualisierung der Cluster und Linien, während dem Prozess, ein großer Vorteil. Sie ermöglicht die Analyse der Cluster und Linien schon während der Berechnung selbst und gibt somit die Option, das Clustering abzubrechen und neuzustarten, falls die Parameter falsch gewählt wurden. Dies hat sich in mehreren Tests als wichtig erwiesen, da ein Durchlauf des Algorithmus einige Zeit in Anspruch nimmt, die dadurch eingespart werden kann, falls festgestellt wird, dass die Teilergebnisse eine ungewollte Tendenz aufweisen.

8.2 Nachteile

Da einige Komponenten, wie etwa die Partitionierung oder die Cluster-Routine, aus anderen Arbeiten aufgegriffen und angepasst wurden, ergeben sich bei dem hier entwickelten Ansatz die selben Probleme, die diese Komponenten mit sich führen. Wird der Parameter für die Partitionierung zu klein gewählt, können Informationen verloren gehen, die in der Analyse eventuell neue Erkenntnisse geliefert hätten. Ein Nachteil des Clusteralgorithmus ist, dass er sich nicht für Datensätze eignet, welche eine stark unterschiedliche räumliche Dichte der Daten in verschiedenen Regionen aufweisen. Deshalb eignet sich der vorgeschlagene Ansatz nicht für alle Datensätze. Des Weiteren ist die hohe Laufzeit des Ansatzes ein Nachteil. Durch die fehlende Indexstruktur ist der Rechenaufwand für das Clustering, bei großen Datenmengen, sehr hoch, wie schon in [21] erwähnt wurde. Ein Nachteil besteht auch in der großen Zahl der Freiheitsgrade, welcher für die entsprechenden Vorteile in Kauf genommen wird. Da die Parameter alle frei einstellbar sind, passiert es oft, dass unpassende Parameter gewählt werden. Das führt dazu, dass das Programm mehrere Male ausgeführt werden muss, bis brauchbare Ergebnisse entstehen. Wenn ein neuer Datensatz benutzt werden soll, müssen demnach mehrere Durchläufe des Programms mit verschiedenen Parametern durchgeführt werden, bis gute Werte gefunden werden. Es kann sein, dass neue Nutzer mit der Vielzahl an Einstellungen überfordert sind. Der Ansatz eignet sich also für Nutzer, die schon einige Erfahrungen im Bereich Trajektorien-Clustering und im Bereich der visuellen Analyse gesammelt haben. Auch die Visualisierung an sich ist noch nicht optimal und liefert nicht alle Informationen, die Nutzer für eine Analyse brauchen könnten, wie zum Beispiel Informationen über die Richtung der einzelnen Liniensegmente.

9 Zusammenfassung

In dieser Arbeit wurde ein interaktives System vorgestellt, welches Trajektorien-Clustering und visuelle Analyse eng miteinander verknüpft und dabei eine Partitionierung verwendet, um den Prozess zu optimieren. Analysten können das System nutzen, um Datensätze nach ihren Anforderungen zu partitionieren und zu gruppieren. Währenddessen können sie die Ergebnisse explorieren und auswerten. Dabei hat der Nutzer immer die Freiheit, alle Vorgänge über Schnittstellen und Mechanismen zu steuern und wird durch interaktive Funktionen bei der Analyse unterstützt. Als Eingabedaten sind alle Bewegungsdatensätze zulässig, welche eindeutige Koordinaten ausweisen können, wobei der Ansatz sich auf Datensätze fokussiert, bei welchen die räumliche Dichte der Daten nicht stark variiert. Im Vergleich zu anderen aktuellen Verfahren der Analyse von Trajektorien, wurde daran gearbeitet, Nutzer interaktiv stärker in den Gesamtprozess miteinzubeziehen und deren Verständnis der automatischen Datenverarbeitung zu verbessern. Es wurde gezeigt, dass mehrere Analysen mit dem vorgestellten Ansatz möglich sind, welche von Nutzern mit Erfahrung im Gebiet der Trajektorien-Analyse durchgeführt werden können. Dies ist bedingt durch die Vielzahl an Konfigurationsmöglichkeiten des Systems. Das Nutzerinterface ist einfach zu überblicken und bietet Anwendern einige Einstellungsmöglichkeiten, um ihre Analyse zu lenken. Im Vergleich zu ähnlichen Trajektorien-Cluster-Algorithmen liefert der hier vorgestellte Ansatz vergleichbare Ergebnisse und auch Elemente der Visualisierung sind vergleichbar mit denen bestehender Systemen der visuellen Analyse. Insgesamt stellt der entwickelte Ansatz eine wichtige Ergänzung zu bisherigen Verfahren für Bewegungsdaten dar, welche automatische Datenverarbeitung, Visualisierung und Interaktion noch nicht so fest miteinander vereinen. Das implementierte System ist auf ein breites Spektrum von Bewegungsdatensätzen anwendbar, aus welchen Analysten potentiell wichtige Informationen extrahieren können. Das stärkere Einbinden von menschlichem Wissen in den Analyseprozess von Trajektorien ermöglicht die Gewinnung und Interpretation von datenbezogenen Informationen. Der Prozess an sich wird für alle Anwender viel transparenter, als bei Verfahren, welche den Nutzer nur die Ergebnisse explorieren lassen, ohne dabei Aufschluss über die Vorgänge der Ergebnisfindung zu liefern. Das ermöglicht Nutzer eine einfachere Orientierung und Steuerung beim Umgang mit dem Programm und der Ergebniserzeugung. Die vorgestellte Lösung ist noch nicht optimal und kann noch in einigen Aspekten verbessert werden. Sie zeigt aber bereits, dass Potential und ein Bedarf an solchen stark interaktiven Ansätzen zur visuellen Analyse von Trajektorien besteht.

10 Ausblick

Dieser Ansatz kann in einigen Bereichen noch verbessert werden. Deshalb folgt nun ein Ausblick auf mögliche Erweiterungen, die einige Nachteile dieser Arbeit minimieren, oder sogar entfernen könnten. Der vorgestellte Cluster-Algorithmus ist anfällig in Bezug auf variierende räumliche Daten-Dichten. Es gibt aber bereits einige Ansätze, die mit variablen Abstandparametern arbeiten, um dieses Problem zu umgehen. OPTICS [5] wurde in diesem Zusammenhang entwickelt, lässt sich aber in seiner ursprünglichen Form nur auf Datenpunkte anwenden. Durch eine Anpassung des Algorithmus an Trajektorien wäre er aber auch für diesen Ansatz geeignet und würde es ermöglichen, unterschiedlich dichte Cluster zu finden. Damit könnte der Ansatz besser auf Datensätzen arbeiten, welche verschieden dichte Gebiete aufweisen, wie es zum Beispiel beim Geolife Datensatz der Fall ist. Bereits angesprochen wurde auch die Indexierung. Durch eine geeignete Indexierung könnte die Effizienz des implementierten Algorithmus gesteigert werden, da bei der Nachbarberechnung viel weniger Elemente betrachtet werden müssen. Auch hier gibt es bereits eine Arbeit, welche es ermöglicht, verschiedene Indexstrukturen an Ansätze, wie den hier behandelten, anzupassen [27]. Dieses Verfahren wurde auch schon von den Autoren der implementierten berechnenden Komponenten [21] in Aussicht gestellt.
In den Verarbeitungsschritten des Programmes werden bisher auch Informationen, wie etwa die Zeit, oder die Höhe der Messpunkte aufgenommen, welche noch nicht genutzt werden. Hier wären Erweiterungen denkbar, welche diese zusätzlichen Informationen verwerten. Dadurch würden sich neue Optionen im Clusteringprozess, oder auch in der Analyse ergeben, welche sich interaktiv in das Verfahren einbringen lassen. Zum Beispiel könnte man die Möglichkeit bieten, nur Bewegungen eines gewissen Zeitraumes zu zeichnen. In [21] werden für jedes Cluster repräsentative Trajektorien ermittelt, um Nutzern die Orientierung zu erleichtern. Diese Methode könnte auch in diesem Ansatz implementiert werden, damit Nutzer schneller einen Überblick bekommen können. Durch weitere Anpassungen im Programm könnte man diese Repräsentativen schon während dem Clustering visualisieren und deren Veränderung in jedem Schritt aufzeigen.
Auch die Visualisierung könnte verbessert werden, indem man beispielsweise die Richtung jeder Trajektorie, anhand von Pfeilen, in die Karte einzeichnet. Ebenfalls denkbar wären problemorientiertere Karten, welche Nutzer, je nach Anwendungsfall, wählen können. Für manche Anwendungsfälle sind beispielsweise Karten mit Höheninformationen, oder Straßennetzkarten sinnvoll. Auch die Benutzeroberfläche und die Interaktionen können hinsichtlich der Benutzerfreundlichkeit optimiert werden. Hier könnte man Werte, wie Epsilon oder MinLns, je nach Datensatz approximieren, damit Nutzer nicht erst nach guten Parametern suchen müssen. Viele etablierte Analyse-Werkzeuge würden Nutzern zusätzlich die Analyse erleichtern. Dazu gehören beispielsweise Werkzeuge, mit denen sich einzelne Trajektorien, hinsichtlich ihrer Distanz, oder ihrer zeitlichen Differenz zueinander, vergleichen lassen. Bei gelabelten Datensätzen könnte man auch das entsprechende Objekt zu jeder Bewegung anzeigen lassen, oder Nutzern die Möglichkeit geben, nur Trajektorien von bestimmten Objekten zu visualisieren. Auch eine einfachere Einbindung von verschiedenen Datensätzen würde den Ansatz benutzerfreundlicher machen.

Literaturverzeichnis

[1] *From Movement Tracks through Events to Places: Extracting and Characterizing Significant Places from Mobility Data*, 2011

[2] A, S.; Wang, X.; Wang, X.; Wang, X.; Hamilton, H. J.; Hamilton, H. J.; Hamilton, H. J.: *DBRS: A Density-Based Spatial Clustering Method with Random Sampling*. In: In: Proc. of the 7th PAKDD, Seoul, Korea (2003) 563 ? 575, Seiten 563–575. 2003

[3] Andrienko, G.; Andrienko, N.; Rinzivillo, S.; Nanni, M.; Pedreschi, D.; Giannotti, F.: *Interactive visual clustering of large collections of trajectories*. In: Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on, Seiten 3–10. 2009

[4] Andrienko, G.; Andrienko, N.; Wrobel, S.: *Visual analytics tools for analysis of movement data*. Vol. 9.2, Seiten 38–46. Dec. 2007

[5] Ankerst, M.; Breunig, M. M.; Kriegel, H.-P.; Sander, J.: *OPTICS: Ordering Points to Identify the Clustering Structure*. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, Seiten 49–60. ACM 1999

[6] Berkhin, P.: *A survey of clustering data mining techniques*. Grouping Multidimensional Data, Seiten 25–71. 2006

[7] Chen, C.-S.; Eick, C. F.; Rizk, N. J.: *Mining Spatial Trajectories Using Non-parametric Density Functions.* In: Perner, P. (Hrsg.): MLDM, Seiten 496–510. Springer 2011

[8] Ester, M.; peter Kriegel, H.; S, J.; Xu, X.: *A density-based algorithm for discovering clusters in large spatial databases with noise*. Seiten 226–231. AAAI Press 1996

[9] Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P.: *The kdd process for extracting useful knowledge from volumes of data*. Commun. ACM, Vol. 39.11, Seiten 27–34. Nov. 1996

[10] Ferreira, N.; Klosowski, J. T.; Scheidegger, C. E.; Silva, C. T.: *Vector field k-means: Clustering trajectories by fitting multiple vector fields*. CoRR, Vol. abs/1208.5801. 2012. URL `http://dblp.uni-trier.de/db/journals/corr/corr1208.html#abs-1208-5801`

[11] Fu, Z.; Hu, W.; Tan, T.: *Similarity based vehicle trajectory clustering and anomaly detection*. In: Image Processing, 2005. ICIP 2005. IEEE International Conference on, Seiten II–602–5. Sept 2005

[12] Gaffney, S.; Smyth, P.: *Trajectory Clustering with Mixtures of Regression Models*. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seiten 63–72. ACM 1999

[13] Grnwald, P. D.; Myung, I. J.; Pitt, M. A.: *Advances in Minimum Description Length: Theory and Applications (Neural Information Processing)*. The MIT Press 2005

[14] Guttman, A.: *R-trees: A Dynamic Index Structure for Spatial Searching*. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, Seiten 47–57. ACM 1984

[15] Han, J.; Kamber, M.: *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc. 2000

[16] Höferlin, M.; Höferlin, B.; Heidemann, G.; Weiskopf, D.: *Interactive schematic summaries for faceted exploration of surveillance video*. IEEE Transactions on Multimedia 15, Vol. 4, Seiten 908–920. 2013

[17] Keim, D.; Andrienko, G.; Fekete, J.-D.; Görg, C.; Kohlhammer, J.; Melançon, G.: Information visualization: Kapitel Visual Analytics: Definition, Process, and Challenges, Seiten 154–175: Springer-Verlag 2008

[18] Keim, E. D.; Kohlhammer, J.; Ellis, G.: *Mastering the information age: Solving problems with visual analytics, eurographics association*, 2010

[19] Knapp, K. R.; Kruk, M. C.; Levinson, D. H.; Diamond, H. J.; Neumann, C. J.: *The international best track archive for climate stewardship (ibtracs) unifying tropical cyclone data*. Bulletin of the American Meteorological Society, Vol. 91.3, Seiten 363–376. 2010

[20] Lee, J.-G.; Han, J.; Li, X.: *Trajectory Outlier Detection: A Partition-and-Detect Framework*. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, Seiten 140–149. IEEE Computer Society 2008

[21] Lee, J.-G.; Han, J.; Whang, K.-Y.: *Trajectory Clustering: A Partition-and-group Framework*. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, Seiten 593–604. ACM 2007

[22] Nanni, M.; Pedreschi, D.: *Time-focused clustering of trajectories of moving objects*. J. Intell. Inf. Syst., Vol. 27.3, Seiten 267–289. Nov. 2006

[23] Palma, A. T.; Bogorny, V.; Kuijpers, B.; Alvares, L. O.: *A Clustering-based Approach for Discovering Interesting Places in Trajectories*. In: Proceedings of the 2008 ACM Symposium on Applied Computing, Seiten 863–868. ACM 2008

[24] Parent, C.; Spaccapietra, S.; Renso, C.; Andrienko, G.; Andrienko, N.; Bogorny, V.; Damiani, M. L.; Gkoulalas-Divanis, A.; Macedo, J.; Pelekis, N.; Theodoridis, Y.; Yan, Z.: *Semantic trajectories modeling and analysis*. ACM Comput. Surv., Vol. 45.4, Seiten 42:1–42:32. 2013

[25] Raab, J.: *Visuelle wissenssoziologie. theoretische konzeption und materiale analysen*. 2008

[26] Rinzivillo, S.; Pedreschi, D.; Nanni, M.; Giannotti, F.; Andrienko, N.; Andrienko, G.: *Visually driven analysis of movement data by progressive clustering*. Information Visualization, Vol. 7.3, Seiten 225–239. 2008

[27] Roth, V.; Laub, J.; Kawanabe, M.; Buhmann, J. M.: *Optimal cluster preserving embedding of nonmetric proximity data*. IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 25. 2003

[28] Sander, J.; Ester, M.; Kriegel, H.-P.; Xu, X.: *Density-based clustering in spatial databases: The algorithm gdbscan and its applications*. Data Min. Knowl. Discov., Vol. 2.2, Seiten 169–194. 1998

[29] Steele, J.; Iliinsky, N.: *Beautiful Visualization: Looking at Data Through the Eyes of Experts*. 1st. Auflage O'Reilly Media, Inc. 2010

[30] Thomas, J. J.; Cook, K. A.: *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr 2005

[31] Van de Weghe, N.; Cohn, A. G.; De Tre, G.; De Maeyer, P.: *A qualitative trajectory calculus*

as a basis for representing moving objects in geographical information systems. Control and Cybernetics, Vol. 35.1, Seite 97. 2006

[32] Wang, J.; Wang, N.; Jia, Y.; Li, J.; Zeng, G.; Zha, H.; Hua, X.: *Trinary-projection trees for approximate nearest neighbor search*. IEEE Trans. Pattern Anal. Mach. Intell., Vol. 36.2, Seiten 388–403. 2014

[33] Yianilos, P. N.: *Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces*. In: Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, Seiten 311–321. Society for Industrial and Applied Mathematics 1993

[34] Zheng, Y.; Fu, H.: *Geolife GPS trajectory dataset - User Guide*, August 2012

[35] Zheng, Y.; Li, Q.; Chen, Y.; Xie, X.; Ma, W.-Y.: *Understanding Mobility Based on GPS Data*. In: Proceedings of the 10th International Conference on Ubiquitous Computing, Seiten 312–321. New York, NY, USA: ACM 2008

[36] Zheng, Y.; Xie, X.; Ma, W.-Y.: *Geolife: A collaborative social networking service among user, location and trajectory.* IEEE Data Eng. Bull., Vol. 33.2, Seiten 32–39. 2010

[37] Zheng, Y.; Zhang, L.; Xie, X.; Ma, W.-Y.: *Mining Interesting Locations and Travel Sequences from GPS Trajectories*. In: Proceedings of the 18th International Conference on World Wide Web, Seiten 791–800. New York, NY, USA: ACM 2009

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

_____

(Philipp Göttlich)