
Masterarbeit

Erkennung von Fahrspuren mittels Fahrzeugtrajektorien aus Luftaufnahmen

im Master-Studiengang Informatik
der Hochschule Furtwangen

Steffen Schmid

Zeitraum: Wintersemester 2018
Prüfer: Prof. Dr. Christoph Reich
Zweitprüfer: Dr. Stefan Kaufmann

Firma: IT-Designers GmbH
Betreuer: Dr. Stefan Kaufmann

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Esslingen, den 16. November 2018 _____
Unterschrift

Zitat

„Some fancy quote“

Foobar Muman

Danksagung

Kurzfassung

Schlagworte:

Inhaltsverzeichnis

Kurzfassung	v
Abkürzungsverzeichnis	xi
1 Einleitung	1
1.1 Rahmen der Arbeit	1
1.1.1 Das Projekt MEC-View	1
1.1.2 Teilprojekt MEC-View Luftbeobachtung	1
1.2 Motivation und Ziele	1
1.3 Aufbau dieser Arbeit	1
2 Grundlagen	2
2.1 Rekonstruktion von Fahrzeugtrajektorien aus Luftaufnahmen	2
2.1.1 Erkennung und Verfolgung von Fahrzeugen in Videoaufnahmen	2
2.1.2 Positionsbestimmung in Videoaufnahmen	2
2.2 Clusteranalyse	2
2.2.1 Eigenschaften von Cluster-Sets und Clustern	4
2.2.2 Cluster-Algorithmen	6
2.3 Distanzmaße zum Vergleich von Fahrzeugtrajektorien	13
2.3.1 HU Distanz	14
2.3.2 Hausdorff Distanz	14
2.3.3 Longest-Common-Subsequence	15
3 Verwandte Arbeiten	17
3.1 Clusteranalyse von Trajektorien	17
3.2 Erkennung und Definition von Fahrspuren	23
3.3 Defizite vorhandener Lösungen und benötigte Neuerungen	27
4 Untersuchung möglicher Straßentopologien	28
4.1 Landstraßen	28
4.2 Autobahnen	28
4.3 Kreuzungen	28
4.4 Kreisverkehre	28
5 Konzeption des Spurerkennung-Moduls	29
5.1 Überblick über das Gesamtsystem	29

5.2	Anforderungen an das Modul	30
5.2.1	Funktionale Anforderungen	30
5.2.2	Nicht funktionale Anforderungen	30
5.3	Entwurf des Moduls Spurerkennung	31
6	Clusteranalyse von Fahrzeugtrajektorien	32
6.1	Trajektorie-Definition	32
6.2	Vorverarbeitung der Trajektorien	35
6.2.1	Resampling von Trajektorien auf minimale Punktdistanz	35
6.2.2	Entfernung zu kurzer Trajektorien	36
6.2.3	Entfernung unterbrochener Trajektorien	37
6.2.4	Anpassung der Trajektorien bei niedrigen Aufnahmewinkeln	38
6.3	Clusteranalyse	38
6.3.1	Ansatz Spectral-Clustering und modifizierte Hausdorff-Distanz	38
6.3.2	Ansatz DBSCAN und LCSS	41
6.3.3	Erkennung von Abbiegespuren	44
7	Fahrspur-Bestimmung aus Trajektorie-Clustern	47
7.1	Ausfilterung von Spurwechselvorgängen	47
7.2	Bestimmung der Spurmittellinien	49
7.3	Bestimmung der Spurhüllen	50
7.4	Partitionierung von Fahrspuren	54
8	Ergebnisse und Auswertung	55
9	Zusammenfassung und Fazit	56
	Literaturverzeichnis	57

Abbildungsverzeichnis

2.1	Rohdaten (links) und erwünschtes Clustering-Ergebnis (rechts)	3
2.2	Ablauf einer Clusteranalyse	4
2.3	Visualisierung verschiedener Clusterarten	6
2.4	Agglomeratives Clustering dargestellt als Dendrogramm (links) und geschachteltes Cluster-Diagramm (rechts)	7
2.5	Funktionsweise von k-Means	9
2.6	Darstellung des EM-Cluster-Algorithmus über mehrere Iterationen	11
2.7	Schritte des DBSCAN Algorithmus	12
2.8	Erreichbarkeit in DBSCAN	12
2.9	Trajektorien im 2-dimensionalen Raum	13
3.1	Zweistufiger Clustering-Vorgang von Hu et al.	18
3.2	Funktionsweise der modifizierten Hausdorff Distanz	19
3.3	Zerlegung einer Trajektorie in Sub-Trajektorien	20
3.4	Verschiebung einer Trajektorie im Raum	21
3.5	Ergebnisse der Spurmittellinien-Erkennung (Ren et al.)	23
3.6	Spurhüllen in Hu et al., 2006	24
3.7	Routen-Definition und Ergebnisse Routen-Erkennung (Makris et al.)	25
3.8	Ergebnisse Histogramm Erstellung und Spurextraktion (Hsieh et al.)	27
5.1	Kontext des Spurerkennung Moduls	29
5.2	Grundstruktur des Spurerkennungs-Algorithmus	31
6.1	Aufbau Trajektorie-Klasse	33
6.2	Unverarbeitete Trajektorien vom Stuttgarter Neckartor	33
6.3	Das Stuttgarter Neckartor	34
6.4	a) Punktwolken vor Lichtsignalanlagen, b) Unterbrechungen aufgrund von Überdeckung	34
6.5	Ergebnisse der zwei ersten Vorverarbeitungsschritte	37
6.6	Definition von Szenen Rand- und Innenbereich	37
6.7	Ergebnisse Clusteranalyse B10-Entennest (Ansatz Atev et al.)	41
6.8	Ergebnisse Clusteranalyse DBSCAN und LCSS-Distanz	44
6.9	a) Abbiegespur, b) zugehörige Trajektorien, c) Spurcluster mit Abbiegespuren	45
7.1	Trajektorie-Cluster mit Spurwechselvorgängen	47
7.2	Trajektorie-Cluster ohne Spurwechselvorgänge	48

7.3	Spurmittellinie in einem Trajektorie-Cluster a), Spurmittellinien Neckartor-Kreuzung b)	50
7.4	Bestimmung Spürhüllen	51
7.5	Aufbau LaneGeometry Klasse	51
7.6	Bestimmung Spürhüllen	53
7.7	Plot Spurmittellinien und Hüllen a), Ergebnis Spur-Geometrien in TrackerApplication b)	54

Listings

6.1	Pseudocode Trajektorie Resampling	36
6.2	Pseudocode LCSS Bestimmung	42
6.3	Pseudocode Split-Punkt Bestimmung	46
7.1	Pseudocode Cluster Post-Processing	48
7.2	Pseudocode Cluster Post-Processing	49
7.3	Pseudocode Überprüfung der Parallelität zweier Mittellinien	52

Abkürzungsverzeichnis

DBSCAN	Density-based spatial clustering of applications with noise
DTW	Dynamic Time Warping
EM	Expectation-Maximization
GMM	Gaussian-Mixture-Models
HD	Hausdorff Distanz
LCSS	Longest Common Subsequence
MEC	Mobile Edge Computing
PCA	Principal Component Analysis
RAA	Richtlinien für die Anlage von Autobahnen
RAL	Richtlinien für die Anlage von Landstraßen

1 Einleitung

Staus und zäh fließender Verkehr sind sowohl auf Schnell- und Autobahnen, als auch in Städten ein großes Problem und Ärgerniss für Autofahrer. Sie kosten diese nicht nur wertvolle Zeit, sondern auch viel Geld. Laut einer Studie von [Cookson u. a.] kostet Stau jeden deutschen Autofahren pro Jahr durchschnittlich 1770 €. In Summe ergeben sich hieraus beinahe 80 Milliarden Euro an Kosten. Stau ist allerdings nicht nur finanziell für Privatpersonen oder auch Unternehmen ein großer Faktor, sondern er erhöht auch das Unfallrisiko und trägt maßgeblich zur schlechten Luftqualität in Innenstädten bei. Aufgrund langerer Fahrzeiten und der häufigen Be- und Entschleunigung, steigt der Kraftstoffverbrauch der Fahrzeuge und dadurch auch die Schadstoffbelastung in der Luft [Hemmerle, 2016].

Um Stau so gut wie möglich vermeiden zu können, muss man den Verkehr verstehen. Nötig ist ein Verständnis des Straßenverkehrs als Ganzes, sowie der Auswirkungen, welche einzelne Verkehrsteilnehmer und deren Verhalten, auf diesen haben. Hierzu ist das Erstellen von Simulationen sowie die Auswertung realer Verkehrsaufkommen unerlässlich. Die auf diese Weise gesammelten Erkenntnisse bilden die Grundlage, um Straßenabschnitte, insbesondere auch in Innenstädten, intelligent zu gestalten. Des Weiteren können sie eingesetzt werden, um beispielsweise Ampelschaltungen in Städten zu optimieren, wovon auch bestehende Infrastrukturen profitieren können.

Diese Arbeit beschäftigt sich mit der Realisierung einer automatischen Fahrspurerkennung aus Luftaufnahmen, welche bei der Analyse von Spurwechselvorgängen zum Einsatz kommt. Hierzu werden die Trajektoriedaten von Fahrzeugen ausgewertet.

1.1 Rahmen der Arbeit

1.1.1 Das Projekt MEC-View

1.1.2 Teilprojekt MEC-View Luftbeobachtung

1.2 Motivation und Ziele

1.3 Aufbau dieser Arbeit

2 Grundlagen

In diesem Kapitel werden die für das Verständnis und die Durchführung der Arbeit benötigten Grundlagenthemen vorgestellt. Als erstes wird erläutert, auf welche Weise Fahrzeugtrajektorien aus Videoaufnahmen rekonstruiert werden können. Anschließend werden verschiedene Konzepte und Algorithmen aus dem Bereich der Clusteranalyse vorgestellt, welche dabei helfen können, Gemeinsamkeiten und Beziehungen in Daten zu entdecken. Am Ende des Kapitels werden Distanzmaße, insbesondere jene, welche sich zum Vergleich von Fahrzeugtrajektorien eignen, betrachtet. Es wird aufgezeigt, welchen Einfluss diese auf die Ergebnisse einer Clusteranalyse haben, und welche Abwägungen bei der Wahl eines Maßes angestellt werden müssen.

2.1 Rekonstruktion von Fahrzeugtrajektorien aus Luftaufnahmen

2.1.1 Erkennung und Verfolgung von Fahrzeugen in Videoaufnahmen

2.1.2 Positionsbestimmung in Videoaufnahmen

2.2 Clusteranalyse

Die Clusteranalyse (kurz Clustering) ist ein wichtiges Werkzeug zur Auswertung von Daten unterschiedlichster Art. Sie stellt dabei kein konkretes Vorgehen oder einen Algorithmus dar, sondern beschreibt ein allgemeines Problem, welches auf unterschiedlichste Weise gelöst werden kann. Grundsätzlich ist das Ziel der Clusteranalyse, Datenobjekte aufgrund ihrer Eigenschaften und Beziehungen untereinander so zu gruppieren, dass sich die Objekte einer Gruppe möglichst stark ähneln und sich von den Objekten anderer Gruppen möglichst stark unterscheiden. Die auf diese Art entstehenden Objektgruppen werden *Cluster* genannt. Je höher die *Homogenität* in einem Cluster und die *Differenz* zwischen den Clustern, desto besser ist die gewählte Clustering Methode. Der Einsatz einer Clusteranalyse ist in vielen Anwendungsbereichen und in den unterschiedlichsten wissenschaftlichen Disziplinen sehr beliebt, um ein Verständnis für Daten zu erhalten, beziehungsweise diese, nach einer Gruppierung, sinnvoll weiter verarbeiten zu können. So kommt die Clusteranalyse unter anderem in den Feldern des maschinellen Lernens, der Mustererkennung, Bildanalyse, der Biologie (Taxonomie) oder im Bereich Data Mining zum Einsatz. [Tan u. a., 2007]

Die Clusteranalyse hat viel mit dem Problem der Klassifizierung von Daten gemein, insofern sie Datenobjekten Label zuordnet. Im Gegensatz zu *überwachten* Klassifizierungsansätzen, wie dem heute populären überwachten Lernen, leiten Cluster-Algorithmen die Label allerdings alleine aus den vorhandenen Daten ab. Es kommen keine Vergleichsobjekte mit bekannten, händisch vergebenen Labels zum Einsatz. Aus diesem Grund wird die Clusteranalyse auch häufig als *unüberwachte Klassifizierung* bezeichnet. [Tan u. a., 2007]

Das Konzept eines *Clusters* ist nicht genau definiert. Es existieren daher viele unterschiedliche Konzepte und Algorithmen, welche sich jeweils für andere Anwendungsfälle eignen und verschiedene Eigenschaften besitzen. Hieraus ergibt sich auch die Tatsache, dass das Clustering kein selbsttätiger Prozess ist, welcher sich auf einheitliche Weise auf unterschiedliche Probleme anwenden lässt. Jedes Problem erfordert die individuelle und sorgfältige Auswahl eines passenden Algorithmus, eines Distanzmaßes und der richtigen Parameter. Die Bestimmung dieser geschieht iterativ und nicht selten nach dem Prinzip des *Trial and Error*. In Abbildung 2.1 ist beispielhaft ein Datensatz (links) mit – für den Menschen intuitiv ersichtlich – 7 unterschiedlichen Clustern (rechts) dargestellt. Nach [Jain, 2010] kann allerdings kein existierender Clustering Algorithmus diese alle erkennen. [Jain u. a., 1999; Tan u. a., 2007]

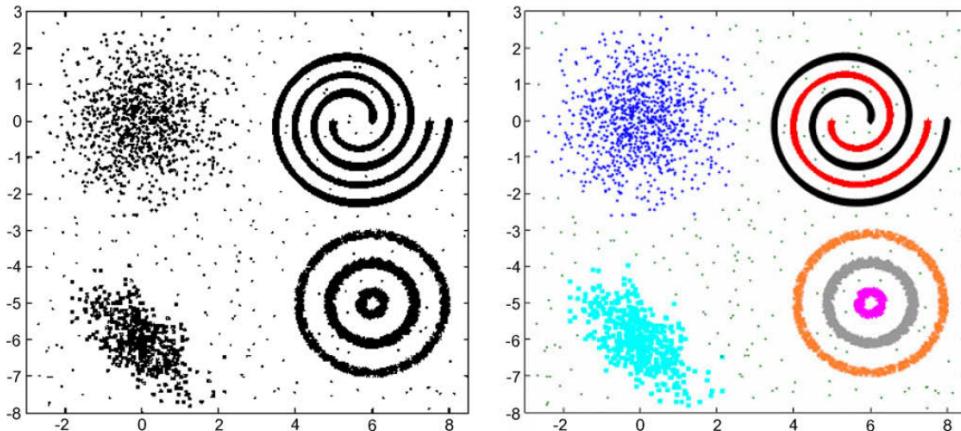


Abb. 2.1: Rohdaten (links) und erwünschtes Clustering-Ergebnis (rechts) [Jain, 2010]

Aufgrund der Limitationen, welche alle Cluster-Algorithmen besitzen, muss der Analyst sich vor deren Anwendung intensiv mit den zu verarbeitenden Daten beschäftigen. Er muss ein Verständnis dafür besitzen, welche Struktur die Daten haben, beziehungsweise annehmen können, und nach welchen Mustern zu suchen ist. Besonders wichtiger ist zudem auch die Auswahl der richtigen, das heißt relevanten, Datenmerkmale („*Feature Selektion*“) und die Wahl deren Repräsentation („*Feature Transformation*“). Die Selektion und gegebenenfalls Transformation der Daten muss in einem Vorverarbeitungsschritt geschehen, dessen Qualität einen maßgeblichen Einfluss auf das finale Clustering Ergebnis hat. Basierend auf vorangegangener Beschreibung und [Jain u. a., 1999], lässt sich der grundlegende Ablauf einer Clusteranalyse wie folgt darstellen:

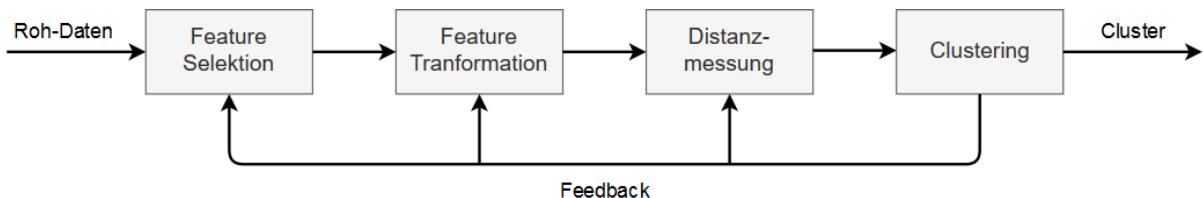


Abb. 2.2: Ablauf einer Clusteranalyse

[Jain, 2010] nennt einige weitere Herausforderungen, welcher man sich bei der Clusteranalyse bewusst sein muss:

1. Daten können Ausreißer enthalten. Wie sollen diese behandelt werden?
2. Die Anzahl der Zielcluster ist üblicherweise nicht bekannt. Wie kann sie im voraus bestimmt werden, wenn die Analyse es erfordert?
3. Wie können gefundenen Cluster validiert werden?

2.2.1 Eigenschaften von Cluster-Sets und Clustern

Cluster-Sets, das heißt die Gesamtheit aller durch eine Analyse gefundenen Cluster, und einzelnen Cluster selbst, können in verschiedene Kategorien unterteilt werden, beziehungsweise unterschiedliche Eigenschaften besitzen. Nachfolgend sind die wichtigsten basierend auf [Tan u. a., 2007] und [Jain u. a., 1999; Jain, 2010] aufgeführt.

Eigenschaften von Cluster-Sets

Bei Cluster-Sets kann grundsätzlich zwischen nachfolgenden Eigenschaften unterschieden werden.

Hierarchisch vs. Partitioniert Von *hierarchischen* Cluster-Sets wird gesprochen, wenn die einzelnen Cluster verschachtelt sind und dabei eine Baum-Struktur bilden. Cluster sind hingegen *partitioniert*, wenn keine Überlagerungen zwischen ihnen existieren.

Exklusiv vs. Überlappend vs. Fuzzy *Exklusive* Cluster-Sets liegen vor, wenn jedem Datenwert ein oder kein Zielcluster zugeordnet wird. Im Gegensatz hierzu können bei *überlappenden* Cluster-Sets Objekte einer oder mehrerer Gruppen angehören. Bei dem sogenannten *Fuzzy* oder *Soft* Cluster-Sets, gehört ein Datenobjekt einem Cluster mit einer bestimmten Wahrscheinlichkeit oder Gewicht an. Algorithmen, welche Daten eine Wahrscheinlichkeit für die Zugehörigkeit zu einem Cluster zuweisen, werden *probabilistische* Cluster-Algorithmen genannt.

Komplett vs. Partielle Von *kompletten* Cluster-Sets wird gesprochen, wenn jedes Element der Eingangsdaten einem Cluster zugeordnet wird. Bei *partiellen* Sets ist dies nicht der Fall. Hier kann ein bestimmter Anteil an Datenwerten als Ausreißer markiert werden, welche keine Gruppe besitzen.

Eigenschaften von Clustern

Da, wie oben erwähnt, nicht klar definiert ist, was ein Cluster ausmacht, können auch diese unterschiedliche Eigenschaften besitzen. Die wichtigsten Cluster-Arten sind nachfolgend erläutert.

Klar separierte Cluster Unter *klar separierten* Clustern versteht man solche, in welchen jedes Datenelement einen geringeren Abstand zu allen anderen Elementen des Clusters hat, als zu Elementen außerhalb des Clusters. Dargestellt ist dies in Abbildung 2.3 a). Diese idealistische Definition eines Clusters ist nur dann erfüllt, wenn die in den Daten enthaltenen Cluster einen großen Abstand voneinander haben. Dies ist in der Realität allerdings selten der Fall.

Prototyp basierte Cluster Von einem *Prototyp basierten* Cluster wird gesprochen, wenn alle Elemente einer Gruppe einen geringeren Abstand zu einem Prototyp oder Referenzwert des Clusters besitzen, als zu denen anderer Gruppierungen (siehe Abb. 2.3 b)). Ein solcher Prototyp ist üblicherweise der Mittelwert der Datenelemente eines Clusters (*Centroid*).

Graphen basierte Cluster Die Definition eines *Graphen basierten* Clusters kann immer dann verwendet werden, wenn Daten als vernetzter Graph dargestellt werden. In einem solchen sind die Elemente Knoten und die Kanten repräsentieren Beziehungen zwischen ihnen. Ein Cluster in einem solchen Graphen ist definiert als Menge von Knoten, welche untereinander verbunden sind, jedoch keine Verbindungen zu Elementen außerhalb des Clusters haben. Dargestellt ist dies in Abbildung 2.3 c).

Dichte basierte Cluster *Dichte basierte* Cluster sind definiert als Regionen mit einer hohen Dichte an Objekten, welche von Regionen umgeben sind, welche eine geringe Objektdichte besitzen (siehe Abb. 2.3 d)). Elemente, welche in einer solchen Region mit geringen Dichte liegen, werden als Ausreißer interpretiert. Dichte Bereiche werden üblicherweise gefunden, indem die Nachbarschaften von Elementen untersucht werden.

Konzeptionelle Cluster Eine sehr allgemeine Definition eines Clusters ist die der *konzeptionellen* Gruppen. Hiermit ist gemeint, dass die Elemente eines Clusters einige gemeinsame Eigenschaften besitzen. Dies schließt die oben genannten Cluster-Arten mit ein, lässt sich allerdings beliebig erweitern. So sind beispielsweise in Abbildung 2.3 e) konzeptionelle Cluster dargestellt, die die Form zweier Kreise und eines Rechtecks haben. Um solche Muster erkennen zu können, benötigt ein Algorithmus ein spezielles Verständnis eines Clusters.

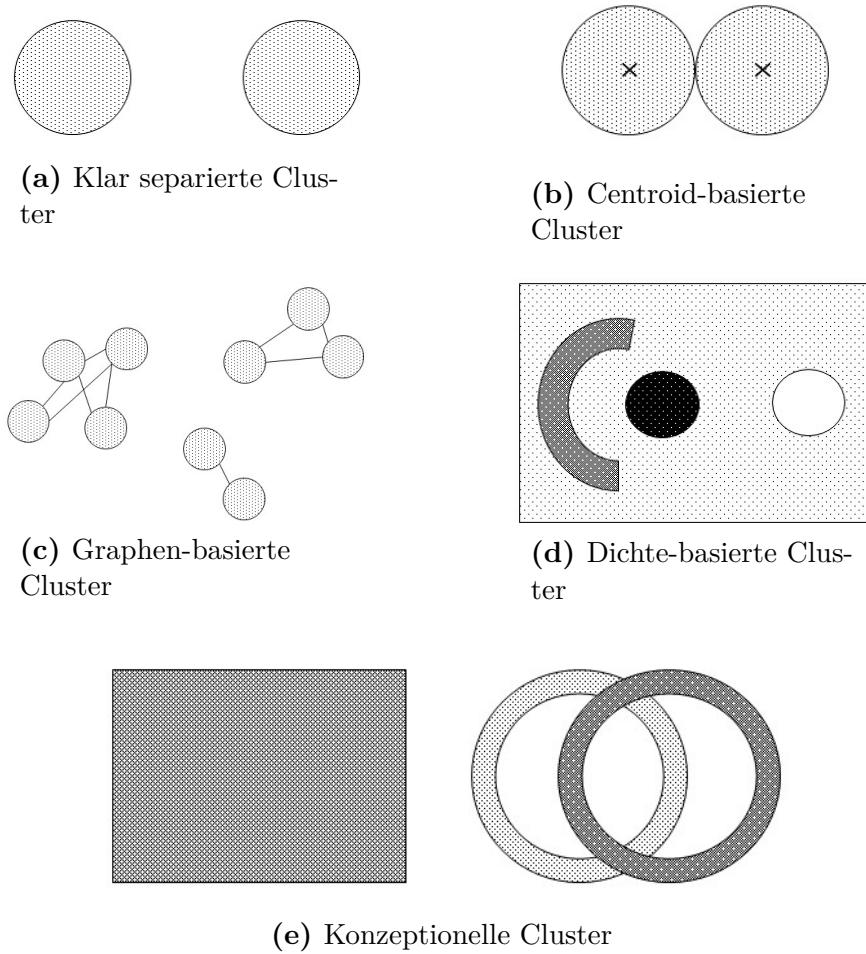


Abb. 2.3: Visualisierung verschiedener Clusterarten (basierend auf [Tan u. a., 2007])

2.2.2 Cluster-Algorithmen

Um mit den oben beschriebenen unterschiedlichen Cluster-Sets und Cluster Definitionen umgehen zu können, existieren verschiedene Clustering-Modelle. Einige wichtige Clustering-Ansätze sind die Vernetzungs-Modelle, Centroid-basierte-Modelle, Verteilungs-Modelle oder

Dichte-Modelle. Für jedes dieser Modelle existieren unterschiedliche Algorithmen. Im Folgenden werden diese Modelle und jeweils exemplarisch ein Algorithmus, der diese vertritt, vorgestellt.

Vernetzungs-Modelle

Vernetzungs-Modelle werden auch häufig *hierarchische Cluster-Modelle* genannt. Sie beruhen auf der Annahme, dass Elemente, welche nahe beieinander liegen, eine höhere Gemeinsamkeit besitzen als solche, welche weiter voneinander entfernt sind. Zur Bestimmung der Nähe zwischen Elementen benötigen Vernetzungs-Modelle, wie auch andere Cluster-Modelle, eine Definition von Distanz. Diese legt ein sogenanntes *Distanzmaß* fest. Zusätzlich ist ein *Link-Kriterium* notwendig, welches bestimmt, wie genau die Entfernung zwischen zwei Clustern ermittelt wird. Übliche Link-Kriterien sind *Minimum-Linkage*, welches die minimale Distanz zwischen den Objekten der Cluster als Distanz verwendet, oder *Maximum-Linkage* beziehungsweise *Average-Linkage*. [Jain u. a., 1999; George Seif, 2018]

Grundsätzlich teilen sich hierarchische Cluster-Algorithmen in zwei Gruppen auf: *Agglomerative* (Bottom-Up) und *Divisive* (Top-Down) Algorithmen. Agglomerative Ansätze weisen zu Beginn des Cluster-Vorgangs jedem Datenelement eine eigene Gruppe zu und vereinigen diese anschließend. Bei divisiven Ansätzen werden hingegen zu Beginn alle Elemente in einem Cluster zusammengefasst und diese in den nachfolgenden Schritten geteilt.

Als Beispiel wird anschließend der *agglomerative-hierarchische Cluster-Algorithmus* genauer vorgestellt. Sein Vorgehen lässt sich sehr gut anhand sogenannter Dendrogramme oder geschachtelter Cluster-Diagramme darstellen (siehe Abbildung 2.4).

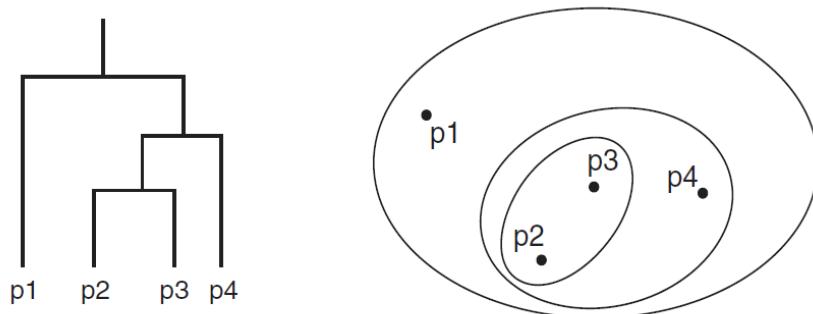


Abb. 2.4: Agglomeratives Clustering dargestellt als Dendrogramm (links) und geschachteltes Cluster-Diagramm (rechts)

Im ersten Schritt des Algorithmus werden alle Datenpunkte als separate Cluster markiert. Diesen Schritt repräsentieren die Blätter des Dendrogramms. Anschließend muss ein Distanzmaß und ein Link-Kriterium gewählt werden. Das am häufigsten verwendete

Distanzmaß ist sicherlich der euklidsche Abstand, welcher die Distanz zwischen zwei Punkten oder Vektoren im n -dimensionalen Raum bestimmt. Er ist definiert durch die Formel 2.1.

$$dist(p, q) = \|q - p\|_2 = \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \quad (2.1)$$

Wird als Link-Kriterium beispielsweise *Minimum-Linkage* gewählt, ist dieses definiert als:

$$link(P, Q) = \min\{dist(p, q) | p \in P, q \in Q\} \quad (2.2)$$

Hierbei entsprechen P und Q zwei Clustern, welche die Elemente $p \in P$ und $q \in Q$ enthalten. Auf Basis des gewählten Link-Kriteriums kann nun eine Distanz-Matrix für die einzelnen Cluster erstellt werden. Die zwei Cluster mit minimalem Abstand voneinander werden anschließend zusammengeführt und die vorherigen Schritte werden wiederholt, bis nur noch ein Cluster (Wurzel des Dendrogramms) beziehungsweise die gewünschte Clusteranzahl übrig ist. [George Seif, 2018; Tan u. a., 2007]

Bei den meisten Varianten des agglomerativen Clusterings muss der Nutzer die Anzahl der Zielcluster im vorraus festlegen, was problematisch ist, da diese meist nicht bekannt ist. Umgangen werden kann dies nur, indem ein Link-Kriterium gewählt wird, das ab einer bestimmten Distanz zwischen den Clustern diese nichtmehr fusioniert [George Seif, 2018].

Die Zeitkomplexität des agglomerativen Clusterings beträgt bestenfalls $O(m^2 \log m)$, weshalb die Menge der Daten, welche mit ihm verarbeitet werden können erheblich begrenzt ist [Tan u. a., 2007].

Prototyp-Modelle

Centroid basierte Cluster-Modelle betrachten im Gegensatz zu hierarchischen Modellen nicht die Distanz zwischen Clustern, sondern die Entfernung von Objekten zu Referenzpunkten, sogenannten *Prototypen*. Die am häufigsten verwendeten Prototypen sind *Centroids*, welche den Mittelpunkt eines Clusters darstellen. *Medoids*, welche auch häufig genutzt werden, repräsentieren hingegen den Median eines Clusters.

Ein Beispiel für einen Centroid-Cluster-Algorithmus ist *k-Means*. Dieser ist aufgrund seines Alters, seiner Einfachheit und der vielen Weiterentwicklungen wohl der bekannteste Cluster-Algorithmus überhaupt.

Das Ziel von k-Mean ist es, für eine n-dimensionale Punktmenge $X = \{x_1 \dots x_n\}$ ein Cluster-Set $C = \{c_1 \dots c_k\}$ zu finden, welches die Summe der quadratischen Abweichung (Gleichung

2.3) zwischen allein Punkten in einem Cluster und deren Mittelwert μ_k (Centroids) minimiert.

$$J(c_k) = \sum_{k=1}^K \sum_{x_i \in c_k} \|x_i - \mu_k\|^2 \quad (2.3)$$

Eine Lösung für dieses Problem zu finden, ist NP-Schwer. Aus diesem Grund ist k-Means ein approximativer Ansatz, welcher nicht garantieren kann, ein globales Minimum zu finden. Die Funktionsweise des Algorithmus ist in Abbildung 2.5 dargestellt. Die Kreuze entsprechen hierbei den Centroids, welche sich über die Iterationen hinweg verschieben.

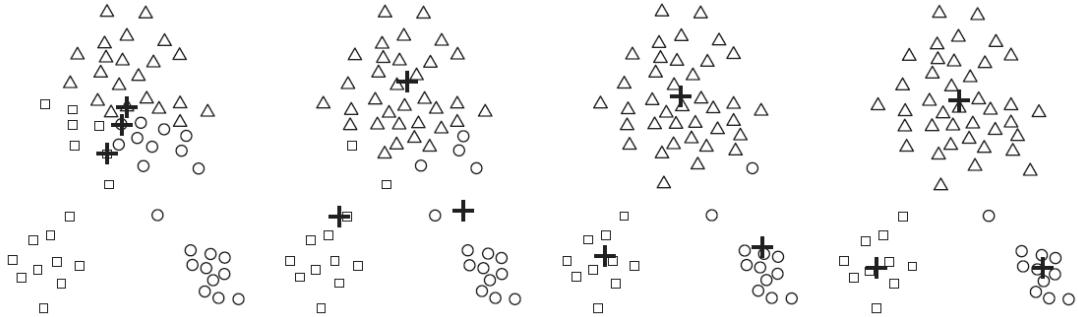


Abb. 2.5: Funktionsweise von k-Means [Tan u. a., 2007]

Ausgehend von der Punktmenge X und der gesuchten Cluster-Anzahl k , werden im ersten Schritt k zufällig positionierte Centroids μ_k definiert. Anschließend wird für alle Punkte x_i der nächstgelegene Centroid μ_j gesucht.

$$j = \arg \min(\text{dist}(x_i, \mu_j)) \quad (2.4)$$

x_i wird daraufhin Mitglied in Cluster C_j . Als Distanzmaß (dist) kann hier wieder der euklidische Abstand (Gleichung 2.1) verwendet werden oder aber auch beliebige andere sinnvolle Metriken. Nachdem alle Punkte x_i einem Cluster zugewiesen wurden, werden die Centroid Positionen neu bestimmt. Hierzu wird der Durchschnitt aller Punkte eines Clusters berechnet:

$$c_j = \frac{1}{n} \sum_{x_j \in C_j} x_j \quad (2.5)$$

Diese zwei Schritte werden mehrfach wiederholt, bis das Ergebnis konvergiert, das heißt die Zuweisungen sich nur noch geringfügig ändern. [Jain, 2010]

Der primäre Nachteil des k-Means Algorithmus ist, dass auch bei ihm die Anzahl der Zielcluster spezifiziert werden muss. Des Weiteren ist sein Ergebnis aufgrund der zufälligen

Initialisierung der Centroids nicht deterministisch. Vorteil von k-Means ist hingegen, dass seine Zeitkomplexität bei $O(n)$ liegt.

Um die genannten Nachteile, zumindest in Teilen, umgehen zu können, existieren diverse Weiterentwicklungen des k-Mean Algorithmus. So stammen beispielsweise von [Hamerly u. a.] und [Pelleg u. a.] die Algorithmen *g-Means* beziehungsweise *x-Means*, welche die Clusteranzahl k auf Basis mehrerer k-Means Durchläufe und statistischer Kennzahlen bestimmen.

Distributions-Modelle

Distributions-Cluster-Modelle basieren auf der Verwendung von statistischen Wahrscheinlichkeitsverteilungen wie beispielsweise der Gauß-Verteilung. Cluster werden darüber definiert, wie wahrscheinlich es ist, dass Objekte der selben Verteilung angehören. Problematisch ist die Verwendung dieser Cluster-Methodik, da sie anfällig für das Problem des “*Overfitting*” ist, wenn die Komplexität der verwendeten Modelle nicht beschränkt wird. Zudem ist die Annahme, dass vielen realen Datensätzen ein statistisches Verteilungsmodell zugrundeliegt, gefährlich. Ist diese These jedoch berechtigt, haben die Modelle den Vorteil, dass sie neben der Zuweisung von Objekten zu Clustern auch Korrelationen zwischen einzelnen Attributen aufzeigen können. [Anders Drachen, 2014]

Nachfolgend wird der bekannteste Vertreter der Distributions-Cluster-Algorithmen vorgestellt: das *Expectation–maximization* (EM) Verfahren unter Verwendung sogenannter *Gaussian-Mixture-Models* (GMM). Die Funktionsweise des EM-Algorithmus hat grundsätzlich viel gemein mit der des k-Mean Ansatzes. Es wird ebenfalls mit einer festen Anzahl zufällig initialisierter Modelle gestartet, welche anschließend über mehrere Iterationen an die Daten angepasst werden. Im Gegensatz zu k-Means, sind die gewählten Modelle hingegen Gauß-Verteilungen, welche zwei Parameter besitzen: ihren Mittelwert und die Standardabweichung. Das Vorgehen des EM-Algorithmus ist nachfolgend, basierend auf [George Seif, 2018], beschrieben und in Abbildung 2.6 grafisch dargestellt.

- 1) Wahl der Clusteranzahl k und Initialisierung der Gauß-Modelle für die entsprechenden Cluster.
- 2) Berechnung der Wahrscheinlichkeit, dass ein Datenpunkt zu einem Cluster gehört. Je näher ein Datenpunkt dem Zentrum einer Gauß-Verteilung ist, desto höher die Wahrscheinlichkeit für dessen Zugehörigkeit.
- 3) Basierend auf den Wahrscheinlichkeiten werden die Parameter der Verteilungen neu berechnet. Hierzu wird die gewichtete Summe der Datenpunkt-Positionen errechnet. Die Gewichte entsprechen dabei den Wahrscheinlichkeiten, dass ein Element zu einem Cluster gehört. Hierdurch werden die Gauß-Modelle automatisch den in den Daten enthaltenen Clustern angepasst.
- 4) Wiederholung der Schritte 2) und 3), bis das Clustering-Ergebnis konvergiert.

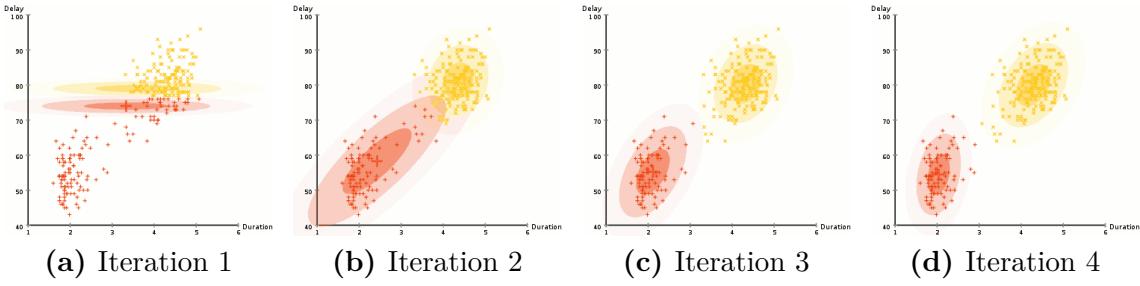


Abb. 2.6: Darstellung des EM-Cluster-Algorithmus über mehrere Iterationen [George Seif, 2018]

Ziel des EM-Algorithmus ist es, die Parameter der Gauß-Modelle so zu optimieren, dass diese die Verteilung der Daten bestmöglich beschreiben. Am Ende des Clusterings besitzt jeder Datenwert die Zugehörigkeits-Wahrscheinlichkeiten für die einzelnen Cluster. Ein Element wird jenem Cluster zugeordnet, für welches es die höchste Wahrscheinlichkeit besitzt.

Dichte-Modelle

Dichte basierte Cluster sind, wie oben beschrieben, definiert als Regionen hoher Objekt-Dichte, welche von Bereichen geringer Dichte umgeben sind. Dichte-Clustering Modelle suchen nach eben solchen Regionen. Großer Vorteil der Algorithmen dieser Klasse ist, dass sie Cluster beliebiger Formen finden können, nicht auf die Vorgabe einer Clusteranzahl angewiesen sind und mit Ausreißern umgehen können.

Als Vertreter der Dichte-basierten Ansätze wird nachfolgend der *DBSCAN* Algorithmus (*Density-Based Spatial Clustering of Applications with Noise*), wie in [Gao, 2012] beschrieben, vorgestellt. Er verwendet als Maß für die Dichte einer Region die sogenannte ϵ -Nachbarschaft (*Eps*). Diese selektiert für ein Objekt p alle Objekte, welche innerhalb des Radius ϵ um dieses liegen:

$$N_\epsilon(p) = \{q \mid dist(p,q) \leq \epsilon\} \quad (2.6)$$

Eine ϵ -Nachbarschaft besitzt eine hohe Dichte, wenn in ihr mindestens *MinPts* Objekte liegen.

Basierend auf der Definition von *Eps*, werden die in einem Datensatz vorhandenen Elemente in drei Klassen unterteilt. Sie sind entweder *Kern-*, *Rand-* oder *Ausreißer-* Objekte. Ein Kernobjekt hat mindestens *MinPts* andere Punkte in *Eps*. Randobjekte besitzen weniger als *MinPts* in *Eps*, liegen aber in der Nachbarschaft eines Kernobjektes. Ausreißerobjekte sind weder Kern- noch Randobjekte.

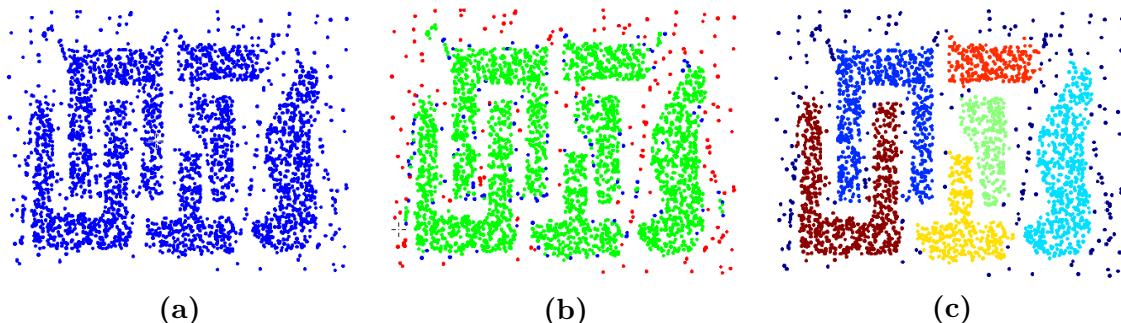


Abb. 2.7: Schritte des DBSCAN Algorithmus, a) Rohdaten, b) Klassifizierung in Kern- (grün), Rand- (blau) und Ausreißer- (rot) Punkte, c) Cluster Ergebnis [Gao, 2012]

Auf Basis der drei Objektklassen, lässt sich das Prinzip der dichte-basierten *Erreichbarkeit* definieren. Ein Objekt q ist von p *direkt* erreichbar, wenn p ein Kernobjekt ist und q in dessen Eps liegt. In Abbildung 2.8 gilt dies beispielsweise für p und p_2 . Zwei Elemente sind *indirekt* erreichbar, wenn sie über eine Reihe von Zwischenschritten (direkte Relationen) verbunden sind (transitiv). Dies ist in Abbildung 2.8 für q und p der Fall.

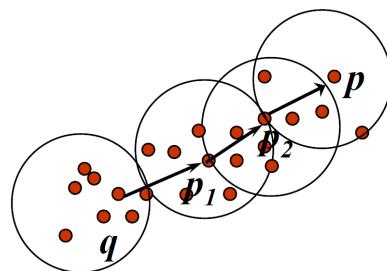


Abb. 2.8: Erreichbarkeit in DBSCAN [Gao, 2012]

Der DBSCAN Algorithmus lässt sich, basierend auf den obigen Definitionen, informell wie folgt beschreiben:

- 1) Unterteilung der Objekte in die drei Objektklassen. (Abb. 2.7 b))
 - 2) Aussortierung der Ausreißer-Objekte.
 - 3) Wahl eines nicht zugewiesenen Kernobjektes.
 - 4) Erstellung eines neuen Clusters für das Kernobjekt und alle von ihm ausgehend direkt oder indirekt erreichbaren Objekte
 - 5) Wiederholung der Schritte 3) und 4), bis alle Kern- und Randobjekte einem Cluster zugewiesen sind. (Abb. 2.7 c))

DBSCAN besitzt die oben beschriebenen Vorteile Dichte-basierter Cluster-Algorithmen. Dank einer Zeitkomplexität von $O(n \log n)$ kann er außerdem auch auf große Datensätze angewendet werden. Nachteil des Ansatzes ist hingegen, dass er schlecht mit Clustern umgehen kann, welche unterschiedliche Dichten besitzen.

2.3 Distanzmaße zum Vergleich von Fahrzeugtrajektorien

Bei der Clusteranalyse ist neben der Wahl des passenden Cluster-Algorithmus insbesondere die Entscheidung, welches Distanzmaß verwendet wird, ausschlaggebend. Im obigen Abschnitt wurden bereits die euklidische Distanz (Gleichung 2.1) als ein mögliches Distanzmaß definiert. Dieses kann jedoch nur zur Bestimmung der Distanz zwischen n -dimensionalen Punkten im euklidischen Raum verwendet werden. Dies gilt ebenso für andere einfache Maße wie die Manhatten-Distanz oder die Pearson-Distanz.

Um Fahrzeugtrajektorien korrekt gruppieren zu können, ist ein Distanzmaß notwendig, welches je nach Anforderungen die unterschiedlichen Aspekte der Trajektorien vergleicht. Häufig werden die Eigenschaften Lage, Form und Länge hierzu herangezogen. In der Literatur werden diverse Maße zum Vergleich von Trajektorien vorgestellt. Diese besitzen alle unterschiedliche Eigenschaften, Vor- und Nachteile.

Nachfolgend werden exemplarisch drei Distanzmaße vorgestellt, anhand welcher ersichtlich ist, welche Abwägungen bei der Wahl des Maßes gemacht werden müssen. In allen drei Fällen werden die Trajektorien als Reihen 2-dimensionaler Punkte mit Länge n interpretiert: $t_i = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$. Der n -te Punkt einer Trajektorie ist gegeben über $t_i(n)$ und deren Punkt-Länge über $\text{len}(t_i)$. Die Menge der zu vergleichenden Trajektorien ist $T = \{t_1, t_2, \dots, t_m\}$. Abbildung 2.9 zeigt eine Auswahl möglicher Trajektorien.

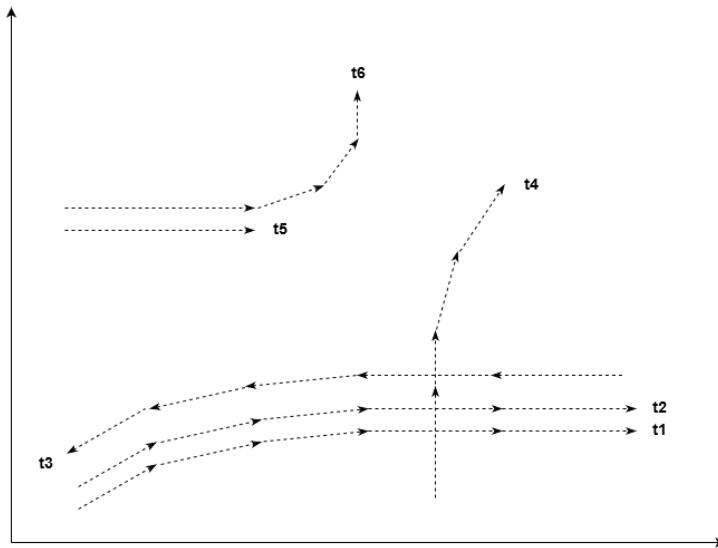


Abb. 2.9: Trajektorien im 2-dimensionalen Raum

2.3.1 HU Distanz

Die HU Distanz wurde erstmals in der Arbeit “*Similarity based vehicle trajectory clustering and anomaly detection*” von [Hu u. a., 2005] verwendet. Es ist ein sehr einfaches Distanzmaß, welches auf der mittleren euklidischen Distanz zwischen zwei Trajektorien basiert. Berechnet wird die HU Distanz für zwei Trajektorien t_1 und t_2 wie folgt:

$$D_{HU}(t_1, t_2) = \frac{1}{N} \sum_{n=1}^N dist(t_1(n), t_2(n)) \quad (2.7)$$

$$\text{wobei } N = \min(\text{len}(t_1), \text{len}(t_2)) \quad (2.8)$$

Aus dieser Formel lassen sich sowohl die Vor- als auch Nachteile der HU Distanz ableiten. Der klare Vorteile der HU Distanz ist deren Einfachheit und die Effizienz von $O(n)$. Nachteil ist hingegen, dass das Distanzmaß nur gut funktioniert, wenn die Trajektorien bestimmte Kriterien erfüllen. So sollten Trajektorien, welche einem Cluster angehören, auch immer möglichst auf selber Höhe beginnen, damit deren mittlerer Abstand nicht, aufgrund einer Verschiebung, erhöht wird. Außerdem ist es notwendig, die Abstände zwischen den Punkten der Trajektorien auf die selbe Länge zu bringen, damit beim paarweisen Vergleich immer Elemente verglichen werden, welche gleichweit vom Start der Spuren entfernt sind. Diese Eigenschaften der Trajektorien müssen über einen Vorverarbeitungsschritt geschaffen werden. Problematisch bei der Verwendung der HU Distanz ist außerdem, dass beim Vergleich zweier Trajektorien immer nur die ersten N Punkte (s. Gleichung 2.8) betrachtet werden. Dies kann dazu führen, dass zwei Trajektorien, welche zu Beginn fast identisch sind und später auseinanderlaufen, trotzdem einen hohen Ähnlichkeitswert besitzen (siehe t_5 und t_6 in Abbildung 2.9).

Die HU Distanz kann aufgrund der genannten Einschränken nur in speziellen Fällen oder unter Verwendung eines Vorverarbeitungsschrittes angewandt werden. Sie liefert ansonsten schlechte Clustering Ergebnisse.

2.3.2 Hausdorff Distanz

Die Hausdorff Distanz ist ein komplexeres Maß zur Bestimmung der Ähnlichkeit zwischen zwei Trajektorien. Sie misst grundsätzlich den Abstand zwischen zwei nicht-leeren, ungeordneten Teilmengen A und B und ist für Trajektorien definiert über die Gleichungen [Atev u. a., 2010]:

$$D_{HD}(t_1, t_2) = \max(h(t_1, t_2), h(t_2, t_1)) \quad (2.9)$$

$$h(t_1, t_2) = \max_{i \in t_1} \min_{j \in t_2} dist(i, j) \quad (2.10)$$

$h(t_1, t_2)$ wird als gerichtete Hausdorff Distanz *von* t_1 *nach* t_2 bezeichnet. Sie findet die maximale Distanz einer Trajektorie zum nächsten Punkt der anderen Trajektorie [Huttenlocher u. a., 1993]. Da h gerichtet ist, gilt $h(t_1, t_2) \neq h(t_2, t_1)$. Aus diesem Grund wird die Hausdorff Distanz *zwischen* zwei Trajektorien mittels D_{HD} bestimmt. $dist$ kann ein beliebiges Maß für die Distanz zweier Punkte sein, wie beispielsweise die euklidische Distanz. Grundsätzlich lässt sich über die Hausdorff Distanz die Form zweier Trajektorien vergleichen. Diese sind ähnlich, wenn jeder Punkt einer Trajektorie einen nahegelegenen Punkt in der Vergleichsbahn besitzt.

Vorteil der Hausdorff Distanz im Vergleich zur HU Distanz ist, dass diese immer vollständige Trajektorien vergleicht und nicht nur Teile. Außerdem ist bei ihrer Verwendung keine Vorverarbeitung in Form von Resampling et cetera notwendig. Problematisch ist das Distanzmaß hingegen, da es mit ungeordneten Sets arbeitet und somit im Fall von Trajektorien, deren Orientierung nicht beachtet. Zwei parallel aber in entgegengesetzte Richtungen laufende Trajektorien, wie beispielsweise die Trajektorien t_2 und t_3 in Abbildung 2.9, würden nach Hausdorff daher eine hohe Ähnlichkeit besitzen. Zudem kann das Distanzmaß schlecht mit Ausreißern umgehen, da bereits ein einzelner dieser Punkte, bei ansonsten identischen Trajektorien, zu einer beliebig kleinen Ähnlichkeit führen kann. Von Nachteil ist auch, dass die Zeitkomplexität der Hausdorff-Distanz bei $O(n m)$ liegt.

2.3.3 Longest-Common-Subsequence

Das *Longest-Common-Subsequence* (LCSS) Distanzmaß basiert auf dem allgemeinen Problem der Findung einer längsten gemeinsamen Subsequenz zwischen zwei Sequenzen. Da Trajektorien, nach obiger Definition, lediglich Punktfolgen sind, lässt sich das Verfahren sehr gut auf diese anwenden. Aufgrund einiger kleiner Erweiterungen des BasisAlgorithmus, besitzt das LCSS Distanzmaß einige besondere Eigenschaften. Der LCSS Algorithmus für Trajektorien ist grundsätzlich wie folgt definiert [Vlachos u. a., 2002]:

$$LCSS_{\epsilon, \delta}(t_1, t_2) = \begin{cases} 0 & \text{if } t_1 \text{ or } t_2 \in \emptyset \\ 1 + LCSS_{\epsilon, \delta}(t'_1, t'_2) & \text{if } dist(t_1(n), t_2(m)) < \epsilon \\ & \wedge |n - m| \leq \delta \\ max(LCSS_{\epsilon, \delta}(t'_1, t_2), LCSS_{\epsilon, \delta}(t_1, t'_2)) & \text{otherwise} \end{cases} \quad (2.11)$$

Hierbei gilt $t'_1 = \{t_1(0), \dots, t_1(n-1)\}$. Die Parameter ϵ und δ bestimmen das Vergleichs-Verhalten des Algorithmus. Über ϵ wird definiert, wieweit zwei Punkte maximal voneinander entfernt liegen können, um immer noch als “übereinstimmend” zu gelten. δ bestimmt hingegen, wieweit man in der Zeit vor beziehungsweise zurück gehen kann, um einen übereinstimmenden Punkt zu finden. Abbildung XXX veranschaulicht die Bedeutung von ϵ und δ . Da die obige LCSS Funktion nur ein diskretes Zählmaß definiert, ist das eigentliche LCSS-Distanz üblicherweise gegeben als [Vlachos u. a., 2002]:

$$D_{LCSS}(\delta, \epsilon, t_1, t_2) = 1 - \frac{LCSS_{\delta, \epsilon}(t_1, t_2)}{\min(\text{len}(t_1), \text{len}(t_2))} \quad (2.12)$$

Vorteile der LCSS Ähnlichkeitdefinition sind, dass sie mit kompletten Trajektorien arbeitet und robust gegenüber Ausreißern ist, da nicht für alle Punkte Übereinstimmungen in den Trajektorien gefunden werden müssen. Über ϵ und δ kann die "Strenge" des Algorithmus geregelt werden. Zudem berücksichtigt das LCSS Maß die Orientierung der Trajektorien, solange δ nicht zu hoch gewählt wird. Die rekursive Definition des LCSS Algorithmus aus Gleichung 6.8 lässt sich mittels dynamischer Programmierung mit Zeitkomplexität $O(n m)$ berechnen.

Übersicht Distanzmaße

Anhand der drei ausgewählten und oben exemplarisch beschriebenen Distanzmaße, ist bereits ersichtlich, dass die Wahl eines passenden Maßes nicht trivial ist. Es muss die Qualität und Form der Daten berücksichtigt werden und abgewogen werden, in wieweit es möglich beziehungsweise gewünscht ist, die Daten vorzuverarbeiten. Das primäre Auswahlkriterium ist allerdings natürlich die situationsabhängige Definition von "Distanz": Sind sich Trajektorien ähnlich, wenn sie lediglich die selbe Form haben und ansonsten irgendwo im Raum liegen? Sind sie sich ähnlich, wenn sie die selbe Form haben und im Raum nahe beieinander liegen? Ist ihre Orientierung relevant? Dies sind wichtige Fragen, welche vor der Wahl eines Distanzmaßes geklärt werden müssen. Da die Maße als Distanzfunktionen bei der Clusteranalyse verwendet werden, ist ihr Verhalten ausschlaggebend für den Erfolg der Gruppierung von Trajektorien.

Wichtige Eigenschaften einiger in der Literatur häufig verwendeten Vergleichsmaße, inklusive der drei oben beschriebenen, sind nachfolgend nochmals in tabellarischer Form festgehalten.

Tab. 2.1: Eigenschaften verschiedener Distanzmaße

Distanzmaß	gerichtet	PreProc. nötig	Ausreißer-resistant
HU [Hu u. a., 2005]	✓	✓	✓
PCA [Bashir u. a., 2003]	✓	✓	✓
DTW [Keogh und Pazzani, 2000]	✓	✗	✗
HD [Chen u. a., 2011]	✗	✗	✗
mod. HD [Atev u. a., 2006]	✓	✗	✓
PF [Piciarelli und Foresti, 2006]	✓	✗	✗
LCSS [Vlachos u. a., 2002]	✓	✗	✓

3 Verwandte Arbeiten

Das folgende Kapitel gibt eine Überblick über einige wichtige und interessante wissenschaftliche Arbeiten, welche sich mit der Analyse von Trajektoriedaten und insbesondere Fahrzeugtrajektorien beschäftigen. Zu Beginn werden diverse Arbeiten vorgestellt, welche sich mit der Clusteranalyse von Trajektorien befassen. Anschließend wird betrachtet, wie in der Literatur die Erkennung von Fahrspuren, vorzugsweise auf Basis von Trajektorien, umgesetzt wird. Am Ende des Kapitels werden Defizite der existierenden Lösungen festgehalten und analysiert, welche spezifischen Neuerungen für die Umsetzung dieser Arbeit nötig sind.

3.1 Clusteranalyse von Trajektorien

Aufgrund der großen Menge an Informationen, welche sich auf Basis von Trajektoriedaten ermitteln lassen, ist ihre Analyse schon seit geraumer Zeit Gegenstand wissenschaftlicher Untersuchungen. Nachfolgend werden einige Arbeiten vorgestellt, welche sich mit der Clusteranalyse von Trajektorien beschäftigen. Die Auswahl zeigt prototypisch, wie unterschiedliche die Anwendungsszenarien und Ziele bei solchen Analyse sind.

Similarity based vehicle trajectory clustering and anomaly detection

Eine Arbeit, welche ein sehr typisches Anwendungszenario behandelt, stammt von [Hu u. a., 2005]. Die Autoren beschreiben in dieser Veröffentlichung ein Verfahren zur Clusteranalyse von Fahrzeugtrajektorien. Ziel dieser ist es, auf Basis der entdeckten Spur-Cluster, anormale Verkehrsmanöver in Live-Aufnahmen von Straßenabschnitten detektieren zu können. Solche Manöver sind beispielsweise “Fahren abseits der üblichen Bahnen” oder “zu schnelles/langsames Fahren”. Die Fahrzeugtrajektorien sind in dieser Arbeit als Sequenzen zweidimensionaler Punkte definiert. Um sie zu gruppieren, setzen Hu et al. auf klassische Clusterverfahren und die Verwendung eines einfachen, metrischen Distanzmaßes. Dieses Maß, bekannt als HU Distanz (siehe Abschnitt 2.3.1), vergleicht Trajektorien über den mittleren Abstand zwischen zusammengehörigen Punktpaaren. Da dies nur zuverlässig möglich ist, wenn die Trajektorien einige Bedingungen erfüllen, müssen die Autoren diese vorverarbeiten. Sie vereinheitlichen daher die Abstände der Punkte einer Trajektorie und erweitern sie zudem in Richtung der Szenen-Grenzen.

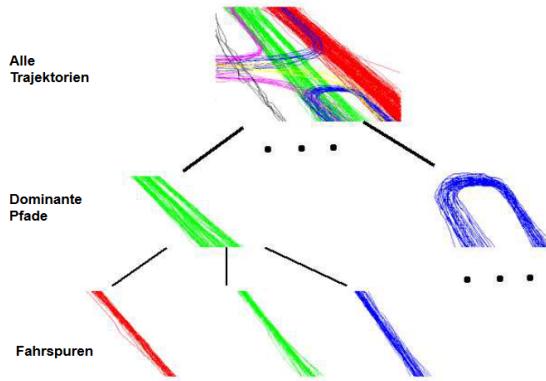


Abb. 3.1: Zweistufiger Clustering-Vorgang von Hu et al. [Hu u. a., 2005]

Unter Verwendung des definierten Distanzmaßes werden die Trajektorien in einem zweistufigen Verfahren verarbeitet. In den zwei Phasen werden, wie in Abbildung 3.1 dargestellt, zuerst dominante Fahrpfade extrahiert, welche anschließend weiter in einzelne Fahrspuren untergliedert werden. Als eigentliche Cluster-Algorithmen vergleichen die Autoren den *Spectral-Clustering* Ansatz [Ng u. a., 2002] mit einem *Fuzzy-k-Means* Verfahren [Xie und Beni, 1991]. Die Untersuchungen zeigen, dass der Spectral Clustering Ansatz nicht nur bessere Ergebnisse liefert, sondern dieser über mehrere Durchläufe hinweg auch stabil sind, wohingegen die Resultate des Fuzzy-Ansatzes variieren.

Multi Feature Path Modeling for Video Surveillance

Eine weitere Arbeit welche das Ziel hat, anormale Bewegungsmuster auf Basis von Trajektorien zu entdecken, stammt von [Junejo u. a., 2004]. In diesem Fall geht es den Autoren allerdings nicht um das Finden von Fahrzeug-Fahrspuren, sondern um die Extraktion von Laufpfaden von Fußgängern. Die Bewegungsbahnen der Passanten werden aus Aufnahmen stationärer Überwachungskameras gewonnen und als zwei-dimensionale Punktreihen repräsentiert. Um die Trajektorien zu vergleichen, verwenden die Autoren die Hausdorff Distanz als Distanzmaß. Die üblicherweise negativen Eigenschaften dieses Vergleichskriteriums (siehe Abschnitt 2.3.2), konkret die Missachtung der Trajektorie-Orientierung, sind bei diesem Anwendungsfall kein Nachteil, sondern gewünscht. Da Fußgänger auf einem Pfad oder Weg in entgegengesetzte Richtungen gehen können, muss die Orientierung ihrer Trajektorien ignoriert werden. Auf Basis der Hausdorff Distanz erstellen Junejo et al. einen vollständigen Graphen, in welchem die Knoten Trajektorien und die gewichteten Kanten den Distanzen zwischen Trajektorien entsprechen. Sie zerlegen diesen Graphen mit Hilfe eines rekursiven *min-cut*-Graphen-Algorithmus, welcher sich an der Arbeit von [Boykov und Kolmogorov, 2004] orientiert, und erhalten so die Cluster für die extrahierten Fußgänger-Trajektorien.

Clustering of Vehicle Trajectories

In der Arbeit [Atev u. a., 2010] ist das Ziel der Autoren, ein Verfahren zu finden, mit welchem Fahrzeugtrajektorien bestmöglich gruppiert werden können, ohne diese im Voraus anzupassen zu müssen. Sie vergleichen hierzu die Performance von drei unterschiedlicher Distanzmaße unter Verwendung von zwei Cluster-Algorithmen. Primäres Augenmerk legen die Autoren auf ein von ihnen bereits in [Atev u. a., 2006] entwickeltes Distanzmaß, welches auf der Hausdorff Distanz basiert und sowohl die Orientierung von Trajektorien berücksichtigt als auch robust gegenüber Ausreißern ist. Dieses neue Maß ist für zwei Trajektorien P und Q wie folgt definiert:

$$h_{\alpha, N, C}(P, Q) = \text{ord}_{p \in P}^{\alpha} \left\{ \min_{q \in N_Q(C_{P, Q}(p))} d(p, q) \right\} \quad (3.1)$$

Hierbei entspricht $C_{P, Q}$ einem Mapping $P \rightarrow Q$, welches einem Punkt $p \in P$ einen entsprechenden Punkt $q \in Q$ zuweist, welcher die selbe relative Position in Q besitzt wie p in P . N_Q definiert ein Subset von Q als Nachbarschaft des Punktes q . Zusammen definieren N_Q und $C_{P, Q}$ eine Struktur, in welcher der Abgleich der Trajektorien stattfindet. Dies ist visuell auch nochmals in Abbildung 3.2 dargestellt. Der Operator $\text{ord}_{p \in P}^{\alpha} f(p)$ selektiert jenen Wert aus $f(p)$, welcher größer ist als α -Prozent der Werte.

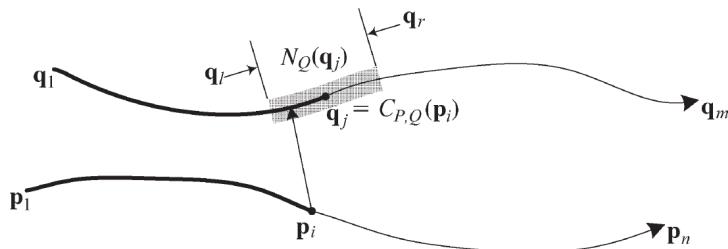


Abb. 3.2: Funktionsweise der modifizierten Hausdorff Distanz [Atev u. a., 2010]

Dank dieser Modifizierungen eignet sich das Distanzmaß gut für den Vergleich von Trajektorien: $C_{P, Q}$ sorgt für den Einbezug der Orientierungen und über die Nachbarschaft N_Q und $\text{ord}_{p \in P}^{\alpha} f(p)$ kann der Einfluss von Ausreißern minimiert werden.

Dieses Distanzmaß vergleichen Atev et al. unter Verwendung eines Spectral und eines Agglomerativen Cluster-Algorithmus mit der *Longest-Common Subsequence* (LCSS) und der *Dynamic Time Warping* (DTW) Distanz. Die Ergebnisse der Untersuchungen für vier verschiedene Datensätze zeigen, dass die beste Cluster-Performance mit Hilfe der modifizierten Hausdorff Distanz und des Spectral-Clustering erreicht wird. Unter Verwendung der LCSS und DTW Distanzmaße, könnten die Autoren nicht die selben Resultate erzielen.

Dass das von Atev et al. vorgeschlagene Distanzmaß sehr gute Clusterergebnisse produziert, wurde auch von [Morris und Trivedi, 2009] bestätigt. In ihrer Untersuchung waren allerdings die Ergebnisse, welche mithilfe des LCSS Maßes erreicht wurden, ebenso gut oder teilweise besser.

Clustering of trajectories based on Hausdorff Distance

Eine weitere interessante Arbeit zur Clusteranalyse von Trajektorien stammt von [Chen u. a., 2011]. Die Autoren haben das Ziel, Muster in den Bewegungsbahnen von Hurrikans, welche im Zeitraum von 1850 bis 2010 über den Atlantik zogen, zu erkennen. Sie verwenden hierzu einen angepassten DBSCAN Cluster-Algorithmus und das Hausdorff Distanzmaß. Um die Missachtung der Orientierung kompensieren zu können, und zudem auch Ähnlichkeiten in Sub-Trajektorien zu erkennen, wählen die Autoren eine etwas andere Darstellung der Trajektorien. Sie definieren eine Bewegungsbahn als eine Folge sogenannter „*Flow-Vektoren*“, welche neben Positions- auch Richtungsinformationen enthalten. Ein solcher Vektor ist definiert über:

$$f_i = (x_i, y_i, dx_i, dy_i) \quad (3.2)$$

wobei gilt:

$$dx_i = (x_{i+1} - x_i)/\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (3.3)$$

$$dy_i = (y_{i+1} - y_i)/\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (3.4)$$

Die Distanz zwischen zwei *Flow-Vektoren* ist ihr euklidischer Abstand. Auf diese Weise wird bei der Berechnung der Hausdorff Distanz (siehe Abschnitt 2.3.2) auch die Richtung der Trajektorien berücksichtigt. Um ähnliche Sub-Trajektorien entdecken zu können, teilen Chen et al. die Trajektorien an den Positionen „charakteristischer“ Vektoren. Diese beschreiben Richtungsänderungen in einer Bewegungsbahn und werden identifiziert über die Abweichungen in den Richtungskomponenten zweier aufeinanderfolgender Flow-Vektoren. Dies ist anschaulich in Abbildung 3.3 dargestellt.

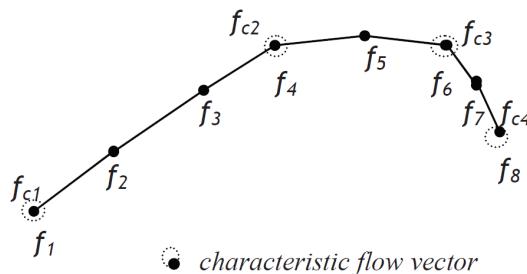


Abb. 3.3: Zerlegung einer Trajektorie in Sub-Trajektorien [Chen u. a., 2011]

Die auf diese Weise erhaltenen Sub-Trajektorien, werden von den Autoren mittels eines DBSCAN Algorithmus gebündelt. Sie können so die üblichen Bewegungsbahnen von Hurrikans über dem Atlantik bestimmen.

Discovering Similar Multidimensional Trajectories

Die Arbeit [Vlachos u. a., 2002] thematisiert nicht direkt die Clusteranalyse von Trajektorien sondern beschäftigt sich mit dem Vergleich von Bewegungsbahnen im drei-dimensionalen Raum. Konkret ist ihr Ziel, Trajektorien vergleichen zu können, welche etwa die Handbewegungen beim Ausführen von Zeichensprache beschreiben. Hierzu definieren die Autoren erstmals die Grundversion des LCSS Distanzmaßes, welches in vielen Arbeiten, unter anderem in [Atev u. a., 2006], [Buzan u. a., 2004] und [Chen u. a., 2005], zum Einsatz kommt. Auf dessen Basis erstellen sie ein Distanzmaß, welche es ermöglicht formgleiche aber im Raum verschobene Trajektorien zu finden. Die Grundversion der LCSS-Distanz und ein darauf basierendes, einfaches Distanzmaß ist, nach Vlachos et al., bereits in Abschnitt 2.3.3 vorgestellt worden. Dieses Maß erweitern die Autoren zudem wie folgt:

$$D_{LCSS}(\delta, \epsilon, A, B) = 1 - \max_{f_{c,d} \in F} D_{LCSS}(\delta, \epsilon, A, f_{c,d}(B)) \quad (3.5)$$

Hierbei ist F eine Menge von Translations-Funktionen, welche die Trajektorien entlang der Achsen verschieben. Sie besitzen die Form

$$f_{c,d}(A) = ((a_{x,1} + c, a_{y,1} + d), \dots, (a_{x,n} + c, a_{y,n} + d)) \quad (3.6)$$

Abbildung 3.4 veranschaulicht die Funktionsweise des Distanzmaßes. Es eignet sich immer dann, wenn Trajektorien mit ähnlicher Form gefunden werden sollen, welche zudem eine gewisse räumliche Verschiebung aufweisen können. Diese kann über die Größe von F gesteuert werden.

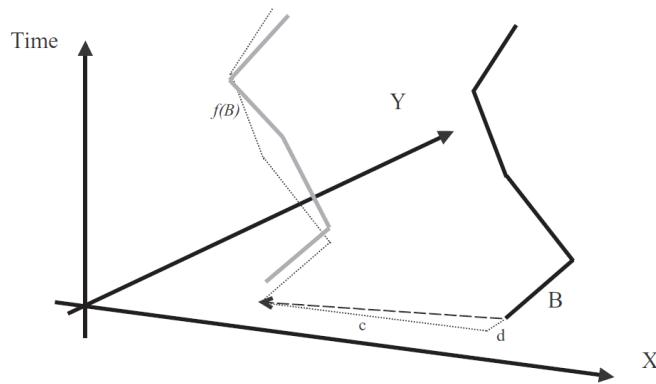


Abb. 3.4: Verschiebung einer Trajektorie im Raum [Vlachos u. a., 2002]

Lane Detection in Video-Based Intelligent Transportation Monitoring via Fast Extracting and Clustering of Vehicle Motion Trajectories

[Ren u. a., 2014] stellen in ihrer Arbeit ein interessantes Vorgehen zur Clusteranalyse von Trajektorien vor, welches auf der *Rough-Set-Theorie* beruht. Sie extrahieren Fahrzeugpositionen aus Aufnahmen stationärer Überwachungskameras und stellen diese, wie die meisten Autoren, als Sequenzen zwei-dimensionaler Punkte dar. Ihr Ziel ist anschließend, anhand einer Gruppierung der gewonnenen Fahrzeugtrajektorien, die Spurmittelpunkte der Fahrbahnen zu bestimmen. Da ein nicht unerheblicher Anteil der Trajektorien Spurwechselvorgänge enthält, welche eine Extraktion der Mittellinien erschweren, verwenden Ren et al. einen iterativen *Rough-k-Means* Algorithmus zur Gruppierung der Trajektorien. Hierbei wird jedes Cluster über eine obere und untere Approximation beschrieben. Die untere Approximation enthält dabei die Trajektorien, welche eindeutig der Spur zugeordnet werden können. Die obere Näherung hingegen auch jene, welche Spurwechsel et cetera beschreiben. Bei der Berechnung der Spurmitten, werden die Trajektorien der unteren Approximation höher gewichtet, als die der oberen. Ein sehr ähnlicher Cluster-Ansatz wurde bereits in [Lingras u. a., 2004] vorgestellt. Die initialen Mittellinien bestimmen die Autoren anhand einer *Aktivitäts-* oder *Heat-Map*, welche sie während der Extraktion der Fahrzeugpositionen erstellen. Die Clusteranzahl k muss händisch definiert werden.

Als Maß für die Distanz zwischen einer Trajektorie A_x und einer Spurmitte c_i verwenden Ren et al. die Hausdorff Distanz $h(A_x, c_i)$. Für eine Trajektorie wird somit die nächste Mittellinie wie folgt gefunden:

$$h(A_x, c_m) = \min_{i=1\dots k} h(A_x, c_i) \quad (3.7)$$

Hieraus ergibt sich die nachfolgende Definition für die Zuordnung der Bewegungsbahnen zu den Cluster-Näherungen:

$$\begin{cases} A_x \in \overline{C_m} \wedge A_x \in \overline{C_j} & \text{if } j \neq m \wedge \frac{h(A_x, c_j)}{h(A_x, c_m)} \leq \lambda \\ A_x \in \underline{C_m} & \text{otherwise} \end{cases} \quad (3.8)$$

Für den Grenzwert λ gilt $1 \leq \lambda \leq 1.5$. $\overline{C_m}$ und $\underline{C_m}$ entsprechen der oberen und unteren Näherung des m -ten Clusters und $\underline{C_m} \subseteq \overline{C_m}$. Nachdem in jeder Iteration des Cluster-Vorgangs die Näherungen auf diese Weise bestimmt wurden, werden die neuen Mittellinien anhand Gleichung 3.9 errechnet.

$$c_i = \begin{cases} \frac{w_l \sum_{A_x \in \underline{C_i}} A_x}{|\underline{C_i}|} + \frac{(1-w_l) \sum_{A_x \in (\overline{C_i} - \underline{C_i})} A_x}{|\overline{C_i} - \underline{C_i}|} & \text{if } \overline{C_i} \neq \underline{C_i} \\ \frac{\sum_{A_x \in \underline{C_i}} A_x}{|\underline{C_i}|} & \text{otherwise} \end{cases} \quad (3.9)$$

Als Gewichtungen w_l verwenden Ren et al. Werte im Bereich $[0.5, 1]$. $|\cdot|$ entspricht hier der Kardinalität einer Menge.

Unter Verwendung dieser Clustering-Methode ist es den Autoren von [Ren u. a., 2014] möglich, auch bei einer hohen Anzahl von Ausreißern und Spurwechselvorgängen, stabile Spurmittellinien zu bestimmen. Ergebnisse, welche dies zeigen, sind in Abbildung 3.5 dargestellt.

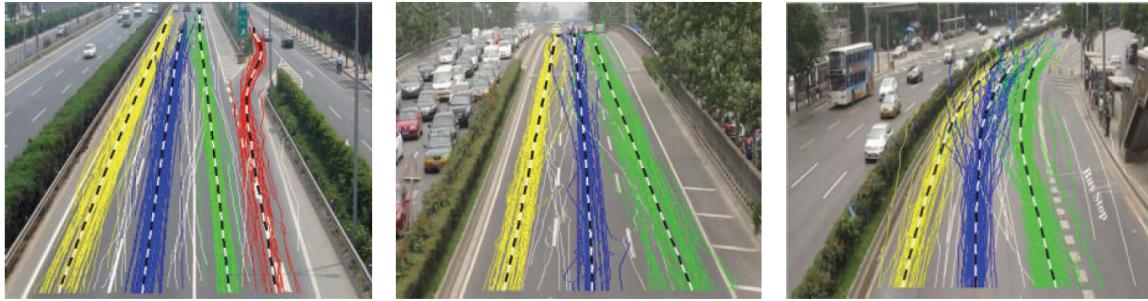


Abb. 3.5: Ergebnisse der Spurmittellinien-Erkennung [Ren u. a., 2014]

3.2 Erkennung und Definition von Fahrspuren

Primäres Ziel dieser Arbeit ist es, Fahrspuren zuverlässig in Videoaufnahmen erkennen zu können. Dieser Abschnitt geht daher auf einige Arbeiten ein, welche ähnliches ebenfalls versuchen.

Die Mehrzahl der Veröffentlichungen in diesem Bereich, löst das Problem, indem sie nach visuellen Merkmalen, primär Spurtrennlinien, in Videoaufnahmen suchen. Die Aufnahmen werden dabei entweder von stationären Kameras, von bemannten oder unbemannten Luftfahrzeugen oder einem Fahrzeug selbst gemacht. Zur Extraktion der Merkmale werden üblicherweise Methoden aus den Gebieten des maschinellen Sehen (CV) oder maschinellen Lernens (ML) verwendet. Arbeiten, welche CV-basierte Ansätze verfolgen, stammen beispielsweise von [Lai und Yung, 2000], [McCall und Trivedi, 2006] oder [Aly, 2008]. ML-gestützte Arbeiten wurden dahingegeng unter anderem von [Kim, 2008] und [Gopalan u. a., 2012] veröffentlicht.

Da oben genannte Ansätze, wie bereits zu Beginn der Arbeit erläutert, aufgrund von Verdeckungen oder Änderungen in der Belichtung problematisch sind, werden nachfolgend ausschließlich Arbeiten vorgestellt, welche Fahrspuren aus Trajektoriedaten extrahieren. Hierbei setzten die meisten als ersten Schritt auf eine Clusteranalyse von Trajektorien. In der Repräsentation und Extraktion der Fahrspuren variieren die Ansätze hingegen.

A System for Learning Statistical Motion Patterns

Die Arbeit [Weiming Hu u. a., 2006] basiert auf dem bereits früher veröffentlichten Artikel [Hu u. a., 2005] der selben Autoren, welcher oben beschrieben wurde. In dieser Publikation gehen Hu et al. nun genauer darauf ein, wie sie auf Basis der Ergebnisse der Clusteranalyse, statistische Informationen über die Fahrbahnen der Fahrzeuge berechnen. Hierzu wird für jedes Cluster zuerst eine Referenz-Trajektorie T_r bestimmt. Dies ist jene Bewegungsbahn, bei welcher die Summe der Distanzen zu allen anderen Trajektorien des Clusters minimal ist. Das verwendete Distanzmaß ist hierbei das selbe, welches auch bei der Clusteranalyse zum Einsatz kam. Die Autoren berechnen anschließend für jede Trajektorie-Gruppe eine Kette Gausscher-Wahrscheinlichkeits-Verteilungen $\{\varphi_1, \varphi_2, \dots, \varphi_l\}$, wobei l die Anzahl der Punkte der Referenz-Trajektorie ist. Für jedes φ_i wird der Mittelwert und die Kovarianz, basierend auf den Trajektorie-Punkten, welche dem i -ten Punkt von T_r am nächsten liegen, berechnet. Anhand der Kovarianz Werte erstellen die Autoren Hüllen für die Fahrbahnen. Ergebnisse dieses Vorgehens sind in Abbildung 3.6 dargestellt.

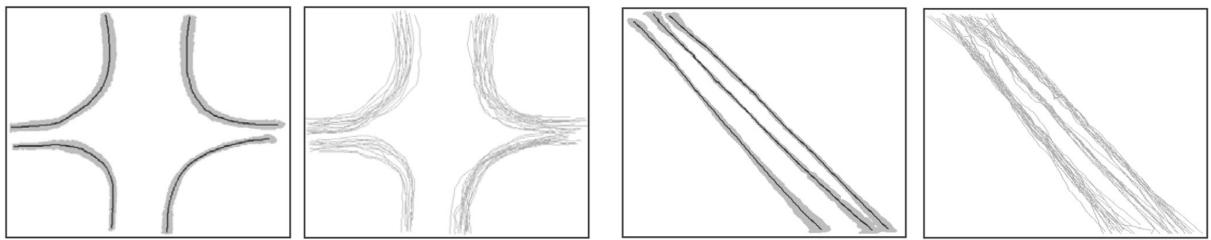


Abb. 3.6: Spurhüllen in [Weiming Hu u. a., 2006]

Anzumerken ist, dass die extrahierten Spurhüllen nicht die tatsächliche Form, insbesondere die Breite, einer Fahrspuren wiederspiegeln. Sie sind deutlich schmäler.

Learning Semantic Scene Models From Observing Activity in Visual Surveillance

In [Makris und Ellis, 2005] extrahieren die Autoren übliche Bewegungsbahnen von Fußgängern aus stationären Videoaufnahmen. Sie verwenden hierzu keine klassische Clusteranalyse, sondern ein iteratives und adaptives Online Verfahren, welches neue Bewegungsstrajektorien automatisch bestehenden Routen zuordnet oder neue initialisiert, falls zu hohe Abweichungen zwischen der Trajektorie und den existierenden Bahnen bestehen. Routen definieren Makris et al. hierbei als eine Sequenz von Knoten, welche folgende Merkmale besitzen:

- 2D-Mittelpunkt
- Gewichtung (Anzahl der Trajektorien im Bereich des Knoten)

- Zwei Hüll-Punkte (Maximale- beziehungsweise Standardabweichung der Trajektorien der Route im Bereich des Knoten)

Abbildung 3.6 a) veranschaulicht nochmals die Definition einer Route. Um aus einfachen Bewegungsbahnen Routen zu erstellen, verwenden die Autoren das nachfolgend beschriebene Verfahren, welches dem agglomerativen Cluster-Ansatz ähnelt.

1. Erste Trajektorie initialisiert erste Route
2. Neue Trajektorien werden mit bestehenden Bahnen abgeglichen
 - a) Bei Übereinstimmung wird Route aktualisiert
 - b) Bei keiner Übereinstimmung wird neue Route erzeugt
3. Aktualisierte Routen werden auf definierten Knotenabstand r gebracht
4. Alle Routen werden miteinander verglichen
 - a) Bei Überlagerung der Routen werden diese fusioniert

Als Vergleichsmaß für die Trajektorien und Routen, verwenden Makris et al. die maximale Distanz zwischen einer Trajektorie und einer Routen-Hülle. Diese muss sich unterhalb eines bestimmten Grenzwertes befinden. Ein Ergebnis des Verfahrens ist in Abbildung 3.7 b) dargestellt.

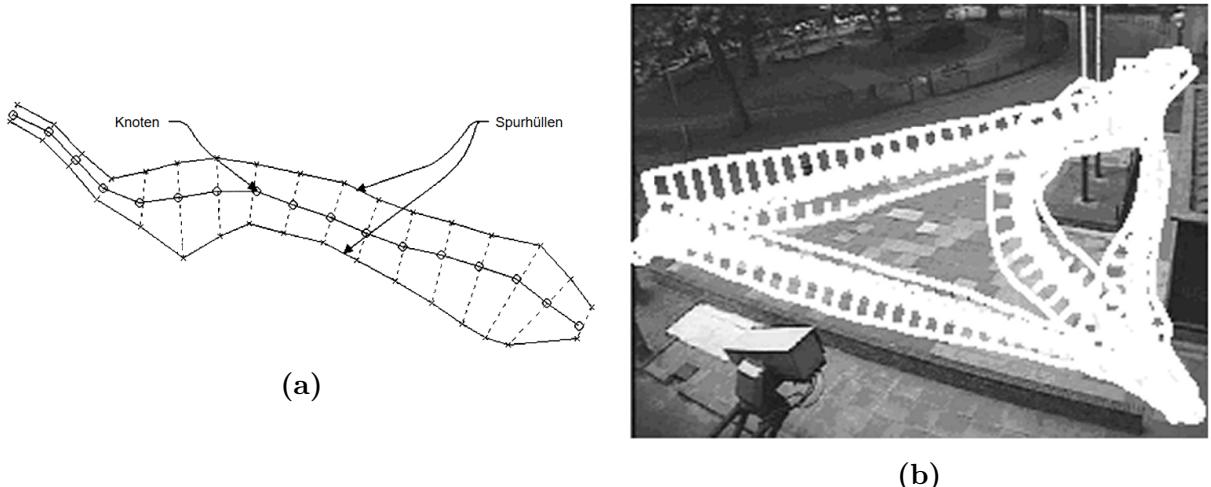


Abb. 3.7: a) Routen Definition, b) Ergebnisse Routen-Erkennung [Makris und Ellis, 2005]

Trajectory Learning for Activity Understanding: Unsupervised, Multilevel, and Long-Term Adaptive Approach

Automatic Traffic Surveillance System for Vehicle Tracking and Classification

[Hsieh u. a., 2006] stellen in ihrer Arbeit ein System zur Extraktion von Fahrzeugpositionen aus Aufnahmen stationärer Verkehrskameras vor. Auf diesen aufbauend definieren sie einen Algorithmus, mit dessen Hilfe es möglich ist, Spurmittel- und Begrenzungs-Linien zu entdecken. Hierzu erzeugen Hsieh et al. ein Histogramm $H_{vehicle}(x,y)$, welches die Häufigkeit abbildet, das ein Fahrzeug sich über eine bestimmte Pixel-Position (x, y) bewegt. Ein solches ist in Abbildung 3.8 a) dargestellt. Die Autoren gehen davon aus, dass die Fahrzeugpositionen sich mehrheitlich in der Mitte einer Spur befinden und ein Histogramm, welches für eine Fahrbahn mit L_n Spuren erstellt wurde, L_n Maxima in jeder Zeile besitzt, welche die Spurmittten darstellen. Sie isolieren daher die Maxima in $H_{vehicle}$ als Mittellinien. Es sei anschließend $C_{L_k}^j$ die k -te Spurmittellinie in Zeile j des Histogramms. Daraus berechnen Hsieh et al. die Spurbegrenzungslinien. Alle innen liegenden Begrenzungen ergeben sich aus Gleichung 3.10. DL_k^j entspricht hierbei einem Punkt in der j -ten Reihe der k -ten Begrenzung.

$$DL_k^j = \frac{1}{2}(C_{L_{k-1}}^j + C_{L_k}^j) \quad (3.10)$$

Die Breite $w_{L_k}^j$ der k -ten Fahrspur ergibt sich zudem wie folgt:

$$w_{L_k}^j = |C_{L_k}^j - C_{L_{k-1}}^j| \quad (3.11)$$

Die Positionen der äußeren Spurbegrenzungen einer Fahrbahn ergeben sich für die j -te Zeile des Histogramms aus den Gleichungen 3.12 und 3.13.

$$(x_{DL_0^j}, j) = (x_{DL_1^j - w_{L_0}^j}, j) \quad (3.12)$$

$$(x_{DL_{N_L}^j}, j) = (x_{DL_{N_{L-1}}^j + w_{L_0}^j}, j) \quad (3.13)$$

Ein Beispiel für die Ergebnisse der Spurerkennung von Hsieh et al. ist in Abbildung 3.8 b) dargestellt. Das Verfahren funktioniert grundlegend gut, wenn Fahrspuren nebeneinander und parallel zueinander liegen.



Abb. 3.8: a) Fahrzeugpositions Histogramm, b) Spurmittellinien und Spurbegrenzungslinien [Hsieh u. a., 2006]

3.3 Defizite vorhandener Lösungen und benötigte Neuerungen

4 Untersuchung möglicher Straßentopologien

4.1 Landstraßen

4.2 Autobahnen

4.3 Kreuzungen

4.4 Kreisverkehre

5 Konzeption des Spurerkennung-Moduls

In diesem Kapitel der Arbeit wird das zu entwickelnde Teilmodul “*Spurerkennung*” der MEC-View *TrackerApplication* konzipiert. Hierzu wird zuerst dessen Rolle und Position im Gesamtkontext der Anwendung betrachtet. Anschließend werden Anforderungen und ein Entwurf des Moduls aufgestellt.

5.1 Überblick über das Gesamtsystem

Das Modul *Spurerkennung* dient der Erreichung der in Abschnitt 1.2 definierten Ziele. Erstellt wird es im Rahmen des MEC-View Teilprojektes *Luftbeobachtung* als Teilmodul der Anwendung *TrackerApplication*. Abbildung 5.1 gibt einen Überblick über das System. Es werden hierbei jene Module beziehungsweise Schritte vorgestellt, welche mit der Spurerkennung in Zusammenhang stehen.

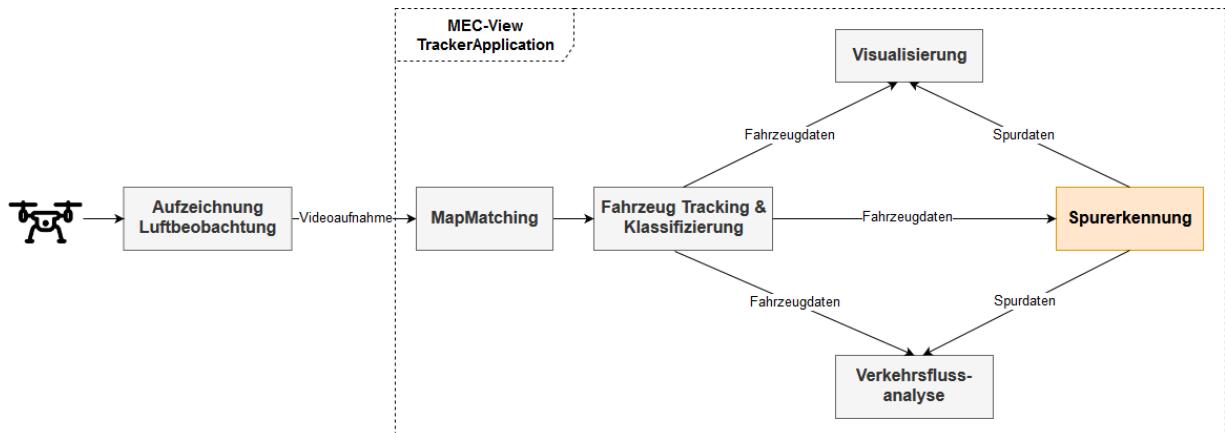


Abb. 5.1: Kontext des Spurerkennung Moduls

Die mithilfe von Drohnen erstellten Videoaufnahmen können in der MEC-View *TrackerApplication* verarbeitet und analysiert werden. In einem ersten Schritt “*MapMatching*”, wird hierzu üblicherweise ein Welt-Koordinatensystem in Metern definiert. Anschließend können die Positionen und Typen der Fahrzeuge bestimmt werden. Diese ersten zwei Schritte, welche der Extraktion von Fahrzeuginformationen dienen, sind in Abschnitt 2.1 genauer beschrieben. Die Fahrzeuginformationen, insbesondere die Positionsinformationen, dienen anschließend dem *Spurerkennung*-Modul als Eingabe. Aus ihnen extrahierte

Spurdaten können anschließend in der Anwendung visualisiert werden oder in Kombination mit den Fahrzeuginformationen zur Analyse des Verkehrsflusses eingesetzt werden.

5.2 Anforderungen an das Modul

In diesem Abschnitt werden die wichtigsten funktionalen und nicht funktionalen Anforderungen des Moduls festgehalten.

5.2.1 Funktionale Anforderungen

Anforderung 1000 (Top-Level) Das *Spurerkennungs*-Modul soll es ermöglichen, mithilfe der *TrackerApplication* automatisch Fahrspuren aus den Positionsinformationen von Fahrzeugen in Luftaufnahmen abzuleiten.

Anforderung 2000 Das Modul soll die Erkennung von Fahrspuren in den in Kapitel 4 vorgestellten Straßentopologien unterstützen.

Anforderung 2100 Das Modul soll unabhängig vom Aufnahmewinkel der Kamera Fahrspuren zuverlässig aus Videoaufnahmen ableiten können.

Anforderung 2200 Das Modul soll Fahrspuren bei Überlagerungen sinnvoll partitionieren können.

Anforderung 2300 Das Modul soll die Enden benachbarter und paralleler Fahrspuren angeleichen. Dies dient ihrer Verwendung in der Verkehrsfluss-Analyse.

Anforderung 2400 Das Modul soll es ermöglichen, die aus den Trajektorien abgeleiteten Fahrspuren in der *TrackingApplication* zu visualisieren.

5.2.2 Nicht funktionale Anforderungen

Anforderung 3000 Das *Spurerkennungs*-Modul muss robust mit Ausreißern und Tracking-Fehlern in den Trajektorien umgehen können.

Anforderung 3100 Die Performance des Spurerkennung-Vorgangs ist nicht von höchster Priorität. Eine Erkennung sollte allerdings dennoch maximal wenige Minuten dauern.

5.3 Entwurf des Moduls Spurerkennung

In diesem Abschnitt wird, basierend auf den Erkenntnissen der Literaturrecherche und den Anforderungen, ein grober Entwurf des *Spurerkennung*-Moduls vorgestellt.

Das Modul definiert primär einen Algorithmus, welcher aus Fahrzeuginformationen wie der Position oder Geschwindigkeit von Fahrzeugen, Fahrspuren ableitet. Die Grundfunktionsweise dieses Algorithmus ist in Abbildung 5.2 in Form eines Aktivitätsdiagramms dargestellt.

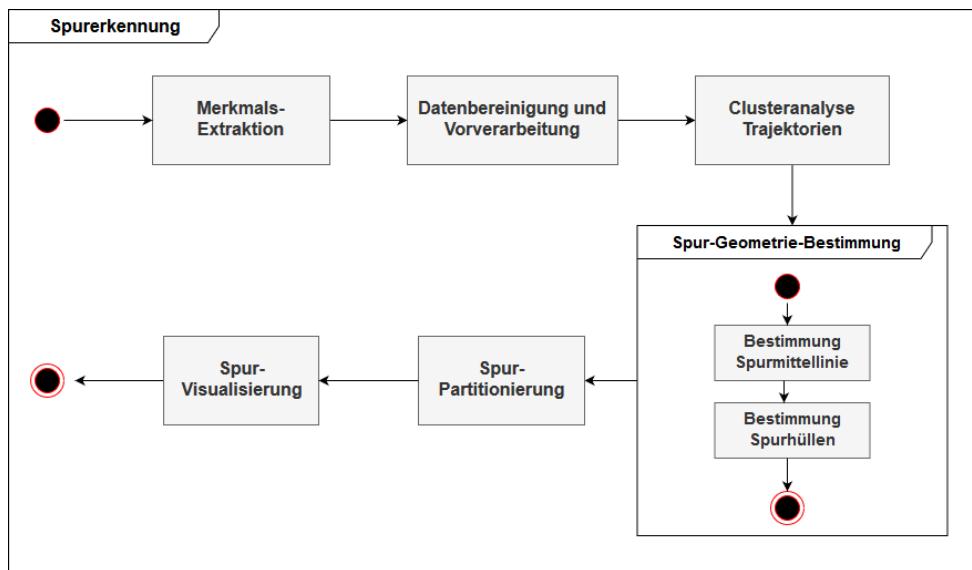


Abb. 5.2: Grundstruktur des Spurerkennungs-Algorithmus

Die einzelnen Schritte des Algorithmus werden im *Spurerkennungs*-Modul der *TrackerApplication* als einzelne Komponenten implementiert.

Die nachfolgenden Umsetzungskapitel beschreiben, wie die einzelnen Schritte des Algorithmus aus Abbildung 5.2 konkret realisiert wurden und welche Probleme es hierbei zu überwinden galt.

6 Clusteranalyse von Fahrzeugtrajektorien

In diesem Kapitel wird die Umsetzung der Clusteranalyse der Trajektorien vorgestellt. Es wird zuerst darauf eingegangen, welche Trajektorie-Repräsentation gewählt wurde. Anschließend werden die verschiedenen Schritte zur Bereinigung und Vorverarbeitung der Daten beschrieben, welche in dieser Arbeit zum Einsatz kommen. Schlussendlich folgt die Erläuterung der eigentlichen Clusteranalyse. Hier werden die verschiedenen untersuchten Ansätze vorgestellt und ihre Ergebnisse diskutiert.

Die *TrackerApplication* ist in Java und Scala implementiert. Ihre Benutzeroberfläche basiert auf JavaFX. Das in dieser Arbeit erstellte Modul *Spurerkennung* wird komplett mit Scala umgesetzt.

6.1 Trajektorie-Definition

Die Ergebnisse der Fahrzeugverfolgung (siehe Abschnitt 2.1) werden in der *TrackingApplication* in Form sogenannter *TrackedObject*'s gespeichert. Ein solches Objekt repräsentiert eine zusammenhängende, nicht unterbrochene Verfolgung eines Fahrzeugs. Wird eine Verfolgung, beispielsweise aufgrund einer Überdeckungen, unterbrochen, so existieren für ein Kraftfahrzeug mehrere Objekte, welche sich einander nicht zuordnen lassen. Die wichtigsten Informationen, die ein *TrackedObject* beinhaltet, sind eine eindeutige ID, die Frame-Positionen des Starts und Endes der Verfolgung und die Objekt-Klasse des Fahrzeugs. Es wird zwischen den vier Klassen “Auto”, “Lastwagen”, “Transporter” und “Zweirad” unterschieden. Für jedes verfolgte Objekt können die zugehörigen Positions-, Geschwindigkeits-, Beschleunigungs- und Dimensions-Informationen abgerufen werden. Diese werden für jedes Frame, welches zwischen dem Start- und End-Frame des Objektes liegt, bestimmt.

Da für die Ableitung von Fahrspuren aus Trajektorien lediglich die positionsbezogenen Eigenschaften der Fahrzeuge relevant sind, werden Bewegungsbahnen in dieser Arbeit über jene definiert. Geschwindigkeit, Beschleunigung und Dimension der Fahrzeuge wird in der Clusteranalyse nicht berücksichtigt. Abbildung 6.1 zeigt den Aufbau einer Trajektorie im Modul *Spurerkennung*.

Trajectory
+ id: Int + objectClass: String + positions: Vector[Point] + pointLength: Int + distanceToStart: Vector[Double] + realLength: Double + clusterLabel: Option[Int]

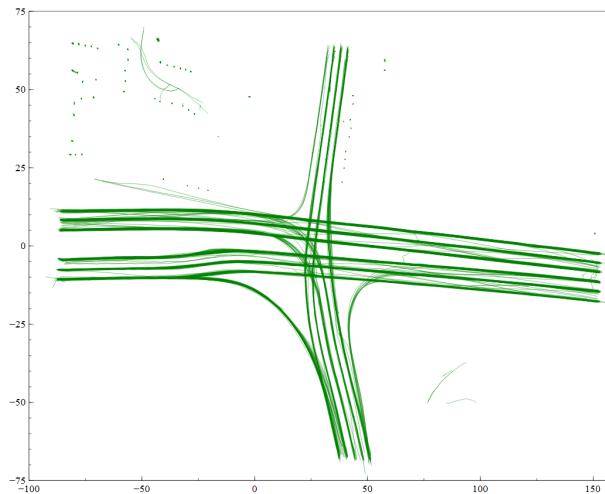
Abb. 6.1: Aufbau Trajektorie-Klasse

Die Felder *id* und *objectClass* werden aus dem der Trajektorie zugrundeliegenden *TrackedObject* übernommen. Die Positionen eines Fahrzeugs werden in Form von 2D-Welt-Koordinaten (siehe Abschnitt 2.1) in *positions* gespeichert und die Anzahl der Koordinaten zusätzlich in *pointLength*. Die Sequenz *distToStart* enthält für jeden Punkt der Bewegungsbahn die Distanz zum Start der Trajektorie in Metern. Die Werte ergeben sich aus Formel 6.1, wobei p_n dem n -ten Punkt in der Trajektorie entspricht und *dist* der euklidischen Distanz zwischen zwei Punkten.

$$distToStart(p_n) = \begin{cases} 0 & \text{if } n = 0 \\ dist(p_n, p_{n-1}) + distToStart(p_{n-1}) & \text{otherwise} \end{cases} \quad (6.1)$$

Aus *distToStart* ergibt sich zudem die Gesamtlänge einer Trajektorie, welche extra gespeichert wird. Das Feld *clusterLabel* ordnet jede Trajektorie nach der Clusteranalyse einem bestimmten Cluster zu. Zuvor enthält es keinen Wert.

Zur Untersuchung der Fahrzeugtrajektorien ist es hilfreich diese zu visualisieren. Abbildung 6.2 zeigt so beispielsweise 1240 Trajektorien, welche aus einer Aufnahme des Stuttgarter Neckartors extrahiert wurden. In Abbildung 6.3 ist ein Ausschnitt der entsprechenden Aufnahme zu sehen.

**Abb. 6.2:** Unverarbeitete Trajektorien vom Stuttgarter Neckartor

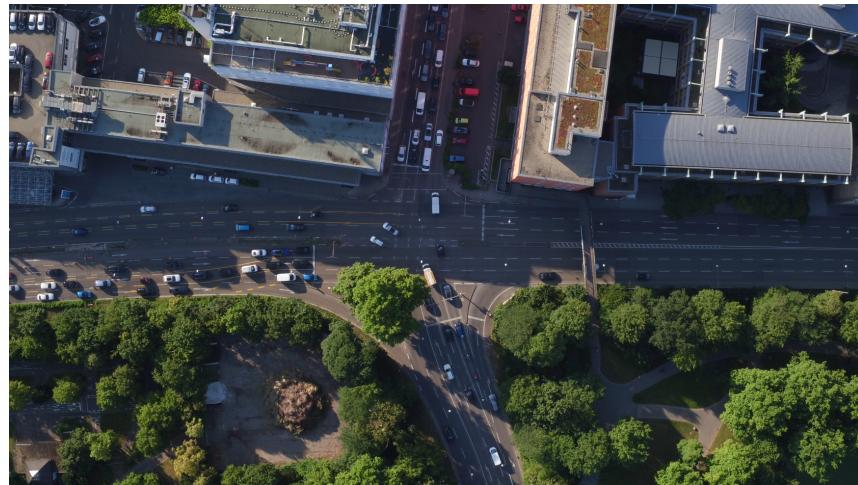


Abb. 6.3: Das Stuttgarter Neckartor

In Abbildung 6.2 sind die verschiedenen Bewegungsbahnen der Fahrzeuge für den menschlichen Betrachter bereits klar erkennbar. Direkt fallen aber auch die Trajektorien der stehenden oder sich auf Parkplätzen bewegenden Autos im oberen Bereich der Aufnahme ins Auge. Diese dürfen nicht in die Clusteranalyse mit einbezogen werden. Bei genauerer Untersuchung der Trajektorien zeigen sich weitere Probleme, welche das Clustering negativ beeinflussen würden. Zwei sind in nachfolgender Abbildung dargestellt. 6.4 a) zeigt, wie Fahrzeuge Punktewolken beim Stillstand vor Lichtsignalanlagen bilden. In 6.4 b) wird deutlich, dass in manchen Bereichen sehr viele Trajektorie-Unterbrechungen auftreten. Hier wird die Straße üblicherweise von Bäumen, Brücken et cetera überlagert.

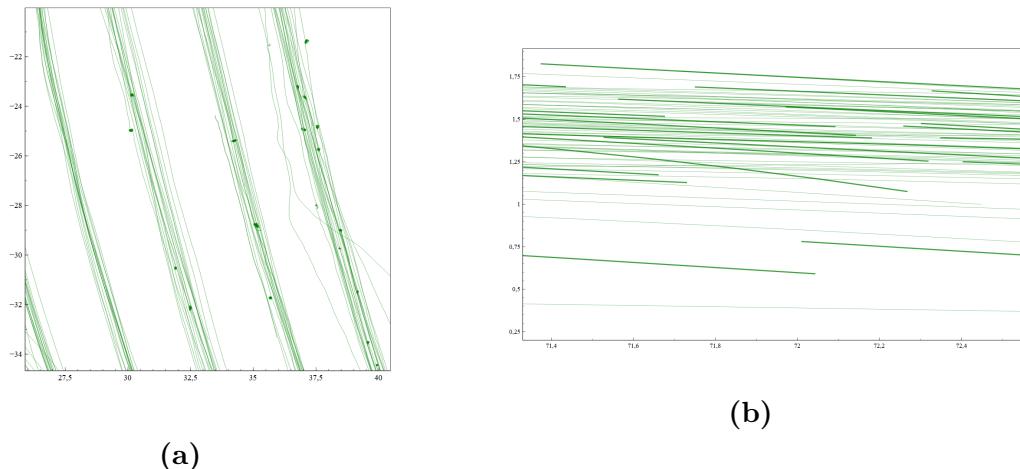


Abb. 6.4: a) Punktewolken vor Lichtsignalanlagen, b) Unterbrechungen aufgrund von Überdeckung

Um von diesen und weiteren Effekten bei der Clusteranalyse nicht beeinflusst zu werden,

durchlaufen die “Roh-Trajektorien” einen Vorverarbeitungsschritt. Dieser wird im nächsten Abschnitt vorgestellt.

6.2 Vorverarbeitung der Trajektorien

Die verschiedenen Schritte, welche zur Vorverarbeitung und Bereinigung der Roh-Trajektorien angewandt werden, sind in diesem Abschnitt geschildert.

Das primäre Ziel der Vorverarbeitung ist es, Ausreißer aus der Trajektorie-Menge zu entfernen, welche das Ergebnis der Clusteranalyse negativ beeinflussen könnten. Idealerweise sollen nur jene Trajektorien beibehalten werden, welche eine komplette Bewegung eines Fahrzeugs auf einem bestimmten Straßenabschnitt repräsentieren. Anhand dieser Trajektorien kann anschließend die Geometrie der realen Fahrspuren ermittelt werden. Ausreißer wie stehende oder unterbrochene Trajektorien liefern hingegen keine verwendbaren Informationen für die Spurerkennung.

In nachfolgender Auflistung sind die vier wichtigsten Vorverarbeitungsschritte und ihre Anwendungsreihenfolge aufgeführt. Die einzelnen Schritte und ihr Hintergrund werden anschließend noch genauer erläutert.

1. Resampling von Trajektorien auf minimale Punktdistanz
2. Entfernung zu kurzer Trajektorien
3. Entfernung unterbrochener Trajektorien
4. Anpassung der Trajektorien bei niedrigen Aufnahmewinkeln

6.2.1 Resampling von Trajektorien auf minimale Punktdistanz

Der erste Vorverarbeitungsschritt reduziert die Anzahl der Koordinaten, welche eine Bewegungsbahn beschreiben, erheblich ohne dabei jedoch wichtige Informationen zu verlieren. Insbesondere dann, wenn sich Fahrzeuge mit niedrigen Geschwindigkeiten bewegen oder teilweise von Ampeln et cetera. stehen, bestehen die Roh-Trajektorien aus sehr vielen Punkten, welche oft beinahe identische Positionsinformationen darstellen, das heist nur sehr geringe Abstände voneinander haben. Für die Beschreibung einer Bewegungsbahn ist diese Punktdichte nicht notwendig und sogar kontraproduktiv, da sie die Performance der nachfolgenden Schritte stark erhöht. Hiervon ist insbesonders die Clusteranalyse betroffen.

Aus diesen Gründen werden im ersten Vorverarbeitungsschritt Trajektorien auf eine minimale Punktdistanz von 0.5 m gebracht. Der hierzu verwendete Algorithmus ist in Listing 6.1 dargestellt. Er verwirft alle aufeinanderfolgende Punkte, welche von einem Referenzpunkt weniger als den geforderten Abstand haben.

```

1 algorithm resampleTrajectory:
2   input: lastRefPoint, newTrajPoints, oldTrajPoints
3   output: resampled trajectory points
4
5   while oldTrajPoints is not empty do:
6     nextPoint := Head(oldTrajPoints)
7     remPoints := Tail(oldTrajPoints)
8
9     if dist(lastRefPoint, nextPoint) < 0.5:
10       resampleTrajectory(lastRefPoint, newTrajPoints, remPoints)
11     else:
12       resampleTrajectory(nextPoint, newTrajPoint ++ nextPoint, remPoints)
13     end
14   end
15
16   return newTrajPoints

```

List. 6.1: Pseudocode Trajektorie Resampling

Nach Anwendung des Algorithmus ist im Fall der Neckartor Aufnahme die durchschnittliche Punktlänge der Trajektorien von 1094 Koordinaten auf 160 gesunken. Die realen Längen der Bewegungsbahnen bleiben hingegen nahezu gleich. Die Punktwolken, welche stehende Fahrzeuge erzeugen, werden mittels dieses Schritts ebenfalls entfernt (siehe Abb. 6.5 b)).

6.2.2 Entfernung zu kurzer Trajektorien

Der zweite Verarbeitungsschritt ist sehr einfach aber dennoch sehr effektiv, da mit seiner Hilfe viele Trajektorien von beispielsweise stehenden Fahrzeugen oder kurz auftretende Tracking-Fehler entfernt werden können. Hierzu werden die Längen aller Trajektorien überprüft und jene entfernt, welche unter bestimmten Grenzwerten liegen. Die hierzu verwendete boolesche Überprüfung ist in Gleichung 6.2 gegeben. Die Grenzwerte wurden experimentell bestimmt und lieferten für die Testaufnahmen gute Ergebnisse.

$$isShortTrajectory(t) = \begin{cases} 1 & \text{if } t.pointLength < 50 \vee t.realLength < 10.0 \\ 0 & \text{otherwise} \end{cases} \quad (6.2)$$

Das Ergebnis der ersten beiden Vorverarbeitungsschritte ist in Abbildung 6.5 dargestellt. Die Trajektorien stehender Fahrzeuge, sowie die Punktwolken vor Lichtsignalanlagen und weitere Defekte aufgrund kleiner Tracking-Fehler wurden entfernt.

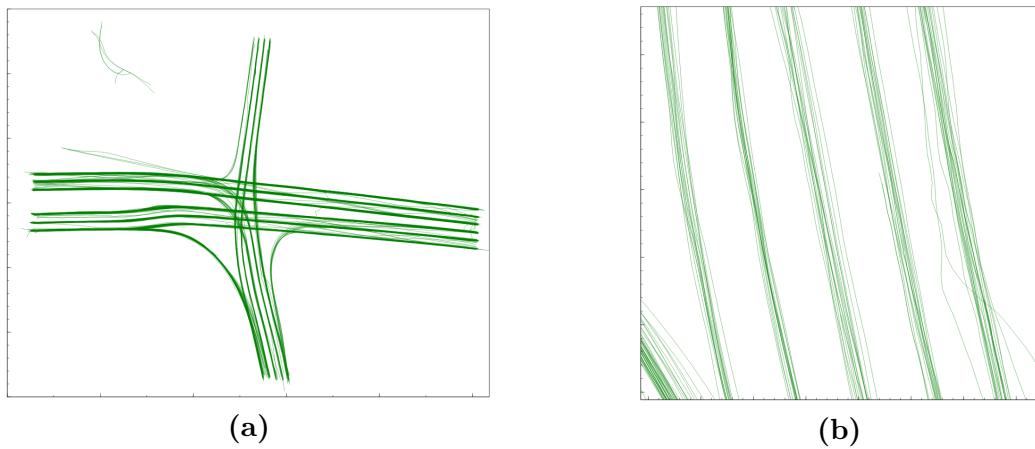


Abb. 6.5: Ergebnisse der zwei ersten Vorverarbeitungsschritte

6.2.3 Entfernung unterbrochener Trajektorien

Nachdem die ersten beiden Verarbeitungsschritte bereits stehende Trajektorien entfernt haben, müssen nun noch unterbrochene Verfolgungen ausgefiltert werden. Hierzu wird angenommen, dass komplette Bewegungsbahnen immer im Bereich der Szenenränder beginnen und enden. Wurde eine Fahrzeugverfolgung unterbrochen, so befinden sich die Anfänge beziehungsweise Enden der daraus resultierenden Trajektorien im Innenbereich der Aufnahme. Diese Trajektorien werden entfernt. Abbildung 6.6 veranschaulicht das Prinzip der Szenen Rand- und Innenbereiche.

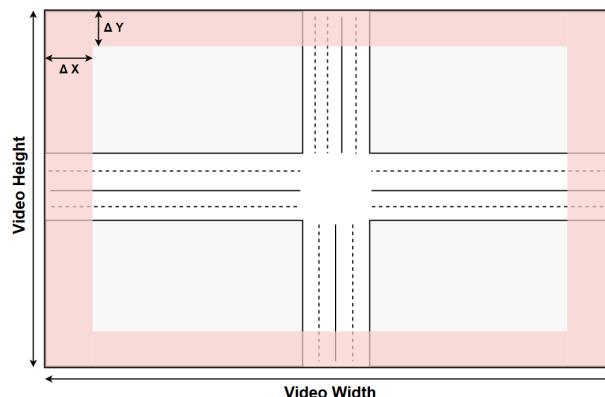


Abb. 6.6: Definition von Szenen Rand- und Innenbereich

Die Werte ΔX und ΔY entsprechen 10% der Video-Breite beziehungsweise Höhe. Ob sich eine Punkt p innerhalb des Szenenrandes befindet, wird mit Hilfe von Formel 6.3 überprüft. Hierbei entsprechen $p.sX$ und $p.sY$ den Bildkoordinaten des Punktes p .

$$inSceneFringe(p) = \begin{cases} 1 & \text{if } (p.sX < \Delta X \vee (p.sX > (vWidth - \Delta X))) \\ & \vee ((p.sY < \Delta Y) \vee (p.sY > (vHeight - \Delta Y))) \\ 0 & \text{otherwise} \end{cases} \quad (6.3)$$

Nach Ausführung dieses Filter-Schrittes, besteht die Menge der übrigen Trajektorien grundsätzlich nur noch aus kompletten Bewegungsbahnen.

Dieser Vorverarbeitungsschritt beruht auf der Annahme, dass für eine Fahrspur neben unterbrochenen auch ununterbrochene Trajektorien vorliegen, welche eine vollständige Bewegung auf der Spur beschreiben. Sind alle Trajektorien einer Fahrspur aufgrund einer Verdeckung in der Aufnahme unterbrochen, werden diese durch das Verfahren vollständig entfernt und eine Extraktion der Spur ist nicht möglich.

6.2.4 Anpassung der Trajektorien bei niedrigen Aufnahmewinkeln

6.3 Clusteranalyse

Nachdem im vorherigen Abschnitt beschrieben wurde, wie die Roh-Trajektorien vorverarbeitet und gefiltert werden, folgt nun die Erläuterung der verwendeten Ansätze zur Clusteranalyse der bereinigten Trajektorien.

6.3.1 Ansatz Spectral-Clustering und modifizierte Hausdorff-Distanz

Als erster Ansatz für die Clusteranalyse wurde das von [Atev u. a., 2010] vorgestellte Verfahren untersucht. Es wurde gewählt, da es sowohl in der Arbeit von Atev et al. selbst, sowie auch in [Morris und Trivedi, 2009], gute Ergebnisse bei der Clusteranalyse brachte. Das Verfahren beruht auf der in [Atev u. a., 2006] vorgestellten modifizierten Hausdorff Distanz und dem Spectral-Clustering-Algorithmus. Die Grundlagen des Distanzmaßes wurden bereits in Abschnitt 3.1 beschrieben. Nachfolgend wird zuerst genauer beschrieben, wie die modifizierte Hausdorff Distanz berechnet wird und in dieser Arbeit implementiert wurde. Anschließend wird auf den eingesetzten Spectral-Cluster-Algorithmus und die erzielten Ergebnisse eingegangen.

Das Verfahren

Grundlegend war die modifizierte Hausdorff Distanz nach Atev et al., wie in Gleichung 3.1 bereits definiert, gegeben über:

$$h_{\alpha, N, C}(P, Q) = \underset{p \in P}{ord} \left\{ \min_{q \in N_Q(C_{P, Q}(p))} d(p, q) \right\}$$

Um die Distanz zwischen zwei Trajektorien P und Q zu bestimmen, müssen zuerst die minimalen Distanzen zwischen allen Punkten $p \in P$ und deren Nachbarschaften $N_Q(C_{P,Q}(p))$ in Q bestimmt werden. Eine Nachbarschaft in Q um den Punkt q_0 ist in Abhängigkeit des Parameters w definiert als:

$$N_Q(q_0) = \{q \in Q \mid |\pi_Q(q_0) - \pi_Q(q)| \leq w/2\} \quad (6.4)$$

$\pi_Q(q)$ entspricht hierbei der relativen Position von q in Q , welche in Gleichung 6.5 definiert ist. $|Q_j|$ steht für die Länge einer Trajektorie bis zum Punkt j und $|Q|$ für die Gesamtlänge einer Bewegungsbahn. Diese Längeninformationen sind beide in der verwendeten Trajektorie-Definition aus Abbildung 6.1 enthalten.

$$\pi_Q(q_j) = \frac{|Q_j|}{|Q|} \quad (6.5)$$

Die Nachbarschaften werden um den Referenzpunkt $q \in Q$ gebildet, welcher die selbe relative Position in Q besitzt wie $p \in P$. Der Index dieses Punktes ergibt sich aus dem Mapping $C_{P,Q}$ wie folgt:

$$C_{P,Q}(p) = \arg \min_{q \in Q} |\pi_P(p) - \pi_Q(q)| \quad (6.6)$$

Zur Berechnung der Distanzen $d(p,q)$ zwischen p und allen Punkten $q \in N_Q(C_{P,Q}(p))$, wird die euklidische Distanz verwendet. Wurden auf diese Weise alle minimalen Distanzen zwischen Punkten und ihren Nachbarschaften in der Vergleichs-Trajektorie bestimmt, so wird aus ihnen der finale Distanzwert bestimmt. Der Operator $ord_{p \in P}^\alpha$ wählt hierfür jene Distanz, welche größer ist als α -Prozent aller Werte. Mit Hilfe dieses Distanzmaßes, wird eine Distanz-Matrix D konstruiert, welche die Distanzen aller Trajektorie-Kombinationen speichert.

Da im Spectral-Clustering Verfahren die modifizierte Hausdorff-Distanz nicht direkt eingesetzt werden kann, muss ein zusätzliches Affinitätsmaß verwendet werden. Dieses definieren Atev et al. als:

$$k(P,Q) = \exp\left(-\frac{h_{\alpha,N,C}(P,Q) h_{\alpha,N,C}(Q,P)}{2\sigma(P)\sigma(Q)}\right) \quad (6.7)$$

$\sigma(P)$ und $\sigma(Q)$ entsprechen hier Schätzungen für die Streuung der Distanzwerte einer Trajektorie zu allen anderen Trajektorien. Sie ergeben sich aus der Distanz-Matrix D .

Der in dieser Arbeit und von Atev et al. verwendete Spectral-Cluster-Algorithmus folgt grundsätzlich der Standard-Definition von [Ng u. a., 2002]. Für eine feste Clusteranzahl k kann er wie folgt zusammengefasst werden:

- 1) Erstellen einer Affinitätsmatrix $A \in \mathbb{R}^{n \times n}$ basierend auf Affinitätsmaß, wobei $A_{ii} = 0$
- 2) Definieren einer Diagonalenmatrix D , deren Elemente an der Stelle (i,i) der Summe der i -ten Zeile von A entsprechen. Basierend auf D wird die Matrix $L = D^{-1/2}AD^{-1/2}$ erstellt.
- 3) Durchführen einer Eigenwert Dekomposition auf L , um die k größten Eigenvektoren $\{x_1, x_2, \dots, x_k\}$ zu finden.
- 4) Erstellen einer Matrix $X = [x_1, x_2, \dots, x_k]$ durch spaltenweises Zusammenführen der Eigenvektoren und Normalisierung der Zeilen auf die Länge 1.
- 5) Gruppierung der Zeilen von Matrix X in k -Cluster unter Zuhilfenahme von k-Means et cetera. Jede Zeile wird als Datenpunkt interpretiert.

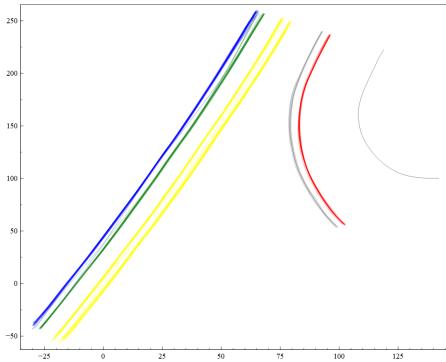
Ein großer Nachteil des Spectral-Clustering-Ansatzes ist es, dass bei seiner Verwendung üblicherweise die Clusteranzahl k bereits im vorraus bekannt und angegeben sein muss. Aus diesem Grund verwenden Atev et al. ein Schätzmaß für k , welches darauf abzielt, die Verzerrung des in Schritt 5) eingesetzten k-Means Algorithmus zu minimieren. Hierzu wird die k-Means Clusteranalyse mit mehreren k 's zwischen Grenzen $kMin$ und $kMax$ durchgeführt und anschließend jeweils ein Verzerrungs-Maß ρ_k berechnet, welches angibt, wie gut die gefundenen Centroids die Datenpunkte beschreiben. Für k wird daher jener Wert gewählt, welcher das kleinste ρ_k erzeugt. Diese Methode zur Schätzung der Clusteranzahl wurde auch in der vorliegenden Arbeit angewandt.

Auswertung des Verfahrens

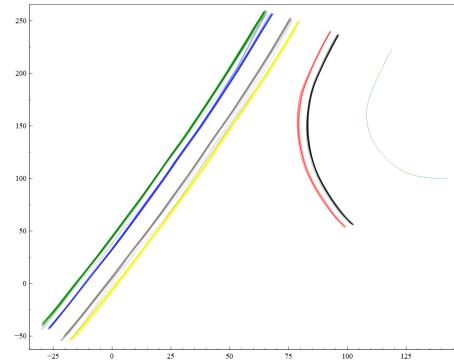
Nach Umsetzung des Verfahrens, wurde seine Qualität evaluiert. Hierzu wurde die Clustering-Performance des Ansatzes anhand von zwei Trajektorie-Datensätzen untersucht. Die Ergebnisse der Clusteranalyse für den einfacheren Datensatz sind in Abbildung 6.7 dargestellt. An diesem Beispiel lassen sich die Probleme bereits identifizieren. Für die Analyse wurden die Parameter $\alpha = 0.85$ und $w = 1.0$ verwendet. Zur Bestimmung von σ kamen die Grenzwerte $stdMin = 0.5$ und $stdMax = 10.0$ zum Einsatz.

Die ersten Untersuchungen und Tests des Ansatzes zeigten sehr schnell Schwächen und Grenzen des Vorgehens auf. Eine Problem sind zum einen die vielen Parameter und Grenzwerte des Verfahrens, welche nicht alle intuitiv verständlich sind und vom Anwender bestimmt werden müssen. Neben den oben aufgeführten Parametern, gibt es noch weitere, welche bei der Bestimmung der Clusteranzahl zum Einsatz kommen. Die Parameter wurden zuerst mit geschätzten, sinnvollen Werten initialisiert und anschließend über verschiedene Tests hinweg optimiert. Eine weitere Optimierung der Parameter und somit der erzielten Ergebnisse wäre sicherlich möglich gewesen, hierauf wurde aber aufgrund der weiteren Probleme verzichtet.

Problematisch ist das Vorgehen auch, da der Spectral-Cluster-Algorithmus nicht mit Ausreißern umgehen kann, welche trotz der Vorverarbeitung der Trajektorien immer noch in



(a) mit Schätzung Clusteranzahl



(b) mit fester Clusteranzahl

Abb. 6.7: Ergebnisse Clusteranalyse B10-Entennest (Ansatz Atev et al.)

den Datensätzen vorhanden seien können. In Abbildung 6.7 a) und b) ist beispielsweise die einzelne Fahrspur im oberen, rechten Bereich einem anderen Spur-Cluster zugeordnet, was nicht korrekt ist. Diese falsche Zuordnung der Ausreißer, würde die Bestimmung der Spur-Geometrien im nächsten Schritt des Algorithmus aus 5.2 erheblich erschweren.

Die schlechten Clustering-Ergebnisse des Ansatzes sind zudem Folge der nicht zuverlässig funktionierenden Schätzung der Clusteranzahl k . Im Fall des Datensatzes aus Abbildung 6.7, wird $k = 5$ statt korrekterweise $k = 6$ geschätzt. Hieraus ergibt sich, dass in a) zwischen zwei Fahrspuren nicht richtig unterschieden wird. In b) wurde die Clusteranzahl händisch spezifiziert, was in einer korrekten Gruppierung der Fahrspuren resultierte.

Ausschlaggenend dafür, dass der Ansatz von Atev et al. nicht weiter verfolgt und optimiert wurde, war schlussendlich jedoch die Performance der Clusteranalyse. Für den Trajektorie-Datensatz aus Abbildung 6.7, welcher nach der Vorverarbeitung nur 132 Bewegungsbahnen beinhaltet, benötigt die Clusteranalyse bereits über 90 Sekunden. Für einen Datensatz mit über 400 Trajektorien, betrug die Verarbeitungsdauer über 8 Minuten. Die schlechte Performance lässt sich primär auf das aufwendige Distanzmaß, aber auch auf die mehrfache Durchführung des k-Means Algorithmus zurückführen.

Aufgrund der oben angeführten Problematiken wurde früh entschieden, dass der Ansatz von Atev et al. nicht weiter verfolgt werden soll. Es wurde ein Ansatz gesucht, welcher mit den beschriebenen Herausforderungen besser umgehen kann.

6.3.2 Ansatz DBSCAN und LCSS

Nachdem das in Abschnitt 6.3.1 beschriebene Verfahren zur Clusteranalyse in den Untersuchungen schlechte Ergebnisse lieferte, wurde nach neuen Ansätzen gesucht. Kriterien für diese waren primär, dass sie mit Ausreißern sinnvoll umgehen können, weniger Parametrisierung benötigen beziehungsweise diese intuitiver ist, und sie eine bessere Performance

besitzen. Aufgrund dieser Anforderungen wurde untersucht, inwiefern sich der DBSCAN-Cluster-Algorithmus in Kombination mit dem LCSS-Distanzmaß für die Clusteranalyse von Trajektoriendaten eignet.

Vorteil des dichte-basierten DBSCAN-Algorithmus, dessen Funktionsweise bereits in Abschnitt 2.2.2 erläutert wurde, ist, dass er Ausreißer erkennt und die Anzahl der Zielcluster selbst bestimmen kann. Das LCSS Distanzmaß, grundlegend in Abschnitt 2.3.3 beschrieben, ähnelt der modifizierten Hausdorff Distanz, lässt sich allerdings performanter implementieren und besitzt eine eingängigere Parametrisierung. Es lieferte in den Arbeiten [Morris und Trivedi, 2011] und [Chen u. a., 2014] gute Clustering-Ergebnisse.

Das Verfahren

Die Grundgleichung des LCSS Distanzmaßes wurde, basierend auf [Vlachos u. a., 2002], bereits in Gleichung 6.8 definiert und ist nachfolgend nochmals dargestellt.

$$LCSS_{\epsilon,\delta}(t_1, t_2) = \begin{cases} 0 & \text{if } t_1 \text{ or } t_2 \in \emptyset \\ 1 + LCSS_{\epsilon,\delta}(t'_1, t'_2) & \text{if } dist(t_1(n), t_2(m)) < \epsilon \\ & \wedge |n - m| \leq \delta \\ max(LCSS_{\epsilon,\delta}(t'_1, t'_2), LCSS_{\epsilon,\delta}(t_1, t'_2)) & \text{otherwise} \end{cases}$$

Mit ihrer Hilfe lässt sich bestimmen, wie groß die längste, übereinstimmende Subsequenz zweier Trajektorien t_1 und t_2 ist. Damit Punkte zweier Trajektorien als übereinstimmend gewertet werden, dürfen sie höchstens die Distanz ϵ zueinander haben und ihre Position in den Bewegungsbahnen sich um δ unterscheiden. Wenn das Zahlmaß, wie oben dargestellt, rekursiv implementiert wird, liegt die Zeitkomplexität des Algorithmus bei $O(2^n)$, was für den Vergleich einer größeren Anzahl von Trajektorien nicht geeignet ist. Die Performance kann auf $O(m n)$ verbessert werden, indem das Maß mit Hilfe von dynamischer Programmierung und *Memoization* umgesetzt wird. Der sich so ergebende Algorithmus ist in Listing 6.2 aufgeführt.

```

1 algorithm LCSS:
2   input: trajectory: t1, trajectory: t2, epsilon, deltaFac
3   output: LCSS distance between t1 and t2
4
5   t1Len := point-length t1
6   t2Len := point-length t2
7   delta := deltaFac * min(t1Len, t2Len)
8   LCS := 2D array with dims. (t1Len+1, t2Len+1)
9
10  for i <- 1 to t1Len do:
11    for j <- 1 to t2Len do:
12      if dist(t1(i-1), t2(j-1)) < epsilon && |i-j| < delta then:
13        LCS(i)(j) = LCS(i-1)(j-1) + 1
14      else
15        LCS(i)(j) = max(LCS(i-1)(j), LCS(i)(j-1))
16      end

```

```

17     end
18 end
19
20 return LCS(t1Len)(t2Len)

```

List. 6.2: Pseudocode LCSS Bestimmung

Der Parameter δ wird hier nicht, wie meist üblich, fest definiert, sondern er ergibt sich als ein Teil der Länge der kurzeren Trajektorie (siehe Lst. 6.2 Zeile 7). Als Distanzfunktion für das LCSS Zählmaß, wird in dieser Arbeit das von [Vlachos u. a., 2002] definierte Maß aus Gleichung 2.12 verwendet:

$$D_{LCSS}(\delta, \epsilon, t_1, t_2) = 1 - \frac{LCSS_{\delta, \epsilon}(t_1, t_2)}{\min(\text{len}(t_1), \text{len}(t_2))}$$

Auf die in Abschnitt 3.1 vorgestellte Erweiterung des Distanzmaßes wird verzichtet, da es nicht gewünscht ist, dass formähnliche aber im Raum verschobene Trajektorien kleine Distanzen besitzen.

Der DBSCAN Cluster-Algorithmus wurde in diesem Fall nicht selbst implementiert. Es wurde die Implementierung der *Statistical Machine Intelligence and Learning Engine*¹ (*Smile*) Bibliothek verwendet. Der dort angebotene DBSCAN Algorithmus kann beliebige Datenobjekte, unter Verwendung eines vom Anwender gestellten Distanzmaßes, gruppieren. Zusätzlich muss lediglich die Größe der gewünschten ϵ -Nachbarschaft und *MinPts* definiert werden (siehe Abschnitt 2.2.2).

Auswertung des Verfahrens

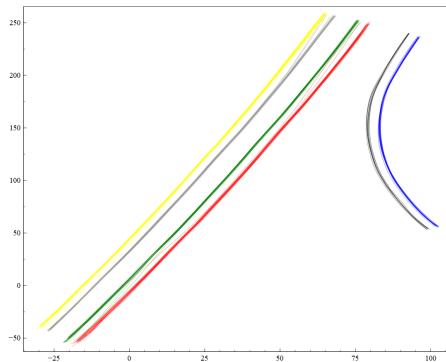
In Abbildung 6.8 sind die Cluster-Ergebnisse des beschriebenen Verfahrens dargestellt. Diese Resultate ergeben sich unter Verwendung der folgenden Parameter: $\epsilon_{LCSS} = 1.0$, $\delta = 0.1$, $\epsilon_{DBSCAN} = 0.4$ und $\text{minPts} = 5$.

Die Bedeutung der Werte ist in diesem Fall leicht und intuitiv verständlich: Punkte auf einer Trajektorie dürfen maximal 1m voneinander entfernt sein und ihre Position sich um maximal $\delta * \text{minTraLength}$ unterscheiden. Sind diese Bedingungen erfüllt, gelten zwei Trajektorie-Punkte als übereinstimmend. Ein Cluster muss zudem aus mindestens 5 Trajektorien bestehen und die Differenzen der Trajektorie-Distanzen untereinander dürfen nicht größer als 0.4 sein.

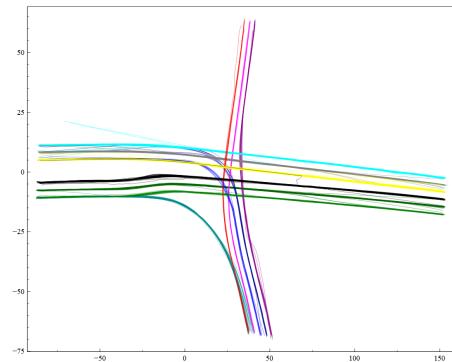
Die Ergebnisse aus Abbildung 6.8 a) und b) zeigen, dass bei Verwendung dieses Verfahrens zwischen allen Fahrspuren korrekt unterschieden wird und die einzelnen Trajektorien richtig gruppiert werden. Zudem funktioniert die Bestimmung der Clusteranzahl

¹ Smile Bibliothek: <https://haifengl.github.io/smile/index.html>

zuverlässig. Auch die Performance der Clusteranalyse konnte unter Verwendung des DBSCAN Algorithmus und der LCSS-Distanz verbessert werden. Die benötigte Zeit für den Autobahn-Datensatz sank auf circa 15 Sekunden und die des Kreuzung-Datensatz auf etwa 85 Sekunden, was in diesem Anwendungsfall akzeptabel ist.



(a) Datensatz B10-Entennest



(b) Datensatz Neckartor

Abb. 6.8: Ergebnisse Clusteranalyse DBSCAN und LCSS-Distanz

Der DBSCAN Algorithmus markiert nun auch atypische Trajektorien als Ausreißer, was den Vorteil hat, dass diese nichtmehr anderen Spurclustern zugeordnet werden. Problematisch ist nun allerdings, dass Trajektorien welche Abbiegevorgänge beschreiben und in unterschiedliche Spuren einbiegen, teilweise auch als Ausreißer klassifiziert werden. In Abbildung 6.8 b) fehlen so beispielsweise die Trajektorien der zwei Abbiegespuren in der oberen linken und unteren rechten Ecke (vergleiche Abbildung 6.2). Um diese weiterhin erkennen zu können, wurde ein extra Verarbeitungsschritt eingeführt, welcher in Abschnitt 6.3.3 beschrieben wird.

Grundsätzlich überzeugten allerdings die Ergebnisse der Clusteranalyse. Die Qualität des Ansatzes könnte auch durch Anwendung auf weitere Datensätze verifiziert werden. Die nachfolgend beschriebenen Schritte der Spurerkennung basieren daher auf den Ergebnissen der hier vorgestellten Cluster-Methode.

6.3.3 Erkennung von Abbiegespuren

In Abschnitt 6.3.2 wurde bereits erwähnt, dass, unter Verwendung der gewählten Cluster-Methodik, einige Bewegungsbahnen als Ausreißer klassifiziert werden, dies eigentlich jedoch nicht sind. Diese fälschlicherweise aussortierten Trajektorien beschreiben für gewöhnlich Abbiegevorgänge. Problematisch an diesen Trajektorien, aus Sicht der Clusteranalyse, ist, dass sie üblicherweise auf einer gemeinsamen Spur – der Abbiegespur – beginnen,

sich dann jedoch aufteilen und auf mehrere Fahrspuren verteilen. Die Trajektorien haben aus diesem Grund, trotz anfänglich identischem Verlauf, zu hohe Distanzen zueinander, um als ein Cluster identifiziert zu werden. Hinzu kommt, dass die Anzahl der Fahrzeuge, welche eine Abbiegespur benutzen, häufig nicht hoch genug ist, damit eine der Abbiege-Varianten ein Cluster formt. In Abbildung 6.9 ist beispielhaft eine Abbiegespur der Neckartor-Kreuzung und die dazugehörigen Trajektorien abgebildet. Anhand der Straßentopologie und der Trajektorien wird deutlich, dass sich die Fahrzeuge nach dem abbiegen auf drei Fahrstreifen verteilen.

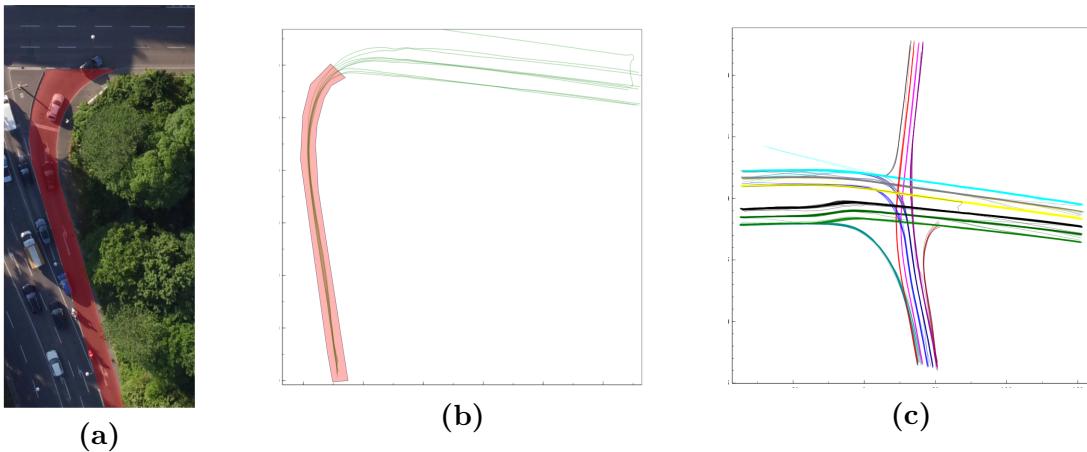


Abb. 6.9: a) Abbiegespur, b) zugehörige Trajektorien, c) Spurcluster mit Abbiegespuren

Ziel des nachfolgend beschriebenen Verarbeitungsschrittes ist es, Trajektorien, welche Abbiegevorgänge beschreiben, zu entsprechenden Clustern zusammenzufassen. Anhand dieser kann anschließend die Geometrie der Spuren, wie für alle anderen Fahrspuren auch, bestimmt werden. Die Cluster sollen hierbei allerdings nicht aus den vollständigen Trajektorien gebildet werden, sondern lediglich aus den Teilen, welche anfänglich identisch verlaufen. Dies ist konzeptionell in Abbildung 6.9 b) dargestellt. Notwendig ist dies, da die Trajektorien nach dem Auseinanderlaufen sehr unterschiedliche Bahnen beschreiben können und dies die Spur-Geometrie-Bestimmung negativ beeinflusst.

Um die oben beschriebenen Spurcluster zu finden, werden ausschließlich die Trajektorien untersucht, welche während der Clusteranalyse als Ausreißer markiert wurden. In einem ersten Schritt wird nach möglichen Trajektorie-Nachbarschaften gesucht. Hierzu wird für jede Trajektorie t der Abstand zwischen deren Start und allen anderen Trajektorie-Starts bestimmt. Liegen mehr als fünf Trajektorie-Anfänge in einem Radius $laneEps = 1.5m$ um den Start von t , so bilden die Trajektorien eine mögliche Nachbarschaft. Dass die so gefundenen Trajektorien auch tatsächlich den Verlauf einer Abbiegespur beschreiben und nicht direkt auseinander laufen, wird anschließend geprüft.

Für eine Trajektorie-Nachbarschaft N wird zunächst eine Referenz Bewegungsbahn $refT$ bestimmt, welche den minimalen mittleren Abstand zu allen anderen Trajektorien besitzt:

$$refT_N = N(\arg \min_{t \in N} \left(\frac{1}{|N|} \sum_{tt' \in \{N \setminus t\}} D_{LCSS}(t, tt') \right)) \quad (6.8)$$

Die Referenz-Trajektorie beschreibt die Nachbarschaft. Im Fall der Trajektorien aus Abbildung 6.9 b) ist es beispielsweise eine Bahn, welche auf die mittlere Fahrspur abbiegt. Nachdem $refT$ bestimmt wurde, wird die Trajektorie Punktweise durchwandert. Es werden die Bereiche um jeden Punkt überprüft, um so festzustellen, wo die Trajektorien der Nachbarschaft auseinanderlaufen. Der hierzu eingesetzte Algorithmus ist in Listing 6.3 beschrieben.

```

1 algorithm findSplitPoint:
2   input: neighborhood: N, reference-trajectory: refT
3   output: point where trajectories of neighborhood split up
4
5   for each point in refT do
6     regionAroundP := create circle around point with r = laneEps
7     neighborsInReg := count intersections of trajectories with regionAroundP
8
9     if neighborsInReg <= 0.75 * |N| && idx(point) > 0.33 * refT.pointLength then
10       return point
11     end
12   end
13
14  return None

```

List. 6.3: Pseudocode Split-Punkt Bestimmung

Zur Bestimmung der Regionen um einen Punkt und der Schnittpunkte der Trajektorien mit diesem Bereich, wird die JTS Topology Suite¹ (JTS) eingesetzt. Der Split-Punkt einer Nachbarschaft ist jener Punkt, in dessen Umfeld die Anzahl der Trajektorien erstmals unter 75% fällt. Er ist zudem nur gültig, wenn er nicht im ersten Drittel der Referenz-Trajektorien liegt.

Nachdem der Split-Punkt einer Nachbarschaft gefunden wurde, werden alle in ihr beinhalteten Bewegungsbahnen an jenem Trajektorie-Punkt geteilt, der dem Split-Punkt am nächsten ist. Die so neu erstellten Trajektorien bilden ein neues Cluster. In Abbildung 6.9 c) sind nun alle Spurcluster dargestellt, welche nach diesem Schritt existieren. Die zwei Cluster der Abbiegespuren, welche noch in Abbildung 6.8 b) fehlten, sind nun auch vorhanden.

Mithilfe der in diesem Kapitel vorgestellten Verfahren können Trajektorie-Cluster entdeckt werden, welche die einzelnen Fahrspuren in einer Aufnahme beschreiben. Im nächsten Kapitel wird darauf eingegangen, wie aus diesen Clustern Spur-Geometrien abgeleitet werden.

¹ JTS Topology Suite: <https://locationtech.github.io/jts/>

7 Fahrspur-Bestimmung aus Trajektorie-Clustern

In diesem Kapitel wird beschrieben, wie aus den bereits identifizierten Trajektorie-Clustern Geometrie-Informationen der Fahrspuren abgeleitet werden. Hierzu sind primär drei Schritte notwendig: Bestimmung der Spur-Mittellinien, Bestimmung der Spurhüllen und anschließend die Partitionierung sich überlagernder Spuren. Hinzu kommen weitere kleine Schritte, welche ebenfalls nachfolgend beschrieben werden.

7.1 Ausfilterung von Spurwechselvorgängen

Bevor mithilfe der im vorherigen Schritt gewonnenen Trajektorie-Cluster Mittellinien von Fahrspuren bestimmt werden können, müssen diese nochmals vorverarbeitet werden. Die einzelnen Cluster enthalten teilweise Bewegungsbahnen, welche Spurwechselvorgänge oder andere Abweichungen von einer Fahrspur beschreiben. Diese Trajektorien müssen, so weit wie möglich, entfernt werden, da sie die anschließende Geometrie-Bestimmung negativ beeinträchtigen. Die Trajektorien eines Clusters sollten eindeutig einer realen Fahrspur zuzuordnen sein. In Abbildung 7.1 sind beispielhaft zwei Cluster dargestellt, welche eine Vielzahl an Spurwechselvorgängen enthalten.

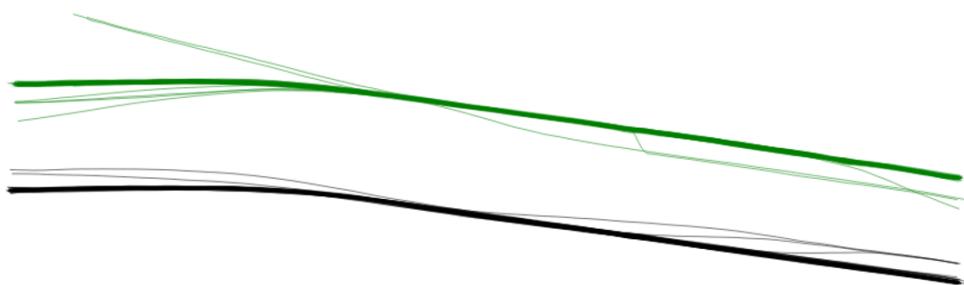


Abb. 7.1: Trajektorie-Cluster mit Spurwechselvorgängen

Die Grundidee, welche der Ausfilterung der Abweichungen zugrunde liegt, ist, jene Trajektorien aus einem Cluster zu entfernen, welche eine überdurchschnittlich hohe mittlere Distanz zu allen anderen Trajektorien des Clusters besitzen. Als Distanzmaß wird erneut die LCSS Distanz verwendet. Ein ähnlicher, Distanz-basierter Ausreißer-Detektionsansatz

wird in [Mirge u. a., 2017] vorgestellt. Der in dieser Arbeit verwendete Algorithmus ist in Listing 7.2 beschrieben.

```

1 algorithm filterCluster:
2   input: unfiltered trajectories of cluster: trajsIn
3   output: filtered trajectories of cluster
4
5   meanTrajectoryDistances :=
6     for each traj in trajsIn do
7       yield mean LCSS distance of traj to all other trajectories of trajsIn
8     end
9
10  clusterCmpVal := select median of meanTrajectoryDistances as comparison value
11
12  resultTrajs :=
13    for each traj in trajsIn do
14      if meanDist of traj < 1.5 * clusterCmpVal then
15        yield traj
16      end
17    end
18
19  return resultTrajs

```

List. 7.1: Pseudocode Cluster Post-Processing

Das gewählte Verfahren ist einfach, eignet sich aber dennoch gut, um das gewünschte Ziel zu erreichen. Die in Abbildung 7.2 dargestellten Ergebnisse zeigen dies. Aus den in Abbildung 7.1 enthaltenen Clustern wurden alle Trajektorien mit Spurwechselvorgängen oder Abweichungen entfernt. Die Effektivität konnte auch durch die Anwendung auf andere Datensätze bestätigt werden. Wenn in einem Cluster viele Trajektorien Abweichungen von einer Spur besitzen, ist es möglich, dass der Algorithmus diese nicht vollständig entfernt. Eine geringe Anzahl von Abweichungen beziehungsweise kleine Ausreißer sind allerdings unproblematisch.



Abb. 7.2: Trajektorie-Cluster ohne Spurwechselvorgänge

Zu beachten ist auch, dass die Zeitkomplexität des Verfahrens bei großen Trajektorie-Clustern nicht unerheblich ist. Dies ist auf die Komplexität des LCSS Distanzmaßes zurückzuführen. Bei der Untersuchung alternativer Vorgehensweise wurde allerdings klar, dass es grundlegend schwer ist, dieses Problem performant zu lösen. In [Meng u. a., 2018] werden hierzu diverse Distanz- und Dichte-basierte Arbeiten vorgestellt und verglichen. Da diese alle allerdings nicht weniger komplex sind und häufig die verfolgten Ziele über

die hier geforderten hinausgehen, wurde entschieden den oben beschriebenen Ansatz beizubehalten. Durch die Parallelisierung der Berechnung der mittleren Abstände konnte die Performance des Ansatzes außerdem nochmals deutlich gesteigert werden.

7.2 Bestimmung der Spurmittellinien

Nachdem die Trajektorie-Cluster nun weitestgehend von Fahrspurwechseln und anderen Abweichungen bereinigt wurden, beschreiben die verbleibenden Bewegungsbahnen eines Clusters idealerweise eine Fahrspur. Anhand dieser Trajektorien können daher nun die Mittellinie der Spuren bestimmt werden.

Die zu bestimmende Mittellinie verläuft durch die Mitte des entsprechenden Trajektorie-Clusters. Die einzelnen Bewegungsbahnen besitzen jeweils leichte Abweichungen von dieser „*Ideallinie*“. Als Spurmittellinie – wie von Hu u. a. [2005] vorgeschlagen – jene Trajektorie zu wählen, welche die kleinste Summe der Distanzen zu allen anderen Trajektorien eines Clusters besitzt, ist nicht praktikabel, da die so gefundene Trajektorie nicht über ihre komplette Länge in der Mitte des Clusters verlaufen muss. Auf eine Bestimmung der Mittellinie mittels linearer oder polynomialer Regression, wie beispielsweise in [Chen u. a., 2014] oder [Mélo u. a., 2006], wurde ebenfalls verzichtet. Der Grund hierfür ist, dass einerseits die Komplexität der Fahrbahnverläufe nicht bekannt ist und andererseits das Erlernen der Spurrepräsentation zu aufwendig ist.

Der in dieser Arbeit gewählte Ansatz zur Bestimmung der Spurmittellinien macht sich die von Atev et al. vorgestellte Notation einer relativen Position innerhalb einer Trajektorie zu nutze, welche in Gleichung 6.5 und 6.6 gegeben ist. Die Koordinaten der Spurlinien ergeben sich aus den Mittelwerten von Trajektorie-Punkten, welche sich alle an der selben relativen Position befinden. Der verwendete Algorithmus ist nachfolgend beschrieben.

```

1 algorithm calculateCenterline:
2   input: trajectories of cluster: trajsIn
3   output: centerline of lane
4
5   meanTrajLength := calculate mean point length of trajectories
6   relPositions := define range with relative positions of size meanTrajLength
7     and step-size (1 / meanTrajLength)
8
9   centerline :=
10  for each relP in relPositions do
11    pointsAtRelPos := get points at relP for each traj in trajsIn
12    yield mean of all points in pointsAtRelPos
13  end
14
15 return centerline

```

List. 7.2: Pseudocode Cluster Post-Processing

Da die Form und der Verlauf von Trajektorien eines Clusters sich nur wenig unterscheiden, liefert dieses Vorgehen gute Ergebnisse. In Abbildung 7.3 a) ist beispielhaft dargestellt, wie

eine Mittellinien innerhalb eines Trajektorie-Clusters verlaufen. In 7.3 b) sind alle Spurmittellinien der Neckartor-Kreuzung, welche auf die oben beschriebene Weise bestimmt wurden, abgebildet.

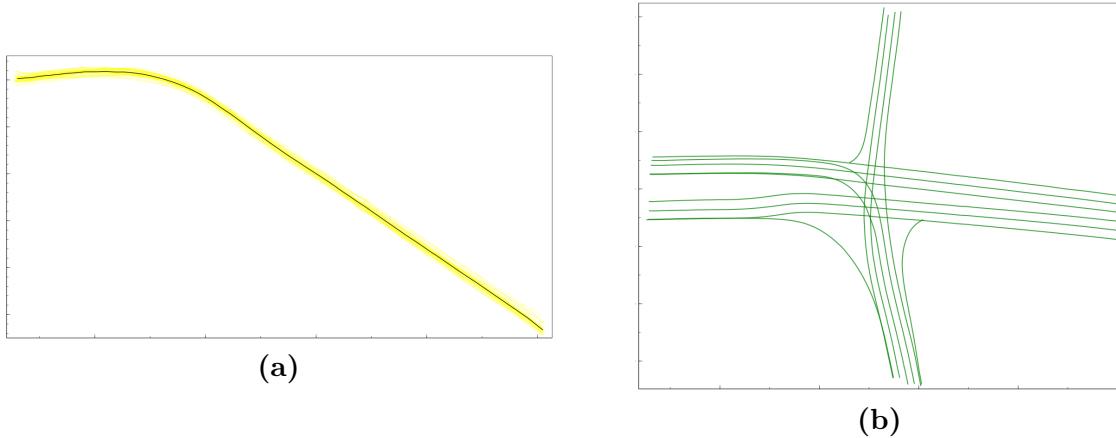


Abb. 7.3: Spurmittellinie in einem Trajektorie-Cluster a), Spurmittellinien Neckartor-Kreuzung b)

Die Mittellinien werden, da es sich bei ihnen ebenfalls nur um Sequenzen von 2D-Koordinaten handelt, als Trajektorie-Objekte repräsentiert (siehe Abbildung 6.1).

7.3 Bestimmung der Spurhüllen

Nachdem im vorherigen Schritt die Mittellinien der Fahrspuren bestimmt wurden, wird nun für jede Spurlinie eine Hülle definiert. In folgendem Abschnitt wird beschrieben, wie diese erstellt werden.

Die Spurhülle bestimmt die Dimension einer Fahrbahn. In dieser Arbeit verlaufen sie idealerweise entlang realer Begrenzungslinien welche zwei Spuren voneinander oder eine Fahrbahn von einem Seitenstreifen et cetera trennen. Eine Mittellinie und eine Hülle beschreiben zusammen die Geometrie einer Fahrspur. Diese Geometrie soll die realen Dimensionen einer Spur möglichst genau abbilden. Aus diesem Grund ist es nicht möglich, die Hüllen lediglich auf Basis von statistischen Informationen der Trajektorie-Cluster zu bestimmen, wie das beispielsweise in [Weiming Hu u. a., 2006] oder [Morris und Trivedi, 2011] gemacht wird. Der in dieser Arbeit verwendete Ansatz zur Bestimmung der Spurhüllen basiert daher lediglich auf den Spurmittellinien und nutzt ihre relative Lage zueinander. Ansätze hierzu stammen aus den Arbeiten von [Hsieh u. a., 2006] und [Makris und Ellis, 2005], welche in Abschnitt 3.2 vorgestellt wurden.

Die grundlegende Idee, auf welcher die Bestimmung der Spurhüllen basiert, ist konzeptiell in Abbildung 7.6 dargestellt. Für zwei parallel zueinander verlaufende Spuren l_1 und

l_2 werden die Hüllen, über den Abstand zwischen ihren Mittellinien, bestimmt. Besitzen die Linien den Abstand d zueinander, so beträgt die Breite der Spuren ebenfalls d . Der Abstand e_d zwischen einer Mittellinie und einer Hüll-Linie beträgt folglich $1/2 d$.

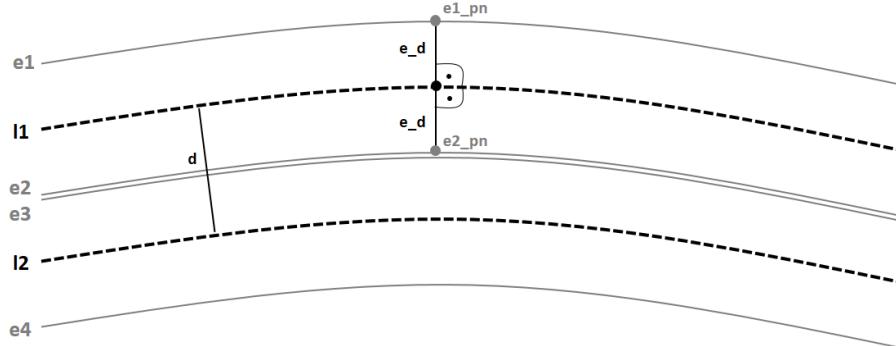


Abb. 7.4: Bestimmung Spurhüllen

Die oben beschriebene Definition einer Bahn-Geometrie wird im Spurerkennungs-Modul über eine Klasse *LaneGeometry* repräsentiert. Diese ist in Abbildung 7.5 dargestellt. Das Feld *variance* entspricht dem Abstand e_d zwischen der *centerline* und den Hüll-Linien.

LaneGeometry
+ id: Int
+ variance: Double
+ centerline: Trajectory
+ envelopeLine1: Vector[Point]
+ envelopeLine2: Vector[Point]

Abb. 7.5: Aufbau LaneGeometry Klasse

Einen ähnlichen Ansatz zur Bestimmung von Spurbegrenzungslinien verwenden auch Hsieh et al. in ihrer Arbeit. Sie gehen jedoch, da die von ihnen verwendeten Aufnahmen von einer statischen Kamera über einer Autobahn stammen, davon aus, dass alle Fahrspuren parallel zueinander verlaufen. Da das in dieser Arbeit entwickelte Spurerkennungs-Modul mit unterschiedlichen Aufnahmen und Straßentopologien umgehen können muss, kann diese Annahme nicht getroffen werden. Nachfolgend werden die Schritte vorgestellt, welche angewandt werden, um in einer Menge von Spuren jene zu finden, welche parallel zueinander verlaufen, auf Basis dieser die Spurbreiten zu bestimmen und anschließend die Hüll-Linien zu definieren.

Identifikation paralleler Fahrspuren

Um in einer Menge von Fahrspuren, welche über Mittellinien repräsentiert werden, jene zu finden, die parallel zueinander verlaufen, werden in einem ersten Schritt benachbarte

Spur-Paare gesucht. Zwei Fahrspuren l_1 und l_2 gelten als benachbart, wenn sich der Start oder das Ende von Spur l_2 in einem Bereich mit Radius σ um den Start von l_1 befindet. Der Wert für σ wurde auf Basis der in Deutschland geltenden „Richtlinien für die Anlage von Autobahnen“ (RAA) [FGSV, 2008] und der „Richtlinien für die Anlage von Landstraßen“ (RAL) [FGSV, 2012] bestimmt. Diese Regelwerke spezifizieren die in der Bundesrepublik derzeit zulässigen Straßenquerschnitte. Die in ihnen festgelegten Spurbreiten variieren zwischen 3.25 und 3.75 Metern. Um auch besonders breite benachbarte Spur-Paare identifizieren zu können, wurde für σ der Wert 4m gewählt.

Die auf diese Weise gefundenen Trajektorie-Paare starten oder enden als benachbarte Spuren. Der Algorithmus welcher prüft, ob zwei Spuren tatsächlich parallel zueinander verlaufen, ist in Listing 7.3 beschrieben. Die zu vergleichenden Mittellinien werden jeweils in eine Reihe von Richtungsvektoren umgewandelt. Anschließend werden die paarweisen Differenzen zwischen den Vektoren berechnet und gemittelt. Liegt die sich so ergebende Abweichung unter einem Grenzwert δ , so handelt es sich um parallele Spuren.

```

1 algorithm lanesAreParallel:
2   input: lane-centerline: l1, lane-centerline: l2, delta
3   output: True or False
4
5   inOppositeDirections := check if l1 and l2 run in opposite directions
6   if inOppositeDirections then
7     reverse points of centerline l2
8   end
9
10  l1DirectionVec := calculate direction vectors for l1
11  l2DirectionVec := calculate direction vectors for l2
12  dirDifferences := calculate pairwise difference between direction vectors
13
14  meanDiff := calculate mean divergence for X- and Y-axis components
15  lengthDiff := calculate difference between length of l1 and l2
16
17  return meanDiff < delta && lengthDiff < 10.0

```

List. 7.3: Pseudocode Überprüfung der Parallelität zweier Mittellinien

Für δ wurde experimentell der Wert 0.1 bestimmt. In den verschiedenen Test-Datensätzen konnten so zuverlässig parallele Spur-Paare identifiziert werden. Anhand dieser werden im nächsten Schritt die Spurhüllen bestimmt.

Berechnung der Hüllen

Bevor für eine Mittellinie die sie umgebende Spurhülle bestimmt werden kann, muss die Breite der Spur ermittelt werden. Diese ergibt sich, für die im vorherigen Schritt bestimmten parallelen Spurpaare, aus deren mittleren Abstand zueinander. Verlaufen zu einer Spurlinie zwei Bahnen parallel, werden die Abstände zu beiden berechnet und der kleinere wird als Spurbreite gewählt. Allen Bahnen, welche keine parallele Nachbarspur besitzen, wird die minimale Spurbreite zugeordnet, welche im vorherigen Schritt bestimmt wurde. Falls sich in einer Aufnahme keine parallelen Spuren befinden, wird für die Spurbreite ein

Standartwert von 3.5 Metern verwendet, welcher sich ebenfalls aus den RAA und RAL Richtlinien ableitet.

Nachdem auf die oben beschriebene Weise Breiten für alle Fahrspuren bestimmt wurden, können die Hüll-Linien berechnet werden. Hierzu werden für jeden Punkt der Mittellinie zwei zugehörige Hüll-Punkte berechnet. Das hierzu verwendete Vorgehen ist in Abbildung 7.6 dargestellt.

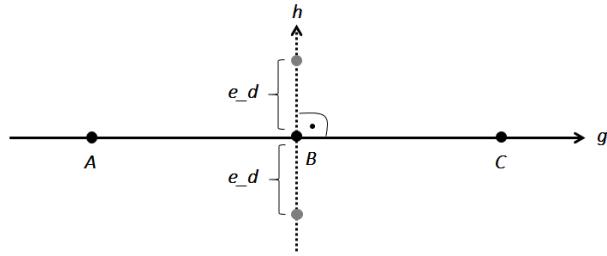


Abb. 7.6: Bestimmung Spürhüllen

Um die Hüll-Punkte für einen Punkt B einer Mittellinie zu bestimmen, wird eine Gerade g durch die Punkte A und C gelegt, welche sich vor und hinter B befinden.

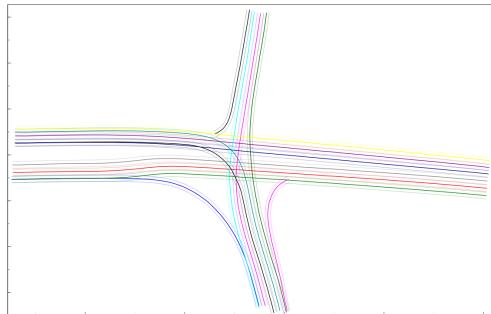
$$g : \vec{x} = \overrightarrow{OA} + t \cdot \overrightarrow{AC} \quad (7.1)$$

Anschließend wird durch B eine Gerade h gelegt, welche orthogonal zu g verläuft.

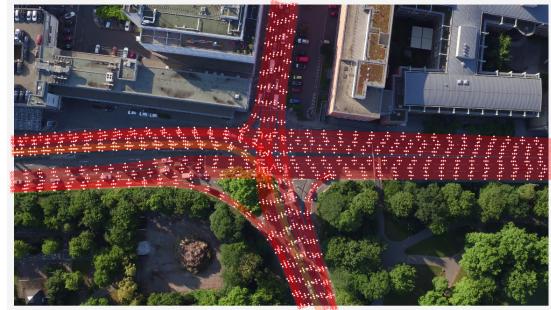
$$h : \vec{x} = \overrightarrow{OB} + t \cdot \hat{X} \quad (7.2)$$

Für ihren Richtungsvektor \hat{X} , welcher sich aus \overrightarrow{AC} ergibt, gilt $\overrightarrow{AC} \cdot \hat{X} = 0$. Da \hat{X} ein Einheitsvektor ist, können die Hüll-Punkte, welche auf h liegen und den Abstand e_d von B besitzen, einfach bestimmt werden, indem e_d beziehungsweise $-e_d$ für t in die Gleichung von h eingesetzt wird.

Auf diese Weise werden die Hüll-Linien für alle Fahrspuren bestimmt. In Abbildung 7.7 sind die Ergebnisse der Spur-Geometrie-Bestimmung dargestellt. Teil a) zeigt die Spurmittellinien mit ihren sie umgebenden Hüllen. Teil b) zeigt die in die TrackerApplication eingefügten Spuren.



(a)



(b)

Abb. 7.7: Plot Spurmittellinien und Hüllen a), Ergebnis Spur-Geometrien in TrackerApplication b)

Die obige Abbildung zeigt, dass die berechneten Spur-Geometrien in den meisten Fällen gut mit den realen Spur-Dimensionen übereinstimmen. Problematisch sind allerdings die Überlagerungen der Spuren in manchen Regionen. Diese werden im nächsten Schritt entfernt.

7.4 Partitionierung von Fahrspuren

8 Ergebnisse und Auswertung

9 Zusammenfassung und Fazit

Literaturverzeichnis

- [Aly 2008] ALY, Mohamed: Real time detection of lane markers in urban streets. In: *IEEE Intelligent Vehicles Symposium, Proceedings*, URL <https://ieeexplore.ieee.org/abstract/document/4621152/>, 2008, S. 7–12. – ISBN 9781424425693
- [Anders Drachen 2014] ANDERS DRACHEN: *Introducing Clustering II: Clustering Algorithms - GameAnalytics*. 2014. – URL <https://gameanalytics.com/blog/introducing-clustering-ii-clustering-algorithms.html>. – Zugriffsdatum: 2018-10-22
- [Atev u. a. 2006] ATEV, Stefan ; MASOUD, Osama ; PAPANIKOLOPOULOS, Nikos: Learning Traffic Patterns at Intersections by Spectral Clustering of Motion Trajectories. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, oct 2006, S. 4851–4856. – URL <http://ieeexplore.ieee.org/document/4059186/>. – ISBN 1-4244-0258-1
- [Atev u. a. 2010] ATEV, Stefan ; MILLER, Grant ; PAPANIKOLOPOULOS, Nikolaos P.: Clustering of vehicle trajectories. In: *IEEE Transactions on Intelligent Transportation Systems* (2010). – ISBN 15249050
- [Bashir u. a. 2003] BASHIR, FI ; KHOKHAR, AA ; SCHONFELD, D.: Segmented trajectory based indexing and retrieval of video data. In: *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)* Bd. 3, URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.7768&rep=rep1&type=pdf>, 2003, S. II–623–6. – ISBN 0-7803-7750-8
- [Boykov und Kolmogorov 2004] BOYKOV, Yuri ; KOLMOGOROV, Vladimir: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In: *IEEE transactions on pattern analysis and machine intelligence* 26 (2004), Nr. 9, S. 1124–1137
- [Buzan u. a. 2004] BUZAN, Dan ; SCLAROFF, Stan ; KOLLIOS, George: Extraction and Clustering of Motion Trajectories in Video. (2004), apr. – URL <https://open.bu.edu/handle/2144/1545>
- [Chen u. a. 2011] CHEN, J ; WANG, R ; LIU, L ; , J S. ; ; CONTROL, Communications ; 2011 undefined: Clustering of trajectories based on Hausdorff distance. In: *ieeexplore.ieee.org* (2011). – URL <https://ieeexplore.ieee.org/abstract/document/6066483/>

- [Chen u. a. 2005] CHEN, Lei ; ÖZSU, M. T. ; ORIA, Vincent: Robust and fast similarity search for moving object trajectories. In: *Proceedings of the 2005 ACM SIGMOD international conference on Management of data - SIGMOD '05*. New York, New York, USA : ACM Press, 2005, S. 491. – URL <http://portal.acm.org/citation.cfm?doid=1066157.1066213>. – ISBN 1595930604
- [Chen u. a. 2014] CHEN, Zezhi ; YAN, Yuyao ; ELLIS, Tim: Lane detection by trajectory clustering in urban environments. In: *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, URL <https://ieeexplore.ieee.org/abstract/document/6958184/>, 2014, S. 3076–3081. – ISBN 9781479960781
- [Cookson u. a.] COOKSON, G ; RESEARCH, B Pishue I. ; FEBRUARY undefined ; 2017 undefined: Inrix global traffic scorecard. In: *jschulteis.de*. – URL http://jschulteis.de/wp-content/uploads/2018/02/INRIX_2017_Traffic_Scorecard_Report___German.pdf
- [FGSV 2008] FGSV: *RAA - Richtlinien für die Anlage von Autobahnen*. Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV), Arbeitsgruppe Straßenentwurf., 2008. – URL http://www.fgsv-verlag.de/catalog/product_info.php?products_id=2451. – ISBN 978-3-939715-51-1
- [FGSV 2012] FGSV: *RAL - Richtlinien für die Anlage von Landstraßen*. Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV), Arbeitsgruppe Straßenentwurf., 2012. – URL http://www.fgsv-verlag.de/catalog/product_info.php?products_id=3206. – ISBN 978-3-86446-039-5
- [Gao 2012] GAO, Jing: Clustering Lecture 4 : Density-based Methods. (2012), S. 1–16. – URL https://cse.buffalo.edu/~jing/cse601/fa12/materials/clustering_density.pdf
- [George Seif 2018] GEORGE SEIF: *The 5 Clustering Algorithms Data Scientists Need to Know*. 2018. – URL <https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68>. – Zugriffsdatum: 2018-10-19
- [Gopalan u. a. 2012] GOPALAN, Raghuraman ; HONG, Tsai ; SHNEIER, Michael ; CHELLAPPA, Rama: A Learning Approach Towards Detection and Tracking of Lane Markings. In: *Intelligent Transportation Systems , IEEE Transactions on* 13 (2012), Nr. 3, S. 1088–1098. – URL <https://ieeexplore.ieee.org/abstract/document/6155090/>. – ISBN 1524-9050
- [Hamerly u. a.] HAMERLY, G ; PROCESSING, C Elkan A. in neural information ; 2004 undefined: Learning the k in k-means. In: *papers.nips.cc*. – URL <http://papers.nips.cc/paper/2526-learning-the-k-in-k-means.pdf>
- [Hemmerle 2016] HEMMERLE, P: Empirische physikalische Eigenschaften des übersättigten innerstädtischen Verkehrs und Energieeffizienz von Fahrzeugen. (2016). – URL <https://d-nb.info/1122559259/34>

- [Hsieh u. a. 2006] HSIEH, J.-W. ; YU, S.-H. ; CHEN, Y.-S. ; HU, W.-F.: Automatic Traffic Surveillance System for Vehicle Tracking and Classification. In: *IEEE Transactions on Intelligent Transportation Systems* 7 (2006), jun, Nr. 2, S. 175–187. – URL <http://ieeexplore.ieee.org/document/1637673/>. – ISSN 1524-9050
- [Hu u. a. 2005] HU, Weiming ; FU, Zhouyu ; TAN, Tieniu: Similarity based vehicle trajectory clustering and anomaly detection. In: *Proceedings - International Conference on Image Processing, ICIP*, 2005. – ISBN 0780391349
- [Huttenlocher u. a. 1993] HUTTENLOCHER, DP ; . . . , GA Klanderman IEEE T. on ; 1993, Undefined: Comparing images using the Hausdorff distance. In: [ieeexplore.ieee.org](https://ieeexplore.ieee.org/abstract/document/232073/) (1993). – URL <https://ieeexplore.ieee.org/abstract/document/232073/>
- [Jain u. a. 1999] JAIN, A. K. ; MURTY, M. N. ; FLYNN, P. J.: Data clustering: a review. In: *ACM Computing Surveys* 31 (1999), sep, Nr. 3, S. 264–323. – URL <http://portal.acm.org/citation.cfm?doid=331499.331504>. – ISSN 03600300
- [Jain 2010] JAIN, Anil K.: Data clustering: 50 years beyond K-means. In: *Pattern Recognition Letters* 31 (2010), jun, Nr. 8, S. 651–666. – URL <https://www.sciencedirect.com/science/article/pii/S0167865509002323>. – ISSN 0167-8655
- [Junejo u. a. 2004] JUNEJO, Imran N. ; JAVED, Omar ; SHAH, Mubarak: Multi feature path modeling for video surveillance. In: *Proceedings - International Conference on Pattern Recognition* Bd. 2, IEEE, 2004, S. 716–719. – URL <http://ieeexplore.ieee.org/document/1334359/>. – ISBN 0769521282
- [Keogh und Pazzani 2000] KEOGH, Eamonn J. ; PAZZANI, Michael J.: Scaling up dynamic time warping for datamining applications. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00*, URL <https://dl.acm.org/citation.cfm?id=347153>, 2000, S. 285–289. – ISBN 1581132336
- [Kim 2008] KIM, Zu W.: Robust lane detection and tracking in challenging scenarios. In: *IEEE Transactions on Intelligent Transportation Systems* Bd. 9, 2008, S. 16–26. – ISBN 1524-9050
- [Lai und Yung 2000] LAI, Andrew H. ; YUNG, Nelson H.: Lane detection by orientation and length discrimination. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 30 (2000), Nr. 4, S. 539–548. – URL <https://ieeexplore.ieee.org/abstract/document/865171/>. – ISBN 1083-4419 (Print)
- [Lingras u. a. 2004] LINGRAS, P ; SYSTEMS, C West Journal of Intelligent I. ; 2004, Undefined: Interval Set Clustering of Web Users with Rough K-Means. In: *Springer - Journal of Intelligent Information Systems* 23 (2004), Nr. 1, S. 5–16. – URL <https://link.springer.com/article/10.1023/B:JIIS.0000029668.88665.1a>

- [Makris und Ellis 2005] MAKRIS, Dimitios ; ELLIS, Tim: Learning semantic scene models from observing activity in visual surveillance. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 35 (2005), Nr. 3, S. 397–408. – URL <https://ieeexplore.ieee.org/abstract/document/1430826/>. – ISBN 1083-4419
- [McCall und Trivedi 2006] McCALL, J.C. ; TRIVEDI, M.M.: Video-Based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation. In: *IEEE Transactions on Intelligent Transportation Systems* 7 (2006), mar, Nr. 1, S. 20–37. – URL <http://ieeexplore.ieee.org/document/1603550/>. – ISSN 1524-9050
- [Mélo u. a. 2006] MÉLO, José ; NAFTEL, Andrew ; BERNARDINO, Alexandre ; SANTOS-VICTOR, José: Detection and classification of highway lanes using vehicle motion trajectories. In: *IEEE Transactions on Intelligent Transportation Systems* (2006). – ISBN 1524-9050
- [Meng u. a. 2018] MENG, Fanrong ; YUAN, Guan ; LV, Shaoqian ; WANG, Zhixiao ; XIA, Shixiong: *An overview on trajectory outlier detection*. feb 2018. – URL <http://link.springer.com/10.1007/s10462-018-9619-1>
- [Mirge u. a. 2017] MIRGE, Vaishali ; VERMA, Kesari ; GUPTA, Shubhrata: Outlier Detection in Vehicle Trajectories. In: *International Journal of Computer Applications* 171 (2017), Nr. 8, S. 1–6. – URL <https://www.ijcaonline.org/archives/volume171/number8/mirge-2017-ijca-915139.pdf>
- [Morris und Trivedi 2009] MORRIS, Brendan ; TRIVEDI, Mohan: Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2009, S. 312–319. – URL <http://ieeexplore.ieee.org/document/5206559/>. – ISBN 978-1-4244-3992-8
- [Morris und Trivedi 2011] MORRIS, Brendan T. ; TRIVEDI, Mohan M.: Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (2011), Nr. 11, S. 2287–2301. – URL <https://ieeexplore.ieee.org/abstract/document/5740921/>. – ISBN 1939-3539 (Electronic) \n0098-5589 (Linking)
- [Ng u. a. 2002] NG, A Y. ; JORDAN, M I. ; WEISS, Y: On spectral clustering: Analysis and an algorithm. In: *Advances in Neural Information Processing Systems 14, Vols 1 and 2* 14 (2002), S. 849–856. – ISBN 1049-5258
- [Pelleg u. a.] PELLEG, D ; ICML, AW M. ; 2000 undefined: X-means: Extending k-means with efficient estimation of the number of clusters. In: *cs.uef.fi*. – URL <http://cs.uef.fi/~zhao/Courses/Clustering2012/Xmeans.pdf>
- [Piciarelli und Foresti 2006] PICIARELLI, C ; FORESTI, G. L.: On-line trajectory clustering for anomalous events detection. In: *Pattern Recognition Letters* 27 (2006), Nr. 15, S. 1835–1842. – URL <https://www.sciencedirect.com/science/article/pii/S0167865506000432>. – ISSN 01678655

- [Ren u. a. 2014] REN, Jianqiang ; CHEN, Yangzhou ; XIN, Le ; SHI, Jianjun: Lane Detection in Video-Based Intelligent Transportation Monitoring via Fast Extracting and Clustering of Vehicle Motion Trajectories. In: *Mathematical Problems in Engineering* (2014). – ISSN 15635147
- [Tan u. a. 2007] TAN, Pang-Ning u. a.: *Introduction to data mining*. Pearson Education India, 2007
- [Vlachos u. a. 2002] VLACHOS, Michail ; KOLLIOS, George ; GUNOPULOS, Dimitrios: Discovering similar multidimensional trajectories. In: *Proceedings - International Conference on Data Engineering* (2002), S. 673–684. – URL <https://ieeexplore.ieee.org/abstract/document/994784/>. – ISBN 0-7695-1531-2
- [Weiming Hu u. a. 2006] WEIMING HU ; XUEJUAN XIAO ; ZHOUYU FU ; XIE, D. ; TIENIU TAN ; MAYBANK, S.: A system for learning statistical motion patterns. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (2006), sep, Nr. 9, S. 1450–1464. – URL <http://ieeexplore.ieee.org/document/1661547/>. – ISSN 0162-8828
- [Xie und Beni 1991] XIE, Xuanli L. ; BENI, Gerardo: A validity measure for fuzzy clustering. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* (1991), Nr. 8, S. 841–847