Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 815

Udo Feuerhake

Erfassung von Trajektorien und Erkennung von Bewegungsmustern

München 2018

Verlag der Bayerischen Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5227-7

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Universität Hannover
ISSN 0174-1454, Nr. 340, Hannover 2018

Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 815

Erfassung von Trajektorien und Erkennung von Bewegungsmustern

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

Vorgelegt von

M.Sc. Udo Feuerhake

Geboren am 07.09.1982 in Hameln

München 2018

Verlag der Bayerischen Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5227-7

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Universität Hannover
ISSN 0174-1454, Nr. 340, Hannover 2018

Adresse der DGK:

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften (DGK)

Alfons-Goppel-Straße 11 ● D – 80 539 München

Telefon +49 – 331 – 288 1685 ● Telefax +49 – 331 – 288 1759

E-Mail post@dgk.badw.de ● http://www.dgk.badw.de

Prüfungskommission:

Vorsitzender: Prof. Dr.-Ing. habil. Jürgen Müller

Referentin: Prof. Dr.-Ing. habil. Monika Sester

Korreferenten: Prof. Dr.-Ing. habil. Christian Heipke
Prof. Dr. Robert Weibel (Universität Zürich)

Tag der mündlichen Prüfung: 01.03.2018

ISSN 0065-5325 ISBN 978-3-7696-5227-7

Abstract

With the ever increasing performance of sensors and data processing, it is now possible to capture and analyze the movements of objects in great detail. In the context of the analysis of those movements, which is based on trajectories, movement patterns play an important role as they show typical movement behavior. This valuable information can be used in various application areas, e.g. to detect congestions in the traffic context, to analyze the behavior of animals during their observation and to identify different tactics in various sports.

However, since the recognition of patterns requires correspondingly large amounts of data, which are generally not freely available, a new approach to capture trajectories is presented in this work. This apporach consists a combination of GNSS and camera tracking. In this way, the high reliability of GNSS tracking and the high accuracy of a camera tracking are used to record high-quality motion data that meets the requirements of many application scenarios, even when low-cost sensors are used. The combination of different sensor technologies, which provide different information with corresponding uncertainties, requires a fusion of heterogeneous data. For this purpose, an approach is developed that models this problem as a *Hidden Markov Model* in two alternative variants. In both case, the *Viterbi algorithm* is used to determine the most likely sequence of states that is used to generate the object trajectories.

Further, a new method to recognize *repetitive a priori unknown* movement patterns is presented that is based on the common procedure of the *knowledge discovery from data (bases)* process. In order to identify the patterns, two alternative methods are described. Using a clustering-based approach, similar trajectory segments that correspond to each instance of a pattern are determined. In this case, a preceding segmentation is required. Depending on the application scenario and the available contextual knowledge, the segmentation is based on identified *interesting places* or on a semantic information provided by *machine learning* techniques. The alternative sequence-based approach identifies repetitive subsequences in the motion sequences of the objects. The alternative sequence-based approach determines repetitive subsequences in the motion sequences of the objects. The latter are generated from the trajectories taking into account the desired transformation invariants of the patterns. Using sequence analysis methods, individual and group movement patterns are extracted from the data.

The two presented methods are evaluated in various experiments. The object tracking is applied to real image sequences and the results are verified using reference data. The discussion shows that the proposed method correctly continues the tracking when scenes with a significant number of object occlusions have been resolved. The verification of the results of the pattern recognition approach shows that, depending on the selected invariance, all reference patterns are recognized. In further experiments, the approach is applied to real data sets from different contexts. In this way, the different analysis possibilities with regard to the later use of the patterns, e.g. the characterization of the observed objects or the prediction of future movements, are demonstrated.

Keywords: object tracking, data fusion, trajectory, pattern recognition, sequence analysis, machine learning, data mining.

Kurzfassung

Mit der stetig steigenden Leistungsfähigkeit von Sensoren und Datenverarbeitung ist es heutzutage möglich, Bewegungen von Objekten sehr detailgetreu zu erfassen und zu analysieren. Die Bewegungen werden in Form von Trajektorien erfasst und verarbeitet. Im Zusammenhang mit der Analyse der Objektbewegungen spielen Bewegungsmuster eine wichtige Rolle, da sie typisches Bewegungsverhalten aufzeigen. Diese wertvollen Informationen können in verschiedenen Anwendungsbereichen, bspw. im Verkehrskontext zur Stauerkennung, bei der Beobachtung von Tieren zur Verhaltensanalyse und im Sport zur Identifikation von unterschiedlichen Strategien, genutzt werden.

Zur Erfassung der Trajektorien werden sowohl GNSS-basierte als auch Kamera-basierte Ansätze genutzt, die jeweils Vor- und Nachteile aufweisen. In der Arbeit wird ein neuer Ansatz vorgeschlagen, der die hohe Verlässlichkeit eines GNSS-Trackings mit der hohen Genauigkeit eines Kamera-Trackings kombiniert. Damit wird ermöglicht, auch mit Low-cost-Sensoren qualitativ hochwertige Bewegungsdaten zu erfassen, die die Anforderungen vieler Anwendungsszenarien erfüllen. Zur Fusion der beiden Datenquellen unter Berücksichtigung ihrer Unsicherheiten wird ein Ansatz entwickelt, der dieses Problem als *Hidden Markov Modell* in zwei alternativen Varianten modelliert. In beiden Fällen werden abschließend mit Hilfe des *Viterbi-Algorithmus* die wahrscheinlichsten Sequenzen von Zuständen ermittelt, die zur Generierung der Objekt-Trajektorien verwendet werden.

Zur Erkennung von *wiederkehrenden a-priori unbekannten* Bewegungsmustern wird ein neues Verfahren präsentiert, das sich an dem üblichen Vorgehen des *Knowledge Discovery from Data(bases)*-Prozess orientiert. Bei der Identifikation der Muster werden zwei alternative Varianten beschrieben. Mit Hilfe eines Clustering-basierten Ansatzes werden ähnliche Trajektoriensegmente, die den einzelnen Instanzen eines Musters entsprechen, bestimmt. Diese Methode setzt eine vorangehende Segmentierung voraus. Je nach Anwendungsszenario und verfügbarem Kontextwissen kann diese auf Basis identifizierter *interessanter Plätze* oder anhand einer durch Verfahren des *Maschinellen Lernens* unterstützte semantische Segmentierung erfolgen. Der alternative sequenzbasierte Ansatz ermittelt wiederkehrende Teilsequenzen in den Bewegungssequenzen der Objekte. Diese Sequenzen werden, unter Berücksichtigung der gewünschten Transformationsinvarianzen der Muster, aus den Trajektorien generiert. Durch die Anwendung von Sequenzanalyseverfahren können auf diese Weise Einzel- und auch Gruppenbewegungsmuster erkannt werden.

In verschiedenen Experimenten werden die beiden vorgestellten Verfahren evaluiert. Das Objekt-Tracking wird dazu auf reale Bildsequenzen angewendet. Die daraus hervorgehenden Ergebnisse werden anhand von Referenzdaten verifiziert. In der sich anschließenden Diskussion wird deutlich, dass das vorgestellte Verfahren Objekte auch dann korrekt verfolgt, wenn diese zeitweise durch Verdeckungen nicht detektiert wurden. Die Verifikation der Ergebnisse des Mustererkennungsansatzes zeigt, dass bei jeder gewählten Invarianz sämtliche Referenzmuster erkannt werden. In weiteren Experimenten wird der Ansatz auf reale Datensätzen aus unterschiedlichen Kontexten angewendet. Dabei werden die unterschiedlichen Analysemöglichkeiten auch im Hinblick auf die spätere Verwendung der Muster zur Charakterisierung der beobachteten Objekte und zur Prädiktion von zukünftigen Bewegungen aufgezeigt.

Schlagworte: Objekt-Tracking, Daten-Fusion, Trajektorie, Mustererkennung, Sequenzanalyse, Maschinelles Lernen, Data Mining.

Inhaltsverzeichnis

1 Einleitung

1.1 Hintergrund und Problemstellung

Heutzutage sind Sensorik und Datenverarbeitung soweit fortgeschritten, dass problemlos große Mengen an qualitativ hochwertigen Daten erfasst, gespeichert und analysiert werden können. In vielen Forschungsbereichen und Anwendungen wird dieses genutzt, um durch eine Analyse von gesammelten Daten neue Erkenntnisse zu erlangen. Dieses gilt auch bei der Beobachtung von sich bewegenden Objekten. So wird zum Beispiel in dem Bereich der Stadt- bzw. Verkehrsplanung der Verkehr inklusive der unterschiedlichen Teilnehmer erfasst. Die Daten der Kraftfahrzeuge, Radfahrer und Fußgänger werden daraufhin für unterschiedliche Zwecke verwendet. Zum Beispiel kann der Verkehrsfluss im Straßennetz untersucht werden. Dabei können Problemstellen identifiziert werden, die bei höherem Verkehrsaufkommen für Behinderungen sorgen. Dieses Wissen ermöglicht zielgerichtete Maßnahmen, wie bspw. eine veränderte Verkehrsführung oder Neubauten, um Abhilfe zu schaffen. Auch Anbieter von Navigationssystemen und -diensten nutzen die Bewegungsdaten von Fahrzeugen (auch als *Floating Car Data* bezeichnet) für eine Reihe von automatischen Verfahren zur Aktualisierung des vorhandenen Kartenmaterials und zur Erweiterung des Angebots an zusätzlichen Diensten. Zu Letzteren zählt die automatische Stauerkennung, wie in Abbildung 1.1 (links) gezeigt wird, sowie eine Detektion von dauerhaften oder temporären Änderungen der Verkehrsführung, welche bspw. durch Sperrungen oder baustellenbedingten Behinderungen hervorgerufen werden. Auch im Marketing-Bereich gelten Bewegungsdaten von Personen im Verkehr oder in Kaufhäusern als wertvolle Informationen. So geben sie bspw. Hinweise darauf, an welchen Stellen Werbung und Produkte effektiv platziert werden können. Eine weitere Domäne ist die Erforschung von Tieren bzw. deren Verhalten. Da eine herkömmliche Beobachtung mit dem bloßem Auge oder einem Fernglas insbesondere bei Tieren in freier Wildbahn schwierig ist, werden auch dort mit Hilfe von Tracking-Verfahren (siehe Abbildung 1.1, rechts) Bewegungsdaten gesammelt und analysiert. Dabei wird versucht, über Rückschlüsse auf Grundlage der Bewegungen Verständnis für das Verhalten der Tiere zu entwickeln.

Abbildung 1.1: Links: Navigationsdienstleister nutzen Bewegungsdaten zur automatischen Stauerkennung. Der TomTom-MyDrive-Routenplaner[1] gibt einen Überblick über die aktuelle Verkehrssituation. Rechts: Zur Erfassung ihrer Bewegungen werden Tiere wie diese Seemöwe[2] mit GPS-Loggern ausgestattet.

[1] Link und Bildquelle: TomTom: https://mydrive.tomtom.com/de_de/#mode=viewport+viewport=50.93448,6.94523,9.88, Zugriff am 27.11.2017

[2] Bildquelle: http://1.bp.blogspot.com/-Xl0zfHCJpzs/TfXM9kqXweI/AAAAAAAACvs/f7p5MwZeJtY/s320/DSC_1516.JPG, Zugriff am 27.11.2017

Gleiches gilt für den Sport. Insbesondere bei populären Sportarten wie Fußball, American Football, Eis-/Hockey und Basketball werden die Bewegungen und Aktionen der Spieler analysiert, um so deren Leistungen zu messen und Informationen über deren typisches Verhalten in speziellen Situationen zu sammeln. Diese Informationen werden von den involvierten Akteuren, also den Sportlern bzw. deren Trainern aber auch von den Medien, die über das Ereignis berichten, jeweils für ihre Zwecke genutzt. Die Sportler und Trainer verwenden die Informationen, um die Leistungen zu bewerten sowie um bestehende Stärken und Schwächen zu erkennen. Häufig werden die Informationen auch für die Vorbereitung auf den nächsten Wettkampf genutzt. Dabei werden Daten über den Gegner ausgewertet, sofern sie vorliegen, um so Erkenntnisse über dessen Stärken bzw. Schwächen, Taktik oder wieder typisches Verhalten zu gewinnen. Die Medien hingegen nutzen die verfügbaren Daten, um die Attraktivität ihrer Berichterstattung zu steigern, indem Sie, mitunter in Echtzeit (live), interessante Statistiken und Details zum aktuellen Geschehen liefern. Siehe hierzu die Beispiele in Abbildung 1.2.

(a) *(b)*

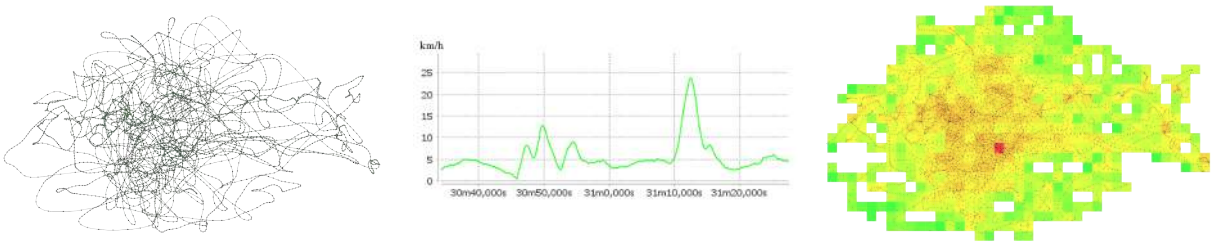
Abbildung 1.2: Dienstleister wie Deltatre und Opta erfassen und analysieren die Bewegungen der Spieler, um auf diese Weise die Medien mit Analysen und Statistiken zu versorgen. Die Ergebnisse sind für sämtliche Spieler in Echtzeit verfügbar und können so live (a)[1] oder im Nachgang präsentiert werden (b)[2].

Bei den in den verschiedenen Anwendungsgebieten gesammelten Informationen spielen die Bewegungsdaten eine wichtige Rolle. In Form von Trajektorien (siehe Abbildung 1.3a) geben diese Daten Aufschluss über den Verlauf der Aufenthaltsorte der Objekte über die Zeit. In vielen Anwendungen stellen sie die Basis für Analysen dar. Häufig dienen sie zur Berechnung von Kennzahlen, Bewegungsprofilen (Abbildung 1.3b) oder sogenannten Heatmaps, die die räumliche Verteilung der Objektaufenthaltsorte zeigen (Abbildung 1.3c). Durch diese grundlegenden Analysen können Fragen in der Art, „Wo befindet sich das Objekt?“, „Wo war es zuvor?“, „In welche Richtung und wie schnell bewegt es sich?“ beantwortet werden. Ist es jedoch das Ziel, detaillierte Informationen über das allgemeine oder situationsbedingte Bewegungsverhalten eines Objekts oder über das Objekt selbst zu erhalten, so sind komplexere Analysen erforderlich. Eine Möglichkeit ist es, die Daten nach Mustern in Form von sich wiederholenden Bewegungen zu durchsuchen. Die Wiederholungen der Bewegungen geben einen Hinweis darauf, dass es sich um typisches Bewegungsverhalten des beobachteten Objekts handeln könnte.

Die Kenntnis darüber, dass diese Bewegungsmuster existieren, und die Informationen über das Verhalten, das diese Muster mit sich tragen, können für unterschiedliche Zwecke genutzt werden.

[1] Bildquelle: Opta sports, http://www.optasports.com/ImageGen.ashx?image=/media/948406/broadcast-image.png&width=597&height=238, Zugriff am 27.11.2017

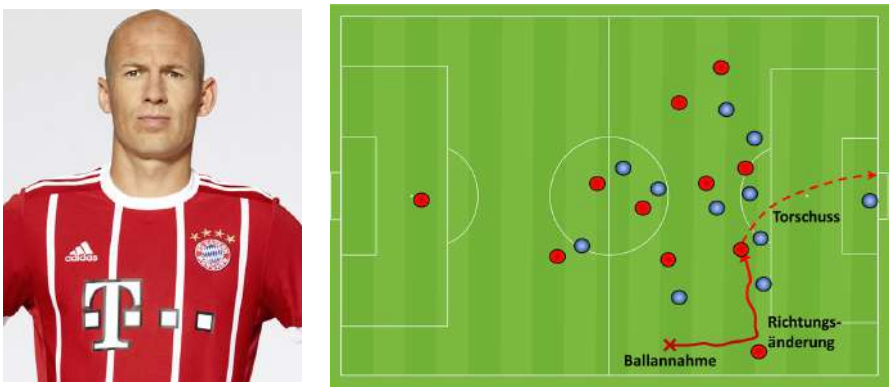
[2] Bildquelle: Deltatre AG, http://www.bundesliga-datenbank.de/typo3temp/pics/Impire_Printfertige_Grafiken_Beispielausgabe_Sonntag_Hamburger_Morgenpost-1_b8e33cdea2.jpg, Zugriff am 27.11.2017



(a) *(b)* *(c)*

Abbildung 1.3: a) Die Analyse der Objektbewegung basiert im Allgemeinen auf Trajektorien. b) Bewegungsprofile geben bspw. Aufschluss über den Verlauf der Geschwindigkeit des Objekts über die Zeit. c) Heatmaps zeigen die räumliche Verteilung der Objektpositionen. Durch die Farbskala von grün (selten) bis rot (häufig) ist die Häufigkeit der Besuche des Gebiets angegeben.

Sie können nach Han u. a. (2011) einerseits zur Charakterisierung des Objekts andererseits zur Prädiktion zukünftiger Bewegungen verwendet werden. Ein eingängiges Beispiel hierfür liefert die Analyse des Fußballspielers Arjen Robben (Abbildung 1.4a). Der Niederländer steht aktuell beim Verein FC Bayern München unter Vertrag und ist für seine temporeichen Dribblings mit dem Ball, seinen guten Torabschluss und damit für zahlreiche Tore bekannt. Der Weg bis zum Torerfolg, den dieser Spieler einschlägt, ist inzwischen allerdings genauso bekannt wie er selbst. Der sogenannte „Robben-Move“ (Aleythe, 2017) gestaltet sich wie in Abbildung 1.4b dargestellt: Positionsbedingt erhält er häufig auf der Außenseite in der Nähe zum Strafraum den Ball. Einem kurzen Lauf entlang der Außenlinie folgt ein schneller Richtungswechsel, sodass er sich Richtung Spielfeldmitte parallel zur Strafraumgrenze bewegt und dabei eine geeignete Lücke für einen Torschuss sucht und nicht selten findet. Da sich diese Aktion regelmäßig wiederholt, kann sie als typisch bzw. als Muster angesehen werden. Dieses Muster könnte nun, wie zuvor beschrieben, zur Charakterisierung und Prädiktion genutzt werden. Im Zusammenhang mit Prädiktion bedeutet dies, dass die Gegenspieler, nachdem sie sich auf Arjen Robben vorbereitet haben, ihn und dieses Muster kennen und daher wissen, wie und wohin er sich typischer Weise bewegt. Theoretisch könnten sie ihn so besser verteidigen. Bei der Charakterisierung verhält es sich prinzipiell anders herum. Dort könnte man bspw. herausfinden, dass es sich bei einem unbekannten Spieler um Arjen Robben handelt, wenn man eben dieses Muster in dessen Bewegungen entdeckt. Diese Art der Analyse ist natürlich auch auf die anderen Anwendungsgebiete, bei denen Objekte beobachtet werden, übertragbar.



(a) *(b)*

Abbildung 1.4: Arjen Robben[2] (a) und sein typischer Weg zum Torerfolg (b).

[2]Bildquelle: https://static.foba1.com/bilder/spieler/gross/3936.jpg, Zugriff am 27.11.2017.

Eine Erkennung von relevanten und sich nicht zufällig ergebenden Bewegungsmustern erfordert einen Zugriff auf einen gewissen Umfang an Bewegungsdaten. Auf Grund von Restriktion, z.B. durch die Einhaltung der Privatsphäre der beobachteten Objekte, sind diese Daten in dem Umfang und der erforderlichen Qualität im Allgemeinen nicht frei verfügbar. Sie müssen daher selbst erfasst werden.

Auf diese Weise ergeben sich zwei Probleme, die im Rahmen dieser Arbeit bearbeitet werden:

- Erfassung der Trajektorien der zu analysierenden Objekte
- Erkennung von Bewegungsmustern in den resultierenden Trajektorien

1.2 Ziel der Arbeit

Der aus dieser Motivation resultierende Prozess zur Erkennung von Bewegungsmustern ist in Abbildung 1.5 dargestellt. Darauf basierend ist es das Ziel dieser Arbeit, Lösungsansätze für die genannten zwei wichtigen Probleme (rot eingerahmt) zu finden und diese umzusetzen. Die Problemstellungen werden in den folgenden Abschnitten näher beschrieben und von verwandten Problemen abgegrenzt.

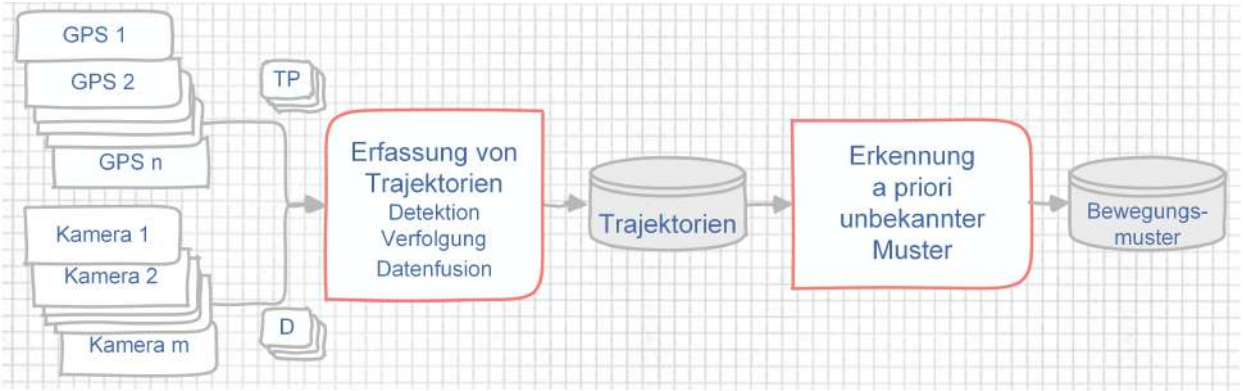


Abbildung 1.5: Die in dieser Arbeit bearbeiteten Problemstellungen: die Erfassung von Trajektorien und die Erkennung von Bewegungsmustern (D: Objektdetektion im Kamerabild, TP: Trajektorie-Punkt).

1.2.1 Problemstellung: Erfassung von Trajektorien

Die erste Forschungsfrage, die mit Hilfe des zu entwickelnden Verfahrens beantwortet werden soll, ist:

1) Lässt sich durch die Kombination aus GNSS- und videobasiertem Tracking eine Objekt-Tracking-Lösung verwirklichen, die trotz der Verwendung von Low-cost-Sensoren und insbesondere im Hinblick auf Objektverdeckungen vollständige und geometrisch genaue Trajektorien liefert?

Im Allgemeinen erfolgt eine Analyse der Objektbewegungen auf Basis von Trajektorien. Liegen diese nicht bereits vor, müssen sie erfasst werden, indem die Objekte detektiert bzw. lokalisiert und über die Zeit verfolgt (*engl. to track*) werden. Im Zusammenhang mit der Umsetzung eines automatisierten *Objekt-Tracking*-Verfahrens gilt es daher generell, die zwei folgenden Teilprobleme zu lösen.

- Erfassung und Lokalisierung der Objekte (Detektion)

– Zuordnung der einzelnen Detektionen zu Objekt-Trajektorien (Verfolgung)

Zwei in diesem Zusammenhang häufig genutzte Technologien sind GNSS- und videobasierte Tracking-Verfahren. Allerdings weisen beide gewisse Schwächen auf, sodass sie in Anwendungsfällen, die sehr genaue und auch durchgängige Trajektorien benötigen, jeweils nicht als eigenständige Lösung geeignet sind. Während eine GNSS-Tracking-Lösung kontinuierliche Trajektorien der Objekte liefert, allerdings unter der typischen Ungenauigkeit einer GNSS-Positionierung leidet, liefern Kameras auf der anderen Seite höhere geometrische Genauigkeiten, sind aber anfällig im Bezug auf Verdeckungen von Objekten, welche zu unterbrochenen oder unvollständigen Trajektorien führen. Eine mögliche Herangehensweise wäre die Verwendung von entweder Multi-Kamera-Systemen, die anhand verschiedener Perspektiven Objektverdeckungen auflösen, oder funkbasierten Echtzeit-Lokalisierungssystemen *(Real-Time Locating Systems, RTLS)*, bei denen die beobachteten Objekte durch das Mitführen eines Senders/Empfängers innerhalb eines fest installierten Systems unterschieden werden. Diese Alternativen sind jedoch häufig durch technische und finanzielle Rahmenbedingungen nicht verwendbar. (Die unterschiedlichen Technologien werden im Abschnitt 2.2.3 des Grundlagenkapitels beschrieben.)

In dieser Arbeit wird daher ein Ansatz entwickelt, der das GNSS- und videobasierte Tracking kombiniert, um auf diese Weise die jeweiligen Stärken zu nutzen und gleichzeitig die Schwächen zu reduzieren. Durch diese Kombination müssen heterogene Sensordaten in Form von Objektdetektionen in Kamerabildern und GNSS-Informationen zusammengeführt werden. Den auf das Objekt-Tracking bezogenen Teilproblemen schließt sich damit das Problem der Fusion heterogener Sensordaten an. Der zu entwickelnde Ansatz muss daher die folgenden Teilprobleme lösen:

– Erfassung und Lokalisierung der Objekte (Detektion)

– Zuordnung der einzelnen Detektionen zu Objekt-Trajektorien (Verfolgung)

– Fusion unterschiedlicher Sensordaten mit ihren jeweiligen Eigenheiten

1.2.2 Problemstellung: Erkennung von Bewegungsmustern in Trajektorien

Die zweite Forschungsfrage lautet:

2) Wie lassen sich unbekannte wiederkehrende Bewegungsmuster in Trajektorien erkennen?

Im Rahmen dieser Arbeit wird weiterhin ein Ansatz zur Erkennung von Bewegungsmustern in Trajektorien entwickelt. Das Wort „Muster“ (auch *Pattern*) lässt sich nach dem Duden als Vorlage bzw. etwas Symbolhaftes oder Schablonenartiges interpretieren, von dem einzelne Objekte (Instanzen) abgeleitet werden können. Übertragen auf den Kontext der Bewegungsanalyse wird durch ein Muster demnach eine gewisse Struktur oder Form definiert, dem die Objekte in ihren Bewegungen folgen. Ist eine gesuchte Musterbewegung als solche *a priori bekannt*, muss zur Erkennung nach entsprechenden Instanzen in den Bewegungsdaten gesucht werden. Dieses Vorgehen könnte prinzipiell als „*Pattern Matching*“ angesehen werden, für das eine Möglichkeit der Beschreibung des Musters und ein Verfahren, um diese anhand der Beschreibung aus den Daten zu extrahieren, erforderlich ist. Beispiele dafür sind die zahlreichen Ansätze zur Erkennung von Gruppenbewegungsmustern (vgl. Abschnitt 3.2). Dort sind die Muster bekannt und können so in den Daten gesucht werden. In diese Richtung sind diverse Ansätze entwickelt worden, die diese Aufgabe erfüllen. In vielen Anwendungsfällen sind jedoch die gesuchten Muster zu Beginn nicht bekannt, sodass die bestehenden Ansätze hier nicht zur Lösung des Problems beitragen können.

Ein im Kontext der Bewegungsmustererkennung weniger erforschtes Problem ergibt sich dann, wenn die Definition des Musters *a priori unbekannt* ist. Das heißt, es wird z.B. nach dem ty-

pischen Verhalten eines Objekts gesucht, ohne zu wissen, wie dieses tatsächlich aussieht. Dabei wird entsprechend davon ausgegangen, dass sich typische Verhaltensweisen durch wiederkehrende Bewegungen auszeichnen. In diesem Fall funktioniert das zuvor beschriebene Vorgehen, bei dem die Instanzen anhand der Beschreibung des Musters gesucht werden, nicht. Unter der Annahme, dass Muster dennoch durch entsprechende Instanzen in den Daten vertreten sind, sind hier wiederkehrende gleiche bzw. ähnliche Instanzen zu identifizieren und von diesen ein entsprechendes Muster abzuleiten. Es handelt sich daher eher um ein „*Pattern Mining*“. Da Objektbewegungen im Kontext dieser Arbeit durch Trajektorien beschrieben werden, wird daher ein Ansatz entwickelt, der in der Lage ist, wiederkehrende Segmente in den Trajektorien zu erkennen und damit *a priori unbekannte* Bewegungsmuster zu identifizieren. Die Objekte selbst werden dabei zu sich bewegenden Punkten abstrahiert, da dort weder ihre Gestalt, Haltung oder Aussehen noch die relativen Bewegungen von bspw. Körperteilen eine Rolle spielen.

1.3 Gliederung

Nach obiger Einleitung und Beschreibung der Problemstellungen gliedert sich der verbleibende Teil dieser Arbeit wie folgt: In Kapitel 2 werden die benötigten Grundlagen beschrieben. Neben einer allgemeinen Beschreibung, wie Objektbewegungen mit Hilfe von Trajektorien modelliert werden, sind dort auch die grundlegenden Methoden in Bezug auf das Erfassen dieser Trajektorien und zur Erkennung von Bewegungsmustern zu finden.

In dem sich anschließenden Kapitel 3 wird der Stand der Forschung beschrieben. Dazu werden die in der Literatur existierenden Arbeiten vorgestellt, die in Bezug auf die Problemstellungen dieser Arbeit relevant sind.

In dem ersten der beiden Hauptteile dieser Arbeit, dem Kapitel 4, wird ein neues Verfahren zur Erfassung der Objekt-Trajektorien vorgestellt, welches die Antwort auf die erste Forschungsfrage ermöglicht. In den einzelnen Abschnitten dieses Kapitels werden sowohl die Gesamtstruktur des Verfahrens als auch dessen einzelne Schritte detailliert erläutert. Dabei werden die verwendeten Sensoren zusammen mit deren jeweiligen Daten, die nötigen Vorverarbeitungen und die eigentliche Fusion dieser heterogenen Daten beschrieben. Zudem werden die Laufzeit des Algorithmus und das Systemdesign veranschaulicht.

Der Lösungsansatz für das zweite Kernproblem, dem Erkennen von Mustern in Trajektorien, wird in Kapitel 5 vorgestellt. Nach einer formalen Definition von Bewegungsmustern wird auch hier zunächst ein Überblick über das gesamte Vorgehen gegeben. Dessen Struktur orientiert sich am allgemeinen *Knowledge Discovery from Data(bases)* (*KDD*)[1]-Prozess. Die nachfolgenden Abschnitte schildern daher die einzelnen Phasen dieses Vorgehens, sodass zuerst die Vorverarbeitungen und später die eigentliche Extraktion der Muster ausführlich erläutert wird.

In den beiden Kapiteln 6 und 7 werden die in dieser Arbeit eingeführten Lösungsansätze anhand von unterschiedlichen Experimenten evaluiert. Dabei wird zunächst jeweils eine Übersicht über die in diesem Zusammenhang verwendete und eigens entwickelte Software gegeben. Im Folgenden werden die entsprechenden Ergebnisse der Ansätze diskutiert.

Diese Arbeit schließt mit einer Zusammenfassung und einem Ausblick über mögliche zukünftige Anwendungsmöglichkeiten bzw. Erweiterungen der Ansätze (Kapitel 8).

[1] Ein auch als *Data-Mining* bekannter Prozess zur Extraktion von Wissen aus großen Datenmengen. Eine Beschreibung dieses Prozesses ist im Grundlagen-Kapitel (Abschnitt 2.3.1) enthalten.

2 Grundlagen

In diesem Kapitel werden die Grundlagen für die in dieser Arbeit vorgestellten Ansätze zur Erfassung von Trajektorien und zur Erkennung von Bewegungsmustern beschrieben. Der Abschnitt 2.1 setzt sich dazu mit der formalen Modellierung von Objektbewegungen anhand von Trajektorien auseinander. Dabei wird auch auf die unterschiedlichen Bewegungsräume eingegangen, die bei einer Analyse der Trajektorien berücksichtigt werden müssen. Im darauffolgenden Abschnitt 2.2 werden die nötigen Grundlagen zur Erfassung von Trajektorien mit unterschiedlichen Sensortechnologien dargelegt. Weil in diesem Zusammenhang die Lokalisierung und Verfolgung der Objekte über die Zeit eine wichtige Rolle spielen, werden jeweils die dafür relevanten Verfahren detailliert erläutert. Da sowohl die Lokalisierung als auch die Verfolgung der Objekte mit Unsicherheiten behaftet sind, wird zudem das Konzept von Graphischen Modellen als Möglichkeit der stochastischen Modellierung erläutert. Der Abschnitt 2.3 umfasst das benötigte Grundlagenwissen in Bezug auf die Bewegungsmustererkennung. Hier werden das konzeptionelle Vorgehen, die erforderliche Vorverarbeitung und die Identifikation von ähnlichen Trajektorien anhand verschiedener Methoden aus dem Bereich des Maschinellen Lernens erklärt. Des Weiteren werden die Grundlagen bezogen auf die Sequenzmustererkennung, die eine alternative Herangehensweise bietet, beschrieben. In den beiden letzten Abschnitten werden mit der dynamischen Programmierung (2.4) und der dezentralen Verarbeitung von Daten (2.5) zwei Möglichkeiten dargestellt, die eine effiziente Umsetzung der Algorithmen bzw. des Gesamtsystems ermöglichen.

2.1 Modellierung von Objektbewegungen

Die Grundlage für eine Bewegungsanalyse ist die Beschreibung von Bewegung, welche sich durch die Positionsänderung eines Objekts über die Zeit definiert. Die kontinuierlichen Bewegungen können als Funktion von Positionen über die Zeit t beschrieben werden. Im 2-dimensionalen (2D) bzw. 3-dimensionalen Raum (3D) gilt:

$$2D: \ P(t) = (x,y), \ \ 3D: \ P(t) = (x,y,z) \tag{2.1}$$

Die kontinuierlichen Bewegungen können jedoch mit heutigen Verfahren zur Bewegungsaufzeichnung, auch *Tracking-Verfahren* (siehe Abschnitt 2.2) genannt, nicht erfasst werden. Die Bewegungen können nur mit einer gewissen Frequenz (*sampling rate*) abgetastet werden. Die sich daraus ergebende Approximation ist eine diskrete Menge an Positionen P über die Zeit, auch *fixes* genannt. Für den 3D-Fall gilt:

$$P_t = (x,y,z)_t \tag{2.2}$$

Werden die jeweiligen Tupel an Positionskoordinaten um den Zeitpunkt des fixes t erweitert, so entstehen die Vierer-Tupel (x,y,z,t), welche sich chronologisch ordnen lassen. Eine chronologisch geordnete Folge dieser Vierer-Tupel, im Folgenden *Trajektoriepunkte* (TP) genannt, entspricht der Trajektorie (T), oder auch Spur, eines Objekts.

$$T = \{TP_0,TP_1,...,TP_k\} = \{(x_0,y_0,z_0,t_0),(x_1,y_1,z_1,t_1),...,(x_k,y_k,z_k,t_k)\} \tag{2.3}$$

TP_i entspricht hierbei dem i-ten von k Trajektoriepunkten einer Trajektorie. Diese Trajektoriepunkte können neben den Positionsinformationen auch zusätzliche semantische Informationen be-

sitzen. Dies können u.a. Informationen über das Objekt selbst, bspw. eine Objektkennung oder der Objekttyp, oder den aktuellen Fortbewegungsmodus (z.B. zu Fuß, Fahrrad, Auto, usw.) sein. Der ursprüngliche Vierer-Tupel eines TP_i wird entsprechend erweitert.

$$TP_i = (x_i,y_i,z_i,t_i,\ objId,\ objTyp,\ modus_i,\ldots)_i \tag{2.4}$$

Die zeitliche Differenz Δt zwischen zwei Trajektoriepunkten wird mit Hilfe von

$$\Delta t_i = t_i - t_{i-1} \tag{2.5}$$

errechnet, die räumliche Distanz d_s durch die Abstandsfunktion d_R entsprechend dem Bewegungsraum R

$$d_{s,i} = d_R(TP_i,TP_{i-1}). \tag{2.6}$$

Die Länge einer Trajektorie, die der zurückgelegten Gesamtdistanz eines Objekts entspricht, ist demnach die Summe der Distanzen zwischen allen k Trajektoriepunkten.

$$l = \sum_{i=1}^{i=k} d_{s,i} \tag{2.7}$$

Ihre Kurvigkeit (*curviness*) cu beschreibt die Winkeländerungen γ_i bezogen auf die Länge und berechnet sich durch

$$cu = \frac{\sum_{i=1}^{i=k} |\gamma_i|}{l}. \tag{2.8}$$

Die Berechnung der Geschwindigkeit v bzw. der Beschleunigung a erfolgt im kontinuierlichen Fall mit Hilfe der ersten beiden Ableitungen der zurückgelegten Wegstrecke s. Die durchschnittliche Geschwindigkeit v_i sowie die Beschleunigung a_i eines Objekts an der i-ten Stelle der Trajektorie, werden im diskreten Fall durch

$$v_i = \frac{d_{s,i}}{\Delta t_i},\ \ a_i = \frac{d_{v,i}}{\Delta t_i} = \frac{v_i - v_{i-1}}{\Delta t_i + \Delta t_{i-1}} \tag{2.9}$$

bestimmt. Da die Objektbewegungen zwischen den einzelnen Trajektoriepunkten nicht erfasst werden, gelten sie als unbekannt. Um die Lücken zu füllen, gibt es zwei Optionen, die sich allerdings beide gegenseitig beeinflussen. Die Erste ist die Möglichkeit, die Bewegung zwischen zwei Trajektoriepunkten zu modellieren. Hierzu gibt es unterschiedliche Methoden. Häufig wird eine lineare Interpolation benutzt, die annimmt, dass die Bewegung zwischen zwei aufeinanderfolgenden Trajektoriepunkten linear verläuft. Andere Interpolationsverfahren, wie z.B. Splines oder andere nichtlineare Interpolationsverfahren, können die realen Bewegungen unter Umständen besser approximieren (siehe Abbildung 2.1), indem sie einen glatteren Verlauf erzeugen. Jedoch benötigen sie einen deutlich höheren Berechnungsaufwand. Das geeignete Verfahren hängt allerdings von der Wahl der Abtastrate ab. Diese stellt die zweite Option dar und bestimmt Δt und somit die Dauer der nicht erfassten Bewegung. Ist diese im Vergleich zur Bewegungsgeschwindigkeit hoch, so genügt im Regelfall eine lineare Interpolation, da die zurückgelegte Distanz zwischen zwei fixes relativ klein ist und dadurch wenig Bewegungsfreiraum gegeben ist. Ist das Verhältnis umgekehrt, also die Abtastrate relativ gering im Vergleich zur Geschwindigkeit, so kann die tatsächlich Form der Bewegung verloren gehen. In diesem Fall muss weiteres Modellwissen über die reale Bewegung genutzt werden, um diese besser zu modellieren. Die Wahl der Abtastrate ist in vielen Fällen jedoch bereits durch das Tracking-Verfahren und die Dauer der Beobachtung gegeben (siehe Abschnitt 2.2) und somit nicht frei wählbar. In diesem Fall muss dann die Bewegungsmodellierung angepasst werden.

Abbildung 2.1: Schematische Darstellung unterschiedlicher Interpolationsmöglichkeiten

Für eine vollständige Modellierung von Objektbewegungen muss neben der Bewegung selbst auch der Raum, in dem sich das Objekt bewegt, modelliert werden. Im folgenden Szenario (Abbildung 2.2), ist sowohl links als auch rechts zweimal eine Trajektorie mit der gleichen Gestalt dargestellt. Im linken Beispiel liegt der Trajektorie ein Euklidischer Raum zu Grunde. Dies bedeutet die Objekte können sich mit Ausnahmen von physikalischen Hindernissen (z.B. Wände, Gewässer, ...) frei bewegen. Zudem ist die reale Bewegung des Objekts (in rot) dargestellt. In diesem Raum gelten demnach auch die üblichen Funktionen, wie bspw. die Euklidische Abstandsfunktion zwischen zwei Punkten. Im rechten Gegenbeispiel ist der Bewegungsraum ein Netzwerk, in dem die Objekte sich nur auf den Kanten (grau) zwischen den Knoten entlang bewegen können. Was hier deutlich wird, ist, dass beide Trajektorien nicht der gleichen realen Bewegungen (vgl. rot-gestrichelte Linien) entsprungen sein können, obwohl sie die gleiche Gestalt besitzen. Dieses muss bei der Trajektorieanalyse und der Berechnung der Trajektorieeigenschaften berücksichtigt werden. So hängt vom vorliegenden Bewegungsraum ab, ob die Euklidische Abstandsfunktion bei der Bestimmung der Länge der Trajektorie angewendet werden muss oder ob die Summe der Längen der genutzten Kanten im Netz der tatsächlichen Länge der Trajektorie entspricht.

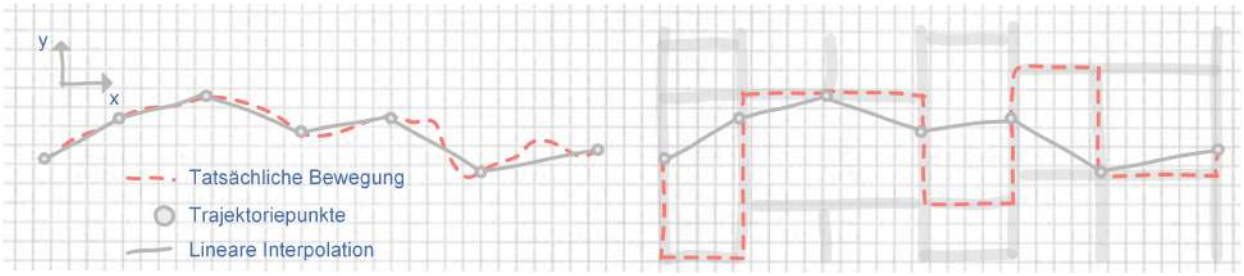


Abbildung 2.2: Zwei Beispielszenarien für die Modellierung einer Trajektorie im Bewegungsraum: Links eine Trajektorie und die reale Bewegung im Euklidischen Raum, rechts die gleiche Trajektorie im Netzwerkraum

Neben den Räumen aus dem obigen Beispiel ist noch die Space-Time-Cube-Repräsentation (Abbildung 2.3, links) (MacEachren, 1995) und der unregelmäßig tesselierte (gekachelte) Raum (rechts) zu erwähnen. Im Space-Time-Cube, der sich für 2D-Bewegungsdaten eignet, wird die 2D-Darstellung um die Zeit als dritte Dimension erweitert. Der durch die neue orthogonale z- bzw. t-Achse entstehende Würfel wird häufig zur Visualisierung des zeitlichen Verlaufs von Trajektorien genutzt. Beispiele für unregelmäßig tesselierte Räume sind die Ergebnisse aus bspw. Delauney-Triangulationen oder Voronoi-Gebietszerlegungen, wie sie unter anderem bei Mobilfunknetzwerken als Funkzellen um die verteilten Funkmasten herum in Erscheinung treten. In diesen Räumen erfolgt eine Lokalisierung von Objekten in der Art, dass angegeben wird, in welcher Zelle sich ein Objekt zu einem Zeitpunkt befinde. Die Position ist demnach nur bezogen auf die Zellgröße bekannt. Eine Trajektorie ist dementsprechend eine zeitlich annotierte Abfolge von Zellen, in denen das jeweilige Objekt verortet worden ist.

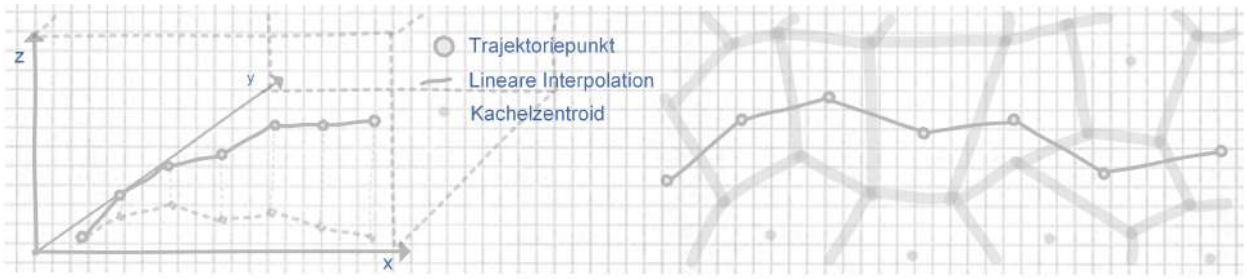


Abbildung 2.3: Space-Time-Cube und unregelmäßig tesselierter-Raum

Objektbewegungen können in jedem dieser vier Bewegungsräume modelliert werden. Jedoch besitzt jeder, wie oben beschrieben, seine eigenen speziellen Eigenschaften bei der Abbildung der tatsächlichen Trajektorien. Bei der Analyse der Trajektorien müssen diese Eigenheiten berücksichtigt werden. Tabelle 2.1 enthält für unterschiedliche Beispielszenarien die jeweiligen typischen Bewegungsräume.

Tabelle 2.1: Beispielszenarien für typische Bewegungsräume

Analyseszenario	**Typischer Bewegungsraum**
Fahrzeuge im Straßenverkehr	Netzwerkraum
Fußgänger auf Plätzen, in Gebäuden, usw.	Euklidischer Raum
Fußgänger im Straßenverkehr	Netzwerkraum oder Euklidischer Raum
Sportler während des Wettkampfs (Ballsportarten, wie z.B. Fußball oder Handball)	Euklidischer Raum
Objektbewegungen mit Hilfe von Mobiltelefonortung	Unregelmäßig tesselierter Raum
Tiere in freier Wildbahn	Euklidischer Raum

2.2 Erfassung von Trajektorien

Es existiert eine Vielzahl an unterschiedlichen Systemen bzw. Verfahren, mit denen das Objekt-Tracking (siehe Abschnitt 1.2.1) gelöst werden kann. Diese können nach Yilmaz u. a. (2006) anhand der genutzten Technologien zur Ortung unterschieden werden. In den folgenden Abschnitten werden die gängigsten Technologien vorgestellt. Dabei wird jeweils auf das Vorgehen, die Objektlokalisierung bzw. -detektion und die Verfolgung eingegangen. Die in der Arbeit verwendeten Video- und GNSS- (*Global Navigation Satellite Systems*) basierten Verfahren werden ausführlicher erläutert und auch hinsichtlich ihrer Genauigkeit betrachtet. Die übrigen Verfahren werden anschließend nur in aller Kürze vorgestellt.

2.2.1 GNSS-Tracking

Vorgehen

Beim Tracking von Objekten auf Basis einer GNSS-Lokalisierung werden die zu beobachtenden Objekte mit GNSS-Empfängern bzw. -Loggern ausgestattet. Auf die aufgezeichneten Trajektorien kann je nach Verfügbarkeit der entsprechenden Kommunikationsschnittstellen, z.B. WLAN, entweder direkt oder nach dem Auslesen des Gerätespeichers zugegriffen werden.

Lokalisierung

Eine verbreitete Variante der Objekt-Lokalisierung nutzt zur Ortung der Objekte mindestens eines der verfügbaren *Globalen Navigationssatellitensysteme* (*GNSS*). Bei diesen Systemen handelt es sich um das amerikanische *NAVSTAR-GPS*[1] (kurz: GPS), das russische *GLONASS*[2] sowie um die sich noch im Aufbau befindlichen GNSS *Galileo* (Europa) und *Beidou* (China). Jedes dieser Systeme besitzt eine gewisse Anzahl an Satelliten, die in Navigationsnachrichten ihre genaue Position und Uhrzeit senden. Ein entsprechender Empfänger empfängt die Signale und wertet diese für eine globale Positionsbestimmung (inkl. Höhe) aus. Das Grundprinzip bei der Positionsbestimmung ist trotz der unterschiedlichen Systeme immer gleich. Die Position wird mittels der Entfernung zu mindestens vier Satelliten bestimmt. Die Entfernung wiederum wird durch die Zeitverschiebung berechnet, die sich durch die Signallaufzeit zwischen Senden und Empfang ergibt. Da zudem die Positionen der Satelliten in den empfangenen Navigationsnachrichten enthalten sind, ist eine Bestimmung der Position des Empfängers in bspw. WGS84-Koordinaten[3] (bei Verwendung von GPS) möglich. Dieses wird schematisch in Abbildung 2.4a dargestellt. Die Zahl der benötigten Satelliten ergibt sich daraus, dass zur Positionsbestimmung drei Satelliten nötig sind, jedoch ein Vierter gebraucht wird, um den Empfängeruhrfehler zu bestimmen, da die Empfänger im Allgemeinen keine hoch-genauen Uhren besitzen. Die Details zu den unterschiedlichen Systemen und Messprinzipien sind in Bauer (2011) zu finden.

Für die Umsetzung eines GNSS-Trackings müssen einige Voraussetzungen erfüllt sein. Zum einen muss die Sicht zu den Satelliten frei sein, was ein Indoor-Tracking natürlich ausschließt. Zum anderen müssen die zu beobachtenden Objekte mit Empfängern ausgestattet werden. Probleme können hier dadurch entstehen, dass dies auf Grund der Beschaffenheit, Größe oder Form der Objekte technisch nicht machbar ist oder die Objekte dieses nicht wollen bzw. zulassen. Beispiele hierfür wären Menschen, die nicht mit einem Eingriff in ihre Privatsphäre einverstanden sind oder Tiere, die sich gegen einen befestigten Fremdkörper wehren. Bei der Verwendung von Loggern wird die Trajektorie des Objekts in den internen Speicher geschrieben. Dieser muss für eine Offline-Analyse später ausgelesen werden. Besitzt der Empfänger eine Kommunikationsschnittstelle (z.B. WLAN oder Bluetooth) so ist auch eine Online-Analyse möglich, da je nach Systemdesign die Daten direkt übertragen werden können.

Die Genauigkeit der GNSS-Positionsmessungen ist häufig mit ca. 5 bis 20 m angegeben. Dieser Wert hängt stark von den Messbedingungen ab. Abschattungen, die die Sichtbarkeit von Satelliten einschränken, sowie *Multipath*-Effekte, die durch Signal-Reflektionen an der Umgebung und damit verbundenen Empfangsverzögerungen entstehen, sorgen für Messfehler. Zusätzlich beeinflussen atmosphärische Effekte die Laufzeit der Signale, was ebenfalls zu Ungenauigkeiten führt. Auch der Empfänger selbst, bzw. die Qualität des darin verbauten Chips samt Firm-/Software, ist ein Einflussfaktor. Je qualitativ hochwertiger diese Kombination ist, desto bessere Ergebnisse sind zu erwarten. Deutlich wird dieses am Beispiel von Smartphones. Die dort verbauten GPS-Chips sind aus Kostengründen eher im unteren Genauigkeitsbereich angesiedelt und liefern dementsprechend schlechtere Ergebnisse als spezialisierte Markengeräte. Dieses ist für den Standardnutzer kein Problem, da die Genauigkeit für die typischen Anwendungen wie z.B. die Navigation ausreichend ist. Durch die beschriebenen Fehlerquellen entstehen systematische Fehler, die sich anhand systematischer Verschiebungen im Vergleich zu den realen Positionen zeigen. In der in Abbildung 2.4b dargestellten Trajektorie (grün) ist dies gut zu erkennen.

[1] **NAV**igation **S**atellite **T**ime **A**nd **R**anging - **G**lobal **P**ositioning **S**ystem
[2] **Glo**balnaja **na**wigazionnaja **s**putnikowaja **s**istema
[3] **W**orld **G**eodetic **S**ystem 19**84**: Geodätisches Referenzsystem zur Vereinheitlichung von Positionsangaben

Im Allgemeinen kann die Verwendung von *Differential Global Positioning System* (*DGPS*) die Positionsgenauigkeit mit Hilfe von stationären Referenzstationen auf deutlich unter einem Meter verbessern. Diese Verbesserung wird dadurch ermöglicht, dass die Lage dieser Stationen exakt bekannt ist und daher die Abweichung der Signallaufzeiten bestimmt werden kann. Diese Abweichungen können von DGPS-Empfängern genutzt werden, um die eigenen Messungen zu korrigieren. Jedoch wird dieses Verfahren nicht von allen GPS-Empfängern unterstützt. Zudem sind die Korrekturdaten nicht kostenfrei, sodass das Verfahren sich nicht für alle Anwendungsszenarien anbietet.

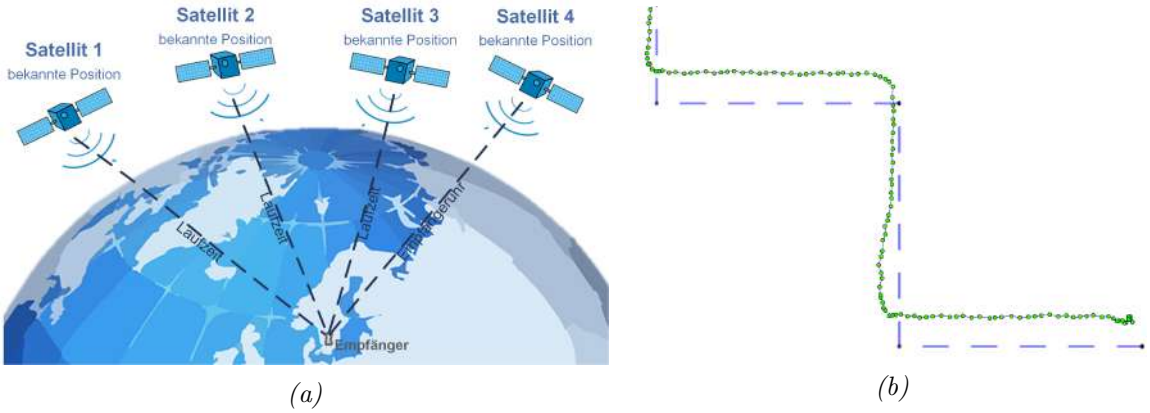


Abbildung 2.4: a) Eine schematische Darstellung der Funktionsweise von GPS. b) Eine Objekt-Trajektorie (grün) zeigt beispielhaft einen systematischen Fehler der GPS-Messungen (N-S-Verschiebung). Die Ground-Truth-Strecke ist in blau, gestrichelt dargestellt.

Wie gut die tatsächliche Bewegung des Objekts erfasst wird, hängt neben der Genauigkeit der Positionierung auch von der Abtastrate ab. Diese ist bei den handelsüblichen, tragbaren Geräten einstellbar, wobei das Loggen meistens auf maximal 10 Hz limitiert ist. Eine untere Schranke diesbezüglich gibt es nicht. Bei der Wahl der Abtastrate müssen vorliegender Bewegungsraum und die zu erwartende Dynamik des Objekts berücksichtigt werden (vgl. Abschnitt 2.1). Da bei zeitlich längeren Messungen zudem der verfügbare Speicher und die notwendige Energie beachtet werden müssen, ergibt sich häufig ein Kompromiss zwischen der Abtastrate und der Beobachtungszeit.

Im Kontext der Abtastrate und Genauigkeit wurden verschiedene Untersuchungen bzgl. der Validität und der Zuverlässigkeit (u.a. Coutts und Duffield, 2010; Gray u. a., 2010; Varley u. a., 2012) durchgeführt. In denen wird die Eignung des GPS-Trackings abhängig von der Abtastrate und der Positionierungsgenauigkeit überprüft. Die Ergebnisse zeigen, dass die untersuchten Geräte akzeptable relative Genauigkeiten, bspw. in Bezug auf die zurückgelegte Distanz, liefern. Die absolute Genauigkeit liegt dort eher im Bereich von wenigen Metern. Wird eine höhere absolute Genauigkeit benötigt, muss auf entweder DGPS oder andere Verfahren zurückgegriffen werden.

Verfolgung der Objekte

Die Lösung des Zuordnungsproblems ist hier trivial, da eine Zuordnung auf Grund der Tatsache, dass jedes Gerät nur seine eigene Position ermittelt, entfällt.

2.2.2 Videobasiertes Tracking

Vorgehen

Eine weitere Möglichkeit, die Position von Objekten zu erfassen, bietet die Erfassung auf Bild- bzw. Videosequenzen. Soll diese Technologie zum Objekt-Tracking verwendet werden, so ist der

allgemeine Ablauf wie folgt: die zu beobachtenden Objekte werden in der Sequenz aus Bildern einer zuvor kalibrierten Kamera detektiert. Die Detektionen werden darüber hinaus den Objekten zugeordnet, woraus sich nach Umrechnung der Bild- in Weltkoordinaten die Trajektorien ergeben. Der Ablauf lässt sich demnach wie folgt darstellen:

1. Kalibrierung der Kamera(s)
2. Detektion der Objekte im neuen, entzerrten Kamerabild
3. Verfolgung der Objekte und Zuordnung der Detektionen zu den Objekt-Trajektorien
4. Wenn Erfassung nicht beendet, weiter bei Schritt 2

Kalibrierung

Die Bilder von Objektiv-Kameras besitzen Verzeichnungen, die dafür sorgen, dass die Objektposition nach der Berechnung der Weltkoordinaten von der tatsächlichen Position abweicht. Zur Korrektur der Verzeichnungen und zur Bestimmung der Transformation zwischen Bild- und Weltkoordinaten müssen vor der Erfassung die innere und äußere Orientierung der Kamera ermittelt werden. Mit Hilfe einer Kalibrierung werden die inneren (*intrinsischen*) Kameraparameter ermittelt. Diese beschreiben das geometrische Kameramodell und dienen zur Korrektur der Kamerabilder (siehe Abbildung 2.5a). Hierbei handelt es sich um die *Kamerakonstante* c, die *Lage des Bildhauptpunktes* H', die *radial-symmetrische* $\Delta r'$, *tangentiale* und *asymmetrische Verzeichnung* sowie die *Affinität* und *Scherung* des Bildkoordinatensystems.

Die äußeren (*extrinsischen*) Parameter beschreiben die äußere Orientierung und damit die Lage und Ausrichtung der Kamera in einem übergeordneten Koordinatensystem (Abbildung 2.5b). Sie wird zur Umrechnung zwischen Bild- in Weltkoordinaten benutzt und durch die Translation X_0 zwischen dem Ursprung des Bildkoordinatensystems und dem Projektionszentrum O' sowie den drei Rotationen $R_{\omega,\phi,\kappa}$ definiert.

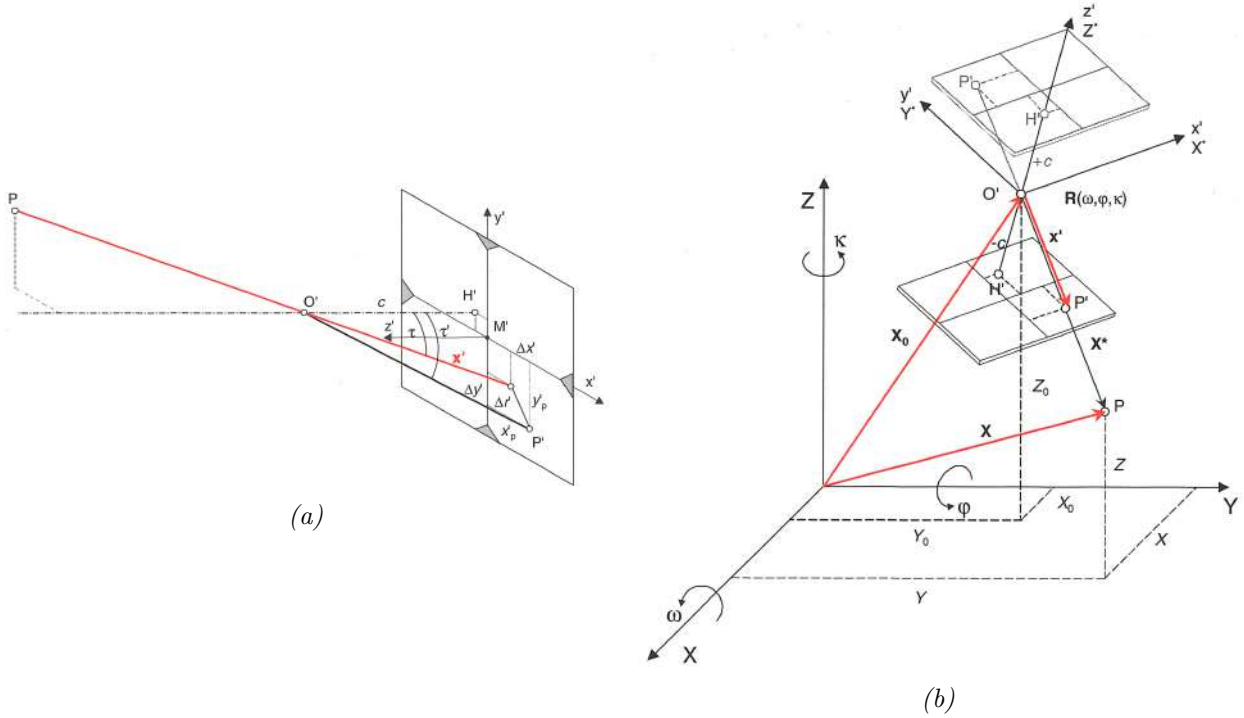


Abbildung 2.5: Die innere (a) und äußere Orientierung (b). Bildquellen: Luhmann (2000), S. 119 und S. 235

Die Parameter der inneren und äußeren Orientierung können durch eine Labor-, Testfeld- oder Simultankalibrierung bestimmt werden (Luhmann, 2000). Die Testfeldkalibrierung erfolgt häufig mit Hilfe eines bekannten (Schachbrett-) Musters. Das entsprechende Vorgehen dazu wird u.a. in Zhang (2000) erläutert.

Objektdetektion

Bei der Objektdetektion hängt die Auswahl des Verfahrens von der Installation der Kameras und deren Einstellungen ab. Beides kann einen Einfluss auf die äußeren und inneren Kameraparameter besitzen. Je nach dem, ob sich die Parameter während der Beobachtung ändern, bieten sich unterschiedliche Methoden zur Detektion der Objekte an. Bleiben beide konstant, was bei einer stationären (starren) Kamera ohne Zoom (bzw. Fokusänderung) der Fall ist, können Verfahren zur Vordergrund-Hintergrund-Trennung angewendet werden. Diese Verfahren zielen darauf ab, dynamische Objekte von einem statischen Hintergrund zu unterscheiden. Hierzu wird im Allgemeinen die Differenz zwischen dem Hintergrund und dem aktuellen Bild bestimmt. In den Bildregionen, in denen Differenzen identifiziert werden, sind Änderungen der Szene enthalten, die von sich bewegenden Vordergrundobjekten hervorgerufen werden. Auf diese Weise können bewegte Objekte im Bild detektiert werden. Ändern sich die Kameraparameter jedoch, so ist ein Lernen des Hintergrunds nicht ohne Weiteres möglich. In diesem Fall werden Methoden wie bspw. der HOG-Detektor (*Histogram of Oriented Gradients*) (Dalal und Triggs, 2005), die verschiedene (Objekt-)Merkmale zur Detektion nutzen, relevanter. Eine übersichtliche Darstellung der unterschiedlichen Verfahren ist in Yilmaz u. a. (2006) gegeben. Bezogen auf die Erfassung von Bewegungen im Sport geben Barris und Button (2008) eine Übersicht über anwendbare Methoden, deren Limitierungen und Zuverlässigkeiten.

Die Berechnung der Objektposition in Weltkoordinaten (X,Y,Z) erfolgt auf Basis der Bildkoordinaten (x',y'), der Parameter der inneren und äußeren Orientierung und den Kollinearitätsgleichungen (Gleichungen 2.10 und 2.11).

$$x' = x'_0 + z' \times \frac{r_{11} \cdot (X - X_0) + r_{21} \cdot (Y - Y_0) + r_{31} \cdot (Z - Z_0)}{r_{13} \cdot (X - X_0) + r_{23} \cdot (Y - Y_0) + r_{33} \cdot (Z - Z_0)} + \Delta x' \tag{2.10}$$

$$y' = y'_0 + z' \times \frac{r_{12} \cdot (X - X_0) + r_{22} \cdot (Y - Y_0) + r_{32} \cdot (Z - Z_0)}{r_{13} \cdot (X - X_0) + r_{23} \cdot (Y - Y_0) + r_{33} \cdot (Z - Z_0)} + \Delta y' \tag{2.11}$$

Die Genauigkeit der Positionen hängt demnach von der Kalibrierung der Kameras ab (Luhmann, 2000). Weitere Einflussfaktoren sind der Abbildungsmaßstab, die Bildauflösung und der Aufnahmewinkel.

Neben einer suboptimalen Kalibrierung können verschiedene weitere Probleme zu Ungenauigkeiten bei der Positionsbestimmung und den Trajektorien führen. Zum einen werden Objekte in vielen Fällen als Punkte repräsentiert. Dies führt zu der Herausforderung, dass eine repräsentative Position eines mehr oder weniger komplexen Objekts, z.B. eines Menschen, festgelegt werden muss. Zur Bestimmung einer 3D-Position wird häufig der Objektschwerpunkt verwendet. Um eine 2D-Objektposition in der Ebene (z.B. den Boden) zu erhalten, wird dieser auf den Boden projiziert. Hierbei stellt sich allerdings die Frage, ob der Schwerpunkt stets eine vernünftige Wahl ist. Beim Tracking von Personen kann dabei nämlich ein Problem entstehen, falls die Extremitäten nicht vollständig erfasst oder die Person sich in keiner symmetrischen Pose befindet. Dieses führt zu einer Verschiebung des Schwerpunkts und damit zu einer gleichermaßen verschobenen Objektposition. Zum anderen können auch bei der Zuordnung der Detektionen zu den Objekten Fehler entstehen. So können z.B. Verdeckungen von Objekten dazu führen, dass Objekte entweder fehlerhaft oder gar nicht detektiert werden. Dieses erschwert das Tracking der Objekte, da dort daraufhin

mit fehlenden Detektion bzw. Daten umgegangen werden muss, was wiederum zu Problemen wie Falschzuordnungen zwischen Detektionen und Objekten als weitere Fehlerquelle führen kann.

Eine Möglichkeit zur Reduzierung der Objektverdeckungen und den damit verbundenen Problemen ist die Verwendung von mehr als nur einer Kamera. So können bspw. durch Stereo-Kameras Tiefen- bzw. 3D-Informationen generiert werden, die bei der Unterscheidung der Objekte helfen können. Die Verteilung mehrerer Kameras in Multi-Kamera-Systemen erlaubt zudem die Betrachtung einer Szene aus unterschiedlichen Perspektiven.

Verfolgung der Objekte

Werden die Bewegungen der Objekte durch Kameras verfolgt, ist das Zuordnungsproblem deutlich komplexer im Vergleich zum GNSS-Tracking. Die bei der Lokalisierung ermittelten Objektdetektionen bzw. -positionen besitzen keine eindeutigen Objektkennungen, über die eine Zuordnung erfolgen könnte. Diese gilt es schließlich zu ermitteln. Zudem werden hier in der Regel mit einem Sensor mehrere Objekte gleichzeitig verfolgt. Das Zuordnungsproblem besteht daher aus einer Zuordnung von m Detektionen zu n Objekten. Im Idealfall gilt $m = n$, d.h. es existiert genau eine Detektion pro Objekt. Durch Fehldetektionen (z.B. Rauschen) und fehlende Detektionen, d.h. Objekte wurden nicht detektiert, ist dieser Idealfall jedoch nicht garantiert. Die Zuordnung gestaltet sich somit schwieriger. Abbildung 2.6 illustriert das Zuordnungsproblem. Dort sind die Kameradetektionen (blaue Kästchen, A - H) für vier aufeinanderfolgende Zeitschritte (t_0 bis t_3) dargestellt. Offensichtlich wurden in diesem Beispiel in Zeitschritt t_0 zwei, in t_1 drei, in t_2 ein und in t_3 zwei Objekte detektiert. Unter der Annahme, dass die zwei Objekte aus Zeitschritt t_0 verfolgt werden sollen und dass keine Mehrfachzuordnungen (eine Detektion wird mehreren Objekten zugeordnet) erlaubt sind, ergeben sich für den Übergang zum nächsten Zeitschritt (t_1), in dem durch eine Fehldetektion (C) $m > n$ gilt, $k = \frac{m!}{(m-n)!} = 6$ Zuordnungsmöglichkeiten. Im dritten Zeitschritt gilt wiederum $m < n$. Es fehlt mindestens eine Detektion. Im vierten Schritt gilt erneut $m = n$. Allerdings muss jeder Zeit die Tatsache berücksichtigt werden, dass es sich bei den Detektionen um Fehldetektionen handeln könnte. Dies bedeutet, dass auch bei ausreichend vielen Detektionen, nicht zwingend eine Zuordnung zu einem Objekt erfolgen muss.

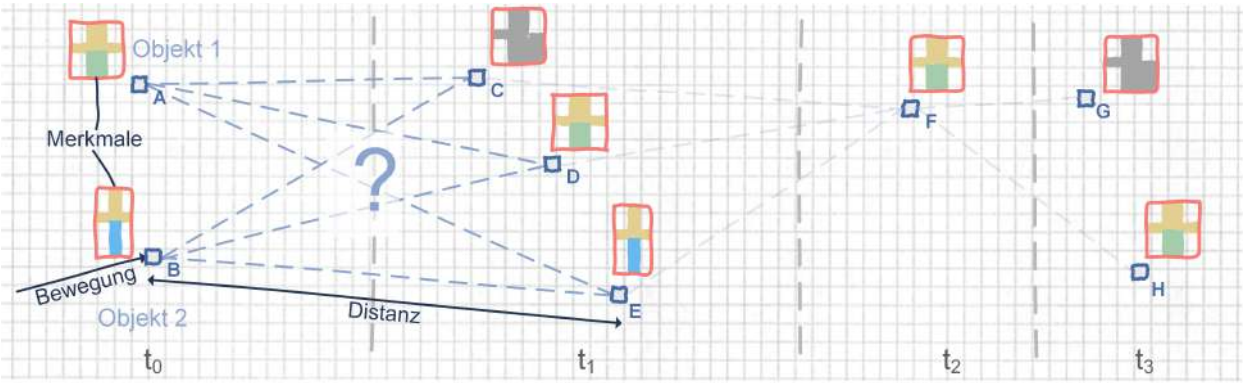


Abbildung 2.6: Das Zuordnungsproblem der Detektionen zu den Objekten

Die bisherige Darstellung des Zuordnungsproblems geht von einer konstanten Objektanzahl aus. Dies bedeutet, es kommen weder Objekte während der Beobachtungszeit hinzu, noch verschwinden welche. Handelt es sich nicht um eine konstante Gruppe, so ergibt sich die zusätzliche Fragestellung, ab wann ein neues Objekt nicht mehr als Fehldetektion verworfen wird, sondern eine entsprechende Trajektorie generiert wird.

Zur Lösung dieser Probleme existieren verschiedene Ansätze. Grundsätzlich lassen sie sich in globale Optimierungs- und Greedy-Verfahren unterscheiden. Während Optimierungsverfahren sämtliche

Daten zur Ermittlung einer global optimalen Lösung verwenden, nutzen Greedy-Verfahren für die Zuordnung lediglich die Daten aus dem aktuellen und letzten Zeitschritt. Die Bestimmung des globalen Optimums ist hier nicht garantiert. Ein Beispiel ist die Zuordnung mit Hilfe der *Ungarischen Methode*, bei der die Kosten der Zuordnung zwischen zwei Zeitschritten hinsichtlich einer Kostenfunktion minimiert werden. Yilmaz u. a. (2006) geben eine umfassende Übersicht (siehe Abbildung 2.7) über die unterschiedlichen Tracking-Methoden.

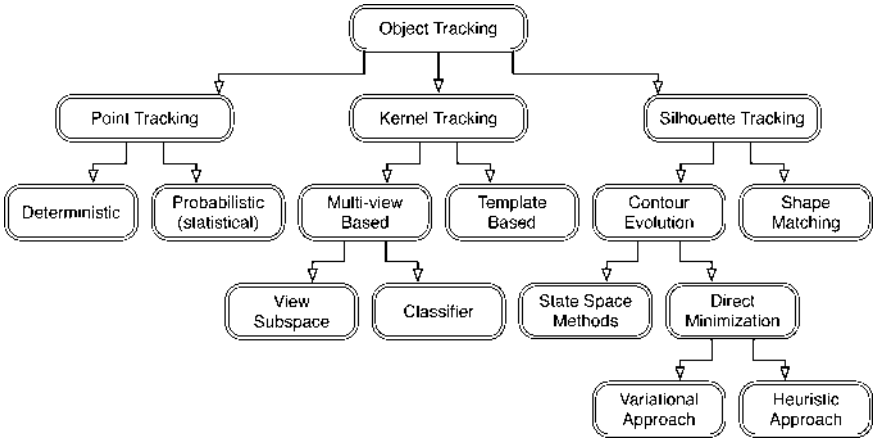


Abbildung 2.7: Taxonomie unterschiedlicher Tracking-Methoden. Bildquelle: Yilmaz u. a. (2006), S. 16.

In dem obigen Beispiel wird das Objekt-Tracking gemäß der gegebenen Taxonomie prinzipiell als *Punkt-Tracking* (*Point Tracking*) modelliert. Die Objekte werden dabei als Punkte repräsentiert. Eine Zuordnung kann daher die Positionen der Detektionen aus den jeweiligen Zeitschritten und daraus abgeleitete Bewegungsmodelle (hauptsächlich Bewegungsrichtung und -geschwindigkeit) nutzen. Ein *Kernel-Tracking* berücksichtigt die Gestalt bzw. Form und das Aussehen der Objekte. Hier werden die Objekte durch Formen wie Rechtecke oder Ellipsen repräsentiert, auf deren Basis Merkmale wie Histogramme berechnet werden. Beim *Silhouette Tracking* werden die Bildregionen bestimmt, in der sich die Objekte in den Kamerabildern befinden. Zur Extraktion der Merkmale werden entsprechend die Informationen verwendet, die sich auf Grundlage der Pixel innerhalb der Silhouetten ermitteln lassen. Merkmale, die in diesem Zusammenhang häufig Verwendung finden, sind ebenfalls die Gestalt und das Aussehen. Die Punkt-Tracking-Methoden lassen sich wiederum in deterministische und probabilistische Ansätze unterteilen. Der später in dieser Arbeit vorgestellte Tracking-Ansatz lässt sich dem probabilistischen Punkt-Tracking zuordnen, nutzt jedoch auch Teile der beiden anderen Varianten, um die Objekte zu identifizieren. Die Zuordnung wird dabei als probabilistisches Modell modelliert. Im nachfolgenden Abschnitt werden daher die Grundlagen dieser Modellierung erläutert.

2.2.3 Vergleich des GNSS- und videobasierten Trackings

Die beiden vorgestellten Verfahren werden anhand der folgenden Eigenschaften verglichen.
Abtastrate: Die Abtastrate (*sampling rate*) im Zusammenhang mit dem Objekt-Tracking ist die Frequenz, mit der die Objekte detektiert/lokalisiert werden. Es gilt: Je höher die Abtastung, desto detaillierter wird die Bewegung durch die Trajektorie abgebildet (2.1).
Genauigkeit: Die Genauigkeit ist durch die räumliche Abweichung bei der Lokalisierung der Objekte gegeben.
Verlässlichkeit: Die Verlässlichkeit beschreibt die Korrektheit der Objektverfolgung über die Zeit. Die zuvor beschriebenen Probleme können zu Fehlern bei der Identifizierung der Objekte und damit zu fehlerhaften Zuordnungen führen.

Technische Rahmenbedingungen: Neben den voranstehenden Eigenschaften gibt es zusätzliche Rahmenbedingungen, die bei der Verwendung der Systeme berücksichtigt werden müssen.

Tabelle 2.2: Vergleich der Tracking-Verfahren anhand ausgewählter Eigenschaften

Eigenschaften	GPS-Tracking	Videobasiertes Tracking
Messfrequenz	bis zu 10 Hz	Abhängig von Bildfrequenz: häufig 25 / 30 Hz, bis mehr als 1000 Hz möglich
Genauigkeit	gewöhnlich 5-20 m, DGPS: wenige Zentimeter möglich	unter 1 bis wenige Zentimeter
Verlässlichkeit	maximal, da Identifizierung über getragenes Gerät	nicht maximal durch Verdeckungen usw.
Technische Rahmenbedingungen	Objekte müssen Gerät tragen, Sicht zu Satelliten nötig	beschränkte Sichtfelder, Objekte müssen im Sichtfeld sein, Spezielle Erkennungsverfahren für die jeweiligen Objekte erforderlich

2.2.4 Weitere Tracking-Verfahren

Neben den zuvor beschriebenen Verfahren existieren auch weitere Methoden, die auf anderen Technologien zur Objekt-Ortung basieren. Darunter sind die **funkbasierten Verfahren** eine weitere prominente Kategorie. Das Ortungskonzept baut dort darauf auf, dass Referenzpunkte (häufig in Form von Antennen), die selbst entweder Sender oder Empfänger sein können im Beobachtungsraum verteilt sind und die zu beobachtenden Objekte mit einem entsprechenden Gegenstück, also Empfänger oder Sender, ausgestattet sind. Die Objektpositionen werden daraufhin mit Hilfe von *Tri-* bzw. *Multilaterations-* oder *Triangulationsalgorithmen* in Bezug auf die empfangenen Signale berechnet. Viele funkbasierte Tracking-Verfahren sind so in der Lage, eine sehr hohe Abtastrate und Genauigkeit anzubieten. Beispiele für existierende Systeme sind u.a. ein auf Ultra-Breitband[1] basierendes Ortungssystem der Firma Ubisens[2] oder das vom FraunhoferIIS[3] entwickelte Echtzeit-Lokalisierungssystems (*Real-Time Locating System*) RedFir ®. Des Weiteren besitzen sie ähnliche Vorteile wie das GPS-Tracking, sind aber zudem Indoor-tauglich, was wiederum eine Vielzahl an weiteren Anwendungsmöglichkeiten bietet. Ein bekanntes Anwendungsbeispiel ist der sogenannte „Chip im Ball“, bei dem der Ball mit einem Chip ausgestattet ist, was wiederum die oben beschriebene Ortung ermöglicht.

Eine ebenfalls auf Funk basierende Möglichkeit zur Ortung ist die Lokalisierung von Objekten mittels **Wireless Local Area Network** (**WLAN**). Hier werden unterschiedliche Techniken genutzt, um die Position eines Objekts zu bestimmen. So kann die Berechnung einer Position mit Hilfe der *Trilateration* und den Signalstärken zu bekannten Zugangspunkten (*access points*), über ein *Fingerprinting*, indem die aktuellen Signalstärken mit zuvor eingemessenen und verorteten Signalstärken verglichen werden, oder über die Auswertung der Winkel (*angle of arrival*), mit denen die Signale eintreffen, erfolgen.

Bluetooth und **GSM** (**G**lobal **S**ystem for **M**obile Communications) können ebenso zur Ortung verwendet werden. Der Unterschied ist jedoch, dass die Objektbewegungen hier in einem unregelmäßig tesselierten Raum (vgl. Abschnitt 2.1) stattfinden. Dieses hat einen Einfluss auf die

[1] engl. Ultra-wideband (UWB): Nutzung von großen Frequenzbereichen (ab 500 MHz) für u.a. Kommunikation
[2] http://indoor-ortung.de/systeme/ubisense/
[3] Fraunhofer-Institut für Integrierte Schaltungen IIS (https://www.iis.fraunhofer.de/de/ff/lv/lok/proj/redfir.html)

generierten Trajektorien, da diese eher als Sequenz aus besuchten Zellen (Checkpoints) zu verstehen sind. Dieses ist dadurch bedingt, dass für die Position des Objekts nicht dessen tatsächliche Position, sondern diejenige des besuchten Zugangspunkts bzw. des *Beacons* (bei Bluetooth) verwendet wird. Sowohl Bluetooth als auch WLAN bieten sich zum Tracken von Personen an, weil diese bereits durch ihre Integration in Smartphones, Smart-Watches oder Ähnlichem sehr verbreitet sind.

2.2.5 Probabilistische Modellierung

Unterschiedliche Fehlerquellen sorgen dafür, dass die für das Objekt-Tracking relevanten Information mit Unsicherheiten versehen sind. Ein Beispiel hierfür ist die zuvor beschriebene Lokalisierung, die abhängig von der Technologie und den Gegebenheiten, eine gewisse Unsicherheit aufweist. In diesem Zusammenhang kann die Wahrscheinlichkeitstheorie zur Modellierung dieser Unsicherheiten herangezogen werden. Sie stellt somit die theoretische Basis für die im Rahmen dieser Arbeit entwickelte Methode zum Tracken der Objekte dar.

Die Wahrscheinlichkeitstheorie ermöglicht die Modellierung von bekannten und unbekannten Zufallsvariablen, also Größen deren Werte bzw. Zustände ausschließlich vom Zufall abhängen und nicht kausal erklärt werden können. Bspw. kann die Wahrscheinlichkeit, dass die Zufallsvariable X den Wert x annimmt, durch $P(X = x) = p$ ausgedrückt werden, wobei $0 \leq p \leq 1$ gilt. Mit $P(X = x)$ wird hier die Wahrscheinlichkeit eines Ereignisses bezeichnet. Durch $P(X)$ hingegen wird die Verteilung über die Zufallsvariable X angegeben. Zwei elementare Regeln für die Rechnung mit Wahrscheinlichkeiten sind die *Summenregel* und die *Produktregel* (Bishop, 2006). Durch die Summenregel

$$P(X) = \sum_{Y} P(X,Y) \tag{2.12}$$

ergibt sich die *Randverteilung* bzw. *Marginalverteilung* $P(X)$ als Summe über alle *gemeinsamen Verteilungen* $P(X,Y)$ für alle möglichen Werte einer anderen Zufallsvariablen Y. Die Produktregel

$$P(X,Y) = P(Y|X)\ P(X), \tag{2.13}$$

besagt, dass die gemeinsame Verteilung $P(X,Y)$ durch das Produkt aus bedingter Wahrscheinlichkeit $P(Y|X)$ und der Randverteilung $P(X)$ gegeben ist.

Aus der Produktregel und der Symmetrieeigenschaft, durch die $P(X,Y) = P(Y,X)$ gilt, leitet sich der *Satz von Bayes*[1], auch *Bayessches Theorem* genannt, ab (Bishop, 2006).

$$P(X|Y) = \frac{P(Y|X)\ P(X)}{P(Y)}. \tag{2.14}$$

Bayessche Wahrscheinlichkeiten

Zur Interpretation von Wahrscheinlichkeiten existieren unterschiedliche Konzepte. Der *objektive* bzw. *frequentistischen* Wahrscheinlichkeitsbegriff interpretiert die Wahrscheinlichkeit als relative Häufigkeit, mit der ein Ereignis in einem Zufallsexperiment mit großer bzw. unendlicher Anzahl von Wiederholungen auftritt. Das *Bayessche* Konzept zur Interpretation von Wahrscheinlichkeiten hingegen nutzt Wahrscheinlichkeiten zur Darstellung von Unsicherheiten bzw. dem Grad der Überzeugung (Bishop, 2006). Durch den Satz von Bayes wird Vorwissen (*A-priori-Verteilung*, kurz:

[1] benannt nach Thomas Bayes (engl. Mathematiker)

Prior) über ein Ereignis w, das mit einer gewissen Unsicherheit behaftet ist, mit neuen Erkenntnissen aus Daten bzw. Beobachtungen $\mathcal{D}$ (*Likelihood*) kombiniert, sodass eine verbesserte Version des Vorwissens (A-posteriori-Verteilung, kurz: Posterior) generiert wird. Im Rahmen dieser Betrachtungsweise von Wahrscheinlichkeiten können die Terme des Bayesschen Theorems wie folgt interpretiert werden:

$$P(w|\mathcal{D}) = \frac{P(\mathcal{D}|w)\ P(w)}{P(\mathcal{D})}. \tag{2.15}$$

$P(\mathcal{D}\|w)$	*Likelihood* (*likelihood*)	entspricht der Wahrscheinlichkeit von $\mathcal{D}$ bei gegebenen w
$P(w\|\mathcal{D})$	*A-posteriori-Wahrscheinlichkeit* (*posterior*)	entspricht der Wahrscheinlichkeit von w bei gegebenen $\mathcal{D}$
$P(w)$	*A-priori-Wahrscheinlichkeit* (*prior*)	für w
$P(\mathcal{D})$	*Evidenz* (*evidence*)	entspricht der A-priori-Wahrscheinlichkeit für $\mathcal{D}$

Bayessche Inferenz

Mit Hilfe der Bayesschen Inferenz, oder auch Bayessches Update genannt, ist es bei gegebenen Daten, gegebener A-priori-Wahrscheinlichkeit über den gesuchten Parameter und einer Likelihood-Funktion, die das Verhältnis der Daten und des Parameters beschreibt, möglich, die A-posteriori-Wahrscheinlichkeit zu bestimmen. Liegt sowohl bei den Daten, der A-priori-Wahrscheinlichkeit als auch bei der Likelihood-Funktion eine Normalverteilung vor, so ist auch die A-posteriori-Wahrscheinlichkeit als Ergebnis wiederum normalverteilt. Bei der Berechnung werden die Bayesschen Update-Gleichungen verwendet (Murphy, 2002). Im Fall von univeraten Normalverteilungen ergibt sich die Varianz der A-posteriori-Verteilung ${\sigma'_0}^2$ durch

$${\sigma'_0}^2 = \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}}, \tag{2.16}$$

der A-posteriori Erwartungswert μ'_0 durch

$$\mu'_0 = \frac{\sigma^2\mu_0 + \sigma_0^2\mu}{\sigma^2 + \sigma_0^2}. \tag{2.17}$$

Hierbei sind μ_0 und σ_0 die Parameter der Prior-Verteilung, μ und σ die der Likelihood-Funktion. Dieses ist auch in Abbildung 2.8a dargestellt. Handelt es sich um multivariate Normalverteilungen (Abbildung 2.8b) berechnet sich die A-posteriori Kovarianzmatrix ${\Sigma'_0}^2$ entsprechend durch

$${\Sigma'_0}^2 = (\Sigma^{-1} + \Sigma_0^{-1})^{-1} \tag{2.18}$$

und der Vektor der Erwartungswerte $\hat{\mu}'_0$ durch

$$\hat{\mu}'_0 = (\Sigma^{-1} + \Sigma_0^{-1})^{-1}\ (\Sigma^{-1}\ \hat{\mu} + \Sigma_0^{-1}\ \hat{\mu}_0). \tag{2.19}$$

Als Beispiel kann dazu die Positionsbestimmung eines Objekts mit mehreren Sensoren bzw. Messungen angeführt werden. Die erste Messung liefert dabei das Vorwissen über die Position. Sie entspricht der Priori-Verteilung. Durch weitere Messungen, mit einem anderen (möglicherweise genauerem) Sensor, wird dieses Vorwissen aktualisiert (*Update*), sodass sich die Unsicherheit der ersten Positionsschätzung verringert und genauere Angaben über die tatsächliche Position gemacht werden können.

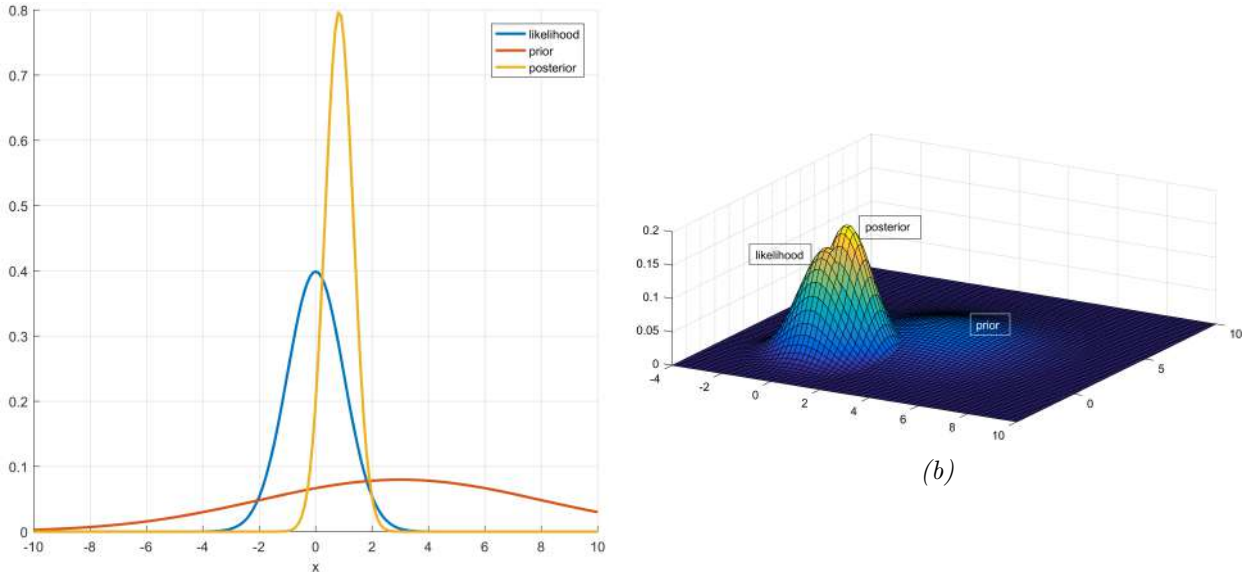


Abbildung 2.8: Das Bayessche Update zur Bestimmung der Posterior-Wahrscheinlichkeitsverteilung (gelb) auf Basis einer normalverteilten Prior-Wahrscheinlichkeitsverteilung (rot) und einer ebenfalls normalverteilten Likelihood-Funktion (blau) als 1D- (a) und 2D-Variante (b) .

Bayessches Netz

Mit steigender Anzahl an Zufallsvariablen und bedingten Abhängigkeiten zwischen diesen Variablen wird die Modellierung eines derartigen Systems deutlich komplexer. Zu diesem Zweck bietet sich ein *Bayessches Netz* an, welches ein probabilistisches graphisches Modell zur Modellierung einer Menge an Zufallsvariablen und deren bedingter Wahrscheinlichkeiten ist. Es ist ein gerichteter Graph ohne Zyklen, in dem die Knoten *unbekannte* oder *beobachtete* Zufallsvariablen und die Kanten deren Abhängigkeiten repräsentieren. Folglich gelten die Variablen, deren Knoten nicht miteinander verbunden sind (es existiert keine Kante zwischen diesen Knoten), als *unabhängig* voneinander. Unter Berücksichtigung der Produktregel (Gleichung 2.13) kann die gemeinsame Wahrscheinlichkeitsverteilung über die im Bayesschen Netz enthaltenen Zufallsvariablen $X_i,\ldots,X_n$ als Produkt von bedingten Wahrscheinlichkeiten $P(X_i|X_{pa(i)})$ mit $X_{pa(i)}$ als Menge der Elternknoten von X_i berechnet werden.

$$P(X_1,\ldots,X_n) = \prod_{i=1}^{n} P(X_i|X_{pa(i)}) \tag{2.20}$$

Für das in Abbildung 2.9 gegebene Beispiel kann die gemeinsame Wahrscheinlichkeitsverteilung daher durch

$$P(X_1,X_2,X_3) = P(X_1)\ P(X_2)\ P(X_3|X_1,X_2) \tag{2.21}$$

ermittelt werden. Im Beispiel werden die Zufallsvariablen „Wetter“, „Essen“ und „Stimmung“ modelliert. Anhand der Kanten wird deutlich, dass die „Stimmung“ sowohl vom „Wetter“ als auch vom „Essen“ abhängt. „Wetter“ und „Essen“ hingegen sind unabhängig voneinander. Die Wahrscheinlichkeitsfunktionen der jeweiligen Knoten werden als Wahrscheinlichkeitstafeln (*Conditional Probability Tables*, kurz *CPT*) angegeben. Die Wahrscheinlichkeit, dass jemand trotz *regnerischen Wetters* auf Grund *leckeren Essens gute Laune* hat, ist somit

$$P(Wetter = Regen,Essen = lecker,Stimmung = gut) = 0{,}6 \cdot 0{,}9 \cdot 0{,}75 = 0{,}405 \tag{2.22}$$

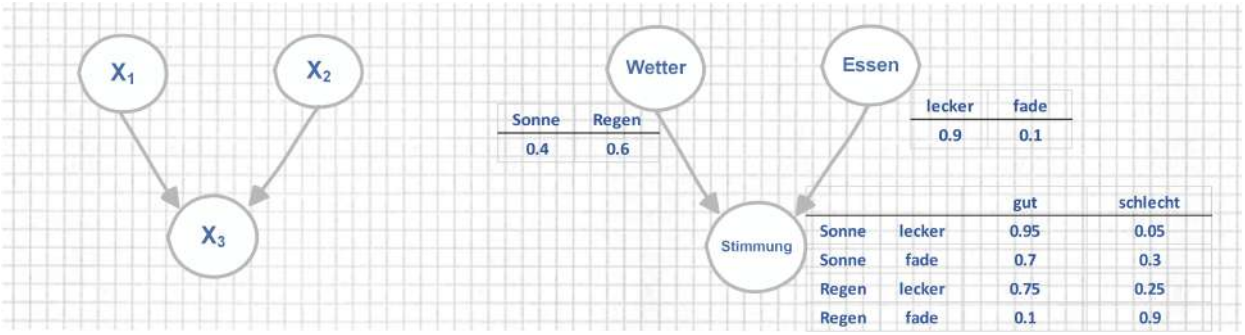


Abbildung 2.9: Links: Ein Bayessches Netz bestehend aus den drei Zufallsvariablen X_1, X_2 und X_3. X_3 hängt dabei sowohl von X_1 als auch von X_2 ab. Rechts: ein Beispiel.

Zur Modellierung von stochastischen Prozessen und damit von Sequenzen von Zufallsvariablen, wie sie in Zeitreihen, u.a. auch in Trajektorien, auftreten, können **Dynamische Bayessche Netze (DBN)** (Dean und Kanazawa, 1989; Murphy, 2002) genutzt werden. Es wird der zeitliche Verlauf der Zufallsvariablen in diskreten Zeitschritten modelliert. Kanten über die Zeitschritte hinweg repräsentieren hierbei temporale Abhängigkeiten. Das einfachste Modell in diesem Zusammenhang ist ein *Markov-Prozess erster Ordnung* (siehe Abbildung 2.10). Dabei handelt es sich um einen stochastischen Prozess, in dem der Zustand einer Zufallsvariablen ausschließlich von ihrem vorherigen Zustand abhängt. Er besitzt die **Markov-Eigenschaft**, gegeben durch

$$P(X_{t+1}|X_t,X_{t-1},\ldots,X_0) = P(X_{t+1}|X_t). \tag{2.23}$$

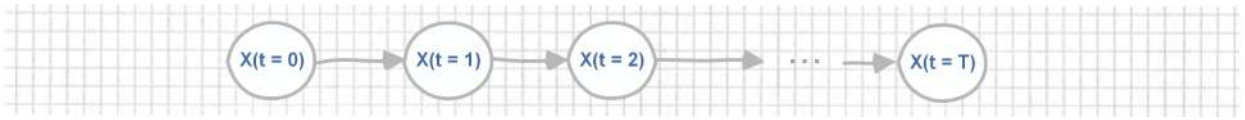


Abbildung 2.10: Markov-Prozess erster Ordnung

Im folgenden Schema wird das Beispiel aus Abbildung 2.9 als DBN dargestellt. In diesem Beispiel hängt der neue Zustand von X_3 demnach vom eigenen Zustand aus dem vorherigen Zeitschritt ab und berechnet sich ebenfalls durch die bedingte Wahrscheinlichkeit $P(X_3(t+1)|X_1(t+1),X_2(t+1), X_3(t))$.

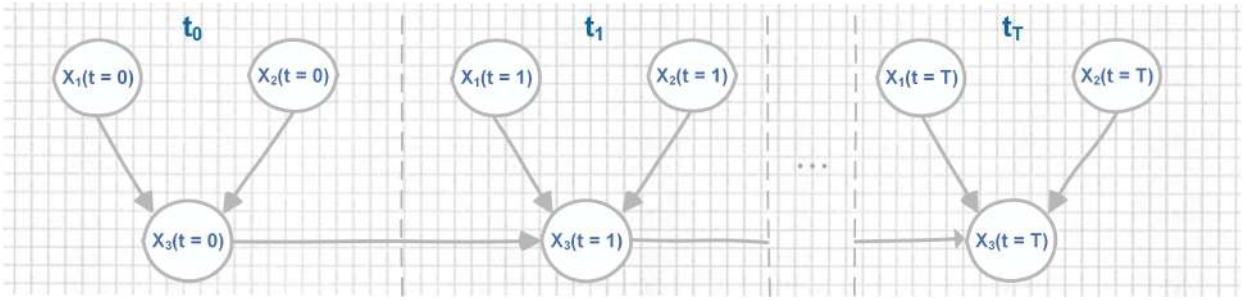


Abbildung 2.11: Dynamisches Bayessches Netz

Hidden Markov Model (HMM)

Sind die Zustände nicht direkt beobachtbar, sondern indirekt über andere Variablen, spricht man von verborgenen (*hidden*) Zuständen. Liegt zudem ein Markov-Prozess vor, handelt es sich um

ein Hidden Markov Model als spezielle Form eines Dynamischen Bayesschen Netzes. Dort werden die Zustände mittels einer *rekursiven Bayesschen Schätzung* ermittelt. Diese schätzt eine unbekannte Wahrscheinlichkeitsfunktion anhand von sequenziellen Messungen und eines mathematischen Modells. Ein solches HMM, wie in Abbildung 2.12 dargestellt, kann als Quintupel aus $\lambda = (X; Y; A; B; \pi)$ definiert werden (Rabiner und Juang, 1986; Dugad und Desai, 1996; Ghahramani, 2001). Dabei sind durch $X\{X_1,X_2,\ldots,X_M\}$ die möglichen verborgenen Zustände und durch $Y = \{Y_1,Y_2,\ldots,Y_K\}$ die Beobachtungen gegeben. Zudem sind die Zustandsübergangswahrscheinlichkeiten für den Zeitschritt t gegeben durch

$$A = \{a_{ij}\}, \quad a_{ij} = P(X_j(t+1)|X_i(t)), \quad i,j = 1 \ldots M. \tag{2.24}$$

Die Beobachtungswahrscheinlichkeiten ergeben sich unter der Bedingung des tatsächlichen Zustands X_j aus

$$B = \{b_{jk}\}, \quad b_{jk} = P(Y_k|X_j(t)), \quad j = 1 \ldots M,k = 1 \ldots K, \tag{2.25}$$

Die Wahrscheinlichkeiten der initialen Zustände für $t = 0$ sind festgelegt zu

$$\pi = \pi_i, \quad \pi_i = P(X_i(t=0)), \quad i = 1 \ldots M. \tag{2.26}$$

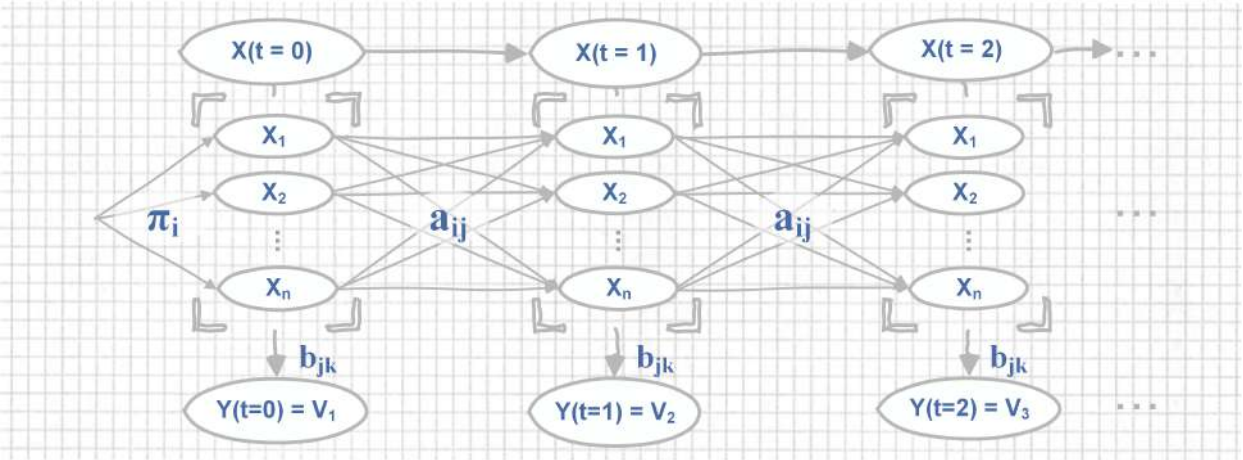


Abbildung 2.12: In diesem Beispiel wird mit Hilfe eines HMM der Verlauf der Zufallsvariablen X und Y über die Zeit t modelliert. Dabei sind durch X_i die nicht beobachtbaren (verborgenen) Zuständen und Y_i die Beobachtungen gegeben. Die Übergangswahrscheinlichkeiten sind durch a_{ij}, die Beobachtungswahrscheinlichkeiten durch b_{jk} und die initiale Wahrscheinlichkeitsverteilung ist durch π_i gegeben.

Typische Anwendungen von HMMs liegen im Bereich der Mustererkennung, z.B. Erkennung von typischen Bewegungen (Bashir u. a., 2005), und der Verarbeitung von sequentiellen Daten, z.B. bei der Spracherkennung (Rabiner und Juang, 1986), dem Objekt-Tracking (Martinerie, 1997) oder der Anomalie-Detektion (Huang u. a., 2014).

2.2.6 Viterbi-Algorithmus

In vielen Anwendungsfällen wird die wahrscheinlichste Sequenz aus Zuständen $X_V = (X_1, \ldots, X_T) \in X$ anhand der gegebenen Sequenz an Beobachtungen $O = (o_1, \ldots, o_T) \in Y$ gesucht. Zu diesem Zweck bietet sich der Viterbi-Algorithmus (Forney, 1973) an. Der Algorithmus besteht aus den drei aufeinanderfolgenden Phasen *Initialisierung*, *Rekursion* und *Backtracking*, die im Nachfolgenden erläutert werden. Dabei beinhaltet $X^*_{V,t}$ den wahrscheinlichsten Vorgänger des Zustands

zur Zeit t, während $P_t(X_i)$ die Wahrscheinlichkeit der wahrscheinlichsten Sequenz aus Zuständen bis zum Zustand X_i ist.

Die **Initialisierung** nutzt die Werte der definierten Anfangszustände sowie die Werte der ersten Beobachtungen und erfolgt durch

$$P_{t=0}(X_i) = \pi_i \cdot b_{0,i}, \quad 1 \leq i \leq m. \tag{2.27}$$

In der **Rekursionsphase** findet die Berechnung der Wahrscheinlichkeit P_t anhand der Beobachtungen und dem Maximum der Produkte aus Übergangswahrscheinlichkeit vom Vorgänger und dessen Wahrscheinlichkeit P_{t-1} statt. Zusätzlich wird für jeden Zustand der wahrscheinlichste Vorgängerzustand ermittelt.

$$P_t(X_i) = b_{t,i} \cdot \max_{1 \leq j \leq M} \left(a_{ji} \cdot P_{t-1}\left(X_j\right)\right) \quad 1 \leq i \leq M,\, 1 \leq t \leq T \tag{2.28}$$

$$X^*_{V,t}(X_i) = \operatorname*{argmax}_{1 \leq j \leq M} \left(a_{ji} \cdot P_{t-1}\left(X_j\right)\right) \quad 1 \leq i \leq M,\, 1 \leq t \leq T \tag{2.29}$$

In der abschließenden **Backtracking**-Phase wird mit Hilfe der in den einzelnen Zuständen gespeicherten wahrscheinlichsten Vorgängern die wahrscheinlichste Sequenz an Zuständen, der sogenannte *Viterbi-Pfad*, ermittelt.

$$X_{V,t} = X^*_{V,t+1}(X_{V,t+1}) \tag{2.30}$$

Der Viterbi-Algorithmus arbeitet rekursiv. Für eine effiziente Implementierung bietet sich die *Dynamische Programmierung* (siehe Abschnitt 2.4) an, die eine kostspielige Rekursion vermeidet.

2.3 Erkennung von Bewegungsmustern

In diesem Abschnitt werden die Grundlagen in Bezug auf die Erkennung von Bewegungsmustern in Trajektorien beschrieben. Dieses Problem wird als *Data Mining*-Aufgabe aufgefasst. Daher wird zunächst das prinzipielle Vorgehen samt der unterschiedlichen Phasen in Vorverarbeitung und Erkennung im Abschnitt 2.3.1 erläutert. Zur Vorverarbeitung von Trajektorien werden einige grundlegende Methoden genutzt. Die entsprechenden Grundlagen zur Filterung (Abschnitt 2.3.2), zur Segmentierung (2.3.3) und zur Bestimmung der Ähnlichkeit (2.3.4) von Trajektorien werden in den folgenden Abschnitten dargelegt. Abschließend werden in den Abschnitten 2.3.5 und 2.3.6 grundlegende Techniken im Zusammenhang mit der Erkennung von Mustern in Trajektorien vorgestellt.

2.3.1 Data Mining

Das Erkennen von Mustern in Daten ist eine typische Problemstellung im breiten Feld Data Mining. Diese befasst sich im Allgemeinen mit der Extraktion von Informationen, die sich in (meistens großen) Datenmengen verstecken und sich nicht auf einfachem Weg abfragen lassen. Hier kommen häufig Algorithmen aus der *Künstlichen Intelligenz* bzw. dem *Maschinellen Lernen*, wie z.B. die in Abschnitt 2.3.5 vorgestellten Clustering- oder Klassifikationsverfahren, zur Anwendung.

Der Begriff „Data Mining“ beschreibt eigentlich nur einen Teilschritt (wenn auch den Kern) des gesamten „*Knowledge Discovery from Data(bases)*“ (kurz *KDD*)-Prozesses. Er wird jedoch häufig als Synonym für den Gesamtprozess benutzt, welcher je nach Definition (es gibt hier unterschiedliche Varianten) aus drei, fünf oder sieben Teilschritten besteht. Die prinzipielle Abfolge sieht dabei eine anfängliche Vorverarbeitung, das eigentliche Data Mining und abschließend eine Ergebnisauf-

bereitung vor. Diese Aufteilung entspricht zugleich der Drei-Schritt-Variante. Bei den Varianten, die aus fünf bzw. sieben Schritten bestehen, werden die Vorverarbeitungs- und die Ergebnisaufbereitungsphase in weitere Teilschritte unterteilt. Abbildung 2.13 zeigt den typischen Ablauf der Sieben-Schritt-Variante nach Han u. a. (2011), wobei die letzten beiden Schritte (Ergebnisevaluation und Wissenspräsentation) zusammengefasst sind.

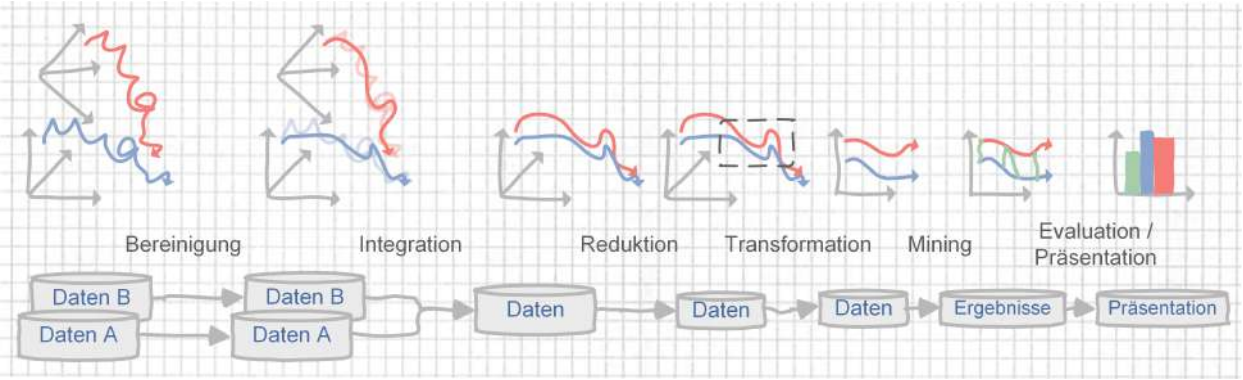


Abbildung 2.13: Eine schematische Darstellung der Phasen des KDD-Prozess.

Die Vorgehensweise zur Erkennung der Bewegungsmuster in Trajektorien in dieser Arbeit orientiert sich an dem KDD-Prozess. Dieser wird im Folgenden näher beschrieben. Entsprechend des Ablaufs werden die einzelnen Phasen kurz erläutert.

In der **Datenbereinigungsphase** (*Data Cleaning*) stehen die Fehlerbehandlung und der Umgang mit fehlenden Werten im Vordergrund. Die Fehlerbehandlung beinhaltet u.a. Methoden zur Glättung und zur Detektion und Eliminierung von Ausreißern. Der Umgang mit fehlenden Werten wird beispielsweise durch einfaches Ignorieren der entsprechenden Samples, falls die Datenmenge hinreichend groß ist oder durch ein entsprechendes Füllen der Lücken realisiert. Letzteres kann dabei entweder manuell (eignet sich bei kleineren Datenbeständen) durch das Einsetzen global festgelegter Werte bzw. Standardwerte oder durch Schätzung mittels Prädiktion, Inter- bzw. Extrapolation erfolgen.

In der **Datenintegration** (*Data Integration*) werden Daten unterschiedlicher Quellen bzw. Repräsentation und unterschiedlichen Eigenschaften zu einem gemeinsamen Datenbestand zusammengefügt. Dabei kommen Analysen zur Identifikation sich entsprechender Entitäten, zur Erkennung von Redundanzen und Korrelation als auch zur Identifikation und Beseitigung von Wertekonflikten zum Einsatz.

Die **Datenselektion bzw. -reduktion** (Data Reduction) verringert die Datenmenge um die für die übergeordnete Aufgabenstellung irrelevanten Daten. Die Dimensionsreduktion ist hierbei ein möglicher Weg. Dabei werden zum einen entweder durch die Transformation/Projektion der Daten in einen anderen Raum oder durch die manuelle Auswahl von Attributen/Dimensionen die Anzahl der Dimensionen verringert. Zudem kann die Datenmenge reduziert werden, indem parametrische Modelle (z.B. eine Regression) oder verschlankende Repräsentationsformen (z.B. Histogramme oder Cluster) anstelle der Rohdaten gespeichert werden. Zum anderen kann das Datenvolumen durch eine entweder verlustfreie oder verlustbehaftete Komprimierung reduziert werden.

In der **Datentransformation** (*Data Transformation*) als letzten Vorverarbeitungsschritt werden die Daten in ein geeignetes Format gebracht, das den darauffolgenden Data Mining-Schritt begünstigt. So können bspw. auf neue Attribute generiert werden, die entsprechende Berechnungen beschleunigen. Auch eine Aggregation einzelner Samples oder eine Normalisierung von Attributen sind Methoden, die in dieser Phase angewendet werden können.

Der Kern des KDD-Prozesses ist der **Data Mining**-Schritt. In diesem findet die eigentliche Extraktion der gesuchten Informationen statt. Hauptsächlich lassen sich die Aufgabenstellungen in zwei Kategorien unterteilen. In der einen werden Informationen zur Beschreibung bzw. zur Charakterisierung von Sachverhalten oder Objekten gesucht. In der anderen wird versucht, Gesetzmäßigkeiten in den Daten zu erkennen und daraus Regeln abzuleiten, die eine Prädiktion ermöglichen. Typische angewendete Verfahren sind die Klassifikation, die Regressionsanalyse, das Clustering, die Ausreißeranalyse oder das für diese Arbeit besonders relevante Erkennen von häufigen Mustern. Letzterem kann das Generieren von Assoziationsregeln und Ermitteln von Korrelationen folgen.

In den beiden nachfolgenden Schritten, der **Ergebnisevaluation** (*Pattern Evaluation*) und der **Wissenspräsentation** (*Knowledge Presentation*), werden die gewonnenen Informationen evaluiert und für eine Darstellung aufbereitet. Die Evaluation kann anhand von entsprechenden Metriken wie dem *support* bzw. der *confidence* (vgl. Abschnitt 2.3.6) durchgeführt werden. Zur Präsentation des neu generierten Wissens werden Techniken zur Visualisierung (z.B. durch Graphen, Diagramme oder einfachen Plots) und Repräsentation (z.B. durch geeignete Datenmodelle) verwendet.

2.3.2 Filterung und Glättung

Die Trajektorien, die mit den Tracking-Verfahren, die in Abschnitt 2.2 beschrieben werden, erfasst werden, sind typischerweise mit Unsicherheiten behaftet. Letztere können durch Sensorrauschen oder Messfehler entstehen. Vergleiche hierzu beispielsweise die Ungenauigkeiten, die bei einem GPS-Tracking entstehen können (Abschnitt 2.2.1). Je nach räumlicher Größenordnung, in der sich das jeweilige Anwendungsszenario bewegt, können diese Fehler entweder toleriert oder müssen reduziert werden. So kann ein GPS-Fehler im Bereich von 10 m toleriert werden, um herauszufinden, in welcher Stadt oder Straße sich ein überwachtes Objekt befindet. Ist es allerdings beabsichtigt, die Straßenseite zu bestimmen, auf der sich das Objekt befindet, so sind Ungenauigkeiten in dieser Größenordnung ein Problem. Ein weiteres Beispiel liefert die Sport- bzw. Fußballanalyse. Dort sind Fehler in der Positionsbestimmung im Bereich von wenigen Zentimetern bis wenigen Metern bei der Bestimmung der Spielfeldhälfte, in der sich der Spieler befindet, akzeptabel. Bei der Ermittlung des ballbesitzenden Spielers oder einer Abseitsstellung wären sie jedoch problematisch.

Um gewisse Frequenzanteile oder Rauschen aus Zeit- bzw. Datenreihen zu entfernen, werden im Allgemeinen Filter-Verfahren genutzt. Einige dieser Verfahren besitzen zudem die Eigenschaft des Glättens. Da es sich bei Trajektorien ebenfalls um Zeitreihen handelt, können solche Filter genutzt werden, um diese zu glätten, wobei etwaiges Rauschen reduziert und Ausreißer herausgefiltert werden. Im Folgenden werden zwei dieser Filter beschrieben. Der Mittelwert-Filter (auch gleitende Mittelwert genannt) dient zur Glättung von Datenreihen. Durch seine Anwendung werden höhere Frequenzanteile entfernt. Dazu wird eine neue Datenreihe erstellt, die aus den Mittelwerten gleich großer Untermengen der Ursprungsdatenmenge besteht. Der einfache gleitende Durchschnitt n-ter Ordnung m_t für eine diskrete Datenreihe $x(t)$ berechnet sich durch

$$m_t = \frac{1}{n} \sum_{i=0}^{n-1} x(t-i) \tag{2.31}$$

Dabei entsteht eine Verzögerung, mit der der Mittelwert der Datenreihe hinterherläuft, da das Intervall, aus dem der Mittelwert berechnet wird, ausschließlich aus vorausgegangen Daten besteht.

Liegen die Daten bei der Analyse bereits vollständig vor, so kann die Verzögerung verhindert werden, indem das Intervall um den entsprechenden Zeitpunkt zentriert wird.

$$m_t = \frac{1}{n} \sum_{i=0}^{n-1} x(t - (i + \frac{n}{2})) \tag{2.32}$$

Auf diese Weise ist jedoch ein Glätten in den Anfangs- ($t < \frac{n}{2}$) und Endbereichen ($t > t_{max} - \frac{n}{2}$) der Datenreihe nicht möglich. Bei der Anwendung auf Trajektorien wird das arithmetische Mittel der Positionen (Gl. 2.32) der Trajektoriepunkte bestimmt. Das Ergebnis einer geglätteten Trajektorie ergibt sich dabei durch

$$P(TP)_t = \frac{1}{n} \sum_{i=0}^{n-1} P(TP(t - (i + \frac{n}{2}))) \tag{2.33}$$

Der nichtlineare Median-Filter arbeitet analog zum Mittelwert-Filter mit dem Unterschied, dass dort anstatt des Mittelwerts der Median bestimmt wird. Dieser entspricht dem zentralen (mittleren) Element einer geordneten Auflistung von Zahlenwerten. Der Unterschied zwischen Mittelwert und Median wird anhand der kleinen Datenreihe $x = [4,2,2,4,5,50,3]$ verdeutlicht:

$$Mittelwert(x) = \frac{70}{7} = 10 \tag{2.34}$$

$$Median(x) = Median(2,2,3,\underline{4},4,5,50) = 4 \tag{2.35}$$

Wie auch in dem obigen Beispiel erkennbar ist, ist der Median-Filter gegenüber dem Mittelwert-Filter in Bezug auf Ausreißer ($x_5 = 50$) robuster, da nicht alle Werte bei der Berechnung berücksichtigt werden.

2.3.3 Segmentierung

Bei der Analyse von Objektbewegungen spielt die Segmentierung von Trajektorien eine wichtige Rolle. Häufig sind nur Teile der Trajektorien von Interesse, da beispielsweise ein entsprechendes Objekt nur dort interessantes Verhalten aufweist (siehe Beispiel in Abbildung 2.14). Unter einer

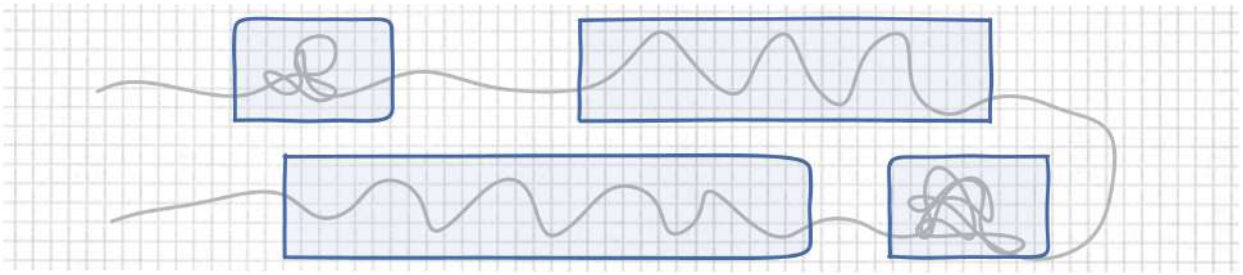


Abbildung 2.14: Segmentierung einer Trajektorie: Nicht immer ist die ganze Trajektorie bei einer Analyse von Interesse. So sind wie in diesem Beispiel häufig nur Teile der Trajektorie relevant, in denen beispielsweise spezielles Verhalten eines Objekts beobachtet werden kann. Um diese Teiltrajektorien zu ermitteln und auszuschneiden, können unterschiedliche Segmentierungsansätze genutzt werden.

Segmentierung ist allgemein die Zerlegung eines Ganzen in Segmente bzw. einzelne Abschnitte oder auch Teilstücke zu verstehen. Angewendet auf Trajektorien bedeutet dies, dass diese in einzelne Trajektoriesegmente zerteilt werden. Diese Segmente werden häufig auch Trajektoriestücke bzw. -teile oder auch Subtrajektorien genannt.

Eine Möglichkeit bei der Segmentierung ist die **Befragung von Domain-Experten**. Diese können abhängig vom jeweiligen Kontext die relevanten Bereiche der Trajektorien identifizieren. Für den Fall, dass kein Expertenwissen verfügbar ist, existieren verschiedene grundlegende Methoden, mit denen Trajektorien auch nur auf Basis ihrer eigenen Eigenschaften segmentiert werden können. Zu den einfachsten gehören die Verfahren, die die Trajektorien anhand einer **Anzahl von Trajektoriepunkten** unterteilen. So können beispielsweise Segmente gleicher Länge erzeugt werden. Auch die Segmentierung auf Basis von **Merkmalen** gehört zu den eher einfachen Lösungsansätzen. Als Merkmale können u.a. geometrische (z.B. Kurvigkeit oder Sinuosität) aber auch zeitabhängige Trajektorieeigenschaften (z.B. Geschwindigkeit oder Beschleunigung) dienen. Komplexere Methoden versuchen die Segmente in einer gewissen Hinsicht zu optimieren. So geht es dort beispielsweise darum, **möglichst wenig homogene Trajektoriesegmente** (Buchin u. a., 2012) zu finden. Mit Hilfe einer Funktion zur Berechnung von charakteristischen Eigenschaftswerten zu jedem Zeitpunkt der Trajektorie und definierten Kriterien bzgl. der maximalen Ausdehnungen dieser Werte, lassen sich Trajektorien in linearer Zeit optimal segmentieren. Ein anderer Ansatz segmentiert Trajektorien anhand von **minimalen umspannenden Rechtecken/Quadern** (Anagnostopoulos u. a., 2006). Auf diese Weise wird die Trajektorie in kleinere und einfachere Primitive zerlegt. Letztere bieten durch ihre bessere Eignung in Bezug auf Indexstrukturen zudem Vorteile bei Speicher- und Abrufprozessen. Die Ermittlung der Segmente ist bei diesen Ansätzen ein Optimierungsproblem, sodass es darum geht, die Kostenfunktion zu finden, die die Kosten in Form unterschiedlicher Eigenschaften der Rechtecke/Quader minimiert. Diese zu minimierende Eigenschaft kann einfachheitshalber die Fläche/das Volumen aber auch eine komplexere Kombination verschiedener Attribute sein. In Abbildung 2.15 werden die unterschiedlichen Segmentierungsansätze auf eine Beispieltrajektorie angewendet und es wird das jeweilige Segmentierungsergebnis dargestellt.

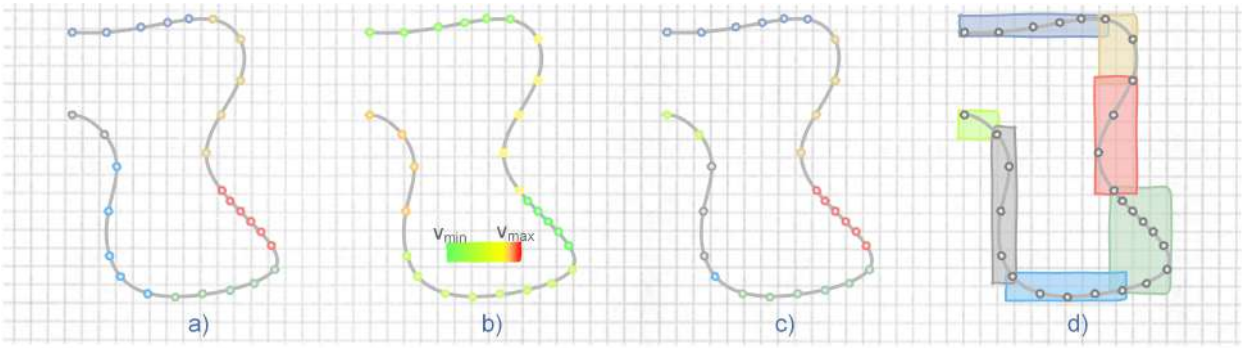


Abbildung 2.15: Eine Beispieltrajektorie wird mit Hilfe der beschriebenen Ansätze segmentiert. Es wird das Ergebnis symbolisch für eine Segmentierung gezeigt. Die Farben repräsentieren die resultierenden Segmente. Die Segmentierung erfolgt anhand einer konstanten Anzahl von Trajektoriepunkten (5 Punkte pro Segment) (a), eines Merkmals (hier: die Geschwindigkeit) (b), der Erzeugung möglichst wenig homogener Segmente (hinsichtlich der Kurvigkeit) (c) und minimaler umspannender Rechtecke (d).

2.3.4 Distanzmaße zur Bestimmung der Ähnlichkeit von Trajektorien

In vielen Aufgabestellungen spielt die Ähnlichkeit von Trajektorien eine wichtige Rolle. Ein Beispiel hierfür ist die Erkennung von typischen Objektbewegungen bzw. Bewegungsmustern (siehe Abschnitt 3.2.1). Dabei wird davon ausgegangen, dass sich ein typisches Bewegungsverhalten durch wiederkehrende, ähnliche Trajektorien auszeichnet. Die Bestimmung der Ähnlichkeit ist jedoch nicht trivial, da sie einerseits von den betrachteten Eigenschaften und andererseits von der Art und Weise, wie die jeweiligen Eigenschaften verglichen werden, abhängt. Die Auswirkung der Wahl der betrachteten Eigenschaften wird in Abbildung 2.16 beispielhaft erläutert. Je nach Wahl, z.B.

Lage (x, y) (a), Farbe (b), Form (c) oder einer Kombination aus diesen Eigenschaften (d), können die dort gezeigten Objekte unterschiedlich gruppiert werden.

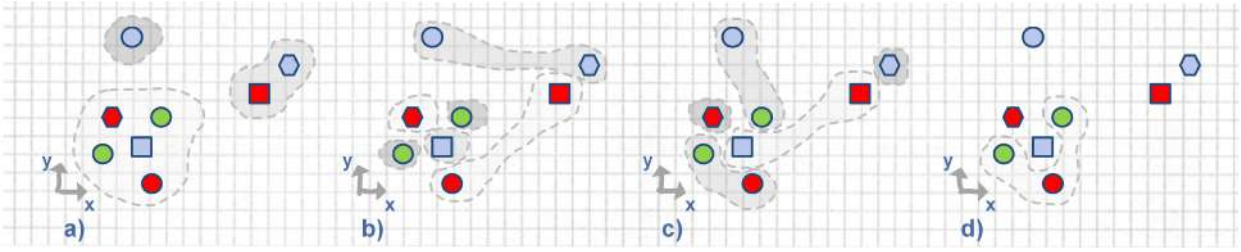


Abbildung 2.16: Ähnliche Objekte werden gruppiert: Die Gruppen werden durch die unterschiedlichen Grautöne in der Umrandung visualisiert. Die Ähnlichkeit von Objekten hängt von den betrachteten Eigenschaften ab: a) die Lage (x, y), b) die Farbe, c) die Form und d) die Kombination aus Lage und Form (hier wird übersichtshalber nur eine Gruppe dargestellt).

Übertragen auf Trajektorien als zu vergleichende Objekte bedeutet dies, dass zum einen die relevanten Eigenschaften und das Distanzmaß definiert sein müssen, um die Ähnlichkeit bestimmen zu können. Mögliche relevante Eigenschaften sind die Geometrie, die Zeit sowie daraus abgeleitete Eigenschaften, z.B. die Kurvigkeit (Gl. 2.8), die Geschwindigkeit bzw. Beschleunigung (Gl. 2.9) oder die Länge (Gl. 2.7). Zudem können ebenfalls zusätzliche annotierte Eigenschaften, z.B. abgegriffene Signale im CAN-Bus eines Autos, wie Blinker oder eingelegter Gang, oder mit Hilfe zusätzlicher Sensorik aufgezeichnete Umgebungswerte, wie z.B. Emissionswerte oder Regenmenge, berücksichtigt werden.
Bei der Wahl eines Distanzmaßes existiert eine Vielzahl von Möglichkeiten mit jeweils unterschiedlichen Eigenschaften, die beim Vergleich von Trajektorien in Frage kommen. Zu diesen gehören u.a. die *Euklidische Distanz*, die *Hausdorff-Distanz*, die *Fréchet-Distanz* oder die Distanz, die mit Hilfe von dem *Dynamic Time Warping* (DTW)-Algorithmus ermittelt wird. Diese werden im Folgenden hinsichtlich ihrer Berechnung sowie den Eigenschaften und Besonderheiten näher beschrieben und in Tabelle 2.3 vergleichend dargestellt.

Die **Euklidische Distanz** D_{Eucl} zwischen zwei Trajektorien ist ein globales Distanzmaß, was bedeutet, dass sie sämtliche Punkte auf den Trajektorien bei der Berechnung berücksichtigt. Sie berechnet sich durch die Summe der Euklidischen Distanzen zwischen korrespondierenden Punktpaaren auf beiden Trajektorien. Dieses setzt voraus, dass die Anzahl der Punkte n auf beiden Trajektorien identisch ist, sodass entsprechende Punktpaare ermittelt werden können. Die Ermittlung erfolgt in diesem Fall anhand der Reihenfolge der Punkte, wie sie in der Trajektorie enthalten sind. Für den in Abbildung 2.17 (links) dargestellten Fall errechnet sich die Distanz zwischen den Trajektorien A und B durch

$$D_{Eucl}(A,B) = \sum_{i=1}^{n} d(tp_{a,i},tp_{a,i}) \tag{2.36}$$

mit $tp_{a,i}$ bzw. $tp_{b,i}$ als i-ter Trajektoriepunkt von A bzw. B. und $d(\cdot,\cdot)$ als Euklidischer Abstand zweier Punkte. Die Euklidische Distanz ist ein metrisches Maß, wobei jedoch berücksichtigt werden muss, dass es sich dabei um eine Summe von Punktabständen handelt, die mit der Anzahl von Punkten auf jeder Trajektorie ansteigt. Dies bedeutet, dass die Distanz zwischen A und B mit steigender zeitlicher Auflösung und Anzahl an Punkten ansteigen würde. In der Literatur existiert eine weitere Variante der Euklidischen Distanz, bei der die Summe durch die Anzahl der Punktpaare geteilt wird. Diese so ermittelte Distanz entspricht damit einer durchschnittlichen Distanz

zwischen den Trajektorien und umgeht das zuvor geschilderte Problem, da in diesem Fall eine repräsentative Distanz bestimmt wird. Aus Gleichung 2.36 wird dadurch

$$D_{Eucl}(A,B) = \frac{1}{n} \sum_{i=1}^{n} d(tp_{a,i},tp_{a,i}). \tag{2.37}$$

Für beide Varianten muss jedoch beachtet werden, dass bei ungleicher Anzahl von Trajektoriepunkten ein Vorverarbeitungsschritt nötig ist, der mit Hilfe eines Resamplings die Punktanzahl angleicht. Die Berechnungskomplexität ohne Vorverarbeitung ist in beiden Fällen $O(n)$.

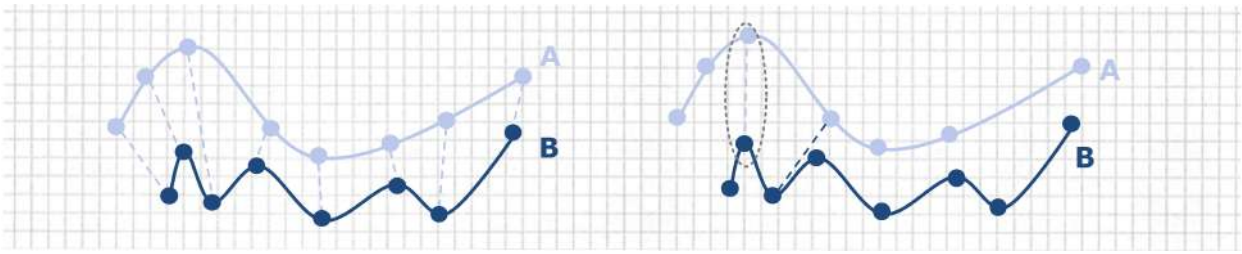


Abbildung 2.17: Schematische Darstellung der Euklidischen Distanz (links) und Hausdorff-Distanz (rechts) zwischen zwei Trajektorien. Links: Die gestrichelten Linien zeigen die korrespondierenden Punkte auf A und B an. Rechts: Die Markierung zeigt die Größere der beiden maximalen Punktdistanzen ($h(\cdot,\cdot)$) und damit die ermittelte Hausdorff-Distanz zwischen A und B an.

Auch die **Hausdorff-Distanz** D_H (Abbildung 2.17, rechts) ist ein globales Maß. Allerdings werden bei ihrer Berechnung keine korrespondierenden Punktpaare bestimmt, sondern jeder Punkt auf beiden Trajektorien mit jedem Punkt der jeweils anderen Trajektorie verglichen. Die resultierende Distanz D_H ist dann die Größere der maximalen aller minimalen Distanzen zwischen den gebildeten Punktpaaren:

$$D_H(A,B) = \max\{h(A,B),h(B,A)\} \tag{2.38}$$

mit

$$h(A,B) = \max_{tp_a \in A} \left\{ \min_{tp_b \in B} \{d(tp_a,tp_b\} \right\} \quad und \quad h(B,A) = \max_{tp_b \in B} \left\{ \min_{tp_a \in A} \{d(tp_a,tp_b\} \right\}. \tag{2.39}$$

Da durch (2.38) und (2.39) eine repräsentative Euklidische Distanz ermittelt wird, ist die Hausdorff-Distanz ein metrisches Maß. Auf Grund der Tatsache, dass bei der Berechnung alle möglichen Kombinationen von Punktpaaren in Betracht gezogen werden und dabei die Reihenfolge der Punkte außer Acht gelassen wird, hat die Richtung der Trajektorien bzw. der zeitliche Aspekt keinen Einfluss auf das Ergebnis. Dieses kann sich bei gewissen Analyseszenarien als problematisch erweisen. Die Berechnungskomplexität ist $O(nm)$.

Eine weitere Möglichkeit den Abstand zwischen Trajektorien zu bestimmen, ist die **Fréchet-Distanz** $D_{Frechet}$. Genau wie die beiden vorherigen Maße, ist auch diese ein globales Maß. Am einfachsten lässt sich die Fréchet-Distanz bildhaft anhand der Hundeleine-Analogie darstellen: Ein Herrchen geht mit seinem Hund an der Leine spazieren. Beide bewegen sich auf ihren Wegen (Trajektorien) und dürfen die Geschwindigkeit variieren und auch stehenbleiben, nicht aber rückwärtsgehen. Der Fréchet-Abstand entspricht dann der kürzest möglichen Leine.

Die Berechnung erfolgt durch

$$D_{Frechet}(A,B) = \inf_{\alpha,\beta} \max_{s,t} \{d(A(\alpha(s)),B(\beta(t)))\}. \tag{2.40}$$

Entsprechend der Analogie werden bei der Ermittlung der Fréchet-Distanz Paare aus Trajektoriepunkten gebildet und die Euklidische Distanz $d(\cdot,\cdot)$ berechnet. Bei der Bildung der Punktpaare ist zu beachten, dass die Punkte auf beiden Trajektorien der Reihe nach, beginnend beim Ersten, schrittweise abgearbeitet werden. Dabei ist es in jedem Schritt erlaubt, entweder auf einer der beiden Trajektorien oder aber auf beiden gleichzeitig fortzuschreiten bzw. den nächsten Punkt zu betrachten. Ein Rückschritt in der Zeit, also wieder zu dem den vorherigen Punkt zurückzukehren, ist nicht erlaubt. Die in (2.40) gegebenen Laufparameter s und t dürfen demnach nicht fallen. Die resultierende Distanz ist schließlich das Infimum der Maxima der möglichen Punktpaardistanzen.

Dieses Vorgehen kann mittels eines Free-Space-Diagramms visualisiert werden, welches eine $n \times m$-Rasterstruktur F_ϵ aufweist und wie folgt berechnet wird:

$$F_\epsilon = \{(s,t) \in [0,n] \times [0,m] \mid d(A(\alpha(s)),B(\beta(t)) \leq \epsilon\}\,. \tag{2.41}$$

Dabei sind n und m die Anzahlen der Trajektoriepunkte. Die einzelnen Zellen entsprechen je einem Punktpaar und werden durch Boolesche Werte (*true*, *false*) belegt. Überprüft wird dabei, ob die Punktdistanz kleiner oder gleich einer gegebenen kritischen Distanz ϵ ist. Im übertragenen Sinn bedeutet dies, dass sie entweder passierbar oder unpassierbar sind. Nach Betrachtung sämtlicher Kombinationen aus Punkten der jeweiligen Trajektorien, ist dieses Raster gefüllt und es kann geprüft werden, ob ein streng monoton steigender Pfad von Zelle (0,0) (links unten) zur Zelle (n, m) (rechts oben) existiert. Erlaubte Übergänge zwischen passierbaren Zellen sind also oben, rechts oder diagonal, was dem Fortschritt auf entweder einer (rechts, oben) oder beiden Trajektorien (diagonal) gleichzeitig entspricht. Existiert ein solcher Pfad, so ist ϵ ein gültiger kritischer Wert. Die Aufgabe besteht schließlich darin, den Kleinsten aller kritischen Wert zu finden, welcher der gesuchten Distanz entspricht. Eine Lösung ist das schrittweises Annähern von ϵ. In Abbildung 2.18 wird ein Beispiel (links) samt gewählten ϵ und dazugehörigem Free-Space-Diagramm (rechts) gezeigt. Für dieses Beispiel existiert ein Pfad, ϵ ist somit gültig.

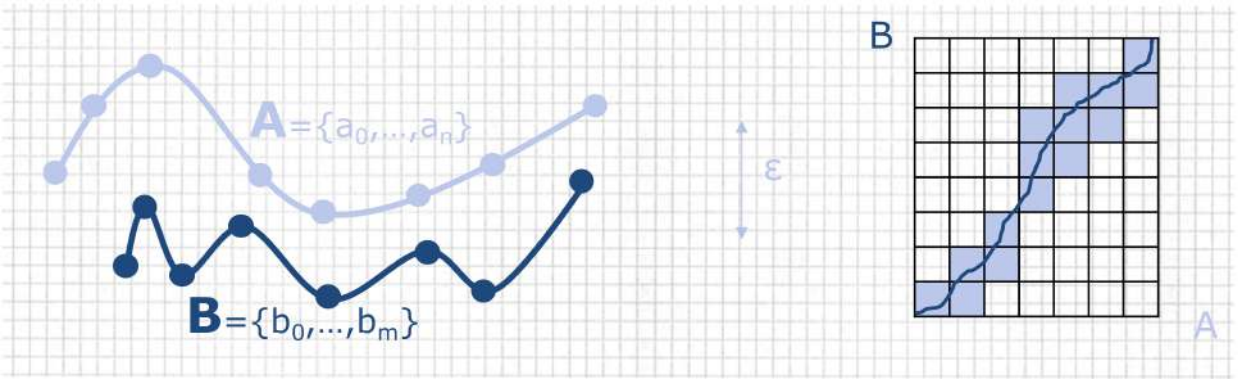


Abbildung 2.18: Schematische Darstellung der Berechnung der Fréchet-Distanz: Die Beispieltrajektorien A und B (links), der kritische Abstand ϵ (Mitte) und die entsprechende Darstellung als Free-Space-Diagramm (rechts).

Die Fréchet-Distanz ist ein metrisches Maß, da wie bei der Hausdorff-Distanz der repräsentative Euklidische Punktabstand gesucht wird. Zudem wird durch die Bedingungen an das Bilden der Punktpaare die Berücksichtigung der Zeit bzw. der Reihenfolge der Punkte gewährleistet. Die Berechnungskomplexität ist $O(nm \cdot \log nm)$.

Dynamic Time Warping (DTW) ist im Allgemeinen ein Algorithmus der Zeitreihen aufeinander abbildet. Dabei versucht er mit Hilfe von zeitlichen Stauchungen bzw. Dehnungen die zu vergleichende Zeitreihen derart zu „formen“, sodass sie der Vergleichsreihe entspricht. Die Ähnlichkeit zweier Zeitreihen ergibt sich durch das Ausmaß an nötigen „Verformungen“. Je geringer diese sind,

desto ähnlicher sind sich die Reihen. Dies lässt sich auch auf Trajektorien übertragen, da diese allgemein gesehen auch Zeitreihen sind. In Abbildung 2.19 wird dieses zugleich anhand eines Beispiels illustriert. Ähnlich wie bei der Fréchet-Distanz werden auch hier die Trajektorien gleichzeitig schrittweise Punkt für Punkt durchwandert, um Punktpaare zu bilden. In der Skizze sind diese durch rote durchgezogene Linien markiert. Die zeitlichen Verformungen in Form von „Stauchungen“ oder „Dehnungen“ ergeben sich dabei durch den Fortschritt auf beiden Trajektorien. Für den Fall, dass auf beiden Trajektorien gleichzeitig fortgeschritten wird, so wie es im Beispiel zu sehen ist, findet keine Verformung statt. Es wird jeweils Punkt 2 betrachtet. Wird im nächsten Schritt nur auf Trajektorie A vorangeschritten, so kommt es zu auf dieser zu einer Stauchung, da von ihr mehrere Trajektoriepunkte (2, 3) auf einen Punkt (2) der Trajektorie B abgebildet werden. Im umgekehrten Fall beim Voranschreiten wären auf einen Punkt der Trajektorie A mehrere Punkte der Trajektorie B abgebildet. Es kommt dadurch zu einer Dehnung von A.

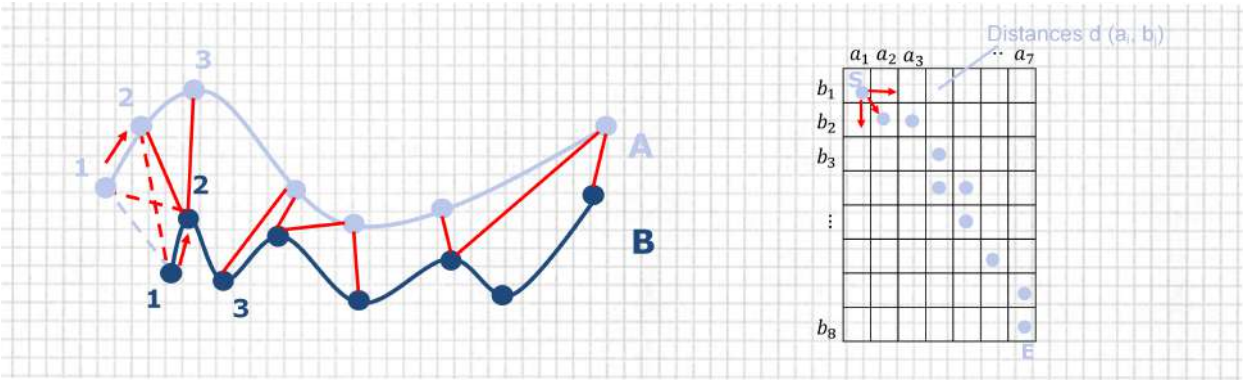


Abbildung 2.19: Schematische Darstellung der Distanz berechnet durch den Dynamic Time Warping-Algorithmus.

Dieses Verfahren lässt sich ähnlich der Fréchet-Distanz als zu füllendes $n \times m$-Raster darstellen (Abbildung 2.19, links). In diesem Fall beinhaltet jede Zelle jedoch die Distanz des jeweiligen Punktpaares addiert mit der Summer der Distanzen, die auf dem Weg zu dieser Zelle, ausgehen von Zelle (0,0), angefallen sind. Das Ergebnis lässt sich letztendlich in Zelle (n,m) ablesen, die das Ziel jedes Vergleichs ist. Für die Ermittlung eines Werts, der die Ähnlichkeit zweier Trajektorien beschreibt, ist dies bereits ausreichend. Soll zusätzlich die verzerrte Trajektorie (das Warping) bestimmt werden, ist ein Backtracking nötigt. Dabei wird der kostengünstigste Pfad (kleinste Distanz) zwischen Ziel- und Startzelle gesucht.

Berechnet wird die Distanz D_{DTW}, die aus dem DTW-Algorithmus resultiert, in rekursiver Weise wie folgt:

$$D_{DTW}(A,B) = \begin{cases} 0 \text{ wenn } n = 0 \text{ und } m = 0 \\ \infty \text{ wenn } n = 0 \text{ oder } m = 0 \\ d\left(Head\left(A\right),Head\left(A\right)\right) + \min \begin{cases} D_{DTW}(A,Rest(B)) \\ D_{DTW}(Rest(A),B) \\ D_{DTW}(Rest(A),Rest(B)) \end{cases} \end{cases} \tag{2.42}$$

mit

$$\begin{aligned} A &= a_1,...,a_n, \quad Head(A) = a_1, \quad Rest(A) = a_2,...a_n \\ B &= b_1,...,b_n, \quad Head(B) = b_1, \quad Rest(B) = b_2,...B_n \end{aligned} \tag{2.43}$$

Diese Abstandsfunktion liefert keine repräsentative Distanz, sondern ermittelt die kostengünstigste Abbildung zwischen beiden Trajektorien. Wird für $d(\cdot,\cdot)$ die Euklidische Distanz verwendet, so ist das Ergebnis eine metrische Distanz. Dieses ist aber nicht zwingend der Fall, da prinzipiell beliebige

Kostenfunktionen genutzt werden können. Die Reihenfolge der Punkte wird in diesem Fall ebenfalls berücksichtigt. Die Berechnungskomplexität ist $O(n^2)$. Eine effiziente Implementierung ermöglicht die *Dynamische Programmierung* (siehe Abschnitt 2.4), die kostspielige Rekursionen vermeidet.

In Tabelle 2.3 werden die beschriebenen Maße samt ihren Besonderheiten und Eigenschaften noch einmal übersichtlich dargestellt. Sowohl die Relevanz der Eigenschaften als auch die Eignung eines Distanzmaßes hängt dabei vom jeweiligen Kontext ab.

Tabelle 2.3: Übersicht über die beschriebenen Distanzmaße

Distanzmaß	**Euklidische Distanz**	**Hausdorff-Distanz**	**Fréchet-Distanz**	**DTW**
Anforderungen	Gleiche Anzahl an Trajektoriepunkten nötig	keine	keine	keine
Berücksichtigung der Zeit / Reihenfolge	ja	nein	ja	ja
Repräsentative Distanz	nein / ja	ja	ja	nein
Berechnungskomplexität	$O(n)$	$O(nm)$	$O(nm \cdot \log nm)$	$O(n^2)$

2.3.5 Maschinelles Lernen im Kontext raum-zeitlicher Daten

Das Forschungsfeld des **Maschinellen Lernens** ist ein Teilgebiet der *Künstlichen Intelligenz* und beschäftigt sich mit der automatischen Generierung von neuem Wissen auf Basis einer aktuellen Wissensbasis (auch *Erfahrung* genannt). Hierfür werden aus dem vorhandenen Wissen Regeln und Mustern induziert, anhand derer das neue Wissen abgeleitet wird. Typische Anwendungsbereiche des Maschinellen Lernens sind u.a. *Computer Vision*, *Sprach- und Texterkennung*, *Empfehlungsdienste*, *Diagnosesysteme*. Die Methoden zur Generierung des Wissens werden Lernverfahren genannt und lassen sich in unterschiedliche Lernansätze unterteilen: das *unüberwachte*, *überwachte* und *bestärkende* Lernen (Murphy, 2012). In Letzterem wird ein System mittels einer Belohnungs- bzw. Bestrafungsstrategie dazu gebracht, in den jeweiligen Situationen entsprechend zu reagieren. Dieser Ansatz ist im Zusammenhang dieser Arbeit nicht von Bedeutung und wird daher nicht weiter berücksichtigt. Die beiden anderen werden im Folgenden detailliert betrachtet.

Verfahren des **unüberwachten Lernens** generieren auf Basis der ihnen zur Verfügung gestellten Eingabedaten (*Samples*) ein Modell, das diese Daten bestmöglich beschreibt. Mit Hilfe dieses Modells lassen sich Vorhersagen bzgl. neuer Daten machen. In diesem Kontext werden häufig *Clustering*-Verfahren genutzt, die es zum Ziel haben, die Daten anhand ihrer Ähnlichkeit bzgl. der relevanten Eigenschaften zu gruppieren. Ergebnis einer Cluster-Analyse ist daher ein Clustering, in dem die einzelnen Instanzen der Daten (*Samples*) zu Clustern zusammengefasst sind. Im Gegensatz zum überwachten Lernen werden die Samples jedoch keiner speziellen Klasse zugeordnet (Murphy, 2012). Ein Beispiel hierzu wird in Abbildung 2.16 gezeigt. In Teil a) dieses Beispiels werden die Objekte nach ihrer Lage im Raum geclustert. Die für die Bestimmung der Ähnlichkeit relevanten Eigenschaften, in vektorieller Schreibweise auch Merkmalsvektor genannt, sind dabei die räumlichen Koordinaten (x, y) der Objekte. Als Maß für die Ähnlichkeit wird dabei die einfache Euklidische Distanz zwischen Punktobjekten benutzt.

Clustering-Verfahren

Beim Trajektorie-Clustering müssen die für die Ähnlichkeit relevanten Eigenschaften angepasst werden. Die in Abschnitt 2.3.4 genannten Trajektorie-Merkmale können zu diesem Zweck verwendet werden. Dieses führt dazu, dass die vom Clustering-Verfahren genutzte Distanzmetrik, bei

räumlichen Daten üblicherweise die Euklidische Distanz, entsprechend durch ein für die Problemstellung geeignetes Distanzmaß ersetzt werden muss.

Im Allgemeinen lassen sich Clustering-Verfahren nach ihren unterschiedlichen Ansätzen in *partitionierende* und *hierarchische* Verfahren kategorisieren (Kaufman und Rousseeuw, 2009), wobei Letztere in dieser Arbeit keine Berücksichtigung finden. Ein bekanntes partitionierendes Verfahren ist der *k-means*-Algorithmus, der den Merkmalsraum entsprechend einer gegebenen Menge an Clusterzentren (auch *Zentroide* genannt) aufteilt. Die Samples werden dabei dem Cluster zugeordnet, deren Zentrum sie am nächsten sind. Durch die iterative Anpassung der Lage der Zentroide, die wiederum durch die Zuordnung der Samples bedingt ist, ist eine Neuzuordnung der Samples zu einem anderen Zentrum möglich. Dieses Verfahren wird im Folgenden näher betrachtet.

Der **k-means-Algorithmus** (MacQueen, 1967) lässt sich formal beschreiben durch

$$\underset{X}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in X_i} \|x - c_i\|^2 \tag{2.44}$$

mit c_i als Clusterzentren und X als Menge der zu clusternden Samples. Es wird die Konstellation aus den k Zentroiden gesucht, die den Abstand zu den Samples minimiert.

Der Lösung dieses Minimierungsproblems wird sich beim k-means-Algorithmus iterativ angenähert. Begonnen wird jedoch mit einem Initialisierungsschritt, in dem die Position der k Clusterzentren festgelegt wird.

$$c_1^{(1)}, \ \ldots, \ c_k^{(1)} \tag{2.45}$$

Dieser Schritt ist wichtig, da er einen großen Einfluss auf das Konvergenzverhalten und damit auf die Laufzeit des Algorithmus hat. Grundsätzlich existieren zwei unterschiedliche Strategien, wie die Zentroide initial verteilt werden können. Beim „Forgy“-Ansatz werden zufällig k Samples als Clusterzentren definiert und der Algorithmus beginnt mit dem Zuordnungsschritt. Dieses führt im Allgemeinen zu einer breiteren Verteilung der Zentren im Raum. Der „Random“-Ansatz hingegen ordnet die Samples zufällig einem Cluster zu. Der Algorithmus beginnt dann mit dem Update-Schritt. Diese Strategie führt dazu, dass die Elemente der Cluster jeweils gleichmäßig über den gesamten Raum verteilt sind. Die sich daraus ergebenden Zentren befinden sich dementsprechend nahe dem Zentrum des Datensatzes. Welche der beiden Strategien zielführender ist, hängt allerdings von den zu clusternden Daten und der Aufgabenstellung ab. Hier ist Vorwissen über die Daten von Vorteil.

Nach der Initialisierung besteht jede Iteration t aus einem Zuordnungsschritt

$$X_i^{(t)} = \left\{ x_p : \left\| x_p - c_i^{(t)} \right\|^2 \leq \left\| x_p - c_j^{(t)} \right\|^2 \quad \forall j, 1 \leq j \leq k \right\}, \tag{2.46}$$

in dem jedes Sample einem Cluster zugeordnet wird, und einem Update-Schritt

$$c_i^{(t+1)} = \frac{1}{\left\| X_i^{(t)} \right\|} \sum_{x_j \in X_i^{(t)}} x_j, \tag{2.47}$$

in dem die Position der Zentroide auf Basis der Zuordnung der Samples neu ermittelt wird. Hierbei handelt es sich um die mittlere Position der zugeordneten Samples. Der Algorithmus terminiert, sobald entweder eine definierte Anzahl an Iterationen erreicht worden ist oder die Änderung der Position der Zentren entsprechend gering ausfällt. Abbildung 2.20 zeigt einen beispielhaften Ablauf des k-means-Algorithmus, wobei $k = 4$ gewählt wurde. Hier sind lediglich die Zustände nach der Initialisierung, nach der ersten Iteration und nach der letzten Iteration dargestellt. Die Clusterzu-

gehörigkeit wird durch die Farben symbolisiert. Die Zentroide, deren Verlauf über die Iterationen in grau gezeigt wird, sind als größere Kreise dargestellt. Im letzten Zustand (ganz rechts) werden zusätzlich die Clustergrenzen gezeigt. Die dadurch entstehende Partitionierung des Raums entspricht einem Voronoi-Diagramm, in dem die Zentroide als Zentren fungieren.

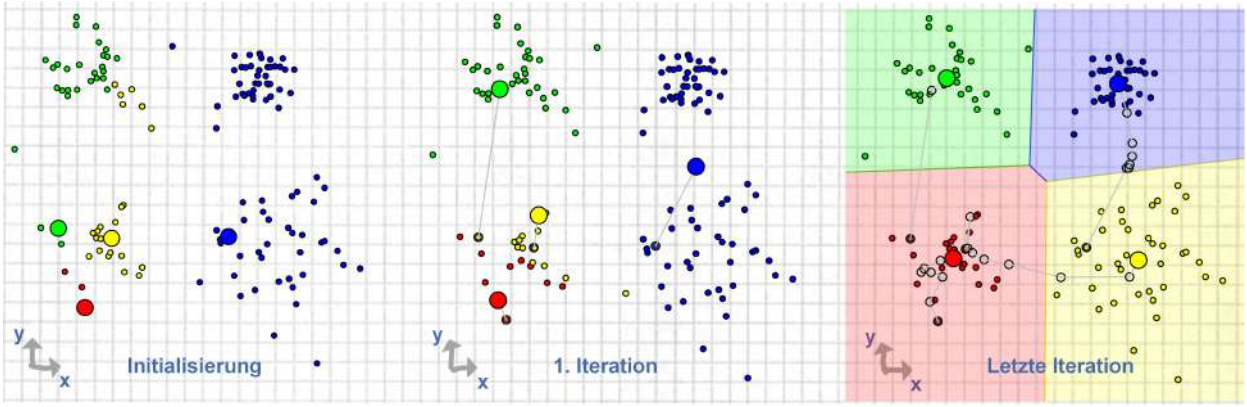


Abbildung 2.20: Der Ablauf des k-means-Algorithmus: die Initialisierung, das Zwischenergebnis nach der ersten Iteration, schließlich das Ergebnis nach Terminierung. Die Iterationen dazwischen werden nicht dargestellt. Die Farbe symbolisiert die Clusterzugehörigkeit. Der in grau dargestellte Verlauf zeigt die Historie der Zentroide.

Durch die Definition von k Zentroiden ist die Anzahl der resultierenden Cluster ebenfalls festgelegt. Dies ist bei der Wahl eines Clustering-Verfahrens bei gegebener Problemstellung zu berücksichtigen. Die Form der Cluster entspricht den Zellen einer Voronoi-Zerlegung des Raums. Das Ergebnis eines k-means-Clusterings ist eine Näherungslösung, bei der nicht garantiert ist, dass sie dem globalen Optimum entspricht. Die Qualität der Lösung hängt dabei stark von den Daten aber auch von der Initialisierung der Zentren ab. Zudem werden in diesem Verfahren Ausreißer bzw. fehlerhaften Daten nicht besonders berücksichtigt und haben direkten Einfluss auf das Ergebnis. Die Laufzeit beträgt $O(nkdi)$, mit n als Anzahl der Samples, d als Dimension der Daten und i als Anzahl der benötigten Iterationen.

Dichte-basierte Verfahren versuchen die Cluster anhand von Dichteunterschieden in den Daten zu bilden. Ein bekanntes Verfahren aus dieser Kategorie ist der **D**ensity-**B**ased **S**patial **C**lustering **A**lgorithm with **N**oise-Algorithmus (**DBSCAN**) nach Ester u. a. (1996). Dieser hat als Ziel, Samples, die sich in einem Gebiet mit entsprechend hoher Dichte befinden, zu Clustern zusammenzufassen. Die erforderliche minimale Dichte wird mit Hilfe des Eingabeparameters ϵ bestimmt, wie später beschrieben wird. Zudem werden Daten als Rauschen (*noise*) markiert, die die Anforderungen hinsichtlich einer erforderlichen Anzahl von Samples, die zur Bildung eines Clusters benötigt werden, nicht erfüllen. Die erforderliche Anzahl an Samples wird durch einen zweiten Eingabeparameter *minIts* gegeben.

Die Idee des DBSCAN-Algorithmus basiert auf der Klassifikation der Samples anhand der folgenden Bedingungen. In Abbildung 2.21 a) wird das Prinzip noch einmal schematisch dargestellt.

- Zwei Samples sind *direkt erreichbar* (*directly reachable*), wenn ihr Abstand, i.A. die Euklidische Distanz, kleiner oder gleich ϵ ist. (Siehe bspw. Punkte C und D in Abbildung 2.21a).
- Ein Sample ist ein *Kernpunkt* (*core point*), wenn von ihm mindestens $minIts - 1$ Samples *direkt erreichbar* sind. (Siehe bspw. Punkt A)
- Zwei Samples sind *Dichte-erreichbar* (*reachable*), wenn zwischen diesen ein Pfad über ausschließlich *direkt erreichbaren Kernpunkten* existiert. Die zwei Samples selbst müssen dabei keine *Kernpunkte* sein. (Siehe Pfad A-B-C-D)

– Ein Sample ist ein *Randpunkt* (*border point*), wenn es *Dichte-erreichbar* jedoch kein *Kernpunkt* ist. (Siehe Punkt D)

– Ein Sample ist *Rauschen* (*noise*), wenn es von keinem Kernpunkt *Dichte-erreichbar* ist. (Siehe Punkt E)

Die Cluster ergeben sich nach der Klassifikation aus allen Dichte-erreichbaren Kern- und Randpunkten.

In Abbildung 2.21 wird zudem der Einfluss des Parameters ϵ in b) $\epsilon = 3$ und c) $\epsilon = 4$ dargestellt. Während in b) das Clustering noch feiner ist, sodass kleinere Cluster relativ dicht nebeneinander liegen, ist es in c) deutlich grober. Dort sind die kleineren Cluster zusammengefasst worden. Jedoch werden gleichermaßen Samples, die in b) noch als Rauschen markiert worden sind, nun zu Clustern zugeordnet. Die Wahl von ϵ hängt daher vom jeweiligen Anwendungsfall ab. Als Hilfestellung für eine sinnvolle Wahl von ϵ kann *k-nearest-neighbor-distance-Diagramm* genutzt werden. Für die Bestimmung des Parameters $minIts$ gilt die Faustformel $minIts = D + 1$, wobei D die Dimension der Daten ist.

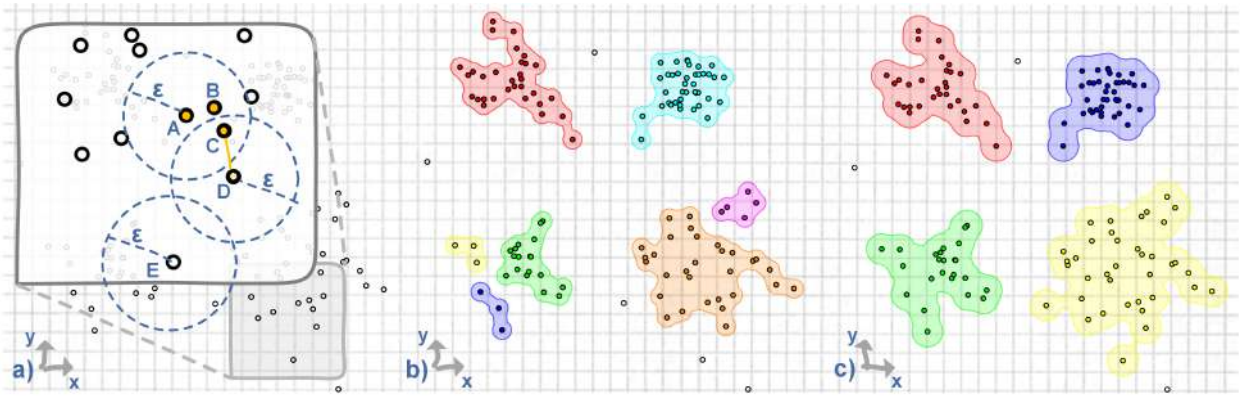


Abbildung 2.21: a) Das Prinzip des DBSCAN-Algorithmus. b) und c) Je nach Wahl des Parameters ϵ entstehen unterschiedliche Ergebnisse.

Durch den dichte-basierten Ansatz ist der DBSCAN-Algorithmus in der Lage, Ausreißer in den Daten zu erkennen und als Rauschen zu markieren. Sie besitzen keinen Einfluss auf die Berechnung der Cluster, wie es beim k-means-Verfahren der Fall ist. Zugleich ist es möglich, Cluster in ihrer natürlichen Form zu ermitteln, sodass dieser Algorithmus bei Szenarien, wie in Abbildung 2.22 (links) dargestellt wird, zu intuitiveren Lösungen kommt als das k-means-Verfahren. Neben diesen positiven Eigenschaften, wirft dieses Verfahren jedoch auch Probleme auf, sobald Datensätze mit bereichsweise inhomogener Dichte analysiert werden sollen. Dort ist die Wahl eines geeigneten ϵ schwierig: Ist ϵ zu groß, wird nur ein gesamtes Cluster gefunden. Ist ϵ hingegen zu klein, so entstehen im weniger dichten Gebiet, dass eigentlich als zu einem Cluster zusammengefasst werden soll, viele Kleine und entsprechend Rauschen. Die Laufzeitkomplexität des DBSCAN-Algorithmus beträgt im Allgemeinen $O(n^2)$, wobei n die Anzahl der Samples ist. Jedoch kann diese mit Hilfe von räumlichen Indexstrukturen auf $O(n \log n)$ reduziert werden.

In der folgenden Übersicht werden die erläuterten Verfahren noch einmal bezüglich ihrer Eigenschaften und Besonderheiten vergleichend dargestellt.

Das **überwachte Lernen** unterscheidet sich dahingehend, dass das jeweilige Lernverfahren neben den Samples auch die dazugehörige Ausgabe (*Klasse*) erhält. Es bekommt also assoziierte Ein- und Ausgabedaten, sogenannte Trainingsdaten. Anhand derer können die Regeln zur Assoziierung neuer Daten bzw. zur Generierung neuen Wissens gelernt werden. Das Ergebnis ist eine *Klassifikation*,

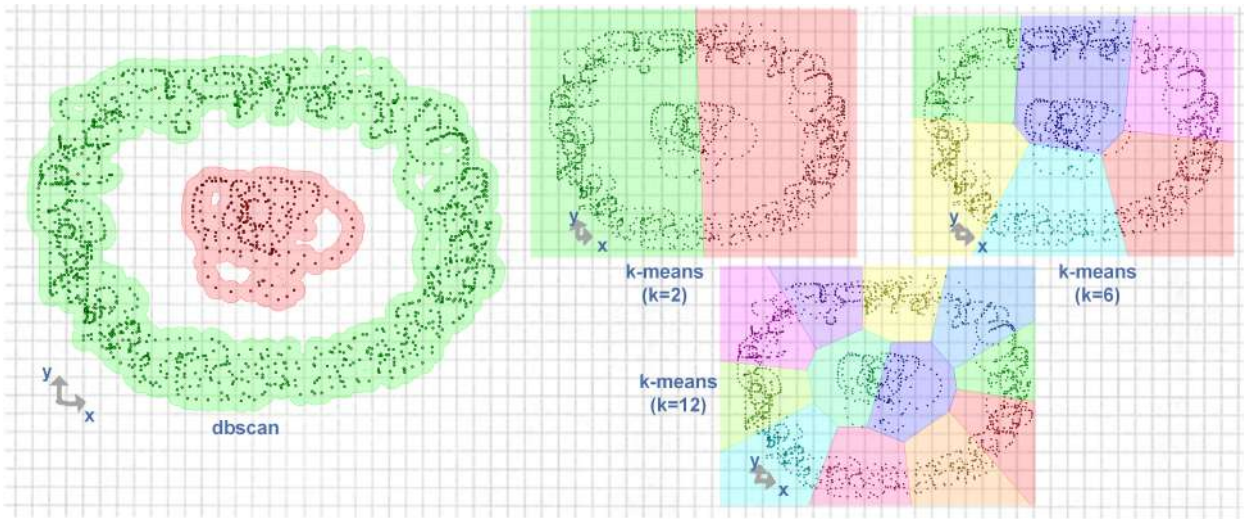


Abbildung 2.22: Der DBSCAN-Algorithmus ist in der Lage, Cluster in ihrer natürlichen Form zu ermitteln (links), wohingegen partitionierende Verfahren wie der k-means-Algorithmus Probleme haben, intuitive Lösungen zu liefern (rechts).

Tabelle 2.4: Vergleich der beschriebenen Clustering-Verfahren

Clustering-Verfahren	**Eingabeparameter**	**Positive Eigenschaften**	**Probleme**	**Laufzeitkomplexität**
k-means	k: Anzahl Cluster	Partitionierung des Raums (Voronoi) Einfache Umsetzung	Ausreißeranalyse Natürliche Cluster-Formen	$O(nkdi)$
DBSCAN	ϵ: Distanz $minIts$: minimale Anzahl Samples pro Cluster	Ausreißeranalyse Natürliche Cluster-Formen	Daten mit inhomogenen Dichten	$O(n \cdot \log n)$

man spricht daher auch von *Klassifikationsverfahren* (Murphy, 2012). Beispiele für solche Verfahren sind u.a. *Lineare Klassifikatoren*, *Support Vector Machines* und auch Verfahren zum Lernen von *Entscheidungsbäumen* (z.B. *ID3*, *C4.5* oder auch *Random Forest*) (Witten u. a., 2016).

ID3-Algorithmus zur Klassifikation

Zur Klassifizierung von Daten kann der *ID3* (*Iterative Dichotomiser* 3)-Algorithmus nach Quinlan (1986) genutzt werden. Auf Grundlage der verfügbaren und als *Attribut-Wert-Listen* vorliegenden Trainingssamples minimiert dieser Algorithmus die Anzahl der verwendeten Merkmale, die für eine hinreichende Klassifikation benötigt werden. Der Algorithmus arbeitet als iterativer Top-down-Prozess, indem zur Erzeugung des nächsten Baum-Knotens in jeder Iteration unter den Verbleibenden das Attribut mit dem höchsten *Informationsgewinn* bestimmt wird. Der Informationsgewinn (*information gain*) wird anhand der *Entropie* berechnet. Dabei ist das Attribut mit dem höchsten Informationsgewinn das mit der kleinsten Entropie. Die Entropie H_i des i-ten Merkmals, angegeben in *bit*, berechnet sich durch

$$H_i = -p_a \; I = -p_a \; log_2 \; p_a = -\frac{Anz_{a,i}}{Anz_{a,alle}} \log_2 \; p_a \; \frac{Anz_{a,i}}{Anz_{a,alle}}, \tag{2.48}$$

mit der $p_a = \frac{Anz_{a,i}}{Anz_{a,alle}}$ als Wahrscheinlichkeit der Klasse a, die durch die relative Häufigkeit von Samples mit der Ausprägung i zu der Gesamtheit an Samples der Klasse a ausgedrückt wird. Durch $I = log_2\ p_a$ ist die Information gegeben. Die Gesamtentropie eines Merkmals $H_{Feature}$ berechnet sich durch

$$H_{feature} = \sum_{i=1}^{m} p_i \cdot H_i = \sum_{i=1}^{m} \frac{Anz_i}{Anz_{alle}} \cdot H_i. \tag{2.49}$$

Hierbei entspricht m der Anzahl an unterschiedlichen Merkmalsausprägungen und p_i der Wahrscheinlichkeit der i-ten Ausprägung, die sich erneut durch die relative Häufigkeit Samples mit der Ausprägung i zu allen existierenden Samples ausdrückt. Mit Hilfe der Merkmalsentropie und einer A-priori-Klassen-Entropie $H_{apriori}$, die durch

$$H_{apriori} = \sum_{i=1}^{m} H_i \tag{2.50}$$

ermittelt wird, kann der Informationsgewinn IG bestimmt werden.

$$IG = H_{apriori} - H_{feature} \tag{2.51}$$

Das Ergebnis des Trainings ist ein Entscheidungsbaum, der auf die Testdaten angewendet wird. Dabei werden die jeweiligen Merkmale der Testsamples an den Knoten des Baums, beginnend bei der Wurzel, ausgewertet, bis das entsprechende Label in einem der Blätter erreicht wird. Dieses Label entspricht dem Klassifikationsergebnis. Der ID3-Algorithmus gilt als Grundlage für den C4.5-Algorithmus (Quinlan, 1993), der einige Verbesserungen, wie bspw. die Anwendbarkeit von kontinuierlichen Attributen, die Handhabung fehlender Attributswerte, usw. besitzt.

2.3.6 Sequenzmustererkennung

Die Sequenzmustererkennung als Teilbereich des *Data Mining* beschäftigt sich allgemein damit, Muster in Abfolgen von üblicherweise großen Datenmengen zu finden. Bei diesen Daten kann es sich um Transaktionsdaten (siehe Tabelle 2.5), Datenströme (z.B. Sensordaten oder Daten aus der Videoüberwachung), räumliche Daten (z.B. Karten) oder zeitbezogene Daten (z.B. Zeitreihen, biologische Sequenzdaten oder Börsenhandelsdaten) handeln. Während bei Datenströmen und zeitbezogenen Daten die Elemente einzeln sequenziell verarbeitet werden, werden die Elemente in Transaktionsdaten transaktionsweise prozessiert. Eine Transaktion besteht dabei aus einer ungeordneten Menge an Elementen, die auch *Itemset* genannt wird. Ein häufig genanntes Beispiel hierfür sind Warenkorbdaten im Online-Handel, wo verschiedene Produkte von Kunden zusammen gekauft werden. Anhand dieser Daten lassen sich mit Hilfe extrahierter Produktassoziationsregeln Schlüsse über den Kunden bzw. dessen Kaufverhalten ziehen. So kann bspw. ermittelt werden, welche Produkte in der Regel zusammengekauft werden. In dem in Tabelle 2.5 gegebenen Beispiel wird das Produkt „B“ insgesamt viermal gekauft. In 75 % der Fälle wird zudem auch das Produkt „C“ gekauft. Diese Informationen sind vor allem für die Händler interessant, die entsprechend Produktvorschläge machen können.

Tabelle 2.5: Ein Beispiel für Transaktionsdaten: Die Transaktionen mit den IDs 1-7 beinhalten jeweils eine ungeordnete Menge an Items (hier: A-F)

Transaktions-ID	**Liste der Items: Itemset**
T1	{A,B}
T2	{**B**,**C**,D,E}
T3	{C,E,A}
T4	{**B**,F,E,**C**}
T5	{F}
T6	{F,E,A}
T7	{A,**B**,**C**,F}
...	...

In dieser Arbeit liegt der Fokus jedoch auf der Verarbeitung von zeitbezogenen, sequentiellen Daten eines Datensatzes D, der aus einer (Gesamt-) Sequenz S_{tot} aus n_{tot} Elementen I_i besteht.

$$D = S_{tot} = \{I_0,I_1,\ldots,I_{n_{tot}-1}\} \tag{2.52}$$

Die Elemente I_i von S_{tot} stammen dabei aus einem *Alphabet*, das sämtliche unterschiedliche Ausprägungen der Elemente beinhaltet. In dem obigen Beispiel entspricht es der Menge an Symbolen/Zeichen $\{A,B,C,D,E,F\}$. Die Länge einer Sequenz wird durch die Anzahl ihrer Elemente bestimmt.

$$l(S) = \|S\| \tag{2.53}$$

S_{tot} hat demnach die Länge n_{tot}. Die Reihenfolge der Elemente ist im Gegensatz zu denen eines Itemsets nicht beliebig und spielt bei der Analyse eine wichtige Rolle. Eine Teilsequenz S (auch Subsequenz genannt) ist ein Teilstück, also eine Abfolge von direkt aufeinander folgenden Elementen (k bis $k+n$), einer übergeordneten Sequenz, wie z.B. S_{tot}, zu verstehen. S besitzt die Länge n.

$$S = \{I_k,I_{k+1},\ldots,I_{k+n}\} \subset S_{tot},\ \ mit\ \ k \leq 0 \wedge k+n \leq n_{tot} \tag{2.54}$$

Das Ziel der Mustererkennung ist es, in diesen sequentiellen Daten wiederkehrende Teilsequenzen zu finden. In diesem Zusammenhang lässt sich ein Muster P definieren durch

$$P = \{S_0,S_1,\ldots,S_m\}\,, \tag{2.55}$$

wobei m für die Anzahl an Sequenzen S_i steht, die in dem gesamten Datensatz enthalten und dem Muster zuzuordnen sind. Man spricht hier auch vom *Support Count (suppc)*

$$suppc(P) = m \tag{2.56}$$

oder auch, relativ bezogen auf die Gesamtanzahl n_{tot} der Sequenzen im Datensatz, vom *Support (supp)*:

$$supp(P) = \frac{m}{n_{tot}}. \tag{2.57}$$

Als Parameter für die Mustererkennung werden für die Mindestanzahl an Teilsequenzen $suppc_{min}$ sowie für die Minimallänge der Teilsequenzen l_{min} jeweils Schwellwerte benötigt. Eine Menge M einer wiederkehrenden Teilsequenz S bildet dann ein Muster, wenn gilt:

$$suppc(M) \geq suppc_{min}, \tag{2.58}$$

$$l(S) \geq l_{min}. \tag{2.59}$$

Bei Gültigkeit von (2.58), spricht man auch von einer *häufigen* Teilsequenz S. Die Menge der n_P gefundenen Muster $\mathcal{P}$ nach Abschluss der Suche wird wie folgt angegeben:

$$\mathcal{P} = \{P_0, P_1, \ldots, P_{n_P}\} \tag{2.60}$$

Die Teilmenge aller Muster, bei denen $suppc = 3$ und $l = 2$ gilt, wird durch $\mathcal{P}_{3,2} \subseteq \mathcal{P}$ gekennzeichnet und enthält $P(suppc = 3, l = 2)$ (in kurz: $P(3,2)$)-Muster.

An der folgenden einfachen Beispielsequenz soll Vorheriges verdeutlicht werden.

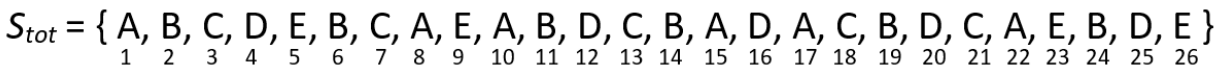


Abbildung 2.23: Beispielsequenz aus 26 Elementen

Sind als Parameter $suppc_{min} = 2$ und $l_{min} = 3$ gegeben, so bedeutet dies, dass sämtliche Muster mit der Mindestlänge 3 gesucht sind, die mindestens zweimal auftreten. Es ergibt sich daher die Menge $\mathcal{P} = \mathcal{P}_{2,3} = \{\{B,D,C\},\{C,A,E\}\}$ an identifizierten Mustern (siehe Abbildung 2.24, oben). Werden die Parameter verändert, indem beispielsweise l_{min} auf 2 herabgesetzt wird, verändert sich die Ergebnismenge $\mathcal{P}$ zu $\mathcal{P} = \mathcal{P}_{2,2} = \{\{B,D,C\},\{C,A,E\},\{B,D\},\{D,C\},\{C,A\},\{A,E\},\{A,B\},\{B,C\},\{C,B\},\{E,B\},\{D,E\}\}$ (siehe Abbildung 2.24, unten). Es gilt demnach $\mathcal{P}_{2,3} \subseteq \mathcal{P}_{2,2}$.

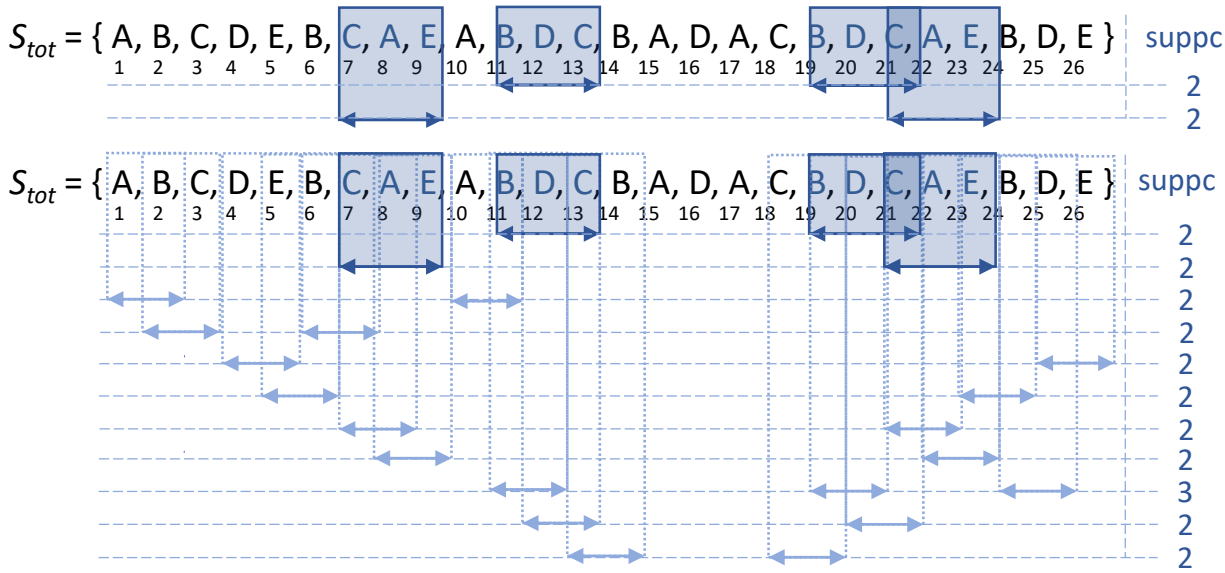


Abbildung 2.24: Oben: wiederkehrende Teilsequenzen (blau) in der Beispielsequenz. Unten: Die zusammengehörenden Teilsequenzen sind durch Intervalle auf gleicher Ebene markiert. Hier sind auch die zusätzlichen Teilsequenzen dargestellt (ohne farbliche Hervorhebung).

An den resultierenden Mustern lässt sich bereits erkennen, dass einige der $P(2,2)$-Muster (bspw. $\{B,D\}$ und $\{D,C\}$) auch in den längeren $P(2,3)$-Mustern (bspw. $\{B,D,C\}$) enthalten sind. Sie sind sozusagen Teil- bzw. Submuster vom übergeordneten (Super-) Muster. Solange die Teilmuster kein höheres Vorkommen aufweisen als ihr übergeordnetes Muster, ist durch sie eine gewisse Redundanz ohne zusätzliches Wissen in der Ergebnismenge gegeben, da sie ausschließlich dieselben Elemente beinhalten. In diesem Beispiel besitzt jedoch das Supermuster $\{B,D,C\}$ einen Support Count von 2, während das Submuster $\{B,D\}$ sogar dreimal vorkommt. Eine Sequenz des $\{B,D\}$-Musters ist demnach nicht in seinem übergeordneten Muster enthalten. Hier sind neben

der Redundanz zusätzliche Informationen gegeben. Zur Beschreibung der Verhältnisse zwischen Super- und Submuster werden die Begriffe *geschlossen* und *maximal* genutzt. Allgemein gilt in diesem Zusammenhang: Existiert zu einem Muster kein übergeordnetes Muster, so ist das Muster *maximal*. Die Menge aller maximalen Muster ist P_m. Existiert ein übergeordnetes Muster, jedoch mit geringerem Support Count, so ist das ursprüngliche Muster *geschlossen* (*closed*). Entsprechend ist P_c die Menge aller geschlossenen Muster. Wie in Abbildung 2.25 geschildert, gilt dabei:

$$\mathcal{P}_m \subseteq \mathcal{P}_c \subseteq \mathcal{P} \tag{2.61}$$

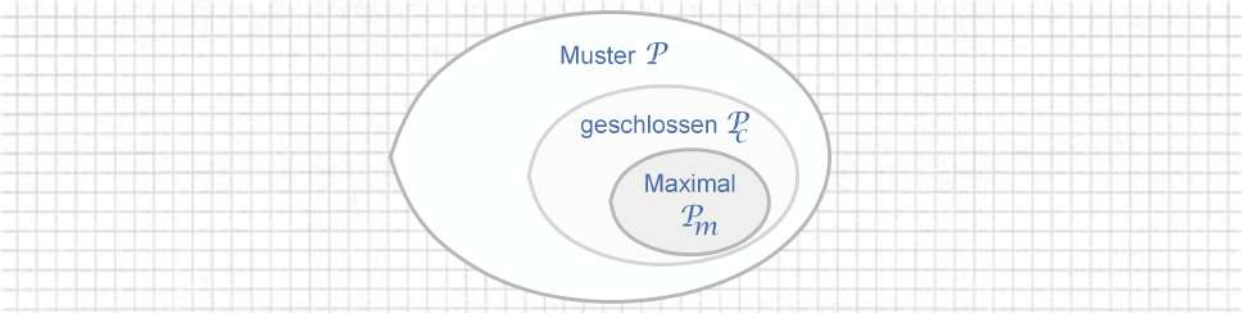


Abbildung 2.25: Geschlossene und maximale Muster

Bei der Mustererkennung ist man häufig jedoch nur an den geschlossenen und maximalen Mustern interessiert, da alle weiteren Muster kein zusätzliches Wissen liefern. Übertragen auf das Beispiel bedeutet das, dass sich die gesuchte Ergebnismenge zu $\mathcal{P}_c = \{\{B,D,C\},\{C,A,E\},\{B,D\},\{A,B\},\{B,C\},\{C,B\},\{E,B\},\{D,E\}\}$ verringert. Nicht mehr enthalten sind die Muster $\{D,C\}$, $\{C,A\}$ und $\{A,E\}$, da sie weder maximal noch geschlossen sind.

Zur Ermittlung von wiederkehrenden Subsequenzen bietet sich der **Apriori-Algorithmus** von Agrawal und Srikant (1994) an. Dieser ist ein Verfahren zum Erkennen von Mustern und Lernen von Assoziationsregeln in Transaktionsdatenbanken. Die Muster entsprechen den zuvor beschriebenen Sequenzmustern. Assoziationsregeln beschreiben Korrelationen zwischen Elementen innerhalb von Transaktionen. Letztere beinhaltet eine Menge an Transaktionen, welche wiederum aus *ungeordneten* Mengen an Elementen bestehen (vgl. Tabelle 2.5). Der Algorithmus selbst besteht aus zwei Teilen:

1. Ermittlung der Muster
2. Ableiten der Assoziationsregeln

Im Rahmen dieser Arbeit sind jedoch die Assoziationsregeln nicht von Interesse, sodass nur auf den ersten Teilschritt eingegangen wird. In diesem stützt sich der Algorithmus auf die Apriori-Eigenschaft, von der er auch seinen Namen hat.

$$\forall S_a,S_b : (S_a \subseteq S_b) \rightarrow suppc(S_a) \geq suppc(S_b) \tag{2.62}$$

Diese nicht-monotone Eigenschaft besagt, dass *alle nicht-leeren Submuster eines Musters einen mindestens genauso hohen Support Count* besitzen. Wird beispielsweise einer Teilsequenz, die zuvor nicht Bestandteil eines Musters war, ein Element angehängt, so kann diese Teilsequenz auch hinterher nicht zu einem Muster gehören. Auf diese Weise kann der Suchraum verkleinert werden, indem übergeordnete Sequenzen bereits verworfener Subsequenzen nicht weiter betrachtet werden müssen.

Der Algorithmus arbeitet iterativ. Jede Iteration besteht aus zwei Phasen. In der ersten, der *Join*-Phase, werden Kandidatensequenzen erzeugt, die in der zweiten, der *Prune*-Phase, wieder entfernt werden, falls sie den benötigten Support nicht erreichen, also (2.58) nicht erfüllt ist. Der in Algorithmus 1 gelistete Pseudocode beschreibt den Ablauf.

Algorithm 1 Apriori-Algorithmus nach Agrawal und Srikant (1994)

```
1: L_1 ← {häufige 1-itemsets}
2: k ← 2
3: for k = 2; L_{k-1} ≠ ∅; k++ do
4:     C_k ← {a ∪ {b} | a ∈ L_{k-1} ∧ b ∉ a} − {c |{s | s ⊆ c ∧ ||s|| = k − 1} ⊈ L_{k-1}}
5:     for all Transaktionen t ∈ T do
6:         C_t ← {c | c ∈ C_k ∧ c ⊆ t}
7:         for all Kandidaten c ∈ C_t do
8:             c.count++
9:         end for
10:    end for
11:    L_k ← {c | c ∈ C_k ∧ c.count ≥ suppc_min}
12: end for
13: return ⋃_k L_k
```

$$S_{tot} = \{\underset{1}{A}, \underset{2}{B}, \underset{3}{C}, \underset{4}{D}, \underset{5}{E}, \underset{6}{B}, \underset{7}{C}, \underset{8}{A}, \underset{9}{E}, \underset{10}{A}, \underset{11}{B}, \underset{12}{D}, \underset{13}{C}, \underset{14}{B}, \underset{15}{A}, \underset{16}{D}, \underset{17}{A}, \underset{18}{C}, \underset{19}{B}, \underset{20}{D}, \underset{21}{C}, \underset{22}{A}, \underset{23}{E}, \underset{24}{B}, \underset{25}{D}, \underset{26}{E}\}$$

Abbildung 2.26: Die ursprüngliche Beispielsequenz

Anhand der oben nochmals dargestellten Beispielsequenz (ursprünglich in Abbildung 2.23 enthalten) soll die Funktionsweise bei der Suche nach $\mathcal{P}_{2,2}$ dargestellt werden. In der ersten Iteration wird die Kandidatenmenge C_1 bestimmt. Die Kandidaten bestehen dabei jeweils nur aus einem Element und entsprechen den in der Sequenz enthaltenen unterschiedlichen Elementen. Dazu ist ein kompletter Daten-/Sequenzscan nötig, bei dem zugleich der Support Count der 1-elementigen Kandidaten ermittelt wird. Im nun folgenden Prune werden die nicht häufigen Kandidaten aus C_1 entfernt, was zur Kandidatenmenge L_1 führt (Abbildung 2.28, erste Zeile). In den folgenden Iterationen (restliche Zeilen) werden in der Join-Phase die Kandidatenmengen C_k anhand der jeweils vorangehenden Kandidatenmenge L_{k-1} ermittelt. Dabei werden jeweils die Kandidaten aus L_{k-1} miteinander verbunden, wenn sie die Bedingung zum Verbinden erfüllen (d.h. sie *joinable* sind). Dies ist der Fall, wenn jeweils die ersten $k-2$ Elemente zweier Itemsets (l_1 und l_2) übereinstimmen:

$$(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1]) \tag{2.63}$$

Der letzte Teil der Bedingung verhindert lediglich die Erzeugung von Duplikaten. Das neue Itemset sieht damit wie folgt aus:

$$\{l_1[1], l_1[2], \ldots, l_1[k-2], l_1[k-1], l_2[k-1]\} \tag{2.64}$$

Zur Erzeugung der Kandidatenmenge C_4 ($k = 4$) müssen demnach $k - 2 = 2$ der Itemsets übereinstimmen. Dieses trifft bspw. auf $\{A,B,C\}$ und $\{A,B,D\}$ zu. Nach der Bildungsvorschrift 2.65 wird das neue Itemset zu

$$\{l_1[1] = A, l_1[4-2] = B, l_1[4-1] = C, l_2[4-1] = D\} = \{A, B, C, D\}. \tag{2.65}$$

In der Prune-Phase werden wie in der ersten Iteration die Support Counts der generierten Kandidatensequenzen ermittelt. Dazu ist stets ein erneuter kompletter Scan der Ausgangsdaten notwendig. Die nicht häufigen Kandidaten werden entfernt, sodass auf diese Weise L_{k+1} entsteht. Der Algorithmus terminiert im Zeitschritt $k = k_{term}$, sobald C_{k+1} bzw. L_{k+1} eine leere Menge ist und dadurch im nächsten Schritt keine weiteren Kandidaten generiert werden können. In dem Beispiel besteht L_3 nur noch aus den Kandidaten $\{B,D,C\}$ und $\{C,A,E\}$. Diese können laut (2.63) nicht mehr miteinander verbunden werden, sodass C_4 und damit auch L_4 leer bleiben. Das Ergebnis der Suche sind damit alle Kandidaten der Mengen L_k mit $k \geq suppc_{min}$.

$$\mathcal{P} = \bigcup_{k=suppc_{min}}^{k_{term}} L_k \tag{2.66}$$

In diesem Beispiel ist Ergebnismenge demnach $\mathcal{P} = \mathcal{P}_{2,2} = \{\{B,D,C\},\{C,A,E\},\{B,D\},\{D,C\},\{C,A\},\{A,E\},\{A,B\},\{B,C\},\{C,B\},\{E,B\},\{D,E\}\}$. Der Apriori-Algorithmus liefert also die Menge aller Muster, nicht nur die der geschlossenen Muster $\mathcal{P}_c$.

Der zuvor vorgestellte Algorithmus ist in der Lage, häufige Subsequenzen zu finden. Voraussetzung dafür ist aber, dass die Sequenzen exakt gleich sind. Sind die Daten und die darin enthaltenen Muster verrauscht, d.h. sie sind nicht mehr exakt gleich, sondern ähneln sich nur noch zu einem gewissen Grad, so stößt der Apriori-Algorithmus schnell an seine Grenzen. In der Beispielsequenz könnten unter Berücksichtigung einer gewissen Fehlertoleranz weitere Muster erkannt werden. Wie in Abbildung 2.27 dargestellt, könnten bei der Tolerierung eines fehlerhaften Elements in der Sequenz, was einer Fehlertoleranz von 20 % entspräche, das $P(2,5)$-Muster $\{C,A,E,B,D\}$ identifiziert werden. So müsste hier lediglich das vierte Element ($S(10) = A$) in der ersten Teilsequenz ignoriert werden.

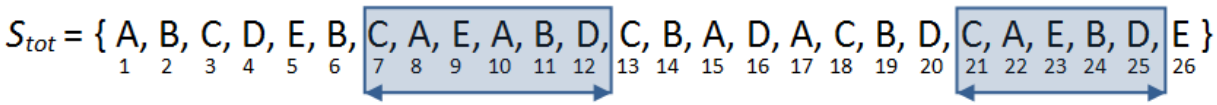
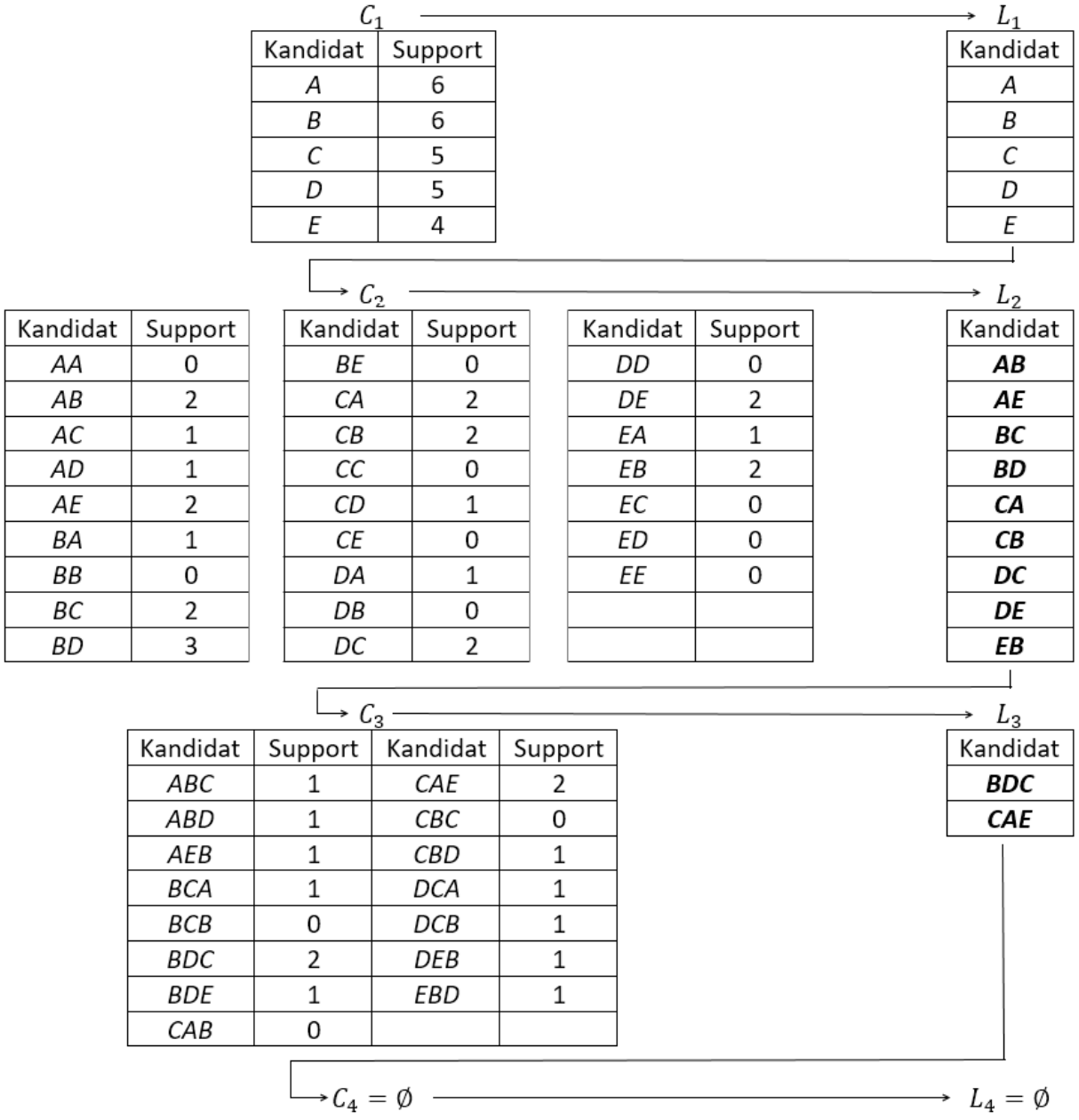


Abbildung 2.27: Nicht erkannte sich ähnelnde Teilsequenzen

In diesem Fall kann auf Algorithmen zurückgegriffen werden, mit deren Hilfe auch Muster bestehend aus ähnlichen Teilsequenzen erkannt werden können. Neben weiteren Algorithmen bietet sich zu diesem Zweck der **Baeza-Yates-Gonnet**- (oder **Bitap**- oder **Shift-or**-) Algorithmus (Baeza-Yates und Gonnet, 1992) an. Der ursprüngliche Algorithmus, ist nach diversen Erweiterungen und Optimierungen in der Lage, ähnliche Zeichenfolgen zu identifizieren. Die Ähnlichkeit wird dabei mit Hilfe der *Levenshtein-Distanz* bestimmt. Diese ist eine *Edit-Distanz*, die die Ähnlichkeit anhand der Anzahl an benötigten Änderungen (Ersetzen, Hinzufügen oder Löschen von Elementen) in der Ausgangssequenz misst, sodass sie der Vergleichssequenz entspricht. Dass der eigentliche Anwendungsfall dieses Algorithmus das String-Matching ist, stellt an dieser Stelle kein Problem dar, da Zeichenketten (*Strings*) prinzipiell nichts Anderes als Folgen von Zeichen bzw. (im allgemeinen Sinn) von Elementen sind. Der Algorithmus kann daher ebenfalls für den Vergleich von allgemeinen Elementsequenzen genutzt werden. Als Eingabe benötigt er die Sequenz sowie ein Maß für die maximal erlaubte Abweichung der Teilsequenzen. Da die Abweichung wie beschrieben durch die Levenshtein-Distanz berechnet wird, gibt der Input-Parameter k_m die erlaubte Anzahl an benötigten Änderungen (*mismatches*) an.

Die Ergebnismenge $\mathcal{P}$ enthält nach Terminierung alle Teilsequenzen, die die gegebenen Anforderungen hinsichtlich der Ähnlichkeit erfüllen. Ähnlich wie beim Apriori-Algorithmus sind dabei jedoch neben den geschlossenen bzw. maximalen Mustern auch auch sämtliche Subsequenzen die-

ser Muster. Da jedoch Letztere keinen Informationsgewinn bringen, ist eine Nachbearbeitung der Ergebnismenge erforderlich. In dieser werden alle Muster, die nicht zu den Geschlossenen bzw. Maximalen zählen, aus der Ergebnismenge herausgefiltert.



Kandidat	Support
A	6
B	6
C	5
D	5
E	4

Kandidat
A
B
C
D
E

Kandidat	Support
AA	0
AB	2
AC	1
AD	1
AE	2
BA	1
BB	0
BC	2
BD	3

Kandidat	Support
BE	0
CA	2
CB	2
CC	0
CD	1
CE	0
DA	1
DB	0
DC	2

Kandidat	Support
DD	0
DE	2
EA	1
EB	2
EC	0
ED	0
EE	0

Kandidat
AB
AE
BC
BD
CA
CB
DC
DE
EB

Kandidat	Support	Kandidat	Support
ABC	1	*CAE*	2
ABD	1	*CBC*	0
AEB	1	*CBD*	1
BCA	1	*DCA*	1
BCB	0	*DCB*	1
BDC	2	*DEB*	1
BDE	1	*EBD*	1
CAB	0		

Kandidat
BDC
CAE

Abbildung 2.28: Der Apriori-Algorithmus am Beispiel

2.4 Dynamische Programmierung

In vielen Anwendungsgebieten und Forschungsbereichen treten Optimierungsprobleme auf, die sich mit entsprechenden Optimierungsverfahren lösen lassen. Auch in dieser Arbeit wird für bspw. das Zuordnungsproblem zwischen Kameradetektionen und beobachteten Objekten oder die Abbildung einer Trajektorie auf die Andere eine optimale Lösung gesucht. Lässt sich das ursprüngliche Problem in einzelne kleine Teilprobleme zerlegen, kann zur effizienten algorithmischen Lösung

das Prinzip der *Dynamischen Programmierung* verwendet werden (Bellman, 1954; Bellman und Dreyfus, 2015). Dieses Prinzip basiert auf der Kombination der Teilproblemlösungen zu einer Gesamtlösung. Die Effizienz beruht auf der Zwischenspeicherung der Teilergebnisse, sodass diese nicht immer wieder neu berechnet werden müssen.

Am Beispiel des Viterbi-Algorithmus bedeutet das, dass das Problem der Bestimmung des Viterbi-Pfads in die Teilprobleme der Berechnung der einzelnen Zustandsübergangswahrscheinlichkeiten zerlegt wird. Diese Berechnungen würden durch die Rekursion (vgl. Gleichungen 2.28 und 2.29) immer wieder benötigt werden. Damit sie jedoch nicht bei jeder Iteration wieder neu durchgeführt werden müssen, werden deren Ergebnisse gespeichert. Hierzu wird eine Matrix-Struktur verwendet, auf die im späteren Verlauf zurückgegriffen wird, um die hinsichtlich der Gesamtwahrscheinlichkeit optimale Lösung zu erhalten.

2.5 Unterschiedliche Varianten der Datenverarbeitung

Bei der Verarbeitung von Daten kann zwischen verschiedenen Varianten unterschieden werden. Hierzu zählen die *zentrale* und *dezentrale* (oder auch *verteilte*) sowie die *sequenzielle* und *parallele* Datenverarbeitung. Hierbei beschreiben zentral/dezentral und sequenziell/parallel unterschiedliche Dinge. Die Begriffe „zentral“ und „dezentral“ geben an, *wo* die Daten verarbeitet werden. Mit „sequenziell“ bzw. „parallel“ wird hingegen ausgedrückt, *wie* die Daten verarbeitet werden. Abbildung 2.29 zeigt symbolhaft die Unterschiede für die jeweiligen Kombinationen aus Ort und Weise der Verarbeitung.

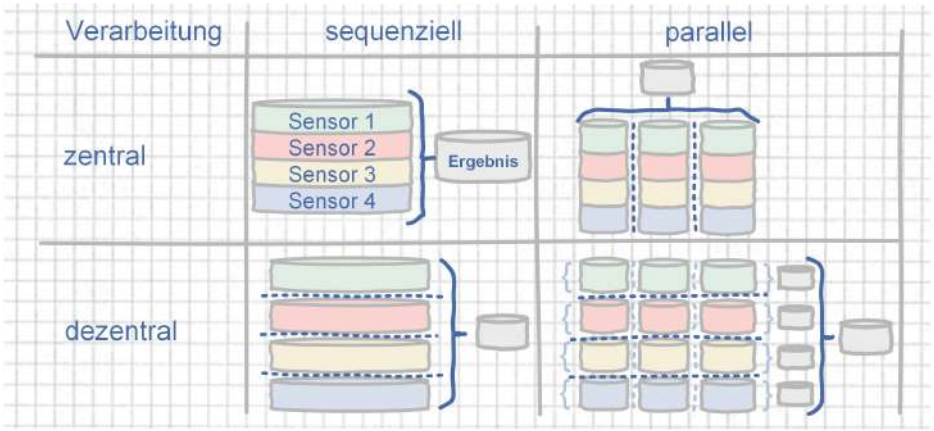


Abbildung 2.29: Die Matrix zeigt symbolhaft die unterschiedlichen Varianten der Datenverarbeitung: die Verarbeitung der jeweiligen Sensordaten kann als Kombination aus zentral/dezentral und sequentiell/parallel erfolgen.

Eine zentrale Verarbeitung (siehe erste Zeile der Matrix in Abbildung 2.29) setzt voraus, dass alle relevanten Daten der zentralen Instanz verfügbar sind bzw. dort gespeichert sind. Die Daten können dann gesamtheitlich prozessiert werden, um das entsprechende Ergebnis zu berechnen. Bei einer sequenziellen Verarbeitung werden die einzelnen Daten nacheinander abgearbeitet. Im Beispiel bedeutet dies, dass möglicherweise zuerst die Daten des ersten Sensors, dann die des zweiten Sensors, usw. prozessiert werden. Hierzu wird prinzipiell nur eine Recheneinheit (z.B. nur ein Rechenkern) benötigt. Stehen jedoch mehrere Rechenkerne (bzw. CPUs oder ganze Rechner) zur Verfügung, kann der Verarbeitungsaufwand auf die einzelnen Instanzen verteilt und parallel abgearbeitet werden. Hierbei findet allgemein während der Berechnung kein Austausch von Informationen zwischen den einzelnen parallelen Prozessen statt. In der Abbildung wird dies in der „zentral/parallel“-Zelle durch die vertikalen blauen gestrichelten Abgrenzungen symbolisiert. Dieses bedingt jedoch,

dass das zu lösende Problem an sich parallelisierbar ist. Eine parallele Verarbeitung ist prinzipiell schneller, obwohl der Aufwand für die Verteilung und das spätere Zusammenfügen der Daten bzw. Teilergebnisse zum Gesamtergebnis berücksichtigt werden muss.

Im Gegensatz dazu werden die Daten bei einer dezentralen Verarbeitung nicht in einer Instanz vorgehalten. Dort befinden sie sich auf den verteilten Instanzen (im Beispiel sind dies die Sensoren 1 bis 4), wo sie ebenfalls verarbeitet werden. Findet kein Informationsaustausch statt, so beruht die Berechnung der lokalen Ergebnisse ausschließlich auf Grundlage der lokal verfügbaren Daten (vgl. in der „dezentral-sequenziell"-Zelle die blau gestrichelten Abgrenzungen zwischen den jeweiligen Sensordaten). Wird ein globales Gesamtergebnis benötigt, so müssen die jeweiligen lokalen Zwischenergebnisse mit Hilfe von entsprechenden Algorithmen (bspw. durch *Tiny-Aggregation*) oder durch bspw. einen vorbestimmten (eventuell mit höherer Rechenleistung ausgestatteten) Sensor aggregiert werden. Je nach Ausstattung und Leistungsfähigkeit der verwendeten Sensoren, können auch hier die Daten parallel verarbeitet werden. Ansonsten wird die parallele Prozessierung hier bereits dadurch realisiert, dass die Sensoren unabhängig voneinander parallel arbeiten.

2.5.1 Zentrale und dezentrale Verarbeitung

Im Fokus dieser Arbeit steht die Unterscheidung eines zentralen und dezentralen Systemdesigns. Der Übergang zwischen beiden ist jedoch fließend. Dieses wird am Beispiel des Kamera-basierten Objekt-Trackings deutlich. Ist es einer Kamera bspw. möglich, das gesamte Beobachtungsgebiet zu überblicken, so ist es prinzipiell auch möglich, alle Objekte gleichzeitig zu verfolgen. Erlaubt die Kamera eine Datenverarbeitung vor Ort, so kann sie in diesem Fall als zentrale Instanz mit globalem Wissen fungieren. Wird das Gebiet jedoch ausgedehnt, so ist es eventuell nicht mehr möglich, sämtliche Objekte mit einer einzigen Kamera zu verfolgen. Es werden Weitere benötigt, sodass nun jede nur noch lokales Wissen besitzt und eine Berechnung der globalen Lösung auf einer Kamera allein nicht mehr möglich ist. Dieses kann entsprechend fortgeführt werden, bis jede Kamera, oder allgemein jeder Sensor, nur noch ein einziges Objekt verfolgt. Ein Beispiel hierfür wäre das GNSS-basierte Tracking, in dem jedes Objekt mit einem Empfänger ausgestattet wird. Die Dezentralität nimmt also zu, das lokal verfügbare Wissen ab. Um dennoch die kompletten Trajektorien der Objekte bestimmen zu können, müssen Informationen ausgetauscht werden. Es müssen z.B. die Objekte zwischen den Kameras übergeben werden, wenn jene die Sichtfelder wechseln. Die Lokalität der Sensordaten wird dadurch aufgeweicht, indem jeder Sensor nicht mehr nur seine eigenen Daten prozessiert.

2.5.2 Informationsaustausch

Der Informationsaustausch erfolgt durch Kommunikation der vernetzten Sensoren. Man spricht daher von *Sensornetzen*. Spielt zudem der Ortsbezug durch z.B. die Position der Sensoren eine Rolle, so sind dies ***Geo**sensornetze*. Zur Vernetzung können vorhandene Schnittstellen wie Kabel oder Funk (z.B. WLAN oder Bluetooth) verwendet werden. Je nach Anwendung und verwendeten Sensoren werden unterschiedliche Daten kommuniziert, bei denen auch der Grad der Vorverarbeitung variiert. Im Fall von Kameras, wie es beispielhaft in Abbildung 2.30 gezeigt ist, können es bspw. die Rohdaten in Form von Bildern, die Objektdetektionen als Resultat einer ersten Vorverarbeitung oder die Trajektoriepunkte bzw. Trajektoriesegmente als Zwischenergebnisse sein. Da jedoch aus Effizienzgründen die Kommunikation möglichst gering gehalten werden soll (Kommunikation ist um Faktor 1000 teurer als das Rechnen mit einer identischen Datenmenge (Duckham, 2012)), soll nach Möglichkeit der Austausch von Rohdaten vermieden werden.

Des Weiteren existieren verschiedene *Kommunikationsstrukturen* und *-algorithmen*, die ebenfalls den Austausch der Informationen optimieren. Zur Modellierung der Strukturen werden Graphen bestehend aus Knoten (Sensoren) und Kanten (Verbindungen/Kommunikationswege) genutzt. Eine realitätsnahe Struktur ist der *Unit Disk Graph* (*UDG*), bei dem die Knoten anhand einer einheitlichen Distanz (Sendereichweite) verbunden werden. Es können entsprechend nur die Sensoren untereinander kommunizieren, die sich in Reichweite befinden. Weitere Verfahren (u.a. *Relative Neighborhood Graph*) oder sogenannte *Overlay-Strukturen* in Form von Bäumen dienen dazu, Kanten künstlich auszudünnen, um so die Effizienz zu steigern und Aggregationsalgorithmen zu unterstützen. Ebenso können unterschiedliche Methoden zum Verteilen der Informationen genutzt werden. Abhängig davon, ob das gesamte Netz oder nur ein gezielter Knoten informiert werden soll, bieten sich *Flooding-* bzw. *Routing-Algorithmen* an. Auch hier existieren verschiedene Ausprägungen mit entsprechenden Vor- und Nachteilen. Eine detaillierte Erläuterung ist in Duckham (2012) gegeben. Aus diesen Gründen berücksichtigt ein effizienter Austausch von Information sowohl Kommunikationsstruktur und -algorithmus als auch die Art der zu übertragenden Daten.

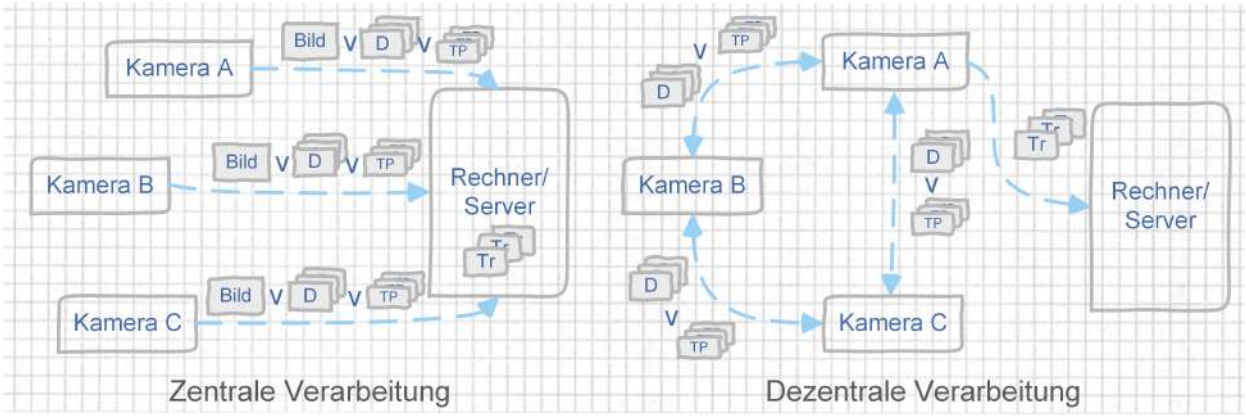


Abbildung 2.30: Gegenüberstellung der zentralen (links) und dezentralen Verarbeitung der Daten (rechts): Während bei der zentralen Variante sämtliche Daten in Form von Bildern, Detektionen (D) oder Trajektoriepunkten (TP) zum Zielcomputer (Rechner bzw. Server) übertragen werden, besteht die Übertragung bei der dezentralen Version lediglich aus (Zwischen-)Ergebnissen (Trajektoriepunkten).

3 Stand der Forschung und verwandte Arbeiten

In dieser Arbeit werden mit der Erfassung von Objekt-Trajektorien und der Erkennung von Bewegungsmustern in Trajektorien zwei unterschiedliche Problemstellungen behandelt. Im Zusammenhang mit der Erfassung von Trajektorien existieren verschiedene Herangehensweisen. Im Abschnitt 3.1 wird die relevante Literatur zu diesem Thema dargelegt. Dabei wird insbesondere auf die Ansätze zur Handhabung von fehlenden Objektdetektion durch bspw. Verdeckungen eingegangen. Des Weiteren werden auch kommerzielle Systemlösungen betrachtet, die bereits in der Praxis ihre Anwendung finden. Die relevante Literatur mit Bezug zur Erkennung von Bewegungsmustern in Trajektorien wird in Abschnitt 3.2 erläutert. Auch hier sind verschiedene Vorgehensweisen verfügbar, die davon abhängen, ob die gesuchten Muster von vornherein bekannt oder unbekannt sind. Die Erkenntnisse aus der Sichtung und Diskussion der relevanten Literatur werden bei der Herangehensweise zur Lösung der beiden Problemstellungen dieser Arbeit berücksichtigt.

3.1 Erfassung von Trajektorien

Im Folgenden werden abschnittsweise die existierenden relevanten Arbeiten aus der Literatur im Zusammenhang mit der Objektdetektion und -verfolgung sowie der Fusion unterschiedlicher Sensordaten mit ihren jeweiligen Eigenheiten beschrieben und diskutiert.

3.1.1 Objektdetektion

Die existierenden Ansätze zur Detektion von Objekten in Kamerabildern lassen sich auf Basis ihrer Vorgehensweise in unterschiedliche Kategorien unterteilen. Eine Gruppe von Ansätzen basiert auf der Annahme, dass sich bewegende Objekte für Veränderungen in der Szene und damit im Bild sorgen. Die veränderten Bildregionen werden als Vordergrund klassifiziert, die Übrigen, in denen keine Veränderungen vorhanden sind, werden entsprechend als Hintergrund klassifiziert. Es handelt sich hier demnach um Verfahren zur Vordergrund-Hintergrund-Trennung. Beispiele hierfür werden u.a. durch Zivkovic (2004),Piérard u. a. (2011) und Barnich u. a. (2006) gegeben. Zivkovic (2004) stellen mit der *Gaussian Mixture-based Background Subtraction*-Methode ein derartiges Verfahren vor. Dieser Ansatz arbeitet auf Pixelebene und nutzt zusammengesetzte Normalverteilungsdichten (*Gaussian mixture probabilitiy densities*), um die Hintergrundpixel des Bilds zu beschreiben. Ist es in einem Folgebild nicht möglich das gleiche Pixel auf gleiche Weise zu beschreiben, so wird dieses Pixel als Vordergrundpixel behandelt. Die zur Beschreibung verwendeten Parameter und Komponenten der zusammengesetzten Verteilung werden dabei automatisch angepasst. Das Hintergrundbild wird so ständig gelernt bzw. angepasst. In Piérard u. a. (2011) wird ebenfalls ein Verfahren zur Vordergrund-Extraktion verwendet, um die Silhouetten der sich bewegenden Objekte zu bestimmen. Mit Hilfe einer Klassifizierung auf Grundlage von zuvor berechneten Merkmalen werden daraufhin die Silhouetten von Personen ermittelt, die die resultierenden Detektionen im Bild darstellen. Barnich u. a. (2006) nutzen ebenfalls Lernverfahren, in diesem Fall eine Variante des *Random Forests*-Ansatz, zur Klassifizierung der achsparallelen Rechtecke, die die Vordergrundbildregionen umspannen. Auf diese Weise ermitteln sie die Detektionen von Personen.

Eine alternative Vorgehensweise bieten Detektoren, die Objekte in den Kamerabildern auf Basis ihrer Merkmale klassifizieren. In diesen Ansätzen wird ein lokales Suchfenster über das Bild

bewegt und nach den entsprechenden Merkmalen gesucht. Werden diese identifiziert, bedeutet das, dass sich in dieser Bildregion ein gesuchtes Objekt befindet. Diese Ansätze unterscheiden sich hauptsächlich anhand der verwendeten Merkmale bzw. Deskriptoren. Zudem können sie in offline und online trainierte Detektoren unterschieden werden. In den Arbeiten von Dalal und Triggs (2005) und Viola und Jones (2001b) werden offline trainierte Detektoren zur Klassifizierung von Objekten in Bildern beschrieben. In Dalal und Triggs (2005) werden mit Hilfe von *Support Vector Machines* (*SVM*) Merkmale aus dem *Histogram of Oriented Gradients* (*HOG*), einer Häufigkeitsverteilung der Ausrichtungen von akkumulierten und gewichteten Kantengradienten, extrahiert. Diese Merkmale werden zur Detektion der Objekte benutzt. Zur Erkennung von Personen werden die *HOG*-Merkmale mehrerer Teilregionen zu einem Personen-Deskriptor, repräsentiert durch einen Merkmalsvektor, zusammengefasst. Ein Suchfenster, das alle Teilregionen umschließt, wird über das gesamte Bild geschoben. Dabei werden unter Berücksichtigung unterschiedlicher Skalierungen entsprechende Merkmalsvektoren anhand der lokalen Bildinformationen ermittelt, die mit Hilfe einer SVM als Person oder Hintergrund klassifiziert werden. Viola und Jones (2001a) nutzen eine Detektor-Kaskade, die aus einer Abfolge von Klassifikatoren mit steigender Komplexität besteht. Weil in dieser Kaskade sehr wahrscheinliche Hintergrundregionen im Bild früh verworfen werden, kann mehr Aufwand in die Überprüfung vielversprechenderer Regionen fließen. Während die vorangehenden Verfahren versuchen, die Objekte anhand ganzheitlicher Merkmale zu erkennen, besteht das Vorgehen einer weiteren Gruppe von Ansätzen darin, einzelne Komponenten der Objekte zu detektieren, mit dem Ziel, diese im Folgenden anhand der bekannten Objektstruktur bestmöglich zusammenzusetzen. Bei der Erkennung von Personen handelt es sich bei den Komponenten bspw. um die einzelnen Körperteile. In Leibe u. a. (2008) und Felzenszwalb u. a. (2010) werden zwei solcher Ansätze beschrieben. Dort geben die detektierten Komponenten Hinweise auf mögliche Objektpostionen. Als Objektdetektion wird jeweils die Komposition der Teile verwendet, die eine maximale Übereinstimmung in Bezug auf die Objektposition liefert. Felzenszwalb u. a. (2010) nutzen dabei entsprechend trainierte *HOG*-Deskriptoren für jede einzelne Objektkomponente. Gleichzeitig erlauben sie Objektdeformationen, sodass Objekte auch dann erkannt werden können, wenn sie teilweise verdeckt sind oder sich in einer untypischen Pose befinden.

Während die offline trainierten Verfahren durch das abgeschlossene Training in Bezug auf u.a. die Art der Objekte, deren Gestalt und deren Erscheinung festgelegt sind, sind online trainierte Detektoren anpassungsfähig gegenüber sich verändernder Bedingungen. So sind sie in der Lage, Veränderungen bezogen auf das Aussehen der Objekte oder der Lichtverhältnisse zu berücksichtigen, was besonders im Fall von partiellen Verdeckungen vorteilhaft ist. Existierende Verfahren nutzen zu diesem Zweck unterschiedliche Lernverfahren. In Saffari u. a. (2009) wird ein auf *Random Forests* basierendes Verfahren vorgestellt, das während der Laufzeit das „Wachsen“ neuer Entscheidungsbäume ermöglicht. Breitenstein u. a. (2011) und Okuma u. a. (2004) verwenden das sogenannte *Boosting*-Prinzip, bei dem mehrere Klassifikatoren kombiniert werden. Letztere kombinieren mehrere *Partikelfilter* für das Tracking der Objekte mit dem *Adaboost*-Verfahren für die Generierung einer Vorschlagsverteilung.

Im Forschungsfeld *Deep Learning* werden u.a. *Convolutional Neuronal Networks* (*CNN*) zur Objektdetektion in Bildern herangezogen. In Ma u. a. (2015) wird in einer Hierarchie von Konvolutionsebenen zur Extraktion von Merkmalen genutzt, die zur Prädiktion von Objektpositionen durch Korrelationsfilter verwendet werden. In Wang u. a. (2015) hingegen wird ein Ansatz präsentiert, der ein Netzwerk zur Objektdetektion und ein Weiteres zur Erfassung von Unterscheidungsmerkmalen beinhaltet. Cao u. a. (2016) ermöglichen eine Ermittlung von 2D-Posen mehrerer Personen gleichzeitig. Dazu verwenden sie eine Netzwerkarchitektur, die die Positionen und Zusammenhänge einzelner Körperteile lernt. In einem Bottom-up-Prozess werden aus den Teilen die Personen generiert.

Wird die Szene mit mehreren Kameras aus verschiedenen Perspektiven beobachtet, können Methoden verwendet werden, die auf der Auswertung der Belegung des Beobachtungsraums (häufig als als Ebene betrachtet) basieren. Die Arbeit von Fleuret u. a. (2008) diskretisiert die Grundfläche mit Hilfe eines regelmäßigen Rasters. Für die einzelnen Zellen wird die Wahrscheinlichkeit ermittelt, dass sich auf dieser eine Person befindet (*Probability Occupancy Map*), indem die Bilder aus verschiedenen Perspektiven ausgewertet werden. In einem ähnlichen Vorgehen berechnen Yang u. a. (2003) die visuellen Hüllen der Objekte durch Verschneidung ihrer Detektionen aus mehreren Kameraperspektiven. Durch die Projektion auf eine Ebene entstehen hieraus Polygone, die nachfolgend anhand verschiedener Kriterien gefiltert werden. Die verbleibenden Polygone stellen die gesuchten Objektpositionen dar. Iwase und Saito (2004) nutzen ein Multi-Kamera-System, in dem jede Kamera die Objekte für sich verfolgt. Bei Problemen durch bspw. Verdeckungen werden die Informationen der jeweils anderen Kameras hinzugezogen und für eine Positionsbestimmung gemittelt.

3.1.2 Objekt-Tracking

In der Literatur existieren verschiedene Ansätze zur Verfolgung von Objekten in Sequenzen von Kamerabildern. Während einige von diesen Verfahren auf einer Modellierung anhand von Zustandsmodellen sowie Filter-Techniken basieren, versuchen andere entweder durch die Verknüpfung von *Tracklets*, also in der Regel kurzen Trajektoriesegmenten, oder durch eine globale Optimierung die Trajektorien der Objekte zu ermitteln.

Im Zusammenhang mit der Verwendung von Zustandsmodellen und Filter-Ansätzen gibt es eine Gruppe von Arbeiten, die auf einer Variante der rekursiven Bayesschen Schätzung basieren. Häufig wird dabei das Kalman-Filter genutzt Zhao und Nevatia (2004); Luber u. a. (2010); Shu u. a. (2012); Yoon u. a. (2015), wobei die Objektposition und -geschwindigkeit als Zustandsvariablen modelliert werden. In weiteren Arbeiten werden hingegen Partikel-Filter verwendet (Okuma u. a., 2004; Breitenstein u. a., 2011; Zhao und Nevatia, 2004; Tran u. a., 2011; Zhang u. a., 2015), die sich dann eignen, wenn das Rauschen als nicht normalverteilt angenommen werden muss. Der Ansatz von Klinger u. a. (2017) stützt sich auf eine Modellierung als Dynamisches Bayessches Netz, in dem sie den Zustandsvektor und die Position eines Objekts als Unbekannte behandeln. Xie und Evans (1990) und Ardo u. a. (2007) nutzen *Hidden Markov Modelle* (*HMM*) zur Beschreibung von Objektbewegungen über einer diskretisierten Umgebung. Die Ermittlung der wahrscheinlichsten Sequenz an Zuständen, anhand der die Objekt-Trajektorien generiert werden, erfolgt dort mit Hilfe des Viterbi-Algorithmus (Forney, 1973). Dieser spielt auch in der Arbeit von Martinerie (1997) eine Rolle, in denen er für das Tracking von Objektbewegungen in einem Sensornetzwerk eingesetzt wird.

Ein andere Herangehensweise zeigen die Ansätze auf, die auf der Assoziation von *Tracklets* basieren. Hierbei wird das Problem in viele kleinere Teilprobleme unterteilt, indem anfangs nur die Detektionen zu kurzen Trajektorien miteinander verbunden werden, bei denen eine relativ eindeutige Zuordnung möglich ist. Im Folgenden werden auf Grundlage der neuen Informationen, die durch die zusammenhängenden Trajektoriesegmente ermittelt werden können, durch immer weitere Verknüpfung längere Tracklets erzeugt. Unterschiedliche Ansätze bauen auf diesem Prinzip auf. Ein Beispiel hierfür ist das *Multi-Hypothesen-Tracking* (*MHT*) nach Reid (1979), bei dem simultan verschiedene Lösungshypothesen ermittelt und weiterverfolgt werden, bis sich am Ende die Wahrscheinlichste herauskristallisiert. Hinz und Schmidt (2017) und Kim u. a. (2015) erweitern diese Methode um die Möglichkeit, mehrere Objekte gleichzeitig zu verfolgen. Zu diesem Zweck kombinieren Hinz und Schmidt (2017) das MHT mit einem Bewegungsmodell auf Basis einer Kalman-Filterung. Kim u. a. (2015) hingegen verwenden *Deep CNNs* zum Lernen des Erschei-

nungsbilds der Objekte für jede Hypothese zur Laufzeit. Ähnliche Vorgehensweisen sind in Ess u. a. (2008) und Schindler u. a. (2010) beschrieben. Dort werden verschiedene Trajektorien-Hypothesen mitgeführt, die aus Kalman-Filterungen hervorgehen. In Henschel u. a. (2016) werden Kameradetektionen und existierende Tracklets als Merkmale verschiedener Levels angesehen. Diese werden mit Hilfe von Clustering-Verfahren und der Linearen Programmierung zeitlich und räumlich zu einer global-optimalen Zuordnung assoziiert. In einem weiteren Ansatz von Henschel u. a. (2015) wird ein Objekt-Tracking-Ansatz vorgestellt, der Tracklets auf Grundlage einer *hierarchischen Assoziation* verknüpft. Auf diese Weise werden in jeder Hierarchieebene weitere Informationen bzgl. der Tracklets gesammelt, die eine verlässlichere Verknüpfung ermöglichen. Dazu organisieren sie die Tracklets in einer Baumstruktur, die weniger Knoten und Kanten als die häufig verwendeten Flussnetzwerke enthält und eine Bestimmung der optimalen Lösung mit linearen Aufwand zulässt. Die Arbeit von Wang u. a. (2017) verfolgt ein sehr ähnliches Herangehen. Während in Yang und Nevatia (2014) *Conditional Random Fields* genutzt werden, um zu ermitteln, welche Tracklets zusammenpassen, wird in Perera u. a. (2006) die *Ungarische Methode* nach Kuhn (2012) für eine optimale Zuordnung herangezogen.

Ein nahezu fließender Übergang von den Tracklet-basierten Ansätzen zu den sogenannten *Offline-Tracking*-Verfahren ergibt sich dann, wenn die gesamte Sequenz aus Kamerabildern inklusiver der Objektdetektionen bereits verfügbar ist. In diesem Fall kann das Tracking als *globales Optimierungsproblem*, genauer gesagt als *Minimum Cost Flow*-Problem, formuliert werden. In entsprechenden *Min Cost Flow-Netzwerken* (*MCFN*) werden die Kameradetektionen durch zeitlich-geordnete Knoten und deren Korrespondenzen durch gerichtete Kanten repräsentiert. In diesem Kontext präsentieren Leal-Taixé u. a. (2011) eine Kombination aus einer neuartigen graphischen Modellierung und dem *Social Force Model* nach Helbing und Molnar (1995), ohne die Komplexität des Problems zu vergrößern, sodass die optimale Lösung mit Hilfe der Linearen Programmierung bestimmt werden kann. Letztere wird auch in einem ähnlichen Ansatz in Berclaz u. a. (2009) benutzt. In Zhang u. a. (2008) wird das MCFN durch eine explizite Modellierung von Verdeckungen erweitert, um auf diese Weise Situationen zu behandeln, in denen die Objekte für einige Zeit nicht sichtbar sind. Alternativ zur Linearen Programmierung kann auch die Dynamische Programmierung zur Ermittlung des wahrscheinlichsten Pfads in einem MCFN verwendet werden (Pirsiavash u. a., 2011; Wang und Fowlkes, 2015).

Auch die erst in den letzten Jahren aufgekommenen Deep Learning-Verfahren finden in dem Tracking-Kontext Anwendung. In Milan u. a. (2017) werden *Recurrent Neuronal Networks* für die Verfolgung von Objekten vorgeschlagen. Andere Arbeiten nutzen *Siamese CNN* für die Zuordnung von Tracklets (Wang u. a., 2016; Leal-Taixé u. a., 2016).

Bei der gleichzeitigen Verfolgung mehrerer Objekte entsteht bei der Zuweisung der Objektdetektionen zu den Objekten ein Zuordnungsproblem, auch *Data Association Problem* genannt. Für dessen Lösung sind diverse Herangehensweisen in der Literatur beschrieben. Dabei wird die Zuordnung häufig als Optimierungsproblem aufgefasst, bei dem die Zielfunktion die Ähnlichkeit der Detektionen zu den Objekten bzw. deren Trajektorien beschreibt. Typische Merkmale zur Beschreibung der Ähnlichkeit sind die räumliche Distanz zwischen Detektionen und Trajektorien sowie das Aussehen der Objekte. Letzteres umfasst Farben und Texturen und kann durch u.a. Histogramme (Okuma u. a., 2004; Schindler u. a., 2010; Andriluka u. a., 2010) oder mit Hilfe von Klassifikatoren (Xiang u. a., 2015; Breitenstein u. a., 2011; Wang u. a., 2017) repräsentiert werden. Für die Berechnung der Ähnlichkeit dieser Informationen bieten sich u.a. mit der *Histogrammverschneidung*, der *Korrelation* oder der *Bhattacharrya-Distanz* verschiedene Distanzmaße an. Jedoch wird durch Histogramme lediglich die numerische Verteilung dargestellt. Die räumliche Verteilung bzw. Anordnung wird dadurch nicht berücksichtigt, sodass bei Objekten, deren äußere Erscheinungen die gleichen Farben aufweisen, Probleme entstehen können. Zudem erweisen sich Veränderungen hinsichtlich des

Aussehen während der Beobachtung, z.B. durch schwankende Lichtverhältnisse, als ebenfalls problematisch. Hier sind Lernverfahren im Vorteil, die eine adaptive Anpassung des gelernten Modells ermöglichen. Die Arbeit von Wang u. a. (2017) beschäftigt sich mit dem Lernen eines Distanzmaßes auf Grundlage des Aussehens und der Bewegungen von Objekten. Dabei wird simultan die Gewichtung beider Merkmale gelernt, um mit Situationen mit Verdeckungen und fehlenden Detektionen umgehen zu können. Xiang u. a. (2015) verwenden das *Reinforcement Learning* zur Ermittlung von Richtlinien für die Modellierung des Problems als *Markov-Entscheidungsproblem*, was dem Lernen einer Ähnlichkeitsfunktion entspricht. Während Yang und Jia (2016) Hidden Markov Modelle zur Voraussage von Änderungen des Erscheinungsbilds der Objekte nutzen, wird Letzteres in Breitenstein u. a. (2011) mittels *Boosting*- bzw. in Saffari u. a. (2009) durch *Random Forests*-basierte Methoden gelernt. Werden Online-Tracking-Verfahren genutzt, müssen die Kameradetektionen zur Laufzeit den Objekten zugeordnet werden. Dieses erfordert die Lösung des Zuordnungsproblems in jedem Zeitschritt. Hierzu können im Allgemeinen Greedy-Algorithmen verwendet werden (Breitenstein u. a., 2011; Pirsiavash u. a., 2011; Shu u. a., 2012). Alternativ existieren Ansätze, die entweder auf *Sampling* (Benfold und Reid, 2011; Brau u. a., 2013) oder auf der *Ungarische Methode* basieren (Wu und Nevatia, 2007; Solera u. a., 2015).

3.1.3 Fusion heterogener Detektionen

Tragen in einem System unterschiedliche Sensoren zu einer gemeinsamen Lösung eines Problems bei, müssen deren Messungen fusioniert werden. Die Herausforderung bei der sogenannten Sensor-Fusion liegt in darin, die heterogenen Informationen verschiedener Sensortypen zu integrieren.

Eine sehr verbreitete Kombination aus Sensoren besteht aus einem GNSS-Empfänger und einer inertialen Messeinheiten[1] (IMU). Bei bspw. der *GPS/INS*-Methode werden die GPS-Messungen zur Kalibrierung eines inertialen Navigationssystems (*INS*) verwendet (Wendel, 2011). Die Integration der Sensordaten ist ein nicht-lineares Filterproblem, zu dessen Lösung häufig Varianten des *Kalman-Filters* (u.a. Wendel, 2011; Hyun u. a., 2009; Grewal u. a., 2007; Crassidis, 2006; Smith und Singh, 2006) oder *Partikel-Filter* (u.a. Wendel, 2011; Smith und Singh, 2006) genutzt werden.

Werden Kameras in ein System integriert, sind prinzipiell zwei unterschiedliche Varianten zu unterscheiden. Bei der Ersten werden die zu beobachtenden Objekte mit sämtlichen Sensoren ausgestattet, sodass sie sich mit den Objekten mitbewegen. Die Kameras beobachten dann in der Regel die Umgebung des Objekts. Häufige Anwendungsbeispiele sind die Positionierung von Fahrzeugen bzw. Robotern oder die Erkennung und Verfolgung von anderen Verkehrsteilnehmern mit Hilfe unterschiedlicher verbauter Sensoren, wie u.a. GPS, IMU, Laserscannern, Kameras und Radgeschwindigkeitssensoren (z.B. Agrawal und Konolige, 2006; Bellotto und Hu, 2009; Cho u. a., 2014). Ansätze aus diesem Kontext stützen sich ebenfalls auf verschiedene Filter-Techniken. Während bspw. in Agrawal und Konolige (2006) und Cho u. a. (2014) *Extendend Kalman-Filter* genutzt werden, werden in z.B. Suhr u. a. (2017) *Partikel-Filter* verwendet.

In der zweiten Variante beobachten die Kameras die Szene von außen, in der sich die relevanten Objekte bewegen. Auf diese Weise erfassen sie die Bewegungen der Objekte direkt. In diesem Zusammenhang werden Objekte bspw. anhand einer Kombination aus Kameras und Mikrophonen Spors u. a. (2001) bzw. Laserscannern Yin u. a. (2014) verfolgt. Rao u. a. (1993) nutzen mehrere Kameras und optische Barrieren zur Verfolgung von Robotern. In sämtlichen Fällen werden Kalman-Filter zur Fusion der Daten benutzt.

[1] Inertiale Messeinheit (*inertial measurement unit, IMU*): Kombination von Sensoren zur Messung von Bescheinigungen und Drehraten.

Auch in Multi-Sensor-Systemen, die aus mehreren gleichartigen Sensoren bestehen, müssen Daten fusioniert werden. In Zhou und Kimber (2006) wird z.B. ein Multi-Kamera-System dafür verwendet, die Trajektorien sich bewegender Objekte zu berechnen. In Martinerie (1997) hingegen wird ein verteiltes Sensornetzwerk aus Sonaren mit dem gleichen Zweck genutzt. Beide Ansätze nutzen Hidden Markov Modelle zur Modellierung. In Martinerie (1997) liefert dabei der Viterbi-Algorithmus die wahrscheinlichste Sequenz aus Zuständen, anhand der die Objekt-Trajektorien generiert werden.

3.1.4 Kommerzielle Systeme

Eine stetig wachsende Zahl an Unternehmen hat das Objekt-Tracking für sich entdeckt, da es auch in kommerzieller Hinsicht, insbesondere im Bereich der Sportanalyse, ein attraktiver Markt ist. Diese Unternehmen bzw. Start-ups verstehen sich in der Regel als Dienstleister. Sie erfassen und analysieren die Bewegungen der Sportler und stellen die Daten dem Auftraggeber, welche die Sportler selbst, die Vereine oder auch die Medien sein können, zur Verfügung. Seit noch nicht allzu langer Zeit werden Objekt-Tracking-Lösungen auch von den Sportverbänden zur Begleitung von Wettkämpfen genutzt. Dort dienen sie zur Überprüfungen von Regelentscheidungen. Ein prominentes Beispiel hierfür ist das Hawk-Eye-System[1], das in einigen Sportarten, wie bspw. beim Tennis, Cricket oder Fußball, zur Verfolgung des Balls genutzt wird. Das System basiert auf der Verwendung von mindestens vier (für höhere Genauigkeiten eher sechs oder mehr) Hochgeschwindigkeitskameras, die das Geschehen aus unterschiedlichen Perspektiven betrachten. Bei der im Fußball eingeführten Torlinien-Technik sind es sieben Kameras, die zur millimetergenauen Positionsbestimmung des Balls in Tornähe beitragen. Weitere Beispiele sind u.a. die Unternehmen *STATS*[2], die ihren ebenfalls bekannten Mitbewerber *Prozone* erworben haben, *deltatre AG*[3] (ehemals *Impire AG* bzw. *Cairos*), *Catapult.*[4] und CHYRONHEGO[5], die teilweise verschiedene Tracking-Lösungen anbieten. Jedoch betreiben sie ihr Kerngeschäft hauptsächlich mit Multi-Kamera-Systemen, wie bspw. in Abbildung 3.1a dargestellt. Weitere kleinere Unternehmen bzw. Start-ups wie *TRACKTICS*, *PLAYERTEK*[6] (gehört zu *Catapult.*) oder *One Sports*[7] vertreiben GPS-basierte Tracking-Lösungen, bei denen die Sportler mit entsprechenden GPS-Empfängern ausgerüstet werden (Abbildung 3.1b). Die auf diese Weise aufgezeichneten Daten werden im Nachhinein ausgelesen und mit der ebenfalls zur Verfügung gestellten Software ausgewertet.

3.1.5 Diskussion und Fazit

Zusammenfassend wird deutlich, dass das reine GPS-Tracking (abgesehen vom DGPS) generell unter der typischen Ungenauigkeit leidet. Anwendungsszenarien mit einem Bedarf an hoher absoluter Genauigkeit scheiden hier aus. Videobasierte Verfahren mit nur einer oder verhältnismäßig wenigen Kameras hingegen leiden oft unter einer geringeren Verlässlichkeit. Diese wird durch Probleme bei der Objektdetektion und -verfolgung hervorgerufen, die wiederum hauptsächlich durch Objektverdeckungen verursacht werden. Eine Möglichkeit der Verringerung dieser Probleme bietet die Nutzung komplexerer Systeme, wie bspw. Multi-Kamera- oder funkbasierter Systeme. Allerdings erweisen sich dort die Installation und die Kosten für Anschaffung und Betrieb in vielen

[1] https://www.hawkeyeinnovations.com/
[2] https://www.stats.com/
[3] http://www.bundesliga-datenbank.de/de/home/
[4] http://www.catapultsports.com/
[5] http://chyronhego.com/
[6] https://www.playertek.com/eu/
[7] http://www.onesports.com.br/en/index.php

(a) *(b)*

Abbildung 3.1: Zwei Beispiele für kommerzielle Tracking-Systeme: a) Beim SportVU-System[1] der Firma STATS werden mehrere Kameras zum Tracken der Objekte verwendet. b) Die Lösung der Firma TRACKTICS[2] besteht aus GPS-Loggern, mit denen die Sportler ausgerüstet werden.

Anwendungsfällen als problematisch. Die Leistungsfähigkeit der kommerziellen Systeme lässt sich nur schlecht bewerten, da Detailinformationen über die Qualität der Ergebnisse nur sehr wage bekannt gemacht werden. Es kann sich daher nur auf die Informationen gestützt werden, die der Entwickler bzw. Betreiber veröffentlicht. Allerdings müssen für eine Verwendung solcher Systeme die Rahmenbedingungen sowohl in technischer als auch in finanzieller Sicht gegeben sein. Das Hawk-Eye-System muss bspw. in einer gewissen Höhe (unter dem Stadiondach) installiert werden und kostet ungefähr 8000 Euro pro Spiel (Penders, 2014). In vielen Anwendungsfällen können diese Anforderungen nicht erfüllt werden. Auf ein bereits existierendes System kann daher für diese Arbeit nicht zurückgegriffen werden, sodass eine neue Lösung entwickelt werden muss. Dabei müssen die in Abschnitt 1.2.1 beschriebenen Teilprobleme gelöst werden.

Objektdetektion

Die vorgestellten Detektionsverfahren besitzen unterschiedliche Anforderungen an das Anwendungsszenario und entsprechende Vorzüge bzw. Nachteile. Die Methoden zur Vordergrund-Hintergrund-Trennung setzen voraus, dass die Kameras statisch sind. D.h. die internen und externen Parameter dürfen sich nicht ändern. Das Gleiche gilt in einem gewissen Maß auch für die Lichtverhältnisse. Zudem dürfen sich die beobachteten Objekte nicht zu statisch verhalten, da sie bei den Verfahren, die den Hintergrund lernen, im Verlauf als dieser klassifiziert werden. Des Weiteren entstehen durch dynamische Hintergrundobjekte, z.B. sich im Wind bewegende Pflanzen, Schatten, usw., zusätzliche Fehldetektionen. In Szenarien mit Verdeckungen sinkt die Qualität der Ergebnisse, da dort Bildregionen mit Veränderungen von sich teilweise verdeckenden Objekten zu einer einzigen Detektion zusammengefasst werden. Diese Mischdetektionen sorgen beim Objekt-Tracking für Probleme, da sie sich nicht ohne Weiteres eindeutig einem Objekt zuordnen lassen.

Merkmalsbasierte Detektoren leiden prinzipiell darunter, dass die Ergebnisse nicht immer zuverlässig und geometrisch korrekt sind. Zudem müssen die Objekte ganz sichtbar sein, da sonst die generierten Merkmalsvektoren verfälscht werden, was zu einer Fehlklassifikation führen kann. Daher sind Szenen mit erheblichen Verdeckungen problematisch. Um dieses Problem zu umgehen, werden die Verfahren vorgeschlagen, die auf der Detektion von Objektkomponenten basieren. Die Komponenten sind in der Regel aber deutlich kleiner als das zugehörige Objekt selbst. Sie

[1] Bildquelle: STATS, https://statsweb-wpengine.netdna-ssl.com/wp-content/uploads/2016/08/SportVU-Setup-1.jpg, Zugriff am 27.11.2017.

[2] Bildquelle: TRACKTICS, https://tracktics.com/wp-content/uploads/TRACKTICS-G%C3%7BCrtel-und-Performance-Tracker.png, Zugriff am 27.11.2017.

sind daher, insbesondere bei größeren Distanzen zwischen Objekten und Kamera, schwierig zu erkennen. Für beide Varianten der merkmalsbasierten Detektion besteht zudem das Problem, dass Hintergrundobjekte mit ähnlicher Gestalt ebenfalls detektiert werden, wodurch die Anzahl an Fehldetektionen steigt. Diese Probleme werden in Ansätzen, wie bspw. in Klinger u. a. (2017) und Hoiem u. a. (2008) beschrieben, adressiert. In Klinger u. a. (2017) werden die Objektdetektionen des HOG-Personen-Detektors auf Basis von 3D-Informationen, die aus den Detektionen der Köpfe und Füße der einzelnen Personen ermittelt werden, und Vorwissen über die Szene validiert. Letzteres wird mit Hilfe von Lernverfahren gewonnen und umfasst Informationen über häufig besuchte Orten der Szene. Auf diese Weise können sowohl die Anzahl der Fehldetektionen als auch die geometrischen Fehler reduziert werden.

Die Deep Learning-Verfahren liefern z.T. eindrucksvolle Ergebnisse. Jedoch sind für das Training der großen Netze ebenfalls große Mengen an Trainingsdaten erforderlich, die nur selten zur Verfügung stehen oder nur aufwendig generiert werden können.

Objekt-Tracking

Die Tracklet-basierten Ansätze erfordern für die Generierung der Tracklets die Daten innerhalb eines gewissen Zeitfensters. Dessen Breite bestimmt zum einen die Anzahl der benötigten Bilder zum anderen den zeitlichen Versatz, mit dem die Objekt-Trajektorien erfasst werden können. Um ein nahezu Echtzeit-Tracking zu erreichen, muss das Fenster klein gewählt werden. Dies hat jedoch zur Folge, dass die Tracklets entsprechend kurz sind und weniger Informationen von diesen abgeleitet werden können.

Offline-Tracking-Verfahren benötigen die Detektionen aus den Bildern der gesamten Sequenz. Eine Anwendung in Szenarien, die ein Echtzeit-Tracking erfordern, ist damit nicht möglich.

Die vorgestellten Deep Learning-Verfahren besitzen auch bei der Verfolgung der Objekte ein großes Potential. Allerdings benötigen sie eine große Menge an Trainingsdaten, deren Bereitstellung in der Regel jedoch mit großem Aufwand verbunden ist.

Die Online-Verfahren besitzen je nach angewendeter Zuordnungsstrategie ebenfalls Probleme. So garantieren Verfahren, die ein Sampling nutzen nicht die global optimale Lösung. Die Ungarische Methode wiederum hat Probleme bei fehlenden oder gemischten Detektionen, die mehrere Objekte gleichzeitig beinhalten. Bei der Zuordnung der Detektionen zu den Objekten wird häufig eine Histogramm-Repräsentation der Merkmale (z.B. Farben oder Texturen) genutzt. Dabei entstehen bei ähnlich aussehenden Objekten Probleme, da die geometrische Verteilung der Merkmale nicht berücksichtigt wird.

Fazit

Im Zusammenhang mit der Objektdetektion sind sämtliche der untersuchten Methoden anfällig gegenüber Verdeckungen von Objekten. Im Rahmen dieser Arbeit wird angenommen, dass in den möglichen Anwendungsszenarien statische Kameras verwendet werden. Zudem sind die Anforderung an die geometrische Genauigkeit der Kameradetektionen hoch. Aus diesen Gründen werden in dem Ansatz zur Erfassung von Trajektorien, der im folgenden Kapitel beschriebenen wird, Vordergrund-Hintergrund-Trennungsverfahren benutzt. Im späteren Verlauf dieser Arbeit werden unterschiedliche Systemdesigns diskutiert. Um die Möglichkeit aufrechtzuerhalten, das System als Online-Tracking-System betreiben zu können, ist die Verwendung von Offline-Verfahren nicht möglich, da diese Detektionen über den gesamten Beobachtungszeitraum benötigen. Ähnliches gilt für die Tracklet-basierten Methoden. Da zudem bei der Erfassung mit der GNSS-Lokalisierung eine zusätzliche Datenquelle integriert werden soll und die Unsicherheiten beider Positionsmessungen berücksichtigt werden sollen, wird für diese Arbeit ein Ansatz auf Basis von Zustandsmodellen als geeignet betrachtet.

3.2 Mustererkennung in Trajektorien

Da unter dem Begriff „Bewegungsmuster“ ganz unterschiedliche Dinge verstanden werden können, soll zur Definition und zur Vereinheitlichung der Benennung von Mustern die Arbeit von Dodge u. a. (2008) herangezogen werden. In dieser wird eine Klassifikation und Beschreibung verschiedener Bewegungsmuster gegeben. Die Autoren gehen dabei von der Annahme aus, dass die beobachteten Objekte auf sich bewegende Punktobjekte (*moving point objects*) reduziert werden. Die Größe der Objekte und sich gegebenenfalls bewegende Komponenten bzw. Körperteile werden daher nicht berücksichtigt. Letzteres gilt auch in dieser Arbeit, in der die Bewegungsmuster auf Grundlage der Objekt-Trajektorien ermittelt werden, die lediglich aus einer zeitlichen Abfolge von Punkten bzw. Positionen im Raum bestehen. In ihrer Taxonomie (siehe Abbildung 3.2) teilen sie die Muster hierarchisch in unterschiedliche Kategorien ein. Sie unterscheiden dabei zwischen generischen Mustern (*generic patterns*) und Verhaltensmustern (*behavioral patterns*). In der ersten Kategorie sind *primitive* und *zusammengesetzte* Mustern (weiter nach *räumlich*, *raum-zeitlich* und *zeitlich* differenziert) zu finden. Als typische Beispiele sind hier die konstanten Bewegungen (*constancy*), d.h. die Bewegungsparameter ändern sich für eine signifikante Zeit nicht, oder die zeitlich periodischen Bewegungen (*periodicity*), d.h. die Bewegungen wiederholen sich in einem gewissen Turnus (z.B. stündlich, täglich, usw.), zu nennen. In der Verhaltensmusterkategorie befinden sich Muster, die einem speziellen Verhalten der Objekte entsprechen. Zum Beispiel beschreibt ein *Flock* eine Gruppe von Objekten, die sich gemeinsam fortbewegen. Dies könnte bspw. ein Vogelschwarm sein. Ein weiterer typischer Vertreter dieser Kategorie ist das *Leadership*-Muster, bei dem mehrere Objekte einem führenden Objekt (*Leader*) folgen.

Zur Erkennung dieser Muster in den Trajektorien der beobachteten Objekte existiert eine Vielzahl verschiedener Ansätze, die sich auf die oben beschriebene Klassifikation stützen. Einige dieser Ansätze werden im Folgenden beschrieben. In bspw. Laube u. a. (2008), Gudmundsson u. a. (2007) oder Benkert u. a. (2008) werden Verfahren zu Identifikation von *Flock*-Mustern vorgestellt. Der *FLAGS*-Algorithmus von Laube u. a. (2008) ist für eine dezentrale Verarbeitung konzipiert und kann somit auch bei verteilten Sensorsystemen angewendet werden. Gudmundsson u. a. (2007) und Benkert u. a. (2008) stellen bei der Detektion der *Flocks* eine effiziente Verarbeitung in der Vordergrund. Gleichzeitig beschreiben sie, wie auch *Leadership*-, *Convergence*- (die Trajektorien treffen sich zeitlich unabhängig) und *Encounter*-Muster (wie Convergence, aber dieses Mal zur selben Zeit) in den Bewegungsdaten von beobachteten Objekten erkannt werden können. In Wang u. a. (2006) wird zwar eine abweichende Benennung des gesuchten Musters verwendet, jedoch handelt es sich nach deren Definition ebenfalls um das Flock-Muster. Hier werden zwei Detektionsmethoden beschrieben, die sich Sequenzanalyseverfahren zunutze machen. Ein dem *Flock* sehr ähnliches Muster ist das *Moving Cluster*-Muster. Hierbei wird jedoch die Bedingung der räumlichen Nähe, die die Objekte besitzen müssen, aufgelockert, sodass sie sich auch zeitweise weiter auseinander bewegen dürfen, solange sie später wieder zusammen kommen. Einen Ansatz zur Identifikation dieser Art Muster ist in Li u. a. (2010a) beschrieben. Ein weiterer Algorithmus, der eine Detektion von sowohl Gruppen- als auch individuellen Mustern ermöglicht, wird von Laube u. a. (2005) vorgestellt. Dort werden diskretisierte, relative Bewegungen (*relative motions*, *REMO*) der beobachteten Objekte analysiert. Dazu wird eine Matrix-Repräsentation, die REMO-Matrix, erstellt, die aus einer Zeile für jedes Objekt und einer Spalte für jeden Zeitschritt besteht. Diese Matrix wird daraufhin mit Hilfe von regulären Ausdrücken, die um eine räumliche Komponente erweitert sind, nach Mustern gemäß der gegebenen Taxonomie durchsucht.

Im Rahmen der vorliegenden Arbeit sollen jedoch sich wiederholende, a-priori unbekannte Bewegungen in den Trajektorien identifiziert werden, von denen sich entsprechende Bewegungsmuster ableiten lassen. Diese Art Muster lässt sich am ehesten in die Kategorie „*Repetition*“ (unter *generic compound patterns*) des in Abbildung 3.2 dargestellten Schemas einordnen. Unter dieser Kategorie

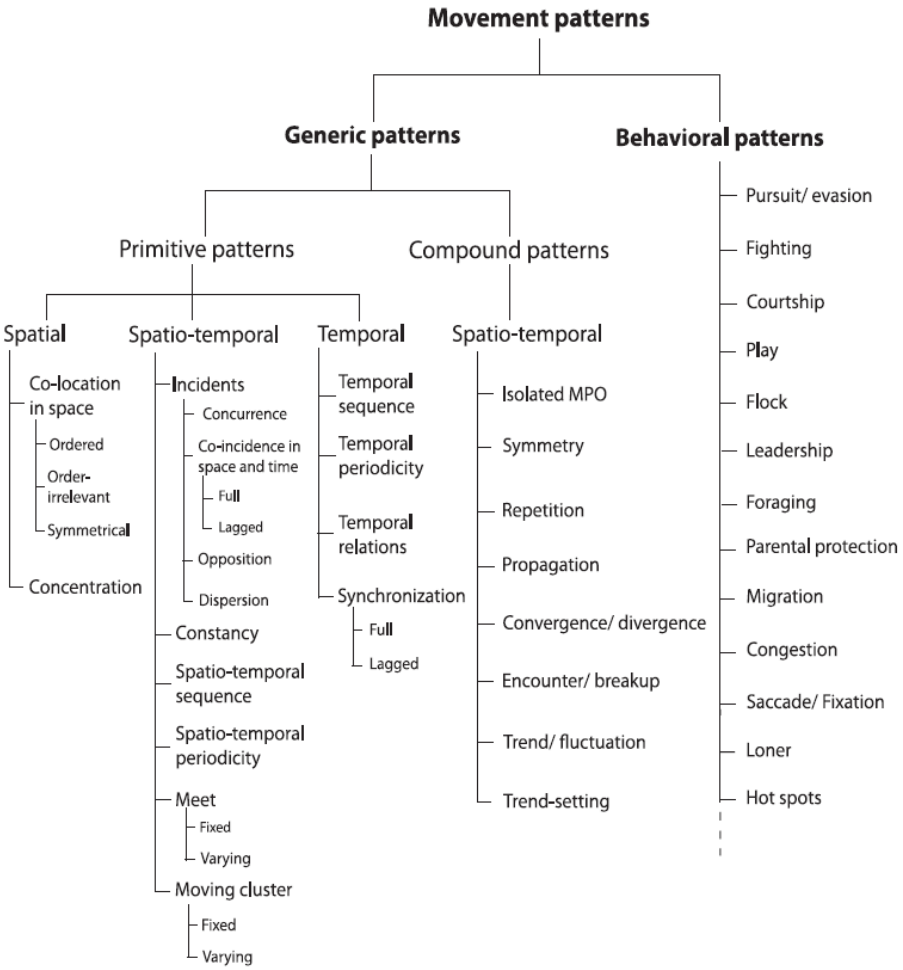


Abbildung 3.2: Eine Klassifikation von Bewegungsmustern nach Dodge u. a. (2008)

fassen Dodge u. a. (2008) die Muster zusammen, die sich durch die Wiederholung von primitiven Mustern auszeichnen. Da die Primitive allerdings wiederum von vornherein bekannt sind, passt diese Einordnung hier nur zum Teil. Existierende Ansätze zur Erkennung dieser repetitiven bzw. primitiven Muster (z.B. Laube u. a., 2005) können daher in diesem Zusammenhang nicht angewendet werden.

3.2.1 Erkennung von wiederkehrenden unbekannten Mustern

Auch für die Erkennung von a-priori unbekannten Mustern sind Ansätze in der Literatur verfügbar. Diese stammen aus ganz unterschiedlichen Forschungsgebieten, z.B. dem Bereich der Verkehrsanalyse, wo auch Fußgänger eine Rolle spielen, der Analyse von Tierbewegungen oder der Sportanalyse. Zusätzlich lassen sie sich hinsichtlich ihrer Herangehensweise unterscheiden.

Lernverfahren

Eine Reihe von Ansätzen basiert auf der Anwendung von Clustering-Algorithmen in Kombination mit Distanzmaßen, wie bspw. Edit-Distanzen, Dynamic Time Warping oder Longest Common Subsequence, um Ähnlichkeiten zwischen Trajektorien zu identifizieren und so typische Objektbewegungen abzuleiten. Yuan u. a. (2017) geben eine umfangreiche Übersicht über die gebräuchlichen Verfahren. Beispiele für diese Vorgehensweise werden in u.a. de Vries und van Someren (2010),

Chen u. a. (2011) und Vafa (2014) geliefert. Die Arbeit von Morris und Trivedi (2009) enthält einen Vergleich von unterschiedlichen Distanzmaßen und Clustering-Algorithmen angewendet auf verschiedene Datensätze. Das Ergebnis dieser Studie ist, dass die Wahl der Algorithmen und Distanzmaße sehr von den Trajektorien abhängt. Nanni und Pedreschi (2006) nutzen eine erweiterte Version des OPTICS-Algorithmus zum Clustering von Trajektorien. Lee u. a. (2008) präsentieren ihr Framework „TraClass“, dass automatisch relevante Merkmale für das Clustering durch eine hierarchische Partitionierung der Trajektorien bestimmt. In einer ihrer weiterführenden Arbeiten schlagen Lee u. a. (2015) ein Framework vor, in dem Trajektorien mit verschiedener zeitlicher Dichte vereint und nach Mustern durchsucht werden können. Ein weiteres Clustering-Framework wird von Hung u. a. (2015) vorgestellt. Dieses nutzt ein Ähnlichkeitsmaß, das mit Unterbrechungen in den Trajektorien umgehen kann. Pelekis u. a. (2009) untersuchen den Effekt von Unsicherheiten beim Trajektorien-Clustering, indem sie die Trajektorien unter Verwendung eines Algorithmus zur Berechnung einer Zentroid-Trajektorie und des Fuzzy-C-means-Algorithmus clustern. Eine thematisch verwandte Arbeit transformiert die Trajektorien in Sequenzen aus Klassensymbolen (Dodge u. a., 2012). Diese symbolische Repräsentation wird dann verwendet, um die Sequenzen anhand einer normalisierten, gewichteten Edit-Distanz als Distanzmaß zu vergleichen. In (Tsai u. a., 2011) ist ein Algorithmus beschrieben, der eine Detektion von Mustern in Bezug auf Objektgruppen mit dem gleichen Bewegungsverhalten ermöglicht. Die Autoren benutzen hierfür einen Suchalgorithmus, der lokale Bewegungsmuster detektiert, die im Nachhinein mittels eines Ähnlichkeitsmaßes geclustert werden, um Gruppenbeziehungen zu identifizieren. Um gegebenenfalls vorhandene semantische Informationen, die mit den Trajektorien verknüpft sind, nicht zu verlieren, schlagen Liu u. a. (2013) mit TODMIS ein allgemeines Framework zur Detektion von Trajektorien-Cluster vor. Dabei werden die Rohdaten in Form von Trajektorien mit den zusätzlichen Informationen kombiniert und mit verschiedenen Ähnlichkeitsmaßen analysiert.

Im Kontext der Fußballanalyse existieren ebenfalls einige Ansätze zur Mustererkennung. In Gudmundsson und Wolle (2014) wird eine umfassende Toolbox entwickelt, die einige Werkzeuge zur Analyse der Spieler-Trajektorien und der gespielten Pässe bereitstellt. Zur Untersuchung der Bewegungen wird dort nach Clustern in Trajektoriesegmenten gesucht, die sich wiederholende Bewegungen beinhalten. Um diese Cluster zu finden, werden die von Buchin u. a. (2011) vorgestellten unterschiedlichen Techniken zur Segmentierung angewendet. Im gleichen Kontext sind zudem weitere alleinstehende Ansätze verfügbar, die auf die Extraktion oder Klassifikation von (taktischen) Bewegungsmustern abzielen. Niu u. a. (2012) und Zhu u. a. (2007) kategorisieren Angriffe nach ihrer Startposition und einem a-priori definierten Schema. Li und Chellappa (2010) wenden ein gelerntes „Spatio-Temporal Driving Force Model“ an, um Gruppenmuster zu charakterisieren. Ein von Kim u. a. (2011) entwickeltes Framework nutzt ein Modell von Merkmalen und deren morphologische Eigenschaften, um Taktiken im Spiel zu untersuchen. Grunz u. a. (2012) benutzen eine hierarchische Architektur von künstlichen neuronalen Netzen zur Identifikation von taktischen Mustern. Zur Extraktion von Ballbewegungsmustern, welche innerhalb von Passsequenzen auftreten können, schlagen Mutschler u. a. (2011) ein schrittweises Suchverfahren vor, das unterschiedliche Ähnlichkeitsmaße nutzt, um die Ball-Trajektorie zu vergleichen, und dabei eine Invarianz gegenüber Translation, Rotation und Skalierung berücksichtigt.

Sequenzanalyseverfahren

Des Weiteren gibt es eine Gruppe von Herangehensweisen, die nach periodischen Mustern in eindimensionalen symbolischen Sequenzen suchen. Deren größte Herausforderung besteht darin, die 2D-Bewegungsdaten in 1D-Sequenzdaten zu transformieren. Zu diesem Zweck generieren sie Sequenzen aus entweder rechteckigen (Cao u. a., 2005), häufig besuchten (Giannotti u. a., 2007) oder vordefinierten (Mamoulis u. a., 2004) Regionen, die von den Trajektorien besucht werden. Ein sehr ähnliches Vorgehen wird auch in Cao u. a. (2007) beschrieben. Dort werden die Trajektorien zu-

erst partitioniert und daraufhin in den resultierenden Trajektorie-Abschnitten nach periodischen Mustern von besuchten Regionen gesucht. Bei diesen Region handelt es sich um entweder vordefinierte oder auf Basis von Stopps aus den Bewegungsdaten extrahierte Plätze. Analog zu dieser Herangehensweise identifizieren Li u. a. (2010b) häufig besuchte ROIs (*regions of interest*) und suchen nach periodischem Verhalten in Bezug auf die Besuche. Monreale u. a. (2009) nutzen identifizierte Sequenzen besuchter Plätze, um Vorhersagen auf die nächsten Plätze zu treffen. In den genannten Arbeiten werden gleichzeitig die Dimension der Daten und auch die große Zahl an Trajektoriepunkten zu bedeutsameren Aggregationen verringert. Die Muster werden schließlich mit Sequenzanalyseverfahren extrahiert.

Im Kontext der Fußballanalyse sind auch hier relevante Arbeiten zu finden. Hirano und Tsumoto (2004) beschäftigen sich zwar nicht mit Bewegungsmustern, beschreiben aber eine Möglichkeit, Passsequenzmuster zu bestimmen. Dafür setzen sie eine multi-skalige Abgleichungsmethode, die auf dem Vergleich von Konturen basiert. In Haaren u. a. (2015) wird eine automatische Erkennung von Offensivstrategien mit Hilfe eines induktiven logischen Programmierungsansatzes präsentiert. Bei der Pass-Analyse wird ebenfalls nach häufigen Passsequenzen gesucht. Auch dort werden Verfahren zur Mustererkennung genutzt. Gudmundsson und Wolle (2014) nutzen zur Detektion von wiederkehrenden Passsequenzen einen erstellten Suffixbaums (*suffix tree*). Dessen Verzweigungen werden durchlaufen, um die unterschiedlichen Passmuster zu extrahieren.

3.2.2 Diskussion und Fazit

Zusammenfassend ist festzustellen, dass viele Ansätze zur Extraktion von Mustern in Bewegungsdaten existieren. Jedoch sind diese größtenteils nicht auf das im Rahmen dieser Arbeit behandelte Problem anwendbar. Auf der einen Seite kann nicht nach vordefinierten Mustern gesucht werden, da die hier relevanten Muster a-priori unbekannt sind. Auf der anderen Seite nutzen die Methoden, die nach unbekannten Mustern suchen, häufig entweder die gesamten Trajektorien, was sich bei langen Beobachtungszeiträumen und dadurch langen Trajektorien als schwierig erweist, oder sie arbeiten auf Trajektorie-Segmenten und setzen daher eine Segmentierung als Vorverarbeitung voraus. Hieraus ergibt sich jedoch das Problem, an welchen Stellen die Trajektorien zerteilt werden. Eine Segmentierung rein auf Basis von Trajektorie-Parametern (Anzahl der Punkte, Länge der Segmente, Geschwindigkeiten, usw.) birgt die Gefahr, relevante Segmente zu zerschneiden und somit die Identifikation von Mustern in diesen Bereichen unmöglich zu machen. Die Frage ist daher, wie die relevanten Bereiche in den Trajektorien erkannt werden können. In vielen Fällen ist dafür entsprechendes Kontextwissen nötig. Die vorgestellten Lösungen, die auf der Analyse von Sequenzen basieren, passen deutlich besser zu der gegebenen Problemstellung dieser Arbeit, da dort das Segmentierungsproblem umgangen wird. Jedoch bleibt bei diesen Ansätzen das Problem, wie die Trajektorien in Sequenzen transformiert werden. Die Bestimmung von räumlichen Regionen, die die Sequenzelemente darstellen, ist nicht in allen Anwendungsszenarien anwendbar. Insbesondere in Szenarien, in denen das Beobachtungsgebiet im Verhältnis zu der Größe der zu erwartenden Plätze klein ist und ein uneingeschränktes Bewegungsverhalten der Objekte vorliegt, gestaltet sich die Ermittlung von sinnvollen Regionen schwierig.

In der Diskussion der relevanten Mustererkennungsansätze ist deutlich geworden, dass nur die Ansätze zu der Problemstellung dieser Arbeit passen, die nach wiederkehrenden unbekannten Mustern suchen. Dazu kommen sowohl die Clustering- als auch die sequenzbasierten Methoden in Frage. Da beide Herangehensweisen ein großes Potenzial aufweisen, wird in dieser Arbeit ein Prozess entwickelt, der beide Ansätze anbietet. Hierfür sind im Folgenden die Probleme im Zusammenhang mit der Segmentierung der Trajektorien bzw. der Transformation der Trajektorien in durchsuchbare Sequenzen zu lösen.

4 Erfassung von Trajektorien - GPS-unterstütztes Kamera-Tracking

In diesem Kapitel wird die Entwicklung eines Objekt-Tracking-Systems beschrieben, das zur Erfassung von Trajektorien verwendet wird, die die Grundlage für eine Bewegungsmustererkennung darstellen. Um eine vielfältige Verwendbarkeit zu gewährleisten, werden hierbei ausschließlich Low-cost-Sensoren verwendet. Um dennoch Trajektorien erfassen zu können, die die Ansprüche hinsichtlich Genauigkeit und Vollständigkeit von möglichst vielen Anwendungen erfüllen, wird eine Kombination aus GNSS- und Kamera-Tracking genutzt. Das Setup der Sensoren für das hier beschriebene Verfahren kann wie folgt beschrieben werden: Mit Hilfe einer oder mehrerer statischer, kalibrierter Kameras wird der gesamte Bewegungsraum der zu beobachtenden Objekte überwacht. Zusätzlich werden diese Objekte mit low-cost GNSS-Empfängern ausgestattet. Eine eigenständige Lösungen auf Basis einer der verwendeten Technologien scheidet auf Grund der folgenden Probleme aus. Das GNSS-Tracking liefert verlässliche Trajektorien, deren Genauigkeit jedoch im Bereich der typischen GNSS-Genauigkeit von ungefähr 5 - 20 m liegt (siehe Abschnitt 2.2.1). Dieses ist jedoch für gewisse Anwendungen nicht ausreichend. Das videobasierte Tracking hingegen ermöglicht deutlich höhere Genauigkeiten (siehe Abschnitt 2.2.2). Allerdings wird dort die Verlässlichkeit durch bspw. Objektverdeckungen in der Weise beeinflusst, dass dort die Ergebnisse schlechter werden, je mehr Objekte beobachtet werden. Da es sich je nach Anwendung um eine große Zahl an Objekten mit entsprechend vielen Verdeckungen handeln kann, ist die Verwendung eines rein videobasierten Ansatzes (abgesehen von einem teuren Mehrkamerasystem) problematisch.

4.1 Überblick über den Lösungsansatz

Für einen Ansatz zur Erfassung von Trajektorien, der eine Kombination aus GNSS- und videobasierten Tracking darstellt, müssen die folgenden Teilprobleme gelöst werden:

- Erfassung und Lokalisierung der Objekte in den Kamerabildern (Detektion)
- Zuordnung der einzelnen Detektionen zu den Objekt-Trajektorien (Verfolgung)
- Fusion unterschiedlicher Sensordaten mit ihren jeweiligen Eigenheiten

Die ersten beiden sind dabei die standardmäßigen Problemstellungen eines Tracking-Verfahrens. Das dritte und zusätzliche Problem der Fusion der unterschiedlichen Sensordaten entsteht jedoch durch dieses kombinierte Verfahren. Der hier beschriebene Ansatz zur Lösung dieser Probleme greift teils auf bereits existierende Verfahren zurück, teils werden neue Modellierungen entwickelt.

Das erste Teilproblem der Detektion der Objekte in den Kamerabildern wird mit Hilfe eines bereits existierenden Verfahrens angegangen. Auf Grund der statischen Installation der Kameras bieten sich hier u.a. Verfahren zur Vordergrund-Hintergrund-Trennung an. Diese finden durch die *Gaussian Mixture-based Background Subtraction*-Methode, welche in Abschnitt 4.2.2 näher beschrieben wird, auch ihre Anwendung.

Die übrigen Teilprobleme erfordern, speziell im Hinblick auf das Verfolgen mehrerer Objekte, ggf. fehlender Detektionen und der Fusion der unterschiedlichen Messungen, eine neuartige Modellierung. Einen Ansatz zur Lösung dieser Teilprobleme stellt die Modellierung als *Dynamisches Bayessches Netz* dar, in dem der Zustand in diskreten Zeitschritten voranschreitet. Je nach Modellierung

(in dieser Arbeit werden zwei Varianten beschrieben) repräsentieren die Zustände (*hidden states*) die nicht direkt-beobachtbaren Zuordnungen zwischen Video- und GPS-Beobachtungen bzw. die Objektposition als Fusionierungsergebnis aus beiden Messungen. Da die Zustände nicht direkt beobachtbar sind, sondern nur anhand der Messungen bzw. der Beobachtungen der Sensoren (*observations*) geschätzt werden können, besitzt ein solches Modell die Form eines *Hidden Markov Models* (*HMM*, vgl. Abschnitt 2.2.5). Da dieses ein lineares Netzwerk ohne Zyklen ist, erlaubt die Anwendung des bekannten *Viterbi-Algorithmus* (vgl. Abschnitt 2.2.6) eine exakte und effiziente Bestimmung der wahrscheinlichsten Sequenz an Zuständen, die zur Generierung der Objekt-Trajektorien notwendig ist. Die Eignung dieses Algorithmus im Zusammenhang mit dem Objekt-Tracking ist bereits in Martinerie (1997); Zen u. a. (2004); Xie und Evans (1990); Ardo u. a. (2007) gezeigt worden. Jedoch hat dort jeweils ein HMM zu Grunde gelegen, dass sich im Vergleich zu dieser Arbeit und der Problematik der Datenfusion unterscheidet.

In dem in Abbildung 4.1 dargestellten Schema wird der gesamte Prozess zur Datenerfassung veranschaulicht. Dieser besteht aus den aufeinander aufbauenden Phasen „Sensoren und Eingangsdaten“, „Vorverarbeitung“ und „Fusion der heterogenen Daten“, die je in einem gleichnamigen Abschnitt dieses Kapitel erläutert werden. Zusätzlich werden in zwei weiteren Abschnitten die Laufzeit des Algorithmus (4.5) und das Systemdesign (4.6) erläutert.

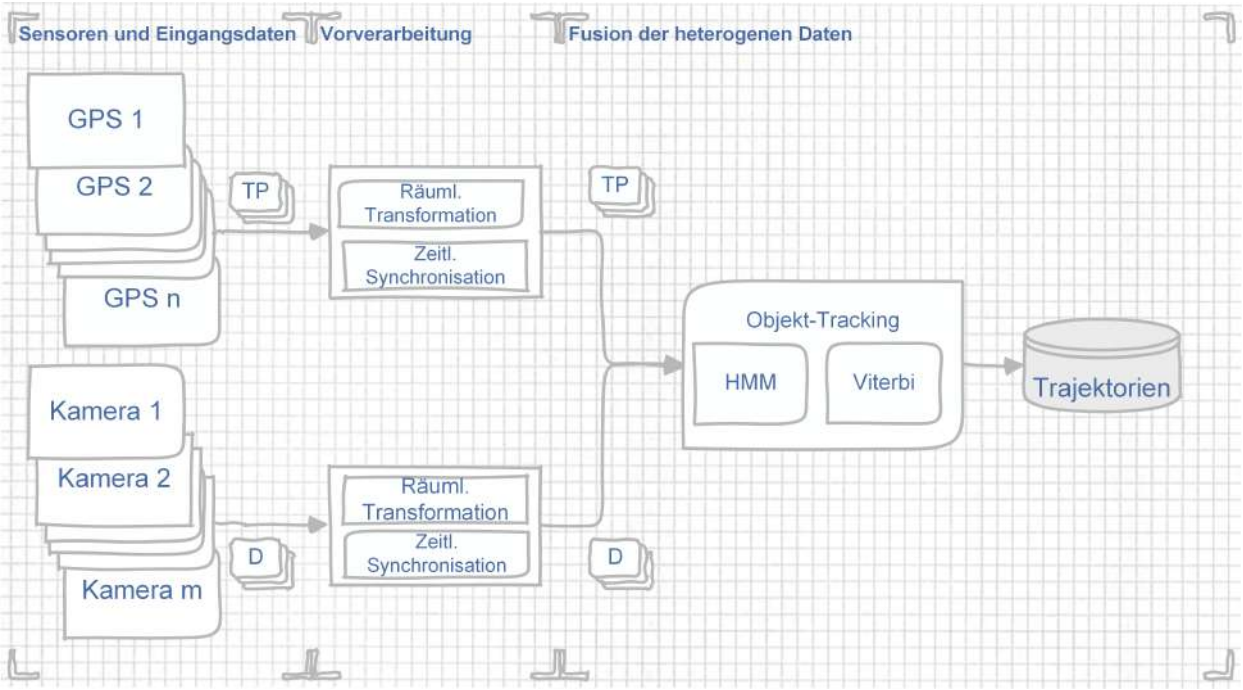


Abbildung 4.1: Übersicht über das Verfahren zur Erfassung der Trajektorien mit Hilfe vom GPS-unterstützten Kamera-Tracking (TP: Trajektoriepunkte, D: Detektionen der Kameras).

4.2 Sensoren und Eingangsdaten

Die in dem zu entwickelnden System verwendeten Sensoren besitzen jeweils ihre Eigenheiten und liefern unterschiedliche Daten. Sowohl die Sensoren als auch die Daten, die die Eingangsdaten des Systems sind, werden in den folgenden Abschnitten beschrieben.

4.2.1 GPS-Daten

Die verwendeten GPS-Geräte liefern abhängig von ihren Spezifikationen 3D-Positionsdaten mit einer Rate von 1 bis 10 Hz. In dynamischen Szenen mit schnellen Bewegungen und häufigen Richtungsänderungen, wie sie in verschiedenen Anwendungsszenarien auftauchen, kann eine Abtastrate von 1 Hz oder niedriger bereits zu gering sein, um die Bewegungen mit allen Details erfassen zu können (vgl. Abschnitt 2.1). In diesen Fällen werden höhere Raten, bspw. 5 Hz, benötigt, um sinnvolle Ergebnisse mit diesem Ansatz zu ermöglichen. Die Trajektoriepunkte (TP), die von den Geräten zu jedem Zeitpunkt bereitgestellt werden, besitzen ihre grundlegende Form (vgl. Gleichung 2.3) und sind durch die Objektkennung $objId$ erweitert. Diese ist beim GPS-Tracking bereits bekannt.

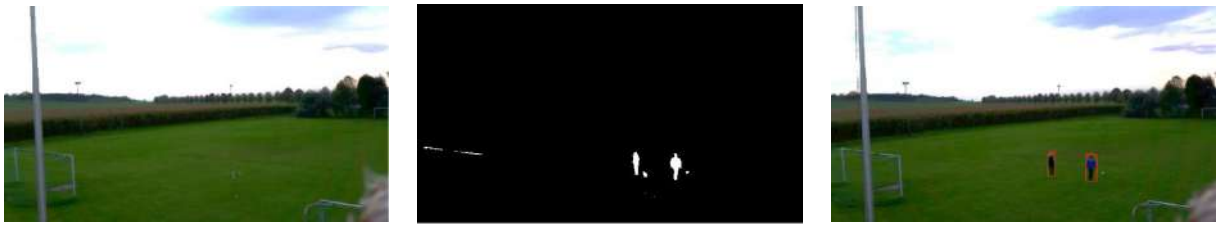
$$TP_i = (x,\, y,\, z,\, t,\, objId)_i. \tag{4.1}$$

Die Ungenauigkeit der Positionskomponente ist abhängig von der Umgebung und vom verwendeten GPS-Empfänger. Angaben hierzu findet man in den jeweiligen Geräte-Spezifikationen. Die Modellierung dieser Ungenauigkeit erfolgt unter der Annahme, dass der Fehler für jede Messung identisch normalverteilt ist, durch die GPS-Standardabweichung σ_{GPS}.

4.2.2 Kameradaten

Zusätzlich zu den GNSS-Empfängern werden in diesem Setup Kameras zur Ermittlung von 2D-Objektpositionen verwendet, da angenommen wird, dass die Objekte sich in der Ebene bewegen. Zur Korrektur der Verzeichnungen der Bilder, wie sie bei Objektiv-Kameras auftreten, und zur Umrechnung der Bild- in Weltkoordinaten werden die Kameras manuell mit Hilfe korrespondierender Punktpaare in Bild und Welt kalibriert (siehe Abschnitt 2.2.2). Die Punktpaare werden manuell identifiziert. Hier bieten sich eindeutige Objekte bzw. Merkmale an. Da es sich um statische Kameras handelt, bei denen sich im Verlauf weder die extrinsischen noch intrinsischen Parameter ändern, muss dieses nur einmal zu Beginn durchgeführt werden. Die Daten der kalibrierten Kameras, im Folgenden einfachheitshalber als *Detektionen* (D) bezeichnet, werden mit der gleichen Frequenz bereitgestellt, in der auch die GPS-Daten verfügbar sind.

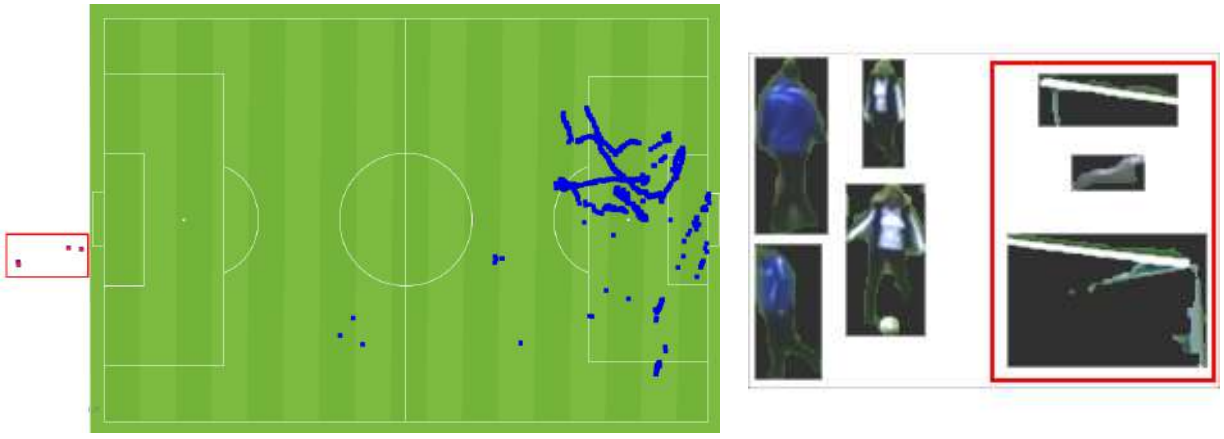
Im diesem Fall, in dem die Kameras sich nicht bewegen, wird mit der *Gaussian Mixture-based Background Subtraction*-Methode nach Zivkovic (2004) ein Vordergrund-Hintergrund-Trennungsverfahren genutzt, um die Objekte im Bild zu detektieren. Dieses Verfahren nimmt eine Segmentierung des Bildes in Vordergrund- und Hintergrundpixel vor und liefert eine Maske der Vordergrundpixel. Im Detail ist dies ein Ansatz, der auf Pixelebene arbeitet und zusammengesetzte Normalverteilungsdichten (*Gaussian mixture probabilitiy densities*) nutzt, um die Hintergrundpixel des Bilds zu beschreiben. Gelingt es nicht, das gleiche Pixel im neuen Bild auf diese Weise zu beschreiben, so wird dieses Pixel als Vordergrundpixel behandelt. Die zur Beschreibung verwendeten Parameter und Komponenten der zusammengesetzten Verteilung werden dabei automatisch angepasst. In der zurückgegebenen Vordergrund-Maske werden die einzelnen Vordergrundpixel zu Regionen zusammengefasst und das Zwischenergebnis mit Hilfe von morphologischen Operationen (siehe Luhmann (2000)) sowie anhand von geometrischen und räumlichen Rahmenbedingungen gefiltert, um Fehldetektionen möglichst zu eliminieren. Ein Beispiel für die Extraktion der Detektionen wird in Abbildung 4.2 gezeigt. Dort wird in 4.2a der kontinuierlich gelernte Hintergrund, in 4.2b die Segmentierung in Vordergrund- (weiß) und Hintergrundpixel (schwarz) und in 4.2c die sich ergebenden Detektionen (rot eingerahmt) dargestellt.



(a) *(b)* *(c)*

Abbildung 4.2: Ein Beispiel für die Detektion von Objekten mittels Vordergrund-Hintergrund-Trennung: a) der Hintergrund der Szene, b) Segmentierung des Hintergrunds (schwarze Pixel) bzw. Vordergrunds (weiße Pixel) und c) die rot eingerahmten Detektionen

Die morphologischen Operationen bestehen aus dem aufeinanderfolgenden Anwenden von *Opening*- und *Closing*-Operationen und dienen zur Beseitigung von Rauschen und zur Trennung von nahe beieinander befindlichen Objekten. Die Detektionen werden durch ein achsparalleles Begrenzungsrechteck (*Boundingbox*) mit zusätzlichen Informationen repräsentiert. Als geometrische Bedingungen werden die Fläche der Boundingbox A_{bbox} und das Verhältnis ihrer Breite und Höhe w/h_{max} in Betracht gezogen, um Detektionen ausschließen zu können, die auf Grund ihrer Form nicht von relevanten Objekten stammen können. Die räumlichen Bedingungen bestehen aus der Überprüfung, ob sich das Objekt noch in dem gültigen Raum befindet bzw. eine Bewegung an oder zu dieser Position überhaupt möglich ist. In Abbildung 4.3a werden für das Beispielszenario des Fußballspieler-Trackings die Detektionen zu einem Zeitpunkt dargestellt. Hier werden die Detektionen, die sich außerhalb des Fußballplatzes befinden als Fehldetektionen verworfen. Dazu werden die Detektionen im Bild verworfen, die die geometrischen Bedingungen nicht erfüllen (4.3b).



(a) *(b)*

Abbildung 4.3: Bei der räumlichen (a) und geometrischen Filterung (b) der Detektionen werden jeweils die in rot eingerahmten Detektionen als Fehldetektionen angesehen und verworfen. In a) sind hierbei die Detektionen durch blaue Kästchen dargestellt. In b) werden die Bildregionen der Detektionen gezeigt.

Im Gegensatz zu den GPS-Trajektoriepunkten, die eindeutige Objektkennungen $objId$ beinhalten, sind die Kameradetektionen keinen Objekten zugeordnet. Für den späteren Prozess, in dem die unterschiedlichen Eingangsdaten fusioniert werden, wird den Detektionen jeweils ein Merkmalsvektor hinzugefügt, der für die entsprechende Region im Kamerabild berechnet wird. Diese Merkmale sollen dazu dienen, die Objekte in Folgebildern möglichst wiederzuerkennen. In diesem Ansatz wird ein Histogramm H_{hue} benutzt, welches die Verteilung der Farbwerte (*Hue*) (nach Transformation des Bildes in den HSV-Farbraum) erfasst. Die Positionskomponente ergibt sich

aus dem Mittelpunkt der Boundingbox, was dem auf den Boden projizierten Körperschwerpunkt entsprechen soll. Die von einer Kamera erhaltenen Informationen in einem Zeitschritt ist so die Menge D an Detektionen D_i

$$D = \{D_1, D_2, \ldots, D_n\}, \tag{4.2}$$

wobei jede Detektion D_i definiert ist durch

$$D_i = (x,\, y,\, t,\, H_{hue})_i. \tag{4.3}$$

Wie die GPS-Positionsmessungen besitzen auch die Positionsinformationen der Detektionen eine gewisse Unsicherheit. Diese resultiert aus Detektionsungenauigkeiten bei denen die Objekte entweder nicht vollständig (z.B. die Füße sind abgeschnitten), zu groß (z.B. Schatten wird als Teil des Objekts angesehen) oder in einer untypischen bzw. asymmetrischen Pose (z.B. ein ausgestreckter Arm) detektiert worden sind. Diese Umstände führen zu einer berechneten Position, die nicht der tatsächlichen Position des Objekts entspricht. Die Ungenauigkeit wird als normalverteilt angenommen und mit Hilfe der Standardabweichung σ_{Kamera} modelliert.

Dieser Ansatz zur Generierung der Kameradaten garantiert leider nicht, dass auch jedes Objekt wirklich detektiert wird. So geht dieses Verfahren bspw. von der Annahme aus, dass Detektionen individuelle Objekte repräsentieren. Diese wird jedoch verletzt, sobald Objekte entweder so nah beieinander sind (*partielle Verdeckung*), dass deren Regionen im Bild zu einer Großen verschmelzen und lediglich durch eine gemeinsame Detektion repräsentiert werden, oder sich gar komplett verdecken. Zudem dürfen sich die Objekte nicht zu statisch verhalten, weil sie sonst bei Verfahren, die ständig den Hintergrund lernen, nicht mehr von diesem unterschieden werden können. Unter diesen Umständen kommt es zu fehlenden Detektionen, mit denen im späteren Verlauf der Tracking-Algorithmus umgehen können muss. Im Allgemeinen geht es dabei darum, die Menge an fehlenden oder falschen Informationen zu minimieren, um das Tracking der Objekte zu vereinfachen.

Sind bei einer Anwendung die Voraussetzungen an die Kameras (statisch) und die Objekte (dynamische) nicht gegeben, so muss zur Detektion der Objekte auf andere Methoden zurückgegriffen werden. Hierfür würden sich je nach Anwendungsfall Verfahren anbieten, die Objekte anhand gewisser Eigenschaften im Bild erkennen. So könnten u.a. Deskriptoren wie eindeutige Farbinformationen oder *HOG* (*histogram of oriented gradients*) (Dalal und Triggs, 2005) verwendet werden. Die Entwicklung neuer Detektionsmethoden liegt jedoch nicht im Schwerpunkt dieser Arbeit, sodass hier wie einleitend beschrieben, auf existierende Verfahren zurückgegriffen wird.

4.3 Vorverarbeitung

Da Daten aus unterschiedlichen Datenquellen (Kameradaten, GPS-Daten) mit ebenso unterschiedlichen Koordinatensystemen integriert werden sollen, müssen diese zunächst zeitlich und „räumlich“ synchronisiert werden. Zu diesem Zweck werden beide in ein gemeinsames Koordinatensystem transformiert. Die zeitliche Synchronisation erfolgt auf Basis der Zeitstempel, sofern diese von einem gemeinsamen Zeitgeber stammen. Ist dies nicht der Fall, wird der zeitliche Offset zwischen den einzelnen Geräten berechnet, indem bspw. ein gemeinsam beobachtetes Event (z.B. eine bestimmte Objektbewegung oder eine Objekt befindet sich an einer eindeutigen Position) als Synchronisationspunkt genutzt wird. Die Zeitstempel der Messungen der Sensoren werden daraufhin entsprechend angepasst. Die integrierten Daten sind der Input für die Datenfusion (siehe Schema in Abbildung 4.1).

4.4 Fusion der heterogenen Daten

Die Ausgangssituation vor der Fusion stellt sich nun wie folgt dar: Es existiert jeweils eine zusammenhängende GPS-Trajektorie für jedes beobachtete Objekt. Zudem gibt es Kameradetektionen, d.h. beobachtete Objekte mit Zeitstempel, Position und Merkmalsvektor. Die Positionsinformationen beider Datenquellen sind mit Unsicherheiten behaftet. Diese liegen für die GPS-Positionen ungefähr im Bereich von 5 - 20 m, für die Kameradetektionen im Bereich von wenigen Zentimetern (siehe Abschnitt 2.2.3). In dem nun folgenden Schritt werden die übrigen drei Teilprobleme behandelt. Mit Hilfe eines Ansatzes zur Fusion von heterogenen Daten sollen die Objekt-Trajektorien generiert werden. Die Fusion der Objektpositionsinformationen aus den Kameradetektionen mit den GPS-Trajektorien basiert dabei auf einer Modellierung als HMM und dem Viterbi-Algorithmus, der letztendlich die wahrscheinlichsten Trajektorien der überwachten Objekte ermittelt. Wie in Abbildung 4.4 schematisch dargestellt ist, besteht die Herausforderung darin, korrespondierende Daten zwischen Kameradetektionen auf der einen Seite und GPS-Trajektorien auf der anderen Seite zu identifizieren, um mit ihnen die Positionen für die Objekt-Trajektorien bestimmen zu können.

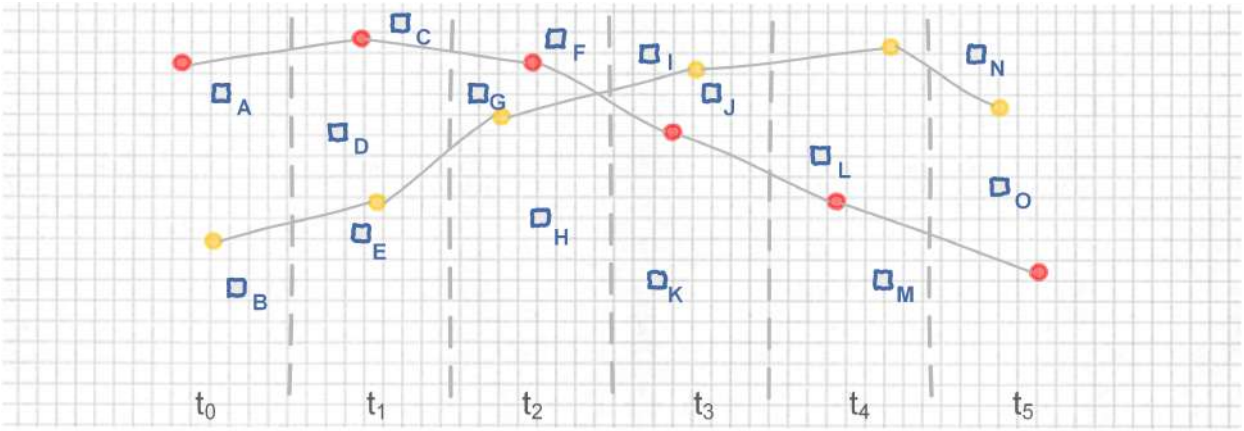


Abbildung 4.4: Die Datenzuordnungsaufgabe: Das initiale Setting beinhaltet zwei durchgängige GPS-Trajektorien (verbundene rote und gelbe Punkte) und nicht zugeordnete Kameradetektionen (einzelne blaue Kästchen), beide im gemeinsamen Koordinatensystem. Der zeitliche Verlauf wird durch die vertikalen Trennlinien symbolisiert und ist unten annotiert.

Die Modellierung als HMM kann dabei auf unterschiedliche Weise erfolgen. In den beiden folgenden Abschnitten werden zwei Varianten beschrieben, die sich in der Modellierung der nicht-beobachtbaren Zustände (*hidden states*) und der Beobachtungen (*observations/emissions*) unterscheiden. Beide Modellierungsvarianten besitzen jeweils eigene Vor- und Nachteile, die ebenfalls in diesem Zusammenhang erläutert werden.

4.4.1 Detektionsbasierte Modellierung

Die erste Variante der Modellierung des HMM zielt darauf ab, die Objektpositionen, die durch die ungenaueren GPS-Messungen ermittelt worden sind, anhand der genaueren Positionen aus den Kameradetektionen zu verbessern (Feuerhake u. a., 2015). Hierbei entsteht ein Zuordnungsproblem, da die Detektionen ohne Weiteres nicht einem Objekt zuzuordnen sind ($objId$ ist wie beschrieben nicht vorhanden). Dieses Problem soll mit Hilfe der GPS-Trajektorien gelöst werden, die Informationen für eine mögliche Zuordnung liefern. Da zudem die Standardabweichung der Kamerapositionsmessungen in der Regel viel kleiner als die der GPS-Messungen ist, ist die beste

Positionsschätzung nahezu unabhängig von der GPS-Position. In dem HMM wird dieses wie folgt repräsentiert: Der aktuelle (nicht-beobachtbare) Zustand X ist ein Tupel aus Informationen und beinhaltet die Zuordnung einer Kameradetektion zu einer GPS-Trajektorie, eine detektierte Region im Kamerabild und ein Hue-Histogramm H_{hue}. Die Beobachtungen Y sind durch die von der Kamera und dem GPS-Sensor gemessenen Positionen gegeben.

Der alleinige Einfluss einer Kameradetektion auf die Objektposition führt zu einer vereinfachten Repräsentation. In dieser wird die Positionskomponente des Zustands auf die Position der Kameramessung gesetzt, die aus der zugeordneten Bildregion abgeleitet worden ist. Daher ist bei einem gegebenen Zustand die Beobachtungswahrscheinlichkeit B allein von der Distanz zwischen der Positionskomponente des Zustands und der GPS-Position abhängig. Dies entspricht wiederum der Distanz zwischen Kamera- und GPS-Position.

Die Zustandsübergangswahrscheinlichkeiten A werden im Allgemeinen durch den Vergleich der Merkmalsvektoren aufeinanderfolgender Zustände berechnet. In diesem Fall werden Histogramm-Ähnlichkeiten und Einschränkungen durch ein Bewegungsmodell genutzt. Als Maß der Ähnlichkeit der Farbwerthistogramme wird eine Histogrammkorrelation (*histogram correlation*) herangezogen, die Werte im Bereich zwischen 0 (keine Ähnlichkeit) und 1 (gleich) liefert. Sie ist definiert als

$$p(H_{hue,1},H_{hue,2}) = \frac{\sum_I (H_{hue,1}(I) - \bar{H}_{hue,1})(H_{hue,2}(I) - \bar{H}_{hue,2})}{\sqrt{\sum_I (H_{hue,1}(I) - \bar{H}_{hue,1})^2 \sum_I (H_{hue,2}(I) - \bar{H}_{hue,2})^2}}, \tag{4.4}$$

mit I als Indizes aller Hue-Werte und $\bar{H}_{hue,k} = \frac{1}{N}\sum_J H_{hue,k}(J)$ als Hue-Mittelwerte. Auch hier wird das Modell vereinfacht, indem das Histogramm für den aktuellen Zustand nicht gefiltert wird. Demnach wird die Histogramm-Komponente des Zustands nach Zuordnung einer Bildregion nicht aktualisiert, sondern durch das neue Histogramm, das auf Basis der zugeordneten Bildregion erzeugt wird, ersetzt. Die zweite Komponente der Übergangswahrscheinlichkeiten ist eine Indikatorfunktion, welche 1 liefert, falls die folgende Position von der Vorherigen erreichbar ist. Das bedeutet, dass die Geschwindigkeit v der Objekte kleiner gleich einer Maximalgeschwindigkeit v_{max} ist, die je nach Anwendungsfall gewählt wird.

$$v \leq v_{max} \tag{4.5}$$

Eine weitere substantielle Modifikation einer klassischen HMM-Modellierung betrifft die simultane Berechnung mehrerer Objekt-Trajektorien. Bis hierher wurde ein HMM beschrieben, bei dem ein Zustand aus einer Einzelposition, einer Bildregionzuordnung und einem Hue-Histogramm besteht. Da jedoch in vielen Anwendungsfällen mehrere Objekte gleichzeitig beobachtet werden sollen, liegt eine Kernaufgabe darin, die korrekte Zuordnung für mehrere Trajektorien zu finden. Aus diesem Grund ist eine simultane Zuordnung von mehreren Tracks von Interesse; besonders dann, wenn diese nahe beieinander liegen. Das Ziel ist daher eine eindeutige Zuordnung von GPS-Positionen zu Bildregionen. Ähnlich wie in Xie und Evans (1990) beschrieben, werden die Zustände so definiert, dass sie anstelle von Einzelzuordnungen Tupel an Zuordnungen enthalten. Die Tupel sind Variationen der aktuellen Menge an Kameradetektionen. Als Beispiel sei eine Menge aus $l = 3$ Detektionen $\{F, G, H\}$ und Beobachtungen von $n = 2$ Objekten gegeben, wie es im zweiten Zeitschritt es im Beispielszenario (Abbildung 4.4) auftritt. In diesem Fall gibt es $M = \frac{l!}{(l-n)!} = 6$ Zuordnungsmöglichkeiten zu den Trajektorien ohne Wiederholungen, nämlich $\{(F,G), (F,H), (G, F), (G,H), (H,F), (H,G)\}$. Demnach ist jede Zustandssequenz eine Sequenz an Zuordnungstupeln.

Des Weiteren muss das Problem der fehlenden Daten durch nicht-detektierte Objekte gelöst werden. Dieses einsteht, wenn Objekte entweder partiell bzw. gänzlich verdeckt sind oder gar das Sichtfeld der Kamera verlassen. Das Problem kann auf unterschiedliche Weise gelöst werden: Zum

einen können die Zeitschritte, in denen zu wenig Objekte detektiert worden sind, gänzlich verworfen werden. Dieses führt jedoch dazu, dass auch eigentlich durchgängige Trajektorien von Objekten, die die gesamte Zeit detektiert worden sind, unterbrochen werden und damit korrekte Informationen verworfen werden. In Szenarien, in denen durch eine hohe Objektdichte viele Verdeckungen zu erwarten sind, wäre es dadurch nur bedingt möglich, längere bzw. durchgängige Trajektorien zu generieren. Eine alternative Möglichkeit zur Lösung dieses Problems ist das Hinzufügen von „*Dummy-Zuordnungen*“ (im späteren auch Dummies oder *Dummy-Detektionen* genannt) für jedes Objekt, um eventuell fehlende Detektionen auszugleichen. Diese Dummy-Werte repräsentieren den Zustand „nicht-detektiert“. Da diese „virtuellen Detektionen“ fehlende Messungen ersetzen und somit keine neuen Informationen in Bezug auf Bildregion und Hue-Histogramm verfügbar sind, bleiben die Positions- und Hue-Histogramm-Komponente des vorherigen Zustands unmodifiziert. Im vorherigen Beispiel ändert sich dadurch die Menge an Detektionen zu $\{F, G, H, \emptyset_1, \emptyset_2\}$, mit $\emptyset_1$ und $\emptyset_2$ als „Dummy-Zuordnungen“ für die beiden Objekte. Die Zuordnungsmenge vergrößert sich zu $M = 20$ Elementen. In Abbildung 4.5 wird das HMM für das Beispiel aus Abbildung 4.4 dargestellt.

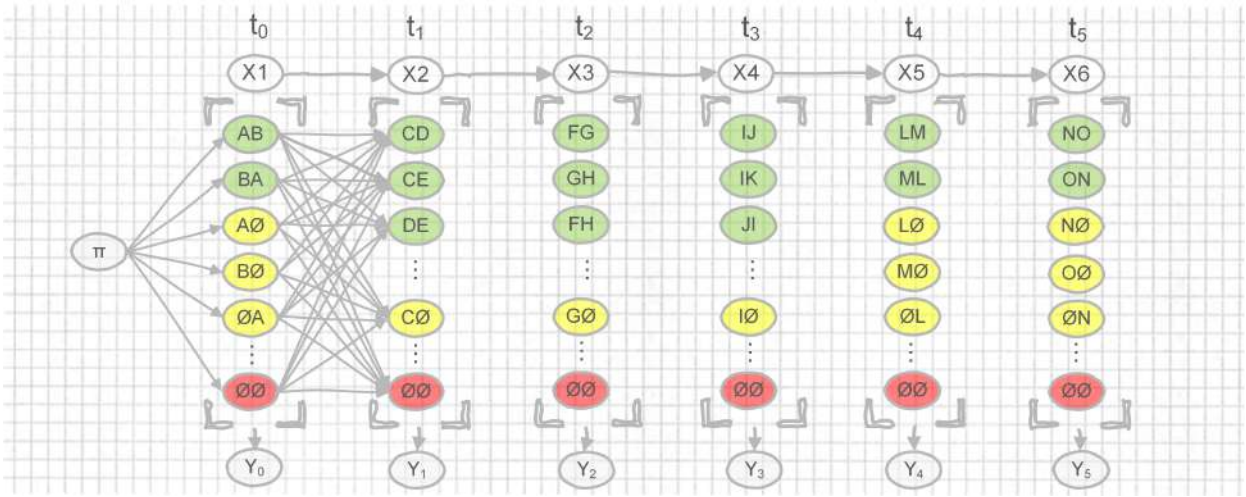


Abbildung 4.5: Ein HMM für mehrere Trajektorien. Oben: Das Dynamische Bayessche Netz wird durch die Sequenz an Zuständen (in weiß) gebildet. Darunter: Die farbigen Knoten sind die möglichen Zustandswerte (die Zuordnungstupel) für die Zustandssequenz X1 bis X6. Durch die Farbe ist die Anzahl an Dummy-Zuordnungen (grün: nur reale Detektionen; gelb: mindestens eine Dummy-Zuordnung; rot: nur Dummy-Zuordnungen) codiert. Die grauen Kanten symbolisieren die Übergangsmöglichkeiten zwischen den entsprechenden Zuständen (hier sind übersichtshalber nur die Übergänge zwischen t_0 und t_1 dargestellt). Die Initialisierung (links) und die Beobachtungen Y (unten) werden durch graue Symbole dargestellt.

Die initialen Zustände des Modells werden mit gleichverteilten Wahrscheinlichkeiten initialisiert:

$$\pi_i = \frac{1}{M}, \quad i = 1 \dots M \tag{4.6}$$

Dies bedeutet, dass keine Vorinformationen bzgl. der Startpositionen der überwachten Objekte bekannt ist, d.h. die wahrscheinlichste Trajektorie hängt nicht vom Anfangszustand ab.

Da die Zustände durch die zuvor beschriebenen Modifikationen Zuordnungstupel enthalten, müssen die Zustandsübergangs- und Beobachtungswahrscheinlichkeiten entsprechend angepasst werden. Für die Zustandsübergänge bedeutet dies, dass die Histogramm-Ähnlichkeiten als Wahrscheinlichkeiten von unabhängigen Zufallsvariablen behandelt werden, sodass deren Verbundwahrscheinlichkeit durch das Produkt ihrer Randwahrscheinlichkeiten gegeben ist. Seien die Tupel X_1 und X_2

mit N als der Anzahl der Objekte sowie $H_{1,i}$ und $H_{2,i}$ als i-te Histogramme des ersten bzw. zweiten Tupels gegeben, so berechnet sich die Gesamtähnlichkeit durch

$$p(X_1,X_2) = \prod_{i=1\dots N} p(H_{1,i},H_{2,i}) = \prod_{i=1\dots N} \frac{\sum_I (H_{1,i}(I) - \bar{H}_{1,i})(H_{2,i}(I) - \bar{H}_{2,i})}{\sqrt{\sum_I (H_{1,i}(I) - \bar{H}_{1,i})^2 \sum_I (H_{2,i}(I) - \bar{H}_{2,i})^2}}. \tag{4.7}$$

Unter Berücksichtigung einer zusätzlichen Maximalgeschwindigkeitsbedingung ergeben sich die folgenden Zustandsübergangswahrscheinlichkeiten:

$$a_{ij} = \begin{cases} p(X_i,X_j) & \text{falls } v_{i,j} \leq v_{max} \\ 0 & \text{sonst} \end{cases}, \quad i,j = 1 \dots M \tag{4.8}$$

Unter der Annahme, dass die GPS-Beobachtungen unabhängig und identisch normalverteilt sind, werden die Beobachtungswahrscheinlichkeiten gemäß dem Produkt ihrer Dichten modelliert. Sie berechnen sich durch

$$b_{jk} = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_{GPS}} e^{-\frac{d_i^2}{2\sigma_{GPS}^2}} = (\frac{1}{\sqrt{2\pi}\sigma_{GPS}})^N e^{-\frac{1}{2}\sum_{i=1}^{N} \frac{d_i^2}{\sigma_{GPS}^2}}, \quad j = 1 \dots M, k = 1 \dots K \tag{4.9}$$

mit N als Anzahl der Objekte (Länge des Tupels) und $d_i = x_j - \mu_{GPS,k}$ als paarweise Euklidische Distanzen zwischen den Positionskomponenten x_j des Zustands X_j und den Positionen der zugeordneten (GPS-) Beobachtungen Y_k.

Diese Modellierung bringt sowohl Vor- als auch Nachteile mit sich. Ein Vorteil ist, dass die resultierenden Trajektorien ausschließlich auf Grundlage der Kameradetektionen generiert werden und somit eine hohe geometrische Genauigkeit aufweisen. Zudem ist die Laufzeit des Algorithmus auf Grund der im Vergleich zu der (im nachfolgenden Abschnitt vorgestellten) rasterbasierten Modellierung deutlich geringeren Anzahl an Zuständen besser. Für zeitkritische Anwendungen bietet sich daher diese Modellierung an. Jedoch hat dieses Vorgehen auch eine Kehrseite. Denn es führt in Situationen, in denen ein Objekt länger verdeckt ist oder sich aus dem Sichtbereich der Kamera bewegt, dazu, dass das Objekt zwar wiedererkannt werden kann, die Trajektorie aber für die Zeit ohne Detektion nicht verfügbar ist (vgl. Abbildung 4.6). Die GPS-Positionen dienen hierbei lediglich der Zuordnung der Detektionen zu den Objekt-Trajektorien und fließen nicht in die Objektposition ein, daher muss hier zwischen den Detektionen interpoliert werden. Ein weiterer Nachteil betrifft ebenfalls den Umgang mit fehlenden Objektdetektionen. Zwar werden zu diesem Zweck künstliche „Dummy-Zuordnungen“ eingeführt, die dieses Problem auffangen sollen, jedoch ist die Bestimmung der Zustandsübergangswahrscheinlichkeiten zwischen Zuständen, die aus diesen „Dummy-Zuordnungen“ bestehen, nicht trivial. Im Fall von echten Detektionen berechnen diese sich anhand der Positions- und Farbwert-Histogramm-Komponenten. Da diese Komponenten der „Dummy-Zuordnungen“ allerdings mit Informationen aus den vorangegangenen zugeordneten Objektdetektionen aufgefüllt werden, ist die Differenz sowohl der Position als auch der Farbwerte null. Die Wahrscheinlichkeit für diesen Übergang wäre entsprechend sehr hoch im Vergleich zu den Übergängen zwischen oder zu Zuständen aus tatsächlichen Objektdetektionen. Das Ergebnis wäre daher höchstwahrscheinlich eine Abfolge von „Dummy-Zuordnungen“. Um dieses Problem zu umgehen, wird die Übergangwahrscheinlichkeit künstlich mit Hilfe des Parameters P_{pen} bestraft, sobald ein oder mehrere „Dummies“ beteiligt sind. Die Festlegung der Höhe der Bestrafung erlaubt eine Steuerung, inwiefern das Tracking zu „Dummy-Zuordnungen“ neigt. Eine ausgeprägte Neigung sorgt dafür, dass gegebenenfalls auch korrekte Zuordnungen den Dummies weichen müssen. Eine zu gering ausgeprägte Neigung führt wiederum dazu, dass fehlende Detektionen durch unrea-

listische Falschzuordnungen anstelle von Dummies kompensiert werden. Die Wahl von P_{pen} hängt vom jeweiligen Anwendungsszenario, jedoch gilt

$$0 < P_{pen} \leq 1 \tag{4.10}$$

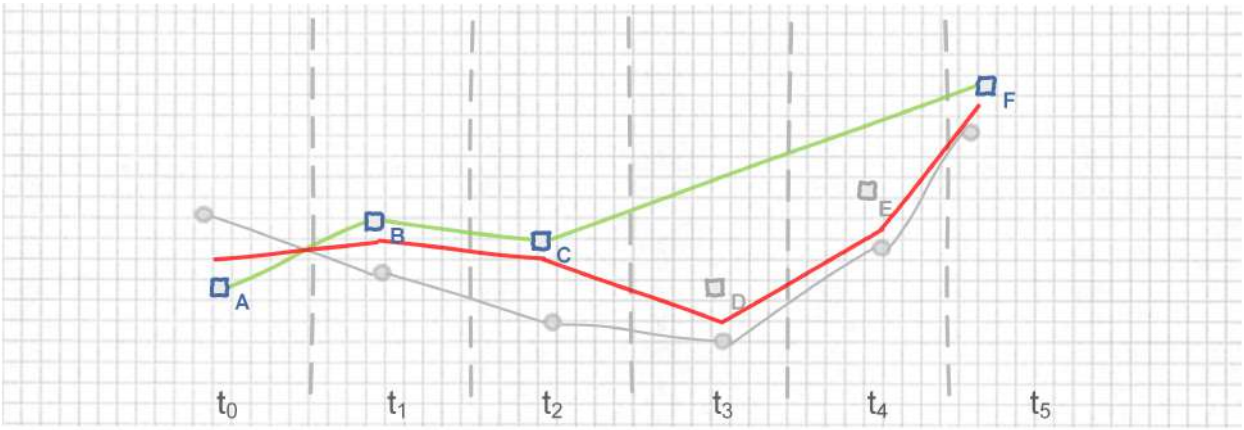


Abbildung 4.6: Bei der detektionsbasierten Modellierungsvariante (grüne Trajektorie) stehen für die Zeit ohne Objektdetektionen (t_3 und t_4, Ground-Truth-Detektionen in grau) keine Positionen zur Generierung der Trajektorien zur Verfügung. Ihr Verlauf muss daher in diesem Zeitraum interpoliert werden. Detaillierte Bewegungen, wie dieser Bogen, gehen mehr oder weniger verloren. Bei der rasterbasierten Variante (in rot, Abschnitt 4.4.2) nähert sich die Trajektorie in der Zeit den etwas ungenaueren GPS-Positionen an. Hier muss nicht interpoliert werden, um etwaige Lücken zu schließen.

4.4.2 Rasterbasierte Modellierung

Die zuvor geschilderten Nachteile einer detektionsbasierten Modellierung führen zu einer zweiten Modellierungsvariante. Diese zielt darauf ab, auf Dummy-Zuordnungen zu verzichten und das Problem der fehlenden Objektdetektionen auf andere Weise zu lösen. Zudem werden zur Positionsbestimmung der Objekte auch die GPS-Daten genutzt, sodass Lücken in den Kamerabeobachtungen mit Hilfe der GPS-Positionsinformationen geschlossen werden und dadurch durchgehende Trajektorien entstehen (vgl. 4.6).

Auf Grund der Tatsache, dass die Kameradetektionen nicht direkt einer Objekt-Trajektorie zugeordnet sind und zudem mit den GPS-Daten fusioniert werden müssen, lassen sich die Objektpositionen als Kombination der unterschiedlichen Positionsinformationen nicht direkt beobachten. Sie werden daher als die nicht-beobachtbaren Zustände X im HMM modelliert. Diese Modellierung verlangt einen diskretisierten Raum, in dem sich die Objekte bewegen können. Durch die Zerlegung des Raums in eine regelmäßige Rasterstruktur entstehen Rasterzellen, die kleinen Gebietsabschnitten und damit möglichen Aufenthaltsorten der Objekte entsprechen. Die Objekt-Trajektorien werden somit anhand von Sequenzen an besuchten Rasterzellen ermittelt. Die Geometrien der Trajektorien hängen aus diesem Grund von der Rasterauflösung ab, welche sich durch die Dimension der Zellen ergibt. Je höher die Auflösung gewählt wird, desto feiner kann der Verlauf der Trajektorie bestimmt werden. Gleichzeitig steigt jedoch auch die Anzahl an Rasterzellen und damit auch die Anzahl an möglichen Zuständen, was wiederum einen Einfluss auf die Laufzeit und den Speicherbedarf des Algorithmus hat (siehe Abschnitt 4.5). In Abbildung 4.7 wird die Rasterstruktur für das bisherige Beispiel gezeigt.

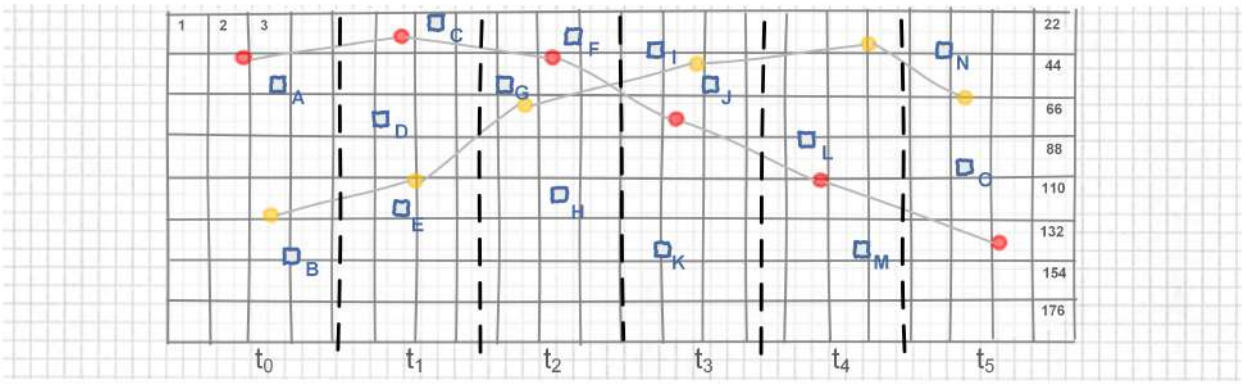


Abbildung 4.7: Zur Diskretisierung des Bewegungsraums der Objekte wird ein Raster genutzt. Dieses unterteilt das gesamte Gebiet in kleinere Abschnitte, zwischen denen die Objekte sich bewegen können.

Die Beobachtungen Y sind die GPS-Daten sowie die Kameradetektionen, welche eine Zuordnung zu einer detektierten Region im Kamerabild, eine Position und ein Hue-Histogramm enthalten. Im Schema aus Abbildung 4.8 ist das resultierende HMM inklusive der möglichen Zustände für jeden Zeitschritt zu sehen.

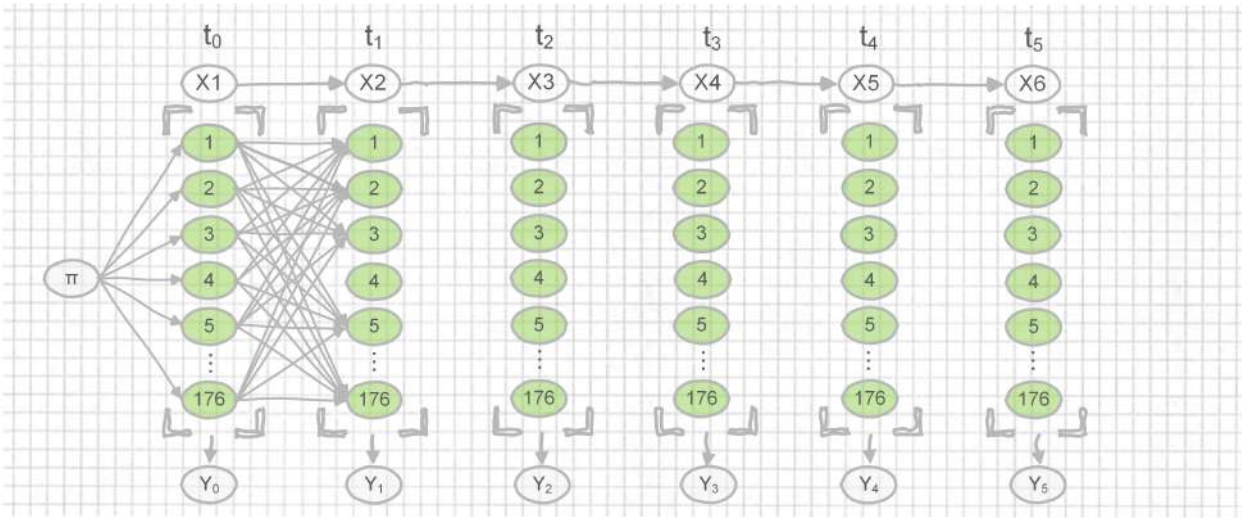


Abbildung 4.8: In dem auf Grundlage des Rasters modellierten HMM können die Zustand in jedem Zeitschritt (oben, weiß) die sich aus den einzelnen Rasterzellen ergebenden möglichen Zuständen (grün) annehmen. Die einheitliche grüne Färbung deutet an, dass hier keine künstlichen Zustände existieren. Die Beobachtungen Y werden durch die grauen Symbole (unten) dargestellt.

Bei der Initialisierung (π) hat sich im Vergleich zu der vorangegangenen Modellierungsvariante nichts geändert. Es wird weiterhin eine Gleichverteilung für die initialen Zustände angenommen. Zur Berechnung gilt somit auch hier Gleichung 4.6.

Bei der Berechnung der Zustandsübergangswahrscheinlichkeiten A werden die Geschwindigkeiten der Objekte berücksichtigt. Für die Berechnung der aktuellen Geschwindigkeit v muss neben der aktuellen auch die vorherige Rasterzelle bekannt sein. Da diese jedoch ohne sich dabei auf eine vorangegangene Abfolge an Rasterzellen bzw. Zuständen festzulegen, nicht bekannt ist und somit keine Geschwindigkeit bestimmt werden kann, wird hier als vereinfachtes Bewegungsmodell eine (objekt-) typische Geschwindigkeitsverteilung angenommen. Diese wird genutzt, um Aussagen darüber zu treffen, wie wahrscheinlich der Besuch einer Zelle ist, indem die Wahrscheinlichkeit dafür ermittelt wird, dass ein Objekt sich mindestens mit der dafür benötigten Minimalgeschwindigkeit

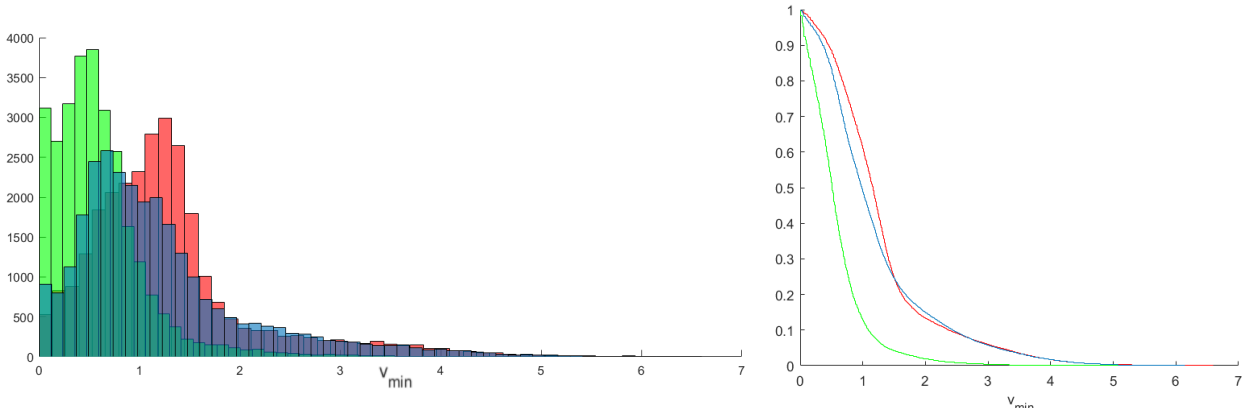
v_{min} weiterbewegt. Letztere berechnet sich anhand der kürzesten euklidischen Distanz zwischen den jeweiligen Zellen $d(X_i,X_j)$ und der verstrichenen Zeit Δt.

$$v_{min}(X_i,X_j) = \frac{d(X_i,X_j)}{\Delta t} \tag{4.11}$$

Die Übergangswahrscheinlichkeiten ergeben sich durch folgende Formel. Sie lassen sich für jede Zelle an Stelle v_{min} der *Überlebensfunktion* ablesen.

$$a_{ij} = P(v \geq v_{min}(X_i,X_j)) = 1 - P(v \leq v_{min}(X_i,X_j)), \quad i,j = 1 \dots M. \tag{4.12}$$

Die Geschwindigkeitsverteilung ist vom Anwendungsfall und den beobachteten Objekten bzw. Objektklassen (z.B. Fußgänger, Autos, Fahrradfahrer aber auch Sportler wie Jogger, Fußballer oder Basketballer) abhängig. Jede dieser Klassen besitzt eine eigene typische Verteilung. Prinzipiell sind die Verteilungen sogar für jedes Objekt individuell und könnten, sofern sie für die jeweiligen Objekte bekannt sind, anstelle der objekttypischen Variante verwendet werden. Für das Beispiel eines Fußballspielers bedeutet dies, dass es neben der allgemeinen Verteilung für Fußballspieler auch eine rollenspezifische (ein Angreifer bewegt sich anders als ein Mittelfeldspieler) und eine individuelle Verteilung gibt. Sofern die Verteilungen nicht schon vorliegen, können sie auf unterschiedliche Weise bestimmt werden. Eine recht einfache Methode ist, die Objekte mittels GPS zu tracken und die Geschwindigkeitsverteilung anhand der GPS-Trajektorien empirisch zu bestimmen. Je nach Belieben und verträglichem Aufwand können so allgemeine objekttypische oder individuelle Verteilungen erzeugt werden. In Abbildung 4.9a werden unterschiedliche individuelle Geschwindigkeitsverteilungen gezeigt. Hierbei handelt es sich um ein Fußballspieler mit unterschiedlichen Rollen: einen Torhüter (grün), der sich eher wenig und langsam bewegt, einen Mittelfeldspieler (blau), der viel mit mittleren Geschwindigkeiten unterwegs ist und einen Angreifer (rot), der sich häufiger schnell bewegt. Es sind Unterschiede zu erkennen und diese wirken sich auf Zustandsübergangswahrscheinlichkeiten aus. Diese sind in den entsprechend gefärbten Überlebensfunktionen in 4.9b dargestellt. Sind keine Geschwindigkeiten der Objekte im Voraus messbar und auch keine externen Quellen bekannt, so kann in diesem Fall eine Gleichverteilung bis zu einer geschätzten Maximalgeschwindigkeit angenommen werden.



(a) *(b)*

Abbildung 4.9: a) Unterschiedliche, empirisch bestimmte Geschwindigkeitsverteilung der zu beobachtenden Objekte. In diesem Beispiel sind es Fußballspieler mit unterschiedlichen Rollen: Torhüter (grün), Mittelfeldspieler (blau), Angreifer (rot). b) Die Unterschiede in den Verteilungen wirken sich ebenfalls auf die Zustandsübergangswahrscheinlichkeiten aus.

Handelt es sich wie im hier gegebenen Fall um empirisch bestimmte Geschwindigkeitsverteilungen, so erfolgt die Berechnung der durch Gleichung 4.12 beschriebenen Wahrscheinlichkeiten mit Hilfe

der kumulierten relativen Häufigkeiten der jeweiligen Geschwindigkeiten $h(v_k) = \frac{n_{v_k}}{n_v}$ mit insgesamt n_v gemessenen Geschwindigkeitswerten.

$$a_{ij} = P(v \geq v_{min}(X_i,X_j)) = 1 - P(v \leq v_{min}(X_i,X_j)) = 1 - \sum_{v_{min} \leq v_k} h(v_k) \tag{4.13}$$

Wird in die Geschwindigkeitsdaten eine Verteilung hinein geschätzt oder liegt sie von vornherein vor, so wird diese entsprechend ihrer Verteilungsfunktion wie in Gleichung 4.12 beschrieben ausgewertet.

Die auf diese Weise limitierte Reichweite der Objekte ist ein weiterer Einflussfaktor bei der Wahl der Zellgröße. Wird diese so groß gewählt, dass das Erreichen der Nachbarzellen mit der Maximalgeschwindigkeit des Objekts nicht möglich ist, wird die Übergangsmatrix A zu einer Einheitsmatrix. Dies bedeutet unter Anwendung des Viterbi-Algorithmus' zur Bestimmung der Trajektorien und Berücksichtigung von Gleichung 2.29, dass die resultierende Abfolge an Zuständen aus einer Wiederholung des gleichen Zustand bzw. der gleichen Rasterzelle besteht. Die Auflösung des Bewegungsraums ist daher auch von den Geschwindigkeiten der Objekte und der Sampling-Rate abhängig und sollte so gewählt werden, dass gilt:

$$\min_{i,j} d(X_i,X_j) < v_{max}\ dt. \tag{4.14}$$

Bei der Berechnung der Beobachtungswahrscheinlichkeiten B werden zum einen die Messungen der Sensoren und zum anderen auch Apriori-Wissen verwendet. Die verfügbaren Informationen werden zu unterschiedlichen Zwecken benutzt. Die Positionsinformationen finden bei der Bestimmung der Wahrscheinlichkeiten der Zustände/Rasterzellen Verwendung. Hier wird sowohl für die Kamera- als auch die GPS-Messungen angenommen, dass diese unabhängig und identisch normalverteilt sind. Die Ungenauigkeiten werden durch die Erwartungswerte μ_{Kamera} und μ_{GPS} und Standardabweichungen σ_{Kamera} und σ_{GPS} modelliert. Für die Modellierung im zweidimensionalem Fall müssen anstelle einer univariaten Wahrscheinlichkeitsverteilung die bivariate Variante verwendet werden. Die Kovarianz-Matrizen der Sensoren Σ_{sensor} ergeben sich dann zu

$$\Sigma_{sensor} = \begin{bmatrix} \sigma sensor & 0 \\ 0 & \sigma sensor \end{bmatrix}. \tag{4.15}$$

Der Vektor der Erwartungswerte $\hat{\mu}_{sensoren}$ ist gegeben durch

$$\hat{\mu}_{sensor} = \begin{bmatrix} x \\ y \end{bmatrix}. \tag{4.16}$$

x und y sind dabei die Koordinaten der Messung. Die Verknüpfung beider Unsicherheiten erfolgt mit Hilfe des *Bayesschen Theorems* und den Update-Gleichungen (Gleichungen 2.18 und 2.19) zu

$$\Sigma_{tot} = (\Sigma_{Kamera}^{-1} + \Sigma_{GPS}^{-1})^{-1} \tag{4.17}$$

bzw.

$$\hat{\mu}_{tot} = (\Sigma_{Kamera}^{-1} + \Sigma_{GPS}^{-1})^{-1}\ (\Sigma_{Kamera}^{-1}\ \hat{\mu}_{Kamera} + \Sigma_{GPS}^{-1}\ \hat{\mu}_{GPS}). \tag{4.18}$$

Diese Parameter ermöglichen eine Steuerung der Einflüsse der unterschiedlichen Messungen. Im Fall $\sigma_{Kamera} < \sigma_{GPS}$ würde die Trajektorie sich eher an den Kameradetektion orientieren. Sind beide Standardabweichungen gleich oder ähnlich groß, würden die Trajektoriepunkte sich eher

zwischen den jeweiligen Messungen zentrieren. Die Fusionierung einer GPS- mit einer Kamera-Messung resultiert somit in einer Verteilung gegeben durch

$$f(\hat{x}|\hat{\mu}_{tot},\Sigma_{tot}) = \frac{1}{\sqrt{(2\pi)^2\ det(\Sigma_{tot})}} e^{-\frac{1}{2}(\hat{x}-\hat{\mu}_{tot})^T \Sigma_{tot}^{-1}(\hat{x}-\hat{\mu}_{tot})}. \tag{4.19}$$

Dieses gilt für die Fusion mit einer einzigen Kameradetektion. Jedoch muss berücksichtigt werden, dass in den meisten Anwendungsfällen durch das Tracking mehrerer Objekte gleichzeitig und/oder durch Fehldetektionen pro Zeitschritt mehr als nur eine Detektion vorhanden ist. Dies geschieht anhand einer Modellierung als *Mischverteilung*, die sich aus den in Gleichung 4.19 beschriebenen Fusionsverteilungen der GPS-Postion und den jeweiligen Detektionspositionen zusammensetzt.

$$f(x) = \sum_{k=1}^{K} w_k \cdot f(x_k|\hat{\mu}_{tot,k},\Sigma_{tot,k}). \tag{4.20}$$

Mit Hilfe der Gewichte w_k, für die $\sum_{k=1}^{K} w_k = 1$ gilt, erfolgt eine Zuordnung der Detektionen zu den Objekten. Dies hat zur Folge, dass die Wahrscheinlichkeiten der Zustände/Rasterzellen entsprechend der Übereinstimmung der jeweiligen Detektion mit dem tatsächlichen Objekt O gewichtet werden. Zur Ermittlung der Übereinstimmung werden wiederum Objektinformationen, wie bspw. ein Hue-Histogramm und eine Apriori-Aufenthaltswahrscheinlichkeitsverteilung, benutzt. Der Vergleich der Hue-Histogramme (Objekt: $H_{hue,O}$, Detektion: $H_{hue,k}$) erfolgt wie zuvor über die Histogramm-Korrelation mittels Gleichung 4.4. Des Weiteren kann vorhandenes Wissen über die Apriori-Aufenthaltswahrscheinlichkeiten (*heatmap*) der Objekte bzw. Objektklassen $p_{HM,O}$ miteinbezogen werden. Beispiele hierfür sind u.a. die unterschiedlichen Spielerrollen beim Fußball, die jeweils eigene typische Aufenthaltswahrscheinlichkeiten besitzen. Die Gewichte ergeben sich dadurch zu

$$w_k = \frac{p(H_{hue,O},H_{hue,k})\ p_{HM,O}(k)}{\sum_{i=1}^{K} (p(H_{hue,O},H_{hue,i})\ p_{HM,O}(i))}. \tag{4.21}$$

Die Ermittlung der Beobachtungswahrscheinlichkeiten für die N-Tupel (N entspricht der Anzahl der Objekte) erfolgt schließlich anhand der Mischverteilung aus Gleichung 4.20:

$$b_{jk} = \sum_{i=1}^{N} w_k \cdot \frac{1}{\sqrt{(2\pi)^2\ det(\Sigma_{tot})}} e^{-\frac{1}{2}(\hat{x}_j-\hat{\mu}_{tot})^T \Sigma_{tot}^{-1}(\hat{x}_j-\hat{\mu}_{tot})}, \quad j = 1 \dots M,\ k = 1 \dots K. \tag{4.22}$$

Durch diese Art der Modellierung kann somit auf Dummy-Zuordnungen verzichtet werden. Die Wahl der entsprechenden Dummy-Übergangswahrscheinlichkeiten entfällt. Zudem ergibt sich die Positionen der Objekte nicht mehr ausschließlich durch die Kameradetektionen, was bei fehlenden Detektionen zu Problemen führt. Sie werden nun mit Hilfe einer Fusion der unterschiedlichen Positionsinformationen der jeweiligen Sensoren bestimmt, sodass bei fehlenden Kameradetektionen auf die GPS-Messungen zurückgegriffen wird. Diesen Vorteilen stehen jedoch auch Nachteile gegenüber. Da die Trajektorien auf Basis einer Sequenz an besuchten Rasterzellen berechnet wird, ist stets eine gewisse Ungenauigkeit in Bezug auf die geometrische Gestalt der Trajektorien vorhanden. Die Ungenauigkeit hängt von der Rasterauflösung ab. Je feiner das Raster, desto detaillierte werden die Trajektorien abgebildet. Auch die Laufzeit der rasterbasierten Methode ist schlechter, da es im Vergleich zur vorherigen Variante im Allgemeinen eine größere Anzahl an möglichen Zuständen gibt. Beim Speicherbedarf verhält es sich ebenso.

4.4.3 Generierung der Trajektorien

In dem auf die eine oder andere Weise modellierten HMM wird daraufhin mit Hilfe des Viterbi-Algorithmus die wahrscheinlichste Sequenz an Zuständen ermittelt, anhand derer letztendlich die Objekt-Trajektorien generiert werden. Ein Vorteil, den dieser Algorithmus bietet, ist die Möglichkeit, Objekte trotz fehlender Detektionen „wiederzuerkennen“, nachdem die Situation sich aufgelöst hat, indem deren wahrscheinlichste Pfade zurückverfolgt werden.

In Abbildung 4.10 wird der wahrscheinlichste Pfad (auch *Viterbi-Pfad* genannt) für das in Abbildung 4.4 gegebene Beispiel visualisiert. Die blau eingerahmten Knoten bilden den Viterbi-Pfad, der die Sequenz an Zuordnungen und Positionen zu den Objekt-Trajektorien repräsentiert.

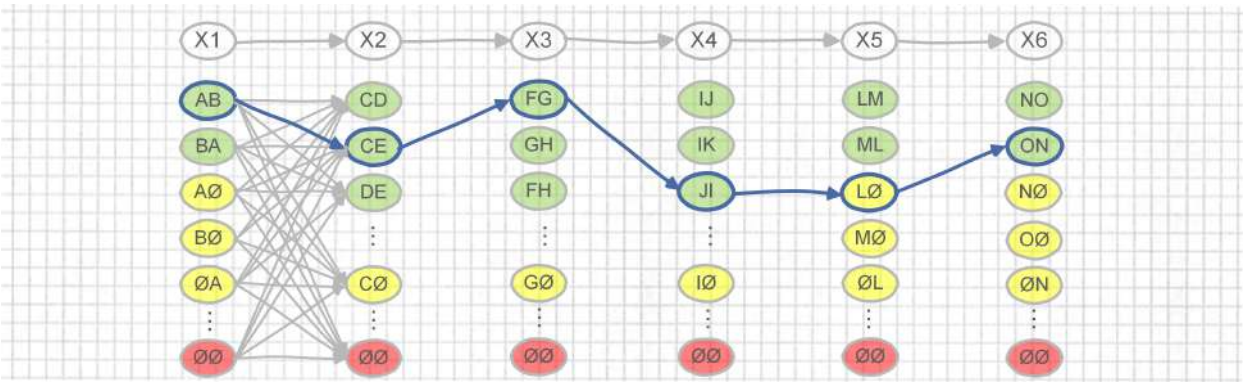
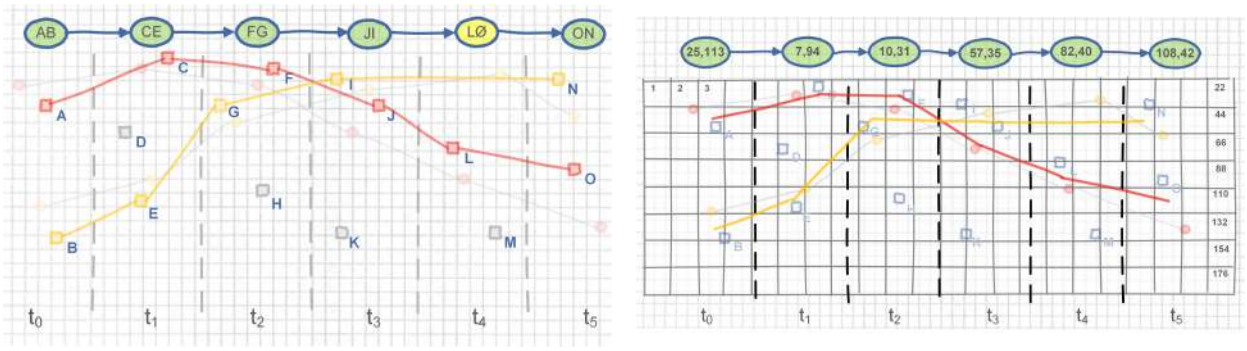


Abbildung 4.10: Der für das in Abbildung 4.4 gegebene Beispiel resultierende Viterbi-Pfad (blaue Pfeile). Hier ist lediglich das Ergebnis der detektionsbasierten Modellierung dargestellt. Der Pfad für die rasterbasierte Variante lässt sich entsprechend darstellen.

Die auf diese Weise ermittelte Sequenz an Zuständen wird nun benutzt, um für jedes Objekt die Trajektorie zu generieren. Zu diesem Zweck werden mit Hilfe der Positionen und Zeitstempel, die entlang des Pfades mit den Zuständen verknüpft sind, Trajektoriepunkte erzeugt. Die verknüpften Positionsinformationen werden je nach Modellierungsvariante entweder aus den zugeordneten Detektionen im Bild (detektionsbasierte Variante) oder den jeweiligen Rasterzellen (rasterbasierte Variante) entnommen. Die chronologisch geordneten Sequenzen an Trajektoriepunkten stellen schließlich die gesuchten Objekt-Trajektorien dar. In Abbildung 4.11 werden die Ergebnisse für die detektionsbasierte a) und für die rasterbasierte Variante b) des begleitenden Beispiels dargestellt.

4.5 Laufzeit des Algorithmus

Die Performance des vorgestellten Verfahrens hängt generell von der Anzahl der beobachteten Objekte und der möglichen Zustände des HMM ab. Die Laufzeitkomplexität ergibt sich im Wesentlichen zu $O(M^2t)$, mit M als Anzahl der Zustände und t als Anzahl der Zeitschritte. Die Anzahl der möglichen Zustände unterscheidet sich wiederum je nach Modellierungsvariante. Bei der detektionsbasierten Variante entspricht sie der Anzahl an Variationen von Detektionen im Bild, inklusive der „Dummy-Zuordnungen“, in jedem Zeitschritt. Aus diesem Grund ist bei dieser Modellierungsvariante ein effizienter Objekt-Detektierungsalgorithmus wichtig, der die Zahl an fehlerhaften Detektionen minimiert. Im Fall der rasterbasierten Modellierung ist M die Anzahl der Rasterzellen-Variationen und bleibt über die Laufzeit konstant. Die Anzahl der Rasterzellen hängt wiederum von der Rasterauflösung, also der Zellgröße und der Ausdehnung des Gebiets ab. Der Speicherbedarf verhält sich gleichermaßen. Es existieren verschiedene Ansatzmöglichkeiten,



(a) *(b)*

Abbildung 4.11: a) Detektionsbasierte Variante: Die Trajektorien (rot, orange) werden mit Hilfe der Zuordnungstupel, die im Viterbi-Pfad enthalten sind, generiert. Einige Detektionen sind verworfen worden (vereinzelte graue Kästchen). b) Rasterbasierte Variante: Hier ergeben sich die Trajektorien aus der Sequenz an besuchten Rasterzellen. Die Geometrien der Trajektorien unterscheiden sich im Vergleich zur detektionsbasierten Variante.

mit denen das Verfahren hinsichtlich der Laufzeit verbessert werden kann. Im Folgenden werden einige beschrieben.

Mit steigender Zahl von gleichzeitig beobachteten Objekten nimmt auch die Zahl der Zustände in beiden Modellierungsvarianten exponentiell zu, da jeder Zustand aus einer Variation von einem möglichen Zustand pro Objekt besteht und die Zahl der Variationen mit den Anzahl der Objekte exponentiell zunimmt. Dieses Problem kann umgangen werden, indem nicht alle Objekte in ein einziges HMM einfließen, sondern für jedes Objekt ein individuelles Modell aufgestellt wird. Der Aufwand würde dadurch nicht exponentiell, sondern nur linear mit der Objektanzahl steigen. Jedoch ist der Preis dieser Optimierungsmaßnahme, dass es zu Mehrfachzuordnungen von einer Detektionen zu verschiedenen Objekten kommen kann. Das Ergebnis kann daher schlechter werden.

Im Fall der Verwendung des rasterbasiertes Verfahrens kann die Laufzeit (analog zu Xie und Evans (1990)) verbessert werden. Dabei wird zuerst eine niedrigere Auflösung (unter Berücksichtigung der möglichen Mindestauflösung) zur Vorberechnung des groben Verlaufs der Trajektorien und zur Ermittlung der relevanten Rasterzellen bzw. Regionen des Bewegungsraums gewählt. Letztere entsprechen dem tatsächlich relevanten Bewegungsraum, in dem sich die beobachteten Objekte aufhalten. Dieser wird in einem zweiten Durchlauf feiner diskretisiert, wodurch deutlich weniger mögliche Zustände entstehen als bei einer gleich feinen Rasterung des gesamten Gebiets. Der Aufwand ergibt sich aus der Summe beider Durchläufe und wird zu $O(M^2r^4t + \frac{t^3}{r^4})$, mit r als Skalierungsfaktor von der gewünschten zur gröberen Auflösung. Werden zudem nur die für einen Zeitschritt relevanten Zustände berücksichtigt, verringert dieses den Berechnungsaufwand noch einmal deutlich zu $O(M^2r^4t + \frac{4t}{r^4})$.

4.6 Systemdesign

Das in diesem Kapitel dargestellte Gesamtsystem bietet je nach Verfügbarkeit von Kommunikation zwischen den Sensoren (z.B. durch drahtlose Kommunikation via Bluetooth oder WLAN) unterschiedliche Arbeitsweisen. So kann zum einen die Bereitstellung der GPS-Trajektoriepunkte und Kameradetektionen „online“ oder „offline“ erfolgen. Dies bedeutet, dass die Informationen entweder direkt übertragen und in diesem Tracking-Verfahren verarbeitet werden, oder dass die

Berechnung der Trajektorien als Post-processing erfolgt, nachdem die Speicher der Geräte ausgelesen worden sind. In letzterem Fall sind Analysen, die diese Trajektorien als Input benötigen, natürlich ebenfalls nur im Nachhinein und nicht online möglich. Die Funktionsweise dieses Ansatzes bleibt davon jedoch unberührt.

Abseits der oben beschriebenen On- und Offline-Varianten sind auch hinsichtlich des Orts der Verarbeitung der Daten unterschiedliche Systemdesigns möglich. So werden bei einer *zentralen* Arbeitsweise sämtliche Informationen in einer zentralen Instanz gesammelt prozessiert (siehe Abbildung 4.12, links). Diese Instanz ist entweder ein leistungsstarker Sensor, z.B. ein als Kamera genutztes Smartphone, oder eine gesonderte Recheneinheit (Rechner/Server). Das davon abhängige Datenvolumen sorgt für ebenso unterschiedliche Ansprüche an die Übertragungstechnik. Findet keinerlei Vorverarbeitung statt, so werden die Kamerabilder übertragen. Dieses bedeutet (abhängig von der Bildwiederholfrequenz bzw. der gewünschten Sampling-Rate) eine großes Volumen. Werden hingegen diese Bilder bereits lokal analysiert und die Objekte detektiert, so müssen lediglich die entsprechenden Detektionsdaten geschickt werden. Das Datenvolumen ist entsprechend kleiner. Können sogar bereits Trajektoriepunkte durch die Kamera bestimmt werden, kann durch das Versenden der Trajektoriepunkte die benötigte Datenmenge minimiert werden.

Bei einem dezentralen Systemdesign nach Duckham (2012) (Abbildung 4.12, rechts) werden die Kameras und GNSS-Empfänger zu einem Sensornetz zusammengeschlossen, sodass sie Informationen untereinander austauschen können. Dieses ist besonders in dem Fall wichtig, in dem die Daten, die in einem Sensor vorliegenden, für eine Vorverarbeitung eventuell nicht ausreichend sind. In dem Kameranetz kann bspw. ein Objekt, das das Sichtfeld einer Kamera verlassen hatte, zu einem späteren Moment wiedererkannt werden. Zu diesem Zweck müssen die entsprechenden Informationen von den Nachbarkameras bereitgestellt werden. Die notwendigen Daten zur Ermittlung der Objekt-Trajektorien werden schließlich von einer ausgewählten Sensorinstanz eingesammelt und zu einer globalen Lösung weiterverarbeitet. Diese kann durch einen externen Rechner/Server abgerufen werden. Eine derartiges Systemdesign bietet Vorteile wie z.B. einen geringeren Traffic im Sensornetz und eine gewisse Skalierbarkeit bzw. Robustheit des Systems.

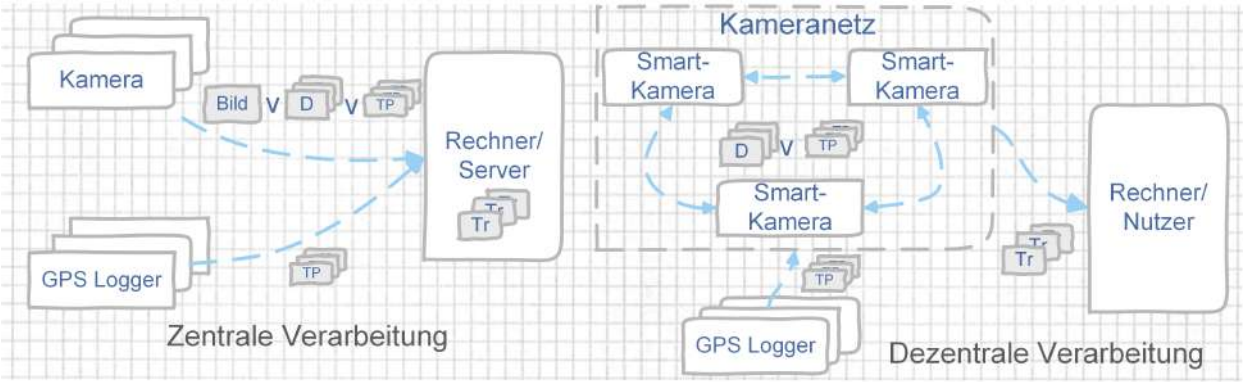


Abbildung 4.12: Gegenüberstellung der zentralen (links) und dezentralen Verarbeitung der Daten (rechts): Bei der zentralen Variante werden sämtliche Daten in Form von Bildern, Detektionen (D) oder Trajektoriepunkten (TP) zum Zielcomputer (Rechner bzw. Server) übertragen und dort zu den Trajektorien verarbeitet. Bei der dezentralen Variante werden lediglich (Zwischen-)Ergebnisse (Detektionen/Trajektoriepunkte) zwischen den vernetzten Smart-Kameras geteilt. Die GPS-Informationen werden in dieses Netz eingeführt, die resultierenden Objekt-Trajektorien an den Nutzer ausgegeben.

5 Mustererkennung in Trajektorien

Bei der Analysen von Trajektorien spielt die Erkennung von Bewegungsmustern eine wichtige Rolle, da die Muster typisches Bewegungsverhalten von Objekten aufzeigen. Diese Informationen können wiederum in verschiedenen Bereichen genutzt werden. In diesem Kapitel wird daher ein Verfahren zur Erkennung von wiederkehrenden a-priori unbekannten Bewegungsmustern in Trajektorien vorgestellt.

5.1 Definition von Bewegungsmustern

Wie aus der Problemstellung dieser Arbeit (siehe Abschnitt 1.2.2) hervorgeht, sollen Bewegungsmuster in Bewegungsdaten identifiziert werden. Ein Bewegungsmuster P entspricht dabei einer Menge an wiederkehrenden ähnlichen Bewegungen S_i. Unter Verwendung der Nomenklatur aus dem Bereich der Sequenzanalyse (siehe Abschnitt 2.3.6), gilt dementsprechend

$$P = \{S_0,S_1,\ldots,S_m\}\,, \tag{5.1}$$

mit m als Anzahl an Bewegungen. Die einzelnen Bewegungen werden auch *Instanzen* des Musters genannt. Je nach Anwendungsszenario gibt es unterschiedliche Anforderungen an ein Bewegungsmuster, anhand derer es in den Daten identifiziert wird. Zum einen sind $suppc_{min}$ (*minimaler support count*) Instanzen erforderlich (vgl. Gleichung 2.58)

$$suppc(P) \geq suppc_{min}. \tag{5.2}$$

Zum anderen müssen sie eine Mindestlänge von

$$l(S_i) \geq l_{min} \tag{5.3}$$

besitzen (vgl. Gleichung 2.59). Nur wenn beide Voraussetzungen erfüllt sind, werden die entsprechenden Instanzen zusammengenommen als ein Bewegungsmuster angesehen.

5.2 Überblick über das entwickelte Mustererkennungsverfahren

Der generelle Ablauf und die einzelnen Schritte des Verfahrens werden in Abbildung 5.1 schematisch dargestellt und in diesem Abschnitt detailliert erläutert.

Der Prozess nutzt Trajektorien als Datengrundlage (5.3), die von einem Tracking-System stammen, das möglicherweise den im vorherigen Kapitel vorgestellten Ansatz zum Objekt-Tracking verwendet. Der Prozess beginnt mit einer Vorverarbeitungsphase (5.4), in der die Trajektorien gemäß der aus dem KDD-Schema bekannten Vorverarbeitungsschritte für die eigentliche Mustererkennung aufbereitet werden. Zur Identifikation der Muster sind zwei unterschiedliche Vorgehensweisen möglich. Während die erste Variante auf einem Clustering von Trajektoriesegmenten (5.5) basiert, arbeitet die Zweite auf Bewegungssequenzen, die aus den Trajektorien generiert werden (5.6). Welcher der beiden Ansätze verwendet wird, hängt dabei vom jeweiligen Anwendungsszenario und der sich daraus ergebenden Problemstellung ab. Handelt es sich bei dem Szenario um die

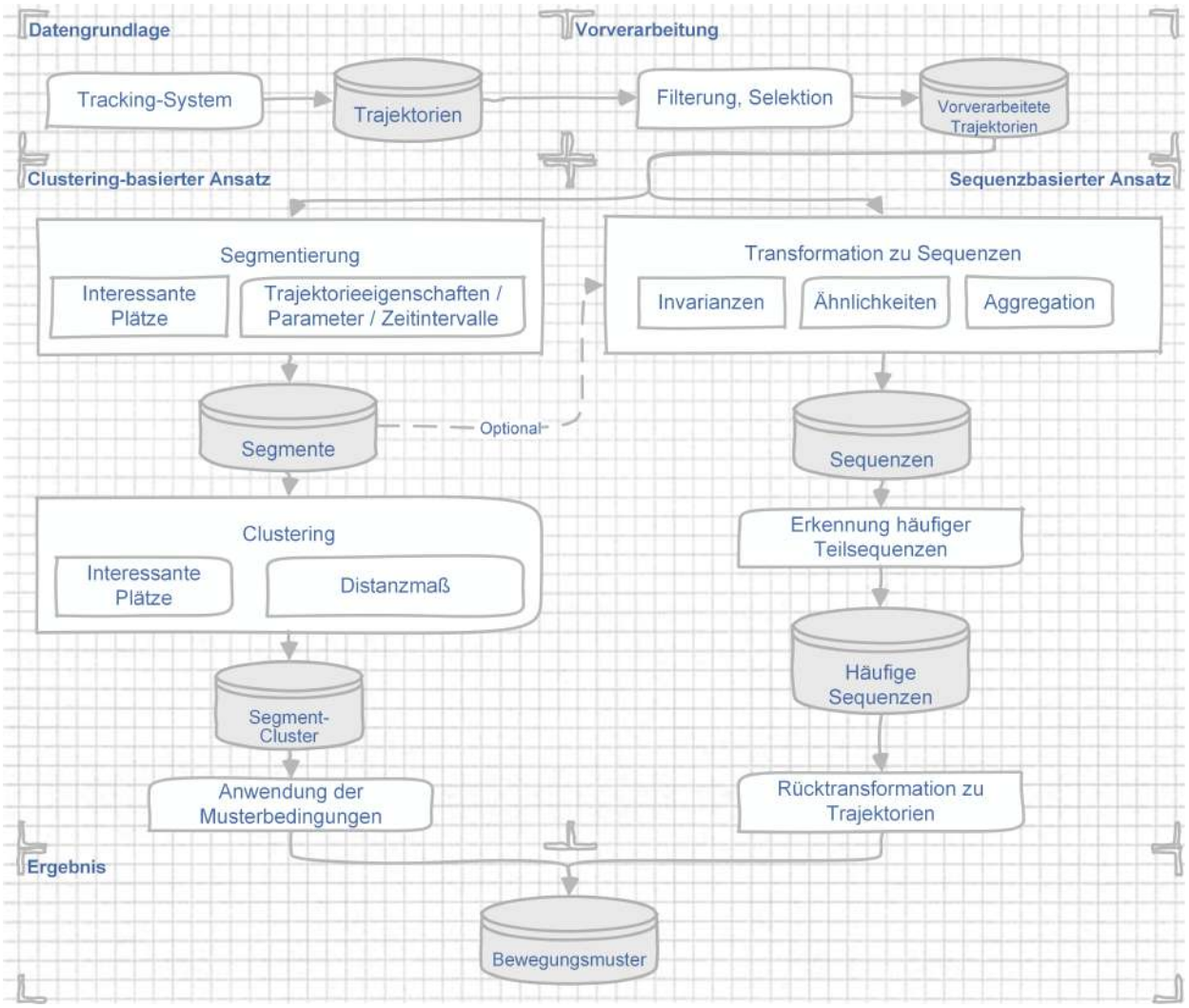


Abbildung 5.1: Übersicht über das Verfahren zur Erkennung und Anwendung von Mustern in Trajektorien

Beobachtung von Objekten über einen längeren Zeitraum, so sind die Trajektorien entsprechend lang. Ein direktes Trajektorie-Clustering macht in diesem Fall keinen Sinn, weil davon auszugehen ist, dass sich, wie in Abschnitt 2.3.3 beschrieben, die (für die Muster) relevanten Teilbereiche nicht über die gesamte Trajektorie erstrecken. Dieser Ansatz erfordert demnach eine Segmentierung, die ein sinnvolles Clustering ermöglicht. Je nachdem, ob und wie viele Informationen bezüglich der Objekte und des Szenarios zur Verfügung stehen, kann die Segmentierung entweder auf einer recht einfachen Auswertung von Bewegungsparametern (z.B. die Änderung der Geschwindigkeit oder der Bewegungsrichtung) oder auf einem gewissen Maß an Vorkenntnis bzw. Kontextinformationen beruhen. Bei Letzterem besteht u.a. die Möglichkeit, auch unter Zuhilfenahme von Lernverfahren, eine *semantische Segmentierung* vorzunehmen. Des Weiteren können zuvor identifizierte relevante Bewegungsinformationen wie bspw. sogenannte *interessante Plätze* für eine Segmentierung herangezogen werden. Eine derartige Vorgehensweise bei der Trajektorie-Segmentierung verleiht den entstehenden Segmenten eine zusätzliche semantische Bedeutung, die dann auch in den resultierenden Bewegungsmustern enthalten ist. Zudem kann der Gesamtdatensatz in unterschiedliche Teildatensätze zerteilt und diese getrennt analysiert werden. Auf diese Weise können kontextabhängige Bewegungsmuster gefunden werden. Ist jedoch kein Kontextwissen für eine sinnvolle Segmentierung verfügbar, oder ist durch die Problemstellung gefordert, sämtliche wiederkehrenden Bewegungsmuster in den Daten zu identifizieren, ist die vorangegangene Variante nicht geeignet. Durch die

Segmentierung besteht dort die Gefahr, dass auch etwaige relevante Bewegungsmuster zerschnitten werden und so nicht mehr gefunden werden können. In diesem Fall wird auf eine alternative sequenzbasierte Variante zurückgegriffen, die gänzlich ohne Segmentierung auskommt. Im Gegensatz zum vorherigen Ansatz besitzen die resultierenden Muster keine semantische Bedeutung. Diese muss durch eine nachfolgende Interpretation der Ergebnisse generiert werden. Unter der Inkaufnahme, nicht alle Muster zu finden, kann aber auch hier vor Beginn der Suche eine kontextabhängige Segmentierung durchgeführt werden, die den zu analysierenden Teilsequenzen wieder eine gewisse Bedeutung zuweist.

Methodisch unterscheiden sich beide Herangehensweisen wie folgt: Beim Clustering-basierten Ansatz werden die Trajektoriesegmente mit Hilfe eines geeigneten Distanzmaßes zu Clustern zusammengefasst. Daraufhin werden die zur Erkennung der Muster geforderten Bedingungen (Gleichungen 5.2 und 5.3) angewendet. Das dabei verwendete Clustering der Trajektorien ist ein bereits gut erforschtes Thema, sodass dort bekannte Methoden angepasst und angewendet werden. In dieser Arbeit liegt der Schwerpunkt daher auf der Segmentierung. Beim sequenzbasierten Ansatz hingegen werden die Trajektorien zu Elementsequenzen transformiert, in welchen dann mit Sequenzanalyseverfahren nach wiederkehrenden Teilsequenzen gesucht wird. Letzterer Ansatz ermöglicht zudem die Analyse von Gruppenbewegungen. Die Erkennung von Gruppenbewegungsmustern unterscheidet sich dabei gegenüber der Einzelbewegungsmustererkennung darin, dass hier die Trajektorien aller Objekte der Gruppe gleichzeitig verarbeitet werden müssen. Zu diesem Zweck werden *Objektkonstellationen* eingeführt, die eben dieses ermöglichen. Das Ergebnis der beiden Ansätze ist jeweils eine Menge an gefundenen Bewegungsmustern, die allerdings auf Grund der unterschiedlichen Herangehensweisen nicht übereinstimmen muss. Die resultierende Mustermenge dient als Grundlage für weitere Analysen bzw. Anwendungen. So kann eine *Bewegungsprädiktion* oder eine *Charakterisierung* eines Objekts anhand seines Bewegungsverhalten auf diese Muster zurückgreifen. Die Ergebnisse dieser Analysen können auch wieder für das Erfassen der Trajektorien, also das Objekt-Tracking, genutzt werden.

5.3 Trajektorien als Datengrundlage

Die Datengrundlage für die Analyse von Objektbewegungen sind deren Trajektorien. Diese unterliegen gewissen Anforderungen hinsichtlich ihrer Länge, Positionsgenauigkeit, Abtastrate und des zugrundeliegenden Bewegungsraums.

Die Länge der Trajektorien ist selbsterklärend ein wichtiger Faktor bei der Analyse. Werden die Bewegungsdaten nur sehr zerstückelt bereitgestellt, also viele kurze Trajektoriesegmente anstelle einer durchgehenden Trajektorie, so fehlen Bewegungsinformationen in den zeitlichen Lücken zwischen den einzelnen Segmenten. Mit der Größe der Lücken und damit Menge an fehlenden Informationen, steigt dementsprechend die Ungenauigkeit der Analyseergebnisse. Im Fall der Mustererkennung als eine mögliche Analyse von Trajektorien bedeutet das, dass Muster lediglich innerhalb der Segmente erkannt werden können und daher maximal die Länge der kurzen Segmente haben können. Längere Bewegungsmuster, die entsprechend mehr Informationen beinhalten und für weitere Anwendungen nützlich sind, könnten unter diesen Bedingungen nicht erkannt werden.

Die Genauigkeit hinsichtlich der Positionen innerhalb der Trajektorien spielt ebenfalls eine wichtige Rolle. Je genauer die Positionen der Objekte erfasst worden sind, desto genauer ist die tatsächliche Bewegung der Objekte modelliert. Ungenauigkeiten bei der Positionierung würden die Bewegungen der Objekte verfälschen. Es könnten zwar weiterhin Muster erkannt werden, diese würden jedoch ebenfalls verfälschte Informationen beinhalten und wären damit für weitere Analysen, z.B. für die Prädiktion der zukünftigen Bewegungen, problematisch.

Abhängig vom Anwendungsfall ist die Sampling-Rate wichtig. Wie in Abschnitt 2.1 beschrieben, bestimmt sie, wie gut die Bewegung durch eine Trajektorie abgebildet werden kann. Eine zu kleine Frequenz bei der Abtastung führt zum Verlust von Details in den Bewegungen. Es ist daher nicht möglich, Muster zu erkennen, die diese Details beinhalten. Allerdings sind es in diversen Anwendungen gerade diese Details, die eine wichtige Rolle spielen. Sollen beispielsweise Bewegungsmuster dazu genutzt werden, Objekte anhand ihrer Bewegungen zu charakterisieren, um sie eventuell später wiedererkennen zu können, so wäre ihr Fehlen natürlich ein Problem. Ist im umgekehrten Fall die Abtastrate höher als benötigt, so wäre das für das Verfahren zur Mustererkennung selbst unproblematisch. Allerdings würde der Berechnungsaufwand entsprechend steigen. Aus diesen Gründen gilt es, eine angemessene Abtastrate festzulegen, sofern diese nicht bereits durch die verwendete Technologie festgelegt ist (siehe Abschnitt 2.2.3). Es muss dabei beachtet werden, dass alle Trajektorien gleich abgetastet sind, um mit den in dieser Arbeit vorgestellten Verfahren sinnvolle Ergebnisse erzielen zu können. Die Wahl ist dabei hauptsächlich von der Dynamik der Objekte, die durch die Geschwindigkeit und die Anzahl an Richtungswechseln gegeben ist, und von dem zugrundeliegenden Bewegungsraum abhängig. Letzter hat wiederum einen Einfluss auf die Bewegungen der Objekte. So sind z.B. die Optionen für Richtungswechsel in einem Netzwerkraum deutlich eingeschränkter als in einem Euklidischer Bewegungsraum.

Die Erfassungsmethode, mit der die Trajektorien generiert worden sind, ist prinzipiell unerheblich. Allerdings sind die jeweiligen Eigenschaften des gewählten bzw. vorgegebenen Tracking-Systems bei der Analyse der Daten zur berücksichtigen. Ein GPS-basiertes Tracking-System liefert beispielsweise Trajektorien versehen mit der GPS-typischen Ungenauigkeit. Das Tracking der Objekte über die Zeit ist dafür aber systembedingt sehr zuverlässig (vgl. Abschnitt 2.2.1). Die für die Analyse bereitstehenden Trajektorien sind daher durchgehend, wenn auch etwas ungenauer als bei anderen Systemen. Beim videobasierten Verfahren verhält es sich umgekehrt (vgl. Abschnitt 2.2.2). Die Genauigkeit mit der die Objekte lokalisiert werden ist deutlich höher, die Zuverlässigkeit des Trackings hingegen geringer. Je nach Szenario ist es möglich, dass vom System anstelle einer langen Trajektorie nur mehrere kürzere aber genauere Trajektoriesegmente pro Objekt geliefert werden. Das in dieser Arbeit vorgestellte GPS-unterstützte Kamera-Tracking (siehe Kapitel 4) kombiniert die positiven Eigenschaften aus beiden zuvor genannten Verfahren und liefert daher eine gute Datengrundlage für die Analyse der Bewegungen und für die Qualität der extrahierten Muster.

5.4 Vorverarbeitung

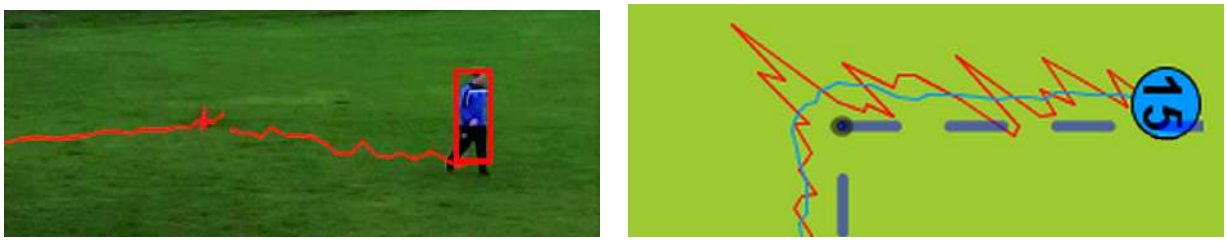
Im Allgemeinen ist vor Beginn der eigentlichen Bewegungsmustererkennung eine Vorverarbeitung nötig. Diese beinhaltet die im KDD-Prozess beschriebenen Vorverarbeitungsschritte Datenbereinigung, -integration, -selektion und -transformation (siehe Abschnitt 2.3.1). Diese Schritte werden im Folgenden beschrieben.

5.4.1 Datenbereinigung

Abhängig von der Objekt-Tracking-Lösung sind die Trajektorien, die als Datengrundlage verwendet werden, fehlerbehaftet. Der Fehler besteht dabei aus einer systematischen als auch aus einer zufälligen Abweichung. Systematische Fehler können durch eine ungenaue Kalibrierung der Sensoren oder durch schlechte Messbedingungen hervorgerufen werden, z.B. nicht-kalibrierte Kameras oder ungünstige Perspektiven beim videobasierten Tracking. Sie sind prädizierbar und über der Beobachtungszeit hinweg konstant. Sie können beispielsweise durch eine Transformation der Daten beseitigt werden. Zufällige Fehler hingegen sind unvorhersehbar und variabel. Abgesehen von

Ausreißern bzw. groben Fehlern können mittelbildende Methoden Mehrfachbeobachtungen nutzen, um diese Art Fehler zu verringern.

Mit dem hier beschriebenen Ansatz können Trajektorien aus unterschiedlichen Tracking-Lösungen verarbeitet werden. Dementsprechend unterschiedlich können auch die Fehler in den Objektpositionen sein. Während GPS-Trajektorien die typischen GPS-Abweichungen aufzeigen, beinhalten die Trajektorien aus einem Video-Tracking mehr zufällige Fehler. Diese sind oft durch Objektverdeckungen oder eine fehlerhafte Ermittlung der Objektsilhouette, was ebenfalls zu einem Fehler bei der Objektlokalisierung führt (Abbildung 5.2a), bedingt. Wird das ebenfalls in dieser Arbeit vorgestellte GPS unterstützte Video-Tracking verwendet, dann entsprechen die Abweichungen, dem Lokalisierungsfehler, der beim Video-Tracking auftreten kann. Die Lokalisierung der Objekte erfolgt nämlich, soweit möglich, durch das Video-Tracking. Die Fehler sind dann, wie oben beschrieben, eher zufällig und können bspw. durch Filterung reduziert werden. Neben dem Kalman Filter, welcher sicherlich eine gute Wahl für diese Aufgabe ist, liefert auch ein einfacher (zentrierter) Mittelwert-Filter, wie er in Abschnitt 2.3.2 beschrieben wird, oft hinreichend gute Resultate (Abbildung 5.2b). Zur Beseitigung der systematischen Ungenauigkeiten werden entsprechende Transformationen genutzt.



(a) *(b)*

Abbildung 5.2: Ungenauigkeiten in der Objektlokalisierung auf Grund einer fehlerhaften Bounding-Box-Ermittlung. a) Die berechnete Bounding-Box passt nicht exakt zum Objekt. b) Die resultierende Trajektorie (in rot) ist mit signifikanten Ungenauigkeiten versehen, welche durch Filterung/Glättung verringert werden können (in blau).

5.4.2 Datenselektion

Für die Analyse der Objektbewegungen sind nicht zwangsläufig alle vom Tracking-System bereitgestellten Daten relevant. Zur Verringerung des Datenvolumens und damit des Berechnungsaufwands und um nicht repräsentatives Verhalten der zu beobachtenden Objekte auszuschließen, kann eine räumliche und/oder zeitliche Selektion der Daten stattfinden. Dabei werden mögliche Ausreißer in Form von untypischem Bewegungsverhalten (die Objekte bewegen sich zu besonderen Zeiten eventuell nicht gewöhnlich) eliminiert und finden bei der späteren Analyse keine Berücksichtigung mehr.

An verschiedenen Beispielen soll dies näher erläutert werden. Bei der Analyse von Trajektorien aus dem Kontext Verkehr handelt es sich um eventuell irrelevante Daten, wenn diese aus irrelevanten räumlichen Bereichen stammen. Bspw. können die Daten aus den Außenbereichen der analysierten Umgebung sein, in denen die Daten nur in einer sehr niedrigen Dichte vorliegen, d.h. die entsprechenden Straßen wurden vielleicht nur einmal passiert, und sind daher nicht repräsentativ für das Bewegungsverhalten des Verkehrsteilnehmers (siehe 5.3a). Des Weiteren können Daten vernachlässigt werden, wenn sie aus irrelevanten Zeitintervallen (z.B. bestimmte Tageszeiten oder Jahreszeiten) stammen.

(a) (b)

Abbildung 5.3: Selektion relevanter Daten: (a) In den Außenbereichen des Beobachtungsgebiets (rot eingerahmt) weisen die Daten nur niedrige Dichten auf. Manche Straßen sind nur sehr wenig befahren. Repräsentatives Objektverhalten kann dort nur schwierig ermittelt werden. (b) Es sind deutliche Unterschiede im Bewegungsverhalten von Fußballspielern während des Spiels (schwarze Trajektorie) und vor/nach dem Spiel bzw. in dessen Pause (rot) zu erkennen. Hier hält sich der Spieler zu den irrelevanten Zeiten in ganz anderen Bereichen des Spielfelds auf.

Bei der Sportanalyse gibt es ebenfalls räumliche und zeitliche Kriterien, die relevante von irrelevanten Daten unterscheiden. Hier sind bspw. Daten nicht von Interesse, die in irrelevanten Zeitintervallen (vor dem Wettkampf, nach dem Wettkampf, während Unterbrechungen) generiert wurden. In diesen Zeiträumen können die Bewegungen der Sportler stark vom typischen Bewegungsverhalten während des Wettkampfs abweichen (siehe Abbildung 5.3b). Für die Fußballanalyse bedeutet dies explizit, dass Daten vor dem Spiel, nach dem Spiel und in der Halbzeitpause nicht berücksichtigt werden. Räumliche Kriterien trennen die Daten anhand von speziellen Bereichen des überwachten Raums. Diese Bereiche hängen von der speziellen Analyse ab. Soll lediglich das Offensivverhalten untersucht werden, sind andere Bereiche relevanter als bei der Analyse von Defensivbewegungen.

Bei den oben beschriebenen zeitlichen Kriterien handelt es sich um Zeitspannen zwischen von vornherein bekannten Zeitpunkten. Der Start und das Ende eines Wettkampfs sind in der Regel bekannt. Gleiches gilt für Tages- bzw. Jahreszeiten. Jedoch können Daten auch anhand von vorliegenden Aktionen bzw. Verhaltensweisen der Objekte selektiert werden. Im Fall der Fußballanalyse bedeutet dies, dass eventuell nur die Intervalle relevant sind, in denen eine gewisse Situation vorliegt. Es sollen bspw. typische Angriffsmuster bzw. Laufwege der Spieler in Offensivsituationen ermittelt werden. Dabei sind dementsprechend die Zeiten, in denen sich das Team oder der Spieler in der Defensive befindet, irrelevant und können vernachlässigt werden. Da diese Zeitpunkte, an denen diese Aktionen oder Verhaltensweisen beginnen oder sich ändern, jedoch nicht von vornherein bekannt sind und im Allgemeinen auch nur selten Regelmäßigkeiten in Bezug auf den Startzeitpunkt und die Dauer besitzen, müssen die relevanten Zeitintervalle zuerst bestimmt werden. Dazu sind Informationen über die Verhaltensweisen der Objekte in derartigen Situationen nötig, die genutzt werden können, um die entsprechenden Intervalle bspw. mit Methoden des Maschinellen Lernens automatisch bestimmen zu können. Eine detaillierte Beschreibung der Vorgehensweise zur Bestimmung der Zeitintervalle ist in Abschnitt 5.5.1 gegeben, da dort dieselbe Methodik zur Segmentierung der Trajektorien genutzt wird.

5.4.3 Datenintegration und -transformation

Der hier beschriebene Prozess zur Erkennung von Bewegungsmustern verwendet lediglich Trajektorien als Input, die die gleiche Struktur besitzen und in der gleichen Art und Weise in einer gemeinsamen Datenbank gespeichert sind. Daher ist keine Integration von heterogenen Daten unterschiedlicher Quellen nötig.

Zur Erfassung einer der Datensätze, der im späteren Verlauf dieser Arbeit (siehe Abschnitt 7.5.1) vorgestellt wird, wurden die Spieler einer Mannschaft über mehrere Spiele hinweg begleitet. Die Spiele fanden dabei auf unterschiedlichen Sportstätten statt. Für eine spielübergreifende Analyse ist es daher notwendig, die Trajektorien in ein gemeinsames lokales Spielfeldkoordinatensystem zu transformieren. Dieses erleichtert zudem die Handhabung und Visualisierung. Die Schwierigkeit besteht jedoch darin, dass es für die Ausmaße von Fußballfeldern zwar eine Norm gibt, diese aber nur gültige Bereiche (90 - 120 m lang, 45 - 60 m breit) festlegt. Die Felder sind demnach nicht gleich groß, was bei der Transformation der Trajektorien in ein gemeinsames Koordinatensystem zu Problemen führen kann, da die Trajektorien nicht skaliert werden dürfen, da sich sonst die Geschwindigkeiten der Objekte verändern würden. Für die Transformation wurde daher der Spielfeldmittelpunkt genutzt. Dies hat allerdings zur Folge, dass Spieler, die exakt der Seitenauslinie eines kleineren Felds entlang gelaufen sind, sich im Nachhinein im Inneren bewegen. Der Einfluss auf das Bewegungsverhalten wird im Rahmen dieser Arbeit jedoch nicht untersucht.

5.5 Mustererkennung: Clustering-basierter Ansatz

Im folgenden Abschnitt wird die erste Variante zur Erkennung von Bewegungsmustern beschrieben. Um die wiederkehrenden Bewegungen zu erkennen, wird auf ein Clustering von Trajektorien zurückgegriffen. Auf Grund der zuvor beschriebenen Tatsache, dass sich die Muster im Regelfall nicht über die gesamte Trajektorie erstrecken, müssen zuerst die relevanten Bereiche in den Trajektorien ermittelt werden. Dieses geschieht mit Hilfe einer Segmentierung, in der die Trajektorien in Trajektoriesegmente unterteilt werden (siehe Abschnitt 5.5.1). Die Segmente werden daraufhin mittels eines Clustering-Verfahrens und einem geeignetem Distanzmaß zu Clustern zusammengefasst (5.5.2), die schließlich die Instanzen der Bewegungsmuster darstellen.

5.5.1 Segmentierung der Trajektorien

Bei der Segmentierung der Trajektorien unterscheidet sich das Vorgehen je nach Verfügbarkeit von Kontextinformationen, also von zusätzlichem Wissen über das Szenario und das typische Bewegungsverhalten der beobachteten Objekte. Liegen diesbezüglich keine Informationen vor, kann die Zerlegung nur anhand der gegebenen Trajektorie-Eigenschaften, wie bspw. ihrer Geometrie oder der Bewegungsparameter (z.B. Bewegungsrichtung, -geschwindigkeit, -beschleunigung), erfolgen. Hier können auch die in den Grundlagen beschriebenen existierenden Ansätze (siehe Abschnitt 2.3.3) verwendet werden. Die resultierenden Trajektoriesegmente besitzen dann jedoch im Regelfall keine semantische Bedeutung. Es ist demnach nicht möglich, einem Segment eine Intention für die Objektbewegung zuzuordnen. Die Schnittpunkte können mitten in Aktionen bzw. in speziellen Verhaltensweisen liegen, sodass diese sich im Nachhinein über unterschiedliche Segmente erstrecken.

Semantische Segmentierung

Für den Fall, dass Informationen über das typische Bewegungsverhalten der Objekte und so auch über die Zeitpunkte, an denen sich die Intention eines Objekts ändert, verfügbar sind, können die

Trajektorien an diesen Stellen zerteilt werden. Die entstehenden Segmente beinhalten dann jeweils eine spezielle Aktion bzw. Verhaltensweise und besitzen daher auch eine semantische Bedeutung. Häufig liegen diese Information jedoch nicht vor. Um trotzdem eine kontextbasierte Segmentierung vornehmen zu können, besteht die Möglichkeit die benötigten Informationen über die Intentionen zu generieren. Zu diesem Zweck können Verfahren des Maschinellen Lernens, genauer gesagt Klassifikationsverfahren, herangezogen werden. Mit Hilfe dieser Verfahren und entsprechenden Trainingsdaten, kann den Bewegungen bzw. den Trajektoriepunkten eine gewisse Bedeutung zugewiesen werden. In dieser Arbeit wird der *ID3-Algorithmus* verwendet, der in der Lernphase anhand der Trainingsdaten und der gegebenen Features einen Entscheidungsbaum generiert. Dieser kann daraufhin auf die vorliegenden Bewegungsdaten (Testdaten) angewendet werden, um den Bewegungen zu jedem Zeitpunkt ein Label zuzuweisen. Die Labels entsprechen der Intention der Objekte und müssen in den Trainingsdaten vorliegen. Die sich ergebende Klassifikation der Bewegungsdaten wird nun dazu genutzt, die Zeitpunkte zu identifizieren, an denen sich die Intention des Objekts ändert. Diese Zeitpunkte entsprechen den Schnittpunkten, an denen die Trajektorien zerteilt werden.

Am Beispiel der Fußballanalyse wird im Folgenden das Vorgehen zur Klassifizierung der Bewegungsdaten näher beschrieben. Die Unterteilung der Klassen kann dabei unterschiedlich sein und hängt von der Analyseaufgabe ab. Soll bspw. nur das Offensiv- bzw. Defensivverhalten der Spieler untersucht werden, so sind die Klassen „Offensiv“ und „Defensiv“ ausreichend. Soll jedoch das Verhalten bei speziellen Aktionen der Spieler evaluiert werden, so können die Klassen z.B. „Freistoß“, „Eckstoß“, „Einwurf“, „Torschuss“, usw. sein. Für die Einteilung der Bewegungsdaten in Klassen werden entsprechende Features benötigt. Zur Ermittlung dieser Features werden sämtliche bereits zur Verfügung stehenden Informationen, wie z.B. die Positionen und evtl. die aktuellen Geschwindigkeiten sowie Bewegungsrichtungen der Spieler, zusammen mit weiteren berechneten Features, wie bspw. die Abstände zu den Toren bzw. Eckpunkten des Spielfelds, die Ausmaße des Begrenzungsrechtecks des gesamten Teams (exkl. Torhüter) und Ähnliches, bezüglich ihres Informationsgehalts (Gleichung 2.51) untersucht. Features, die keinen relevanten Beitrag für die Klassifizierung liefern, werden nicht weiter berücksichtigt. Die Restlichen werden vom ID3-Algorithmus zur Generierung des Entscheidungsbaums verwendet. Tabelle 5.1 zeigt die verwendeten Features für die Offensiv-Defensiv-Analyse inklusive des jeweiligen Informationsgehalts, der im Zusammenhang mit dem ID3-Algorithmus berechnet wird. Zu beachten ist dabei, dass die räumlichen Features normiert werden, um so Problemen mit unterschiedlichen Spielfeldgrößen oder Spielrichtungen vorzubeugen.

Tabelle 5.1: Zur Klassifizierung von Offensiv- und Defensivverhalten eines Teams verwendete Features absteigend geordnet nach ihrem Informationsgehalt [Bit]. Je höher dieser ist, desto hilfreicher ist das Feature im Zusammenhang mit der Klassifikation.

Feature	**Beschreibung**	**Informationsgehalt [Bit]**
x_{min}	min. x-Koordinate aller Objekte	0.519
x_{mean}	mittlere x-Koordinate aller Objekte	0.502
x_{max}	max. x-Koordinate aller Objekte	0.446
y_{min}	min. y-Koordinate aller Objekte	0.187
y_{max}	max. y-Koordinate aller Objekte	0.170
y_{mean}	mittlere y-Koordinate aller Objekte	0.134
v_{mean}	mittlere Geschwindigkeit aller Objekte	0.066
h_{mean}	mittlere Bewegungsrichtung aller Objekte	0.049

Zum Lernen des Klassifikationsmodells sind manuell gelabelte Trainingsdaten im gesamten Umfang von ungefähr acht Minuten, insgesamt 2345 Bewegungen eines Teams, verwendet worden. Zur Evaluation wird dieser Datensatz zwecks Training und Test halbiert. Es werden 1147 Samples richtig, 25 falsch klassifiziert. Daher ergeben sich die folgenden Kennzahlen: Sensitivität (*recall*)

= 0,98, Genauigkeit (*precision*) = 0,98. Andere Klassifikationsverfahren, wie bspw. *Neuronale Netze*, *Random Forrests*, usw., liefern in diesem Fall sehr ähnliche Ergebnisse, was auch ein Grund ist, warum hier der anschauliche ID3-Algorithmus genutzt wird. Werden die Analysen komplexer, d.h. es existieren mehr als zwei Klassen, können durchaus andere Algorithmen bessere Ergebnisse erzielen. In Abbildung 5.4 werden beispielhaft die Ergebnisse der Klassifikation eines Spiels und ausgewählte Situationen gezeigt, die das Klassifikationsergebnis widerspiegeln.

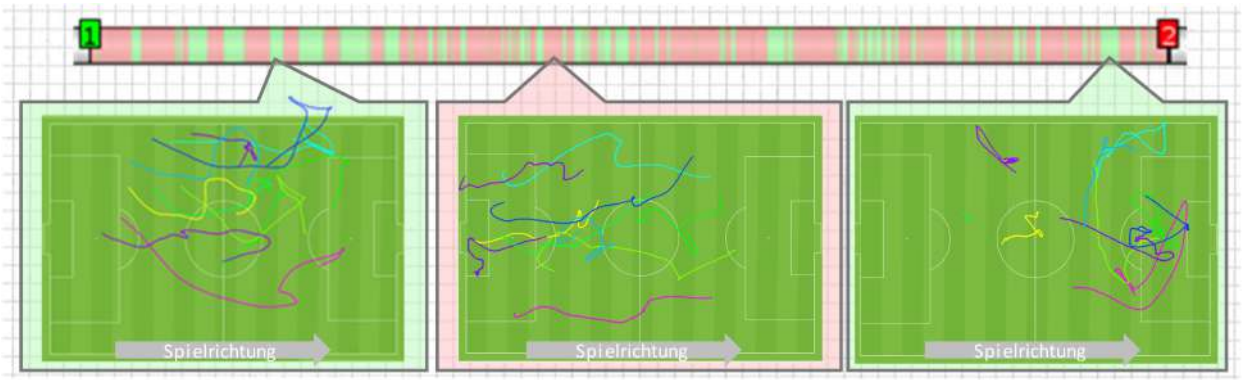


Abbildung 5.4: Oben: Das Ergebnis der Klassifikation des Verhaltens eines Teams über ein Spiel in einer Zeitleiste dargestellt (grün = offensiv, rot = defensiv). Unten: Ausgewählte Intervalle und die Trajektorien der Spieler in dieser Zeit. Letztere lassen erkennen, dass das klassifizierte und tatsächliche Verhalten in der Regel übereinstimmt. Die allgemeine Bewegungsrichtung der Spieler im ersten und dritten Beispiel ist gleich der Spielrichtung. Im zweiten, defensiven Beispiel ist sie entgegengesetzt zu dieser.

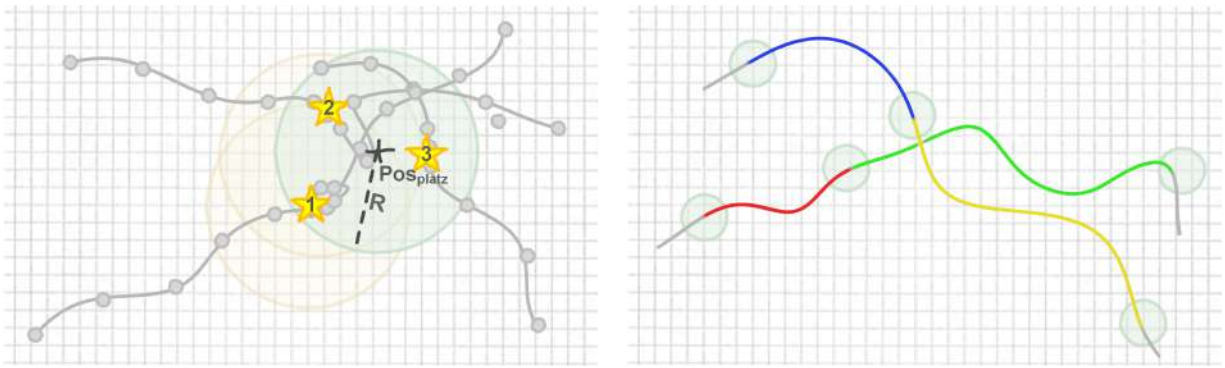
Interessante Plätze.

Eine andere Möglichkeit zur Segmentierung basiert auf sogenannten interessanten Plätzen, die den entstehenden Segmenten eine gewisse Bedeutung verleihen, da sie deren Anfangs- bzw. Endpunkt darstellen. Interessante Plätze werden dabei noch einmal in *attraktive Plätze* und Plätze, an denen Objekte häufig das Beobachtungsgebiet *verlassen* bzw. *betreten* unterschieden. Attraktive Plätze sind Orte, die auf Objekte eine gewisse Attraktivität ausüben. Dies kann bspw. eine Sehenswürdigkeit, ein Schaufenster in der Einkaufsstraße, eine Haltestelle für öffentliche Verkehrsmittel oder, wenn es sich bei den Objekten um Tiere handelt, eine gute Futterstelle sein. Dies alles sind Plätze, die von den Objekten häufig aufgesucht werden und an denen sie sich entweder langsamer bewegen, um sich z.B. etwas anzuschauen oder etwas aufzunehmen, oder sogar ganz stehen bleiben. Dabei spielt es keine Rolle, ob es immer das gleiche oder ein anderes Objekt ist. Die Begrifflichkeit „Attraktiver Platz“ unterscheidet sich vom sogenannten *Point of Interest* (*POI*) in der Weise, dass Letzterer von vornherein bekannt ist und einer Karte der Umgebung entnommen werden kann. Ein Attraktiver Platz hingegen ist nicht zwingend in einer Karte enthalten, sondern wird erst im Laufe der Analyse ermittelt. Es könnte auch eine beliebige Zone auf einer offenen Fläche (bspw. eines Bahnhofsvorplatzes) sein, der auf gewisse Weise eine Attraktivität auf die Objekte ausübt. Bei der anderen Art von interessanten Plätzen handelt es sich um Orte, an denen die beobachteten Objekte häufig das erste bzw. letzte Mal von dem verwendeten Sensor erfasst werden. Bei einer Kameraüberwachung können dies u.a. die Stellen sein, an denen Objekte häufig das Sichtfeld der Kamera betreten oder verlassen. In diesem Fall liegen diese Plätze häufig an den Rändern. Es kann aber auch eine sich im Sichtfeld befindende Tür sein, durch die die Objekte eine Gebäude betreten oder verlassen können. Im Fall einer GPS-Überwachung verhält es sich ähnlich. Dort sind es die Orte, an denen die Trajektorie der Objekte startet oder endet. Im Allgemeinen sind das die Stellen, an denen Objekte sich nicht mehr unter freiem Himmel bewegen oder der Empfänger ein- bzw. ausgeschaltet wird.

Folgende Annahmen werden für die Detektion interessanter Plätze getroffen:

- Ein interessanter Platz wird durch eine *geometrische Form* (typischerweise einen *Kreis*) begrenzt.
- Er besitzt eine gewisse Ausdehnung (für Kreis ist dies der Radius R).
- Er finden innerhalb dieser Zone N_{IP} Ereignisse statt.

Unter Ereignis ist das relevante Verhalten zu verstehen. Bei attraktiven Plätzen ist dies die Verringerung der Geschwindigkeit $v \leq v_{max}$, bei den anderen Plätzen die Tatsache, dass ein Objekt das erste oder letzte Mal detektiert wird. Unter Berücksichtigung dieser Annahmen funktioniert die Detektion wie folgt (siehe Abbildung 5.5a): In jedem Zeitschritt werden die Bewegungen der Objekte untersucht, ob dort eines der Ereignisse stattfindet. Ist dies der Fall, wird an dieser Stelle (Pos_{platz}) ein Kandidat für einen interessanten Platz erstellt. Findet das Ereignis (evt) innerhalb der Zone eines bereits existierenden Kandidaten oder interessanten Platzes statt, d.h. $d(Pos_{platz}, Pos_{evt}) \leq R$, so wird dessen Ereigniszähler n_evt entsprechend erhöht. Erreicht der Zähler eines Kandidaten die erforderlichen N_{IP} Ereignisse, d.h. es gilt $n_{evt} \geq N_{IP}$, so wird aus ihm ein neu gefundener interessanter Platz. Die Koordinaten des Platzes werden bei jedem neuen Ereignis korrigiert, indem das Mittel aller Ereigniskoordinaten gebildet wird. Für die jeweilige Art von interessanten Platz wird jedoch nur die entsprechende Art von Ereignis gezählt.



(a) *(b)*

Abbildung 5.5: a) Die Detektion interessanter Plätze erfolgt auf Basis von Ereignissen (gelbe Sternsymbole, Nummer gibt die Reihenfolge an), die durch ein gewisses Objektverhalten ausgelöst werden. Das erste Ereignis veranlasst die Erzeugung eines Kandidaten (gelbe Kreise). Wird die benötigte Anzahl an Ereignissen (hier $N_{IP} = 3$) erreicht, wird der Kandidat zu einem interessanten Platz (grün). Die Postion des Platzes wird bei jedem neuen Ereignis angepasst. In grau werden die Trajektorien und die entsprechenden Trajektoriepunkte dargestellt. Kurze Abstände zwischen den Punkte symbolisieren geringe Geschwindigkeiten. b) Bei der Segmentierung werden die Trajektorien bei Schnitt mit den interessanten Plätzen in Segmente (unterschiedliche Farben) geteilt.

Diese auf diese Weise ermittelten Plätze können zur Segmentierung der Trajektorien genutzt werden. Die Trajektorien werden in diesem Fall an den Stellen zerteilt, an denen sie einen Platz schneiden (siehe Abbildung 5.5b). Sie können aber auch zur Generierung eines Bewegungsmodells verwendet werden, das als Grundlage für weitere Analysen, z.B. der Bewegungsprädiktion, dient. Eine ausführliche Beschreibung hierzu und auch zur Detektion Interessanter Plätze ist in Feuerhake u. a. (2011) und Feuerhake (2012) gegeben.

Jeder der zuvor beschriebenen Wege zur Segmentierung der Trajektorien liefert eine Menge an Segmenten, die im nächsten Schritt verwendet werden.

5.5.2 Clustering der Trajektorien

Zur Bestimmung ähnlicher Trajektoriesegmente wird eine geeignete Kombination aus relevanten Merkmalen, Distanzmaß (siehe Abschnitt 2.3.4) und Clustering-Verfahren (2.3.5) genutzt. Mit Hilfe der Merkmale können die benötigten Invarianzen der Muster festgelegt werden. Werden die Objektpositionen, d.h. die x- und y-Koordinaten, als Merkmale verwendet, so bedeutet dies, dass zwei Segmente nur dann als ähnlich angesehen werden, wenn sie räumlich entsprechend kleine Abweichungen aufweisen. Sie besitzen demnach keine Invarianzen. Werden hingegen Bewegungsvektoren $[dx\ dy]^T$, also die Änderung in x und y, oder die Bewegungsrichtung ϕ in Kombination mit der zurückgelegten Distanz r genutzt, so dürfen die Segmente verschoben sein. Ein entsprechendes Muster ist demnach translationsinvariant. Bei der Verwendung der Änderung der Bewegungsrichtung $d\phi$ und r, ist neben der Translation auch eine beliebige Rotation erlaubt. Tabelle 5.2 gibt diesbezüglich eine Übersicht.

Tabelle 5.2: Die unterschiedlichen Anforderungen an die Invarianzen der Muster werden durch die entsprechend berücksichtigten Merkmale ermöglicht (x,y: Koordinaten, ϕ: Bewegungsrichtung (heading), r: Länge des Bewegungsvektors)

Invarianzen	**Merkmale**
Keine (K)	$[x\ y]^T$ (5.4)
Translation (T)	$[dx\ dy]^T$ (5.5) oder $[\phi\ r]^T$ (5.6)
Translation & Rotation (T+R)	$[d\phi\ r]^T$ (5.7)

Abhängig von den Voraussetzungen an die Daten und der Berücksichtigung der Bewegungsrichtung der Trajektorie bieten sich unterschiedliche Distanzmaße an. So verlangt bspw. die *Euklidische Distanz* identisch viele Punkte auf den Segmenten. Die *Hausdorff-Distanz* wiederum ignoriert die Bewegungsrichtung. Eine generelle Lösung ist dabei schwierig anzugeben, da die Eignung sehr von der Problemstellung abhängt. Dieses gilt auch bei der Wahl des Clustering-Verfahrens. Wird das *DBSCAN*-Verfahren verwendet, das ein Clustering auf Basis der Trajektorien-Dichte durchführt, ermöglichen die Parameter ϵ und $minIts$ eine Steuerung der Ähnlichkeit der Segmente, die erforderlich ist, damit diese dem gleichen Cluster zugewiesen werden. Unter der Annahme, dass $minIts$ sich nicht ändert, gilt dabei: je kleiner ϵ gewählt wird, desto ähnlicher müssen sich die Segmente sein, um dem gleichen Cluster zugewiesen zu werden. Dies bedeutet, es werden im Allgemeinen mehr Cluster mit jedoch weniger Segmenten gefunden. Dementsprechend höher ist auch die Zahl der potenziell erkannten Muster, da diese auf Basis der Cluster und den darin enthaltenen Segmenten bestimmt werden. Die Ähnlichkeit der Instanzen innerhalb eines Muster steigt mit fallendem ϵ. Bei steigendem ϵ verhält es sich umgekehrt. Die Größe der Cluster steigt, die Anzahl der Cluster und damit der potenziellen Muster sinkt ebenso wie die Ähnlichkeit der Instanzen innerhalb eines Musters. Hierbei ist zudem wichtig, dass Informationen über die typische Dichte in den Daten vorhanden sein müssen, sodass ϵ sinnvoll gewählt werden kann. Des Weiteren sind nicht alle Kombinationen aus DBSCAN-Verfahren und den beschriebenen Distanzmaßen problemlos anwendbar. Beispielhaft hierfür ist die Kombination mit einem Distanzmaß, bei dem keine repräsentative, sondern eine aufsummierte Distanz berechnet wird. Dies ist beim *Dynamic Time Warping* oder bei der *Euklidischen Distanz* (die nach Gleichung 2.36 berechnet wird) der Fall. Dort ist die Bestimmung eines sinnvollen Werts für den Parameter ϵ problematisch.

Wird hingegen das *k-means*-Verfahren benutzt, um die Trajektorien zu clustern, gibt es bei der Wahl des Distanzmaßes keine Einschränkungen. Lediglich die Eigenschaft der Hausdorff-Distanz, dass die Bewegungsrichtung nicht berücksichtigt wird, muss in Betracht gezogen werden. Dort ermöglicht der Parameter k eine Steuerung der benötigten Segmentähnlichkeit. Je höher k und damit die Zahl der resultierenden Cluster gewählt wird, desto ähnlicher müssen sich die Segmente

sein, um dem gleichen Cluster anzugehören. Die Zahl der Segmente pro Cluster wird demnach geringer. Dies bedeutet, dass je nach Anforderungen mehr unterschiedliche Muster mit jedoch weniger Instanzen identifiziert werden. Bei hohen Anforderungen in Bezug auf die Anzahl an Instanzen, besteht jedoch die Gefahr, dass insgesamt weniger Muster identifiziert werden können. Wird k kleiner gewählt, verhält es sich umgekehrt. Dies hat zur Folge, dass die Instanzen eines Musters sich nicht mehr so ähnlich sein müssen.

Die Cluster-Analyse liefert eine Menge an Clustern, die den potenziellen Bewegungsmustern samt ihrer Instanzen entspricht. Nach einer Filterung anhand der geforderten Bedingungen (vgl. Gleichungen 5.2 und 5.3) entspricht sie der Ergebnismenge.

5.6 Mustererkennung: Sequenzbasierter Ansatz

In diesem Abschnitt wird mit dem sequenzbasierten Ansatz eine alternative Variante zur Erkennung wiederkehrender Bewegungsmuster vorgestellt. Diese erfordert keine vorangegangene Segmentierung und basiert auf der Identifikation von wiederkehrenden Teilsequenzen in der Sequenz von Objektbewegungen (Feuerhake, 2016). Letztere wird auf Grundlage der vorverarbeiteten Trajektorien generiert. Im Gegensatz zur Clustering-basierten Varianten ist dieser Ansatz in der Lage, sowohl Bewegungsmuster eines einzelnen Objekts (*individuelle Muster*) als auch die einer Gruppe von Objekten (*Gruppenmuster*) zu erkennen (Feuerhake und Sester, 2013). Bei Letzterem wird allerdings eine konstante Mitgliederzahl der Gruppe vorausgesetzt, sodass in der zu analysierenden Zeitspanne Objekte weder die Gruppe verlassen noch sich dieser anschließen dürfen.

Wie in Abbildung 5.6 gezeigt wird, besteht auch diese Variante aus mehreren aufeinanderfolgenden Teilschritten. Diese sind für die Erkennung von Einzel- bzw. Gruppenbewegungsmustern identisch. Lediglich die Repräsentation der Bewegungen in den einzelnen Sequenzelementen unterscheidet sich. Dieses wird in den folgenden Abschnitten detailliert erläutert.

5.6.1 Eingangsdaten

Allgemein nutzt dieser Mustererkennungsansatz die vorverarbeiteten Trajektorien als Input. Je nach Anwendungsfall, in dem entweder individuelle Muster oder Gruppenmuster gesucht werden, ändert sich der Input jedoch. Während für eine Einzelanalyse die Trajektorien der Objekte individuell untersucht werden, werden zur Erkennung von Gruppenmustern alle Trajektorien der Gruppenmitglieder gleichzeitig analysiert. Zu diesem Zweck werden *Konstellationen* genutzt, die die Positionen der Objekte zu einem Zeitpunkt entweder absolut (z.B. anhand von Koordinaten) oder relativ zueinander (z.B. durch Distanzen, Winkel, usw.) beschreiben. Eine Konstellation wird je nach Auswahl durch $\binom{n_{grp}}{2}$ Positionsrelationen beschrieben, wobei n_{grp} die Anzahl der Gruppenmitglieder ist.

5.6.2 Generierung der Sequenzen aus Bewegungen

Dieser Ansatz basiert auf der Transformation der Trajektorien in eine Sequenz von Bewegungen S_{tot}, die mit Hilfe von Sequenzanalysemethoden auf wiederkehrende Teilsequenzen untersucht werden kann. Diese Sequenz erstreckt sich über die gesamte Beobachtungszeit hinweg und besteht aus den Sequenzelementen I_i, die jeweils Informationen über die Objektbewegungen in einem Zeitschritt beinhalten.

$$S_{tot} = \{I_0, I_1, \ldots, I_{n_{tot}}\} \tag{5.8}$$

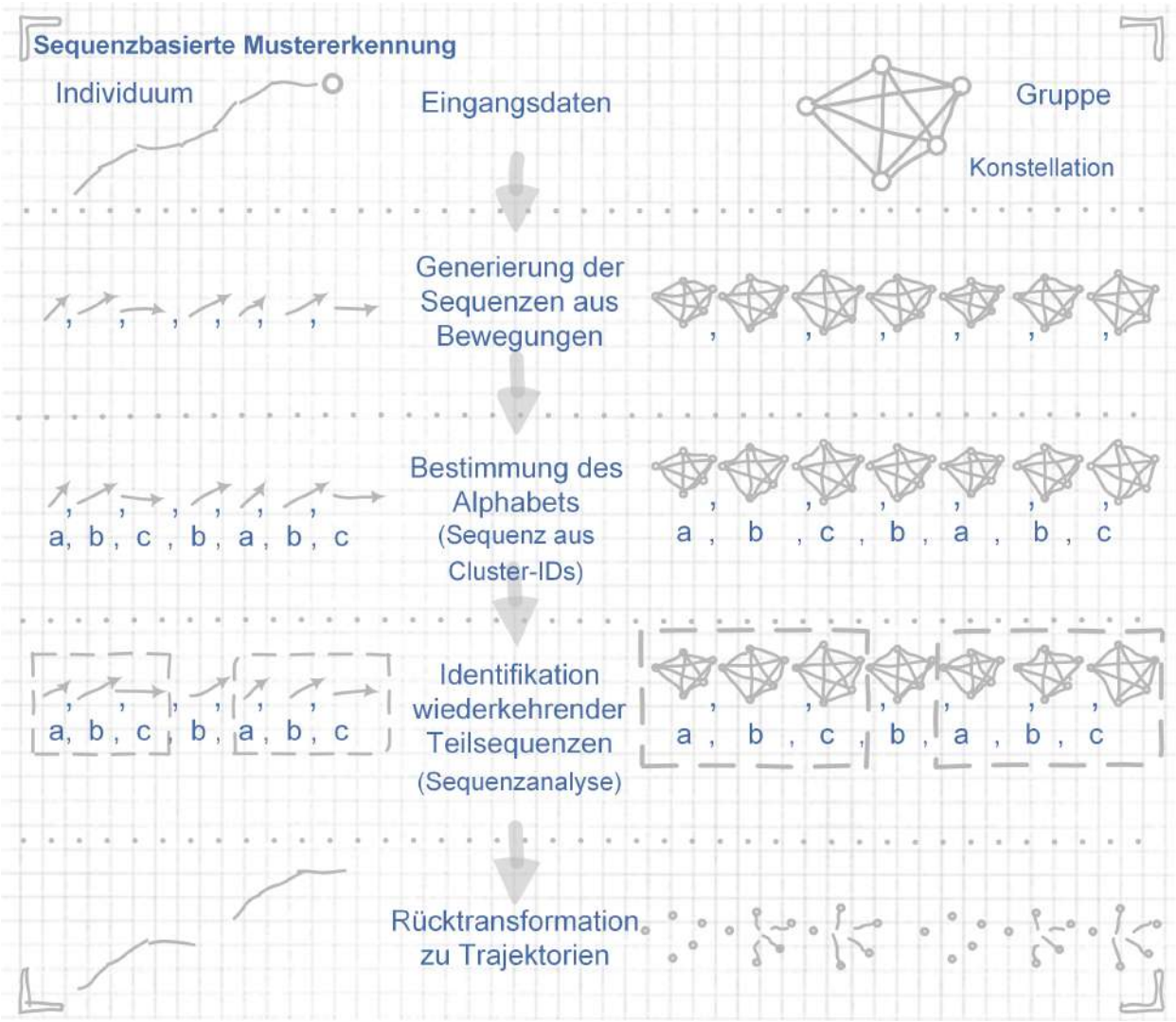


Abbildung 5.6: Eine detaillierte Darstellung der aufeinanderfolgenden Teilschritte der Mustererkennungsphase (siehe Abbildung 5.1) zur Identifikation von individuellen und Gruppenbewegungsmustern.

Auf Grund der Tatsache, dass auch transformierte Muster (Translation und/oder Rotation) detektiert werden sollen, müssen dafür geeignete Merkmalsvektoren gewählt werden. Für den Fall, dass die Bewegungen eines einzelnen Objekts analysiert werden und die resultierenden Muster weder translations- noch rotations- noch skaleninvariant sein müssen, so entsprechen den Elementen die durch die x- und y-Koordinaten (im 2D-Fall) gegebenen Objektpositionen. Falls nach Gruppenbewegungsmustern gesucht wird, so ist jedes Element eine Konstellation, die Informationen über die Positionen sämtlicher Gruppenmitglieder zu einem Zeitpunkt enthält. Dürfen die Teilsequenzen, die die Instanzen eines Musters sind, gegeneinander verschoben sein, d.h. sie sind translationsinvariant, dann muss der Inhalt der Sequenzelemente bei der Analyse einzelner Objekte zu Bewegungsvektoren geändert werden. Im Fall der Gruppenmusteranalyse werden hingegen die Distanzen zwischen den einzelnen Mitgliedern verwendet. Abbildung 5.7 visualisiert ein Beispiel für eine Konstellation und die Anforderungen hinsichtlich ihrer Transformationsinvarianz. Je nach Anwendung werden beide Szenarien, also der Vergleich a) mit b) und a) mit c), als identisch behandelt. In Tabelle 5.3 ist eine Übersicht über die Szenarios und die dazugehörigen Elementinhalte gegeben.

Die Abtastrate, die die Länge der Zeitschritte bestimmt, muss sinnvoll gewählt werden. Sie hängt dabei stark vom Anwendungsfall ab. Werden zum Beispiel Objekte beobachtet, die sich mit ei-

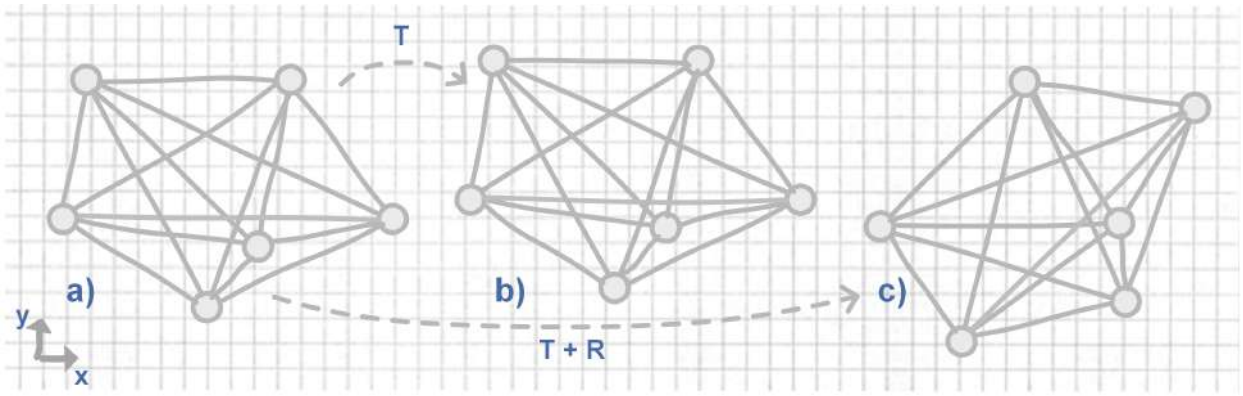


Abbildung 5.7: Abhängig von den gewählten Invarianzen bzgl. Translation T (b) und zusätzlicher Rotation T+R (c) werden die gezeigten Konstellationen als identisch zu a) behandelt.

Tabelle 5.3: Unterschiedliche Anforderungen an die Invarianzen der Muster führt zu unterschiedlichen Inhalten der Sequenzelemente (x,y: Koordinaten, ϕ: Bewegungsrichtung (heading), r: Länge des Bewegungsvektors)

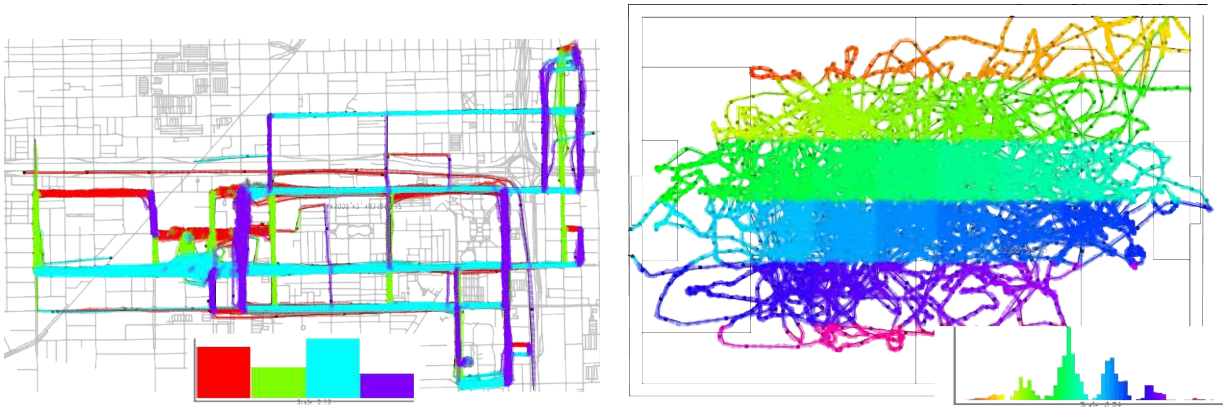
Invarianzen	**Inhalt der Elemente**	
	Individuell	**Gruppe aus n Objekten** ($j,k = 1,\ldots,n$)
Keine	$I_i = [x\ y]^T$ (5.9)	$I_i = [x_j\ y_j]^T$ (5.10)
Translation T	$I_i = [dx\ dy]^T$ (5.11) oder $I_i = [\phi\ r]^T$ (5.12)	$I_i = [dx_{j,k}\ dy_{j,k}]^T$, $j \neq k$ (5.13)
Translation & Rotation T+R	$I_i = [d\phi\ r]^T$ (5.14)	$I_i = [d_{j,k}]^T, j \neq k$ (5.15)

ner recht hohen Dynamik in einem Euklidischen Bewegungsraum uneingeschränkt bewegen, wie es bspw. bei Fußballspielern der Fall ist, dann müssen Bewegungen mit möglicherweise vielen Richtungs- und Geschwindigkeitsänderungen gehandhabt werden. Um die Details solcher Bewegungen nicht zu verlieren, muss die Abtastrate entsprechend hoch sein. Generell bedeutet dies, dass eine höhere Abtastrate eine präzisere Erfassung der Objektbewegungen und somit auch ein Erkennen von detaillierteren Bewegungsmustern ermöglicht. Zugleich bedingt es jedoch auch längere Sequenzen von Bewegungen, die einen ebenfalls höheren Rechenaufwand erfordern.

5.6.3 Bestimmung des Alphabets

Die zu analysierenden Objektbewegungen sind kontinuierlich und im Fall eines Euklidischen Bewegungsraums ohne Einschränkungen in Bezug auf Richtung und Geschwindigkeit. Des Weiteren besitzen die Sequenzelemente gewisse Unsicherheiten, die aus den Ungenauigkeiten der unterschiedlichen Lokalisierungsmethoden (siehe Abschnitt 2.2) resultieren. Aus diesen Gründen ist die Zahl an sich wiederholenden Elementen in der Sequenz sehr gering, d.h. das *Alphabet*, das sämtliche unterschiedliche Elemente beinhaltet, ist sehr lang. Es ist daher nicht zu erwarten, dass innerhalb einer begrenzten Beobachtungszeit exakt gleiche Teilsequenzen identifiziert werden können. Um das Alphabet zu verkleinern und damit die Chance zu erhöhen, dass mehr wiederkehrende Teilsequenzen gefunden werden können, wird eine gewisse Toleranz bezüglich der in den Elementen gespeicherten Werte eingeführt, sodass bereits ähnliche Werte einem Element des Alphabets zugeordnet werden. Zu diesem Zweck werden die Elemente von S_{tot} diskretisiert und nach Ähnlichkeit gruppiert.

Dazu existieren zwei Möglichkeiten: Auf der einen Seite können vordefinierte Gruppen bzw. Cluster genutzt werden. Letztere sind je nach geforderter Invarianz unterschiedlich. Sollen individuelle und translationsinvariante Bewegungsmuster identifiziert werden, können die Gruppierung bspw. nach Bewegung in die entsprechende Himmelsrichtung („Norden“, „Süden“, „Westen“, „Osten“) stattfinden (siehe Abbildung 5.8a). Dadurch würden sich vier Gruppen (Länge des Alphabets ist 4) ergeben. Eine feinere Untergliederung oder die Verwendung von anderen Features ist natürlich ebenso möglich. Werden Muster ohne jede Invarianz gesucht, können die Gruppen durch eine Rasterisierung des Bewegungsraums ermittelt werden (5.8b). Dabei würde jede Rasterzelle einem Element im Alphabet entsprechen. Die in der Zelle befindlichen Sequenzelemente werden dem Alphabetelement zugeordnet. Zur Erinnerung: Bei keiner Invarianz besteht der Vektor aus den Koordinaten des Objekts und kann daher verortet werden. Bei Translations- und Rotationsinvarianz ist in den Vektoren die Richtungsänderung gespeichert. Eine vordefinierte Gruppierung entspricht einer Einteilung in Winkelbereiche.

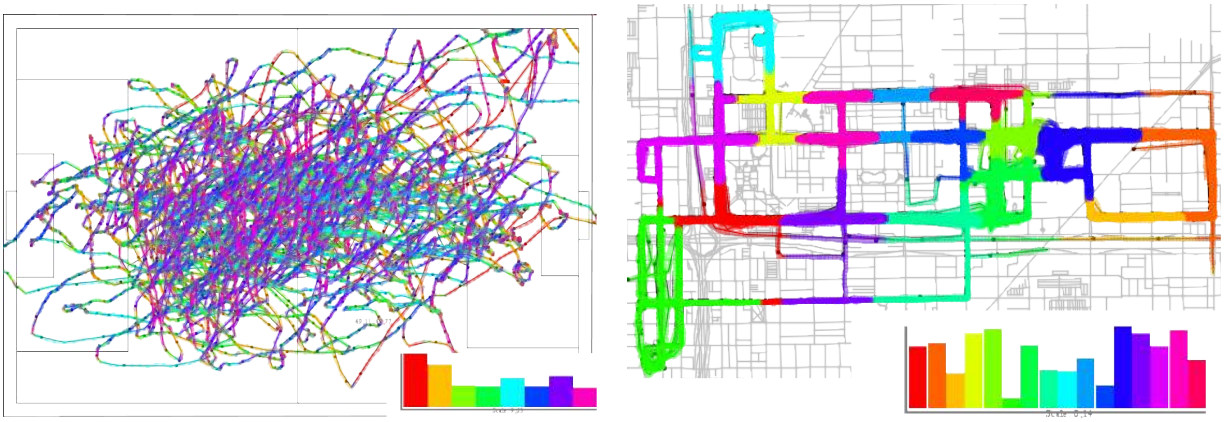


(a) *(b)*

Abbildung 5.8: Die Gruppierung der Elemente der Bewegungssequenz S_{tot} ist abhängig von den Invarianzen. Die unterschiedlichen Farben kennzeichnen die jeweilige Gruppenzugehörigkeit. Das Histogramm zeigt jeweiligen Gruppengrößen. (a) Translationsinvarianz: die Elemente der Sequenz können u.a. nach den Himmelsrichtungen gruppiert werden. Dieses bietet sich bei dem gegebenen Straßennetz besonders gut an. (b) Keine Invarianz: Das Spielfeld wird in Rasterzellen unterteilt. In diesem Fall ist es ein regelmäßiges Gitter mit einer Zellbreite /-höhe von 5 m. Hier könnte weiteres Kontextwissen einfließen, sodass spezielle Spielfeldzonen ermittelt werden, die ein unregelmäßige Kachelung bilden.

Auf der anderen Seite kann mittels eines Clustering-Verfahrens nach natürlichen Clustern gesucht werden (siehe Abbildung 5.9). Dieses ist besonders dann relevant, wenn keine Kontextinformationen über existierende Cluster vorliegen. Hier kommen entweder dichtebasierte Verfahren, z.B. *DBSCAN*, oder partitionierende Verfahren, z.B. *k-means*, in Frage. In diesem Fall wird die Länge des Alphabets durch die Wahl der benötigten Clustering-Parameter bestimmt. Am Beispiel des k-means-Verfahrens wird die Anzahl der resultierenden Cluster durch den Parameter k festgelegt. Die Länge des Alphabets ist demnach k. Der erforderlich Grad der Ähnlichkeit, dass Elemente dem gleichen Cluster zugeordnet werden nimmt mit steigendem k zu. Die Ähnlichkeit der sich wiederholenden Teilsequenzen steigt ebenso. Jedoch ist zu beachten, wie eingangs erwähnt, dass mit zunehmender Alphabetlänge, die Anzahl an wiederkehrenden Teilsequenzen sinkt. Für das DBSCAN-Verfahren kann die Länge des Alphabets nicht direkt angegeben werden. Dort hängt sie von der Wahl der Parameter ϵ und $minIts$ ab. Je niedriger die geforderte Dichte und kleiner die Mindestanzahl an Cluster-Elementen gewählt wird, desto mehr Cluster entstehen und desto länger wird das Alphabet. Das in beiden Fällen benötigte und vom Anwendungsfall abhängige Ähnlichkeitsmaß bzw. Distanzmaß ist die berechnete räumliche Distanz, die von den in den Elemen-

ten gespeicherten Vektoren aufgespannt wird. Ähnliche Sequenzelemente werden so dem gleichen Cluster zugeordnet und werden durch das entsprechende Alphabetelement repräsentiert.



(a) (b)

Abbildung 5.9: Das Clustering der Elemente der Bewegungssequenz S_{tot} ist ebenfalls von den Invarianzen abhängig: (a) Zur Identifikation natürlicher Cluster in Bezug auf die Bewegungsrichtung kann ein Clustering verwendet werden. Hier werden die Bewegungen in 8 Cluster unterteilt. (b) Ist kein Vorabwissen für eine sinnvolle Kachelung des Bewegungsraums bekannt, so können die Bewegungen bzw. Sequenzelemente räumlich geclustert werden. In diesem Fall werden 16 Cluster angenommen.

Die resultierende Sequenz $S_{tot,alpha}$ besteht aus den Alphabetelementen. Um weitere Toleranzen gegenüber leichten Abweichungen zwischen Teilsequenzen einzuräumen, wird $S_{tot,alpha}$ mit Hilfe eines gleitenden Mehrheitsfilters geglättet. Daraufhin wird sie vereinfacht, indem aufeinanderfolgende gleichartige Elemente zu einem aggregiert werden. Dieses verringert die Gesamtlänge der Sequenz und damit auch den nötigen Berechnungsaufwand.

5.6.4 Identifikation wiederkehrender Teilsequenzen

Unter Berücksichtigung der in Abschnitt 5.1 gegebenen Definition eines Bewegungsmusters, wird in $S_{tot,alpha}$ nach entsprechenden Mustern gesucht. Mit Angaben für den Mindest-Support $suppc_{min}$ und die Mindestmusterlänge l_{min} ist es möglich, existierende Sequenzanalyseverfahren (*frequent pattern mining methods*) anzuwenden, die häufige Muster in den Daten erkennen. Wenn zudem leichte Abweichungen zwischen den häufigen Teilsequenzen innerhalb eines Musters erlaubt sein sollen, kann ein approximatives Verfahren genutzt werden. Ein solches Verfahren verlangt im Allgemeinen einen weiteren Parameter d, der die erlaubte Abweichung quantifiziert. Sie nutzen gewisse Distanzmetriken, um die Abweichung zweier Teilsequenzen zu bestimmen. Gewöhnlich sind hierbei sogenannte *Edit-Distanzen* im Einsatz, wie bspw. die *Levenshtein-Distanz*, die die minimale Anzahl an benötigten Änderungen (d.h. Ersetzen, Einfügen oder Löschen) liefert, um die eine in die andere Sequenz zu transformieren. Abhängig davon, ob nach exakten oder ähnlichen häufigen Teilsequenzen gesucht wird, bieten sich entsprechende Algorithmen an. Während für die exakte Suche der *Apriori-Algorithmus* geeignet ist, kommt für die nicht-exakte Suche u.a. der *Baeza-Yates-Gonnet-Algorithmus* in Frage. In beiden Fällen werden alle häufigen Teilsequenzen bestimmt, die die gegebenen Anforderungen erfüllen. Jede dieser gefundenen häufigen Teilsequenzen formt zusammen mit all ihren Instanzen ein Muster in der Cluster-Sequenz $S_{tot,alpha}$.

5.6.5 Rücktransformation zu Trajektorien

Da jedoch nicht die gefundenen wiederkehrenden Cluster-Sequenzen das Endergebnis darstellen, sondern deren jeweilige Instanzen in der Sequenz S_{tot} bzw. in den Daten, müssen die Cluster-Sequenzen zu den originalen Bewegungssequenzen bzw. zu den Objekt-Trajektorien zurücktransformiert werden. Zu diesem Zweck werden einfache Zuordnungstabellen (*maps*) verwendet, die $S_{tot,alpha}$, S_{tot} und die Trajektorien miteinander verknüpfen und so eine Transformation in beide Richtungen ermöglichen.

5.7 Laufzeit des Algorithmus

Wird die Vorverarbeitung der Trajektorien außer Acht gelassen, so hängt die Laufzeit des Mustererkennungsprozesses von der gewählten Variante und den dort verwendeten Algorithmen ab.

Clustering-basierter Ansatz

Beim Clustering-basierten Ansatz müssen die Laufzeiten der Segmentierung und des Trajektorie-Clusterings berücksichtigt werden. Die Laufzeit der Segmentierung unterscheidet sich je nach gewähltem Verfahren. Zur Detektion der interessanten Plätze müssen sämtliche Trajektoriepunkte TP einmal verarbeitet werden, sodass sich eine Komplexität von $O(n_{TP})$ ergibt, mit n_{TP} als Anzahl der Trajektoriepunkte. Die Segmentierung der Trajektorien auf Grundlage der Plätze erfordert eine zusätzliche Iteration. Bei der semantischen Segmentierung müssen die Trajektoriepunkte anhand eines zuvor gelernten Klassifikationsmodells klassifiziert werden. Wird das Modell als gegeben angenommen, ist auch hier eine Komplexität von $O(n_{TP})$ zu erwarten. Das Clustering der resultierenden Segmente S erfolgt mit Hilfe eines Clustering-Verfahrens und einem geeigneten Distanzmaß. Die Berechnungskomplexität kann den Tabellen 2.4 und 2.3 entnommen werden. Wird bspw. der DBSCAN-Algorithmus ($O(n_S \cdot \log n_S)$) in Kombination mit dem Euklidischen Distanzmaß ($O(n_{TP,S})$) verwendet, so ergibt sich eine Komplexität von $O(n_{TP,S} \cdot n_S \cdot \log n_S)$, mit $n_{TP,S}$ als mittlere Zahl an Trajektoriepunkten pro Segment und n_S als Anzahl der Segmente. Werden anstelle der Euklidischen Distanz die komplexeren Fréchet-, Hausdorff- oder DTW-Distanz genutzt, steigt entsprechend die Berechnungskomplexität.

Sequenzbasierter Ansatz

Bei der sequenzbasierten Variante hängt der Berechnungsaufwand von den verwendeten Verfahren zur Erstellung des Alphabets und zur Erkennung der wiederkehrenden Teilsequenzen ab. Demnach spielen auch die Laufzeiten der Clustering-Methoden, gegeben in Tabelle 2.4, eine Rolle. Wird das k-means-Verfahren benutzt, kann der Aufwand zur Erstellung des Alphabets durch $O(n_{TP}kdi)$ angegeben werden. Hierbei sind n_{TP} die Anzahl der Sequenzelemente, die der Anzahl der Trajektoriepunkte entspricht, k die Clusterzahl, d die Dimension der Daten und i die benötigten Iterationen. Die Glättung mit Hilfe des Mittelwert-Filters erfolgt in $O(n_{TP})$, die Aggregation sich wiederholender Elemente in $O(n_{TP})$. Die Suche nach Mustern in der resultierenden Sequenz $S_{tot,alpha}$ mit der Länge $l(S_{tot,alpha}) = l_S$ und einer Alphabetlänge l_{alpha} erfordert bei Verwendung des Apriori-Algorithmus einen Aufwand in der Komplexität von $O(l_{alpha}^2 \cdot l_S)$ (Goh u. a., 2007) . Der Baeza-Gonnet-Algorithmus ermöglicht eine Suche in $O(l_S)$, benötigt jedoch eine Vorverarbeitung, die $O(l_{min} + l_{alpha})$ erfordert (Baeza-Yates und Gonnet, 1992).

6 Experimente und Evaluation der Erfassung der Trajektorien

In diesem Kapitel wird das vorgestellte Verfahren zur Erfassung von Trajektorien evaluiert und diskutiert. Zu diesem Zweck werden zuerst im Abschnitt 6.1 die verwendeten (eigens implementierten) Software-Tools vorgestellt. In den weiteren Abschnitten werden daraufhin die verwendeten Sensoren beschrieben (Abschnitt 6.2) sowie die Korrektheit der Objektdetektionen und Zuordnungen (Abschnitt 6.3) untersucht. Des Weiteren wird in einem zusätzlichen Experiment die geometrische Genauigkeit der resultierenden Trajektorien ermittelt (Abschnitt 6.4). Dieses Kapitel schließt mit einer Laufzeitanalyse (6.5) und einem Fazit (6.6).

6.1 Verwendete Software

Zur Ermittlung der Ergebnisse wurden unterschiedliche eigens realisierte Softwares genutzt. Die entwickelten Tools werden hier in aller Kürze vorgestellt.

GPSDeviceImport

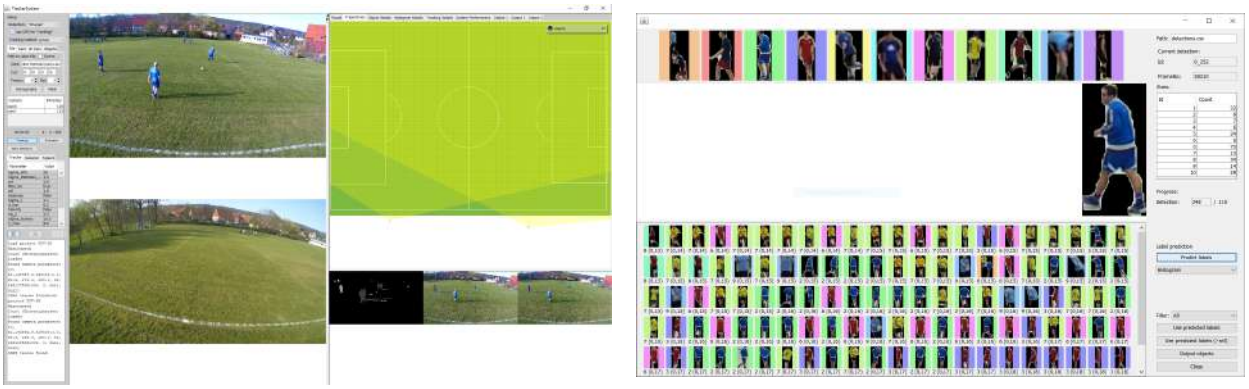
Dieses kleine Tool dient zum Auslesen der GPS-Logger, die zur Ermittlung der GPS-Positionsinformationen verwendet werden (siehe Abschnitt 6.2). Werden bei einem Experiment bis zu 22 Objekte mit diesen Loggern ausgestattet, so ist anschließend eine sequenzielle Verarbeitung der Geräte mit herkömmlicher existierender Software recht mühsam. Dieses Tool ermöglicht daher eine Batch-Verarbeitung von allen angeschlossenen Loggern. In Verbindung mit einem Multi-USB-Hub und einer automatischen Zuordnung von Daten eines Geräts zu einem speziellen Objekt (anhand einer einmalig erstellten, erweiterbaren Zuordnungstabelle) wird der Ausleseprozess deutlich vereinfacht und beschleunigt.

TrackingTest

Diese Software bildet mit der Erfassung der Trajektorien einen der beiden Themenschwerpunkte dieser Arbeit ab. Sie kann „offline“ bzw. „online“ betrieben werden. Im „Offline“-Modus können zuvor aufgezeichnete Video- und GPS-Daten gemäß des hier vorgestellten Verfahrens verarbeitet werden. Im „Online“-Modus werden die Daten direkt von verbundenen Web-/IP-Kameras bereitgestellt und je nach Möglichkeit in Echtzeit prozessiert. Das Tool besitzt auf der graphischen Oberfläche eine umfassende Visualisierungskomponente (siehe Abbildung 6.1a), mit der (Zwischen-) Ergebnisse visuell aufbereitet werden. Zusätzlich können sämtliche Eingabe-Parameter für die entsprechenden Algorithmen gesetzt und zur Laufzeit verändert werden. Die Ausgabe der Trajektorien erfolgt als CSV-Datei, die die Schnittstelle zu sich anschließenden Analysen, wie bspw. die Mustererkennung, ist. Die Experimente dieser Arbeit sind allesamt mit dieser Software durchgeführt worden.

Labeling-Tool

Mit Hilfe des Labeling-Tools (siehe Abbildung 6.1b) wurden die zur Evaluation verwendeten Referenzdatensätze erzeugt. Dieses Werkzeug ermöglicht eine effizientere Generierung der tatsächlichen Objektkennungen, indem es anhand von Beispielzuordnungen weitere Zuordnungen prädiziert. Die Überprüfung der Prädiktionen auf der graphischen Oberfläche und eine eventuelle Neuzuweisung ist in den meisten Fällen schneller als eine komplett eigenhändige Zuweisung der Detektionen.



(a) (b)

Abbildung 6.1: a) In dem TrackingTest-Tool wurde das in dieser Arbeit vorgestellte Verfahren zur Erfassung von Trajektorien implementiert. Es wurde zur Ermittlung sämtlicher Ergebnisse, die in diesem Kapitel beschrieben werden, genutzt. b) Das Labeling-Tool unterstützte die Erstellung der Referenzdatensätze.

6.2 Verwendete Sensoren

Hochgenaue Objekt-Trajektorien lassen sich heutzutage mit State-of-the-Art-Mitteln und teurem Equipment in Form von Multi-Kamera-Systemen erzeugen. Der in dieser Arbeit beschriebene Erfassungsansatz fusioniert Daten unterschiedlicher Sensoren (GPS-Daten und Kamera-Daten), um ebenfalls hoch-genaue Trajektorien zu generieren. In diesen Experimenten wird allerdings bewusst auf verhältnismäßig günstige Sensoren zurückgegriffen, um zu zeigen, dass auch mit Low-Cost-Systemen sehr gute Ergebnisse erzielt werden können. Die Sensoren, die zur Erfassung der entsprechenden Daten genutzt werden, sind zum einen GPS-Logger der Firma *QSTARZ* (Abbildung 6.2a) und unterschiedliche Kameras. Bei Letzteren handelt es sich zum einen um ein als Kamera genutztes *Samsung Galaxy S5* Smartphone (6.2b, Hersteller und Typ sind hier eher unbedeutend, solange die Kamera bzw. Leistung des Smartphones ausreichend sind) und zum anderen um die Action-Kamera *SJCAM SJ 5000x Elite* (6.2c, bzgl. Hersteller und Typ verhält es sich hier gleich).

(a) (b) (c)

Abbildung 6.2: Die in den Experimenten verwendeten Sensoren: a) der QSTARZ BT-Q1300ST GPS-Logger, b) das Samsung Galaxy S5 Smartphone als Smart-Kamera und c) die SJCAM SJ 5000x Elite Action Cam.

Handelsübliche GPS-Logger liefern abhängig von ihren Spezifikationen 3D-Positionsdaten mit einer Rate von 1 bis 10 Hz. In Anwendungsfällen mit schnellen Bewegungen und häufigen Richtungsänderungen kann eine zeitliche Auflösung von 1 Hz bereits zu niedrig sein, um die Trajektorien detailgetreu zu erfassen (Coutts und Duffield, 2010; Gray u. a., 2010). Für die nachfolgenden

Experimente werden daher GPS-Logger genutzt, die Objektposition mit einer Rate von 5 Hz updaten und auch speichern. Letzteres geht nicht immer deutlich aus den Angaben der Hersteller hervor, sodass manche Geräte zwar die Positionen mit 5 Hz bestimmen aber nur sekündlich aggregiert speichern. Ferner sind in den Spezifikationen Positionsgenauigkeiten von 3.0 m 2D-RMS (ohne Unterstützung) angegeben. Die zu beobachtenden Objekte werden mit diesen Loggern ausgestattet.

Die Kameras werden dazu verwendet, die gesamte Szene, in der sich die Objekte bewegen, zu überblicken. Das verwendete Smartphone erlaubt unterschiedliche Betriebsmodi mit unterschiedlichen Auflösungen und Bildraten. Diese reichen von einer HD-bzw. Full-HD-Auflösung mit mindestens 60 fps bis hin zu einer 4K-Auflösung mit 30 fps. Je nach Größe der zu beobachtenden Fläche wird der Modus angepasst. Die verfügbaren und gewählten Varianten werden in Tabelle 6.1 aufgelistet. Hierbei gilt es zu beachten, dass mit höherer Auflösung und Bildrate die zu verarbeitende Datenmenge zunimmt und sich die Laufzeit der entsprechenden Algorithmen verschlechtert. Die Auflösung hat einen Einfluss auf die Genauigkeit der Positionsbestimmung der Objekte.

Die Bilder der hier verwendeten Kameras besitzen standardmäßig eine Verzeichnung. Diese würden zu Ungenauigkeiten bei der Objektpositionierung führen und müssen daher herausgerechnet werden. Zu diesem Zweck müssen die Kameras kalibriert werden, d.h. ihre intrinsischen Parameter bestimmt werden. Unter der Verwendung von existierenden Software-Tools bzw. Bildverarbeitungs-Frameworks, wie bspw. MATLAB oder OpenCV, und einem festen, bekannten Schachbrettmusters kann sowohl die Kalibrierung als auch die Korrektur der Bilder durchgeführt werden. Abbildung 6.3 zeigt ein Beispielbild vor und nach der Korrektur.

(a) *(b)*

Abbildung 6.3: Die Kamerabilder müssen zu Beginn entzerrt werden. Das Bild vor (a) und nach (b) der Korrektur.

Um schließlich die Objektpositionen in Weltkoordinaten zu bestimmen, wird eine vorberechnete Homographie verwendet. Diese wird mit Hilfe der gleichen Tools und bekannten korrespondierenden Punktpaaren (mind. 4) im Bild und in der Welt berechnet. Sie dient zur Transformation der Bildkoordinaten in Weltkoordinaten.

Für die Experimente wird das System aus GPS-Loggern und Kameras *offline* betrieben. Das bedeutet, es findet keine Kommunikation statt. Die GPS-Geräte verfügen zwar über eine Bluetooth-Schnittstelle, die jedoch nicht für den Austausch der Daten in Echtzeit und über die benötigten Distanzen hinweg genutzt werden kann. Würden andere GPS-Geräte verwendet, die über eine geeignete Kommunikationsschnittstelle verfügen, könnte dieses System prinzipiell auch *online* betrieben werden. Bei der Nutzung von Smartphones könnten diese als Smart-Kameras fungieren, Daten bereits lokal verarbeiten und sich mit ihren Nachbarn austauschen. Das System könnte damit dezentral arbeiten (vgl. Abschnitt 4.6).

In Tabelle 6.1 werden die Eigenschaften der Sensoren übersichtshalber zusammengefasst. Bei einem Blick auf die Anschaffungskosten für die Geräte wird deutlich, dass es sich hier um ein kostengünstiges (*Low-cost-*) System handelt.

Tabelle 6.1: Ein Überblick über die wichtigsten Eigenschaften der verwendeten Sensoren.

Eigenschaft	**BT-Q1300ST**	**Galaxy S5**	**5000x Elite**
Sensor-Typ	GPS-Logger	Kamera	Kamera
Auflösungen [Pixel] / Daten-Rate	- / 5 Hz	1280×720 / 60 fps 1920×1080 / 60 fps 3840×2160 / 30fps	1280×720 / $>$ 60 fps 1920×1080 / 60 fps 3840×2160 / 30 fps
Speicher	8 MB	bis zu 128 GB	bis zu 64 GB
Kommunikation	Bluetooth	Bluetooth, WLAN, GSM	WLAN
Positionsgenauigkeit	3 m (ohne Unterstützung)	-	-
Preis (Stand Juli 2017)	ca. 120 €	ca. 250 €	ca. 120 €

6.3 Korrektheit der Zuordnungen

Sowohl bei der detektions- als auch bei der rasterbasierten Modellierungsvariante werden die Objektdetektionen in den Kamerabildern zur Ermittlung der Objektpositionen genutzt. Bei der detektionsbasierten Variante werden die Positionen ausschließlich aus den Kameradetektionen bestimmt. Bei der rasterbasierten Variante werden sie durch eine Kombinationen aus Kameradetektionen und GNNS-Positionen berechnet. Die Qualität der Objekt-Trajektorien hängt somit in beiden Fällen von der Korrektheit der Zuordnungen von Detektionen zu Objekten ab. Da jedoch bei beiden Modellierungen die gleichen Features für die Zuordnung genutzt werden und somit dieselben Kameradetektionen bei der Bestimmung der Objektpositionen involviert sind, müssen nicht beide Varianten gesondert evaluiert werden. Hier ist es ausreichend eine der beiden Varianten stellvertretend anzuwenden. Allerdings gilt es zu beachten, dass auf Grund der unterschiedlichen Positionsbestimmung, die jeweils resultierenden Trajektorien nicht geometrisch identisch sein werden. Die geometrische Genauigkeit beider Varianten wird in einem zusätzlichen Experiment (Abschnitt 6.4) evaluiert.

In den beiden nachfolgenden Experimenten werden daher die Ergebnisse der Objektdetektion und des Objekt-Trackings untersucht. Zur Ermittlung der Ergebnisse werden die Detektionsmethode und die detektionsbasierte Tracking-Variante auf unterschiedliche Datensätze angewendet. Zur Evaluation der Ergebnisse werden selbst erzeugte Referenzdaten benutzt, die Informationen über die tatsächlichen Objektidentitäten enthalten. Mit diesen Informationen können typische Kennzahlen bezogen auf die Zuordnungskorrektheit ermittelt werden. Bei diesen handelt es sich um die *Sensitivität* TPR (*true positive ratio* oder *recall*), die durch

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FP} \tag{6.1}$$

berechnet wird, und die *Falsch-negativ-Rate* FNR (*false negative ratio*), die sich durch

$$FNR = \frac{FN}{FN + TP} = 1 - TPR \tag{6.2}$$

ermitteln lässt. Dabei sind TP (*true positives*) die korrekten sowie FP (*false positives*) und FN (*false negatives*) die falschen Zuordnungen. P entspricht der Anzahl der tatsächlichen Detektionen des jeweiligen Objekts.

Ein Vergleich zu anderen Verfahren gestaltet sich allerdings schwierig, da die üblichen Referenz-Datensätze zwar über die tatsächlichen Objektdetektionen bzw. -identitäten verfügen, jedoch keine zusätzlichen Sensor-Messungen wie bspw. die GNSS-Trajektorien enthalten. Diese sind aber für den vorgestellten Ansatz essentiell. Daher werden die Experimente zweimal durchgeführt: einmal mit Berücksichtigung der GNSS-Informationen, einmal ohne. Die daraus resultierenden Ergebnisse werden miteinander verglichen, um die mögliche Verbesserung durch die GNSS-Unterstützung bewerten zu können.

6.3.1 Experiment: 2 Personen

In diesem Datensatz werden zwei Personen mit Hilfe der Smartphone-Kamera (Samsung Galaxy S5, siehe Abschnitt 6.2) und jeweils einem GPS-Logger (mit je einer Sampling-Rate von 5 Hz) über ca. vier Minuten beobachtet. Die beiden Personen bewegen sich zufällig durch den Szene. Da die Laufwege sich nahe beieinander befinden und auch kreuzen, entstehen Verdeckungen, sodass eine der beiden Personen aus der Kameraperspektive zeitweise partiell oder ganz verschwindet. Zudem verlässt eine der Personen für einige Sekunden die Szene. Die Daten werden auf allen Geräten für eine Prozessierung im Nachhinein aufgezeichnet. Die Bilder der Kamera werden wie im vorherigen Abschnitt beschrieben korrigiert. Die Umrechnung der Bildkoordinaten in Welt- bzw. Spielfeldkoordinaten erfolgt mit Hilfe einer vorberechneten Homographie. Letztere ist anhand von korrespondierenden Punktpaaren (z.B. markante Punkte auf dem Spielfeld) bestimmt worden. Die GPS-Daten werden in das gleiche Koordinatensystem umgerechnet.

Objektdetektionen und Referenzdaten

Zur Erzeugung der Referenzdaten wird die Objektdetektion in Form des *Gaussian Mixture-based Background Substraction*-Verfahrens ohne anschließendes Tracking auf die Bilddaten angewendet. Die resultierenden Detektionen werden gespeichert und mit Hilfe des Labeling-Tools manuell mit der korrekten Objektkennung versehen oder als Rauschen deklariert. Die Übereinstimmung der ermittelten Objektbildregionen mit den tatsächlichen Objekten im Bild wird hier nicht überprüft oder korrigiert. Die Detektionen werden zudem auf Vollständigkeit kontrolliert, d.h. es wird geprüft, ob in jedem Bild alle Objekte detektiert worden sind. Dadurch lässt sich ebenfalls die Anzahl an Fehldetektionen ermitteln, die im späteren Verlauf das Tracking und auch die Laufzeit der Algorithmen beeinflussen.

Die erforderlichen Parameter zur geometrischen Filterung der Detektionen werden wie folgt gesetzt: minimale Fläche eines Begrenzungsrechtecks $A_{bbox,min} = 100\ Pixel$, maximales Verhältnis von Breite zu Höhe des Begrenzungsrechtecks $w/h_{max} = 0{,}8$. Auf diese Weise werden in diesem Datensatz insgesamt 3373 Detektionen in 1260 Bildern ermittelt. Die Zahlen im Hinblick auf die Vollständigkeit der Detektionen und die Anzahl an Fehldetektionen sind in Tabelle 6.2 angegeben. Bei einem Blick auf die Zahlen fällt auf, dass Person 1 weniger oft detektiert worden ist. Dies liegt hauptsächlich daran, dass sie kurzzeitig die Szene verlässt. Zudem werden hier bei beiden Personen die Mischdetektionen herausgerechnet. Letztere entstehen dadurch, dass sich die beiden Personen gegenseitig partiell verdecken. Die resultierende Detektion besteht aus den zusammengefassten Bildregionen beider Personen. Sie werden gesondert aufgeführt, da sie wie die fehlenden Objektdetektionen ebenfalls ein Problem bei der Zuordnung darstellen. Die Fehldetektionen werden durch sich bewegende Objekte im Hintergrund hervorgerufen. In Bezug auf die Laufzeit des Verfahrens spielt die Anzahl an Detektionen pro Frame eine wichtige Rolle, da diese die Anzahl

an Detektionsvariationen bestimmt (vgl. Abschnitt 4.4.1). Bei einem aufgerundeten Wert von 3 Detektionen und zwei zusätzlichen Dummy-Detektionen (2 Personen: $n = 2$, 5 Detektionen: $l = 5$) wäre demnach eine Menge aus $M = \frac{l!}{(l-n)!} = 20$ Variationen zu verarbeiten (vgl. Abschnitt 4.4.1). Diese Zahl stellt kein großes Problem dar. Für eine weitere Beschleunigung müsste die Anzahl der Fehldetektionen weiter gesenkt werden, wobei ein Wert von unter einer fehlerhaften Detektion pro Frame recht niedrig ist. Unter Berücksichtigung, dass hier lediglich ein Standardverfahren angewendet worden ist, sind diese Werte akzeptabel.

Tabelle 6.2: Die Ergebnisse der Objektdetektion im 2-Personen-Experiment

Frames (gesamt)	1260
Detektionen (gesamt / pro Frame)	3373 / 2,67
Person 1 (gesamt / gesamt nicht erkannt)	1104 / 156
Person 2 (gesamt / gesamt nicht erkannt)	1130 / 130
Mischdetektionen (gesamt)	81
Fehldetektionen (gesamt / pro Frame)	1058 / 0,84

Durchführung und Ergebnisse des Objekt-Trackings

Die beschriebenen Detektionen und Referenzdaten werden für die Durchführung und Evaluation des Objekt-Trackings benutzt. Unter Verwendung der detektionsbasierten Variante des Ansatzes werden diese Detektionen den Objekten zugeordnet. Die Ergebnisse werden darauffolgend mit den Referenzzuordnungen verglichen. Bei der Durchführung werden die folgenden Werte für die erforderlichen Parameter genutzt: $\sigma_{GPS} = 7\ m$, $v_{max} = 12\ \frac{m}{s}$, $P_{pen} = 0{,}1$. Dabei handelt es sich um einen realistischen Wert für die Genauigkeit der GPS-Positionierung sowie für die Maximalgeschwindigkeit, mit der sich ein Mensch bewegen kann. Letztere wird auch bei der zweiten Durchführung ohne GPS-Unterstützung genutzt. σ_{GPS} ist dort natürlich nicht relevant. Dies führt zu den in Tabelle 6.3 und Abbildung 6.4 gezeigten Ergebnissen. In Abbildung 6.4 (unten) werden die Trajektorien dargestellt, die sich ohne Berücksichtigung der GPS-Trajektorien ergeben.

Tabelle 6.3: Tracking-Ergebnisse des 2 Personen-Experiments

	m. GPS-Unterstützung	**o. GPS-Unterstützung**
Objektdetektionen (nicht zuzuordnen)	2234 (81)	2234 (81)
Sensitivität (*recall*) (%)	2053 (91,9 %)	1771 (79,3 %)
Falsch-negativ-Rate (*misses*) (%)	181 (8,1 %)	463 (20,7 %)

Diskussion

Bei insgesamt 2315 Objektdetektionen wird in diesem Fall eine Sensitivität von 91.9 % und eine Falsch-negativ-Rate von 8.1 % ermittelt. Hierbei ist zu beachten, dass die gemischten Detektionen in den Referenzdaten als „nicht zuweisbar“ deklariert worden sind, da dort eine eindeutige Zuordnung nicht möglich ist. Die ermittelten Sensitivitätswerte beziehen sich also auf die zuweisbaren Detektionen (2234). Werden die GPS-Informationen außer Acht gelassen, sinkt die Sensitivität auf 79,3 %. Da die Personen in diesem Experiment nicht sonderlich ähnlich gekleidet sind, sehen die Trajektorien bis auf einige Fehlzuordnungen trotzdem gut aus (siehe Abbildung 6.4, unten). Die fehlerhaften Zuordnungen sind an den Sprüngen in den Trajektorien zu erkennen (lange gerade Linien). Diese resultieren daraus, dass zeitweise falsche Detektionszuordnungen für entsprechend falsche Objektpositionen sorgen. Ein Großteil der Fehlzuordnungen treten auf, wenn die Personen sich weit von der Kamera wegbewegen und dadurch nur noch sehr klein im Bild zu sehen sind. Die Farbwert-Informationen die zur Unterscheidung verwendet werden, sind dementsprechend schlechter. Dies führt zu Verwechslungen und entsprechend schlechteren Ergebnissen. Werden hingegen

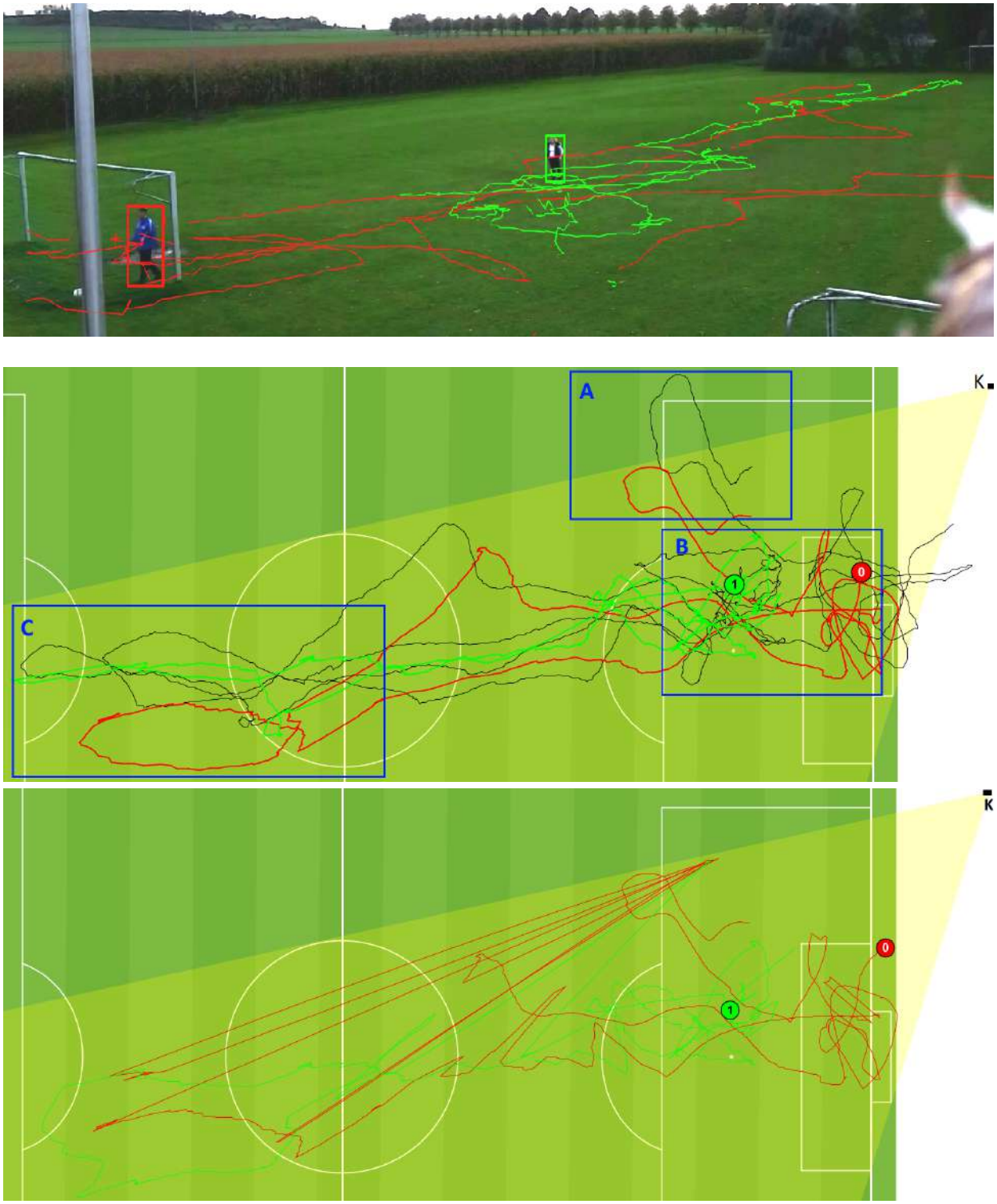


Abbildung 6.4: Die resultierenden Trajektorien des 2-Personen-Experiments dargestellt im Bild (oben) und auf dem Spielfeld mit GPS-Unterstützung (Mitte) und ohne GPS-Unterstützung (unten). Die Trajektorien der Personen sind in rot und grün dargestellt. Zusätzlich sind in schwarz die GPS-Trajektorien und der Kamerastandpunkt (K) eingezeichnet.

die GPS-Informationen berücksichtigt, kommt zu den Farbwert-Histogrammen ein weiteres Unterscheidungsmerkmal hinzu, sodass die Objekte auch auf größere Distanzen besser auseinander gehalten werden können.

In den Ergebnissen sind einige Situationen enthalten, die bei vielen videobasierten Tracking-Lösungen Probleme verursachen. Diese werden im Folgenden näher betrachtet. In der Bildsequenz in Abbildung 6.5 wird bspw. eine Situation gezeigt, in der eine Person die Szene verlässt und nach ungefähr 8 Sekunden wieder betritt. Der Algorithmus ist in der Lage, mit den fehlenden Daten umzugehen und den Personen die korrekten Kennungen zuzuweisen. Dies ist an den entsprechend gefärbten Trajektorien und Begrenzungsrechtecken zu sehen. In der Übersicht, die in Abbildung 6.4 gegeben ist, findet sich diese Situation in der Region A wieder. In der Region B ergibt sich eine weitere problematische Situation, die in der Bildsequenz 6.6 abgebildet wird. Hier verdecken sich die beiden Personen kurzzeitig gegenseitig, sodass die beiden Personen zu Mischdetektionen verschmelzen. Eine eindeutige Zuweisung dieser gemeinsamen Detektionen ist nicht ohne Weiteres möglich. Dennoch ist der Algorithmus auch in dieser Situation in der Lage, nach ihrer Auflösung das Tracking korrekt fortzusetzen. In dem gesamten Datensatz befinden sich mehrere solcher Situationen, welche auch durch andere Objekte wie z.B. dem Fahnenmast im Vordergrund des Bildes oder dem Tor induziert werden.

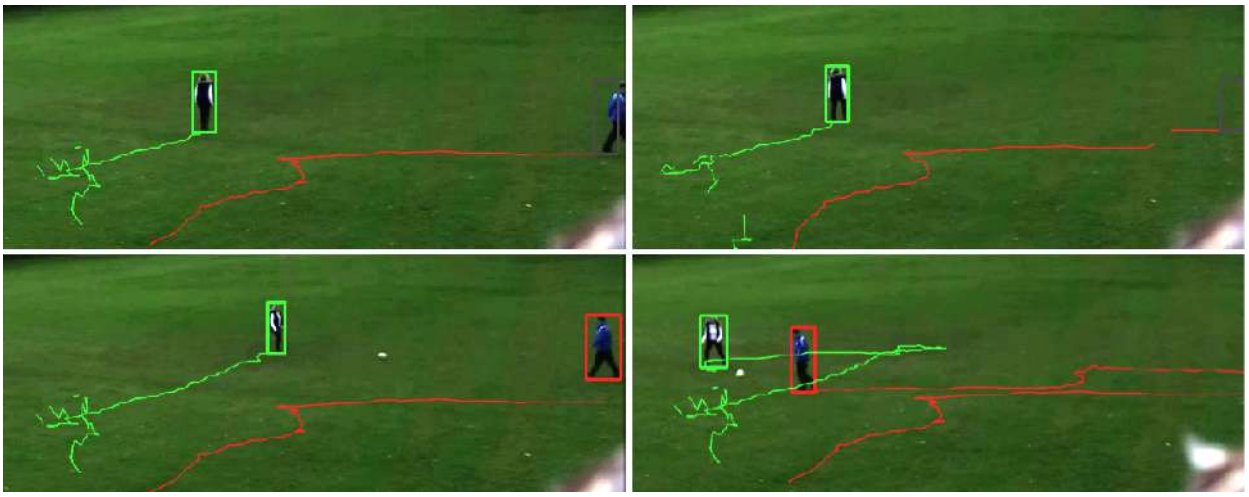


Abbildung 6.5: In dieser Situation (beschrieben durch die Bildsequenz von oben links nach unten rechts) verlässt eine der beiden Personen das Sichtfeld der Kamera für ungefähr 8 Sekunden. Der Algorithmus ist dennoch in der Lage, eine korrekte Zuordnung aufrechtzuerhalten.

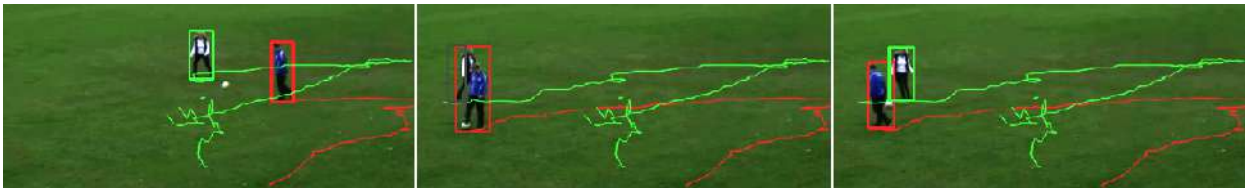


Abbildung 6.6: In einer weiteren Situation verdecken sich die beiden Personen kurzfristig. Die Zuordnung wird dennoch korrekt fortgesetzt, nachdem sich die Situation aufgelöst hat.

Für den Fall, dass die Zuordnung dennoch fehlschlägt, besitzt der verwendete Viterbi-Algorithmus die Fähigkeit, Zuweisungen rückwirkend zu ändern. Dieses unterscheidet ihn von Greedy-Algorithmen, bei denen die Entscheidungen im aktuellen Zeitschritt auf Basis der vorangegangenen Zeitschritte getroffen werden. Der Viterbi-Algorithmus hingegen nutzt alle Beobachtungen, um die wahrscheinlichste Sequenz aus Zuständen zu finden. In Abbildung 6.7 wird ein Beispiel für eine rückwirkende Korrektur der Zuordnungen gezeigt. Während dort anfangs die Objekte und damit auch deren Trajektorien verwechselt worden sind (richtige Zuordnung: blau-gekleidete Person besitzt rote Trajektorie), werden die Zuordnungen im weiteren Verlauf korrigiert. Dies hat dann

ebenfalls einen Einfluss auf die zurückliegende Trajektorie. Diese Eigenschaft führt jedoch dazu, dass dieses Verfahren bei einer Echtzeitanalyse der Bewegungen nur begrenzt nutzbar ist.

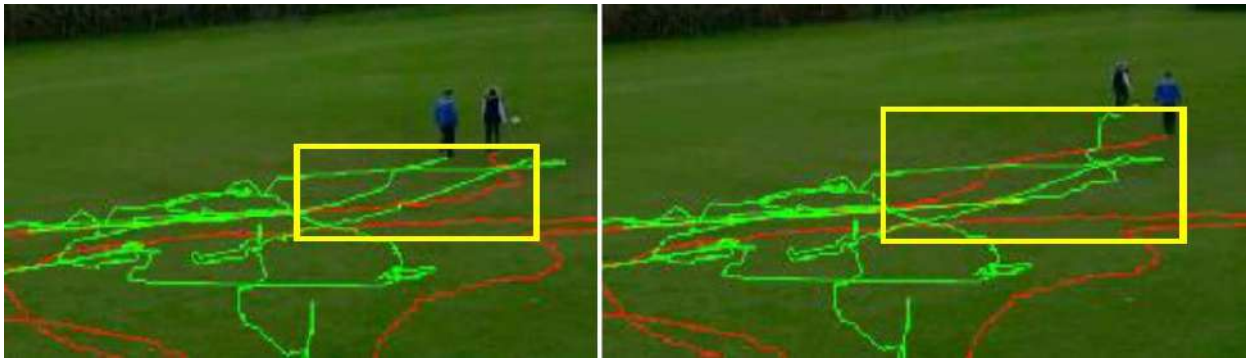


Abbildung 6.7: Der verwendete Viterbi-Algorithmus ermöglicht eine rückwirkende Korrektur der Zuweisungen der Detektionen zu den Objekten. Die Trajektorien im markierten Bereich sind zunächst verwechselt, werden aber im Verlauf rückwirkend korrigiert.

6.3.2 Experiment: Fußballanalyse

In diesem Experiment werden die Trajektorien von Fußballspielern während eines Spiels erfasst. Die Zahl der bewegten Objekten ist dabei mit insgesamt 22 Spielern und einem Schiedsrichter deutlich höher als im vorherigen Experiment. Aus diesem Grund sind ebenfalls deutlich mehr Situationen zu erwarten, in denen sich die Spieler gegenseitig verdecken. Zudem sind die Feldspieler eines Teams, abgesehen von den Schuhen, nahezu identisch gekleidet, was eine weitere Herausforderung bei der Verfolgung darstellt. Da die Erzeugung eines Referenzdatensatzes für das gesamte Spiel und für sämtliche Objekte zu viel Aufwand bedeuten würde, wird die Evaluation in zwei kleineren Teilexperimenten durchgeführt. In dem Ersten wird für eine detaillierte Überprüfung des Trackings von mehreren Spielern gleichzeitig eine ausgewählte Szene mit einer Länge von ca. einer Minute aus dem Gesamtdatensatz betrachtet. Bei dieser handelt es sich um eine Situation während des Spiels, in der sich auf Grund eines Angriffs zeitweise sehr viele Spieler auf engem Raum befinden. Für dieses Experiment wird diese Szene noch einmal zeitlich in drei Phasen unterteilt: In der ersten beginnt der Angriff und die Spieler sind noch recht gut verteilt. Die erwartete Anzahl an Verdeckungen ist hier relativ niedrig. In der mittleren Phase befinden sich viele Spieler im oder in der Nähe des Strafraums. Hier wird mit entsprechend vielen Verdeckungen gerechnet. In der letzten Phase ist der Angriff vorüber und die Spieler verteilen sich wieder. Die Anzahl der Verdeckungen sollte demnach wieder abnehmen. In dem zweiten Teilexperiment wird untersucht, wie gut die Ergebnisse dieses Ansatzes über einen längeren Zeitraum sind. Zu diesem Zweck wird ein ausgewählter Spieler über die erste Hälfte der ersten Halbzeit (ca. 22,5 Minuten) verfolgt.

Das Setup der Sensoren gestaltet sich wie folgt: Die Spieler eines Teams werden mit den in Abschnitt 6.2 beschriebenen GPS-Loggern ausgestattet. Die Spieler des anderen Teams werden aus diversen Gründen nicht mit Loggern versehen. Wie in Abbildung 6.8 dargestellt wird, werden zusätzlich zwei der vorgestellten Action-Kameras (A, B) in einer Höhe von ca. 6 m an zwei der Flutlichtmasten auf einer Seite des Platzes angebracht. Beide Kameras nutzen ein Weitwinkelobjektiv (*fisheye*) mit einem Öffnungswinkel von 170 Grad, sodass nahezu das komplette Spielfeld überblickt werden kann. Die Videos werden für eine Offline-Verarbeitung mit einer 2K-Auflösung (2304×1296 Pixel) und einer Bildwiederholfrequenz von 30 fps aufgezeichnet. Die Kameras besitzen keine Smart-Eigenschaften (keine Recheneinheit), sodass die Daten nicht gemäß einer dezentralen Umsetzung des Systems vorverarbeitet werden können. Es besteht jedoch die Möglichkeit, die Bilder via WLAN an einen Rechner bzw. eine zentrale Instanz zu übertragen, wo die Verarbeitung

stattfinden würde. Jedoch gilt dies nicht für die hier genutzten GPS-Logger, da diese keine verwendbare Kommunikationsschnittstelle besitzen. Eine Echtzeitprozessierung der Daten ist daher technisch bedingt in diesem Fall nicht möglich. Wie im vorherigen Experiment wird auch hier die Verzeichnung der Kamerabilder zunächst anhand der ermittelten internen Kameraparameter korrigiert. Die im Folgenden bestimmten Objektpositionen in Bildkoordinaten werden mit Hilfe von zwei vorberechneten Homographien (eine für jede Kamera) in Spielfeldkoordinaten transformiert. Die GPS-Daten werden in das gleiche Koordinatensystem transformiert.

Abbildung 6.8: Das Kamera-Setup für das Fußballanalyse-Experiment besteht aus zwei Kameras (A, B), deren Sichtfelder das Spielfeld nahezu komplett abdecken (inkl. Überlappung in der Mitte des Spielfelds).

Objektdetektionen und Referenzdaten

Die Referenzdaten für die Evaluation des ersten Teilexperiments werden analog zum vorherigen Experiment mit dem gleichen Verfahren erzeugt. Da dieses Mal jedoch andere Kameras verwendet werden und die Videos mit einer höheren Auflösung aufgezeichnet werden, müssen die Parameter angepasst werden. Die Mindestfläche wird auf $A_{bbox,min} = 300\ Pixel$ erhöht. Das maximale Verhältnis von Breite zu Höhe des Begrenzungsrechtecks wird auf $w/h_{max} = 1{,}0$ erhöht, weil die Spieler gelegentlich auch Posen einnehmen, bei denen sie ihre Extremitäten von sich strecken (z.B. beim Torschuss). In Tabelle 6.4 werden die Ergebnisse der Spielerdetektion gezeigt. In diesem Szenario treten deutlich mehr Detektionen auf. Dies liegt hauptsächlich an der größeren Anzahl an Objekten in der Szene. Über die Beobachtungsdauer von ungefähr einer Minute (301 Frames bei 5 Hz) werden insgesamt 5211 Detektionen (ca. 18 Detektionen pro Frame) ermittelt. Darunter sind 555 Fehl- (ca. 10,7 %) bzw. 639 Mischdetektionen (ca. 12,2 %), die nicht eindeutig zugewiesen werden können. In einem manuellen Labeling-Prozess werden die Detektionen den involvierten Spielern zugeordnet. Dabei werden aber nur die Spieler berücksichtigt, die mit einem GPS-Logger ausgestattet worden sind. Die übrigen Spielerdetektionen (des anderen Teams) werden unter „Andere Spieler“ zusammengefasst. Der Schiedsrichter wird gesondert betrachtet.

Tabelle 6.4: Die Ergebnisse der Objektdetektion im Fußballanalyse-Experiment

	Phase 1	**Phase 2**	**Phase 3**	**Gesamt**
Frames	100	100	101	301
Detektionen	1573	1872	1766	5211
Fehldetektionen	186	170	199	555
Mischdetektionen	138	309	192	639
Andere Spieler	458	669	574	1701
Schiedsrichter	52	56	60	168
Spieler 1 (Torhüter)	87	64	86	237
Spieler 2	83	42	62	187
Spieler 7	66	60	66	192
Spieler 9	54	67	36	157
Spieler 10	48	86	56	190
Spieler 11	50	71	53	174
Spieler 12	84	81	100	265
Spieler 13	61	62	82	205
Spieler 21	60	29	67	156
Spieler 22	73	32	61	166
Spieler 23	73	74	72	219
Objektdetektionen	739	668	741	2148

Mit insgesamt ca. 22,9 % ist der Anteil der Detektionen, die beim Tracking nicht verwendet werden können, erwartungsgemäß deutlich höher als beim vorangegangenen 2-Personen-Experiment. Obwohl die relevanten Objekte trotzdem noch in durchschnittlich ca. 65 % der Kamerabilder detektiert werden, ist hier ein gewisses Verbesserungspotenzial gegeben, da die Gesamtanzahl der Detektionen die Laufzeit des Ansatzes bedingt (siehe Abschnitt 4.5). Bei der Untersuchung der einzelnen Phasen zeigen die ermittelten Zahlen verschiedene Trends. Die Anzahl der Fehldetektion bleibt nahezu unverändert, was daran liegt, dass sich die Szene, insbesondere der Hintergrund, in der relativ kurzen Zeit nur sehr wenig verändert. Die Zahl der Mischdetektionen verhält sich wie erwartet: von der ersten auf die zweite Phase steigt sie, von der Zweiten auf die Dritte hingegen nimmt sie wieder ab. Hierbei ist auch zu berücksichtigen, dass ein gewisser Teil der Mischdetektionen auf Grund ihrer Ausmaße bereits herausgefiltert worden ist. Die tatsächliche Zahl ist demnach höher. Des Weiteren sind die unterschiedlichen Werte der einzelnen Spieler auffällig. Die Spieler, die sich eher im Vordergrund, also näher zur Kamera (z.B. die Spieler 2, 3 oder 13), oder etwas abseits vom Spielgeschehen aufhalten, wie z.B. der Libero (12) bzw. der Torwart (1) in den Phasen 1 und 3, werden zuverlässiger detektiert. Die Spieler in der Spielfeldmitte oder auf der gegenüberliegenden Seite werden weniger oft erkannt. Dies begründet sich dadurch, dass diese Spieler entsprechend häufiger durch andere verdeckt sind und dass sie teilweise nach den morphologischen Operationen der Detektionsregionen nicht mehr die benötigte Detektionsfläche $A_{bbox,min}$ aufweisen. Eine Reduzierung der Operationen würde dieses Problem etwas verringern. Daraus würden allerdings gleichzeitig mehr Mischdetektionen resultieren, da durch die morphologischen Operationen nahe beieinanderstehende Spieler getrennt werden. Eine Verringerung von $A_{bbox,min}$ hätte wiederum zur Folge, dass deutlich mehr Fehldetektionen verarbeitet werden müssten.

Durchführung und Ergebnisse des Objekt-Trackings

Für das Generierung der Ergebnisse für die ausgewählte Szene ist die Vorgehensweise analog zu der im vorherigen Experiment. Auch hier wird die detektionsbasierte Variante einmal mit GPS-Unterstützung und einmal ohne angewendet. Die Ergebnisse werden daraufhin anhand der Referenzdaten evaluiert. Zur Evaluation des zweiten Teilexperiments werden die dem Spieler zuge-

ordneten Detektionen erst im Nachhinein manuell mit Hilfe des Labeling-Tools überprüft. Zudem werden die erfasste und GPS-Trajektorie anhand verschiedener Parameter verglichen. So können bspw. signifikant unterschiedliche Bewegungsrichtungen oder Distanzen zwischen korrespondieren Trajektoriepunkten, die größer als die typische GPS-Ungenauigkeit sind, einen Hinweis darauf geben, dass zu diesen Zeitpunkten fälschlicherweise ein anderes Objekt verfolgt wird. Da das Objekt-Tracking in den Versuchen ohne GPS-Unterstützung relativ schnell den Zielspieler verliert und daher nur sehr geringe Werte für die Sensitivität zu erwarten sind, wird dieses im Weiteren nicht weiter ausgeführt. Ein Vergleich findet hier demnach nicht statt. Die Parameter werden jeweils wie folgt angesetzt: $\sigma_{GPS} = 7\ m$, $v_{max} = 12\ \frac{m}{s}$, $P_{pen} = 0{,}1$. Ein Unterschied zum vorherigen Experiment besteht darin, dass die Anzahl an Detektionen pro Bild deutlich höher ist. Dies hat zur Folge, dass die Zahl an möglichen Zuständen, die durch die Variationen der Detektionen gebildet werden (siehe Abschnitte 4.4.1 und 4.4.2), so hoch ist, dass ein gemeinsames Modell für alle beobachteten Objekte nicht handhabbar ist. Daher wird wie in Abschnitt 4.5 beschrieben für jedes Objekt ein eigenes Modell verwendet. Jedes Objekt wird demnach eigenständig verfolgt. Das bedeutet, dass Mehrfachzuordnungen von einer Detektion zu mehreren Objekten nicht ausgeschlossen werden.

Die Resultate der Auswertungen beider Teilexperimente sind in Tabelle 6.5 dokumentiert. In den ersten drei Bildern der Abbildung 6.9 wird die ausgewählte Szene des ersten Teilexperiments veranschaulicht. Im ersten Bild ist die Situation vor dem Angriff zu sehen (Phase 1), die noch wenig Verdeckungen beinhaltet. Im Zweiten (Phase 2) und Dritten (Phase 3) werden entsprechend Zeitpunkte während (viele Verdeckungen) und nach dem Angriff gezeigt. In der Übersicht in Abbildung 6.9 (unten) sind die erfassten Trajektorien der Spieler und deren GPS-Trajektorien zum letzten Zeitpunkt dargestellt. In Abbildung 6.10 wird die Trajektorie des ausgewählten Spielers (Nummer 13) aus dem zweiten Teilexperiment gezeigt.

Tabelle 6.5: Die Ergebnisse des Objekt-Trackings im Fußballanalyse-Experiment

	m. GPS-Unterstützung	**o. GPS-Unterstützung**
Teilexperiment 1: Phase 1		
Objektdetektionen (n. zuzuordnen)	739 (138)	739 (138)
Sensitivität (*recall*) (%)	662 (89,6 %)	401 (54,3 %)
Falsch-negativ-Rate (*misses*) (%)	77 (10,4 %)	338 (45,7 %)
Teilexperiment 1: Phase 2		
Objektdetektionen (n. zuzuordnen)	668 (309)	668 (309)
Sensitivität (*recall*) (%)	362 (54,2 %)	146 (21,9 %)
Falsch-negativ-Rate (*misses*) (%)	306 (45,8 %)	522 (78,1 %)
Teilexperiment 1: Phase 3		
Objektdetektionen (n. zuzuordnen)	741 (192)	741 (192)
Sensitivität (*recall*) (%)	646 (87,2 %)	276 (37,2 %)
Falsch-negativ-Rate (*misses*) (%)	95 (12,8 %)	465 (62,7 %)
Teilexperiment 1: Gesamt		
Objektdetektionen (n. zuzuordnen)	2148 (639)	2148 (639)
Sensitivität (*recall*) (%)	1670 (77,7 %)	823 (38,3 %)
Falsch-negativ-Rate (*misses*) (%)	478 (22,3 %)	1325 (61,7 %)
Teilexperiment 2: Spieler 13 (22,5 Minuten)		
Zuordnungen	5442	-
Sensitivität (*recall*) (%)	4147 (76,2 %)	-
Falsch-negativ-Rate (*misses*) (%)	1295 (23,8 %)	-

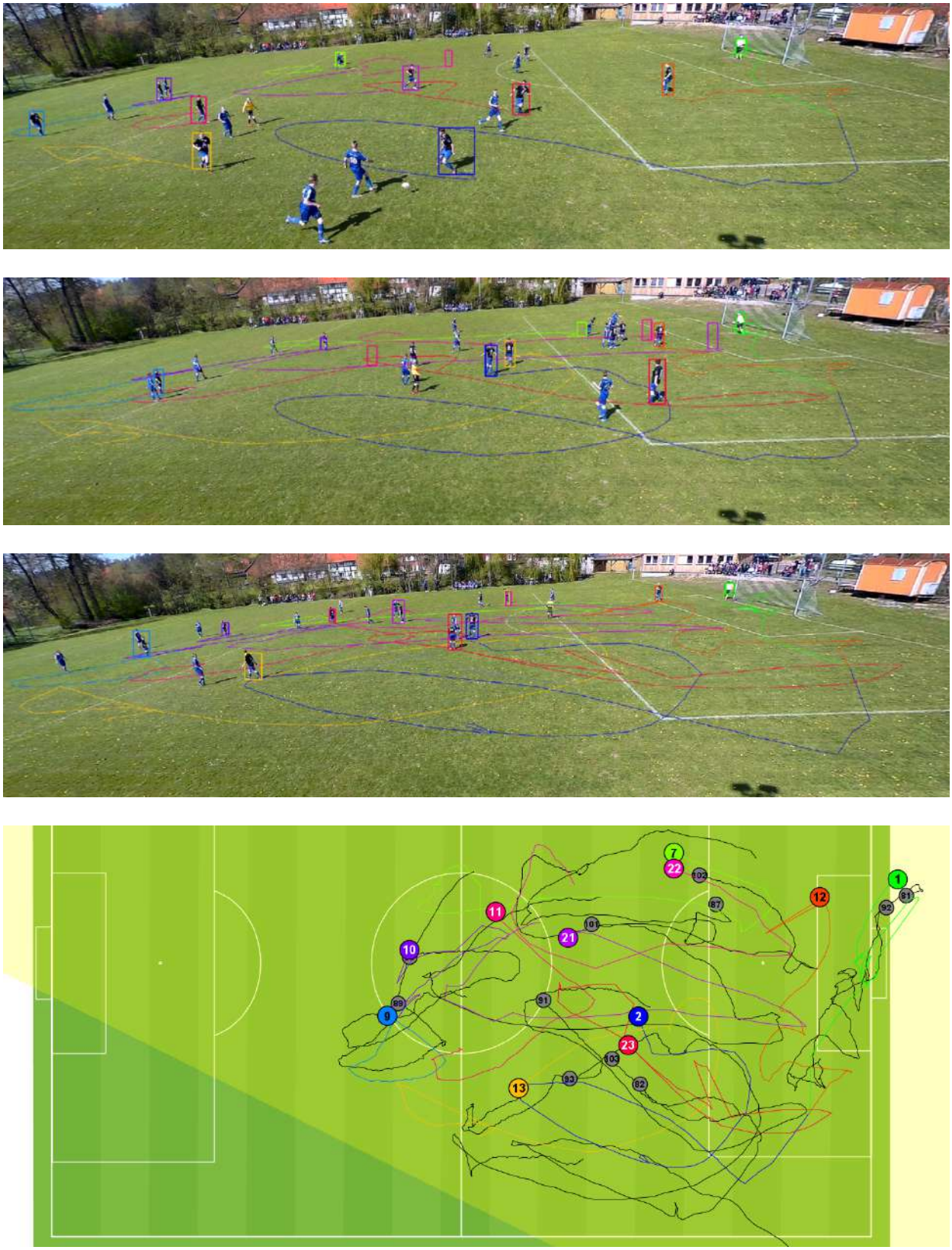


Abbildung 6.9: In den ersten drei Bildern werden die unterschiedlichen Phasen der ausgewählten Szene, in der die schwarz-blau gekleideten Spieler verfolgt werden, gezeigt: die Situation zu Beginn des Angriffs der ganz in blau gekleideten Mannschaft, die während des Angriffs und die nach dem Angriff. Im unteren Bild ist eine Übersicht über die erfassten (bunt) und die GPS-Trajektorien (schwarz) gegeben. Die Nummern entsprechen den Rückennummern der Spieler. Zur Berechnung der Nummern der GPS-Trajektorien (grau) werden die Spielernummern um den Wert 80 erhöht.

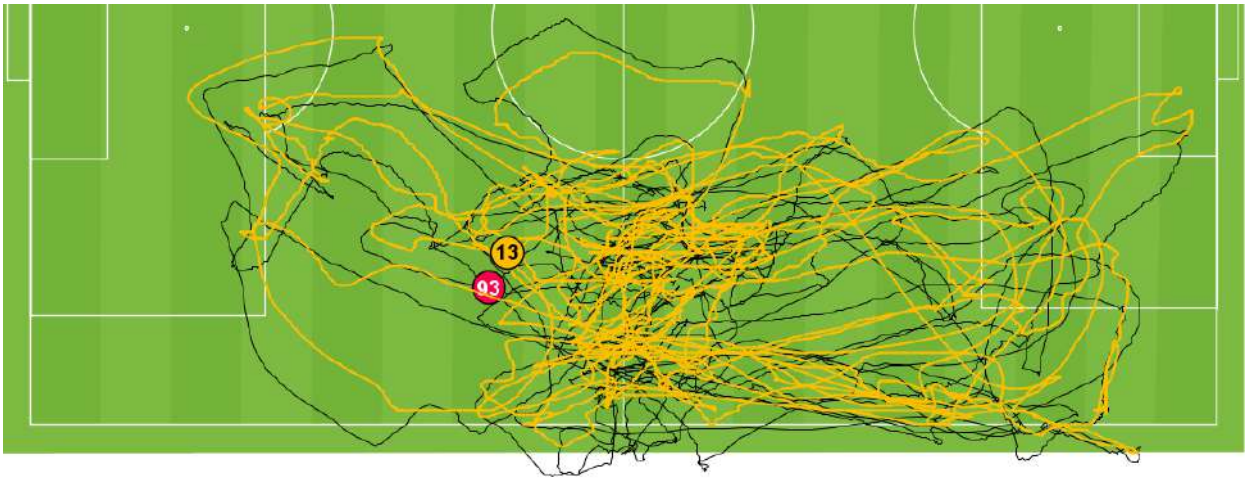


Abbildung 6.10: Die resultierende (orange) und die GPS-Trajektorie (schwarz) des ausgewählten Spielers (13), die in dem Beobachtungszeitraum von 22,5 Minuten in der ersten Halbzeit mit Hilfe der rasterbasierten Modellierung ermittelt wird.

Diskussion

Die Ergebnisse des ersten Teilexperiments belegen, dass das Objekt-Tracking in den einzelnen Phasen erwartungsgemäß unterschiedlich gut funktioniert. In der Anfangsphase der Szene mit relativ wenig Verdeckungen können die Spieler gut verfolgt werden. Die Sensitivität liegt dort entsprechend bei 89,6 %. In der mittleren Phase der Szene, in der sich viele Spieler auf engem Gebiet befinden, ist die Zahl an Verdeckungen allerdings hoch. Die Verfolgung der Spieler ist schwierig, da die entsprechenden Detektionen fehlen. Die Sensitivität fällt deutlich auf 54,2 %. In der Endphase der Szene, in der sich die Spieler wieder auf dem Feld verteilen und die Situation sich auflöst, nimmt die Sensitivität wieder zu. Sie liegt dann bei 87,2 %. Hieraus geht hervor, dass die Verfolgung korrekt fortgeführt wird, obwohl einige Spieler zeitweise nicht detektiert werden konnten. Dieses wird durch das kombinierte Verfahren ermöglicht, in dem die GPS-Informationen genutzt werden, um die Objekte „wiederzuerkennen“. Die Zahlen des Durchlaufs ohne GPS-Unterstützung sind hier niedriger, was neben den vielen Verdeckungen auch an der einheitlichen Kleidung der Spieler liegt. Lediglich die Verfolgung des Torhüters auf Grund seines herausstechenden Trikots und die der Spieler, die sich abseits vom Geschehen befinden, funktioniert recht zuverlässig. Letztere müssen in der Regel nur von ihrem Gegenspieler unterschieden werden. Im zweiten Teilexperiment ergeben sich nach der Überprüfung der 5442 zugeordneten Detektionen des ausgewählten Spielers eine Sensitivität von insgesamt 76,2 % und eine Falsch-negativ-Rate von 23,8 %. Die Kennwerte sind vergleichbar mit denen aus dem kürzeren Teilexperiment und belegen somit, dass das Objekt-Tracking auch über einen längeren Zeitraum ähnlich gute Ergebnisse liefert.

Bei der Analyse der Trajektorie-Parameter (siehe Abbildung 6.11) liegen die Distanzwerte (d) allesamt unter 5 m und sind daher nicht auffällig. Die Differenzen in den Bewegungsrichtungen (dh) weisen jedoch größere Unterschiede auf. Bei einer näheren Betrachtung der entsprechenden Intervalle stellt sich heraus, dass es sich um Szenen handelt, in denen sich viele Spieler auf engem Gebiet befinden und damit die Anzahl an Verdeckungen hoch ist. Der relevante Spieler ist hier ebenfalls zeitweise verdeckt, sodass es entweder zu Dummy- oder Fehlzuordnungen kommt. Zur Ermittlung eines Maßes, das eine grobe Abschätzung der Korrektheit der Zuordnung erlaubt, wird der Anteil der Bewegungen ermittelt, deren Abweichung unter 60 Grad liegt. In diesem Beispiel liegt dieser Anteil bei ca. 85,5 %. Dieser Wert ist jedoch recht unsicher, da bei dieser Betrachtungsweise u.a. die Fälle nicht erkannt werden, in denen ein fälschlicherweise verfolgtes Objekt sich in eine ähnliche Richtung wie das eigentliche Objekt bewegt.

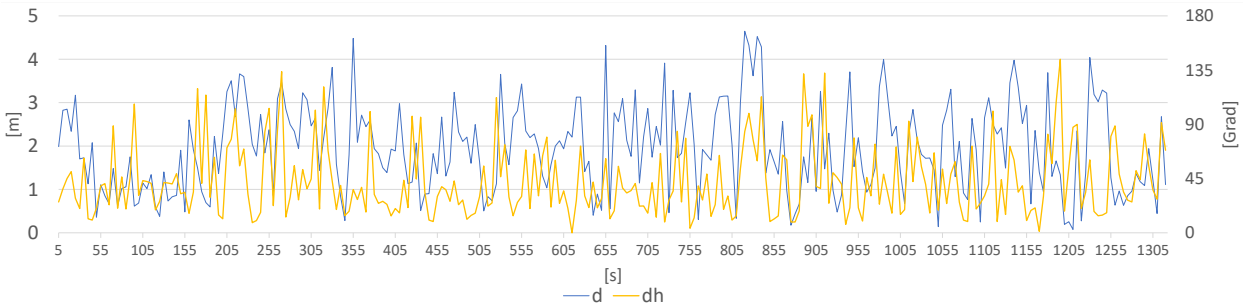
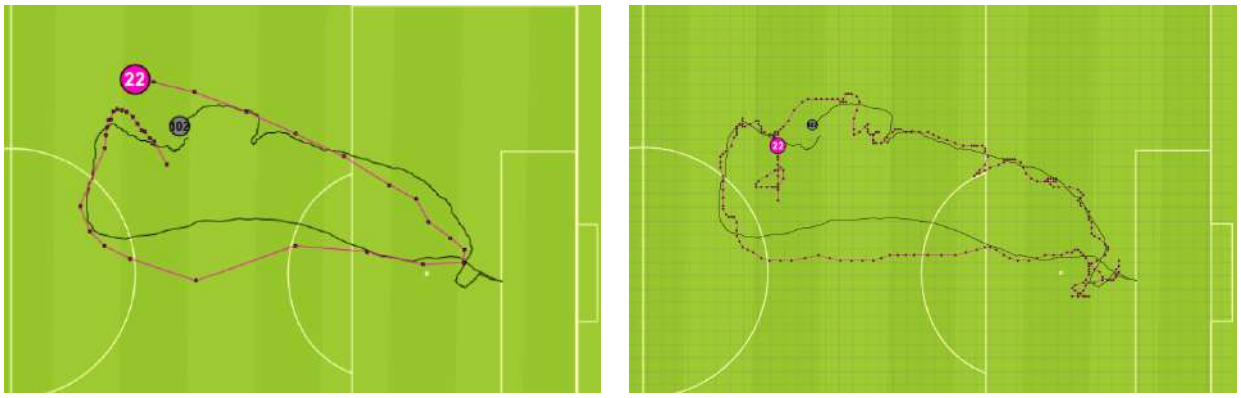


Abbildung 6.11: Die Unterschiede bzgl. der Bewegungsrichtungen (dh) und der Abstand (d) zwischen der GPS- und der erfassten Trajektorie können Hinweise auf ein inkorrektes Objekt-Tracking geben.

Die Probleme bei der Detektion der weit entfernten Spieler macht sich auch beim Tracking bzw. bei der Generierung der Trajektorien mit Hilfe des detektionsbasierten Ansatzes bemerkbar. Weil die Spieler seltener detektiert werden, und damit ihre Position ebenfalls seltener aktualisiert werden kann, müssen ihre Bewegungen häufig über längere Zeit interpoliert werden. Dies zeigt sich in den zum Teil geradlinigen Verläufen der Trajektorien (siehe Abbildung 6.12a). Zur Lösung dieses Problems bieten sich verschiedene Möglichkeiten: Zum einen können mehr Kameras, z.B. auch auf der gegenüberliegenden Seite, installiert werden, sodass mehr Perspektiven verfügbar sind und die Distanz zu einer Kamera verringert wird. Dies führt jedoch zu höheren Kosten des Systems und zu einer aufwändigeren Datenverarbeitung. Zum anderen kann hier die rasterbasierte Variante dieses Verfahrens verwendet werden, die auch die GPS-Informationen zur Positionsbestimmung der Objekte nutzt. Auf diese Weise können die Lücken in der Trajektorie durch die GPS-Positionen geschlossen werden (Abbdildung 6.12b).



(a) *(b)*

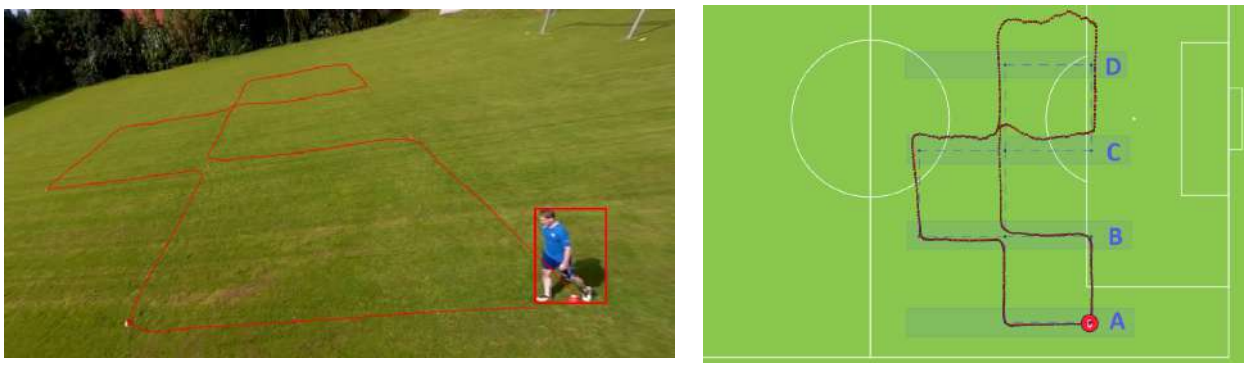
Abbildung 6.12: Bei der Verfolgung eines häufig verdeckten Spielers (22, pink) mit Hilfe des detektionsbasierten Ansatzes muss durch die vielen fehlenden Detektionen über längere Strecken interpoliert werden. b) Bei der rasterbasierten Variante werden entsprechend die GPS-Informationen verwendet, um die Lücken zu füllen. In grau wird die GPS-Trajektorie dargestellt.

Ein weiterer Problemfall in diesem Szenario ist die gleiche Kleidung und damit ein sehr ähnliches Äußeres der Spieler. Dieses kann zu Verwechslungen führen und beeinflusst die Ergebnisse. Letztere könnten durch die Hinzunahme weiterer Merkmale, z.B. der Rückennummer der Spieler, verbessert werden. Die Nummer wäre ein eindeutiges Merkmal, jedoch ist die Erkennung nicht trivial, da die Nummer nicht immer zu sehen ist und durch ein unebenes Trikot verzerrt sein kann. Dies führt zu einer gewissen Unsicherheit bei der Identifikation (siehe das Beispiel in Abbildung 6.13), sodass eine gewisse Verwechslungsgefahr besteht, die sich wiederum kontraproduktiv auf das Tracking auswirken könnte.

Abbildung 6.13: Im Fußballanalyse-Szenario können die Rückennummern der Spieler das Tracking unterstützen. Jedoch ist deren Erkennung nicht immer trivial. Die tatsächlichen Nummern von links nach rechts: 2, 10, 13, 13, 13.

6.4 Geometrische Genauigkeit der Trajektorien

In diesem Experiment wird die geometrische Genauigkeit der Objekt-Trajektorien analysiert. Die Genauigkeit der Trajektorien hängt bei der detektions- und der rasterbasierten Modellierung von verschiedenen Faktoren ab. Bei der detektionsbasierten Variante werden die Trajektorien ausschließlich auf Grundlage der Positionen, die aus den Kameradetektionen abgeleitet werden, generiert. Die via GPS ermittelten Positionen besitzen hier keinen Einfluss. Im Gegensatz dazu werden die Objektpositionen bei der rasterbasierten Variante aus den Messungen beider Sensoren berechnet. Hierbei spielen die angenommenen Unsicherheiten σ_{Kamera} und σ_{GPS} sowie die Rasterzellengröße eine Rolle. Zudem ist in beiden Varianten die Bildauflösung wichtig, sodass diese in einem zusätzlichen Durchlauf ebenfalls verändert wird. Zum Vergleich der beiden Modellierungsvarianten und zur Ermittlung des Einflusses der zuvor genannten Faktoren werden die jeweils resultierenden Trajektorien sowie die GPS-Trajektorien untersucht. Zu diesem Zweck werden sie analog zu Coutts und Duffield (2010) und Gray u. a. (2010) mit einer Referenzstrecke als Ground Truth verglichen, indem punktweise die kürzeste Distanz zu dieser berechnet wird. Die Unsicherheiten und die Rasterzellengröße werden dabei variiert. Zur Bestimmung der Ground Truth-Strecke ist ein Parcours definiert worden, auf dem sich ein Objekt, in diesem Fall eine Person, bewegen soll. Im Kamerabild in Abbildung 6.14 ist dieser an den roten Markierungen zu erkennen. Die Ground Truth-Strecke ist in 6.14b in blau dargestellt. Mit A, B, C und D sind vier Distanzbereiche zur Kamera gekennzeichnet, in denen sich das Objekte horizontal durch das Bild bewegt. Die vertikalen Bewegungen werden nicht unterteilt. Zudem werden Sensorunsicherheiten verändert, um auch deren Einflüsse bestimmen zu können.



(a) *(b)*

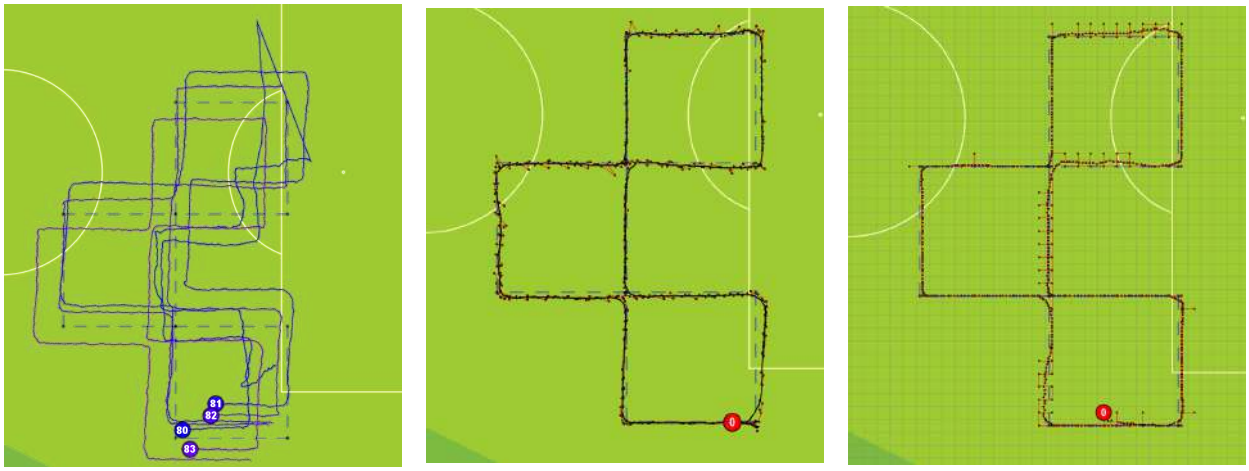
Abbildung 6.14: a) In einem Experiment zur Genauigkeitsanalyse bewegt sich ein Objekt entlang einer vordefinierten Referenzstrecke (gekennzeichnet durch rote Markierungen). In rot wird die ermittelte Trajektorie des Objekts dargestellt. b) Der Parcours (blau gestrichelt) und die generierte Trajektorie (rot).

Das Setup der Sensoren gestaltet sich wie folgt: Das Objekt wird mit mehreren GPS-Loggern ausgestattet und eine Kamera beobachtet die Szene. Die Bilder besitzen dabei eine Auflösung

von 2880 × 2160 Pixel bzw. 1280 × 720 Pixel (HD) für den variierten Durchlauf. Die baugleichen Logger werden an unterschiedlichen Positionen befestigt, um festzustellen, ob die Trageposition einen Einfluss auf die Messung besitzt. Die Tragepositionen befinden sich an beiden Oberarmen (*GPS 82*, *GPS 83*), auf dem Rücken zwischen den Schulterblättern (*GPS 80*) und vorne an der Hose (*GPS 81*). Die Sensorunsicherheiten werden auf $\sigma_{GPS} = 5\ m$, $\sigma_{Kamera} = 1\ m$ gesetzt. Die relativ hohe Kameraunsicherheit resultiert daraus, dass die Person sich zeitweise recht weit von der Kamera wegbewegt. Es wird angenommen, dass in dieser Zeit die Ungenauigkeit der Objektpositionierung größer ist.

Ergebnisse

In den Abbildungen 6.15 sind die aufgezeichneten Trajektorien zu sehen. In a) sind die unterschiedlichen GPS-Trajektorien dargestellt, in b) und c) die jeweiligen Ergebnisse der beiden Modellierungsvarianten. Zudem werden die mit Hilfe des Mittelwert-Filters geglätteten Versionen der Trajektorien gezeigt. Der Plot in Abbildung 6.16 zeigt den Verlauf der Abweichungen zur Referenzstrecke. In Tabelle 6.6 werden bereichsweise die mittlere Abweichung bzw. der RMSE (*root mean square error*) zusammengefasst. Da hier mehrere GPS-Trajektorien für nur ein Objekt zur Verfügung stehen, wird diejenige für den kombinierten Ansatz verwendet, die am Nächsten an der Referenz liegt. In diesem Fall ist das die Trajektorie, bei der der Logger am linken Oberarm (*GPS 82*) befestigt wurde.



(a) *(b)* *(c)*

Abbildung 6.15: Die Ergebnisse des Experiments: Die GPS-Trajektorien (a), die Trajektorien aus der detektionsbasierten (b) und der rasterbasierten Modellierung (c). Bei den letzten beiden ist in rot die berechnete und in schwarz die geglättete Trajektorie dargestellt.

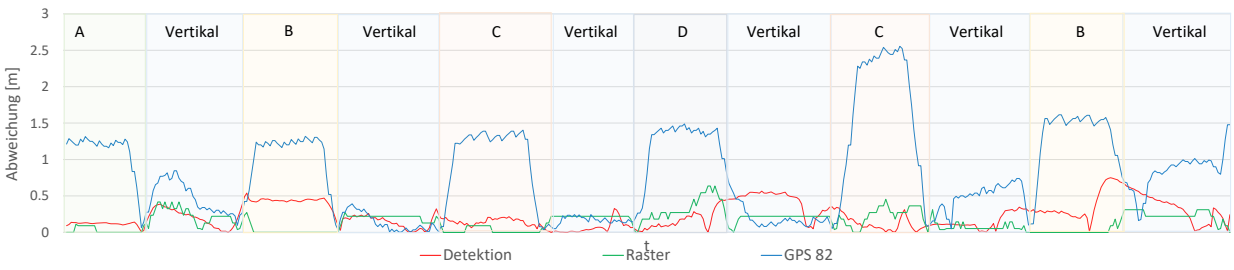
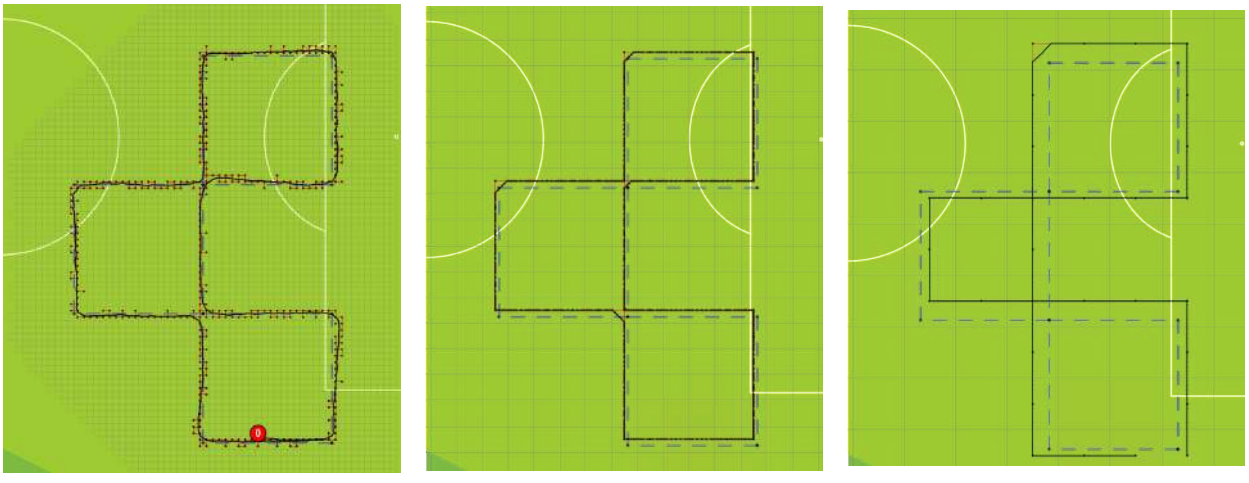


Abbildung 6.16: Der Verlauf der Abweichungen der Trajektorien zur Referenzstrecke. Hier wird mit der GPS 82-Trajektorie nur die Beste der GPS-Trajektorien dargestellt. Die farblichen Bereiche im Diagramm symbolisieren die unterschiedlichen Distanzbereiche (siehe Abbildung 6.14b) zur Kamera.

Tabelle 6.6: Die Abweichungen der erfassten Trajektorien insgesamt und für die einzelnen Abschnitte des Parcours.

Trajektorie	**Mittlere Abweichung [m] / RMSE [m]**					
	Gesamt	**A**	**B**	**C**	**D**	**Vertikal**
GPS 82 (linker Arm)	0.80 / 1.01	1.03 / 1.10	1.01 / 1.08	1.05 / 1.14	1.28 / 1.30	0.40 / 0.50
GPS 83 (rechter Arm)	1.49 / 1.63	1.54 / 1.64	1.36 / 1.42	1.19 / 1.34	1.34 / 1.39	1.75 / 1.88
GPS 80 (Rücken)	1.72 / 1.99	2.61 / 2.80	1.23 / 1.28	0.96 / 1.16	2.00 / 12.33	1.88 / 2.15
GPS 81 (Hose)	1.21 / 1.52	1.26 / 1.33	1.39 / 1.51	1.79 / 2.08	2.00 / 2.17	0.70 / 0.85
Kamera + GPS 82 (rasterbasiert, 0.5 m)	0.18 / 0.22	0.14 / 0.15	0.16 / 0.17	0.08 / 0.11	0.30 / 0.32	0.19 / 0.24
Kamera + GPS 82 (rasterbasiert, 1 m)	0.15 / 0.20	0.04 / 0.07	0.03 / 0.07	0.03 / 0.07	0.33 / 0.37	0.20 / 0.22
Kamera + GPS 82 (rasterbasiert, 2 m)	0.37 / 0.39	0.47 / 0.48	0.45 / 0.46	0.47 / 0.47	0.48 / 0.48	0.28 / 0.28
Kamera + GPS 82 (rasterbasiert, 4 m)	0.86 / 0.96	0.48 / 0.49	1.25 / 1.28	0.50 / 0.51	1.30 / 1.35	0.89 / 0.95
Kamera + GPS 82 (detektionsbasiert)	0.23 / 0.28	0.12 / 0.13	0.40 / 0.41	0.14 / 0.17	0.22 / 0.26	0.23 / 0.29
Kamera + GPS 82 (detektionsbas., HD)	0.32 / 0.46	0.12 / 0.13	0.16 / 0.18	0.86 / 0.79	0.39 / 0.40	0.30 / 0.48

Da bei der rasterbasierten Variante die geometrische Genauigkeit auch von der Rasterauflösung abhängt, wird diese noch einmal gesondert untersucht, indem die Auflösung variiert wird. Die resultierenden Trajektorien sind in Abbildung 6.17 dargestellt. Die entsprechenden Abweichungen sind ebenfalls in Tabelle 6.6 zu finden.



(a) *(b)* *(c)*

Abbildung 6.17: Die Ergebnisse der rasterbasierten Modellierung mit Rasterauflösung von 0,5 m (a), 2 m (b) und 4 m (c) (1 m: siehe Abbildung 6.15c).

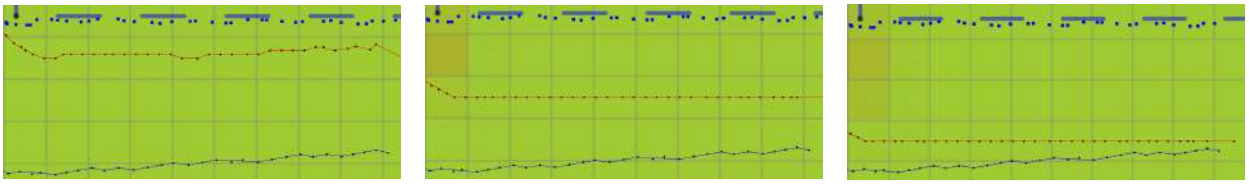
Diskussion

Bei der Bewertung der Ergebnisse ist zu beachten, dass die erfassten Trajektorien mit einer Referenzstrecke und nicht mit der realen Objekt-Trajektorie verglichen werden. Dieses hat zur Folge, dass es dort bereits gewisse Abweichungen gibt, da nicht gewährleistet ist, dass sich das Objekt exakt auf der vorgegebenen Strecke bewegt hat. Die tatsächlichen Fehler sind daher etwas geringer. Zudem handelt es sich bei den ermittelten Abweichungen um die kürzeste Distanz zur Strecke. Eine Abweichung in Richtung des Streckenverlaufs wird nicht berücksichtigt. Der Fehler wird dadurch wiederum etwas unterschätzt.

Folgend der Motivation, die zu diesem Ansatz führt, müsste die Genauigkeit der Objektlokalisierung mit Hilfe eines videobasierten Verfahrens deutlich genauer als die eigenständige GNSS-basierte Variante sein. Die Trajektorien in den Abbildungen, der zeitliche Verlauf der Abweichungen und die Kennwerte in der Tabelle machen dieses deutlich. Die mit Hilfe dieses Ansatzes generierten Trajektorien weisen eine mittlere Abweichung von unter 25 cm auf. Wird eine geringere Bildauflösung (HD) gewählt, nimmt der mittlere Fehler etwas zu und liegt bei ca. 32 cm. Die Abweichung der GPS-Trajektorien liegt bei ca. 0,5 bis 2 m. In diesem Zusammenhang ist anzumerken, dass die GPS-Abweichungen verhältnismäßig gut sind. In weiteren (hier nicht ausgeführten) Experimenten lagen diese Werte eher im Bereich von 3 - 10 m, was der typischen GPS-Ungenauigkeit entspricht. Die Trajektorien der Logger, die an den Armen und an der Hose befestigt worden sind, weisen einen systematischen Versatz auf, die des Loggers am Rücken, ist zudem noch verzerrt. Die Abweichungen sind dort teilweise sehr groß. Ein Grund hierfür könnte eine ungünstige Anbringung unter der Kleidung und/oder eine kurzzeitige Abschattung durch den Körper sein. Die Genauigkeiten und die unterschiedlichen Verformungen der GPS-Trajektorien im Verhältnis zur Referenz lassen darauf schließen, dass die Trageposition einen Einfluss auf die Genauigkeit besitzt. Sie muss daher so gewählt werden, dass Abschattungen sowie häufige Erschütterungen vermieden werden. Eine Befestigung am Arm bietet sich hier an.

Beim Vergleich der Trajektorien der beiden Modellierungsvarianten muss berücksichtigt werden, dass die rasterbasierte Variante mit der Rasterauflösung und mit dem Verhältnis der Sensorungenauigkeiten zwei weitere Einflussfaktoren besitzt, die zuerst untersucht werden müssen. Daher wird die Auflösung des Rasters variiert. Die Auflösungen sind 0,5 m, 1 m, 2 m und 4 m. Aus den Werten der Tabelle 6.6 geht hervor, dass die Genauigkeit von der gewählten Zellgröße abhängt. Je höher die Auflösung gewählt wird, desto kleiner sind die Abweichungen. Dieses zeigt sich in Abbildung 6.17 deutlich. Ab einer Auflösung von 0,5 m ist jedoch keine Verbesserung zu erkennen. Dieses begründet sich dadurch, dass als Objektposition der Zellmittelpunkt verwendet wird. Sollte sich das Objekt tatsächlich auf dem Rand der Zelle befinden, entsteht allein hierdurch eine Abweichung von der halben Zellbreite/-höhe (bzw. das $\sqrt{2}$-fache der halben Zellbreite/-höhe, wenn sich das Objekt in der Ecke einer Zelle befindet). Neben der Genauigkeit lassen sich auch Veränderungen in der Geometrie der Trajektorie beobachten. Mit steigender Auflösung erhöht sich der Detailgrad, mit der die Objektbewegung erfasst wird. Eine geringere Auflösung bewirkt, dass die Bewegungen gradliniger und rechtwinkliger werden. Dieses ist in den Abbildungen 6.17b und 6.17c gut zu erkennen. Die Angaben über die Sensorungenauigkeiten steuern den Einfluss der jeweiligen Messung auf die berechnete Objektposition. In diesem Experiment wird eine GPS-Ungenauigkeit von 5 m (in x- und y-Richtung) und eine Kamera-Ungenauigkeit von 1 m (ebenfalls in x- und y-Richtung) angenommen. Dies hat zur Folge, dass die resultierende Trajektorie sich stärker an den Kameradetektionen orientiert. In Abbildung 6.18 werden die Ungenauigkeiten variiert, was dazu führt, dass sich der Einfluss der beiden Messungen verändert.

Um die Ergebnisse der beiden Modellierungsvarianten zu vergleichen, wird daher die Auflösung von 0,5 m herangezogen, welche die höchste Genauigkeit besitzt. Die Ungenauigkeiten werden unverändert gelassen, da sie die Realität recht gut widerspiegeln. In diesem Fall besitzt die ras-



(a) *(b)* *(c)*

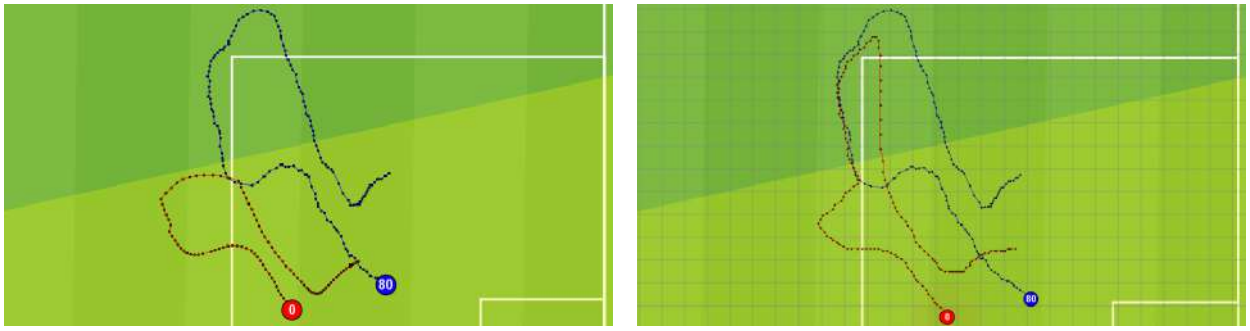
Abbildung 6.18: Das Verhältnis der Sensorunsicherheiten beeinflusst das Ergebnis. Ist das Verhältnis (Kamera-GPS) 5:1 (a) liegt die Trajektorie näher an den Kameradetektionen. Ist es ausgeglichen (1:1) liegt sie in der Mitte (b). Bei einem Verhältnis von 1:5 verläuft sie entsprechend näher an der GPS-Trajektorie (c). Die kleinen blauen Kästchen symbolisieren die Kameradetektionen.

terbasierte Variante die kleineren Abweichungen zur Referenz und entspricht daher besser der tatsächlichen Objektbewegung.

Um das Verhalten bei fehlenden Informationen zu untersuchen, d.h. wenn das beobachtete Objekt für einige Zeit nicht sichtbar ist, wird dieses Experiment in der Art verändert, dass das Blickfeld der Kamera künstlich eingeschränkt wird. Hierzu wird eine Bildmaske verwendet, die dafür sorgt, dass der in Abbildung 6.19a ausgegraute Bereich nicht berücksichtigt wird. Das Objekt kann dort entsprechend nicht detektiert werden. Die resultierenden Trajektorien verändern sich entsprechend der genutzten Modellierung. In den Abbildungen 6.19b und 6.19c sind die Ergebnisse zu sehen. Da bei der rasterbasierten Modellierung die Objektpositionen aus beiden Sensormessungen bestimmt werden, orientiert sich die Trajektorie bei nicht vorhandenen Kameradetektionen ausschließlich an den GPS-Messungen. Der Übergang zu diesem Verhalten ist durch die roten Markierungen in der Abbildung gekennzeichnet. Im Gegensatz zur detektionsbasierten Variante bildet sie die tatsächlich Objektbewegung besser nach. Dies sieht bei den Trajektorien der Szene des 2-Personen-Experiments, in der eine Person das Kamerasichtfeld verlässt, ähnlich aus (siehe Abbildung 6.20). Auch dort entspricht das Ergebnis der rasterbasierten Varianten eher der Realität.

(a) *(b)* *(c)*

Abbildung 6.19: Die Ergebnisse des veränderten Experiments mit künstlicher Verdeckung (ausgegraute Fläche) (a). Das Objekt ist dadurch für einige Zeit nicht sichtbar. Die Trajektorien (rot) aus der detektionsbasierten (b) und der rasterbasierten Modellierung (c) spiegeln die tatsächliche Objektbewegung unterschiedlich gut wieder. Die GPS-Trajektorie ist jeweils in blau dargestellt.



(a) *(b)*

Abbildung 6.20: Auch bei der entsprechenden Szene im 2-Personen-Experiment wird die tatsächliche Bewegung durch die rasterbasierte Modellierungsvariante besser erfasst. Die GPS-Trajektorie ist jeweils in blau dargestellt.

6.5 Laufzeit

Um die in Abschnitt 4.5 beschriebenen theoretischen Laufzeiten beider Modellierungsvarianten zu verifizieren, werden die tatsächlichen Laufzeiten unter Verwendung des Fußballanalyse-Experiment-Datensatzes gemessen. Die Gesamtlaufzeit in jedem Zeitschritt wird dabei noch einmal in die entsprechenden Anteile, die für die Objektdetektion, für das Tracking und für sonstige notwendige Berechnungen benötigt werden, unterteilt. Da der Berechnungsaufwand mit der Bildauflösung, der Rasterauflösung und der Anzahl an beobachteten Objekten von unterschiedlichen Faktoren abhängt, werden diese wie folgt variiert: Als Bildauflösungen werden die originale 2K-(2560×1440 Pixel) und eine heruntergerechnete HD-Auflösung (1280×720) verwendet. Die Rasterauflösungen für die rasterbasierte Variante sind 1 m, 2 m und 4 m. In den ersten Durchläufen wird ein einzelnes Objekt verfolgt. Im letzten Durchlauf wird die Anzahl auf elf Objekte angehoben. Da die genannten Faktoren die Laufzeit unabhängig voneinander beeinflussen, wird hier nicht jede mögliche Kombination untersucht, sondern jeweils an einem Beispiel gezeigt. Aus diesem Grund wird das Tracking der elf Objekte nur auf Basis der detektionsbasierten Variante durchgeführt. Die Ergebnisse der Messungen auf einem Rechner (i7-4720HQ mit 2,6 GHz, 16 GB RAM) ohne Grafikkartenunterstützung sind in Abbildung 6.21 dargestellt.

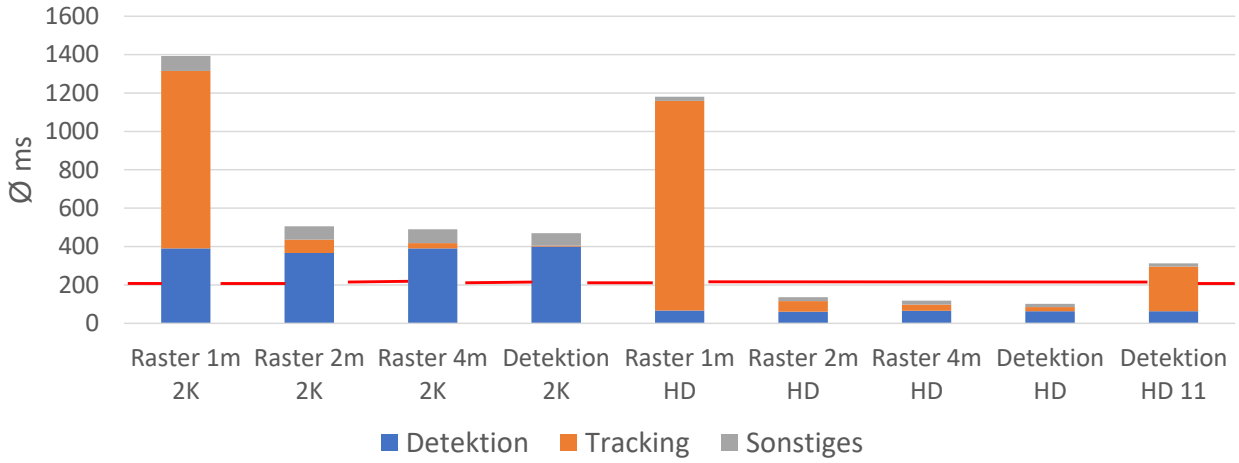


Abbildung 6.21: Die tatsächlichen Laufzeiten beider Modellierungsvarianten pro Zeitschritt unterteilt in Objektdetektion und Verfolgung sowie sonstige Berechnungen

Die gezeigten Werte sind Durchschnittswerte über den gesamten Messzeitraum. Mit der Zeit ist ein sehr geringer Anstieg zu erkennen, der jedoch hauptsächlich aus nicht relevanten Berechnungen resultiert. Da die GPS-Informationen mit 5 Hz zur Verfügung gestellt werden, ist für die Echtzeitfähigkeit des Systems eine Gesamtlaufzeit pro Zeitschritt unter 200 ms (rote Linie in Diagramm) zu erreichen. Wie in der Abbildung gezeigt wird, ist dies bei der aktuellen Systemkonfiguration nur bei Verwendung der HD-Auflösung in Kombination mit der detektionsbasierten bzw. der rasterbasierten Variante mit einer Auflösung von kleiner als 2 m möglich. In den übrigen Fällen sind entweder die Detektion oder die Verfolgung (oder Beide) zu ineffizient. Bei der Objektdetektion zeigt sich eine lineare Zunahme der Laufzeit mit der Bildauflösung. Es wird ungefähr die vierfache Zeit für die vierfache Auflösung benötigt. Dies lässt sich durch den gewählten pixelbasierten Ansatz begründen. Der Aufwand bei der Verfolgung ist abhängig von der Anzahl an möglichen Zuständen. Bei der rasterbasierten Modellierung ist er daher im Allgemeinen höher als bei der detektionsbasierten Variante. Durch die quadratische Komplexität der Modellierung erfordert eine vervierfachte Auflösung einen 16-fachen Berechnungsaufwand. Dieses wird bspw. deutlich, wenn die Durchläufe „Raster 1m 2K“ und „Raster 2m 2K“ miteinander verglichen werden. Wird die Anzahl der Objekte auf elf erhöht, so steigt entsprechend auch hier die Verarbeitungszeit (siehe Durchlauf „Detektion HD 11“). Würde das Tracking mittels Detektionsvariationen in nur einem Modell für sämtliche Objekte realisiert, wäre der Aufwand sehr viel höher. Bei ca. 18 Detektionen pro Frame (siehe Abschnitt 6.4) ($n = 11$ Objekte, $l = 18$ Detektionen) wären $M = \frac{l!}{(l-n)!} \approx 1$, 3 *Billionen* Variationen bzw. Zustände zu verarbeiten. Da in dieser Untersuchung die elf Objekte individuell verfolgt werden, d.h. es wird ein Modell für jedes Objekt erzeugt, nimmt der Aufwand jedoch nur linear zu. Diese individuellen Modelle erlauben eine parallele Verarbeitung, die bei einem Rechner bzw. Rechenkern pro Objekt theoretisch in Echtzeit erfolgen könnte. In den übrigen Fällen, in denen die Detektion der Objekte im Bild den größten Aufwand darstellt, könnte die Berechnung durch Verwendung einer Grafikkarte deutlich beschleunigt und die erforderliche Zeit auf einen Bruchteil reduziert werden (OpenCV team, 2017). Auf diese Weise wäre auch hier eine Echtzeitverarbeitung möglich.

6.6 Fazit

Die vorangegangene Evaluation zeigt, dass der vorgestellte Ansatz zur Erfassung von Trajektorien qualitativ hochwertige Ergebnisse hervorbringt. Dies wird durch die in den Experimenten ermittelten Kennzahlen belegt. In den Experimenten zur Bestimmung der Korrektheit der Zuordnungen sind die Ergebnisse insgesamt mit einer Sensitivität von ca. 90 % zufriedenstellend. Sie variieren jedoch je nach Objektdichte und der damit verbundenen Zahl an Verdeckungen sowie nach der Entfernung der Objekte zur Kamera und nach der Vielfältigkeit der äußeren Erscheinung der Objekte. Die Verfolgung funktioniert insgesamt besser, wenn sich die Objekte nahe der Kamera bewegen. Dies ist zum einen durch das verwendete Detektionsverfahren und zum anderen durch die schwächer ausgeprägten Unterscheidungsmerkmale, die sich aus den kleineren Bildregionen ergeben, bedingt. Die zusätzliche Verwendung der GNSS-Informationen bringt diverse Vorteile mit sich. Zum einen werden die Objekte auch nach längerer Verdeckung „wiedererkannt“, sodass deren Verfolgung korrekt fortgeführt werden kann. Die Trajektorien werden dabei auch in der Zeit, in der die Objekte nicht detektiert werden, detailliert erfasst. Zum anderen wird auch die Zuordnung in Szenarien unterstützt, in denen die Objekte anhand ihres Äußeren nur schwer unterschieden werden können.

Das Experiment zur Untersuchung der Genauigkeit der resultierenden Trajektorien liefert eine mittlere Abweichung, die bei beiden Modellierungsvarianten bei ca. 30 cm liegt (eine Rasterzellenbreite von 0,5 - 1 m vorausgesetzt). Befinden sich die Objekte im Nahbereich der Kamera, liegt

die Genauigkeit noch deutlich darunter (ca. 1 - 15 cm). Der Einsatz dieses Systems ist daher in den Anwendungsszenarien möglich, die entsprechende Anforderungen an die Genauigkeit stellen. Sie kann bspw. (wie in den Experimenten) zur Verfolgung von Fußballspielern genutzt werden, um deren Leistungen zu messen oder Bewegungsmuster zu erkennen. Der Einsatz als Torlinien-Technik kommt auf Grund der Distanzen zu den Toren und der ermittelten Genauigkeiten jedoch nicht in Frage, da dort bereits sehr wenige Zentimeter entscheidend sein können. Um dieses Szenario zu ermöglichen müssten auch hier zusätzliche Kameras eingesetzt werden, um einerseits die Distanz zu verringern und andererseits die Sichtbarkeit des Balls anhand von verschiedenen Perspektiven zu gewährleisten.

In den beschriebenen Experimenten wird ein System verwendet, das aus den in Abschnitt 6.1 vorgestellten Sensoren besteht. Die Anschaffungskosten für die Variante, die bspw. für das Fußballanalyse-Experiment genutzt wird (22 GPS-Logger, 2 Action-Kameras), belaufen sich dabei auf ca. 2880 Euro. Verglichen mit den kommerziellen Systemen (siehe Abschnitt 3.1.4) könnte dieses, auch bei der Hinzunahme weiterer Kameras, was sowohl die Verfolgung der Objekte als auch die Genauigkeit der Trajektorien begünstigen würde, weiterhin als Low-cost-Lösung angesehen werden. Dazu wird in der Laufzeitanalyse deutlich, dass unter Verwendung von entsprechend leistungsstarken Sensoren, wie es bspw. Smartphones sind, eine Erfassung der Trajektorien in Echtzeit möglich ist.

Das Ziel, eine kostengünstige Lösung zur Erfassung von qualitativ hochwertigen Trajektorien zu entwickeln, wird damit erreicht, wenn das Anwendungsszenario gewisse Anforderungen erfüllt. Dazu gehört neben einer festen Installation der Kameras auch, dass sich die relevanten Objekte mit GNSS-Empfängern ausstatten lassen. Prinzipiell würde der Ansatz zwar auch ohne GNSS-Unterstützung funktionieren, jedoch gehen dann auch die damit verbundenen Vorteile verloren.

7 Experimente und Evaluation der Mustererkennung

Nach der Evaluation der Datenerfassung wird in diesem Kapitel der in dieser Arbeit entwickelte Prozess zur Erkennung von Bewegungsmustern evaluiert und diskutiert. Dazu wird im folgenden Abschnitt 7.1 zunächst wieder die verwendete Software vorgestellt. In dem sich anschließenden Abschnitt werden die Datensätze, die die Grundlage für die Experimente darstellen, beschrieben. Zum Zweck der Evaluation wird ein Interessantheitsmaß für Bewegungsmuster eingeführt (Abschnitt 7.2), das im Zusammenhang mit verschiedenen Parameterstudien (7.4) und zur Diskussion der Ergebnisse der einzelnen Experimente (7.5.2 - 7.5.5) genutzt wird.

7.1 Verwendete Software

Zur Durchführung der Experimente und Auswertung der Ergebnisse werden zusätzlich zur bereits vorgestellten Software folgende eigens entwickelte und implementierte Tools genutzt.

DatasetInspector

Der *DatasetInspector* erlaubt einen ersten Überblick über die zuvor ausgelesenen oder auf andere Weise erfassten Trajektorie-Daten. Mit einer einfachen Daten-Visualisierung und unterschiedlichen Funktionen, wie z.B. einer zeitlichen Trimm-Funktion, einer Filterung/Glättung und einer Resampling-Funktion, können die Daten inspiziert und auch bereits vorverarbeitet werden.

PatternMiner

Der andere der beiden Themenschwerpunkte ist die Bewegungsmustererkennung. Dieses wird in der eigenständigen Software *PatternMiner* implementiert. Diese besitzt diverse Schnittstellen zum Einladen von Bewegungsdaten. So können diese aus Dateien oder aus Datenbanken gelesen werden und zugleich nach unterschiedlichen Kriterien (u.a. zeitliche oder räumliche Ausdehnung oder Objekt-Ids) gefiltert werden. Nach dem Laden stehen die in dieser Arbeit präsentierten Varianten der Mustererkennung zur Verfügung. Weiterhin kann aus den unterschiedlichen Möglichkeiten, bspw. der Segmentierung, innerhalb einer Variante gewählt werden. Über eine graphische Bedienoberfläche (siehe Abbildung 7.1a) lassen sich sämtliche benötigte Parameter setzen und Entscheidungen treffen. Die Ergebnisse können mittels einer umfassenden visuellen Ausgabe inspiziert werden. Gleichzeitig besteht die Möglichkeit, die Bewegungsmuster in einer Datenbank zu speichern, sodass auf sie später in weiteren Analysen zugegriffen werden kann.

FootballAnalysisTool und DBTrajectoryViewer

Sowohl das *FootballAnalysisTool* als auch das *DBTrajectoryViewer*-Tool sind speziell auf den Kontext Fußballanalyse ausgerichtet und dienen daher zur Untersuchung von Trajektorien aus eben diesem Bereich. Der Hauptunterschied zwischen diesen beiden Softwares liegt darin, dass das FootballAnalysisTool eine dynamische, das DBTrajectoryViewer-Tool eine statische Auswertung der Daten bietet. Bei einer dynamischen Auswertung wird das entsprechende Spiel „abgespielt“, animiert visualisiert und dazu entsprechend in Echtzeit die Ergebnisse der Analysen ermittelt. Da diese Software ebenfalls im Server-Betrieb laufen kann, ist es auch möglich, die Daten der verbundenen Sensoren tatsächlich „live“ zu verarbeiten. Bei der statischen Methode hingegen werden sämtliche Daten in den Hauptspeicher geladen. Nach Abschluss stehen unterschiedliche Analyse-Methoden

zur Verfügung, die sich auf Knopfdruck starten lassen. In beiden Fällen ermöglicht eine graphische Nutzerfläche (siehe Abbildung 7.1b) die Auswahl der zu analysierenden Daten und die Einstellung der relevanten Parameter. Zudem besteht die Möglichkeit die Ergebnisse persistent in einer Datenbank zu speichern. Diese beiden Werkzeuge werden insbesondere im Fußballanalyse-Experiment dieser Arbeit verwendet.

(a) *(b)*

Abbildung 7.1: Die graphischen Oberflächen des PatternMiner-Tools und des FootballAnalysisTools. Beide Werkzeuge werden in den Experimenten genutzt.

FootballAnalysisWebApp

Im Zusammenhang mit der Fußballanalyse steht mit der *FootballAnalysisWebApp* eine WebAnwendung zur Verfügung, die darauf abzielt, die Ergebnisse der Analysen online verfügbar zu machen. Die entsprechenden Daten werden in einer Datenbank vorgehalten und stammen aus den Analysen der beiden vorherigen Offline-Tools. Die Web-Anwendung bietet jedoch im Vergleich zu den Tools einen schmaleren Funktionsumfang. Der Schwerpunkt liegt dabei eher auf der Präsentation der Analyseergebnisse. So können u.a. Informationen über die analysierten Spiele und über die Performances der Spieler bzw. Teams abgerufen werden, sofern die Daten zur Verfügung stehen. Die Spiele können auch hier erneut animiert wiedergegeben werden.

7.2 Ergebnisverifikation

Die Verifikation von grundlegenden Bewegungsanalysemethoden, wie bspw. die Distanzmessung oder die Erzeugung von Heatmaps, gestaltet sich recht unkompliziert, da dort in den meisten Fällen auf Referenzergebnisse oder alternative Methoden zurückgegriffen werden kann. Die Verifikation von Bewegungsmusterergebnissen in Realdatensätzen ist jedoch deutlich herausfordernder, weil dazu in der Regel keine Referenzergebnisse existieren. Der Vergleich zu anderen Verfahren zur Mustererkennung ist zwar möglich, jedoch können dort auf Grund komplett unterschiedlicher Ansätze und Voraussetzungen nicht die gleichen Ergebnisse erwartet werden. Aus diesem Grund wird zur Überprüfung der Korrektheit und Vollständigkeit der Ergebnisse auf einen künstlich generierten Trajektorien-Datensatz zurückgegriffen. Dieser Datensatz wurde so designt, dass er die unterschiedlichen Herausforderungen bei der Mustererkennung abbildet. Er umfasst daher neben einigen willkürlich generierten Trajektorien auch wiederkehrende, exakt übereinstimmende Bewegungen. Zwei dieser Bewegungen liegen nahe beieinander und sind demnach nur leicht verschoben (A in Bild 7.2). Zwei Weitere sind deutlich verschoben (B), wobei eine lediglich einen Abschnitt einer längeren Trajektorie darstellt (C). Noch eine weitere Musterbewegung weist zusätzlich zu einer deutlichen Verschiebung auch eine beliebige Rotation auf (D). In Abbildung 7.2 wird dieser Datensatz gezeigt.

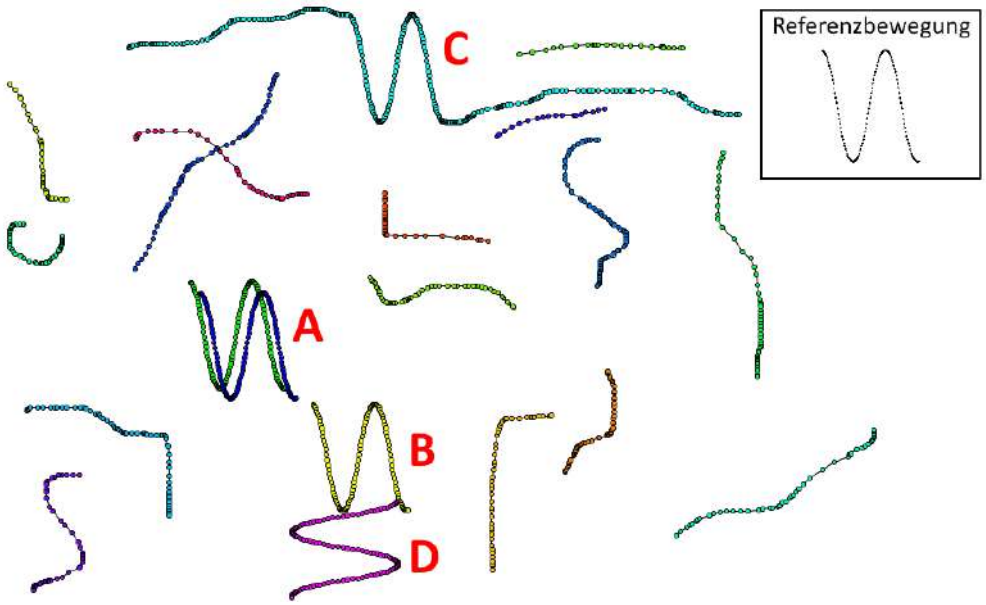
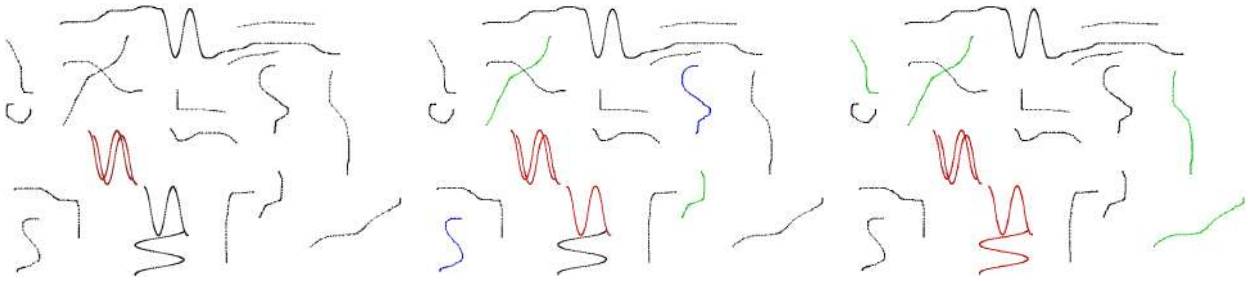


Abbildung 7.2: Der zur Verifikation verwendete Referenzdatensatz. Die Farben kennzeichnen die unterschiedlichen Trajektorien.

Clustering-basierter Ansatz

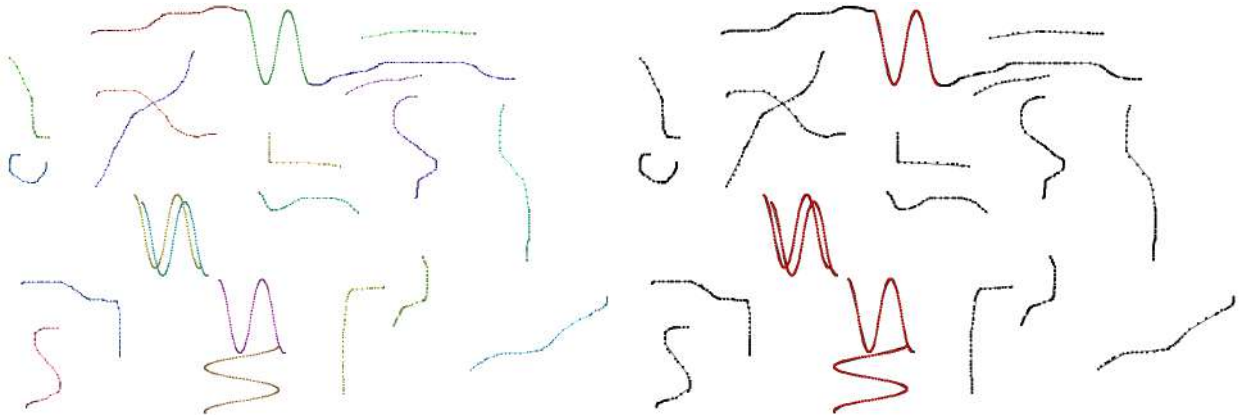
Die Ergebnisse hängen beim Clustering-basierten Ansatz stark von der Segmentierung ab. Ohne Kontextwissen gestaltet sich eine Segmentierung allerdings schwierig. Zur Analyse dieses künstlichen Datensatzes wird keine Segmentierung durchgeführt. Dies führt dazu, dass nur die Trajektorien als Instanzen eines Musters erkannt werden, die sich über die gesamte Länge ähneln. Abbildung 7.3 zeigt die Ergebnisse aus einem Clustering auf Basis der Fréchet-Distanz. Der Übersicht halber sind nicht alle erkannten Muster dargestellt. Sofern sie gefunden werden, was von den gewählten Invarianzen abhängt, sind die Referenzbewegungen jedoch immer dabei. Bei keiner Invarianz oder einer Translationsinvarianz werden zwei (a) bzw. drei (b) Instanzen (rot) erkannt. Unter Berücksichtigung einer Translations- und Rotationsinvarianz werden vier der fünf Referenzbewegungen erkannt (c). Lediglich die Bewegung (C), die durch ein Segment innerhalb einer längeren Trajektorie beschrieben wird, wird nicht erkannt.



(a) *(b)* *(c)*

Abbildung 7.3: Die Ergebnisse der Verifikation des Clustering-basierten Ansatzes ohne vorangegangene Segmentierung: a) Ohne Invarianzen, b) mit Translationsinvarianz und c) mit Translations- und Rotationsinvarianz. In rot sind jeweilig die erkannten Muster hervorgehoben.

Ist eine Segmentierung bekannt, die die Referenzbewegung C aus der Trajektorie herausschneidet und zu einem eigenen Segment macht, dann können bei Translations- und Rotationsinvarianz sämtliche Referenzbewegungen erkannt werden (siehe Abbildung 7.4).

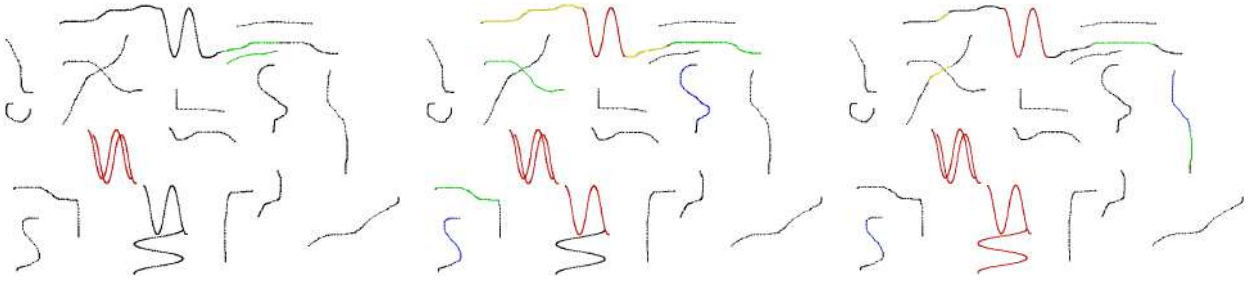


(a) (b)

Abbildung 7.4: Die Ergebnisse der Verifikation des Clustering-basierten Ansatzes mit bekannter Segmentierung: a) die Segmentierung (die Farben kennzeichnen die unterschiedlichen Segmente). b) Bei erlaubter Translation und Rotation werden sämtliche Referenzbewegungen (rot) erkannt.

Sequenzbasierter Ansatz

Der sequenzbasierte Ansatz liefert je nach Wahl der erlaubten Transformationen der Muster unterschiedliche Ergebnisse. Ist keine Transformation erlaubt, werden zwei unterschiedliche Muster (rot, grün) erkannt, die in Abbildung 7.5a gezeigt werden. Während sich das eine (grün) eher zufällig ergibt, besteht das Zweite aus den beiden Referenzbewegungen, die nahe beieinander liegen. Dass dieses Muster erkannt wird, obwohl eine kleine Verschiebung existiert, liegt an der Verwendung des Clustering-Verfahrens, durch das eine gewisse Fehlertoleranz gegeben ist. Die anderen Referenzbewegungen werden nicht als Instanz dieses Musters erkannt, da dort die Distanzen zu groß sind. Durch die Tolerierung einer Transformation steigt zum einen die Zahl der erkannten Muster und zum anderen werden nun auch die verschobenen Referenzbewegungen dem gleichen Muster (rot) hinzugefügt (siehe Abbildung 7.5b). Die verschobene und rotierte Bewegung wird auch in diesem Fall nicht dem Muster zugeordnet. Dies ändert sich, sobald Translation und Rotation erlaubt sind. Dann werden sämtliche Referenzbewegungen als Instanzen eines Musters angesehen (siehe Abbildung 7.5c). Zusätzlich steigt auch hier die Anzahl der erkannten Muster.



(a) (b) (c)

Abbildung 7.5: Die Ergebnisse der Verifikation des sequenzbasierten Ansatzes: a) keine Invarianzen, b) Translation ist erlaubt, c) Translation und Rotation sind erlaubt. In b) und c) werden nicht alle Ergebnisse gezeigt. Die unterschiedlichen Farben zeigen die erkannten Muster. Die einzelnen Instanzen eines Musters werden in der gleichen Farbe dargestellt.

7.3 Interessantheitsmaß für Bewegungsmuster

Bei der Untersuchung der Muster, die aus dem sequenzbasierten Verfahren hervorgegangen sind, kann die Korrektheit des Ergebnisses durch einen elementweisen Vergleich der Teilsequenzen überprüft werden. Hiervon ist allerdings auszugehen, da bereits getestete und korrekt arbeitende Sequenzanalyse-Algorithmen verwendet werden. Die Vollständigkeit der Ergebnisse kann jedoch bei realen Datensätzen nicht ohne Weiteres überprüft werden. Es werden nur die Muster identifiziert, die durch die jeweiligen Ansätze modelliert werden, d.h. die sich entweder zu Segment-Clustern zusammenfassen lassen oder als wiederkehrende Teilsequenzen in dem Datensatz codiert sind. Muster, auf die weder das eine noch das andere zutrifft, können dementsprechend nicht erkannt werden, obwohl ein Mensch das könnte. Auf Grund dieser Problematik wird hier ein Interessantheitsmaß eingeführt, mit dem die Ergebnisse vergleichbar gemacht werden können und das den Informationsgehalt der Muster beschreibt. Der Informationsgehalt hängt stark vom Anwendungsszenario ab. In den üblichen Szenarien, in denen Bewegungsmuster Verwendung finden, nämlich der Bewegungsprädiktion und der Charakterisierung, wird der Gehalt durch die Ausdehnung der Muster und die Ähnlichkeit der Segmente bzw. der Teilsequenzen bestimmt. Die Ausdehnung eines Musters wird bestimmt durch dessen Länge $l(P)$ und der Anzahl *suppc* der Trajektoriesegmente S_i. Bei der Analyse von Einzelobjektbewegungen ist $l(P)$ die mittlere räumliche Distanz $s(S_i)$, die das Objekt in den Segmenten zurückgelegt hat.

$$l(P) = \underset{1 \leq i \leq suppc}{\text{mean}} s(S_i), \tag{7.1}$$

Im Fall der Untersuchung von Gruppenbewegungsmustern ist $s(S_i)$ die mittlere aller Distanzen D_j der beteiligten Objekte pro Segment.

$$s(S_i) = \underset{1 \leq j \leq n}{\text{mean}} D_j. \tag{7.2}$$

Die Ähnlichkeit $sim(P)$ berechnet sich durch

$$sim(P) = \frac{1}{1 + \underset{1 \leq i,j \leq suppc}{\max} d(S_i,S_j)}. \tag{7.3}$$

Die Ermittlung der Distanz zweier Segmente $d(S_i,S_j)$ erfolgt mit Hilfe der Fréchet-Metrik. Im Fall der Einzelbewegungen wird hierzu jede Kombination aus Segmenten berücksichtigt. Da jedoch bei der Gruppenbewegungsanalyse eine Trajektorie für jedes Objekt vorliegt, müssen zuerst korrespondierende Trajektorie-Paare zwischen den Segmenten (die jeweils aus einer Menge an Einzel-Trajektorien bestehen) bestimmt werden. Daraufhin können deren Distanzen berechnet werden. Das Interessantheitsmaß I ergibt sich schließlich durch

$$I(P) = suppc(P)\ l(P)\ sim(P). \tag{7.4}$$

Diese Bewertung der Muster eignet sich auch im Kontext der *Visuellen Analyse* (*Visual Analytics*), da es den Nutzer auf die interessanten Muster hinweist. Die endgültige Interpretation der Muster muss jedoch vom Nutzer erfolgen. Abbildung 7.6 zeigt einige Musterbeispiele aufsteigend geordnet nach Interessantheitsmaß. In den folgenden Abschnitten wird dieses Maß zur Evaluierung der Ergebnisse benutzt. Zudem werden die Muster, die im Rahmen der Analyse der Fußballdatensätze erkannt werden, unter Verwendung von ausreichend vorhandener Fußballkompetenz visuell inspiziert und analysiert.

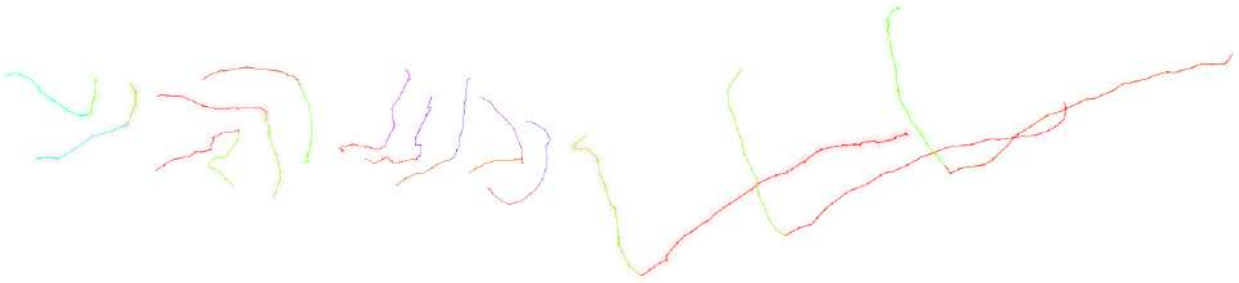


Abbildung 7.6: Illustration des Interessantheitsmaßes für Bewegungsmuster anhand von Einzelbewegungsmustern: es steigt von links nach rechts (13,37; 23,81; 53,66; 77,57). Die unterschiedlichen Farben repräsentieren die unterschiedlichen Cluster-Zuordnungen jeder einzelnen Bewegung.

7.4 Parameterstudien

Die Ergebnisse der beiden vorgestellten Mustererkennungsansätze hängen von verschiedenen Faktoren ab. Diese sind die benötigen Eingabeparameter, die möglichen Invarianzen der Muster und die Dichte der Bewegungsdaten. Für beide Ansätze werden Parameterstudien durchgeführt, um die entsprechenden Einflüsse zu ermitteln. Die Ergebnisse werden anhand des zuvor eingeführten Interessantheitsmaßes sowie dessen einzelner Faktoren bewertet. Die Interessantheitsmaße der Muster werden für jedes Setting aufsummiert, um einen Wert für die Gesamtinteressantheit zu erhalten. Diese gibt Aufschluss darüber, wie gut die jeweiligen Parameter gewählt sind.

7.4.1 Eingabeparameter

Clustering-basierter Ansatz

Der Clustering-basierte Ansatz erfordert Parameter für die Segmentierung und für das Clustering der Segmente. Die Segmentierung besitzt zwar ebenfalls einen Einfluss auf die resultierende Mustermenge, wird aber in der Parameterstudie als gegeben angenommen. Abhängig vom gewählten Verfahren handelt es sich bei den Clustering-Parametern um k für das k-means-Verfahren bzw. ϵ und $minIts$ für den DBSCAN-Algorithmus. Stellvertretend werden in dieser Studie die DBSCAN-Parameter untersucht. Der Parameter $minIts$ entspricht hierbei dem Mindest-Support Count und wird auf $minIts = suppc_{min} = 2$ festgelegt. Er bleibt im Verlauf unverändert. Als Datengrundlage werden drei Trajektorien aus dem ACM DEBS 2013-Datensatz (siehe Abschnitt 7.5.1) verwendet, die nach der Segmentierung in 1277 Segmente unterteilt sind. Die Ergebnisse werden im Plot in Abbildung 7.7a gezeigt.

Die Anzahl der erkannten Muster beschreibt eine Kurve, die nach einem steilen Anstieg im Bereich um $\epsilon = 3{,}5\ m$ ihren Höhepunkt besitzt und danach langsamer wieder abklingt bis sie ab ca. $\epsilon = 10{,}5\ m$ sich nicht mehr verändert. Dies entspricht auch dem typischen Verlauf der Clusteranzahl bei Verwendung des DBSCAN-Algorithmus. Wird eine kleine Distanz für ϵ angesetzt, so ist die erforderliche Dichte der Objekte hoch. Kleine Dichteunterschiede in den Daten führen zu unterschiedlichen Clustern in den Daten. Wird die Distanz erhöht, so steigt die Zahl der Cluster bis zu dem Punkt, an dem nahe beieinander liegende Cluster zusammengefasst werden. Dieses führt wiederum zu einer geringeren Clusteranzahl, bis sämtliche Objekte in einem großen Cluster zusammenfasst sind. Da in diesem Fall die Muster aus den Clustern generiert werden, kann das beschriebene Verhalten direkt übertragen werden. Dieses wird ebenfalls an der Kurve für den Support Count deutlich. Die Anzahl der Instanzen pro Muster steigt mit wachsendem ϵ. Dieses entspricht der Anzahl der Objekte pro Cluster und damit dem zuvor beschriebenen Verhalten. Mit

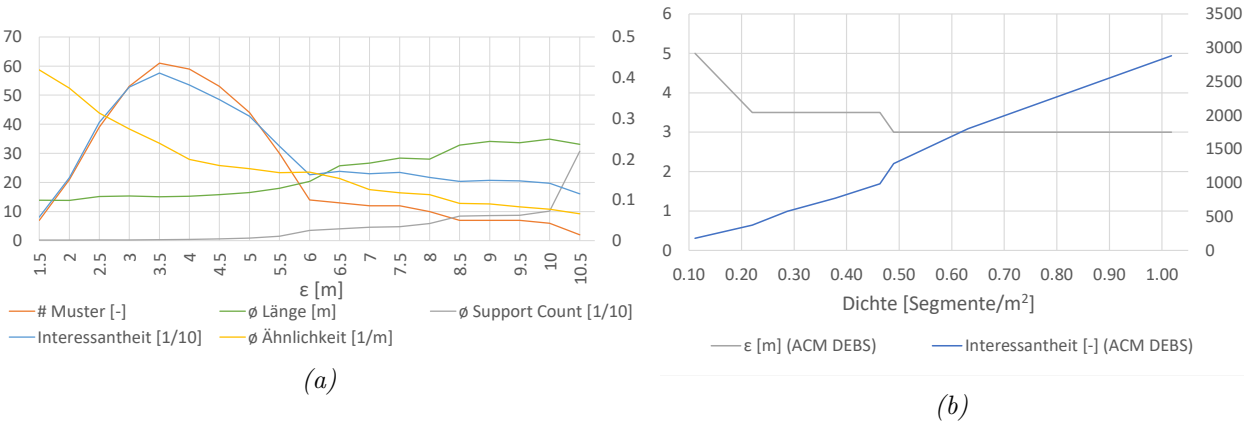


Abbildung 7.7: Die Ergebnisse der Parameterstudie des Parameters ϵ für den DBSCAN-Algorithmus ***ohne Invarianzen****: (a) die Anzahl der Muster (orange), die durchschnittliche Länge der Segmente (grün) und die Gesamtinteressantheit aller erkannten Muster (blau), die durchschnittliche Ähnlichkeit der Instanzen innerhalb der Muster (gelb) und der durchschnittliche Support Count (grau). (b) Auswertung des Werts für ϵ (grau), der abhängig von der Dichte der Daten die höchste Interessantheit (blau) liefert.*

dem steigendem ϵ sinkt die Ähnlichkeit der Segmente, die demselben Cluster zugeordnet werden. Gleichzeitig steigt aber deren Länge. Mit dem Parameter ϵ lässt sich so die Interessantheit der Muster steuern, da er die zu Grunde liegenden Faktoren beeinflusst. Ein Maß zur Identifikation eines geeigneten Werts für ϵ ist die Gesamtinteressantheit, als die Summe der Interessantheit aller erkannten Muster. Diese besitzt einen ähnlichen Verlauf wie die Kurve der Musteranzahlen und erreicht ebenfalls im Bereich um $\epsilon = 3{,}5\ m$ ihr Maximum. Bei Verwendung dieses Parameterwerts werden nach dieser Definition der Interessantheit die besten Ergebnisse gefunden.

Sequenzbasierter Ansatz

Beim sequenzbasierten Ansatz werden ebenfalls entsprechende Clustering-Parameter benötigt. Mit diesen kann die Länge des Alphabets festgelegt werden, das zur Ähnlichkeitsbestimmung der Sequenzelemente genutzt wird. Stellvertretend wird dieses Mal der Einfluss des k-means-Parameters k evaluiert, der die Anzahl an Clustern festlegt und damit gleichzeitig der Länge des Alphabets entspricht. Da die übrigen benötigten Größen für den Mustererkennungsalgorithmus, $suppc_{min}$ und l_{min}, lediglich die Menge an potentiellen Mustern eingrenzen, werden für diese die festen Werte $suppc_{min} = 2$ (als minimal möglicher Wert) und $l_{min} = 10\ m$ definiert. Durchführung und Datengrundlage entsprechen der Parameterstudie des Clustering-basierten Ansatzes. Der Plot in Abbildung 7.8a zeigt das Verhalten bei sich änderndem k.

Auch hier beeinflusst die Anzahl der Cluster die Zahl der gefundenen Muster. Mit steigender Zahl an Clustern und damit längerem Alphabet steigt ebenso die mittlere Ähnlichkeit der Instanzen innerhalb eines Musters. Dies begründet sich darin, dass die Bewegungen mit einem umfangreicherem Alphabet detaillierter beschrieben werden können. Die Anzahl der Instanzen pro Muster sinkt, da weniger ähnliche Bewegungen gefunden werden können. Die Zahl der Muster steigt zunächst, fällt dann jedoch wieder ab. Dies liegt daran, dass anfangs viele ähnliche Bewegungen identifiziert werden, jedoch alle nur einem oder wenigen Clustern zugeordnet werden. Dies ändert sich mit steigendem k, sodass mehr Cluster ermittelt werden. Ab einem gewissen Punkt werden nicht mehr ausreichend ähnliche Bewegungen für jedes Cluster gefunden, sodass deren Zahl sinkt. Die Interessantheit verhält sich analog. Die geeignetste Wahl für k ist nach dieser Studie $k = 64$.

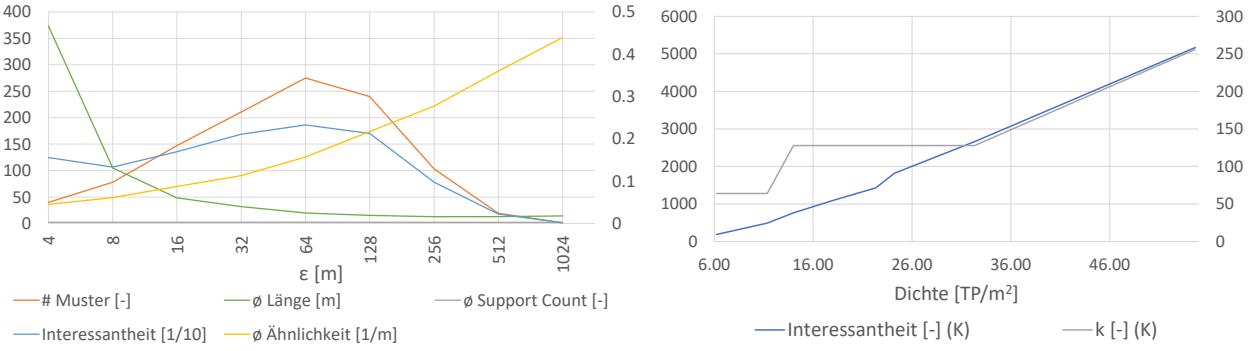


Abbildung 7.8: Die Ergebnisse der Parameterstudie des Parameters k für den k-means-Algorithmus ***ohne Invarianzen****: (a) die Anzahl der Muster (orange), die durchschnittliche Länge der Segmente (grün) und die Gesamtinteressantheit aller erkannten Muster (blau), die durchschnittliche Ähnlichkeit der Instanzen innerhalb der Muster (gelb) und der durchschnittliche Support Count (grau). (b) Auswertung des Werts für k (grau), der abhängig von der Dichte der Daten die höchste Interessantheit (blau) liefert.*

7.4.2 Datendichte

In beiden Ansätzen korreliert die Gesamtinteressantheit augenscheinlich stark mit der Anzahl der erkannten Muster. Da diese wiederum von der Dichte der Daten abhängt, wird in diesem Zusammenhang der Verlauf des Clustering-Parameters, die bei den jeweiligen Datendichten die besten Ergebnisse liefern, untersucht. Die Dichte der Daten berechnet sich durch die Anzahl der Trajektoriepunkte pro deren aufgespannter Fläche. Die Fläche wird je nach Bewegungsraum unterschiedlich bestimmt. Sowohl im euklidischen Raum als auch im Netzwerkraum entspricht sie der konvexen Hülle. Jedoch werden beim Netzwerkraum die Flächen abgezogen, die keine Daten enthalten können, d.h. wo prinzipiell keine Bewegung möglich ist, z.B. auf bebauten Flächen.

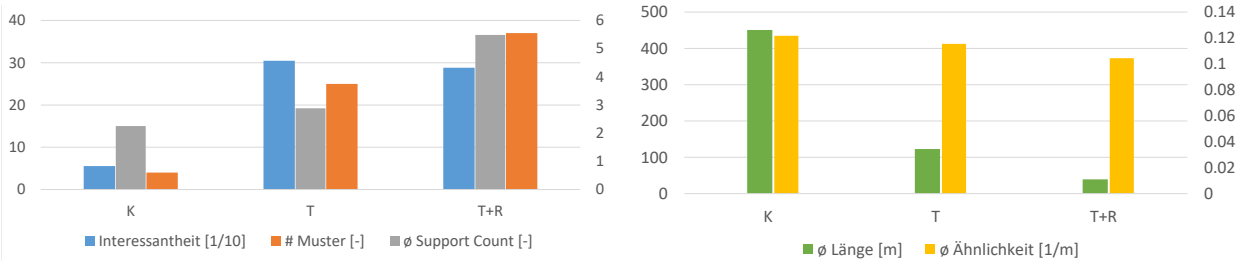
In Abbildung 7.7b werden die Ergebnisse dieser Untersuchung für den ϵ-Parameter des Clustering-basierten Ansatzes, in Abbildung 7.8b die Ergebnisse für k aus sequenzbasierten Ansatz gezeigt. Die Ergebnisse zeigen in beiden Fällen, dass mit zunehmender Dichte die Interessantheit der Muster steigt. Der Optimalwert für ϵ nimmt ab, der für k nimmt hingegen zu. Diese Studie hilft bei bekannter Datendichte, die optimalen Clustering-Parameter zu bestimmen.

7.4.3 Invarianzen

Zur Evaluierung der Einflüsse der unterschiedlichen Musterinvarianzen werden die jeweiligen Ergebnismengen in Bezug auf die Interessantheit und die übrigen Faktoren untersucht. Unter der Annahme, dass die Einflüsse bei beiden Ansätzen ähnlich sind, wird zu deren Untersuchung stellvertretend der sequenzbasierte Ansatz auf die Referenzdaten angewendet. In verschiedenen Durchläufen werden dabei die Invarianzen variiert. Während die Dichte der Daten in jedem dieser Durchläufe mit 0,001 $\frac{TP}{m^2}$ unverändert bleibt, werden die erforderlichen Parameter jeweils gemäß der vorangegangen Studie gewählt.

Die Diagramme in Abbildung 7.9 zeigen die Werte für die Gesamtinteressantheit, die Anzahl der Muster, deren mittlere Länge und Ähnlichkeit, sowie den mittleren Support Count. Die Ergebnisse zeigen, dass mit zunehmender Zahl an Invarianzen die Gesamtinteressantheit, die Anzahl der Muster und der Support Count zunehmen. Der Grund für die steigende Zahl an Mustern liegt darin, dass die Muster toleranter in Bezug auf die Ähnlichkeit von Teilsequenzen sind, d.h. die Wahrscheinlichkeit ist höher, zwei ähnliche Teilsequenzen zu finden. Dass auch der Support Count

zunimmt, wird bereits bei der Verifikation in Abschnitt 7.2 deutlich. Dort werden dem Referenzmuster mit jeder zusätzlichen Invarianz weitere Instanzen hinzugefügt. Im Gegensatz dazu nehmen die Ähnlichkeit wenig und die mittlere Länge der Muster deutlich ab. Der Grund hierfür ist, dass dort neben den recht langen Mustern auch viele sehr kurze Muster gefunden werden. Dies führt zu einer geringen mittleren Länge.



(a) *(b)*

Abbildung 7.9: Die Einflüsse auf die Eigenschaften der erkannten Bewegungsmuster: a) Gesamtinteressantheit aller erkannten Muster (blau), die Anzahl der Muster (orange) und der durchschnittliche Support Count (grau). b) Die durchschnittliche Länge der Segmente (grün) und deren mittlere Ähnlichkeit (gelb).

7.5 Experimente auf realen Datensätzen

Nach der Verifikation und den Parameterstudien wird die Mustererkennung ebenfalls auf realen Datensätzen durchgeführt, um zu zeigen, dass die vorgestellten Ansätze auch dort funktionieren. Die vorgestellten Ansätze werden zu diesem Zweck auf vier unterschiedlich umfangreiche Datensätze mit verschiedenen Charakteristiken angewendet. Die ersten beiden Datensätze enthalten Bewegungsinformationen von Fußballspielern. Der Dritte beinhaltet Trajektorien aus dem Bereich Verkehr, der Vierte von Tieren. Insgesamt handelt es sich um Daten aus jeweils einer komplett anderen Domäne, was gleichzeitig die Portabilität und die Skalierbarkeit des Verfahrens zeigen soll.

Um eine gewisse Redundanz in Bezug auf die Analyse der Daten und auf die Präsentation sowie Diskussion der Ergebnisse zu vermeiden, werden nicht für jeden Datensatz beide Ansätze in sämtlichen Ausführungen genutzt. Zudem wird auf Analysen verzichtet, bei denen davon auszugehen ist, dass keine gezielten Bewegungsmuster erkannt werden.

7.5.1 Beschreibung der Datensätze und Experimente

ACM DEBS 2013 Data Challenge

Der erste Datensatz wurde vom FraunhoferIIS im Rahmen der *ACM DEBS 2013 Data Challenge* (Mutschler u. a., 2013) generiert und umfasst die Daten eines Kleinfeld-Fußballspiels mit 8 Spielern pro Team und einem Schiedsrichter (siehe Abbildung 7.10a). Die Erfassung der Trajektorien erfolgte mit einem eigens entwickelten funkbasierten Echtzeit-Lokalisierungssystems (*Real-Time Locating System*) RedFir ®. Über ein 60 minütiges Spiel wurden die Spieler, der Schiedsrichter und die Bälle mit Sensor-Tags ausgestattet, mit deren Hilfe die Position der Beteiligten ermittelt wurde. Die Spieler und der Schiedsrichter wurden dabei mit zwei Sensor-Tags nahe der Füße versehen, sodass für beide Füße eine Trajektorie ermittelt wurde. Da für dieses Experiment jedoch nur eine Trajektorie pro Objekt benötigt wird, wurde die beiden Fuß-Trajektorien zuvor zu einer repräsentativen Spieler-Trajektorie zusammengefügt. Die Sampling-Rate der Trajektorien beträgt ca. 200 Hz, die der Ball-Trajektorien sogar 2000 Hz. Die Genauigkeit der Positionierung wird

vom FraunhoferIIS mit wenigen Zentimetern angegeben, sodass es sich hierbei um einen insgesamt räumlich hoch-genauen und zeitlich sehr hoch-aufgelösten Datensatz handelt.

In diesem Experiment werden beide Verfahren angewendet. Die Segmentierung, die für den Clustering-basierten Ansatz benötigt wird, erfolgt auf Grundlage der vorliegenden Spielsituationen. Da die Spieler sich als Gruppe über das Spielfeld bewegen, wird auch eine Gruppenmuster-Analyse durchgeführt.

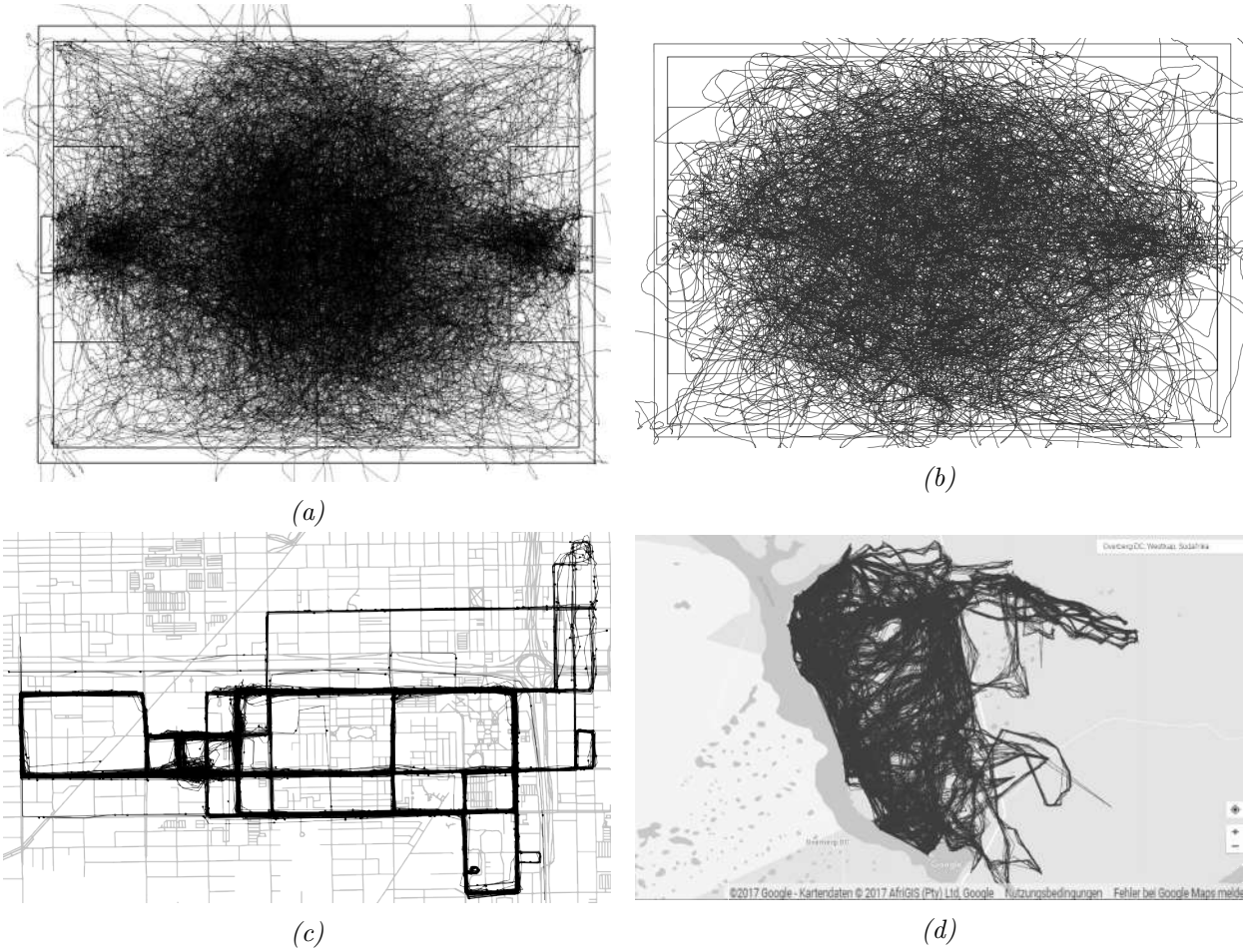


Abbildung 7.10: a) Die hoch-genauen Trajektorien des Datensatzes für das erste Experiment. b) Die GPS-Fußball-Trajektorien (hier wird nur ein Spiel gezeigt), die im zweiten Experiment genutzt werden. c) Die im dritten Experiment prozessierten Bus-Trajektorien. d) Die Trajektorien der Mantelpaviane des letzten Experiments[1].

GPS Fußball

Die zweite Datenquelle ist ein selbst aufgezeichneter umfangreicher Datensatz aus Fußballspieler-GPS-Trajektorien (Abbildung 7.10b). Er enthält die Bewegungsinformationen eines Teams (zwischen 11 und 14 Spieler) in mehr als 20 kompletten Begegnungen. Die GPS-Geräte (vgl. Abbildung 6.2a), die zur Aufzeichnung der Trajektorien genutzt wurden, liefern Daten mit einer Sampling-Rate von 5 Hz mit einem durchschnittlichen Fehler von ca. 3 - 10 m. Daher müssen pro Spieler-Trajektorie ungefähr 27 000 Punkte (der Datensatz umfasst insgesamt ca. 7 Millionen Punkte) prozessiert werden. Obwohl dieser Datensatz im Vergleich zum Vorherigen sowohl räumlich ungenauer als auch zeitlich deutlich schlechter aufgelöst ist, beinhaltet er jedoch Daten über einen

[1] Hintergrundkarte: Google maps. Zugriff am 27.11.2017

wesentlichen größeren Zeitraum (1 x 60 min gegenüber 20 x 90 min). Unter der Annahme einer maximalen Geschwindigkeit von ca. 40 km/h (11,11 m/s) können sich die Spieler maximal 2,22 m in einem Zeitschritt fortbewegt haben. Bei solch hohen Geschwindigkeiten sind die Bewegungen jedoch recht gradlinig. Werden die Bewegungen komplexer, d.h. es gibt mehr Richtungswechsel, so sind auch die Geschwindigkeiten deutlich niedriger. Insgesamt sind die Trajektorien der Spieler mit einer Sampling-Rate von 5 Hz für Analysen wie die Mustererkennung ausreichend detailliert erfasst.

Dieses Experiment besteht lediglich aus der Anwendung des sequenzbasierten Verfahrens, da in diesem Fall nach allgemeinen Bewegungsmustern zur Charakterisierung unterschiedlicher Spieler gesucht wird. Auf eine Gruppenmuster-Analyse wird verzichtet, da sie sich nicht wesentlich vom vorangegangenen Experiment unterscheidet.

MapConstruction.org

Der dritte Datensatz beinhaltet Daten auf dem Kontext „Verkehr“ und stammt von Ahmed u. a. (2015). Im „Chicago“-Datensatz wurden die Bewegungen von Shuttle-Bussen der *University of Illinois at Chicago* (*UIC*) mittels GPS und einer durchschnittlichen Sampling-Rate von ca. 4 Hz erfasst. Er enthält 889 Trajektorien mit insgesamt 118 000 Punkten. In diesem Szenario bewegen sich die Objekte in einem Netzwerkraum, was zu weniger Freiheitsgraden bzgl. der Bewegungsrichtung, weniger Richtungswechseln und konzentrierteren Trajektorieaufkommen führt. Die Daten sind damit dichter, obwohl der Raum, in dem sich die Objekte bewegen, deutlich größer ist als in den übrigen Datensätzen. In Abbildung 7.10c ist der komplette Datensatz zu sehen.

In diesem Experiment werden beide Verfahren angewendet. Die für den Clustering-basierten Ansatz erforderliche Segmentierung wird auf Basis von interessanten Plätzen durchgeführt. Auf eine Gruppenmuster-Analyse wird verzichtet, da Busse im Allgemeinen nicht als Gruppe unterwegs sind und mögliche Ergebnisse eher Zufallsprodukte wären.

Mantelpaviane

Der vierte und letzte Datensatz resultiert aus der Beobachtung von Tieren und wird von Bonnell u. a. (2017) zur Verfügung gestellt (Abbildung 7.10d). In diesem Fall wurden die Bewegungen von 14 Mantelpavianen (*Papio hamadryas ursinus*) über einen Zeitraum von 74 Tagen im *DeHoop Nature Reserve* in Südafrika aufgezeichnet. Die Aufzeichnung der Daten erfolgte mit Hilfe eines manuellen GPS-Trackings der einzelnen Tiere. Bei einer durchschnittlichen Zeitspanne von ungefähr 15 Minuten zwischen zwei Positionsmessungen eines Tieres umfasst der Datensatz insgesamt ca. 62k Trajektoriepunkte. Im Vergleich zu den vorherigen Datensätzen wurden die Bewegungen durch die verhältnismäßig geringe Sampling-Rate nicht sehr detailliert erfasst. Zudem existieren Messunterbrechungen von einigen Stunden, meisten über Nacht. Die Tiere haben sich als Gruppe über die gesamte Zeit fortbewegt.

In dem dazugehörigen Experiment wird lediglich der Clustering-basierte Ansatz angewendet. Zur Segmentierung werden interessante Plätze detektiert. Das unregelmäßige Sampling und die langen Unterbrechungen in diesem Datensatz sorgen dafür, dass die Ergebnisse des sequenzbasierten Verfahrens recht willkürlich erscheinen. Aus diesem Grund wird auf dessen Beschreibung und auf die Gruppenmusteranalyse verzichtet.

Übersicht

Folgende Tabelle fasst die Datensätze und deren Charakteristika noch einmal zusammen. Zusätzlich beschreibt sie knapp, welcher der Ansätze angewendet wird.

Tabelle 7.1: Überblick über die in den Experimenten genutzten unterschiedlichen Datensätze.

Datensatz	Charakteristiken	Experiment
ACM DEBS 2013 Data Challenge (Mutschler u. a., 2013) (Kontext: Fußball)	2 Teams, 16 Spieler, 1 Schiedsrichter 11.5M Punkte (1 Spiel) Räuml. Genauigkeit: wenige cm Zeitl. Auflösung: 200 Hz Euklidischer Bewegungsraum	1 Clustering-bas. Ansatz (Segm.: semantisch) Sequenzbas. Ansatz Gruppenmuster
GPS Fußball (Kontext: Fußball)	Mehr als 20 Spiele, 11 - 14 Spieler/Spiel ca. 7M Punkte Räuml. Genauigkeit: 5 - 10 m (GPS) Zeitl. Auflösung: 5 Hz Euklidischer Bewegungsraum	2 Sequenzbas. Ansatz
MapConstruction.org Chicago (Ahmed u. a., 2015) (Kontext: Verkehr)	118K Punkte Räuml. Genauigkeit: 5-20 m (GPS) Zeitl. Auflösung: ca. 4 Hz Netzwerkraum	3 Clustering-bas. Ansatz (Segm.: Interes. Plätze) Sequenzbas. Ansatz
Mantelpaviane (Bonnell u. a., 2017) (Kontext: Tiere)	14 Tiere 62k Punkte Räuml. Genauigkeit: GPS-Genauigkeit Zeitl. Auflösung: ∅ 1 Messung/15 Min. Euklidischer Bewegungsraum	4 Clustering-bas. Ansatz (Segm.: Interes. Plätze)

7.5.2 Experiment 1 - ACM DEBS 2013-Datensatz

Da dieser Datensatz die Datengrundlage für die Parameterstudien gewesen ist und dort die unterschiedlichen Kennzahlen wie bspw. die Interessantheit der Muster bereits ausgewertet worden sind, wird in diesem Experiment auf eine erneute Evaluation der Ergebnisse in dieser Hinsicht verzichtet. Stattdessen werden hier die erkannten Bewegungsmuster des Clustering-basierten Ansatzes als Grundlage für eine detailliertere Analyse der Spielerbewegungen genutzt. Die Ergebnisse des hier angewendeten sequenzbasierten Verfahrens werden abhängig von den Musterinvarianzen untersucht. Zudem wird mit Hilfe des gleichen Verfahrens nach Gruppenmustern gesucht.

Clustering-basierter Ansatz

Die Trajektorien der beobachteten Spieler erstrecken sich über die Gesamtdauer des Spiels und sind somit zu lang, um mit dem Clustering-basierten Ansatz direkt verarbeitet werden zu können. Aus diesem Grund ist eine vorangehende Segmentierung erforderlich, mit der die Trajektorien in bedeutungsvolle Segmente unterteilt werden. Zu Identifikation der Segmentenden wird hier das in Abschnitt 5.5.1 beschriebene Klassifikationsmodell angewendet. Dieses ermöglicht eine Bestimmung von unterschiedlichen Spielsituationen. In diesem Fall beschränken sich diese auf Offensiv- und Defensiv-Situationen der beiden Teams. Eine derartige Segmentierung ermöglicht eine gesonderte Betrachtung der Verhaltensweisen der Teams bzw. der Spieler in eben diesen Situationen. Abbildung 7.11 zeigt die resultierenden Trajektoriesegmente eines Angreifers in den Offensivsituationen des eigenen Teams. Die unterschiedlichen Spielrichtungen in beiden Halbzeiten werden auf die Weise berücksichtigt, dass die Bewegungen der zweiten Halbzeit gespiegelt werden. So wird in beiden Halbzeiten in die gleiche Richtung gespielt. Die Dichte der Daten steigt entsprechend, sodass sich laut der Parameterstudie die Chance erhöht, Muster zu erkennen.

Die identifizierten Muster zeigen ein unterschiedliches Verhalten der beiden Teams auf. Team 1 (7.12a) nutzt die gesamte Breite des Platzes aus und bewegt sich dabei recht geradlinig in Rich-

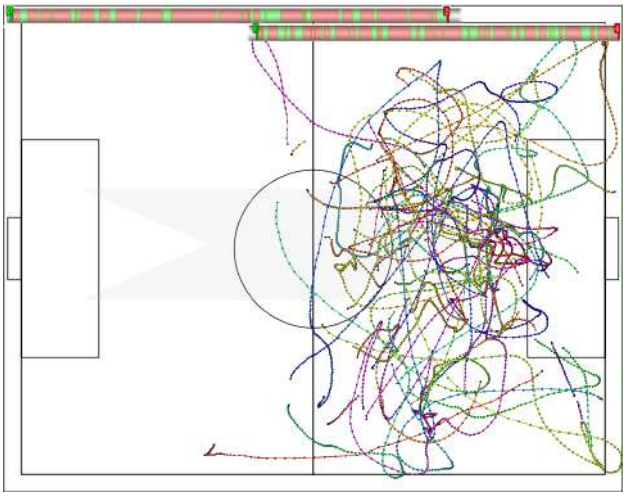
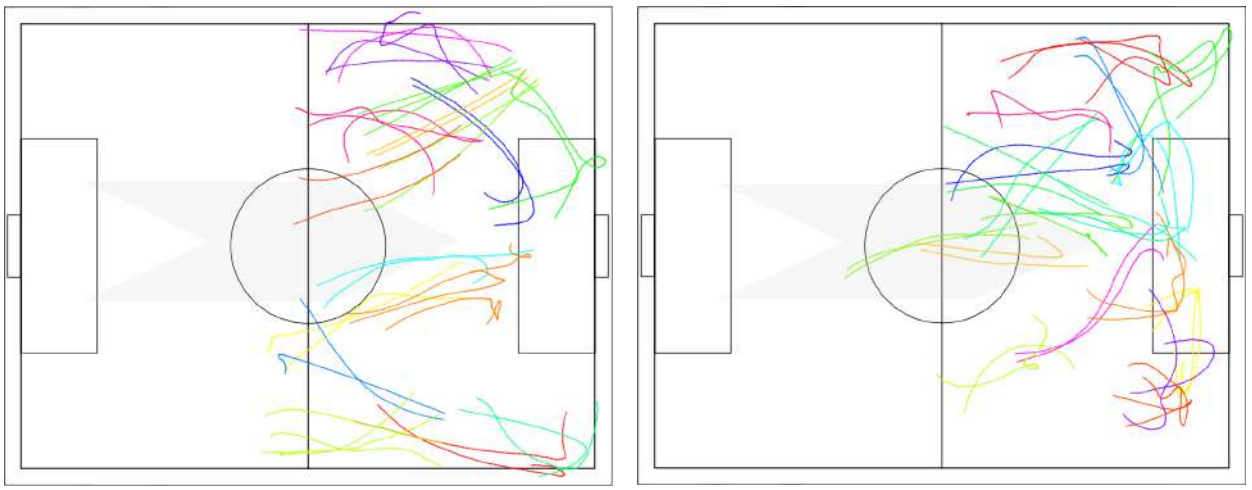


Abbildung 7.11: Die segmentierten Offensivbewegungen eines Angreifers. Oben wird zudem der segmentierte Zeitstrahl dargestellt. Die grünen Intervalle stellen die Offensivsituationen dar.

tung gegnerisches Tor. Zudem ist zu erkennen, dass die Bewegungen die von der halblinken bzw. halbrechten Seite starten, ebenfalls auf der entsprechenden Außenseite des Felds enden. Nur die Bewegungen, die zentral beginnen, führen direkt Richtung Tor. Das Angriffsspiel von Team 2 (7.12b) hingegen ist nicht so breit ausgelegt. Viele der Bewegungen zeigen weiterhin eine etwas andere Form. Sie bestehen aus einer Bewegung entlang der Seitenauslinie und einem Richtungswechsel in Richtung gegnerisches Tor. Sie sind weniger geradlinig.



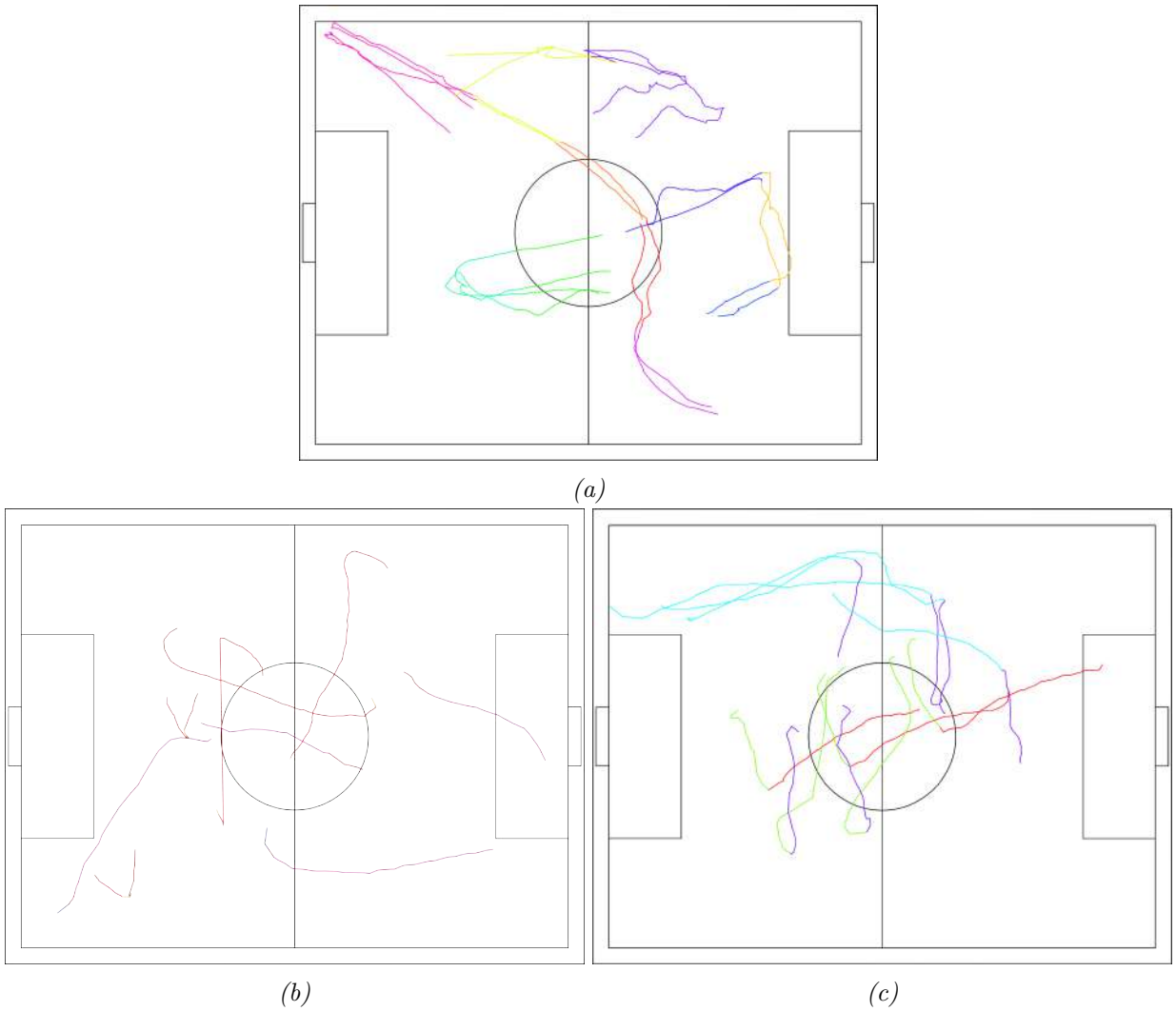
(a) *(b)*

Abbildung 7.12: Die verschieden eingefärbten Muster zeigen die unterschiedlichen Verhaltensweisen der beiden Teams in Offensivsituationen: a) Team 1 nutzt für die Angriffe die gesamte Breite des Platzes. Die Angriffsbewegungen sind in diesem Fall recht geradlinig. b) Team 2 konzentriert sich hingegen mehr auf die Bewegungen über die linke Seite und die Mitte. Viele der Muster zeigen eine Bewegung entlang der Seitenauslinie und einen Richtungswechsel in Richtung gegnerisches Tor.

Sequenzbasierter Ansatz

Bei der vorangegangenen Analyse sind keine Invarianzen berücksichtigt worden. Daher werden bei der Anwendung des sequenzbasierten Ansatzes die unterschiedlichen Ergebnisse abhängig von den Invarianzen der Muster untersucht. Die Parameter werden dazu jeweils gemäß der Studien in Abschnitt 7.4 gewählt.

Wird keine Invarianz berücksichtigt, resultiert dies in 983 erkannten Bewegungsmustern. Einige von diesen sind in Abbildung 7.13a beispielhaft dargestellt. Die ermittelten Ergebnisse beruhen auf sich räumlich überlappenden Trajektorien. Verfolgen sich zwei Spieler während des Spiels, kann dies zu relativ langen ähnlichen Trajektorien und je nach gewähltem Mindest-Support Count auch zu ebenso langen Mustern führen. Abbildung 7.13b zeigt einige Beispiele der insgesamt 5609 Muster, die gefunden werden, wenn sowohl Translation als auch Rotation erlaubt sind. Die Muster sind vergleichsweise kürzer, bestehen dafür aber aus mehr Instanzen. Die Geometrien sind insgesamt recht einfach. Ist nur Translation erlaubt, befinden sich die Ergebnisse im Allgemeinen in der Mitte zwischen den beiden anderen Varianten. Einige der 1895 Muster sind recht lang und besitzen komplexe Geometrien, wie bspw. die doppelt auftretende Schleife (grüne und lila Segmente) in Abbildung 7.13c.



(a)

(b) *(c)*

Abbildung 7.13: Einige Bewegungsmuster aus dem Ergebnis des ersten Experiments: a) keine Invarianzen: Muster aus räumlich überlappenden Trajektorien werden gefunden. b) Translation- u. Rotation-Invarianz: die Trajektorien, die zum gleichen Muster gehören, können verschoben und/oder gedreht sein. c) Translation-Invarianz: In diesem Fall sind nur Verschiebungen der Segmente erlaubt. Die Farben symbolisieren die Zugehörigkeit der Einzelbewegungen bzw. der Sequenzelemente zu den Clustern.

Bei der Interpretation der Muster müssen deren Invarianzen ebenfalls in Betracht gezogen werden. So spielt bspw. die Bewegungsrichtung in dem Kontext der Fußballanalyse eine Rolle. Ob die Bewegung in Richtung Tor oder quer zum Tor erfolgt, ist bei vielen Analysen relevant. Ist jedoch

eine Rotation der Bewegungen erlaubt, so wird bei den vorgestellten Ansätzen zur Erkennung der Muster kein Unterschied zwischen den Bewegungsrichtungen gemacht. Die resultierenden Muster können demnach sämtliche Bewegungsrichtungen enthalten. Dies ermöglicht u.a. eine Analyse der Kurvigkeit der Bewegungen, d.h. ob ein Spieler dazu neigt häufig die Richtung zu ändern, wobei die Richtung selbst keine Rolle spielt. Die Eignung hinsichtlich der Analyse von Laufwegen der Spieler, wo nur die Richtungen nicht aber der Ort auf dem Feld relevant sind, ist deutlich einschränkt.

Gruppenbewegungsmuster

Unter der Verwendung des sequenzbasierten Ansatzes und mit den Parametern $k = 128$ Cluster sowie $suppc_{min} = 2$ werden im gleichen Datensatz zudem 207 Gruppenbewegungsmuster gefunden. Abbildung 7.14 zeigt beispielhaft die Bewegungen eines Teams über 6 Sekunden (Sequenzelemente 408-413 und 831-836). Dieses Muster besteht aus drei verschiedenen Konstellationstypen (Clustern, je zwei grün, dunkelblau und blau) und findet sich in zwei Instanzen im Datensatz wieder. Dies bedeutet, dass sich die fünf (rot markierten) Spieler während der gesamten Beobachtungszeit zwei Mal als Teilgruppe von oben nach unten im Bild (bzw. von rechts nach links auf dem Spielfeld) bewegen. Dieses Muster zeigt ein typisches Defensivverhalten, in dem ein Team die Seiten verlagert, um Druck auf das ballbesitzende gegnerische Team auszuüben. Der einsame Spieler ganz rechts ist der Torhüter des Teams.

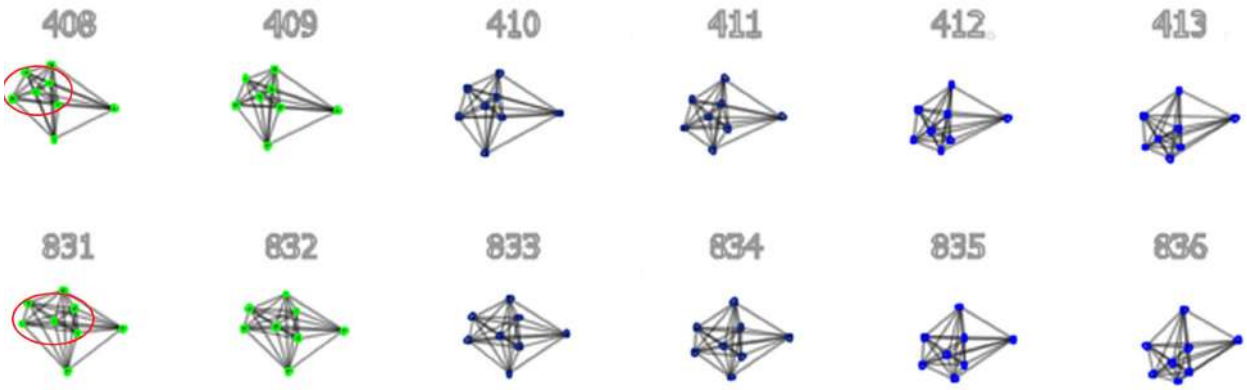


Abbildung 7.14: Ein Beispielergebnis für ein Gruppenbewegungsmuster, das zweimal beobachtet wird. Interessant für die Interpretation sind rot markierten Spieler.

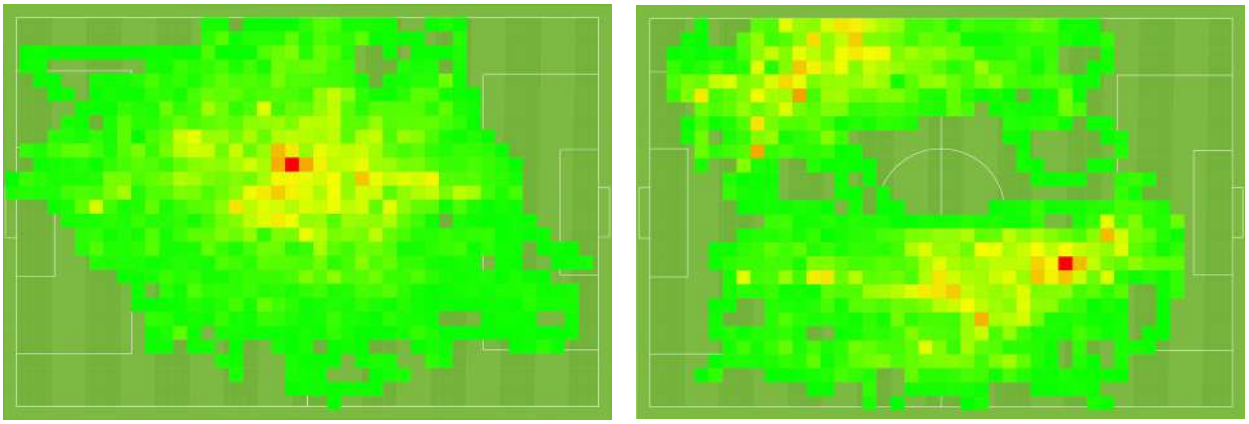
Fazit und Verwendung der Bewegungsmuster

Die mit diesen Ansätzen erkannten Muster geben einen Einblick in die Taktik eines Teams. Sie enthalten wertvolle Informationen, die von den Trainern zur Bewertung der Leistung des eigenen Teams (Einhaltung der Vorgaben) und zur Vorbereitung auf den nächsten Gegner (*Scouting*) genutzt werden können. Wird dieses Wissen an die Spieler weitergegeben, dann können diese es zur Prädiktion der nächsten Bewegungen des Gegenspielers nutzen und sich so besser auf diesen einstellen.

7.5.3 Experiment 2 - GPS-Fußball-Datensatz

Im Gegensatz zum ersten Experiment, in dem nur die Daten eines Fußballspiels verarbeitet worden sind, wird in diesem Experiment der zweite beschriebene Datensatz verwendet, der Daten aus mehreren Spielen und daher auch mehr Bewegungsinformationen von jedem einzelnen Spieler enthält. Aus diesem Grund werden dieses Mal die Spieler einzeln prozessiert, um auf diese Weise spielerspezifische Bewegungsmuster zu finden. Im ersten Experiment wurde nicht zwischen den

Trajektorien unterschiedlicher Spieler unterschieden. Die Aussagekraft der Muster in Bezug auf ein spielertypisches Bewegungsverhalten ist daher gering. Es können dort nur Aussagen in Bezug auf allgemeines fußballtypisches Bewegungsverhalten getroffen werden. Daher werden dieses Mal Spieler mit unterschiedlichen Rollen ausgewählt, um herauszufinden, ob sich rollenspezifische Muster ergeben. Es werden dazu ein zentraler Mittelfeldspieler und ein Flügelspieler, deren Heatmaps (Abbildung 7.15) die typischen Aufenthaltsorte während eines Spiels zeigen, miteinander verglichen. Zu diesem Zweck werden sämtliche Spiele ausgewertet, in denen die Spieler teilgenommen haben.



(a) *(b)*

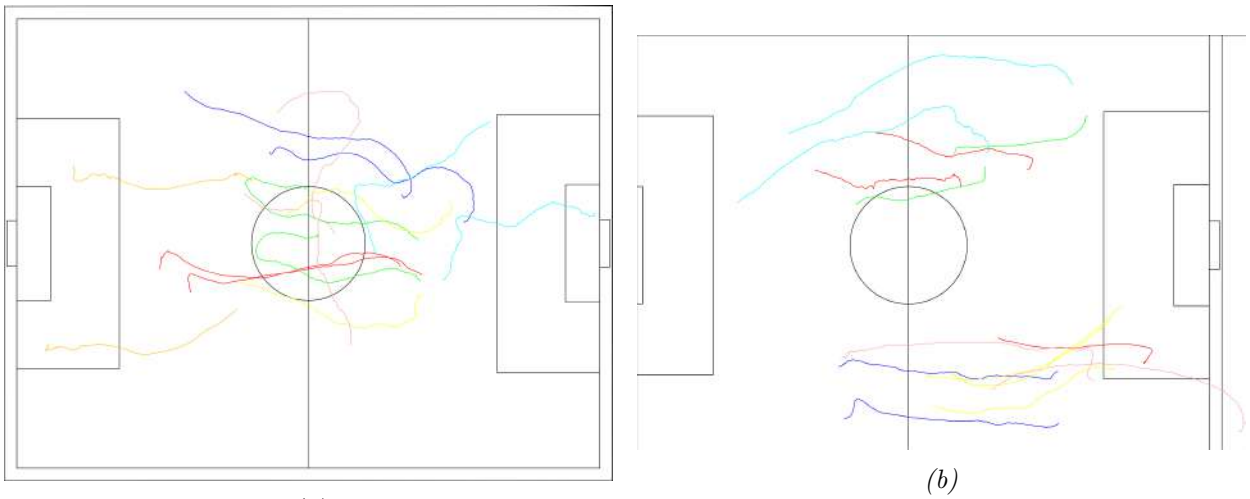
Abbildung 7.15: Typische Heatmaps für einen zentralen Mittelfeldspieler (a) und einen Flügelspieler (b) während eines Fußballspiels.

Sequenzbasierter Ansatz

Da in diesem Experiment die Verhaltensweisen die Laufwege der Spieler unabhängig vom Ort untersucht werden sollen, dürfen die gesuchten Muster auch verschoben sein. Eine Translationsinvarianz wird demnach berücksichtigt. Wie im vorangegangenen Experiment erläutert, wäre eine zusätzliche Rotationsinvarianz in dieser Analyse nicht zielführend. Die zur Erkennung erforderlichen Parameter werden entsprechend der Parameterstudie auf $k = 8$ sowie $suppc_{min} = 2$ und $l_{min} = 10\ m$ eingestellt.

Die Abbildung 7.16 zeigt einige der extrahierten Muster für beide Spieler. Die Trajektorien der Muster unterscheiden sich in ihrer Gestalt. Die Laufwege des Flügelspielers (Abbildung 7.16b) sind hauptsächlich geradlinige Läufe mit nur wenigen bzw. geringen Richtungsänderungen. Im Gegensatz dazu beinhalten die Bewegungen der zentralen Mittelfeldspielers (7.16a) deutlich mehr Richtungsänderungen. Dieses passt im Allgemeinen auch auf die typischen Verhaltensweisen beider Spielerrollen. Während ein zentraler Mittelfeldspieler generell mehr Freiheiten in der Wahl der Bewegungsrichtung in den meisten Spielsituationen genießt, muss der Flügelspieler die meiste Zeit seine Position auf der Außenseite des Spielfelds halten, um das Spiel der eigenen Mannschaft möglichst „breit“ anzulegen. Letzteres dient dazu, der gegnerischen Mannschaft die Verteidigung zu erschweren. Es ist auch möglich, die Ergebnisse aus taktischer Sicht zu untersuchen. Dieses ist insbesondere für Trainer oder Scouts als spätere Nutzer von Systemen, die ein derartiges Wissen bereitstellen, sehr interessant. Beispielsweise ist es möglich, unterschiedliches Verhalten des gleichen Außenspielers in beiden Halbzeiten zu beobachten. Die unterschiedlichen Spielrichtungen wurde hierbei natürlich herausgerechnet. Werden zu diesem Zweck die unteren drei Muster (blau, gelb und pink), die aus der ersten Halbzeit stammen, mit den oberen drei Mustern (cyan, rot und grün) aus der zweiten Halbzeit verglichen, so zeigen speziell die in cyan und blau hervorgehobenen Muster unterschiedliche Charakteristika auf. Während der Spieler in der ersten Halbzeit (blaues Muster)

länger auf dem Flügel blieb und sich daher recht gerade der Linie entlang bewegte, verließ er in der zweiten Spielhälfte (cyan) seine Position ziemlich früh (bereits einige Meter vor der Mittellinie) und lief direkt Richtung gegnerisches Tor. Der Grund für die Veränderung des Verhaltens ist mit diesen Daten und dieser Analyse nicht zu ermitteln. Nichtsdestotrotz ergeben sich wichtige Informationen, die von Trainern genutzt werden können, um entweder die Leistung der eigenen Spieler objektiver zu bewerten oder das eigene Team auf das Verhalten gegnerischer Spieler vorzubereiten.



(a) *(b)*

Abbildung 7.16: Ein Vergleich von Bewegungsmustern von Spielern mit unterschiedlichen Rollen: a) zentraler Mittelfeldspieler, b) Flügelspieler

Fazit und Verwendung der Bewegungsmuster

Diese Beispielanalyse zeigt, dass sich die Bewegungsmuster der beobachteten Spieler unterscheiden. Dieses Wissen kann ebenfalls im umgekehrten Sinn genutzt werden. So können bspw. anhand von gegebenen Laufwegen Spielerrollen, bzw. bei noch detaillierteren Analysen, die tatsächlichen Spieler identifiziert werden. Diese Art von Information kann beim Spieler-Tracking verwendet werden, sodass ein zeitweise verdeckter Spieler automatisch wiedererkannt werden kann. Dieses lässt sich entsprechend auf einen anderen Kontext übertragen.

7.5.4 Experiment 3 - MapConstruction.org

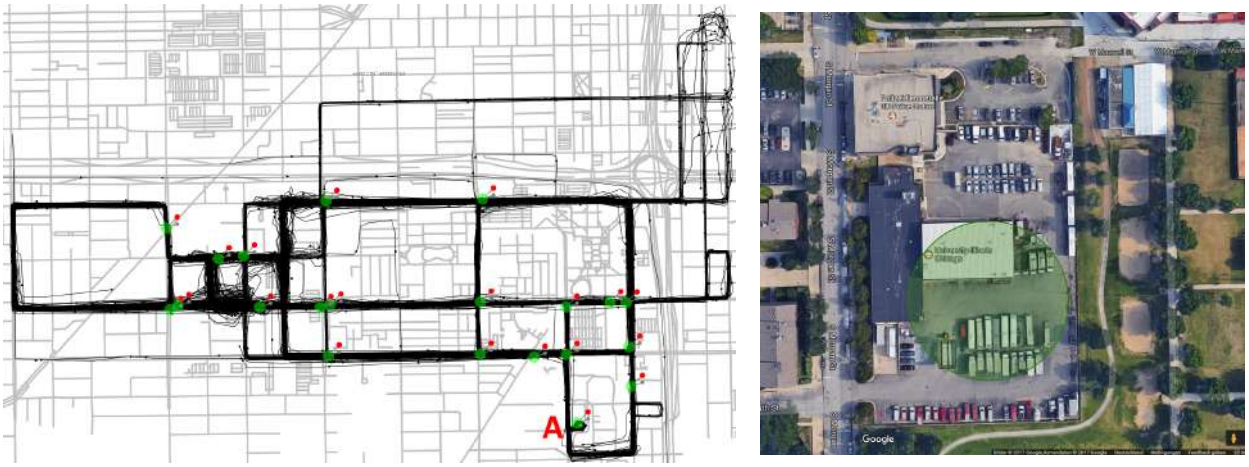
Um die Portabilität des vorgestellten Ansatzes zu zeigen, werden in diesem Experiment Bewegungsdaten aus dem Verkehrskontext analysiert. Der Algorithmus funktioniert weiterhin in der gleichen Art und Weise. Jedoch müssen die speziellen Charakteristika dieses Datensatzes berücksichtigt werden. Unter anderem werden die Bewegungen der Objekte stark vom vorliegenden Bewegungsraum, nämlich dem Straßennetzwerk, beeinflusst. Dies hat zur Folge, dass die Trajektorien sich dem Straßennetzwerk anpassen. Bei der Suche nach Mustern wird daher auf jegliche Invarianz verzichtet. Es würden zwar ebenfalls Ergebnisse gefunden werden, diese würden jedoch lediglich wiederkehrende Netzwerkstrukturen aufzeigen, die hier nicht von Interesse sind.

Clustering-basierter Ansatz

Der Clustering-basierte Ansatz arbeitet auf Trajektoriesegmenten und erfordert eine vorausgehende Segmentierung. In diesem Fall is jedoch kein Vorwissen über eine sinnvolle Segmentierung vorhanden und einfache parameter- bzw. geometriebasierte Methoden liefern ebenfalls keine bedeutungsvollen Ergebnisse. Aus diesem Grund wird zuerst nach interessanten Plätzen gesucht, die

anschließend zur Identifikation der Segmente verwendet werden. Die für die Detektion benötigten Parameter werden wie folgt gesetzt: Die Ausdehnung der Plätze wird in diesem Fall an die GPS-Genauigkeit, die in der Stadt eher im Bereich von 10 - 20 m liegt, angepasst und auf $R = 20\ m$ gesetzt. Der Geschwindigkeitsschwellwert wird mit $v_{max} = 0.1\ m/s$ niedrig gewählt. Die erforderlichen Besuche eines Platzes wird auf $N_{IP} = 10$ festgelegt und so an die Dichte der Daten angepasst, um zufällige Interessante Plätze zu vermeiden.

Zur Evaluation der detektierten Plätze werden diese mit Hilfe einer überlagerten Kartenansicht (siehe Abbildung 7.17a) visuell inspiziert. Dabei wird deutlich, dass die Plätze sich nahezu ausschließlich an Kreuzungen befinden. Dies ist erklärbar, da dort die Busse häufig auf Grund der Verkehrsregeln anhalten müssen. Zudem befinden sich an den gleichen Orten ebenfalls Bushaltestellen, an denen die Busse halten. Auf einen erkannten Platz (A) trifft dies jedoch nicht zu. Bei näherer Betrachtung stellt sich heraus, dass es sich dort um den Parkplatz der *UIC Transportation Facility* handelt, auf dem die Busse abgestellt werden (siehe Abbildung 7.17b). Die extrahierten Plätze sind demnach interessant und ermöglichen eine bedeutungsvolle Segmentierung in 1816 Trajektoriesegmente.



(a) *(b)*

Abbildung 7.17: a) Eine Übersicht über die detektierten interessanten Plätze (grüne Kreise mit einem roten Pin). b) Einer der interessanten Plätze (A) ist der Parkplatz der UIC Transportation Facility, an dem die Busse abgestellt werden.[1]

Das Clustering erfolgt mit Hilfe des DBSCAN-Algorithmus mit $\epsilon = 10\,m$ und $minIts = suppc_{min} = 2$ als Parameter. Mit diesem Setting werden 44 Muster erkannt, von denen einige beispielhaft in Abbildung 7.18 dargestellt sind. Der Mittelwert für die Länge liegt in diesem Fall bei ca. 395 m, der des Support Counts bei 3,7 und der der Ähnlichkeit bei ca. 0,25 $\frac{1}{m}$, was nach Gleichung 7.3 einer maximalen Fréchet-Distanz von ca. 5 m zwischen den Instanzen entspricht. Für die Gesamtinteressantheit ergibt sich ein Wert von 25169,90. Werden diese Kennwerte mit denen der Parameterstudie, wo der ACM DEBS 2013-Datensatz verwendet wird, verglichen, so wird dort nicht annähernd so eine hohe Gesamtinteressantheit erreicht. Diese liegt dort nur bei 2881, obwohl deutlich mehr Muster (115) gefunden werden. Dieses lässt sich hauptsächlich durch die in diesem Fall wesentlich höhere durchschnittliche Länge (verglichen mit 5,32 m) erklären, während die übrigen Werte (mittlerer Support Count: 5,3, mittlere Ähnlichkeit: 0,22 $\frac{1}{m}$) vergleichbar sind. Obwohl der ACM DEBS 2013-Datensatz ca. die doppelte Menge an Daten enthält, ist jedoch die Datendichte des hier verwendeten Datensatzes auf Grund des darunterliegenden Netzwerkbewe-

[1] Hintergrundkarte: Google Maps

gungsraums höher. Diese Tatsache erhöht die Wahrscheinlichkeit, überhaupt bzw. längere Muster zu finden.

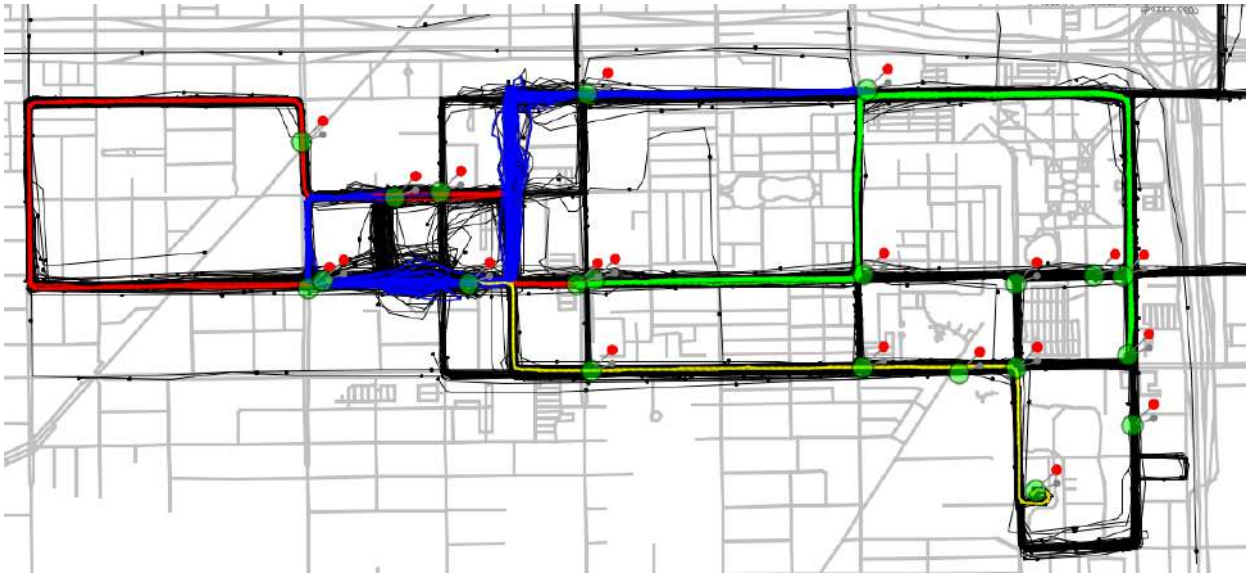


Abbildung 7.18: Vier mit dem Clustering-basierten Ansatz erkannte Bewegungsmuster. Die Farben stehen für unterschiedliche Muster.

Der Einfluss des ϵ-Parameters ist in diesem Szenario begrenzt, da die Trajektorien durch den Zugrunde liegenden Bewegungsraum bereits relative ähnlich sind. Sollten zwei Plätze durch viele Segmente miteinander verbunden sein (siehe Abbildung 7.19), die sich hinsichtlich ihrer Fahrtrichtung unterscheiden, so würden diese bereits durch die Verwendung der Fréchet-Distanz, die bekanntlich die Bewegungsrichtung berücksichtigt, unterschiedlichen Clustern bzw. Mustern zugewiesen werden. Eine Identifikation von Clustern, die unterschiedliche Fahrspuren in die gleiche Richtung repräsentieren, ist durch die GPS-Ungenauigkeit und die Verwendung eines global gültigen Dichte-Parameters nicht möglich.

Da in diesem Szenario die Busse in der Regel auf ihren vorgegeben Routen fahren und sich der Datensatz über eine Zeitspanne erstreckt, sodass die Routen mehrfach abgefahren werden, müssten sich daraus Muster ergeben, die diesen Routen entsprechen. Mit diesem Ansatz lassen sich diese Muster jedoch nicht identifizieren, da die Segmentierung die Routen in Segmente unterteilt. Es werden demnach nur Teilstücke der Routen erkannt, die sich zwischen zwei interessanten Plätzen erstrecken. Zur Extraktion der Routen-Muster kann der sequenzbasierte Ansatz genutzt werden, der ohne Segmentierung auskommt.

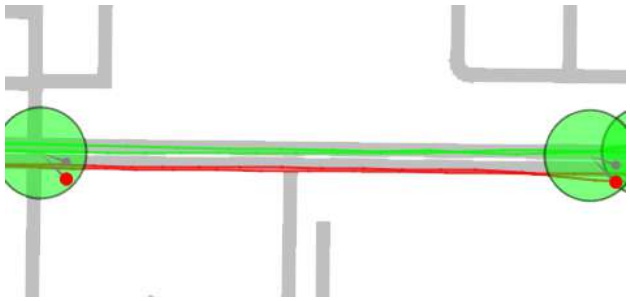


Abbildung 7.19: Nahe beieinander liegende gegenläufige Segmente werden durch Verwendung der Fréchet-Distanz unterschiedlichen Clustern zugewiesen. Eine Erkennung der einzelnen Fahrspuren in die gleiche Richtung ist jedoch nicht möglich.

Sequenzbasierter Ansatz

Zur Verarbeitung des gleichen Datensatzes mit dem sequenzbasierten Ansatz wird das folgende Parametersetting festgelegt. Der Parameter k des k-means-Clustering-Verfahrens, der die Länge des Alphabets und damit die Toleranz bzgl. der Abweichungen unter den Instanzen eines Musters steuert, wird in diesem Experiment auf $k = 64$ gesetzt. Zur Identifikation der Muster, wird der Apriori-Algorithmus mit den Parametern $suppc_{min} = 2$ und $l_{min} = 100\ m$ verwendet. Mit diesem Setting werden insgesamt 207 Bewegungsmuster mit einer mittleren Länge von 3,2 km, einem mittleren Support Count von 7,6 sowie einer durchschnittlichen Ähnlichkeit von 0.01 $\frac{1}{m}$ identifiziert. Im Vergleich zum Ergebnis des Clustering-basierten Ansatzes werden annähernd doppelt so viele Muster mit fast doppelt so vielen Instanzen erkannt. Dass die Gesamtinteressantheit mit ca. 42717 deutlich höher ist, liegt daran, dass zum einen deutlich mehr Muster erkannt werden und dass sich deren mittlere Länge ebenfalls deutlich erhöht hat. Dieses lässt sich dadurch begründen, dass hier keine Segmentierung durchgeführt wird, die für kürzere Trajektoriesegmente und damit eine eingeschränkte Maximallänge der Muster sorgt.

Im Gegensatz zum Clustering-basierten Ansatz befinden sich in der Ergebnismenge dieses Ansatzes auch die Buslinien-Muster. In Abbildung 7.20 werden diese Muster und die entsprechenden offiziellen Routen der Busse (*Semester Express*, *Intracampus Evening*) dargestellt. Die Daten und Abbildungen stammen von dem Bus-Tracking-Service[1], der durch die UIC mittels *DoubleMAP*[2] bereitgestellt wird. Die Übereinstimmungen sind, bis auf eine kleine Abweichung bei der zweiten Linie, deutlich zu erkennen. Die Gründe für diese Abweichung sind nicht bekannt. Sie könnte aber aus einer Änderung der Route oder einer bspw. temporär geänderten Verkehrsführung resultieren.

Abbildung 7.20: Ein Vergleich der erkannten Muster mit der Realität: In der linken Spalte werden die erkannten Bewegungsmuster der Busse dargestellt, in der Rechten die offiziellen Routen der Buslinien[3].

[1] Intracampus Bus Service: http://uic.doublemap.com/map/. Zugriff am 27.11.2017
[2] DoubleMAP: https://www.doublemap.com/. Zugriff am 27.11.2017
[3] Bildquellen: http://uic.doublemap.com/map/. Zugriff am 27.11.2017

Fazit und Verwendung der Bewegungsmuster

Durch die Ergebnisse dieses Experiments wird deutlich, dass die Bewegungsmustererkennung auch im Kontext „Verkehr“ funktioniert. Die auf diese Weise erkannten Bewegungsmuster beinhalten Informationen, die in unterschiedlichen Bereichen genutzt werden: Städteplaner oder -management können anhand dieses Wissens das Straßennetz und das Verkehrsmanagement optimieren. Navigationssysteme können diese Informationen zur Prädiktion der nächsten Nutzer-Trajektorie verwenden. Auf diese Weise könnten sie dem Nutzer bereits beim Start das nächste Ziel vorschlagen oder gar schon die optimale Route berechnen, ohne dass der Nutzer überhaupt Eingaben bzgl. des nächsten Ziels machen muss. Im selben Kontext können die Informationen auch in kollaborativer Art und Weise benutzt werden, indem sie zwischen den Verkehrsteilnehmern geteilt werden, um den Verkehr in dem entsprechenden Gebiet zu prädizieren.

7.5.5 Experiment 4 - Mantelpaviane

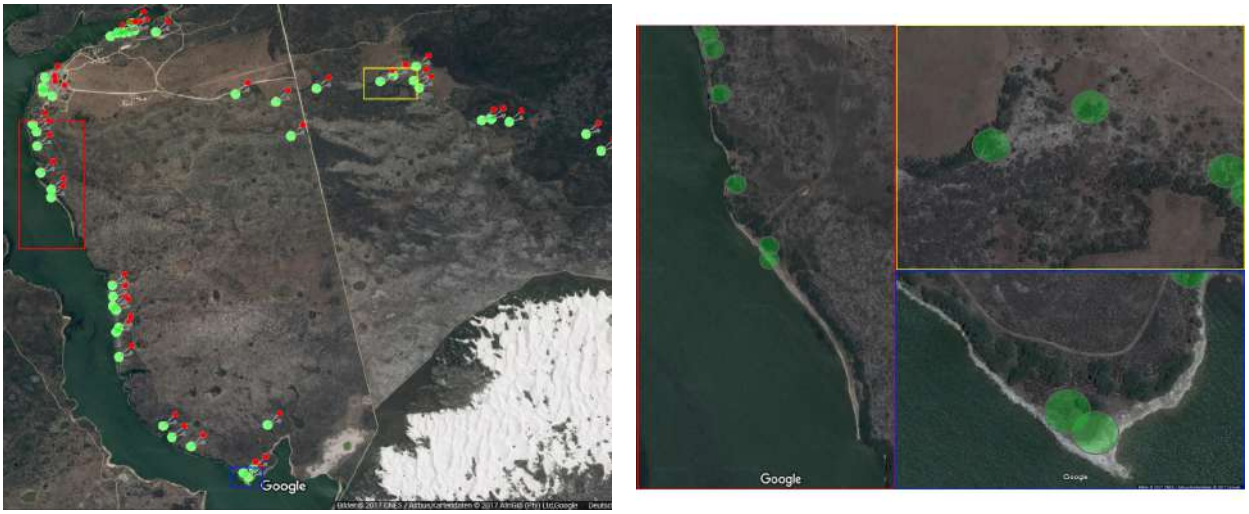
Im letzten Experiment wird ein weiteres Mal der Kontext gewechselt. Dieses Mal werden die Trajektorien von Tieren untersucht. Es handelt sich dabei um eine Gruppe von 14 Mantelpavianen, die sich durch ein Gebiet von ca. 4,7 km x 4.8 km bewegen. In diesem Datensatz sind die räumliche und zeitliche Auflösung vergleichsweise niedrig. Diese Charakteristika müssen in den Analysen berücksichtigt werden.

Clustering-basierter Ansatz

Da wie im vorherigen Experiment auch hier kein Vorwissen über eine sinnvolle Segmentierung vorhanden ist, erfolgt Letztere auf Basis von detektierten interessanten Plätzen. Zur Erkennung dieser Plätze werden die Parameter $R = 50\ m$, $v_{max} = 0.01\ m/s$ und $N_{IP} = 28$ verwendet. Hierbei wird berücksichtigt, dass die Tiere als Gruppe unterwegs sind. Die Anzahl der erforderlichen Besuche wird daher auf das Doppelte der Gruppengröße festgelegt, um sicherzustellen, dass die Gruppe diesen Platz mehr als einmal aufgesucht hat. Die Ausdehnung der Plätze wird ebenfalls an das weitläufige Gebiet und eine möglicherweise etwas zerstreute Gruppe angepasst. Mit diesem Parametersetting werden 46 Plätze gefunden, deren räumliche Verteilung in Abbildung 7.21a gezeigt wird. Bei näherer Betrachtung wird deutlich, dass sich die Plätze überwiegend an Stellen mit Pflanzenbewuchs und in der Nähe von Wasser befinden (siehe Abbildung 7.21b). Bei den in dieser Regionen üblichen hohen Temperaturen, ist dies nicht weiter verwunderlich. Die Segmentierung auf Basis dieser Plätze ergibt 1632 Segmente.

Auch bei der Suche nach Mustern in den Bewegungen zwischen diesen interessanten Plätzen spielt die Tatsache, dass hier eine Gruppe von Tieren beobachtet worden ist, eine Rolle. Denn dieses muss auch bei der Wahl der Musterkriterien berücksichtigt werden. Für die Mindestanzahl an Instanzen pro Muster würde ein Wert, der nicht oberhalb der Gruppengröße liegt, wenig Sinn machen, da sonst jegliche Bewegungen der Gruppe abhängig von den Clustering-Parametern als potenzielles Muster in Frage kommt. Diese Muster wären zwar im Sinn des vorgestellten Mustererkennungsprozesses valide, ergeben aber unter den in diesem Szenario vorherrschenden Bedingungen wenig Aufschluss über die Bewegungen der gesamten Gruppe. Als Parameter werden daher $\epsilon = 50\ m$ und $minIts = suppc_{min} = 15$ festgelegt.

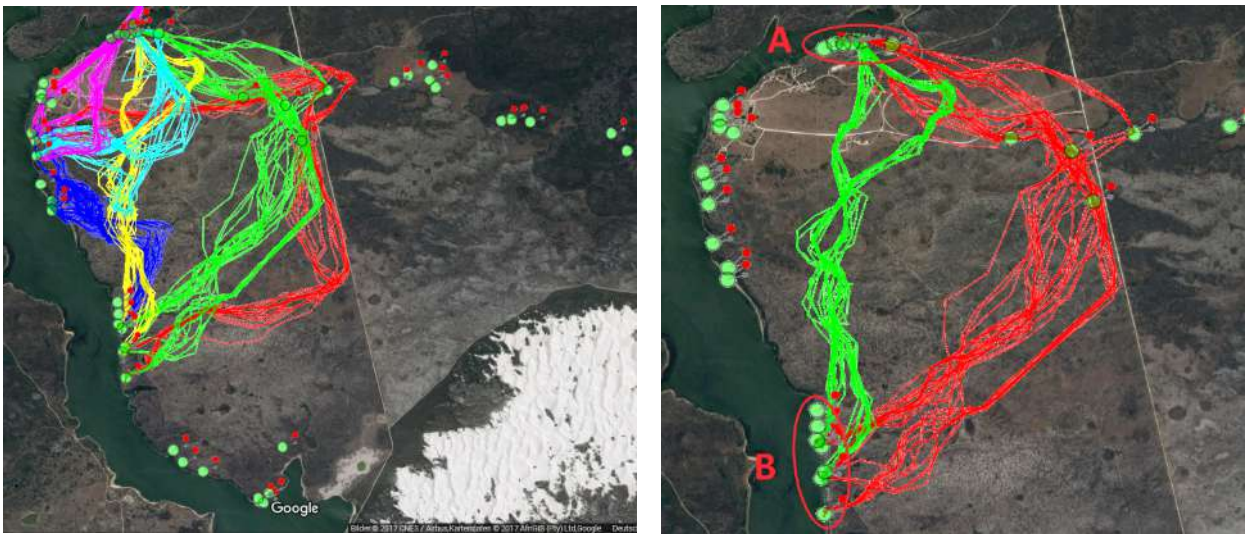
Mit diesen Einstellungen werden 14 Muster gefunden, von denen sechs in Abbildung 7.22 dargestellt sind. Die Gesamtinteressantheit der Muster liegt bei nur ca. 1352, obwohl der mittlere Support Count mit 77.5 und die mittlere Länge mit ca. 2,4 km recht hoch sind. Die durchschnittliche Ähnlichkeit ist jedoch mit ca. 0.002 sehr gering. Dies resultiert durch die den hohen Wert für ϵ.



(a) *(b)*

Abbildung 7.21: a) Die räumliche Verteilung der detektierten interessanten Aufenthaltsorte der Paviane.[1]. b) Die Plätze befinden sich überwiegend an Grünstellen mit schattenspendendem Pflanzenbewuchs und in der Nähe von Wasser[1].

Die ermittelten Bewegungsmuster zeigen typische Routen der Tiere. Die in Abbildung 7.22 gezeigten Muster besitzen eine Support Counts im Bereich von 16 bis 49. Dies bedeutet, dass die dadurch aufgezeigten Routen in der verhältnismäßig kurzen Zeitspanne (bezogen auf die Größe des Gebiets) mindestens zweimal genutzt wurden.



(a) *(b)*

Abbildung 7.22: a) Sechs der insgesamt 14 erkannten Bewegungsmuster. Die verschiedenen Muster werden durch unterschiedliche Farben gekennzeichnet[1]. b) Zwischen zwei Regionen (A, B) zeigen die Muster die verschiedene Routen der Tiere (grün, rot)[1].

Weitere interessante Informationen ergeben sich, wenn die verschiedenen Routen zwischen Gruppen von Plätzen, die nahe beieinander liegen, untersucht werden. In Abbildung 7.22b sind einige Plätze zu Regionen (A, B) zusammenfasst worden. Zudem werden dort die Muster gezeigt, die diese

[1]Hintergrundkarten: Google maps. Zugriff am 27.11.2017

Regionen verbinden. Dabei ist zu erkennen, dass es anscheinend zwei typische Routen von der einen zur anderen Region gibt. Wird bei der Betrachtung dieser Routen berücksichtigt, dass bei der Mustersuche die Fréchet-Distanz benutzt wird, bedeutet das, dass sie jedes Mal jeweils in die gleiche Richtung abgelaufen werden.

Fazit und Verwendung der Bewegungsmuster

Auf Grundlage der Trajektorien der Tiere lassen sich sinnvolle interessante Plätze detektieren. Mit deren Hilfe lassen sich die langen Trajektorien in der Art segmentieren, dass Bewegungsmuster erkannt werden können. Wie in der darauffolgenden Analyse bzw. Diskussion der Muster ansatzweise gezeigt wird, können die ermittelten Bewegungsmuster genutzt werden, um Erkenntnisse über das gewöhnliche Bewegungsverhalten und damit auch über die Tiere selbst zu erlangen.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

Diese Arbeit beschäftigt sich mit der Analyse der Bewegungen von Objekten. Diese lassen sich mit Hilfe von Trajektorien beschreiben, welche wiederum die Grundlage für eine Bewegungsmustererkennung darstellen. Diese Muster enthalten wertvolle Informationen über das Objektverhalten, die in unterschiedlichen Anwendungs- bzw. Forschungsbereichen genutzt werden können. Aus dieser Motivation heraus ergeben sich mit der Erfassung dieser Trajektorien und der Erkennung der Bewegungsmuster zwei Problemstellungen, zu denen im Rahmen dieser Arbeit mögliche Lösungsansätze dargestellt werden.

Zunächst wird für beide Problemstellungen das benötigte Grundlagenwissen beschrieben. Dieses umfasst neben der allgemeinen Modellierung von Bewegungen durch Trajektorien auch die relevanten Grundlagen zu deren Erfassung. Dabei werden insbesondere die unterschiedlichen Verfahren zur Lokalisierung und zur Verfolgung der Objekte erläutert. Da die Messungen der verschiedenen Technologien mit entsprechenden Unsicherheiten behaftet sind, wird dabei zudem das Konzept von Graphischen Modellen als Möglichkeit der stochastischen Modellierung erläutert. Im Zusammenhang mit der Mustererkennung werden zusätzlich das allgemeine Vorgehen beim Data-Mining, die Vorverarbeitung der Daten, die Verfahren zur Bestimmung der Ähnlichkeit von Trajektorien und die dazu relevanten Verfahren aus dem Bereich des Maschinellen Lernens sowie der Sequenzmustererkennung detailliert beschrieben.

Darauffolgend wird der aktuelle Stand der Forschung in beiden Bereichen beschrieben, indem die existierende Literatur angegeben wird. Im Kontext der Datenerfassung werden neben der Betrachtung existierender kommerzielle Systemlösungen auch die relevanten Ansätze zur Objektdetektion und -verfolgung sowie zur Fusion heterogener Daten untersucht und diskutiert. Hierbei wird insbesondere auf die Ansätze zum Umgang mit fehlenden Objektdetektion eingegangen. Bei der Diskussion der relevanten Literatur bezogen auf die Erkennung von Bewegungsmustern in Trajektorien werden die verschiedenen Vorgehensweisen berücksichtigt, die davon abhängen, ob die gesuchten Muster von vornherein bekannt oder unbekannt sind.

Auf Grundlage der gegebenen Motivation und der diskutierten Vorarbeiten wird in dieser Arbeit ein Ansatz zur Erfassung von Trajektorien vorgestellt, der auf der Kombination von Kameras und GNSS-Empfängern basiert. Diese unterschiedlichen Sensoren unterstützen sich gegenseitig, um hoch-genaue und durchgehende Trajektorien der beobachteten Objekte zu erzeugen. Dabei ist neben der Objektdetektion und -verfolgung zusätzlich das Problem der Verarbeitung von heterogenen Sensordaten zu lösen. Bei diesen Daten handelt es sich um die GNSS-Informationen und die Detektionen in den Kamerabildern. Letztere ermöglichen zwar eine sehr genaue Objektlokalisierung, sind allerdings in der Regel, bedingt durch Verdeckungen, unvollständig und die Objektidentität ist unbekannt. Bei den GNSS-Informationen verhält es sich umgekehrt. Diese sind im Allgemeinen vollständig und können den beobachteten Objekten zugeordnet werden. Jedoch leiden die Positionsinformationen an der typischen GNSS-Ungenauigkeit (5 - 20 m). Zur Fusion dieser Daten verwendet der entwickelte Ansatz Hidden Markov Modelle. Diese liegen in zwei unterschiedlichen Ausprägungen vor, die jeweils ihre Vor- und Nachteile haben.

In der ersten Variante repräsentieren die verborgenen Zustände des HMMs die nicht direkt-beobachtbaren Zuordnungen zwischen Video- und GNSS-Beobachtungen. Unter der Annahme, dass die anhand der Bildinformationen ermittelten Objektpositionen wesentlich genauer als die GNSS-Positionen sind, werden die resultierenden Trajektorien lediglich auf Grundlage der Detektionen im Bild generiert. Die GNSS-Trajektorien werden dabei als Beobachtung zur Ermittlung der korrekten Objektidentitäten benutzt. Ein Vorteil dieser Variante ist die im Vergleich zur Alternative geringere Laufzeit. Als Problem kann jedoch die Einstellung der Wahrscheinlichkeiten der benötigten „Dummies“ zum Umgang mit fehlenden Detektionen angesehen werden. Diese erweist sich als nicht trivial und hängt vom Anwendungsfall ab.

Die zweite Variante hingegen ist die rasterbasierte Modellierung, die eben auf Grund der Problematik der „Dummies“ entwickelt worden ist. Hier wird das beobachtete Gebiet mit Hilfe eines Rasters diskretisiert. Die Rasterzellen entsprechen den möglichen Objektpositionen und werden im HMM als verborgene Zustände modelliert. Gesucht wird dementsprechend die wahrscheinlichste Sequenz aus Rasterzellen, die die Positionen für die Trajektorien liefern. Da hier beide, sowohl die Bild- als auch die GNSS-Informationen, als Beobachtungen in das Modell einfließen, handelt es sich bei dieser Variante um ein Fusion der Positionsinformationen. Dabei werden die Unsicherheiten beider Messungen berücksichtigt. Der Vorteil dieser Modellierung liegt in dem Verzicht auf den sogenannten „Dummy“-Detektionen. Denn in dieser Variante kann auch dann eine Position ermittelt werden, wenn einer oder beide Sensoren keine Informationen liefern. Ein Nachteil ist der höhere Berechnungsaufwand, der u.a. von der gewählten Zellgröße des Rasters abhängt. Von dieser hängt wiederum die geometrische Genauigkeit der Ergebnisse ab, da die Zellmittelpunkte für die Trajektoriepunkte verwendet werden.

Anhand verschiedener Experimente wird der vorgestellte Lösungsansatz hinsichtlich unterschiedlicher Eigenschaften evaluiert. Es wird sowohl die Korrektheit des Objekt-Trackings als auch die geometrische Genauigkeit der resultierenden Trajektorien untersucht. Zu diesem Zweck wird das Verfahren auf reale Datensätze angewendet und dessen Ergebnisse diskutiert. Das Fazit der Diskussion ist, dass das GNSS-unterstützte Kamera-Tracking eine kostengünstige Lösung darstellt, die für viele Anwendungen ausreichend genaue Ergebnisse liefert. Der Einsatz dieses Systems stellt jedoch gewisse Anforderungen an das Anwendungsszenario. So müssen die Kameras fest installiert werden und die beobachteten Objekte sich mit GPS-Empfänger ausstatten lassen.

Im Zusammenhang mit der Mustererkennung, der zweiten Problemstellung dieser Arbeit, wird ein Ansatz vorgestellt, der sich am Knowledge Discovery from Data(bases)-Prozess orientiert. Dieser besteht ebenfalls aus zwei alternativen Vorgehensweisen, die sich jedoch mit Trajektorien die gleiche Datengrundlage teilen. Die erste Variante zur Identifikation der gesuchten Bewegungsmuster ist ein Clustering-basierter Ansatz. In diesem werden ähnliche Bewegungen mittels Clustering-Verfahren (z.B. k-means oder DBSCAN) zu Clustern zusammengefasst. Die resultierenden Cluster entsprechen den gesuchten Bewegungsmustern, die Cluster-Elemente den jeweiligen Musterinstanzen. Das Kernproblem hierbei besteht darin, dass die Trajektorien in vielen Fällen (nach einer entsprechend langen Beobachtungszeit) zu lang sind, um sie direkt und ohne Vorverarbeitung miteinander vergleichen zu können. Zudem ist davon auszugehen, dass nicht alle Bereiche innerhalb einer Trajektorie für eine Analyse relevant sind. Aus diesem Grund ist eine Segmentierung erforderlich, die die Trajektorien in bedeutungsvolle Segmente zerlegt und auf diese Weise vergleichbar macht. Zur Segmentierung selbst, die bei diesem Ansatz das Hauptproblem darstellt, werden unterschiedliche Verfahren entwickelt, mit denen bedeutungsvolle Schnittpunkte innerhalb der Trajektorien bestimmt werden. Eine Möglichkeit bietet die Identifikation interessanter Plätze. Diese stellen das Bewegungsverhalten der beobachteten Objekte in der Art dar, dass sie Orte aufzeigen, die entweder eine gewisse Attraktivität ausstrahlen oder häufig als Ein- bzw. Ausstiegspunkt in die beobachtete Szene dienen. Da diese Plätze somit häufig als Start-, Ziel- oder Wegpunkt der

Objektbewegungen angesehen werden können, können sie als Grundlage für eine Segmentierung dienen. Eine andere Möglichkeit der Segmentierung basiert auf der Identifikation von Zeitpunkten in den Trajektorien, an denen sich die Bewegungsintention eines Objekts ändert. Da die hierfür notwendigen Informationen jedoch nur selten vorliegen, werden diese mit Hilfe eines Klassifikationsverfahren, in diesem Fall der ID3-Algorithmus, und geeigneter Features eigens bestimmt.

Ein sequenzbasierter Ansatz stellt die zweite Variante zur Erkennung der Bewegungsmuster dar. Hierbei werden die im zwei-dimensionalen Raum vorliegenden Trajektorien in 1D-Element-Sequenzen transformiert. Zu diesem Zweck werden Clustering-Verfahren genutzt, um aus den vielen unterschiedlichen Einzelbewegungen ein Alphabet mit beschränkter Länge generieren, bei dem ähnliche Bewegungen einem Symbol zugeordnet werden. Das Alphabet wird daraufhin zur Beschreibung der Bewegungssequenzen benutzt. Die resultierenden Sequenzen werden danach mittels Sequenzanalyseverfahren (z.B. mit dem Apriori-Algorithmus) nach häufig wiederkehrenden Teilsequenzen durchsucht. Die gefundenen Teilsequenzen entsprechen den einzelnen Musterinstanzen und können zu den gesuchten Bewegungsmustern zusammengefasst werden. Im Gegensatz zur Clustering-basierten Variante kann diese auch zur Identifikation von Gruppenmustern verwendet werden. Die Erkennung folgt dem gleichen Prinzip. Allerdings werden in diesem Fall gleichzeitig sämtliche Objektbewegungen mit Hilfe von sogenannten Konstellationen berücksichtigt.

Wie bei der Erfassung der Trajektorien werden auch hier die Ergebnisse beider Herangehensweisen in verschiedenen Experimenten evaluiert. Mit Hilfe eines künstlich generierten Referenzdatensatzes werden die jeweiligen Ergebnisse verifiziert. Die Referenz-Muster werden in beiden Fällen erkannt. Die weiteren Experimente stützen sich auf Datensätze aus unterschiedlichen Anwendungsbereichen (Fußballanalyse, Verkehr, Tierbewegungen), was die Übertragbarkeit des Lösungsansatzes zeigt. In den entsprechenden Ergebnisdiskussionen wird deutlich, dass die erkannten Bewegungsmuster wertvolle Informationen enthalten und interessante weiterführende Analysen, auch hinsichtlich der Charakterisierung von Objekten und der Vorhersage von Bewegungen, ermöglichen.

Ein abschließendes Gesamtfazit soll sich den beiden gestellten Forschungsfragen widmen:

1) Lässt sich durch die Kombination aus GNSS- und videobasiertem Tracking eine Objekt-Tracking-Lösung verwirklichen, die trotz der Verwendung von Low-cost-Sensoren und insbesondere im Hinblick auf Objektverdeckungen vollständige und geometrisch genaue Trajektorien liefert?

Die Ergebnisse des entwickelten Ansatzes zur Erfassung von Trajektorien zeigen, dass dieser auch in Szenen mit einer signifikanten Zahl an Objektverdeckungen durchgängige und sehr genaue Trajektorien liefert. Die hierbei verwendeten Low-cost-Sensoren ermöglichen eine kostengünstige Lösung. Für die Szenarien, in denen die Anwendung beider Technologien möglich ist, kann diese Frage somit mit „Ja“ beantwortet werden.

2) Wie lassen sich unbekannte wiederkehrende Bewegungsmuster in Trajektorien erkennen?

Die Evaluation des im Rahmen dieser Arbeit entwickelten Mustererkennungsverfahrens hat gezeigt, dass beide Varianten dieses Ansatzes zur Erkennung wiederkehrender, a-priori unbekannter Bewegungsmuster in Trajektorien genutzt werden können. Die Entscheidung, welche der beiden Varianten genutzt werden sollte, hängt vom Anwendungsszenario und dem verfügbaren Kontextwissen ab.

8.2 Ausblick

Obwohl in den Experimenten gezeigt wird, dass die vorgestellten Ansätze funktionieren, bleiben einige offene Punkte, an denen sie verfeinert und erweitert werden können. Beim Erfassungsansatz gibt es im Bereich der Objektdetektion in den Kamerabildern Verbesserungspotential. Auf der einen Seite kann dieser robuster gemacht werden, wenn Vorabinformationen über die zu erwartenden Objektpositionen verfügbar sind. Diese können genutzt werden, um schwierige Situationen wie partielle Objektverdeckungen aufzulösen und so fehlende Detektionen zu vermeiden. Dies würde das Tracking der Objekte deutlich vereinfachen, da dort die fehlenden Detektionen ein Kernproblem sind. Auf der anderen Seite könnten auch neue Verfahren aus dem Bereich Deep Learning den bisherigen Detektionsansatz ersetzen, der mit der Anforderung, dass statische Kameras verwendet werden, gewisse Einschränkungen mit sich bringt. Verfahren, wie in Cao u. a. (2016) vorgestellt, besitzen eine höhere Robustheit, gerade was komplexere Szenarien mit vielen Objekten und Verdeckungen betrifft. Sie könnten daher genauere und vollständigere Detektionsergebnisse liefern. Allerdings benötigen diese Verfahren ein Training, das den späteren Gegebenheiten (z.B. Objektgrößen, Bildauflösung, usw.) entspricht. Das genannte Verfahren funktioniert außerordentlich gut bei Szenarien, die dem in Abbildung 8.1 ähneln. Eine Anwendung ohne erneutes Training auf die Videodaten, die im Zusammenhang mit der Fußballanalyse in dieser Arbeit aufgezeichnet worden sind, funktioniert nicht. Ein Grund hierfür sind großen Unterschiede in den Spielergrößen im Bild. Ein Spieler auf der gegenüberliegenden Seite des Spielfelds ist deutlich kleiner als einer im Vordergrund.

Abbildung 8.1: Zur Detektion der Objekte in Bildern können auch Deep Learning-Verfahren genutzt werden: bspw. der Ansatz nach Cao u. a. (2016) (gleichzeitig auch Bildquelle), der Skelette in die erkannten Personen schätzt.

Des Weiteren könnten die Features zur Unterscheidung der Objekte, welche im Moment die Farbwert-Histogramme sind, erweitert werden, sodass ähnlich aussehende Objekte besser auseinander gehalten werden können.

Auch im Kontext der Bewegungsmustererkennung gibt es einige Möglichkeiten für zukünftige Weiterentwicklungen. Ein offener Punkt ist die Realisierung der Echtzeitfähigkeit, die mit dem aktuellen Ansatz nicht möglich, aber für gewisse Anwendungen nützlich sein kann. Kontextinformationen könnten in diesem Zusammenhang genutzt werden, die eine Verkleinerung des Suchraums bewirken, indem nur noch die Trajektorien in den relevanten räumlichen und zeitlichen Bereichen und der relevanten Objekte durchsucht werden. Zudem könnte auf diese Weise ebenfalls eine Menge an zu erwartenden Mustern benutzt werden, um schon bei der Suche unrealistische Musterkandidaten auszuschließen, wobei hier natürlich die Gefahr besteht, die Ergebnismenge von vornherein

zu stark einzuschränken, sodass keine Möglichkeit von überraschenden Ergebnissen gegeben ist. Am Beispiel der Fußballanalyse bedeutet dies, dass u.a. Spieler-Rollen (z.B. Verteidiger, Angreifer oder linker bzw. rechter Mittelfeldspieler) als Vorabinformationen verwendet werden könnten, um den relevanten Bewegungsraum einzuschränken. Da sich bspw. ein Angreifer nur selten im eigenen Strafraum aufhält, ist die Wahrscheinlichkeit sehr gering, dort Bewegungsmuster zu finden. Weiterhin könnten auch die Trajektorien zusätzlicher Objekte, die eigentlich nicht beobachtet werden, in die Suche einfließen, da diese eventuell einen Einfluss auf die Bewegungen der beobachteten Objekte ausüben. Dieses könnte dadurch realisiert werden, dass die entsprechenden Positionsinformationen in die Vektoren, die in den Sequenzelementen von S_{tot} enthalten sind, integriert werden.

Ein weiterer offener Punkt ist die Anwendung der extrahierten Bewegungsmuster. In Han u. a. (2011) werden mit der Prädiktion und der Charakterisierung zwei typische Möglichkeiten genannt. Bei der Charakterisierung werden die Muster benutzt, um die beobachteten Objekte zu beschreiben, indem sie wertvolle Informationen über ihr typischen Bewegungsverhalten liefern. Dieses typische Verhalten kann für die Objekte unterschiedlich sein, sodass es sie in gewisser Weise charakterisiert. Dieses würde ein Wiedererkennen von Objekten rein auf Basis ihrer Trajektorien erlauben und könnte u.a. das Tracking unterstützen (siehe Abbildung 8.2). Dort stellt eben dieses Wiedererkennen ein Problem dar, wenn das Objekt die beobachtete Szene für eine gewisse Zeit verlässt und später wieder zurückkehrt.

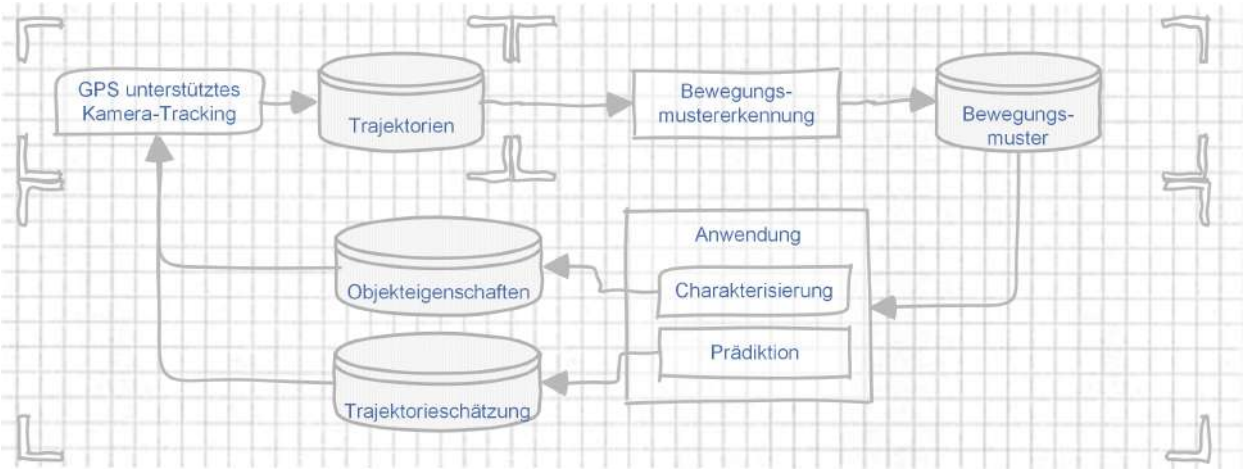
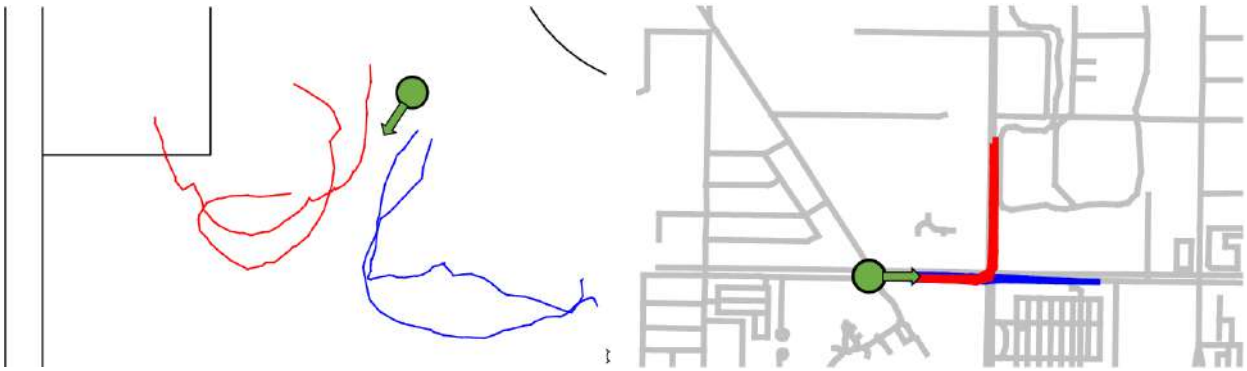


Abbildung 8.2: Zwei Möglichkeiten wie die erkannten Bewegungsmuster das Objekt-Tracking unterstützen können: Die Bewegungsprädiktion liefert Informationen über die zukünftigen Objekt-Trajektorien. Die Charakterisierung ermöglicht ein Wiedererkennen von Objekten.

Die Prädiktion liefert Informationen über die nächsten Bewegungen der Objekte. In Abbildung 8.3 werde zwei Beispiele (aus dem Fußballanalyse- und dem Verkehrskontext) gezeigt. Dort beschreiben unterschiedliche Muster (rot und blau) die Möglichkeiten, wie die zukünftige Objekt-Trajektorie aussehen könnte. Dabei wird vorausgesetzt, dass sich das Objekt (grün) nahe dem Anfang von mindestens einem Muster befindet. Um eine Vorhersage treffen zu können, muss das Bewegungsmuster bestimmt werden, dem das Objekt gerade folgt. Dieses Muster enthält Informationen, wie sich das Objekt in ähnlichen Situationen zuvor verhalten hat, welche dann zur Prädiktion via Extrapolation der aktuellen Trajektorie genutzt werden. Dementsprechend wird eine hohe Zahl an Mustern für jedes Objekt benötigt, die möglichst viele Situationen abdecken, sodass dort Vorhersagen getroffen werden können und diese zugleich möglichst präzise sind. Auch die Prädiktion kann auf diese Weise das Objekt-Tracking unterstützen, indem die vorberechneten Trajektorien Hinweise auf die Aufenthaltsorte der Objekte in den nächsten Kamerabildern geben. Im Kontext Verkehr können sogenannte nutzerbezogene Navigationssysteme, die Muster in ähnlicher Art und Weise nutzen, um automatisch das nächste Ziel des Anwenders vorherzusagen und entsprechend

dorthin zu navigieren, ohne dass vom Anwender eine Eingabe erfolgt. Dazu würden die bisherigen Routen bzw. Trajektorien (und möglicherweise zusätzliche Informationen) ausgewertet und so die gewöhnlichen Ziele oder Routen, bspw. der Weg zur Arbeit, nach Hause oder zu regelmäßigen Aktivitäten, ermittelt. Zur Bestimmung der möglichen Ziele müssten diese Muster mit der aktuellen Trajektorie abgeglichen werden.



(a) (b)

Abbildung 8.3: Bewegungsmuster können zur Vorhersage von zukünftigen Objektbewegungen genutzt werden. In beiden Beispielfällen, sowohl bei der Fußballanalyse (a) als auch bei der Analyse von Fahrzeug-Trajektorien (b), beschreiben zwei Muster (rot und blau) mögliche Trajektorien des beobachteten Objekts (grün).

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Agrawal, M., Konolige, K., 2006. Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS. In: *18th International Conference on Pattern Recognition (ICPR'06)*. Bd. 3. S. 1063–1068.

Agrawal, R., Srikant, R., 1994. Fast algorithms for mining association rules. In: *Proc. 20th int. conf. very large data bases, VLDB*. Bd. 1215. S. 487–499.

Ahmed, M., Karagiorgou, S., Pfoser, D., Wenk, C., 2015. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica* 19 (3), S. 601–632.

Aleythe, S., 2017. Never change a winning scene. *sueddeutsche.de*.

Anagnostopoulos, A., Vlachos, M., Hadjieleftheriou, M., Keogh, E., Yu, P. S., 2006. Global distance-based segmentation of trajectories. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, S. 34–43.

Andriluka, M., Roth, S., Schiele, B., 2010. Monocular 3D pose estimation and tracking by detection. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. S. 623–630.

Ardo, H., Astrom, K., Berthilsson, R., 2007. Real Time Viterbi Optimization of Hidden Markov Models for Multi Target Tracking. In: *IEEE Workshop on Motion and Video Computing, 2007. WMVC '07*. S. 2–2.

Baeza-Yates, R., Gonnet, G. H., 1992. A New Approach to Text Searching. *Commun. ACM* 35 (10), S. 74–82.

Barnich, O., Jodogne, S., Droogenbroeck, M. V., 2006. Robust Analysis of Silhouettes by Morphological Size Distributions. In: *Advanced Concepts for Intelligent Vision Systems*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, S. 734–745.

Barris, S., Button, C., 2008. A Review of Vision-Based Motion Analysis in Sport. *Sports Medicine* 38 (12), S. 1025–1043.

Bashir, F., Qu, W., Khokhar, A., Schonfeld, D., 2005. HMM-based motion recognition system using segmented PCA. In: *IEEE International Conference on Image Processing, 2005. ICIP 2005*. Bd. 3. S. III–1288–91.

Bauer, M., 2011. Vermessung und Ortung mit Satelliten: Globale Navigationssatellitensysteme (GNSS) und andere satellitengestützte Navigationssysteme. Wichmann Heidelberg.

Bellman, R., 1954. The theory of dynamic programming. Tech. rep., RAND CORP SANTA MONICA CA.

Bellman, R. E., Dreyfus, S. E., 2015. Applied dynamic programming. Princeton university press.

Bellotto, N., Hu, H., 2009. Multisensor-Based Human Detection and Tracking for Mobile Service Robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39 (1), S. 167–181.

Benfold, B., Reid, I., 2011. Stable multi-target tracking in real-time surveillance video. In: *CVPR 2011*. S. 3457–3464.

Benkert, M., Gudmundsson, J., Hübner, F., Wolle, T., 2008. Reporting flock patterns. *Computational Geometry* 41 (3), S. 111–125.

Berclaz, J., Fleuret, F., Fua, P., 2009. Multiple object tracking using flow linear programming. In: *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*. IEEE, S. 1–8.

Bishop, C. M., 2006. Pattern recognition and machine learning. Springer.

Bonnell, T. R., Henzi, S. P., Barrett, L., 2017. Direction matching for sparse movement data sets: determining interaction rules in social groups. *Behavioral Ecology* 28 (1), S. 193–203.

Brau, E., Guan, J., Simek, K., Pero, L. D., Dawson, C. R., Barnard, K., 2013. Bayesian 3D Tracking from Monocular Video. In: *2013 IEEE International Conference on Computer Vision*. S. 3368–3375.

Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., Van Gool, L., 2011. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE transactions on pattern analysis and machine intelligence* 33 (9), S. 1820–1833.

Buchin, K., Buchin, M., Van Kreveld, M., Luo, J., 2011. Finding long and similar parts of trajectories. *Computational Geometry* 44 (9), S. 465–476.

Buchin, M., Driemel, A., Kreveld, M. v., Sacristan, V., 2012. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *Journal of Spatial Information Science* 2011 (3), S. 33–63.

Cao, H., Mamoulis, N., Cheung, D. W., 2005. Mining frequent spatio-temporal sequential patterns. In: *Data Mining, Fifth IEEE International Conference on*. IEEE, S. 8–pp.

Cao, H., Mamoulis, N., Cheung, D. W., 2007. Discovery of Periodic Patterns in Spatiotemporal Sequences. *IEEE Transactions on Knowledge and Data Engineering* 19 (4), S. 453–467.

Cao, Z., Simon, T., Wei, S.-E., Sheikh, Y., 2016. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *arXiv:1611.08050 [cs]*ArXiv: 1611.08050.

Chen, J., Wang, R., Liu, L., Song, J., 2011. Clustering of trajectories based on Hausdorff distance. In: *2011 International Conference on Electronics, Communications and Control (ICECC)*. S. 1940–1944.

Cho, H., Seo, Y. W., Kumar, B. V. K. V., Rajkumar, R. R., 2014. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. S. 1836–1843.

Coutts, A. J., Duffield, R., 2010. Validity and reliability of GPS devices for measuring movement demands of team sports. *Journal of Science and Medicine in Sport* 13 (1), S. 133–135.

Crassidis, J. L., 2006. Sigma-point Kalman filtering for integrated GPS and inertial navigation. *IEEE Transactions on Aerospace and Electronic Systems* 42 (2), S. 750–756.

Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Bd. 1. S. 886–893 vol. 1.

de Vries, G., van Someren, M., 2010. Clustering Vessel Trajectories with Alignment Kernels Under Trajectory Compression. In: *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part I*. ECML PKDD'10. Springer-Verlag, Berlin, Heidelberg, S. 296–311.

Dean, T., Kanazawa, K., 1989. A model for reasoning about persistence and causation. *Computational intelligence* 5 (2), S. 142–150.

Dodge, S., Laube, P., Weibel, R., 2012. Movement similarity assessment using symbolic representation of trajectories. *International Journal of Geographical Information Science* 26 (9), S. 1563–1588.

Dodge, S., Weibel, R., Lautenschütz, A.-K., 2008. Towards a Taxonomy of Movement Patterns. *Information Visualization* 7 (3-4), S. 240–252.

Duckham, M., 2012. Decentralized spatial computing: foundations of geosensor networks. Springer.

Dugad, R., Desai, U. B., 1996. A tutorial on hidden Markov models. *Signal Processing and Artificial Neural Networks Laboratory Department of Electrical Engineering Indian Institute of Technology*.

Ess, A., Leibe, B., Schindler, K., Van Gool, L., 2008. A mobile vision system for robust multi-person tracking. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, S. 1–8.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Kdd*. Bd. 96. S. 226–231.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., Ramanan, D., 2010. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (9), S. 1627–1645.

Feuerhake, U., 2012. Prediction of Individual's Movement Based on Interesting Places. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* I-2, S. 31–36.

Feuerhake, U., 2016. Recognition of Repetitive Movement Patterns—The Case of Football Analysis. *ISPRS International Journal of Geo-Information* 5 (11), S. 208.

Feuerhake, U., Brenner, C., Sester, M., 2015. GPS-Aided Video Tracking. *ISPRS International Journal of Geo-Information* 4 (3), S. 1317–1335.

Feuerhake, U., Kuntzsch, C., Sester, M., 2011. Finding interesting places and characteristic patterns in spatio-temporal trajectories. In: *8th LBS symposium*.

Feuerhake, U., Sester, M., 2013. Mining group movement patterns. In: *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, S. 510–513.

Fleuret, F., Berclaz, J., Lengagne, R., Fua, P., 2008. Multicamera People Tracking with a Probabilistic Occupancy Map. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30 (2), S. 267–282.

Forney, G.D., J., 1973. The viterbi algorithm. *Proceedings of the IEEE* 61 (3), S. 268–278.

Ghahramani, Z., 2001. An introduction to hidden Markov models and Bayesian networks. *International journal of pattern recognition and artificial intelligence* 15 (01), S. 9–42.

Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D., 2007. Trajectory Pattern Mining. In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '07. ACM, New York, NY, USA, S. 330–339.

Goh, S. S., Ron, A., Shen, Z., 2007. Mathematics and Computation in Imaging Science and Information Processing. World Scientific, google-Books-ID: KOqmm4fihn0C.

Gray, A. J., Jenkins, D., Andrews, M. H., Taaffe, D. R., Glover, M. L., 2010. Validity and reliability of GPS for measuring distance travelled in field-based team sports. *Journal of Sports Sciences* 28 (12), S. 1319–1325.

Grewal, M. S., Weill, L. R., Andrews, A. P., 2007. Global Positioning Systems, Inertial Navigation, and Integration. In: *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons.

Grunz, A., Memmert, D., Perl, J., 2012. Tactical pattern recognition in soccer games by means of special self-organizing maps. *Human Movement Science* 31 (2), S. 334–343.

Gudmundsson, J., Kreveld, M. v., Speckmann, B., 2007. Efficient Detection of Patterns in 2D Trajectories of Moving Points. *GeoInformatica* 11 (2), S. 195–215.

Gudmundsson, J., Wolle, T., 2014. Football analysis using spatio-temporal tools. *Computers, Environment and Urban Systems* 47, S. 16–27.

Haaren, J. V., Dzyuba, V., Hannosset, S., Davis, J., 2015. Automatically Discovering Offensive Patterns in Soccer Match Data. In: *Advances in Intelligent Data Analysis XIV*. Lecture Notes in Computer Science. Springer, Cham, S. 286–297.

Han, J., Kamber, M., Pei, J., 2011. Data mining: concepts and techniques. Elsevier.

Helbing, D., Molnar, P., 1995. Social force model for pedestrian dynamics. *Physical review E* 51 (5), S. 4282.

Henschel, R., Leal-Taixé, L., Rosenhahn, B., 2015. Solving Multiple People Tracking in a Minimum Cost Arborescence. In: *2015 IEEE Winter Applications and Computer Vision Workshops*. S. 71–72.

Henschel, R., Leal-Taixé, L., Rosenhahn, B., Schindler, K., 2016. Tracking with multi-level features. *arXiv:1607.07304 [cs]*ArXiv: 1607.07304.

Hinz, S., Schmidt, F., 2017. Personentracking in Luftbildsequenzen. In: *Photogrammetrie und Fernerkundung*. Springer Reference Naturwissenschaften. Springer Spektrum, Berlin, Heidelberg, S. 685–732.

Hirano, S., Tsumoto, S., 2004. Finding interesting pass patterns from soccer game records. In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, S. 209–218.

Hoiem, D., Efros, A. A., Hebert, M., 2008. Putting Objects in Perspective. *International Journal of Computer Vision* 80 (1), S. 3–15.

Huang, H., Zhang, L., Sester, M., 2014. A recursive Bayesian filter for anomalous behavior detection in trajectory data. In: *Connecting a Digital Europe Through Location and Place*. Springer, S. 91–104.

Hung, C.-C., Peng, W.-C., Lee, W.-C., 2015. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. *The VLDB Journal* 24 (2), S. 169–192.

Hyun, D., Yang, H. S., Yuk, G. H., Park, H. S., 2009. A dead reckoning sensor system and a tracking algorithm for mobile robots. In: *2009 IEEE International Conference on Mechatronics*. S. 1–6.

Iwase, S., Saito, H., 2004. Parallel tracking of all soccer players by integrating detected positions in multiple view images. In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*. Bd. 4. S. 751–754 Vol.4.

Kaufman, L., Rousseeuw, P. J., 2009. Finding groups in data: an introduction to cluster analysis. Bd. 344. John Wiley & Sons.

Kim, C., Li, F., Ciptadi, A., Rehg, J. M., 2015. Multiple hypothesis tracking revisited. In: *Proceedings of the IEEE International Conference on Computer Vision*. S. 4696–4704.

Kim, H.-C., Kwon, O., Li, K.-J., 2011. Spatial and Spatiotemporal Analysis of Soccer. In: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '11. ACM, New York, NY, USA, S. 385–388.

Klinger, T., Rottensteiner, F., Heipke, C., 2017. Probabilistic multi-person localisation and tracking in image sequences. *ISPRS Journal of Photogrammetry and Remote Sensing* 127, S. 73–88.

Kuhn, H., 2012. The Hungarian Method for the Assignment Problem. *Naval Research Logistic Quarterly* 2.

Laube, P., Duckham, M., Wolle, T., 2008. Decentralized movement pattern detection amongst mobile geosensor nodes. In: *International Conference on Geographic Information Science*. Springer, S. 199–216.

Laube, P., Kreveld, M. v., Imfeld, S., 2005. Finding REMO - Detecting Relative Motion Patterns in Geospatial Lifelines. In: *Developments in Spatial Data Handling*. Springer Berlin Heidelberg, S. 201–215.

Leal-Taixé, L., Canton-Ferrer, C., Schindler, K., 2016. Learning by tracking: Siamese CNN for robust target association. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. S. 33–40.

Leal-Taixé, L., Pons-Moll, G., Rosenhahn, B., 2011. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. S. 120–127.

Lee, J. G., Han, J., Li, X., 2015. A unifying framework of mining trajectory patterns of various temporal tightness. *IEEE Transactions on Knowledge and Data Engineering* 27 (6), S. 1478–1490.

Lee, J.-G., Han, J., Li, X., Gonzalez, H., 2008. TraClass: Trajectory Classification Using Hierarchical Region-based and Trajectory-based Clustering. *Proc. VLDB Endow.* 1 (1), S. 1081–1094.

Leibe, B., Schindler, K., Cornelis, N., Gool, L. V., 2008. Coupled Object Detection and Tracking from Static Cameras and Moving Vehicles. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30 (10), S. 1683–1698.

Li, R., Chellappa, R., 2010. Group motion segmentation using a spatio-temporal driving force model. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, S. 2038–2045.

Li, Z., Ding, B., Han, J., Kays, R., 2010a. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment* 3 (1-2), S. 723–734.

Li, Z., Ding, B., Han, J., Kays, R., Nye, P., 2010b. Mining Periodic Behaviors for Moving Objects. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '10. ACM, New York, NY, USA, S. 1099–1108.

Liu, S., Wang, S., Jayarajah, K., Misra, A., Krishnan, R., 2013. TODMIS: mining communities from trajectories. In: *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. CIKM '13. ACM, New York, NY, USA, S. 2109–2118.

Luber, M., Stork, J. A., Tipaldi, G. D., Arras, K. O., 2010. People tracking with human motion predictions from social forces. In: *2010 IEEE International Conference on Robotics and Automation*. S. 464–469.

Luhmann, T., 2000. Nahbereichsphotogrammetrie. Wichmann Verlag.

Ma, C., Huang, J.-B., Yang, X., Yang, M.-H., 2015. Hierarchical convolutional features for visual tracking. In: *Proceedings of the IEEE International Conference on Computer Vision*. S. 3074–3082.

MacEachren, A. M., 1995. How maps work: representation, visualization, and design. Guilford Press.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Bd. 1. Oakland, CA, USA., S. 281–297.

Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., Cheung, D. W., 2004. Mining, indexing, and querying historical spatiotemporal data. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, S. 236–245.

Martinerie, F., 1997. Data fusion and tracking using HMMs in a distributed sensor network. *IEEE Transactions on Aerospace and Electronic Systems* 33 (1), S. 11–28.

Milan, A., Rezatofighi, S. H., Dick, A. R., Reid, I. D., Schindler, K., 2017. Online Multi-Target Tracking Using Recurrent Neural Networks. In: *Association for the Advancement of Artificial Intelligence*. S. 4225–4232.

Monreale, A., Pinelli, F., Trasarti, R., Giannotti, F., 2009. Wherenext: a location predictor on trajectory pattern mining. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, S. 637–646.

Morris, B., Trivedi, M., 2009. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, S. 312–319.

Murphy, K. P., 2002. Dynamic Bayesian Networks: Representation, Inference and Learning. University of California, Berkeley.

Murphy, K. P., 2012. Machine Learning: A Probabilistic Perspective. MIT Press.

Mutschler, C., Kókai, G., Edelhäußer, T., 2011. Online data stream mining on interactive trajectories in soccer games. In: *2nd International Conference on Positioning and Context-Awareness: PoCA 2011. Proceedings*. S. 15–22.

Mutschler, C., Ziekow, H., Jerzak, Z., 2013. The DEBS 2013 Grand Challenge. In: *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*. DEBS '13. ACM, New York, NY, USA, S. 289–294.

Nanni, M., Pedreschi, D., 2006. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems* 27 (3), S. 267–289.

Niu, Z., Gao, X., Tian, Q., 2012. Tactic analysis based on real-world ball trajectory in soccer video. *Pattern Recognition* 45 (5), S. 1937–1947.

Okuma, K., Taleghani, A., Freitas, N. d., Little, J. J., Lowe, D. G., 2004. A Boosted Particle Filter: Multitarget Detection and Tracking. In: *Computer Vision - ECCV 2004*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, S. 28–39.

OpenCV team, t., 2017. CUDA - OpenCV library.

Pelekis, N., Kopanakis, I., Kotsifakos, E. E., Frentzos, E., Theodoridis, Y., 2009. Clustering trajectories of moving objects in an uncertain world. In: *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. IEEE, S. 417–427.

Penders, P., 2014. 15:3 für Torlinien-Technik: Nie mehr Phantomtore in der Bundesliga! *FAZ.NET*.

Perera, A. A., Srinivas, C., Hoogs, A., Brooksby, G., Hu, W., 2006. Multi-object tracking through simultaneous long occlusions and split-merge conditions. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Bd. 1. IEEE, S. 666–673.

Piérard, S., Lejeune, A., Droogenbroeck, M. V., 2011. A probabilistic pixel-based approach to detect humans in video streams. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. S. 921–924.

Pirsiavash, H., Ramanan, D., Fowlkes, C. C., 2011. Globally-optimal greedy algorithms for tracking a variable number of objects. In: *CVPR 2011*. S. 1201–1208.

Quinlan, J. R., 1986. Induction of decision trees. *Machine Learning* 1 (1), S. 81–106.

Quinlan, J. R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rabiner, L., Juang, B., 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine* 3 (1), S. 4–16.

Rao, B., Durrant-Whyte, H., Sheen, J., 1993. A Fully Decentralized Multi-Sensor System For Tracking and Surveillance. *The International Journal of Robotics Research* 12 (1), S. 20–44.

Reid, D., 1979. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control* 24 (6), S. 843–854.

Saffari, A., Leistner, C., Santner, J., Godec, M., Bischof, H., 2009. On-line Random Forests. In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. S. 1393–1400.

Schindler, K., Ess, A., Leibe, B., Van Gool, L., 2010. Automatic detection and tracking of pedestrians from a moving stereo rig. *ISPRS Journal of Photogrammetry and Remote Sensing - ISPRS J PHOTOGRAMM* 65, S. 523–537.

Shu, G., Dehghan, A., Oreifej, O., Hand, E., Shah, M., 2012. Part-based multiple-person tracking with partial occlusion handling. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. S. 1815–1821.

Smith, D., Singh, S., 2006. Approaches to Multisensor Data Fusion in Target Tracking: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 18 (12), S. 1696–1710.

Solera, F., Calderara, S., Cucchiara, R., 2015. Learning to Divide and Conquer for Online Multi-Target Tracking.

Spors, S., Rabenstein, R., Strobel, N., 2001. A Multi-Sensor Object Localization System. In: *Vision Modeling and Visualization*. Bd. 1. S. 19–26.

Suhr, J. K., Jang, J., Min, D., Jung, H. G., 2017. Sensor Fusion-Based Low-Cost Vehicle Localization System for Complex Urban Environments. *IEEE Transactions on Intelligent Transportation Systems* 18 (5), S. 1078–1086.

Tran, Q., Vo, B., Dinh, T., Duong, D., 2011. Automatic Player Detection, Tracking and Mapping to Field Model for Broadcast Soccer Videos. In: *Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia*. MoMM '11. ACM, New York, NY, USA, S. 240–243.

Tsai, H.-P., Yang, D.-N., Chen, M.-S., 2011. Mining Group Movement Patterns for Tracking Moving Objects Efficiently. *IEEE Trans. on Knowl. and Data Eng.* 23 (2), S. 266–281.

Vafa, K., 2014. Trajectory Clustering Using a Variation of Fréchet Distance. Dissertation, Université d'Ottawa/University of Ottawa.

Varley, M. C., Fairweather, I. H., Aughey1, 2, R. J., 2012. Validity and reliability of GPS for measuring instantaneous velocity during acceleration, deceleration, and constant motion. *Journal of Sports Sciences* 30 (2), S. 121–127.

Viola, P., Jones, M., 2001a. Rapid object detection using a boosted cascade of simple features. In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Bd. 1. IEEE, S. I–I.

Viola, P., Jones, M., 2001b. Robust Real-time Object Detection. In: *International Journal of Computer Vision*.

Wang, B., Wang, G., Chan, K. L., Wang, L., 2017. Tracklet Association by Online Target-Specific Metric Learning and Coherent Dynamics Estimation. *IEEE Transactions on pattern analysis and machine intelligence* 39 (3), S. 589–602.

Wang, B., Wang, L., Shuai, B., Zuo, Z., Liu, T., Chan, K. L., Wang, G., 2016. Joint Learning of Siamese CNNs and Temporally Constrained Metrics for Tracklet Association. *arXiv preprint arXiv:1605.04502*.

Wang, L., Ouyang, W., Wang, X., Lu, H., 2015. Visual tracking with fully convolutional networks. In: *Proceedings of the IEEE International Conference on Computer Vision*. S. 3119–3127.

Wang, S., Fowlkes, C. C., 2015. Learning Optimal Parameters For Multi-target Tracking. In: *BMVC*. Bd. 1. S. 6.

Wang, Y., Lim, E.-P., Hwang, S.-Y., 2006. Efficient mining of group patterns from user movement data. *Data & Knowledge Engineering* 57 (3), S. 240–282.

Wendel, J., 2011. Integrierte Navigationssysteme: Sensordatenfusion, GPS und Inertiale Navigation. Walter de Gruyter.

Witten, I. H., Frank, E., Hall, M. A., Pal, C. J., 2016. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

Wu, B., Nevatia, R., 2007. Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors. *International Journal of Computer Vision* 75 (2), S. 247–266.

Xiang, Y., Alahi, A., Savarese, S., 2015. Learning to Track: Online Multi-object Tracking by Decision Making. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. S. 4705–4713.

Xie, X., Evans, R., 1990. Multiple target tracking using hidden Markov models. In: *Radar Conference, 1990., Record of the IEEE 1990 International*. S. 625–628.

Yang, B., Nevatia, R., 2014. Multi-target tracking by online learning a CRF model of appearance and motion patterns. *International Journal of Computer Vision* 107 (2), S. 203–217.

Yang, D. B., Gonzalez-Banos, H. H., Guibas, L. J., 2003. Counting people in crowds with a real-time network of simple image sensors. In: *Proceedings Ninth IEEE International Conference on Computer Vision*. S. 122–129 vol.1.

Yang, M., Jia, Y., 2016. Temporal Dynamic Appearance Modeling for Online Multi-person Tracking. *Comput. Vis. Image Underst.* 153 (C), S. 16–28.

Yilmaz, A., Javed, O., Shah, M., 2006. Object Tracking: A Survey. *ACM Comput. Surv.* 38 (4).

Yin, J., Tian, G., Li, G., 2014. Object localization and tracking based on multiple sensor fusion in intelligent home. In: *The 26th Chinese Control and Decision Conference (2014 CCDC)*. S. 5266–5270.

Yoon, J. H., Yang, M. H., Lim, J., Yoon, K. J., 2015. Bayesian Multi-object Tracking Using Motion Context from Multiple Objects. In: *2015 IEEE Winter Conference on Applications of Computer Vision*. S. 33–40.

Yuan, G., Sun, P., Zhao, J., Li, D., Wang, C., 2017. A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review* 47 (1), S. 123–144.

Zen, H., Tokuda, K., Kitamura, T., 2004. A Viterbi algorithm for a trajectory model derived from HMM with explicit relationship between static and dynamic features. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04)*. Bd. 1. S. I–837–40 vol.1.

Zhang, L., Li, Y., Nevatia, R., 2008. Global data association for multi-object tracking using network flows. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, S. 1–8.

Zhang, T., Liu, S., Ahuja, N., Yang, M.-H., Ghanem, B., 2015. Robust visual tracking via consistent low-rank sparse learning. *International Journal of Computer Vision* 111 (2), S. 171–190.

Zhang, Z., 2000. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (11), S. 1330–1334.

Zhao, T., Nevatia, R., 2004. Tracking multiple humans in crowded environment. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Bd. 2. S. II–406–II–413 Vol.2.

Zhou, H., Kimber, D., 2006. Unusual Event Detection via Multi-camera Video Mining. In: *18th International Conference on Pattern Recognition (ICPR'06)*. Bd. 3. S. 1161–1166.

Zhu, G., Huang, Q., Xu, C., Rui, Y., Jiang, S., Gao, W., Yao, H., 2007. Trajectory based event tactics analysis in broadcast sports video. In: *Proceedings of the 15th ACM international conference on Multimedia*. ACM, S. 58–67.

Zivkovic, Z., 2004. Improved adaptive Gaussian mixture model for background subtraction. In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*. Bd. 2. S. 28–31 Vol.2.

Lebenslauf

Persönliches

Name	Udo Feuerhake
Geburtsdatum	07.09.1982 in Hameln

Ausbildung

2002	Schiller-Gymnasium, Hameln *Abitur*
2003 - 2006	BHW / Postbank, Hameln *Ausbildung zum Fachinformatiker* Siemens Professional Education, Paderborn *Fachberater für Softwaretechniken*
2006 - 2010	Leibniz Universität Hannover *M.Sc. Computergestützte Ingenieurwissenschaften*

Beruf

2002 - 2003	Wehrdienst
seit 2011	Institut für Kartographie und Geoinformatik, Leibniz Universität Hannover *Wissenschaftlicher Mitarbeiter*

Danksagung

Diese letzten Zeilen möchte ich nicht dem Thema dieser Arbeit, sondern den Menschen widmen, die sie auf die eine oder andere Weise ermöglicht haben.

An erster Stelle möchte ich daher Frau Prof. Dr.-Ing. habil. Monika Sester für die Betreuung meiner Arbeit, für die vielen Anregungen und Ideen bei der Umsetzung sowie dafür, dass sie mir die schöne Zeit am ikg ermöglicht hat, danken.

In diesem Sinn bedanke ich mich auch bei meinen Kollegen, die diese Zeit so schön gemacht haben.

Herrn Prof. Dr.-Ing. Christian Heipke und Herrn Prof. Dr. Robert Weibel danke ich für die freundliche Übernahme des Korreferats.

Meiner Fußballmannschaft vom TSV Brünnighausen danke ich dafür, dass sie sich für die Erhebung der verwendeten Daten in den vielen Spielen und Trainingseinheiten zur Verfügung gestellt hat.

Meinen Schwiegereltern danke ich für die äußerst hilfreiche Unterstützung in vielen Belangen, besonders aber für die Betreuung unserer kleinen Sophia.

Meiner Familie und ganz besonders meinen Eltern, Heiner und Renate Feuerhake, bin ich unendlich dankbar, dass sie immer für mich da waren und da sind. Ihre bedingungslose Unterstützung hat mir alles überhaupt erst ermöglicht.

Zu guter Letzt danke ich meiner Frau Janina für den Rückhalt, für die Liebe, die sie mir entgegenbringt, und für unsere kleine Sophia, die uns immer wieder durch Kleinigkeiten auf den „Boden der Tatsachen“ zurückholt. Ich liebe Euch!