Institut für Visualisierung und Interaktive Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3269

Skizzenbasierte Trajektoriensuche in Videos

Johannes Engelhardt

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Daniel Weiskopf
Betreuer:	Dipl.-Inf. Markus Höferlin, Dipl.-Inf. Benjamin Höferlin
begonnen am:	22. November 2011
beendet am:	22. Mai 2012
CR-Klassifikation:	H.3.3, H.5.2, I.3.3

Kurzfassung

Die visuelle Analyse von Videodaten aus Überwachungsvideos ist eine mühsame und zeitaufwändige Aufgabe. Eine Möglichkeit zur Beschleunigung des Analysevorgangs ist die Extraktion von Trajektorien. In dieser Arbeit werden zwei Methoden zur Suche nach Trajektorien vorgestellt. Diese ermöglichen das skizzenbasierte sowie zeitabhängige Durchsuchen von Trajektorienmengen in Form von Filtern, die in ein bestehendes Visual-Analytics-System integriert sind. Für die skizzenbasierte Suche wurde eine Oberfläche zur Modellierung der Trajektorienskizze entwickelt. Für die vorgestellten Filter werden detaillierte Konfigurationsmöglichkeiten zur Verfügung gestellt. Dabei wird auch das Formulieren unscharfer Anfragen unterstützt.

Abstract

Visual analysis of surveillance video data is a tedious and time-consuming task. One possibility to enhance the analysis process is the extraction of trajectories. This paper introduces two methods for trajectory search. These methods offer sketch-based and time-relative querying of trajectory databases, which are implemented as filters and integrated in an existing visual analytics system. A user interface for the modeling of trajectory sketches has been developed. Detailed configuration possibilities are proposed for the presented filters. Additionally, the filters support the formulation of fuzzy querys.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die Analyse von Überwachungskameraaufnahmen ist mühsam und zeitaufwändig. Überwachungskameras produzieren ständig Unmengen an Videodaten. Die Menge der weltweit aufgezeichneten Videos wächst zudem immer rasanter. Die relevanten Informationen haben in Überwachungsvideos im Vergleich zu der gesamten Datenmenge in der Regel einen sehr geringen Anteil. Daher wird bei der manuellen Auswertung durch Analysten ein Großteil der Zeit vergeudet. Gleichzeitig liefern automatische Analysesysteme allerdings noch immer unbefriedigende Ergebnisse [Ble10].

Eine Möglichkeit, die manuelle Auswertung von Überwachungsvideos zu beschleunigen und zu vereinfachen, ist die Extraktion der Bewegungen von Objekten im Video. Diese Bewegungen und ihre Eigenschaften werden in Trajektorien beschrieben. Beispiele der Eigenschaften von Trajektorien sind Position, Geschwindigkeit, Richtung oder Objektklasse (Fußgänger, Radfahrer, Fahrzeug, etc.). Diese extrahierten Trajektorien können leicht nach Eigenschaften gruppiert ("Clustering") oder durchsucht ("Querying") werden. So kann der Benutzer eines solchen Extraktionssystems schnell potentiell nützliche Informationen im Video finden, ohne sich das ganze Video anschauen zu müssen.

Interactive Schematic Summaries

Interactive Schematic Summaries (ISS) [HHWH11a] ist ein System, das die visuelle Analyse von Überwachungsvideos durch Bereitstellen von extrahierten Trajektorien unterstützt. Es bietet die Möglichkeit, die Trajektorien im Video durch hierarchisches Explorieren manuell zu durchsuchen. Dabei werden die im Video enthaltenen Trajektorien entsprechend ihrer Ähnlichkeit anhand bestimmter Kriterien in verschiedene skizzenähnliche Cluster zusammengefasst. Aus den visuell dargestellten Clustern kann der Benutzer eines oder mehrere auswählen, deren Trajektorien wieder nach ihrer Ähnlichkeit in verschiedene Cluster zusammengefasst werden. So kann der Benutzer iterativ und explorativ durch die Trajektorien im Video navigieren, bis hinunter zu einzelnen Trajektorien. Von den so vom Benutzer

gefundenen Trajektorien kann dann der entsprechende Videoabschnitt angefordert werden. Die Kriterien lassen sich dabei in jedem Schritt verändern, um so die explorative Suche individuell anzupassen. *ISS* hilft so dem Benutzer, durch die schematische Darstellung schnell einen groben Überblick über das Geschehen im Video zu bekommen. Interessante Stellen sind schnell und explorativ zu finden, ohne dass durch das Anschauen irrelevanter Videoabschnitte Zeit verloren geht.

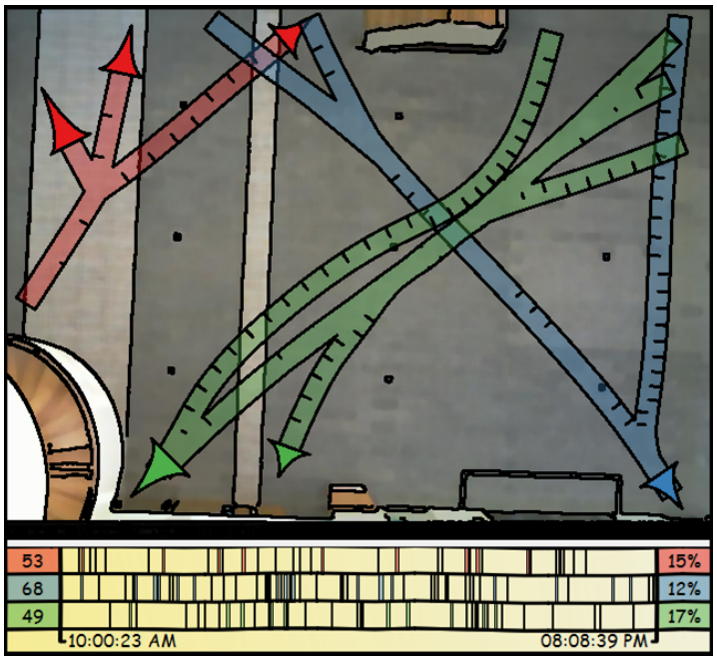

Abbildung 1.1: *ISS*: Trajektorien werden in verschiedene Cluster zusammengefasst und farbig dargestellt [HHWH11a].

Für das Erstellen der Cluster werden alle vorhandenen Trajektorien anhand der gegebenen Ähnlichkeitskriterien miteinander verglichen. Mit zunehmender Anzahl der Trajektorien steigt der Berechnungsaufwand somit quadratisch an, was sich vor allem bei rechenaufwändige Vergleichsalgorithmen negativ auf die Berechnungsgeschwindigkeit auswirkt. *ISS* löst dieses Problem durch Vorausberechnen der Ähnlichkeiten der Trajektorien, welches die Analyse an zwei Bedingungen knüpft: Erstens können nur Trajektorien berücksichtigt werden, die zum Zeitpunkt der Vorausberechnung der Ähnlichkeiten bekannt sind. Zweitens kann die Analyse erst dann erfolgen, wenn die Vorausberechnung abgeschlossen ist, was bei vielen Trajektorien unter Umständen sehr lange dauern kann. *ISS* ist also erst dann einsetzbar, wenn die Daten des ganzen Videos analysiert wurden.

Skizzenbasierte Trajektoriensuche

ISS ermöglicht ein exploratives Erkunden der Videoinhalte. Wie die Trajektorien in Cluster zusammengefasst werden, ist von Benutzer aber schwer kontrollierbar. So eignet sich das

Konzept eher dazu, eine Gesamtübersicht über die extrahierten Inhalte zu bekommen, jedoch weniger dazu, gezielte Suchanfragen zu beantworten. Eine gezielte Suche nach bestimmten Trajektorien oder deren Eigenschaften gestaltet sich somit schwierig. Für diesen Anwendungsfall muss die Möglichkeit zur Exploration um eine Suchfunktion erweitert werden. Für das Durchsuchen der Trajektorien von Videos wurden verschiedene Methoden entwickelt. Eine verbreitete Methode ist die skizzenbasierte Trajektoriensuche, bei der Suchanfragen nach Trajektorien mittels einer Skizze modelliert werden. Diese Methode findet z.B. bei Chang et al. [CCM+98], Hsieh et al. [HYC06] und Yajima et al. [YNT02] Verwendung.

Trajektorien enthalten sowohl zeitliche als auch positionale Informationen. Soll nach bestimmten Trajektorien gesucht werden, muss eine Anfrage erstellt werden, die die gewünschten zeitlichen als auch räumlichen Eigenschaften der Trajektorien enthält. Anfragen an Datenbestände werden i.A. in Textform formuliert. Textbasierte Methoden, wie z.B. die Verwendung von Schlüsselwörtern (z.B. "Ein Auto fährt in diese Richtung") sind für Trajektorienqueries aber meist nicht geeignet, da sich die benötigten Informationen oft schwer oder nur sehr aufwändig in Textform ausdrücken lassen [HXF+07]. Das skizzenbasierte Erstellen einer Query-Trajektorie löst das Problem, in dem der Benutzer eine schematische Zeichnung der gesuchten Trajektorie vorgeben kann. Die temporalen und positionalen Eigenschaften der Trajektorie werden dabei visuell und intuitiv durch eine Skizze modelliert. Durch die visuelle Darstellung ist eine präzise Modellierung der gesuchten Informationen möglich und kann u.U. besser in den Kontext, in dem gesucht werden soll, eingeordnet werden.

Einsatz im Visual Analytics Center

Das *Visual Analytics Center (VAC)* ist ein Framework für die Extraktion von Trajektorien aus Videos und stellt die Rahmenanwendung, in die auch *ISS* integriert ist. Das *VAC* unterstützt sowohl die Analyse von Videodaten in Echtzeit (währen des Abspielens), als auch die statische Analyse, bei der zunächst alle Daten aus dem Video in einen Cache extrahiert werden und dann komplett zur Verfügung stehen. Durch Filter kann die extrahierte Trajektorienmenge nach bestimmten Kriterien reduziert werden. Jeder Filter kann entscheiden, ob eine Trajektorie einem bestimmten Kriterium entspricht oder nicht und diese entsprechend bewerten. Diese Filter sind sowohl in Echtzeit als auch statisch einsetzbar und lassen sich für die Trajektoriensuche nutzen. Die Browsingschritte durch die Cluster in *ISS* stellen ebenfalls Filter dar. Somit lässt sich die Suche nach Trajektorien als Filter in das *VAC* integrieren und somit auch in den *ISS* nutzen. In Kapitel 3 auf Seite 25 wird das *VAC* näher beschrieben.

1.2 Aufgabenstellung

In dieser Arbeit soll das *VAC* um eine Möglichkeit der skizzenbasierten Trajektoriensuche erweitert werden. Dies soll über die Integration von Filtern geschehen, die sowohl *ISS* als auch anderen verfügbaren Anwendungen innerhalb des *VAC* zur Verfügung stehen. Ein Filter hat die Aufgabe, Trajektorien anhand definierter Eigenschaften zu bewerten. Die

Eigenschaften sollen dabei mittels einer Skizze modelliert werden. Zusätzlich sollen die Suche nach zeitlichen Aspekten ermöglicht werden, wie z.B. "zwei Trajektorien treffen sich zur gleichen Zeit" oder "nachdem sich eine Trajektorie bewegt, bewegt sich eine andere". Die entwickelten Filter sollen in den Filterbereich des *VAC* eingefügt werden können und über ihre grafische Repräsentation zu einer Übersicht über die einzelnen Filterschritte beitragen. Zudem sollen die Parameter jedes Filters über eine Benutzeroberfläche angezeigt und konfiguriert werden können.

Die Filter sollen für das Erstellen unscharfer Anfragen ("Fuzzy Queries") geeignet sein. Demnach soll ein Filter nicht nur entscheiden, ob ein bestimmtes Kriterium erreicht wird, sondern auch zu welchem Grad. Für diesen Zweck wird das Konzept der Fuzzy-Logik (siehe Abschnitt 2.5 auf Seite 23) eingesetzt. Ein spezieller Container kümmert sich um die Umwandlung der unscharfen Ergebnisse dieser Fuzzy-Filter in scharfe, boolesche Werte (siehe Abschnitt 3.5.2 auf Seite 32).

Für die Lösung dieser Aufgabe wurden im Rahmen dieser Arbeit zwei solcher Fuzzy-Filter entwickelt. Der erste Filter bietet eine Benutzeroberfläche für das interaktive Erstellen einer Trajektorienskizze und ermöglicht so die Trajektoriensuche anhand der modellierten Eigenschaften. Ergebnis des Fuzzy-Filters ist somit die Ähnlichkeit der verglichenen Trajektorien bezüglich ihrer Eigenschaften. Dieser Filter wird in Kapitel 5 auf Seite 47 vorgestellt. Der zweite Filter vergleicht Trajektorien in ihrem zeitlichen Zusammenhang, genauer ihre Eigenschaften zu einem gemeinsamen Zeitpunkt. So lassen sich Trajektorien finden, die zur selben Zeit gleiche oder ähnliche Eigenschaften aufweisen, etwa den gleichen oder ähnlichen Ort. Ihre Ähnlichkeit zum selben Zeitpunkt ist das Ergebnis dieses Filters. Kapitel 6 auf Seite 59 befasst sich mit diesem zeitabhängigen Filter.

1.3 Gliederung der Arbeit

Die Arbeit ist acht Kapitel gegliedert, deren Inhalt im Folgenden kurz vorgestellt wird:

Kapitel 1 - Einleitung Dieses Kapitel dient zur Einführung in die Thematik und erläutert die Motivation und Aufgabenstellung der Arbeit.

Kapitel 2 - Grundlagen Hier werden einige Grundlagen vorgestellt, die für das Verständnis für die Thematik der Arbeit von Bedeutung sind, bzw. als Basis für die vorgestellten Konzepte dienen.

Kapitel 3 - Das Visual Analytics Center In diesem Kapitel wird das *VAC* vorgestellt, in welches die skizzenbasierte Trajektoriensuche eingebettet ist. Es werden die Funktionsweise des Systems sowie die Bedeutung und Einbettung von Filtern und Views erläutert.

Kapitel 4 - Skizzieren einer Query-Trajektorie Dieses Kapitel befasst sich mit dem Skizzieren einer Query-Trajektorie, die zum Filtern anderer Trajektorien verwendet wird. Die

Oberfläche zum Erstellen der Skizze wird vorgestellt und die Vorgänge und Möglichkeiten bei der Skizzierung näher erläutert. Zudem wird erklärt, wie die erstellte Trajektorienskizze im System verwendet werden kann.

Kapitel 5 - Filtern anhand einer skizzierten Trajektorie In diesem Kapitel wird der erste Filter vorgestellt: ein Trajektorienfilter, der auf Grundlage einer skizzierten Query-Trajektorie arbeitet. Dabei wird er Vergleich von Trajektorien, die Konfigurationsmöglichkeiten des Filters sowie seine Arbeitsweise näher erläutert.

Kapitel 6 - Filtern von Trajektorien im zeitlichen Zusammenhang Dieses Kapitel beschreibt den zweiten Filter, der Trajektorien im zeitlichen Zusammenhang filtert – auf Grundlage ihrer Eigenschaften zum selben Zeitpunkt. Dabei werden auch hier die Konfigurationsmöglichkeiten und die Arbeitsweise des Filters erläutert.

Kapitel 7 - Validierung Die Validierung dient dazu herauszufinden, wie sich die vorgestellten Filter einsetzen lassen und wie gut sie funktionieren. Durch ein Fallbeispiel wird der Einsatz der Filter hinsichtlich ihrer Effektivität und Effizienz beschrieben und bewertet.

Kapitel 8 - Zusammenfassung Dieses Kapitel fasst die Arbeit abschließend zusammen und gewährt einen Ausblick auf offene Probleme und künftige Anwendungsgebiete.

Kapitel 2

Grundlagen

In diesem Kapitel werden einige Grundlagen vorgestellt, die für die Thematik bzw. den weiteren Verlauf von Bedeutung sind. Zudem werden einige Definitionen, Konzepte und Techniken erläutert, auf deren Grundlage die Konzeption der skizzenbasierten Trajektoriensuche basiert.

2.1 Visual Analytics

Visual Analytics ist ein Begriff für die visuelle Analyse von Sachverhalten, die sowohl durch automatische Analysemethoden als auch durch visuelle Benutzerschnittstellen unterstützt wird. Präziser ausgedrückt ist Visual Analytics ein iterativer Prozess, der in der Analyse Informationserfassung, Datenvorverarbeitung, Wissensrepräsentation, Interaktion und Entscheidungsfindung umfasst [KMS+08]. Das Ziel von Visual Analytics ist das Erstellen von Werkzeugen und Techniken, die es Benutzern ermöglichen,

- Informationen zu synthetisieren und Einblick in umfangreiche, dynamische, mehrdeutige und oft widersprüchliche Daten zu erhalten,
- Erwartetes zu erkennen und Unerwartetes zu entdecken,
- die Daten schnell, fundiert und verständlich einzuschätzen und
- diese Einschätzung effizient für das weitere Vorgehen zu nutzen [KAF+08].

Visual Analytics verbindet zwei Welten: Die vollautomatische, rechnergestützte Analyse und die visuelle Analyse von Problemen [KMT10]. Für viele analytische Probleme eignet sich auf Grund ihrer Beschaffenheit eine vollautomatische Analyse zur Lösung. Dies ist dann der Fall, wenn sich das Problem durch einen Algorithmus oder eine ähnliche logische Struktur ausdrücken lässt und darüber hinaus kein weiteres Kontextwissen zur Lösung des Problems benötigt wird. Mit der hohen Rechenleistung moderner Computer können solche Probleme sehr schnell, effizient und fehlerfrei gelöst werden. Jedoch lassen sich einige Probleme nicht oder nur mit großem Aufwand vollautomatisch durch Rechner lösen. Oft sind

diese Probleme einfach zu komplex, um eine exakte Definition zu finden, die ein Rechner zu lösen versteht oder es wird zusätzliches (menschliches) Kontextwissen benötigt, um die Richtigkeit einer Lösung zu erkennen.

Visualisierungen sind dafür zuständig, umfangreiche Daten schnell und übersichtlich grafisch darzustellen, um die Analyse dieser Daten visuell zu ermöglichen. Sie dienen zur Kommunikation zwischen den Welten der automatischen und der manuellen Analyse. Ziel der Visualisierung ist es, die Information aus den Rohdaten oder bereits vorprozessierten Daten dem Benutzer auf effiziente Weise zu übermitteln. Dies soll so geschehen, dass die kommunizierte Information für den Benutzer verständlich und leicht zu erfassen ist. Durch Interaktion kann der Benutzer auf die analysierte Information reagieren und weiteres Vorgehen im Visual-Analytics-Prozess steuern.

Visual Analytics kombiniert die stärken beider Gebiete: auf der einen Seite steht die schnelle und automatische Analyse intelligente Algorithmen und hohe Rechenleistung zur Verfügung, auf der anderen Seite die visuelle Analyse, die die Vorteile menschlicher Fähigkeiten nutzt, komplexe und unbekannte Sachverhalte zu erkennen. Ein Visual-Analytics-System erlaubt somit dem Benutzer, analytische Probleme mittels einer Kombination aus effizienten, automatisierten Vorgängen und eigenem Wissen zu lösen – oft sogar ohne eine genaue Definition des vorhandenen Problems zu kennen. Dabei ist das Hauptziel von Visual Analytics meist die effiziente Analyse großer Mengen von Daten, um relevante Informationen visuell zu identifizieren und herauszufiltern [KAF+08].

Abbildung 2.1 illustriert die generellen Probleme in der IT und die analytischen Probleme, welche sich teilweise durch automatische Analyse, Visualisierung oder die Kombination aus beiden – Visual Analytics – lösen lassen. Zu beachten ist, dass nicht jedes durch automatische oder visuelle Analyse lösbare Problem gleich ein Visual-Analytics-Problem ist, falls sich das Problem durch andere effektive Methoden lösen lässt [KMT10].

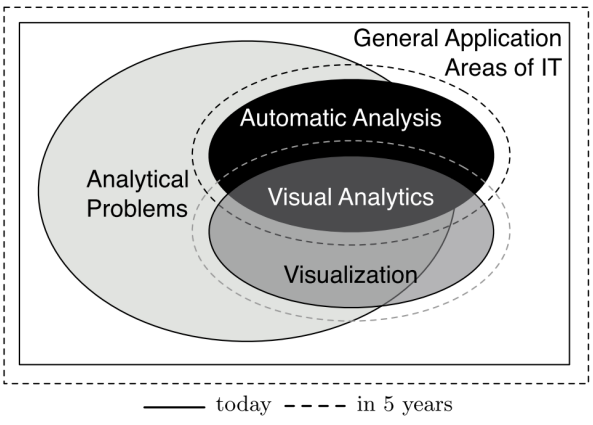

Abbildung 2.1: Analytische Probleme lassen sich teilweise durch automatische Analyse, Visualisierung und ihrer Kombination (Visual Analytics) lösen [KMT10].

Das Mantra der visuellen Analyse

Shneiderman hat als Prozess der visuellen Informationssuche folgendes Mantra formuliert: "Overview first, zoom and filter, then details-on-demand" [Shn96]. Demnach soll sich der Benutzer zuerst einen Überblick über die zu analysierende Information verschaffen können. Die iterative Eingrenzung der Datenmenge geschieht mittels Filtern und Aggregation (Zoom). So kann die gesamte Datenmenge auf die relevante Information reduziert werden, die meist nur einen Bruchteil der gesamten Daten darstellt. Wurde die relevante Information aufgespürt, kann diese mit allen Details dargestellt werden.

Viele Systeme begründen ihre Designentscheidungen genau auf diesem Mantra. Dieses Mantra wurde von Keim im Hinblick auf die visuelle Analyse angepasst: "Analyze first, Show the Important, Zoom, filter and analyze further, Details on demand" [KAF+08]. So formuliert fordert dieses Mantra speziell die Kombination von Analyseansätzen mit intelligenten Visualisierungstechniken. Der iterative Analyseprozess im *VAC* kann ebenfalls mit Hilfe von Keims Mantra umschrieben werden, wie in Abschnitt 3.7 auf Seite 35 verdeutlicht wird.

2.2 Object-Tracking in Überwachungsvideos

Das automatische Auffinden von Objekten in Videos ist ein starkes Merkmal der visuellen Analyse von Videos, auf das viele Analysetechniken zurückgreifen. Dabei sind vor allem Objekte interessant, die sich bewegen, d.h. im Video ihre Position verändern und sich so von ihrer (unbewegten) Umgebung abheben. Im Bereich der Verkehrsüberwachung sind das z.B. Autos, Radfahrer oder Fußgänger. Object-Tracking ist eine Methode, Objekte auf Grund ihrer Bewegung innerhalb des Videokontextes zu finden.

Oft ist nicht nur die Bewegung eines Objektes von Interesse, sondern auch seine genaue Position. Eine Bewegung im Kontext eines Videos geschieht allerdings immer relativ zur Kamera. Daher ist die exakte Position eines Objekts immer abhängig von Position und Blickwinkel der Kamera. Überwachungsvideos sind eine Art von Video, die dieses Problem durch ihre Charakteristik deutlich eingrenzen und eine exakte Positionsbestimmung eines Objekts im Video zulassen.

2.2.1 Charakteristik von Überwachungsvideos

Ein Überwachungsvideo ist ein Video, das von einer Überwachungskamera aufgezeichnet wird. Die Charakteristik einer Überwachungskamera ist, dass sie ihren Standort nicht ändert. Dabei fängt die Kamera entweder immer den selben Bildausschnitt ein oder bewegt sich in einem definierten und sehr begrenzten Rahmen.

Da Überwachungsvideos aber oft rund um die Uhr aufgezeichnet werden, kann ein Ausschnitt eines Überwachungsvideos sehr lang sein – eventuell mehrere Stunden oder sogar

Tage. Die Menge an relevanten Informationen, die aus solch einem langen Video extrahiert werden, kann enorm sein, etwa wenn auf dem Video sehr viele Menschen zu sehen sind – wie in einer Fußgängerzone oder einer Eingangshalle. Ein Object-Tracking-System, das Bewegungsdaten extrahiert und analysiert, sollte so entworfen sein, dass es mit diesen großen Datenmengen umgehen kann, bzw. in dieser Hinsicht skalierbar ist.

2.2.2 Object-Tracking und Trajektorien

Das Ergebnis von Object-Tracking sind sogenannte Trajektorien. Eine Trajektorie beschreibt in erster Linie die Bewegung eines Objekts im räumlichen und zeitlichen Zusammenhang. Sie beschreibt den Pfad, den ein Objekt innerhalb einer gewissen Zeit im Raum zurücklegt. Meist wird eine Trajektorie durch eine Reihe von Beobachtungen oder Messwerten definiert, die jeweils den Ort des Objekts zu einem bestimmten Zeitpunkt angeben. Für die Definition der Trajektorie zwischen ihren Beobachtungen werden Interpolationsmethoden angewendet. Trajektorien werden oft in der Geoinformatik für Object-Tracking verwendet [AAW07, LDFW07]. So macht es auch Sinn, Trajektorien für das Verfolgen von Objekten innerhalb eines Überwachungsvideos zu verwenden.

In dieser Problemstellung werden Trajektorien verwendet, um Bewegungen von einzelnen erkannten Personen, Fahrzeugen, o.Ä. in einem Überwachungsvideo zu beschreiben. Normalerweise ist dabei nicht die Position im Bild, sondern die (zweidimensionale) Position in der tatsächlichen, aufgenommenen Umgebung interessant. Unter der Annahme, dass die Grundebene der Szene im Kameraausschnitt eben ist, kann die echte Position der Trajektorien mittels Projektion von der Bild- in die Grundebene ermittelt werden. Sind also alle Projektionsinformationen der Kamera bekannt (Ort, Blickwinkel, Zoomstufe, etc.), kann in jedem Videoframe die Position der Objekte aus der Bildebene in ihre tatsächliche Position in der Szene transformiert werden (siehe Abschnitt 3.1.2 auf Seite 27). Somit können auch Abstände zwischen Objekten und ihre Entfernung zur Kamera korrekt berechnet werden.

2.2.3 Systeme zur Videoanalyse

Calderara et al. [CPC09] stellt ein Videoüberwachungssystem vor, das auf Trajektorien arbeitet, die sich nach Position und Form durchsuchen lassen. Durch Clustering werden die Anzahl der Vergleiche zu einer Abfragetrajektorie verringert und die Suche effizienter gestaltet. Dabei werden die Trajektorien ebenfalls in den von der Kamera erfassten Raum transformiert.

Ein weiteres Überwachungssystem, das in ähnlicher Weise Trajektorien aufzeichnet, ist das von Girgensohn et al. entwickelte *Dynamic Object Tracking System* (*DOTS*) [GSTW07, GKV+07]. *DOTS* benutzt jedoch gleich eine ganze Reihe von Kameras, um Bewegungen über mehrere Kameraausschnitte hinweg zu erfassen. Die Trajektorien werden in eine Top-View-Sicht transformiert und gesamt von oben dargestellt, z.B. in einem Stockwerksplan eines

Gebäudes. Höferlin et al. [HHW09] verwendet ein *VideoPerpetuoGram* [BSE08] zur Analyse von Trajektorien in Überwachungsvideos, welche ihre Fortsetzung in [HHWH11b] findet.

2.3 Query-Methoden

Ein Query ist eine Anfrage an ein Daten enthaltendes System mit dem Ziel, einen definierten Teil dieser Daten als Antwort zu erhalten. Meist wird der Begriff "Query" in Verbindung mit relationalen Datenbanken gebracht, die textbasierte Queries mittels geeigneter Sprachen (z.B. SQL) ermöglichen. Multimediadatenbanken, die Bilder oder Videos enthalten, eröffnen neue Anforderungen an die gestellten Datenbank-Queries. So sind die Objekte in so einer Datenbank sehr viel komplexer als Einträge in einer klassischen relationalen Datenbank. Multimedia-Anfragen enthalten oft Anforderungen an Positionen, Formen oder Farben, die sich gar nicht exakt, sondern nur ungefähr definieren lassen [Fag98]. Daher ist das Formulieren einer geeigneten Anfrage für Multimediadatenbanken von besonderer Wichtigkeit.

Für die skizzenbasierte Trajektoriensuche in Videos sollen Queries an eine Trajektoriendatenbank gestellt werden, deren Inhalte zuvor aus Videos extrahiert wurden. Dafür gibt es generell drei Abfragemethoden, die für das Durchsuchen von Trajektorien geeignet sind: Query by Example, Query by Sketch und Semantic-based Query [BPB+10].

Query by Example

Das Erstellen einer Abfrage anhand eines bestehenden Beispiels ist die einfachste und schnellste Möglichkeit, eine Abfrage zu formulieren. Für die Abfrage wird eine bestehende Trajektorie aus dem Datenbestand von Videos extrahierten Trajektorien ausgewählt. Die Abfrage zielt darauf ab, Trajektorien zu finden, die der ausgewählten Trajektorie in bestimmen Eigenschaften ähnlich sind.

Der Vorteil dieser Variante ist, dass der Benutzer die Trajektorie nicht von Hand zeichnen muss und somit relativ schnell eine Abfrage formulieren kann. Allerdings muss sich eine geeignete Trajektorie bereits im Datenbestand befinden. Es lassen sich lediglich Abfragen formulieren, für die bereits eine Beispieltrajektorie existiert. Zudem sind die räumlichen Koordinaten von den Aufnahmebedingungen des Videos abhängig. Ändern sich Kamerawinkel, Zoomstufe etc. kann es sein, dass die räumlichen Eigenschaften der ausgewählten Trajektorie und die der Trajektorien aus dem zu durchsuchenden Video nicht zusammenpassen. Befindet sich eine passende Trajektorie im Datenbestand, muss diese auch erst darin gefunden werden. Das Verfahren eignet sich also nicht für die Suche nach einer prinzipiellen Idee des Benutzers, sondern eher als Folge einer vorangegangenen Anfrage, aus deren Ergebnismenge die Beispieltrajektorie ausgewählt wird.

Query by Sketch

Der Benutzer erstellt eine Anfrage mittels einer selbst erstellten Trajektorie, die er zuvor skizziert hat. Das Zeichnen einer Skizze bietet eine sehr einfache Möglichkeit, die räumlichen Aspekte einer Trajektorie zu modellieren. So kann auf einfachem Weg eine Trajektorie für die Anfrage erstellt werden, ohne dass ein entsprechendes Beispiel im Video existieren muss. Zu beachten ist jedoch, dass solche Queries ohne Wissen über den Inhalt des Datenbestands u.U. keine oder unerwartete Ergebnisse liefern können, so dass der Benutzer Zeit mit sinnlosen Queries vergeudet.

Der Query-by-Sketch-Ansatz wird in dieser Arbeit für die skizzenbasierte Trajektoriensuche verwendet (siehe Kapitel 4 auf Seite 37). Chang et al. implementiert den Ansatz auf ähnliche Weise in seinem System *VideoQ* [CCM+98].

Semantic-based Query

Die wohl bisher vielversprechendsten Ansätze der trajektorienbasierten Videoanalyse versuchen eine semantische Beschreibung des Bewegungspfades eines Objekts zu extrahieren und das Objekt für eine spätere Suche zu annotieren. Auf Basis dieser Semantiken werden die Suchanfragen formuliert, sogenannte "Semanic-based Queries". So eine Anfrage enthält eine semantische Beschreibung der gesuchten Trajektorien. Will man z.B. im Überwachungsvideo einer Eingangshalle die Trajektorien aller Personen finden, die den Aufzug benutzen, könnte die semantische Anfrage etwa so aussehen: "Finde alle Personen, die zum Aufzug gehen". Es wird nicht nach unmittelbaren Eigenschaften der Trajektorien gefragt, sondern nach semantischen Informationen, die vom System interpretiert werden, im Fall des Beispiels die Information "Person geht zum Aufzug".

Vorteil dieser Methode ist, dass keine technischen Aspekte über die Art der Trajektoriendaten bekannt sein müssen, sondern nur deren Semantik interessiert. Allerdings können in Anfragen lediglich Semantiken verwendet werden, die dem System bekannt sind. Eine unbekannte Semantik kann das System nicht verarbeiten. Semantic-based Queries kommen z.B. bei Hu et al. [HXF+07] zum Einsatz.

2.4 Skizzenbasierte Trajektorien-Queries

In diesem Abschnitt werden einige Grundlagen und Konzepte vorgestellt, die für das Erstellen und Durchführen von Trajektorienqueries von Bedeutung sind. Hier zu gehört die Modellierung der Trajektorienskizze sowie Methoden für den Vergleich mit anderen Trajektorien bezüglich ihrer Eigenschaften. Zusätzlich ist bei unscharfen Anfragen die Definition von Unsicherheiten und Relevanzen wichtig.

2.4.1 Trajektorienmodellierung

Für die Erstellung eines skizzenbasierten Queries muss die Skizze der Query-Trajektorie zunächst modelliert werden. Hierbei erhält sie die Eigenschaften, die für das Query relevant sind. Die Modellierung geschieht normalerweise über eine Zeichenoberfläche, auf die die Trajektorie schematisch gezeichnet (skizziert) wird.

Das Videoanalysesystem *VideoQ* von Chang et al. [CCM+97] bietet die skizzenbasierte Modellierung von Trajektorien über eine abstrakte Zeichenfläche. Im Folgenden wird das System kurz vorgestellt, um die Konzepte zu erläutern.

VideoQ

VideoQ ist ein automatisches System zum Durchsuchen von Videos mittels skizzenbasierten Anfragen. Laut eigenen Angaben war es das erste Online-Suchsystem mit objektbasierter Indizierung und Unterstützung für raumzeitliche Suchanfragen [CCM+98].

Das Hauptaugenmerk von *VideoQ* liegt auf der Unterstützung von "animierten" Skizzen, um raumzeitliche Suchanfragen zu formulieren. Eine Skizze beschreibt dabei als Schlüsselkomponenten die räumliche Laufbahn eines sich bewegenden Objekts so wie die zeitliche Dauer der Bewegung. Für die räumliche Beschreibung der Bewegung "zeichnet" der Benutzer

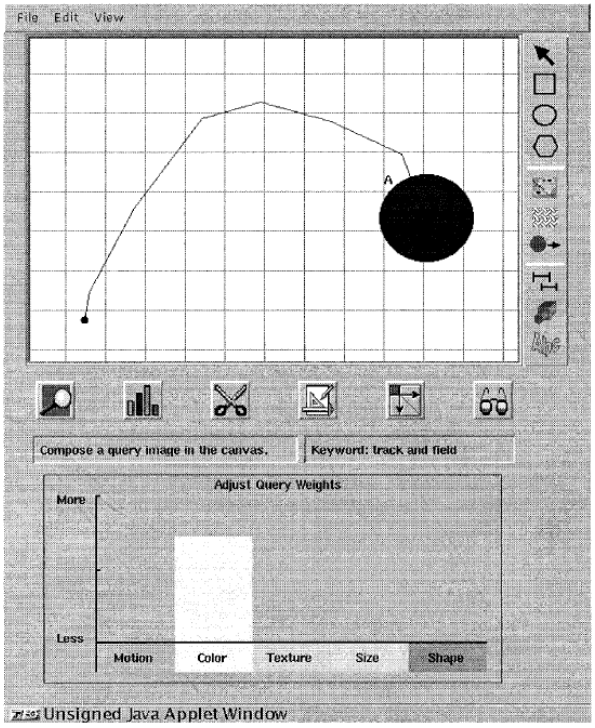

Abbildung 2.2: Benutzeroberfläche in *VideoQ* zum Zeichnen einer Trajektorie [CCM+98].

durch Anklicken mehrerer Punkte auf der Zeichenoberfläche eine polygonale Trajektorie. Die Dauer der spezifizierten Bewegung wird dabei sehr intuitiv angegeben, mit den Werten “kurz”, “mittel” oder “lang”. Zusätzlich können Eigenschaften wie Form, Farbe oder Oberflächenstruktur definiert und entsprechend ihrem Maß an Bedeutung gewichtet werden. Abbildung 2.2 zeigt die Benutzeroberfläche von *VideoQ* zum Erstellen der Anfragetrajektorie und Modellieren ihrer Eigenschaften.

Damit die gezeichnete Trajektorie als Anfrage verarbeitet werden kann, wird sie so interpoliert, dass ihre Stützpunkte zeitliche Abstände entsprechend der Framerate des Videos aufweisen. Die angegebene zeitliche Information “kurz”, “lang”, oder “mittel” wird vom System auf Basis der Framerate interpretiert.

Changs System zeigt, dass sich das Erstellen einer Skizze sehr gut zum Modellieren einer Query-Trajektorie eignet. Die gleiche Idee wird auch in dieser Arbeit beim Erstellen einer Trajektorienskizze verfolgt und weiter präzisiert (siehe Kapitel 4 auf Seite 37). So bietet die entwickelte Skizzierungsoberfläche im Gegensatz zu *VideoQ* zusätzlich eine Einordnung der Skizze in den zu durchsuchenden Videoausschnitt, sowie präzisere Möglichkeiten zur Modellierung der zeitlichen Eigenschaften.

Trajektoriendarstellung

Um das Skizzieren einer Trajektorie einfach und intuitiv zu gestalten, ist eine geeignete Darstellung für die Skizze sehr wichtig. Über die visuelle Darstellung bekommt der Benutzer Feedback, wie und an welcher Stelle er die Skizze verändern kann und welche Auswirkungen die Änderungen haben.

VideoQ verwendet zur Darstellung der gezeichneten Trajektorie eine einfache schwarze Linie (siehe Abbildung 2.2). Dabei ist nicht ersichtlich, ob und wie die gezeichnete Trajektorie verändert werden kann. Zudem scheint es durch den fehlenden Hintergrund schwer, die gezeichnete Trajektorie im Videokontext einzuordnen.

ISS von Höferlin et. al [HHWH11a] fasst die Trajektorien eines Videos nach bestimmten Kriterien zusammen und stellt die zusammengefassten Cluster schematisch dar. Dabei wird eine cartoonartige Darstellung für die Trajektorien verwendet, die als Repräsentation ihres Clusters dienen. Diese werden als breite Pfeile dargestellt (siehe Abbildung 2.3). Pfeile, die zum gleichen Cluster gehören, werden als einzelnes, auswählbares Objekt kombiniert dargestellt. Durch die Transparenz und die verschiedenen Farben der Pfeile können diese leicht unterschieden und ausgewählt werden. Die Trajektorien sind hier nicht zum Verändern durch den Benutzer gedacht. Dementsprechend werden keine Möglichkeiten zur Verfügung gestellt, die eine Bearbeitung zulassen.

Die Darstellung auf dem Hintergrund des analysierten Videos ermöglicht es dem Benutzer, die Trajektorien direkt im räumlichen Kontext des Videos zu betrachten. Beim Skizzieren der Trajektorien im *VAC* wird diese Pfeildarstellung für die Skizzentrajektorie teilweise wiederverwendet und um Konstrukte, die zum Bearbeiten der Trajektorie nötig sind, erweitert (siehe Abschnitt 4.2 auf Seite 39).

Abbildung 2.3: Darstellung von Trajektorienclustern in *ISS* [HHWH11a].

2.4.2 Trajektorienvergleich

Die Bearbeitung einer Suchanfrage nach Trajektorien erfordert eine Möglichkeit, die Trajektorien miteinander zu vergleichen. Da Trajektorien komplexe Gebilde sind und eine Vielzahl verschiedener Eigenschaften aufweisen, lassen sich unterschiedliche Kriterien definieren, die Grundlage für einen Trajektorienvergleich bieten. Der Vergleich kann in einer oder mehreren dieser Eigenschaften erfolgen. Vergleichbare Trajektorieneigenschaften wie z.B. Form, Position, Richtung, Geschwindigkeit oder Zeit haben jedoch viele verschiedene Werte innerhalb der Trajektorie, die z.B. durch Stützpunkte definiert sind. Zudem kann die Länge von Trajektorien und somit auch die Anzahl von Stützpunkten stark variieren. Zusätzlich zu den zu vergleichenden Eigenschaften einer Trajektorie müssen also auch vernünftige Methoden definiert werden, die beschreiben, wie die komplexen Eigenschaften miteinander verglichen werden.

Eine naheliegende und auch weit verbreitete Methode für den Vergleich von Trajektorieneigenschaften ist das Abbilden von Unterschieden auf skalare Distanzwerte. Eine Distanz definiert, wie unterschiedlich zwei Trajektorien bezüglich einer Eigenschaft zueinander sind. Kann diese Eigenschaft auf einen skalaren oder vektoriellen Wert abgebildet werden, kann z.B. die euklidische Distanz zur Bestimmung des Unterschieds verwendet werden. Jedoch sind auch andere Methoden zur Berechnung von Distanzen möglich. Zhang et al. [ZHT06] stellt z.B. neben der euklidischen Distanz auch andere Vergleichsmethoden vor, vor allem für den Vergleich der positionalen Eigenschaften von Trajektorien.

2.4.3 Unsicherheit und Relevanz

Das Definieren einer Unsicherheit ist beim Einsatz von unscharfen Anfragen in Trajektorienfiltern von essentieller Bedeutung. Durch sie kann für jede Trajektorie ausgedrückt werden, mit welcher Sicherheit (oder Unsicherheit) sie den gewünschten Beitrag zum Ergebnis liefert. Dies ist Grundlage für die Entscheidung, ob die Trajektorie zur Ergebnismenge gehört oder nicht – also ob die Sicherheit hoch genug ist, dass die Trajektorie zum gewünschten Ergebnis gehört.

Die Bestimmung der Relevanz einer Trajektorie dient dazu, eine Aussage darüber zu treffen, wie bedeutsam die Trajektorie für das Ergebnis ist. Dadurch können Trajektorien in der gefilterten Ergebnismenge daran unterschieden werden, wie relevant sie als Ergebnis der Anfrage sind. Höferlin et al. [HHWH11b] verwendet Relevanzen in verschiedenen Filtern, durch welche jeweils eine Eigenschaft der Trajektorien (z.B. Position, Bewegungsrichtung, Geschwindigkeit und Dauer) je nach Wert und ihrer Bedeutung im Gesamtergebnis bewertet werden kann. Abbildung 2.4 zeigt ein Beispiel eines solchen Filters, in diesem Fall ein Positionsfilter. Hier kann durch ein Malwerkzeug der Bereich auf dem Videohintergrund definiert werden, der für den Benutzer von Interesse ist. Somit werden Trajektorien in diesem Bereich als relevant eingestuft. Durch einen einstellbaren Fuzzy-Wert lässt sich die Relevanz des definierten Bereiches verändern.

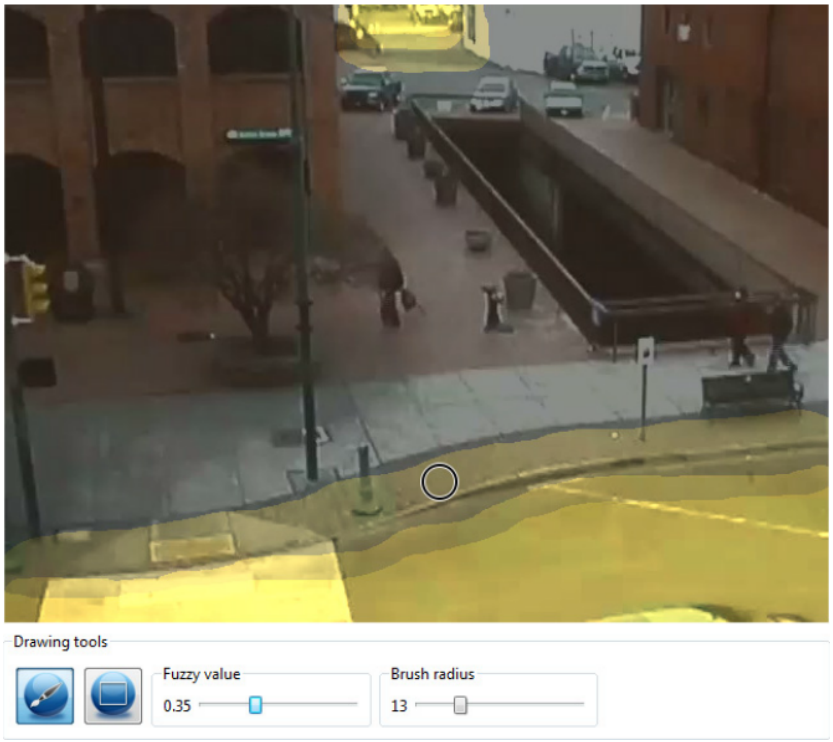

Abbildung 2.4: Zeichenoberfläche des Positionsfilters bei Höferlin et al. [HHWH11b].

Letztendlich können durch solch eine Bewertung Trajektorien gefunden werden, deren Eigenschaften den eingestellten Filtern entsprechen. Das Query-by-Sketch-Prinzip wird in dieser Arbeit leicht abgewandelt: es lassen sich beinahe die selben Trajektorieneigenschaften in den Filtern formulieren, jedoch liegt der Fokus in der Modellierung einer eigenen Trajektorie mit kompletten oder teilweise definierten Eigenschaften (siehe Kapitel 4 auf Seite 37). Zusätzlich kann eingestellt werden, wie weit gesuchte Trajektorien in bestimmten Eigenschaften von der modellierten abweichen dürfen.

Im *VAC* (Kapitel 3 auf Seite 25) läuft die Bewertung der Trajektorien zwar ähnlich ab, wird aber in zwei Stränge aufgeteilt: durch eine Reihe von Filtern in einem Filtergraphen wird (teilweise unter Verwendung einer Unsicherheit) ausgewertet, ob eine Trajektorie zur Ergebnismenge gehört oder nicht (siehe Abschnitt 3.5 auf Seite 31). In einen weiteren, parallel geführten Graphen lassen sich sog. Relevanzbausteine einfügen, die die Trajektorien nicht filtern, sondern die Aufgabe haben, die Trajektorien zusätzlich hinsichtlich der Relevanz für das Ergebnis zu bewerten (siehe Abschnitt 3.5.3 auf Seite 34). Durch Filter wird als entschieden, *ob* eine Trajektorie im Ergebnis auftaucht, durch die Relevanz können die Trajektorie *innerhalb* des Ergebnisses z.B. in ihrer visuellen Repräsentation (Form, Farbe, ...) unterschieden werden.

2.5 Fuzzy-Logik und Fuzzy Queries

Die klassische Aussagenlogik kennt genau zwei Ergebniswerte: “wahr” und “falsch”. Manchmal ist es jedoch notwendig Aussagen zu modellieren, die nur zu einem gewissen Anteil wahr oder falsch sind. Für solche “unscharfen” Aussagen bedarf es einer Erweiterung der Aussagenlogik auf flexiblere Werte. Eine solche Erweiterung bietet die Fuzzy-Logik (“unscharfe” Logik) [Zad65, Zad88].

Die Fuzzy-Logik erweitert die klassische Aussagenlogik auf die Ergebniswertmenge $[0, 1]$. Der Wert 0 bedeutet *falsch*, ein Wert von 1 entspricht dem booleschen *wahr*. In allen Werten dazwischen zerfließt die Aussage über *wahr* und *falsch* in prozentuale Anteile. Ist z.B. eine Aussage A zu 60% wahr, erhält sie den Wahrheitswert 0,6.

Um die Vorteile von Fuzzy-Logik zu verdeutlichen, führt Fagin [Fag96] ein einfaches Beispiel mit Datenbankabfragen auf: Man stelle sich vor, man möchte an eine Datenbank bestehend aus Informationen über Musikalben einige Anfragen stellen. Die atomare Abfrage “$Interpret = Beatles$” kann für jedes Objekt eindeutig mit *wahr* (1) oder *falsch* (0) beantwortet werden. Fragen wir allerdings auf multimedialer Ebene z.B. nach der Farbe des Album-Covers (“$AlbumCover = rot$”) kann das Ergebnis mittels der Fuzzy-Logik für jedes Objekt ein Fuzzy-Wert zwischen 0 und 1 enthalten, je nach dem wie viel “rot” das Cover enthält. So eine Abfrage nennt sich *Fuzzy Query*.

Fuzzy-Operatoren

Um eine vollständige Logik zu definieren, müssen geeignete Operatoren für Konjunktion, Disjunktion und Negation gefunden werden. Zadeh [Zad65] schlägt hierbei folgende Funktionen vor: Sei $\mu_A(x)$ der Grad der Zugehörigkeit eines Objekts x in der Menge A und $\mu_B(x)$ der Grad der Zugehörigkeit von x in der Menge B. Dann lässt sich jeweils die Zugehörigkeit des Objekts x in Schnitt, Vereinigung und Komplement der Mengen wie folgt ausdrücken:

$$\begin{aligned} \text{Konjunktion:} \quad & \mu_{A \wedge B}(x) &&= \min\{\mu_A(x), \mu_B(x)\} \\ \text{Disjunktion:} \quad & \mu_{A \vee B}(x) &&= \max\{\mu_A(x), \mu_B(x)\} \\ \text{Negation:} \quad & \mu_{\neg A}(x) &&= 1 - \mu_A(x) \end{aligned}$$

Die Min- und Max-Operatoren stellen eine T-Norm und T-Conorm dar. Im Bereich der Fuzzy-Logik sind dies die meist verbreiteten Operatoren. Theoretisch kann hier aber jede beliebige T-Norm und T-Conorm als Operator eingesetzt werden [Web83]. Jedoch sind die hier verwendeten Funktionen als sinnvoll definierte Fuzzy-Operatoren anzusehen [BG73, TZZ79].

Trajektorien und Fuzzy-Logik

Wie in Fagins Plattencover-Beispiel lassen sich mit Hilfe der Fuzzy-Logik und deren Operatoren auch komplexere Abfragen aufstellen. Der Vergleich von Trajektorien in dieser Arbeit erfolgt mit Hilfe dieser Logik. So lässt sich nicht nur bestimmen, ob sich zwei Trajektorien in einer bestimmten Eigenschaft ähnlich sind oder nicht, sondern auch zwischen verschiedenen Graden von Ähnlichkeit unterscheiden (siehe Abschnitt 5.2 auf Seite 49).

Mit dieser Differenzierung lassen sich nun Fuzzy-Queries für Trajektorien aufstellen, die besser auf ungenaue Angaben – wie es beim Zeichnen einer Skizze oft der Fall ist – zugeschnitten sind. Beispielsweise werden Trajektorien gesucht, die sich mit einer skizzierten Trajektorie positional (weitgehend) überdecken und ungefähr in die gleiche Richtung verlaufen. In diesem Fall ließe sich für beide Kriterien einzeln ein Schwellenwert festlegen, unter welcher Überdeckungs- bzw. Richtungsdifferenz die einzelnen Kriterien als *wahr* bezeichnet werden. Zieht man jedoch die Möglichkeit in Betracht, dass auch Trajektorien akzeptabel sind, die nur eine geringe Überdeckung aufweisen, jedoch in der Richtung sehr gut zusammenpassen (oder umgekehrt), dann lässt dieser Fall keinen eindeutigen Schwellenwert zu. Mit dieser Gegebenheit lässt sich umgehen, wenn die Kriterien mittels Fuzzy-Logik zu einem Fuzzy-Query kombiniert werden [DP97].

Ein weiterer Punkt ist die Gewichtung der einzelnen Kriterien. Ist ein Teil eines Queries bedeutsamer als der Rest, können die "unscharfen" Ergebnisse problemlos entsprechend ihrer Bedeutung gewichtet werden.

Kapitel 3

Das Visual Analytics Center

Das *Visual Analytics Center (VAC)* ist ein Framework für das Extrahieren und Verarbeiten von Inhalten (sog. "Features") aus diversen Datenquellen. Der primäre Zweck dieses Frameworks ist die Verarbeitung von Videoquellen, um darin enthaltene Informationen visuell und interaktiv für eine Analyse zur Verfügung zu stellen. In diesem Fall wird das Framework für die Extraktion von Trajektorien aus Überwachungsvideos benutzt. Diese Trajektorien werden dann für eine weitere Analyse visuell zur Verfügung gestellt.

Die Benutzeroberfläche enthält drei wichtige Komponenten: Mit Hilfe einer Timeline kann durch das geöffnete Video navigiert werden. In einem Filterbereich lassen sich diverse Filter für die extrahierten Inhalte definieren (siehe Abschnitt 3.5 auf Seite 31). In die eigentliche Arbeitsoberfläche können verschiedene Ansichtsfenster (sog. "Views") eingefügt werden, die die Ergebnisse visuell darstellen und ggf. interaktive Funktionen anbieten. Abbildung 3.1 zeigt die Benutzeroberfläche mit diesen drei Komponenten. Das Framework wird u.a. für *ISS* [HHWH11a, HHWH12] verwendet, das in die Hauptanwendung integriert ist. Die skizzenbasierte Trajektoriensuche ist ebenfalls in das Framework eingebettet und steht als Filter für Trajektorien zur Verfügung.

Echtzeit-Analyse und statische Analyse

Das *VAC* bietet zwei verschiedene Methoden zur Analyse von Videos oder anderer Daten. Eine Methode ist die Echtzeit-Analyse. Dies bedeutet, dass die extrahierten Daten beim Abspielen des Videos (oder anderer zeitbezogener Daten) exakt zu dem Zeitpunkt zur Verfügung stehen, zu dem sie wiedergegeben werden. Im Gegenzug wird das Video nicht weiter abgespielt, wenn noch nicht alle notwendigen Daten dieses Zeitpunktes extrahiert sind. Durch Vorausberechnung und Caching läuft die Wiedergabe allerdings auch bei rechenintensiveren Operationen oder großen Datenmengen meist flüssig.

Die Echtzeit hat den Vorteil, dass theoretisch auch Live-Videoströme direkt verarbeitet werden können. Durch geschicktes Caching und dem Löschen nicht mehr relevanter Daten wäre sogar die Echtzeit-Analyse einer endlosen Videoaufzeichnung denkbar. Allerdings wäre eine flüssige Wiedergabe in diesem Fall bei rechenaufwändigen Daten wohl nur

Abbildung 3.1: Benutzeroberfläche des *VAC*. Im unteren Bereich befindet sich die Timeline, rechts der Filterbereich. Im mittleren Bereich finden die geöffneten Views Platz.

zeitverzögert möglich. Die Echtzeit-Verarbeitung der Daten bringt zudem Probleme mit sich, insbesondere wenn extrahierte Daten untereinander Verglichen werden sollen. Das Problem wird in Abschnitt 6.1 auf Seite 59 im Zusammenhang mit dem Vergleich von Trajektorien verdeutlicht.

Die zweite Analyse-Methode ist die statische Analyse. Dabei wird ein Video (oder ein Datensatz anderer Daten, z.B. Trajektorien) komplett in einen Cache geladen, bevor eine für die statische Analyse geeignete Views alle extrahierten Informationen auf einmal bekommt. So kann unabhängig von der Abspielzeit eine Gesamtübersicht aller enthaltenen Informationen erstellt und analysiert werden. Allerdings kann diese Methode je nach Größe der Daten sehr speicherhungrig sein, da die gesamten Daten zur Analyse vorgehalten werden müssen. *ISS* nutzt die statische Analyse, um alle Trajektorien eines Videos oder Datenquelle gleichzeitig visuell in Clustern zur Verfügung zu stellen.

3.1 Trajektorien im Visual Analytics Center

Das *VAC* wird dazu benutzt, Trajektorien aus Videos zu extrahieren, zu Filtern und für die visuelle Analyse zur Verfügung zu stellen. Aus dem Video extrahierte Trajektorien werden im weiteren Verlauf als Videotrajektorien bezeichnet. Solche Trajektorien haben – wie andere Features auch – im *VAC* eine bestimmte Datenstruktur, mit dem die verschiedenen Teile der Anwendung umgehen müssen. Das Hauptaugenmerk liegt hier auf der Datenstruktur für Trajektorien, die im Folgenden vorgestellt wird.

3.1.1 Aufbau von Videotrajektorien

Videotrajektorien werden aus bewegten Objekten in einem Video extrahiert und beschreiben deren zeitabhängiges Bewegungsprofil innerhalb des Videobildes. Dieses Bewegungsprofil muss in der Datenstruktur der Trajektorie abgebildet werden. Eine Videotrajektorie im *VAC* besteht aus sog. Beobachtungen ("Observations"). Für jedes Videoframe, in dem das gesuchte Objekt entdeckt wird, wird solch eine Beobachtung erzeugt. Sie enthält den Zeitstempel des Videoframes, einen Positionsrahmen um das gefundene Objekt in Bildkoordinaten sowie die Projektionsmatrix für den aktuellen Kamerawinkel (siehe Abschnitt 3.1.2). Bei einem Video mit 25 Bildern pro Sekunde werden also für jede Sekunde, in der das Objekt im Video zu sehen ist, 25 Beobachtungen generiert.

Resultat ist eine Trajektorie, die durch Stützpunkte ("Samples") und jeweiligen Positions- und Zeitangaben definiert ist. Jede Beobachtung repräsentiert dabei einen solchen Stützpunkt. Berechnungen von Eigenschaften innerhalb der Trajektorie, wie z.B. Abstände oder Geschwindigkeiten, sind dabei nicht auf die Werte der Beobachtungen beschränkt. Werden Werte benötigt, die zwischen diesen Punkten liegen, werden sie durch lineare Interpolation ermittelt.

Eine Videotrajektorie besteht im Endeffekt aus den in einer Liste angeordneten Beobachtungen und diversen Funktionen zur transparenten Berechnung von Eigenschaften der Trajektorie. Da Videotrajektorien für die Repräsentation von bestehenden Informationen (aus dem Video) konzipiert sind, sind sie nach dem Erstellen nicht mehr veränderbar. Das Format eignet sich daher nicht zum Modellieren einer Trajektorie, z.B. für Trajektorien-Queries.

3.1.2 Perspektivische Projektion

Um Trajektorien von ihren Beobachtungen in der Bildebene des Videos in Bezug zur aufgezeichneten, realen Welt zu bringen, werden diese durch eine perspektivische Projektion in Weltkoordinaten der realen Szene transformiert. Die Projektion von Bildkoordinaten in Weltkoordinaten ist von Kameraposition und -winkel abhängig. Durch sie können Längen und Abstände aus den Bildkoordinaten errechnet werden. Dafür wird angenommen, dass die im Bildausschnitt der Kamera zu sehende Szene ebenerdig ist, also eine flache Grundebene

hat, auf dem sich alle Objekte fortbewegen. Höhenunterschiede können somit nicht erfasst werden. Durch diese Bedingung lässt sich aber über die Position im Videobild ermitteln, wie weit ein Objekt von der Kamera entfernt ist (unter der Annahme, dass sich dieses auf der definierten Ebene befindet). Auf diese imaginäre Ebene werden die Trajektorien aus der Videobildebene projiziert, eine Rückprojektion ist ebenso möglich. Sowohl Projektion als auch Rückprojektion sind durch eine Projektionsmatrix definiert. Abbildung 3.2 veranschaulicht die Projektion eines Punktes durch die Matrixtransformation T und die Rückprojektion durch die inverse Transformation T^{-1}.

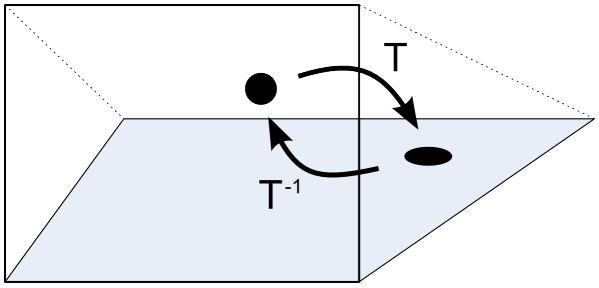

Abbildung 3.2: Die Videobildordinaten werden auf die definierte Grundfläche der Bildszene projiziert. Die Grundfläche ist durch eine Homographie definiert.

Die Projektion der Bildkoordinaten auf die Grundebene und die entsprechende Rückprojektion bilden eine Homographie. Das Framework unterstützt die Analyse von Überwachungsvideos, in denen sich die Kamera im Raum bewegt oder dreht. Somit kann jedem Videoframe eine eigene Homographie mitgegeben werden, die Position und Winkel der Kamera beschreibt. Bleibt die Kamera starr, ist nur eine einzige Homographie nötig.

3.2 Video- und Datenverarbeitung

Kernstück des Frameworks ist die Extraktion von Daten in einem pipelineartigen Vorgang. Hier wird eine Datenquelle (meist ein Video) in vier wesentlichen Schritten verarbeitet, bis die extrahierten Informationen schließlich an eine Reihe von verschiedenen Views zur graphischen Darstellung weitergegeben werden. Die Verarbeitungsschritte sind in vier Komponenten aufgeteilt:

1. Datenquelle
2. Bildmanipulator
3. Feature-Extraktor
4. Filter und Relevanz

Die vier Schritte und das Konzept der Views werden im Folgenden detailliert vorgestellt. Abbildung 3.3 zeigt eine schematische Darstellung der einzelnen Komponenten.

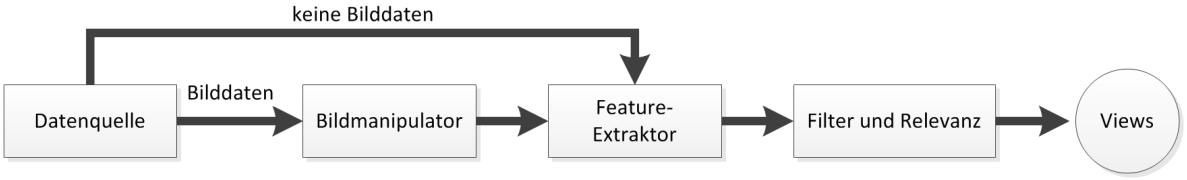

Abbildung 3.3: Die Pipeline des *VAC*.

3.3 Datenquelle und Bildmanipulator

Als Datenquelle können verschiedene Arten von möglichen Daten dienen, die in der Pipeline verarbeitet werden und den Feature-Extraktoren zur Verfügung stehen. Primär werden Videodaten als Quelle zur Analyse bereitgestellt. Ist dies der Fall, können die einzelnen Bilder des Videos durch den Bildmanipulator verändert werden, bevor sie die Feature-Extraktoren erreichen. Dies dient dazu, die Bilddaten durch diverse Bildfilter so aufzubereiten, dass das automatische Extrahieren von Features aus den Bilddaten möglichst gut funktioniert.

Aufbereitung der Videobilder

Die Qualität der Videobilder ist für die maschinelle Extraktion von Daten nicht immer perfekt. Bevor die einzelnen Videoframes vom Feature-Extraktor verarbeitet werden, können diese im Bildmanipulator grafisch aufbereitet werden. Hier können diverse Bildfilter, z.B. zum Entfernen von Rauschen oder zur Farbkorrektur, eingesetzt werden. Anschließend werden die aufbereiteten Bilder zum Extrahieren der Features verwendet.

Der Bildmanipulator ist nur für die grafische Bearbeitung der Videobilder zuständig. Werden weder vom Feature-Extraktor noch von irgendwelchen Views Bilddaten aus dem Video benötigt (evtl. weil die angeforderten Features schon vorberechnet sind oder keine Bildinformationen verwenden), kann der Schritt wegfallen. Das Gleiche gilt natürlich auch für Datenquellen, die keine Bildinformationen enthalten.

Weitere Datenquellen

Eine Datenquelle kann aber durchaus auch andere Daten als Videobilder liefern. Die können beliebige Daten sein, solange es einen Feature-Extraktor gibt, der diese Daten versteht und als Feature zur Verfügung stellt. Beispielsweise können auch aufgezeichnete GPS-Daten von bewegten Objekten als Quelle dienen, aus denen Trajektorien extrahiert und analysiert werden können. Im Fall der Videoanalyse können z.B. bereits aus einem Video extrahierte Trajektoriendaten als Quelle eingesetzt werden, sofern die Bilddaten des Videos nicht benötigt werden.

Die Einspeisung von Daten ist nicht auf eine einzige Datenquelle beschränkt, sondern kann auch durch mehrere Datenquellen gleichzeitig geschehen. So können z.B. neben einem Video als Datenquelle zusätzliche Informationen bereitgestellt werden, die von Filtern oder Views

benötigt werden. Die Feature-Extraktoren kümmern sich darum, dass die Informationen als Features zur Verfügung stehen, sofern diese durch eine geeignete Datenquelle geliefert werden (siehe Abschnitt 3.4). Einige mögliche zusätzliche Daten sind:

Trajektorien Die Berechnung von Trajektorien aus bewegten Objekten in einem Video ist sehr rechenintensiv und ist u.U. sehr zeitaufwändig, so dass eine Extraktion beim Abspielen des Videos kaum möglich ist oder das Video nicht flüssig dargestellt werden kann. Um ein wiederholtes Abspielen des Videos zu beschleunigen und redundante Berechnungen zu vermeiden, lassen sich die extrahierten Trajektorien in einer separaten Datenquelle abspeichern. Ist diese Datenquelle beim erneuten Laden des Videos bereits vorhanden, werden automatisch die bereits bestehenden Trajektoriendaten verwendet, anstatt sie neu zu berechnen. Dadurch lassen sich die Trajektorien ohne erneuten, aufwändigen Berechnungsaufwand während der Videowiedergabe darstellen.

Hintergrundbild Wird anstatt der Videoframes lediglich ein statisches Hintergrundbild des Videos benötigt (z.B. als Kontext für das skizzieren von Trajektorien), kann dies als Quelle beigelegt werden, um die rechenintensive Verarbeitung der Videodaten zu umgehen.

Grundflächen-Homographie Die Grundflächen-Homographie steht ebenfalls als zusätzliche Datenquelle zur Verfügung und enthält die Projektionsinformationen, die den Kamerawinkel beschreiben, mit dem das Video aufgenommen wurde. Mit diesen Informationen werden die Bilddaten aus der Bildebene auf die Grundfläche der Szene transformiert, um Positionen und Abstände zu erhalten, die der realen Welt entsprechen (siehe Abschnitt 3.1.2 auf Seite 27).

3.4 Extrahieren der Features

Die Extraktion der verschiedenen Features aus den Rohdaten der Datenquelle erfolgt durch den Feature-Extraktor. Sowohl Views als auch vorgeschaltete Filter können Features, die sie benötigen, anfordern. Für jedes angeforderte Feature wird ein dafür zuständiger Extraktor aufgerufen, der das Feature aus der entsprechenden Datenquelle extrahiert und zur Verfügung stellt. Theoretisch können so beliebige Daten aus einer Datenquelle extrahiert werden, sofern ein Extraktor dafür existiert. Extraktoren können auch Abhängigkeiten untereinander haben. Benötigt ein Extraktor selbst wiederum die Ergebnisse anderer Extraktoren, kann er Abhängigkeiten zu diesen definieren. Beispielsweise benötigt der Extraktor für Trajektorien die Videoframes, die der Videoframe-Extraktor bereitstellt. Diese Abhängigkeiten werden bei der Reihenfolge der Berechnung der Features berücksichtigt.

Bei der Analyse von Überwachungsvideos sind die Features zur Bereitstellung von Bilddaten und Trajektorien von besonderem Interesse. So werden diese auch in den vorgestellten Filtern verwendet, die diese Features beim Feature-Extraktor anfordern. Auch die meisten Views verwenden diese beiden Features zur Darstellung. Andere Einsatzgebiete sind aber ebenfalls

möglich, so kann eine View z.B. auch Trajektorien auf einer Landkarte anzeigen, die aus GPS-Daten extrahiert wurden.

3.5 Filter

Bevor die Features zu der View gelangen, von der sie angefordert wurden, können sie durch eine Reihe von Filtern auf eine gewünschte Menge reduziert werden. Dafür steht für jede angeforderte Art von Feature ein eigener Filtergraph zur Verfügung, in den zum jeweiligen Feature passende Filter eingefügt werden können. So können z.B. im Filtergraph für Trajektorien nur Filter eingesetzt werden, die mit dem Feature *Trajektorie* umgehen können. Das Filterkonzept wurde für den Einsatz von Filtern im Verlauf dieser Arbeit entwickelt und in das *VAC* integriert.

Jeder Filtergraph hat einen Eingang und einen Ausgang. Die Filter im Filtergraphen können beliebig viele Verbindungen zu vorhergehenden bzw. nachfolgenden Filtern oder zum internen Ein- bzw. Ausgang des Graphen haben. Ein Filtergraph ist dann gültig, wenn eine Verbindung (über diverse Filter) vom Eingang zum Ausgang besteht und der Graph keine Schleifen enthält. Nicht oder teilweise verbundene Filter werden nicht berücksichtigt.

Die zu filternden Features werden auf allen möglichen Pfaden, die mit dem Eingang des Filtergraphen verbunden sind, evaluiert. Passiert ein Feature auf irgendeinem Weg zum Ausgang des Graphen alle Filter auf diesem Weg, wird es an die Views weitergegeben, ansonsten verworfen. Parallel angeordnete Filter stellen somit eine logische Oder-Verknüpfung, seriell verbundene Filter eine logische Und-Verknüpfung dar. Eine Sonderstellung hat der leere Graph, in diesem Fall werden alle Features durchgelassen, also nichts herausgefiltert.

Damit der Filtervorgang flexibler gestaltet werden kann, kann jeder Filter vom Benutzer auf Wunsch deaktiviert, bzw. wieder aktiviert werden. Dadurch können Filter leicht vorübergehend aus dem Vorgang herausgenommen werden, ohne dass der Graph umständlich umstrukturiert werden muss. Ein deaktivierter Filter hat keine filternde Funktion, alle eingehenden Features werden durchgelassen. Als weitere Möglichkeit kann ein Filter im Graphen eine Endmarkierung erhalten. Ein Filter mit dieser Endmarkierung verhält sich so, als wäre er mit dem Ausgang des Graphen verbunden, d.h. nach diesem Filter endet die Auswertung und das Ergebnis wird ausgegeben. Dadurch kann ein Filtergraph nur bis zu einem gewissen Punkt ausgewertet werden. In jedem Filtergraphen kann immer nur ein Filter die Endmarkierung besitzen.

3.5.1 Filter und Container

Ein Filter bewertet jedes ankommende Feature mit *ja* oder *nein* (den booleschen Werten *wahr* oder *falsch*). Er entscheidet also, ob das Feature den Filterpfad weiterverfolgt oder verworfen wird. Die Kriterien, nachdem die Features gefiltert werden, sind von der Implementierung

des jeweiligen Filters abhängig. Dafür stellt jeder Filter einen Benutzerdialog zu Verfügung, mit dem die zur Bewertung benötigten Parameter eingestellt werden können.

Container dienen zur Verschachtelung von Filtern. Ein Container enthält in seinem Innern einen eigenen Filtergraphen, in den wieder Filter und auch Container eingefügt werden können. Nach außen verhält sich der Container wie ein normaler Filter, der das Ergebnis des enthaltenen Filtergraphen nach außen weitergibt. Abbildung 3.4 enthält ein schematisches Beispiel, wie die Struktur eines Filtergraphen aussehen kann.

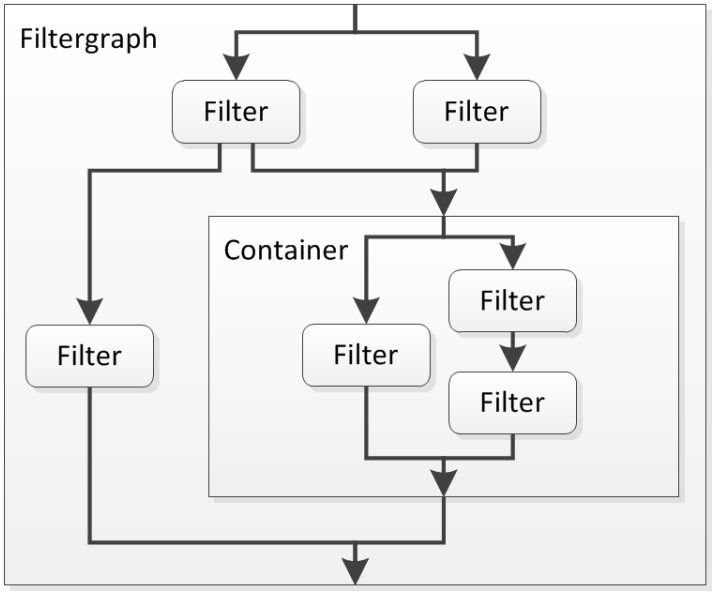

Abbildung 3.4: Beispiel eines Filtergraphen. Die Filter können beliebig viele Verbindungen besitzen und in Containern verschachtelt werden.

3.5.2 Fuzzy-Filter

Manchmal ist eine einfache Bewertung mit *ja* oder *nein* wenig befriedigend. Das macht sich vor allem bei der Kombination von Filtern bemerkbar. Werden diese z.B. seriell miteinander verbunden (logische Und-Verknüpfung), reicht das *Nein* eines einzigen Filters aus, um das Filterergebnis zu verwerfen. Dabei spielt keine Rolle, wie "knapp" die Entscheidung des Filters war. Dabei ist es in einigen Fällen jedoch relevant zu wissen, zu welchem Grad ein Feature den Erwartungen des Filters entspricht. Dafür gibt es einen zweiten Typ von Filter, den Fuzzy-Filter.

Ein Fuzzy-Filter bewertet mit Hilfe von Fuzzy-Logik (siehe Abschnitt 2.5 auf Seite 23) eine Feature mit einem reellen Wert zwischen 0 und 1. Da er keine Ja/Nein-Entscheidung trifft, kann er nicht direkt in einen Filtergraphen eingefügt werden, sondern benötigt einen speziellen Container. Dieser spezielle Fuzzy-Container kann nur Fuzzy-Filter aufnehmen

und fasst deren einzelne Ergebnisse zusammen. Damit der Fuzzy-Container von einem Filtergraphen aufgenommen werden kann, muss dieser sich äußerlich wie ein normaler Container verhalten und die reellen Filterwerte zuerst in boolesche Werte umwandeln, um diese nach außen weiterzugeben. Der Container benutzt hierfür einen einstellbaren Schwellenwert ("Threshold"). Fällt die Bewertung eines Features gleich oder höher aus als der eingestellte Schwellenwert, darf das Feature den Fuzzy-Container passieren, im anderen Fall wird sie verworfen.

Die Idee des Fuzzy-Containers ist die Bereitstellung einer Möglichkeit, die Bewertung einer Trajektorie durch mehrere Fuzzy-Filter vorzunehmen, *bevor* der Vergleich mit einem Schwellenwert stattfindet. Hierdurch können die Ergebnisse der einzelnen Filter vor dem Schwellenwert mit dem Fuzzy-Und- oder dem Fuzzy-Oder-Operator kombiniert werden, je nach dem ob sie seriell oder parallel angeordnet sind. Dadurch ist eine bessere Differenzierung zwischen den einzelnen Ergebnissen der Fuzzy-Filter möglich. Würde jeder Fuzzy-Filter einen eigenen Schwellenwert implementieren und nur boolesche Werte ausgeben, wäre dies nicht möglich.

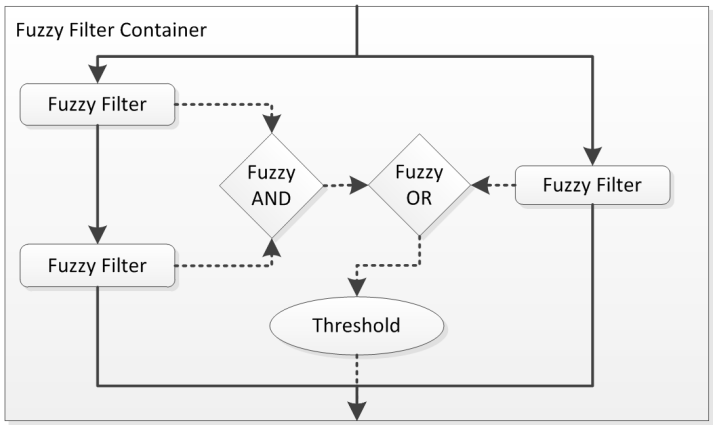

Abbildung 3.5: Fuzzy-Container mit Fuzzy-Filtern.

Abbildung 3.5 zeigt ein Beispiel eines Fuzzy-Containers mit Fuzzy-Filtern. Die durchgezogenen Pfeile bilden die tatsächlichen, sichtbaren Verbindungen zwischen den Filtern ab. Die gestrichelten Verbindungen stellen den Ablauf dar, wie die einzelnen Ergebnisse der Filter im Container miteinander kombiniert und welche Operatoren verwendet werden. Analog zu den normalen Filtern werden seriell geschaltete Fuzzy-Filter mit einem Fuzzy-Und-Operator, parallel geschaltete mit einem Fuzzy-Oder-Operator versehen. Letztendlich wird der resultierende Fuzzy-Wert des Containers durch eine Schwellenwert-Funktion in einen booleschen Filterwert umgewandelt, welcher das Filterergebnis des Fuzzy-Containers darstellt. Der Schwellenwert für die Umwandlung kann dabei über den Eigenschaftendialog des Containers festgelegt werden. Dabei werden alle Trajektorien durch den Fuzzy-Container durchgelassen, deren ermittelter Fuzzy-Wert größer oder gleich dem eingestellten Schwellenwert ist. Alle anderen Trajektorien werden herausgefiltert.

3.5.3 Relevanzen

Für jedes angeforderte Feature gibt es nicht nur einen Filtergraphen, sondern auch einen Relevanzgraphen. Mit diesem können die Features unabhängig von den Filtern in ihrer Relevanz für das Ergebnis eingestuft werden. Diese Relevanzinformationen können dann ebenfalls von den Views zur Gestaltung der Anzeige verwendet werden. Somit wird durch den Filtergraphen entschieden, welche Features angezeigt werden, der Relevanzgraph entscheidet über die Art der Darstellung der verbliebenen Features.

Die Funktionsweise des Relevanzgraphen ist zu der des Filtergraphen sehr ähnlich. Es können genauso beliebige Relevanzbausteine miteinander verbunden werden, Eine Verschachtelung in Containern ist ebenfalls möglich. Die Ausgabe eines Relevanzbausteins ist ein Fuzzy-Wert zwischen 0 und 1. Die Ergebnisse parallel bzw. seriell miteinander verbundenen Bausteine werden wie die Fuzzy-Filter mittels Fuzzy-Operatoren verknüpft. Der Fuzzy-Wert muss hier allerdings nicht mehr mittels eines Schwellenwertes in ein logisches Ergebnis konvertiert werden, sondern wird direkt als Relevanzwert dem Feature zugewiesen.

3.6 Views

Am Ende der Verarbeitungskette steht die visuelle Aufbereitung und Darstellung der Daten zur visuellen Analyse. Für die visuelle Darstellung der extrahierten und gefilterten Features sind die Views zuständig, die jeweils in einem eigenen Bereich die Daten beliebig darstellen können. Dabei können mehrere Views gleichzeitig mit Daten versorgt und angezeigt werden. Eine View kann mitteilen, welche Features sie für die Darstellung benötigt und bekommt diese dann (soweit verfügbar) durch die Feature-Extraktoren geliefert.

Abbildung 3.6 zeigt zwei geöffnete Views. Die linke View zeigt das laufende Videobild mit den extrahierten Trajektorien. Die zu den Trajektorien gehörenden Objekte sind mit einem Rahmen versehen. Rechts ist das laufende Video und seine Trajektorien durch ein *VideoPerpetuoGram* [BSE08] dargestellt. In beiden Fällen passiert die Visualisierung zur Abspielzeit des Videos, es wird also immer die gerade abgespielte Szene im Video gezeigt. Views haben aber auch die Möglichkeit, statischen Inhalt anzuzeigen, wie z.B. *ISS* [HHWH11a], die gesammelte Trajektorien aus einem ganzen Videoabschnitt unabhängig von der Abspielzeit zur Verfügung stellen.

Views können die Informationen nicht nur anzeigen. Sie haben auch die Möglichkeiten, dem Benutzer interaktive Funktionen zur weiteren Analyse zur Verfügung zu stellen. So könnten etwa Trajektorien selektiert und auf Wunsch weitere Details über die Trajektorien oder andere Features angezeigt werden. Entsprechend Shneidermans bzw. Keims Mantra (siehe Abschnitt 2.1 auf Seite 13) können die zur Verfügung gestellten Views im Visual-Analytics-Prozess nach dem Filterschritt (“filter and analyze further”) weitere Details auf Anfrage (“details on demand”) liefern.

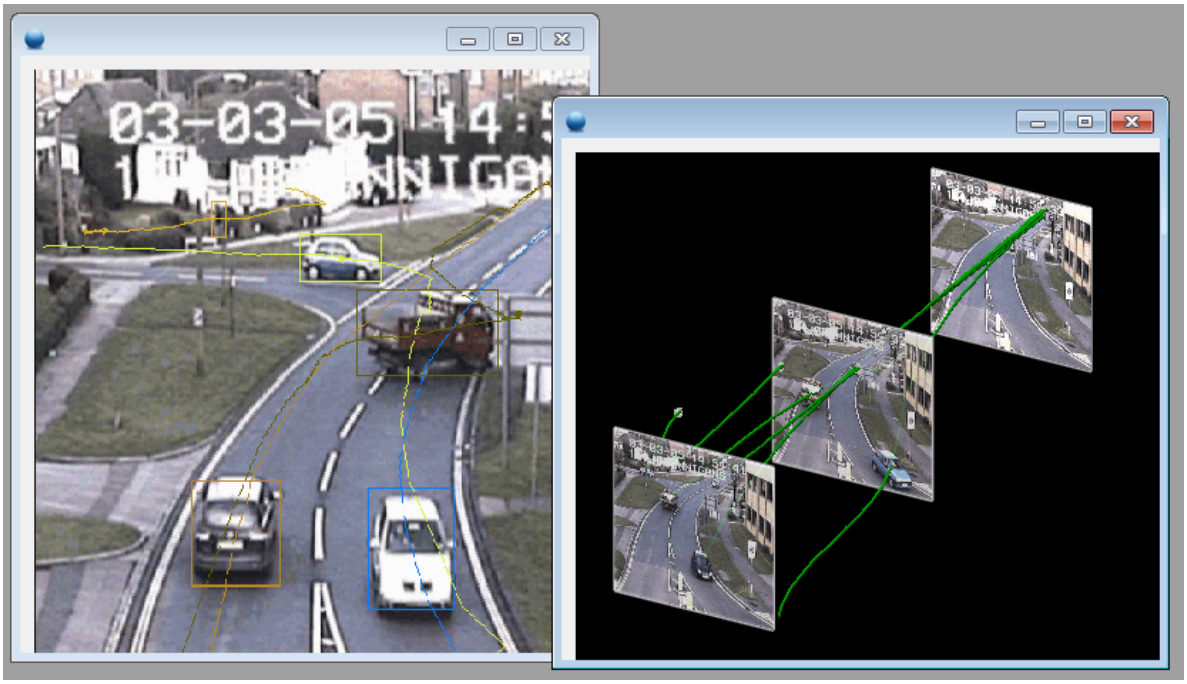

Abbildung 3.6: Die Features (hier Videobild und Trajektorien) lassen sich durch unterschiedliche Views grafisch darstellen.

3.7 Analyseprozess

Ein Prozess zur Analyse von Videos im *VAC* kann z.B. wie folgt aussehen. Dabei wird die Einbettung des Visual-Analytics-Mantra nach Keim (siehe Abschnitt 2.1 auf Seite 13) in den Prozess verdeutlicht. Zunächst werden durch das Abspielen des Videos die Features (z.B. Trajektorien) extrahiert und in den geöffneten Views angezeigt, wo der Benutzer sich ein erstes Bild über die Daten machen kann ("Analyze first") sowie über die Wahl der Views eine geeignete Darstellung wählt, um die interessanten Informationen anzuzeigen ("Show the important"). Durch Einfügen von Filtern kann die visualisierte Datenmenge auf den relevanten Teil reduziert werden ("Zoom, filter"). Über die Konfiguration der Filter kann das

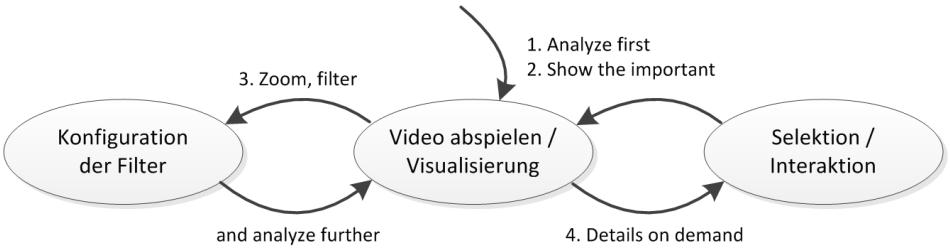

Abbildung 3.7: Visual-Analytics-Prozess im *VAC* nach Keims Mantra.

Filterergebnis angepasst werden. Durch weiteres Abspielen oder Neustarten des Videos kann nun die gefilterte Datenmenge analysiert werden (“Analyze further”). Die einzelnen Views sind dafür verantwortlich, dass dargestellte Daten (wie etwa Trajektorien) interaktiv weiter exploriert werden können (“Details on demand”), etwa durch Selektion und Anzeige weiter Details. Abbildung 3.7 zeigt die Komponenten des Beispielprozesses und die entsprechenden Schritte nach Keims Mantra.

Kapitel 4

Skizzieren einer Query-Trajektorie

Das Ziel des Query-by-Sketch-Ansatzes [BPB+10] ist die Modellierung einer Trajektorie, die mit den Trajektorien aus einer Datenquelle verglichen werden kann. Dieses Kapitel stellt die Benutzeroberfläche zum Zeichnen von Trajektorien vor, die in dieser Arbeit entwickelt und in das *VAC* integriert wurde. Diese Oberfläche wird vom skizzenbasierten Filter (siehe Kapitel 5 auf Seite 47) zum Erstellen der Query-Trajektorie verwendet.

4.1 Oberfläche zum Skizzieren von Trajektorien

Die Query-Trajektorie kann mittels eines umfangreichen Benutzerdialogs skizziert werden. Der Skizzierungsvorgang erfolgt ähnlich wie im vorgestellten System *VideoQ* (siehe Abschnitt 2.4.1 auf Seite 19). Neu ist, dass die Charakteristik des Überwachungsvideos (starrer Blickwinkel) es leicht ermöglicht, die Trajektorie direkt im Kontext des Videobildes zu zeichnen. Gleichzeitig kann die gezeichnete Trajektorie dem Blickwinkel entsprechend perspektivisch dargestellt werden, was einen besseren Tiefeneindruck zur Folge hat. Somit ergibt sich eine recht genaue Einschätzung der Position und Lage der gezeichneten Trajektorie innerhalb des Kamerablickwinkels.

Abbildung 4.1 zeigt die Benutzeroberfläche, in der die Skizze einer Query-Trajektorie erstellt werden kann. Im Zeichenbereich können durch direkte Manipulation per Drag & Drop die Stützpunkte (“Samples”) der Trajektorienskizze hinzugefügt oder räumlich verschoben werden. Im rechten Teil werden weitere Eigenschaften der Skizze angezeigt und können dort verändert werden. Dies sind hauptsächlich zeitliche Eigenschaften der Skizze. Dort kann aber auch der Name der Trajektorie eingestellt und (falls mehrere Trajektorien gleichzeitig skizziert werden) die aktive Skizze ausgewählt werden. Zusätzlich besteht die Möglichkeit, zur besseren Übersicht die Größe der Skizzendarstellung zu verändern. Zu diesem Zweck kann der Radius der visuellen Darstellung der Samples eingestellt werden, die Breite der Verbindungssegmente zwischen den Punkten wird entsprechend angepasst. Veränderungen an der visuellen Darstellung haben keine Auswirkung auf die räumlichen oder zeitlichen Eigenschaften der Skizze.

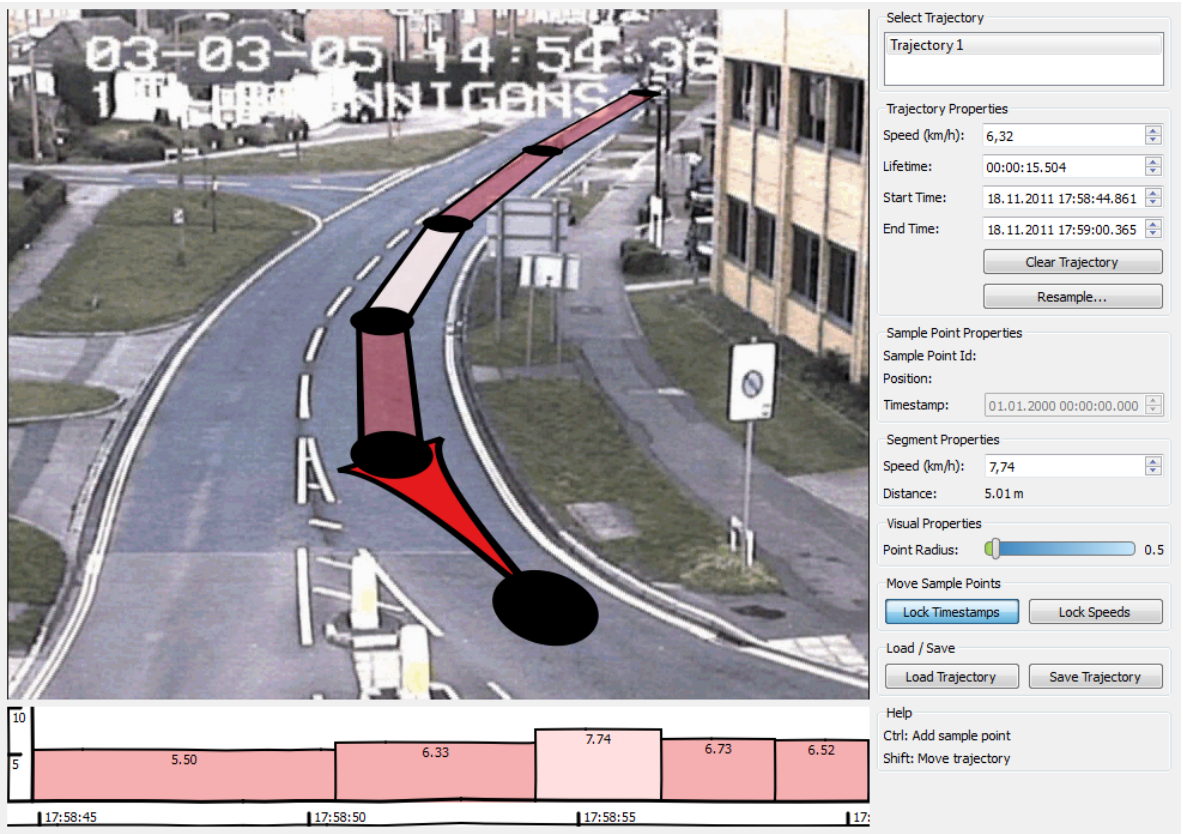

Abbildung 4.1: Benutzeroberfläche zum Skizzieren von Query-Trajektorien: Der Zeichenbereich mit Hintergrund des Videos, darunter eine Übersicht der Geschwindigkeiten in den einzelnen Segmenten. Rechts können vor allem die zeitlichen Eigenschaften der Trajektorie verändert werden werden.

Unter dem Zeichenbereich werden die verschiedenen Geschwindigkeiten in den einzelnen Trajektoriensegmenten zwischen den Samples auf einer Zeitachse dargestellt. Damit ist der Geschwindigkeitsverlauf der Trajektorie sowie die zeitlichen Abstände zwischen den Samples leicht zu erkennen. Die Höhe der Balken gibt die Geschwindigkeit im jeweiligen Segment an.

Das Erstellen der Trajektorie geschieht mittels Festlegen von Samples. Jeder der definierten Samples wird sowohl räumlich (Position) als auch zeitlich (absoluter Zeitpunkt) in den Kontext des Videos eingeordnet. Die Trajektorie beschreibt somit eine räumlich und zeitlich definierte Bewegung innerhalb des erfassten Blickwinkels. Die Positionen der Samples werden dabei mit Hilfe der zu Verfügung gestellten Homographie (siehe Abschnitt 3.1.2 auf Seite 27) von der Zeichenebene (Längeneinheit: Pixel) perspektivisch korrekt in den metrischen Raum der von der Kamera erfassten Szene transformiert. Die transformierten Positionsdaten werden als Koordinaten in der Trajektorie gespeichert. Somit können auch

Geschwindigkeiten und Zeitpunkte dem Blickwinkel entsprechend korrekt berechnet werden, was später für einen korrekten Vergleich mit Trajektorien aus dem Video auf Grund der perspektivischen Verzerrung notwendig ist.

Sowohl die räumlichen als auch die zeitlichen Eigenschaften der Trajektorie können über die Oberfläche modelliert werden. Im Folgenden werden diese Möglichkeiten näher beschrieben. Allerdings lassen sich nicht alle Parameter nicht beliebig genau einstellen. Die kleinste einstellbare Zeiteinheit beträgt eine Millisekunde. Auf Grund dieser Einschränkung sind zeitabhängige Parameter wie die Geschwindigkeit zwischen den einzelnen Punkten ebenfalls nicht beliebig genau, was zu leichten Abweichungen in der Umrechnung zwischen Abständen, Zeit und Geschwindigkeit führt. Jedoch sind die Abweichungen zu gering, um eine feststellbare Auswirkung auf das spätere Filterergebnis zu haben.

4.2 Darstellung der Trajektorie

Die Darstellung der skizzierten Trajektorie ähnelt der Darstellung, die in *ISS* [HHWH11a] verwendet wird. Die Trajektorie wird durch einen breiten Pfeil repräsentiert, die Pfeilspitze gibt die Bewegungsrichtung an. Um die räumliche Einordnung der Trajektorie in den Videokontext zu erleichtern, wird einerseits das Videobild im Hintergrund angezeigt, zum anderen trägt die perspektivische Darstellung dazu bei, dass der Benutzer Entfernungen im Video besser erfassen kann.

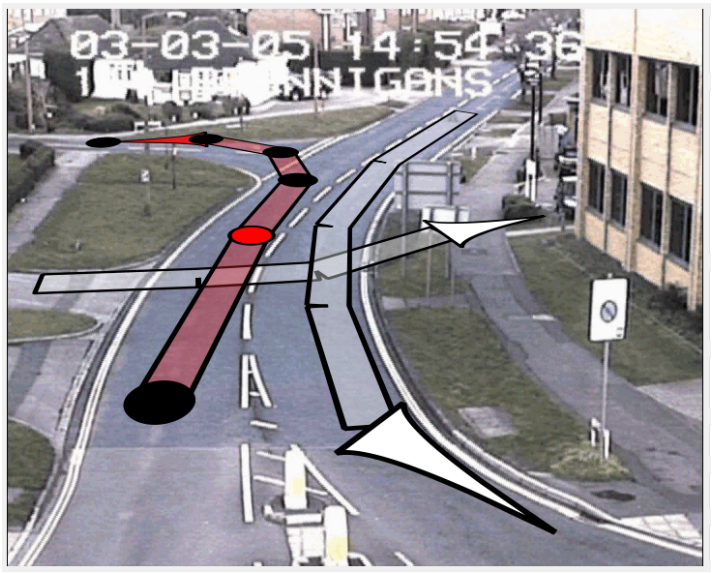

Abbildung 4.2: Oberfläche für das Skizzieren von Trajektorien. Die aktive Skizze kann bearbeitet werden, die inaktiven Skizzen sind gesperrt und werden in grau dargestellt.

Um die Bearbeitung der Trajektorie zu ermöglichen, werden ihre Samples durch gefüllte (projizierte) Kreise visualisiert, an denen das jeweilige Sample mit der Maus verschoben werden kann. Die Größe verändert sich entsprechend der Perspektive je nach Entfernung des Punktes zur Kamera. Um die Skizze übersichtlich zu halten, wird die Trajektorie transparent dargestellt, damit der Hintergrund darunter erkennbar bleibt. Die Zeichenoberfläche unterstützt prinzipiell das gleichzeitige Erstellen von mehreren Trajektorienskizzen. Die Skizzen können einzeln ausgewählt werden, dabei kann die gerade aktive Skizze normal bearbeitet werden, die inaktiven Skizzen werden grau im Hintergrund dargestellt und sind vor unabsichtlichen Veränderungen geschützt. Abhängig von dem Kontext, in dem die Zeichenoberfläche verwendet wird, kann festgelegt werden, wie viele Trajektorien der Benutzer skizzieren muss bzw. darf. In Abbildung 4.2 ist die Zeichenoberfläche mit drei Skizzen zu sehen, wovon die aktive rot und mit den verschiebbaren Samples angezeigt wird, während die inaktiven Skizzen im Hintergrund grau dargestellt werden.

4.3 Räumliche Modellierung

Die räumliche Modellierung erfolgt durch direkte Manipulation der Samples (schwarze Punkte) auf der Zeichenfläche. Durch einen Klick mit der Maus auf eine leere Stelle der Zeichenfläche wird ein Sample an das Ende der Trajektorie hinzugefügt. Für dieses wird der Zeitpunkt so festgelegt, dass die Geschwindigkeit zwischen diesem und dem vorhergehenden Sample der Durchschnittsgeschwindigkeit der Trajektorie entspricht. Bereits bestehende Samples können durch einfaches Ziehen mit der Maus verschoben werden. Zudem gibt des die Möglichkeit, die ganze Trajektorie mit allen Samples gleichmäßig zu verschieben.

Für das Verschieben eines einzelnen Samples stehen zwei verschiedene Modi zur Verfügung, die die Zeitstempel der Samples in der Skizze beeinflussen:

“Lock Timestamps”: Der Zeitstempel des markierten Samples sowie der aller anderen bleibt beim Verschieben unangetastet. Die Geschwindigkeit der Segmente direkt vor bzw. hinter dem verschobenen Sample ändert sich auf Grund der veränderten räumlichen Distanzen zwischen den Punkten.

“Lock Speeds”: Die Geschwindigkeiten der Segmente zwischen den Samples bleiben konstant. Dementsprechend werden die Zeitstempel der bearbeiteten und aller nachfolgenden Samples so angepasst, dass trotz der veränderten Entfernungen zwischen den Samples (Längen der Segmente) die Geschwindigkeit gleich bleibt.

Durch diese beiden Modi kann verhindert werden, dass beim räumlichen Verschieben der einzelnen Samples unabsichtlich zeitliche Eigenschaften verändert werden, die evtl. bereits modelliert wurden. So kann entweder die zeitliche Einordnung der Samples oder die Geschwindigkeiten dazwischen vor Veränderungen bei der räumlichen Bearbeitung geschützt werden.

4.4 Zeitliche Modellierung

Die zeitlichen Eigenschaften der Trajektorie können über Steuerelemente neben der Zeichenfläche eingestellt werden. Dabei wird zwischen Eigenschaften von Samples, Segmenten (den Abschnitten zwischen zwei Samples) und Eigenschaften, die die gesamte Trajektorie betreffen, unterschieden.

Eigenschaften eines Samples

Ein Sample hat neben seinen räumlichen Koordinaten nur eine einzige zeitliche Eigenschaft: den absoluten Zeitstempel. Der Zeitstempel des markierten Samples kann über ein Zeitauswahl-Steuerelement eingestellt werden. Wird der Zeitstempel dabei über den des vorhergehenden oder nachfolgenden Samples hinaus verschoben, ändert sich die Reihenfolge der Samples in der Skizze, so dass die zeitliche Abfolge korrekt bleibt.

Eigenschaften eines Segments

Ein Segment beschreibt die Beziehung zwischen zwei zeitlich benachbarten Samples. Hier lässt sich die Geschwindigkeit zwischen den beiden Stützpunkten einstellen. Die Zeitstempel aller nachfolgenden Samples werden so angepasst, dass das markierte Segment die eingestellte Geschwindigkeit hat, aber die Geschwindigkeiten der nachfolgenden Segmente nicht verändert werden. Die Geschwindigkeit zwischen zwei Samples ist keine direkt definierte Eigenschaft der Trajektorie, da sie von der Entfernung und der zeitlichen Differenz zwischen den Samples abhängt und auch über diese modelliert wird.

Zur besseren Übersicht über die Geschwindigkeiten der einzelnen Segmente werden diese in einem kleinen Balkendiagramm unter der Zeichenfläche angezeigt. Dabei werden die Segmente in eine Zeitachse eingeordnet, vertikale Linien repräsentieren die Zeitstempel der Samples. An den Höhen der Balken ist in jedem Segment die Geschwindigkeit in Kilometern pro Stunde abzulesen, die im Balken zusätzlich als Zahlenwert dargestellt wird. Abbildung 4.3 zeigt das Balkendiagramm. Per Mausklick auf einen Balken kann – wie auch in der Skizze – ein Segment ausgewählt werden, dessen Geschwindigkeit dann über ein Steuerelement bearbeitet werden kann.

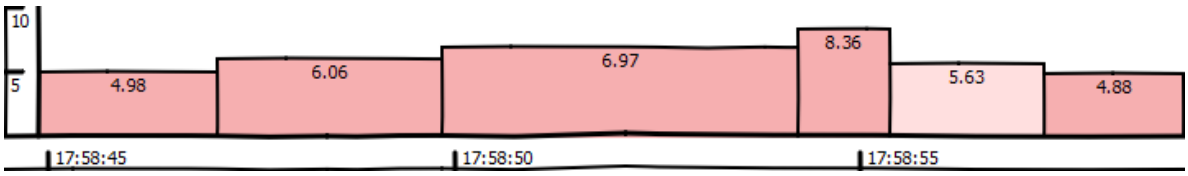

Abbildung 4.3: Balkendiagramm zur Übersicht über die Geschwindigkeiten in den einzelnen Segmenten. Das ausgewählte Segment wird farblich hervorgehoben.

Eigenschaften der Trajektorie

Zur Vereinfachung der Bearbeitung der Zeitstempel lassen sich diese auch indirekt über globale Eigenschaften der Trajektorie einstellen. Die folgenden vier Eigenschaften lassen sich bearbeiten:

"Speed" Hier kann die Geschwindigkeit aller Segmente der Trajektorie eingestellt werden. Sämtliche Zeitstempel werden (ausgehend vom Startzeitpunkt der Trajektorie) so angepasst, dass die Geschwindigkeiten zwischen den Stützstellen dem eingestellten Wert entsprechen.

"Lifetime" Diese Eigenschaft definiert die "Lebenszeit" der Trajektorie, also die Dauer zwischen dem ersten und dem letzten Zeitstempel. Wird dieser Wert verändert, werden die Zeitstempel ausgehend vom Anfang der Trajektorie gleichmäßig so skaliert, dass die Lebenszeit der Trajektorie dem eingestellten Wert entspricht.

"Start Time" Dieser Wert entspricht dem Zeitstempel des ersten Samples. Die Trajektorie wird dabei entsprechend der Veränderung des Zeitstempels komplett entlang der Zeitachse verschoben, ohne ihre Dauer zu verändern.

"End Time" Diese Eigenschaft verhält sich genauso wie "Start Time", nur dass hier die Endzeit der Trajektorie eingestellt werden kann. Die Dauer der Trajektorie bleibt dabei ebenfalls unverändert.

Einige Trajektorien-Videoanalysesysteme wie z.B. das von Calderara et al. [CPC09] stellen neben der Geschwindigkeit auch die Modellierung von Beschleunigung innerhalb der Trajektorien zur Verfügung. Um die Modellierung möglichst einfach zu halten, wird hier die Beschleunigung nicht als explizite Eingabemöglichkeit verwendet. Sie kann jedoch durch geeignetes Festlegen der Geschwindigkeiten zwischen den Samples angenähert werden.

4.5 Resampling

Die Resampling-Funktion im Skizzendialog dient in erster Linie zur Bequemlichkeit, die es dem Benutzer ermöglicht, schnell und unkompliziert die Anzahl der Samples der Trajektorie zu verändern. Dies geschieht ohne Auswirkung auf die Gesamtdauer der Trajektorie, die Zeitstempel des ersten und des letzten Samples bleiben unverändert. Die Trajektorie wird mit neuen Samples vollständig neu aufgebaut. Die Position der neuen Samples wird dabei so berechnet, dass diese auf den linear interpolierten Strecken zwischen den alten Samples liegen. Für das Resampling gibt es zwei Möglichkeiten:

Anzahlbasiertes Resampling Hier wird die Trajektorie so in zeitlich gleiche Intervalle aufgeteilt, dass die Anzahl der Samples der angegebenen Anzahl entspricht. Dabei wird bei der Berechnung der Zeitintervalle auf eine Millisekunde gerundet, da dies die kleinste einstellbare Zeiteinheit ist.

Intervallbasiertes Resampling In diesem Fall wird das zeitliche Intervall zwischen den Samples festgelegt. Falls die Dauer der Trajektorie nicht durch das eingestellte Intervall teilbar ist, ist das letzte Intervall entsprechend kürzer, so dass die Dauer der Trajektorie nicht verändert wird.

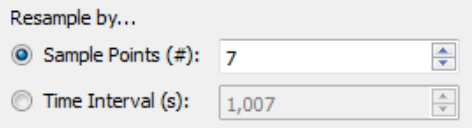

Abbildung 4.4: Einstellmöglichkeiten für das Resampling der skizzierten Trajektorie.

Durch das Verändern der Anzahl der Samples bzw. ihrer Position wird in erster Linie der räumliche Verlauf der Trajektorie verändert. Durch das Verschieben der Samples auf der ursprünglichen Trajektorienstrecke kommt es durch die lineare Interpolation meist zu einer zur Veränderung der Strecken zwischen den Samples, die unproportional zur veränderten Dauer ist. Somit verändert sich beim Resampling mit hoher Wahrscheinlichkeit die zurückgelegte Strecke der Trajektorie und – auf Grund der gleichbleibenden Dauer – auch die Geschwindigkeit zwischen den Samples sowie die Durchschnittsgeschwindigkeit der Trajektorie.

Die Möglichkeit des Resampling ist vor allem notwendig, wenn eine aus dem Video extrahierte Trajektorie ("Query by Example") Grundlage für das Erstellen eines skizzenbasierten Querys ist (siehe Abschnitt 5.1 auf Seite 47). Solch eine Trajektorie kann ohne Weiteres in eine Skizzentrajektorie umgewandelt und als solche bearbeitet werden. Allerdings ist diese durch die viel zu große (weil frameratebasierte) Anzahl an Samples nicht zum Bearbeiten als Skizze geeignet, weswegen die Anzahl der Samples erst durch den Resampling-Schritt reduziert werden muss. Dies ist an dieser Stelle dem Benutzer überlassen, damit er sowohl auf die Resampling-Methode als auch auf die Anzahl der Samples Einfluss nehmen kann.

Der gleiche Resampling-Algorithmus wird verwendet, um die Anzahl der Samples der skizzierten Trajektorie wieder der Framerate des Videos anzugleichen, was für die Verwendung der Skizzentrajektorie (siehe Abschnitt 4.7 auf der nächsten Seite) benötigt wird.

4.6 Speichern von skizzierten Trajektorien

Eine skizzierte Trajektorie lässt sich innerhalb des Skizzendialogs zur späteren Wiederverwendung in eine Datei abspeichern bzw. aus einer Datei laden. Von der Trajektorie werden die Samples mit ihren Koordinaten und Zeitstempeln abgespeichert. Das Format der gespeicherten Trajektorie entspricht dem der im *VAC* eingesetzten Trajektorien mit sog. Beobachtungen (siehe Abschnitt 3.1.1 auf Seite 27). Die zweidimensionalen Koordinaten jedes Samples beziehen sich auf die (unprojizierte) Bildebene. Das Dateiformat der gespeicherten

skizzierten Trajektorie unterscheidet sich nicht von dem extrahierter Videotrajektorien. Der einzige Unterschied liegt in der unterschiedlichen Anzahl an Samples.

Da sich der Zeichenkontext (Hintergrund, Homographie, etc.) mit jedem Video ändert, passt die mit einer Trajektorie mit abgespeicherte Homographie nicht unbedingt zum Video, sofern die Trajektorie im Kontext eines anderen Videos gezeichnet wurde. Beim Laden einer skizzierten Trajektorie aus einer Datei wird daher die mitgelieferte Homographie ignoriert. Die geladene Trajektorie, behält zwar ihre Koordinaten, in einem anderen Videokontext kann es jedoch sein, dass auf Grund der veränderten Homographie die Trajektorie verschoben oder verzerrt wird.

4.7 Aufbau und Verwendung einer skizzierten Trajektorie

Der strukturelle Aufbau einer skizzierten Trajektorie unterscheidet sich von dem einer Videotrajektorie (siehe Abschnitt 3.1.1 auf Seite 27). Primäre Verwendung einer Skizzentrajektorie ist das ständige Verändern der Trajektorieneigenschaften. Diese werden dem polygonalen Ansatz [BPB+10] entsprechend einzig durch die Samples der Trajektorie repräsentiert, die jeweils einen (innerhalb der Trajektorie eindeutigen) Zeitstempel und eine Positionsangabe enthalten. Alle anderen Eigenschaften werden aus den Werten der Samples berechnet.

Die Positionsangaben der Samples sind in Metern angegeben und beziehen sich auf die projizierte Szene, welche durch die Homographie festgelegt ist. Während die Struktur von Videotrajektorien potenziell für jedes Sample eine eigene Homographie ermöglicht, ist die Homographie beim Skizzieren durch die Szene festgelegt und für die gesamte skizzierte Trajektorie konstant.

Die Reihenfolge der einzelnen Samples in der Trajektorie ist durch die Zeitstempel gegeben. Um auch bei zeitlichen Veränderungen die richtige Reihenfolge der Samples und die Eindeutigkeit der Zeitstempel zu gewährleisten, wird ein Verzeichnis eingesetzt, das für jeden verfügbaren Zeitstempel den dazugehörigen positionalen Stützpunkt speichert. Ändert sich der Zeitstempel eines Samples, wird dieses in der Reihenfolge neu einsortiert.

Aufbereitung der Skizze für die Verwendung im Filter

Bevor die erstellte Skizze als Query-Trajektorie in einem Filter eingesetzt werden kann, muss sie erst in das Videotrajektorien-Format umgewandelt werden. Die Vergleichsalgorithmen müssen so nur ein Format verstehen (das der Videotrajektorien), was sie flexibel im Einsatz macht. So können mit dem selben Algorithmus auch zwei aus dem Video extrahierte Trajektorien miteinander verglichen werden.

Für die Umwandlung einer Skizze in das Videotrajektorien-Format sind zwei Schritte notwendig:

1. **Resampling**
   Die Anzahl der Samples muss so angepasst werden, dass die zeitlichen Abstände

zwischen ihnen den Samples der zu vergleichenden Videotrajektorien entsprechen. Die zeitlichen Abstände sind von der Framerate des Videos abhängig, aus dem die Trajektorien extrahiert wurden. Dies ist notwendig, damit bei Überdeckungsalgorithmen (siehe Abschnitt 5.2.1 auf Seite 50) die einzelnen Samples korrekt miteinander verglichen werden können [HXF+07]. Dieser Schritt ist identisch zum manuellen Resampling beim Skizzieren der Trajektorie (Abschnitt 4.5 auf Seite 42), nur dass der zeitliche Abstand zwischen den einzelnen Samples in diesem Fall fest durch die Framerate vorgegeben ist.

2. **Konvertierung in eine Videotrajektorie**
   Die neu gesampelte Skizzentrajektorie wird nun in eine Videotrajektorie umgewandelt. Hierzu werden die Koordinaten der Samples über die zum skizzieren verwendete Homographie in Bildschirmkoordinaten (Pixel) rücktransformiert und zusammen mit ihrem Zeitstempel und der Homographie in eine Beobachtung gespeichert. Die erzeugten Beobachtungen ergeben zusammen die Videotrajektorie.

Die in den beiden Schritten umgewandelte Trajektorienskizze kann so als Query-Trajektorie verwendet werden und ist für den Einsatz in Trajektorienfiltern zum Vergleich mit Videotrajektorien geeignet.

Kapitel 5

Filtern anhand einer skizzierten Trajektorie

Für das *VAC* soll ein Filter bereitgestellt werden, der ein gezieltes Filtern von Trajektorien anhand ihrer räumlichen und zeitlichen Eigenschaften ermöglicht. Die Definition der gesuchten Eigenschaften geschieht dabei durch die Modellierung einer Trajektorienskizze ("Query by Sketch"), durch die der Benutzer die Eigenschaften der gesuchten Trajektorien schnell und intuitiv beschreiben kann. Durch Vergleichen der zu filternden Trajektorien mit der Skizze wird die Ähnlichkeit zwischen diesen berechnet, welche das Filterergebnis als Fuzzy-Wert darstellt. Der skizzenbasierte Trajektorienfilter ist somit ein Fuzzy-Filter (siehe Abschnitt 3.5.2 auf Seite 32).

Für den Entwurf der Skizze wird die Benutzeroberfläche zum Skizzieren einer Trajektorie (Kapitel 4 auf Seite 37) verwendet. Neben der Skizzierungsmöglichkeit stehen umfangreiche Einstellungsmöglichkeiten zur Verfügung, um die verschiedenen modellierten Eigenschaften der Trajektorie für den Vergleich mit anderen Trajektorien zu aktivieren und anhand ihrer Bedeutung zu gewichten. Die Vergleichsergebnisse werden dabei mittels Fuzzy-Logik kombiniert.

Aus der Skizze wird für den Vergleich mit anderen Trajektorien eine eigene Trajektorie erstellt. Diese wird auf Grund ihrer Verwendung als Referenz für den Vergleich mit anderen Trajektorien im Filter im Folgenden als Query-Trajektorie bezeichnet. Abschnitt 5.1 befasst sich mit dem Erstellen dieser Query-Trajektorie für den Filter. In Abschnitt 5.2 werden Funktionen zum Vergleichen der Trajektorien vorgestellt, Abschnitt 5.3 beschreibt detailliert die einzelnen Merkmale des Filters sowie seine Funktionsweise.

5.1 Erstellen einer Query-Trajektorie

Der Filter soll mittels einer skizzierten Query-Trajektorie die ankommenden Video-Trajektorien filtern. Dafür kann im Eigenschaftendialog des Filters die Zeichenoberfläche

zum Skizzieren der Trajektorie geöffnet werden. Die Zeichenoberfläche ist in Kapitel 4 auf Seite 37 beschrieben. Die fertig skizzierte Trajektorie wird dem Filter zur Verfügung gestellt, der zur besseren Übersicht einige signifikante Informationen der Trajektorie in seinem Eigenschaftendialog anzeigt. Abbildung 5.1 stellt die Informationsanzeige mit den Daten der skizzierten Trajektorie dar. Die Trajektorie ist hier bereits dem Resampling unterzogen, das die Anzahl der Samples an die Framerate des aktuellen Videos anpasst (siehe Abschnitt 4.5 auf Seite 42). So wird in den Informationen eine wesentlich größere Anzahl an Samples angezeigt als in der Skizze vorhanden sind. Über den Button *Sketch* kann die skizzierte Trajektorie dennoch in ihrer ursprünglichen Form weiter bearbeitet werden und wird danach wieder entsprechend für den Filter aufbereitet.

Abbildung 5.1: Informationen über die skizzierte Trajektorie im Eigenschaftendialog des Filters. Die Eigenschaft *Source* gibt an, aus welcher Quelle die Trajektorie stammt (Skizze, Datei oder Video).

Als Alternative zum Skizzieren der Query-Trajektorie stehen zwei weitere Optionen zur Verfügung, die über den Button *Load...* aufgerufen werden können. Die erste Möglichkeit verfolgt den Query-by-Example-Ansatz und stellt in einem Auswahldialog Trajektorien aus dem aktuellen Video zur Verfügung. Die Auswahl ist dabei auf einen Zeitraum im zuletzt abgespielten Videoteil beschränkt. Dies ist notwendig, damit auch bei theoretischer unendlicher Laufzeit des Videos (wie etwa bei Videoströmen) sich der Speicherbedarf der Trajektorien in Grenzen hält. Im Auswahldialog kann eine von den angezeigten Trajektorien ausgewählt werden und wird dann im Eigenschaftendialog des Filters (Abbildung 5.1) angezeigt. Im Gegensatz zum Query-by-Sketch entspricht eine aus dem Video ausgewählte Trajektorie verständlicherweise bereits der Framerate des Videos und muss daher keinem Resampling unterzogen werden. In Abbildung 5.2 ist der Auswahldialog zu sehen, in dem im Kontext des Videohintergrundes eine Trajektorie ausgewählt werden kann.

Die zweite Option bietet die Möglichkeit, die Query-Trajektorie aus einer Datei zu laden. Somit können auch Trajektorien verwendet werden, die bereits zur späteren Verwendung in einer Datei abgespeichert wurden. Die Trajektorien können dabei auch aus einem anderen Video oder Datenquelle stammen. Dabei muss allerdings beachtet werden, dass die Trajektorie in einen neuen Kontext gesetzt wird und somit innerhalb des neuen Videos veränderte Eigenschaften aufweisen kann. So können z.B. die von der Homographie des Videos abhängigen Positionswerte beim Wechseln des Videokontextes auf eine andere Stelle im Videobild verweisen, oder auch außerhalb des Videobildes liegen.

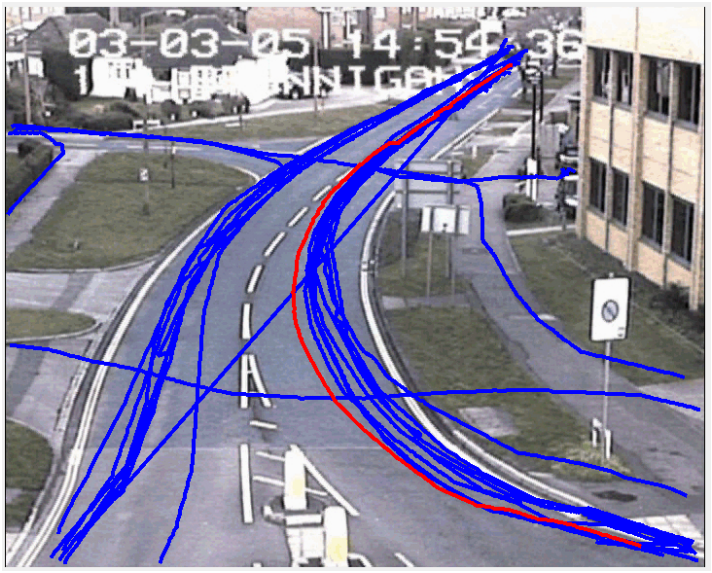

Abbildung 5.2: Dialog zum Auswählen einer Query-Trajektorie aus dem aktuellen Video ("Query by Example"). Die ausgewählte Trajektorie wird als Query-Trajektorie im Filter verwendet.

5.2 Vergleichen von Trajektorien

Das Filtern einer Trajektorienmenge anhand einer skizzierten Trajektorie erfordert das Vergleichen bestimmter Merkmale der zu filternden Trajektorien mit der skizzierten Trajektorie. Ziel des Filterns ist, Trajektorien aus der Menge zu Finden, die in gewissen Eigenschaften zur skizzierten Trajektorie ähnlich sind. Diese Eigenschaften sind in den zur Verfügung stehenden Merkmalen definiert. Demnach hat jede Trajektorie T bezüglich eines Merkmals ("Feature") $\mathcal{F}$ einen oder mehrere (innerhalb des Videokontextes absolute) Werte. Voraussetzung für den Vergleich von Trajektorien bezüglich eines Merkmals ist, dass sich die beiden Werte des Merkmals vergleichen lassen und der euklidische Abstand (Distanz) der Werte zueinander bestimmbar ist. In diesem Fall ist der Wert jedes Merkmals eine reeller skalarer Wert (Richtung, Geschwindigkeit, Zeit) oder ein reeller zweidimensionaler Vektor (Position in der Ebene).

Beim Vergleich zweier Werte bzw. Vektoren lässt sich zwischen Distanz und Ähnlichkeit unterscheiden, wobei beide voneinander abhängig sind. Der Begriff der Ähnlichkeit ("Similarity") $A_{\mathcal{F}}$ definiert sich über die euklidische Distanz $D_{\mathcal{F}}$ zwischen zwei Trajektorien T und T' bezüglich des Merkmals $\mathcal{F}$. Ist $D_{\mathcal{F}} = 0$, ist die Ähnlichkeit $A_{\mathcal{F}}$ maximal. Mit zunehmender Distanz nimmt die Ähnlichkeit $A_{\mathcal{F}}$ immer mehr ab. Dabei ist der Verlauf der Ähnlichkeitsfunktion mit zunehmender Distanz nicht eindeutig festgelegt. Die Abbildung der Distanz auf die Ähnlichkeit erfolgt mittels einer Ähnlichkeitsfunktion (siehe Abschnitt 5.2.2 auf Seite 51).

5.2.1 Vergleichsalgorithmen

Höferlin et al. [HHWH12] schlägt drei verschiedene Ähnlichkeitsmaße ("Similarity Measures") zum Vergleich von Trajektorien bezüglich bestimmter Merkmale vor: *Überdeckung*, *Distanz zwischen Mittelwerten* und *Distanz zwischen Standardabweichungen*. Dort werden diese Algorithmen verwendet, um ähnliche Trajektorien zu einem Cluster zusammenzufassen. Da beim Filtern von Trajektorien auch die Ähnlichkeit der Trajektorien zueinander ausschlaggebend ist, werden diese drei Maße hier für den Filter wiederverwendet.

Überdeckung

Das Ähnlichkeitsmaß *Überdeckung* ("Coverage") D_c basiert auf einem sampleabhängigen Vergleichsverfahren. Es berechnet den Grad der Überdeckung der zweier Trajektorien T und T' mit den Samples t_i und t_j innerhalb eines Merkmals $\mathcal{F}$:

$$D_c\left(T, T', \epsilon\right) = \frac{1}{2}\left(\sum_{t_i \in T} \frac{c\left(t_i, T'\right)}{|T|} + \sum_{t_j \in T'} \frac{c\left(t_j, T\right)}{|T'|}\right)$$

mit

$$c\left(t, T\right) = \begin{cases} 1, & \min\limits_{t_k \in T}\left(d\left(\mathcal{F}\left(t\right), \mathcal{F}\left(t_k\right)\right)\right) < \epsilon \\ 0, & \text{sonst} \end{cases}$$

wobei ϵ angibt, wie weit zwei Samples voneinander entfernt sein dürfen, damit sie noch als überdeckt gelten. $\mathcal{F}\left(t\right)$ ist hierbei der (ein- oder zweidimensionale) Wert des Merkmals $\mathcal{F}$ am Sample t und $d\left(\cdot, \cdot\right)$ die euklidische Distanz zwischen zwei (skalaren oder vektoriellen) Werten.

Da bei der Überdeckung die einzelnen Samples zweier Trajektorien verglichen werden, ist es von Bedeutung, dass die beiden Trajektorien annähernd gleiche Abstände zwischen ihren Samples besitzen. Insbesondere skizzierte Trajektorien, die im Normalfall weit weniger Samples besitzen als Videotrajektorien, müssen für diesen Zweck einem Resampling (erneute Berechnung unter Veränderung der Anzahl der Samples) unterzogen werden, sodass die zeitlichen Abstände der Samples denen der Videotrajektorie entsprechen (siehe Abschnitt 4.7 auf Seite 44).

Distanz zwischen Mittelwerten

Die Berechnung der *Distanz zwischen Mittelwerten* hängt von der einstellbaren Granularität κ ab. Hierbei werden die Trajektorien in κ zeitlich gleichlange Segmente zerteilt. Ein Segment ist somit eine Untermenge der Samples einer Trajektorie. Falls die Teilung nicht genau auf einem Sample der Trajektorie erfolgt, wird zwischen den beiden umschließenden Samples linear interpoliert. Die Segmentierungsfunktion $\mathcal{S}(T, \kappa, k)$ definiert die Menge der Samples

im k-ten von κ Segmenten der Trajektorie T. Mit $S_k = \mathcal{S}(T, \kappa, k)$ und $S'_k = \mathcal{S}(T', \kappa, k)$ wird die Distanz zwischen den Mittelwerten berechnet:

$$D_\mu\left(T, T', \kappa\right) = \frac{1}{\kappa} \sum_{k=1}^{\kappa} d\left(E\left(S_k\right), E\left(S'_k\right)\right)$$

mit

$$E\left(S\right) = \frac{1}{|S|} \sum_{t_i \in S} \mathcal{F}\left(t_i\right)$$

wobei $E\left(S\right)$ das arithmetische Mittel des Segments S berechnet.

Distanz zwischen Standardabweichungen

Bei ermitteln der *Distanz zwischen Standardabweichungen* werden die Trajektorien ebenfalls nach der Granularität in κ Segmente unterteilt. Die geschieht in gleicher Weise wie bei der Berechnung der Mittelwerte. Die Distanz zwischen den Standardabweichungen berechnet sich wie folgt:

$$D_\sigma(T, T', \kappa) = \frac{1}{\kappa} \sum_{k=1}^{\kappa} d\left(\sqrt{\mathrm{Var}\left(S_k\right)}, \sqrt{\mathrm{Var}\left(S'_k\right)}\right)$$

mit

$$\mathrm{Var}\left(S\right) = \frac{1}{|S|} \sum_{t_i \in \mathcal{S}} \left(\mathcal{F}\left(t_i\right) - E\left(S\right)\right)^2$$

wobei $\mathrm{Var}\left(S\right)$ die Standardabweichung ("Variance") des Segments S darstellt.

5.2.2 Ähnlichkeitsfunktionen

Für die Abbildung der positiv reellwertigen Distanz $D_\mathcal{F}$ auf eine Ähnlichkeit $A_\mathcal{F}$ mit Werten zwischen 0 und 1 wird eine Ähnlichkeitsfunktion $\mathcal{A}_\mathcal{F} : D_\mathcal{F} \in \mathbb{R}^+ \rightarrow A_\mathcal{F} \in [0, 1]$ verwendet. Die Ähnlichkeitsfunktion eines Merkmals $\mathcal{F}$ bestimmt anhand der Distanz zweier Trajektorien einen Wert, der die Ähnlichkeit der beiden Trajektorien innerhalb des Merkmals angibt. Dabei hat eine Ähnlichkeitsfunktion in der Regel bei einer Distanz $D_\mathcal{F} = 0$ den Wert $\mathcal{A}_\mathcal{F}\left(0\right) = 1$ (maximale Ähnlichkeit) und fällt mit steigender Distanz gegen 0 ab.

Ähnlichkeitsfunktionen werden für die Berechnung der Ähnlichkeit aus der *Distanz zwischen Mittelwerten* $A_\mu = \mathcal{A}_\mu\left(D_\mu\right)$ und der Ähnlichkeit aus der *Distanz zwischen Standardabweichungen* $A_\sigma = \mathcal{A}_\sigma\left(D_\sigma\right)$ verwendet, die jeweils separat in einem Dialog eingestellt werden können. Dafür stehen in erster Linie zwei verschiedene Funktionstypen zur Verfügung, die als Ähnlichkeitsfunktionen in Fuzzy-Anwendungen weite Verbreitung finden [MK96]: Eine lineare Funktion und eine (asymmetrische) Gauß-Funktion. Bei beiden Funktionen lässt sich eine Distanz eingeben, bis zu der die Ähnlichkeit konstant 1 ist. Wird eine lineare Funktion

$f(x,a,b)$ verwendet, ist der berechnete Ähnlichkeitswert der Distanz x bis zur Distanz a konstant 1, fällt linear bis zur Distanz $b \geq a$ ab und ist danach konstant 0:

$$f(x,a,b) = \begin{cases} 1, & x \leq a \\ \frac{b-x}{b-a}, & a < x \leq b \\ 0, & x > b \end{cases}$$

In der asymmetrischen Gauß-Funktion $g(x,\mu,\sigma)$ ist die Ähnlichkeit bist zum Mittelwert μ der (halben) Gauß-Glocke konstant 1, darüber fällt sie mit einstellbarer Standardabweichung gegen 0 ab:

$$g(x,\mu,\sigma) = \begin{cases} 1, & x \leq \mu \\ e^{-\left(\frac{x-\mu}{\sigma}\right)^2}, & x > \mu \end{cases}$$

Abbildung 5.3 zeigt die beiden Kurven wie sie im Benutzerdialog des Filters konfigurierbar sind.

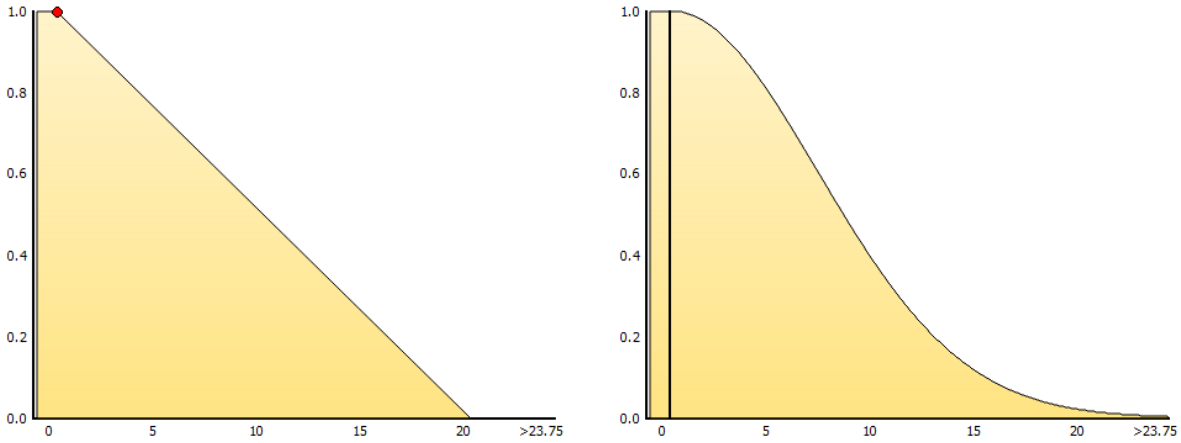

Abbildung 5.3: Lineare und Gauß-Funktion als Ähnlichkeitsfunktion.

Durch die Ähnlichkeitsfunktionen kann im Kontext der skizzierten Trajektorie eine Art Unsicherheit modelliert werden. Damit kann ausgedrückt werden, mit welcher Sicherheit der in der Trajektorienskizze modellierte Wert eines Merkmals exakt mit dem erwünschten Ergebnis übereinstimmt. Durch die Ähnlichkeitsfunktion lässt sich einstellen, wie groß der Bereich um den modellierten Wert sein soll, in dem die Trajektorien noch genügend ähnlich sind, um gefunden zu werden.

Das Überdeckungsmaß gibt auf Grund der Berechnungsart keine Distanz, sondern direkt eine Ähnlichkeit der verglichenen Trajektorien an. Dabei wird bei Nichtüberdeckung der Wert 0 zurückgegeben, bei zunehmender Überdeckung steigt der Wert gegen 1 an. In diesem Falle ist $A_c = D_c$, eine spezielle Ähnlichkeitsfunktion wird nicht benötigt.

Weitere Ähnlichkeitsfunktionen

Zu den beiden oben vorgestellten Ähnlichkeitsfunktionen bietet der Dialog noch drei weitere, nicht monoton fallende Funktionen. Die Auffälligkeit bei diesen ist, dass der Distanzwert 0 nicht unbedingt eine Ähnlichkeit von 1 nach sich zieht. Dies mag auf den ersten Blick paradox erscheinen, da ja im intuitiven Sinne zwei Trajektorien (in einer Eigenschaft) als zueinander ähnlich bezeichnet werden, wenn die Distanz zwischen beiden (in dieser Eigenschaft) möglichst gering ist. So bilden die vorgestellten Funktionen eine Ähnlichkeit ab, die dann ansteigt, wenn sich die Distanz einem bestimmten Wert (oder Wertebereich) nähert, der nicht unbedingt 0 sein muss.

Beim Modellieren eines skizzenbasierten Queries ist diese Möglichkeit eher eine weitere Flexibilität in der Formulierung des Queries, da das Query in jeder Eigenschaft beliebig angepasst werden kann und somit eine "intuitive" Ähnlichkeitsfunktion ausreicht. Die Anwendung der drei Funktionen findet dann ihren Sinn, wenn Eigenschaften von Trajektorien miteinander verglichen werden, die beide aus der Datenquelle stammen und somit nicht veränderlich sind (siehe Kapitel 6 auf Seite 59). Hier lassen sich über diese Ähnlichkeitsfunktionen nicht nur Trajektorien finden, die sich besonders ähnlich sind, sondern auch solche, die eine bestimmte durch die Funktion festgelegte Distanz zueinander aufweisen.

Als weitere Ähnlichkeitsfunktionen stehen drei verschiedenen Typen zur Auswahl: eine Dreiecks-, eine Trapez- und eine symmetrische Gauß-Funktion. Dreiecks- und Trapezfunktion werden als lineare Funktionen über drei bzw. vier Knickpunkte definiert. Die Schenkel müssen dabei nicht symmetrisch sein. Die symmetrische Gauß-Funktion wird analog zur ihrem asymmetrischen Pendant durch Mittelwert und Standardabweichung beschrieben. Abbildung 5.4 zeigt die drei verschiedenen Funktionstypen.

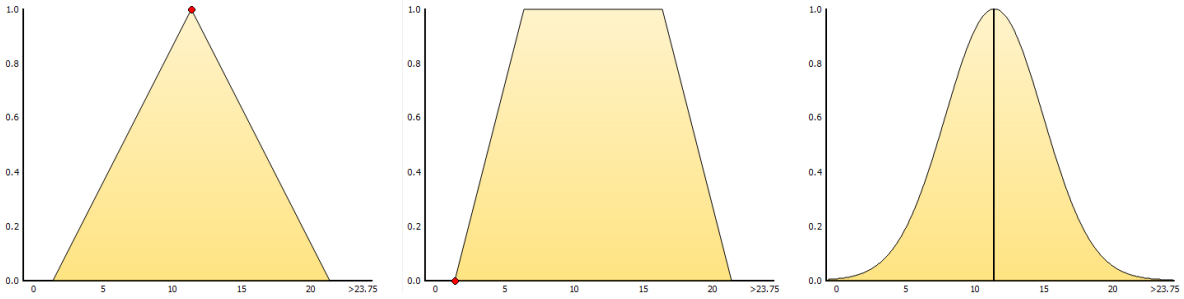

Abbildung 5.4: Weitere Ähnlichkeitsfunktionen: Dreiecksfunktion, Trapezfunktion und Gaußfunktion.

5.3 Merkmale und Algorithmus des Filters

Im Konfigurationsfenster des Filters lassen sich verschiedene Filterkriterien einstellen. Die ausgewählten Kriterien orientieren sich an verschiedenen Trajektorieneigenschaften und

bestimmen, welche Merkmale beim Vergleich der zu filternden Trajektorien mit der skizzierten Trajektorie berücksichtigt werden. Die einzelnen Merkmale decken sowohl temporale als auch räumliche Eigenschaften ab, wobei beliebige Kombinationen möglich sind. Durch die Auswahl wird angegeben, welche Eigenschaften der skizzierten Trajektorie tatsächlich für den Filtervorgang relevant sind. Abbildung 5.5 zeigt die auswählbaren Merkmale im Eigenschaftendialog des Filters.

Abbildung 5.5: Auswahl der Merkmale im Eigenschaftendialog des skizzenbasierten Trajektorienfilters. Aktiviert sind die Merkmale *Postion Coverage* (Positionsüberdeckung), *Direction Variance* (Richtungsvarianz) und *Mean Time* (mittlerer Zeitpunkt).

Die Merkmale des Filters und ihre Darstellung im Dialog sind wie die Vergleichsalgorithmen aus *ISS* entnommen [HHWH12]. Da sowohl in *ISS* als auch in diesem Filter Trajektorien zum Vergleich herangezogen werden und die Merkmale von *ISS* bereits in das *VAC* integriert sind, liegt es nahe, hier die selben Merkmale für die Auswahl der Filterkriterien zu verwenden. Einzig das in *ISS* zusätzlich vorhandene Merkmal zum Vergleich von Objektklassen steht im Filter nicht zur Verfügung, da Objektklassen keine Distanz zueinander haben und somit ein Vergleich in einem Fuzzy-Filter keinen Sinn macht. Die zur Auswahl stehenden Merkmale sind in die Gruppen *Position*, *Richtung*, *Geschwindigkeit* und *Zeit* unterteilt.

Position

Das Filtern Position der Trajektorien in der Videoszene steht in drei Merkmalen zur Verfügung: Überdeckung (“Coverage”), Durchschnitt (“Mean”) und Varianz (“Variance”), wobei die drei vorgestellten Vergleichsalgorithmen verwendet werden (siehe Abschnitt 5.2.1 auf Seite 50). Der Vergleich der Trajektorien geschieht anhand ihrer absoluten zweidimensionalen Koordinaten auf der projizierten Grundebene des Videoausschnitts (siehe Abschnitt 3.1.2 auf Seite 27). Die Distanz zwischen zwei Koordinaten ist die euklidische Distanz. Eine Längeneinheit entspricht dabei (sofern die Projektion der Koordinaten mit der tatsächlichen Szene übereinstimmt) einem Meter in der Szene.

Die Positionsüberdeckung gibt an, zu welchem Anteil die beiden verglichenen Trajektorien deckungsgleich sind. Mit dem Durchschnittsmerkmal werden die positionalen Mittelpunkte der einzelnen Segmente der Trajektorien miteinander verglichen. Über die Granularität lässt sich die Anzahl der Segmente, und somit die Anzahl der Durchschnittsvergleiche angeben. Über das Merkmal der Standardabweichung lässt sich feststellen, wie ähnlich die beiden Trajektorien bezüglich der Standardabweichung ihrer Positionen zum jeweiligen Mittelwert sind, wie sehr also ihre Position auf dem Weg vom Anfang zum Ende der Bewegung um den Mittelwert schwankt. Dafür steht ebenfalls eine Einstellung für die Granularität zur Verfügung, mit der die Anzahl der Unterteilungen der Trajektorien angegeben werden kann.

Richtung

Die Richtungsmerkmale geben an, in welche Richtung sich eine Trajektorie bewegt. Die Richtung wird über einen Winkel im Bezug zur Szene definiert. Dabei ist von der Winkeldistanzfunktion in den Algorithmen zu berücksichtigen, dass der volle Winkel von 360° einem Winkel von 0° entspricht und die “Distanz” zwischen zwei Winkeln somit nicht mehr als 180° betragen kann. Als Merkmale stehen ebenfalls Überdeckung, Durchschnitt und Varianz zur Verfügung. Für die Berechnung der Überdeckung ist die Bestimmung des Winkels an einem einzigen Sample erforderlich. Dafür wird der durchschnittliche Winkel in einem zeitlichen Bereich von zwei Sekunden um das angegebene Sample berechnet, wobei das Sample in der Mitte des Bereiches liegt. Der Berechnete Winkel ist also ein Mittelwert des Bereiches um das Sample.

Geschwindigkeit

Über die Geschwindigkeitsmerkmale lässt sich das Geschwindigkeitsverhalten von Trajektorien vergleichen. Hier sind wiederum die drei Merkmale Überdeckung, Durchschnitt und Varianz verfügbar, welche sich analog zu den Richtungsmerkmalen verhalten, mit dem einzigen Unterschied, dass die Distanz von zwei Geschwindigkeiten nicht beschränkt ist. Die Geschwindigkeiten werden in Kilometern pro Stunde angegeben. Für das Überdeckungsmaß wird wie bei der Richtungsüberdeckung ein Zwei-Sekunden-Bereich zur Berechnung der Geschwindigkeit an einem bestimmten Sample herangezogen und daraus der Mittelwert berechnet.

Zeit

Die Gruppe der Zeitmerkmale enthält Kriterien zum Vergleich der zeitlichen Trajektorieneigenschaften. Hier stehen nur zwei Merkmale zur Auswahl: die durchschnittliche (absolute) Zeit (Zeitstempel) und die Dauer ("Lifetime") der Trajektorie, welche über die Standardabweichung (Varianz) repräsentiert wird. Die Unterteilung der Trajektorien in mehrere Segmente hat auf Grund der Kontinuität der Zeitdimension keine Auswirkung auf das Ergebnis, somit ist die Angabe einer Granularität in den Zeitmerkmalen überflüssig. Ebenso ist eine Berechnung der zeitlichen Überdeckung in einem gesonderten Merkmal nicht nötig, da diese als Kombination der beiden genannten Merkmale angesehen werden kann [HHWH12].

5.3.1 Gewichtung

Die relative Bedeutung der aktivierten Merkmale lässt sich über ihre Gewichtung ausdrücken. Die Gewichtung bestimmt, wie stark sich die Ähnlichkeit der verglichenen Trajektorien bezüglich des einzelnen Merkmals auf die Gesamtähnlichkeit auswirkt. Für jedes Merkmal $\mathcal{F}$ in der Menge der aktivierten Merkmale $M_{\mathcal{F}}$ lässt sich eine Gewichtung $\theta_{\mathcal{F}}$ zwischen 0 und 1 einstellen. Der Anteil $\Theta_{\mathcal{F}}$ des Gewichtes des einzelnen Merkmals am Gesamtgewicht $\Theta = 1$ ergibt sich dann aus dem Anteil des eingestellten Gewichtes $\theta_{\mathcal{F}}$ an der Summe der Gewichte aller aktivierten Merkmale:

$$\Theta_{\mathcal{F}} = \frac{\theta_{\mathcal{F}}}{\sum_{\mathcal{F}_i \in M_{\mathcal{F}}} \theta_{\mathcal{F}_i}}$$

Somit erhalten Merkmale mit gleichem Gewichtungswert einen gleichen Anteil am Gesamtergebnis. Die Summe der anteiligen Merkmal-Gewichte ergibt in der Summe ein Gesamtgewicht von $\Theta = \sum_{\mathcal{F}_i \in M_{\mathcal{F}}} \Theta_{\mathcal{F}_i} = 1$. Eine Gewichtung von $\theta_{\mathcal{F}} = 0$ ist äquivalent zu einem deaktivierten Merkmal, da das dieses so keinen Beitrag zum Gesamtgewicht leistet.

5.3.2 Minimale Ähnlichkeit

Für jedes Merkmal lässt sich im Filterdialog ein minimaler Ähnlichkeitswert zwischen 0 und 1 eingeben, der bei der Berechnung der Ähnlichkeit der Trajektorien bezüglich dieses Merkmals als K.o.-Kriterium dient. Somit lässt sich für jedes Merkmal eine minimale Ähnlichkeit einstellen, die durch das Merkmal (unabhängig von seiner Gewichtung) erreicht werden muss. Dieses Kriterium wird für alle aktivierten Merkmale geprüft, für die ein minimaler Ähnlichkeitswert eingetragen ist (wenn kein Wert angegeben ist, wird eine minimale Ähnlichkeit von 1 angenommen). Wird in einem der Merkmale dieser Wert nicht erreicht, ist das Gesamtergebnis des Trajektorienvergleiches 0. Ansonsten hat dieses Kriterium keine Auswirkung auf das Vergleichsergebnis.

Der Beitrag $\Phi_{M_\mathcal{F}}$ der minimalen Ähnlichkeiten $\phi_\mathcal{F}$ zum Gesamtergebnis für jedes Merkmal $\mathcal{F}$ in der Menge der aktivierten Merkmale $M_\mathcal{F}$ lässt sich als Multiplikation der Prädikate für eine erreichte minimale Ähnlichkeit eines Merkmals ausdrücken:

$$\Phi_{M_\mathcal{F}} = \prod_{\mathcal{F}_i \in M_\mathcal{F}} P\left(A_{\mathcal{F}_i}, \phi_{\mathcal{F}_i}\right)$$

mit

$$P\left(A_\mathcal{F}, \phi_\mathcal{F}\right) = \begin{cases} 1, & \text{falls } A_\mathcal{F} \geq \phi_\mathcal{F} \\ 0, & \text{sonst} \end{cases}$$

wobei $P\left(A_\mathcal{F}, \phi_\mathcal{F}\right)$ das Prädikat der erreichten minimalen Ähnlichkeit eines Merkmals darstellt. $\Phi_{M_\mathcal{F}}$ ist also entweder exakt 1 (falls alle minimalen Ähnlichkeiten erreicht wurden) oder exakt 0 (falls mindestens eine minimale Ähnlichkeit nicht erreicht wurde).

Da die Ähnlichkeiten der aktivierten Merkmale anteilig miteinander verrechnet werden, kann es passieren, dass ein Merkmal keine Auswirkung auf das Ergebnis des Trajektorienvergleiches hat. Dies ist der Fall, wenn der vom Fuzzy-Container vorgegebene Schwellenwert bereits durch den (gewichteten) Beitrag anderer Merkmale erreicht wird. So ist der Ergebniswert des Filtervorgangs u.U. hoch, obwohl die Ähnlichkeit bezüglich dieses Merkmals sehr gering ist. Dies kann verhindert werden, in dem im entsprechenden Merkmal eine minimale Ähnlichkeit eingestellt und somit ein gewisser Beitrag dieses Merkmals zum Ergebnis erzwungen wird.

5.3.3 Algorithmus des Filters

Mit den berechneten Ähnlichkeiten der einzelnen Merkmale, den Gewichten und den minimalen Ähnlichkeiten lässt sich nun das Filterergebnis als Gesamtähnlichkeit $A_{M_\mathcal{F}}\left(T, T'\right)$ der zu vergleichenden Trajektorien T und T'ausdrücken:

$$A_{M_\mathcal{F}}\left(T, T'\right) = \left(\sum_{\mathcal{F}_i \in M_\mathcal{F}} \left(A_{\mathcal{F}_i}\left(T, T'\right) \cdot \Theta_{\mathcal{F}_i}\right) \right) \cdot \Phi_{M_\mathcal{F}}$$

Als Fuzzy-Filter gibt der Filter keine Ja/Nein-Antwort, ob eine Trajektorie den Filter passieren darf oder nicht. Das Ergebnis des Filters ist die Ähnlichkeit hinsichtlich der eingestellten Merkmale der zu prüfenden Videotrajektorie mit der skizzierten Trajektorie als reeller Zahlenwert im Intervall $[0, 1]$. Da ein Fuzzy-Filter immer in einem Fuzzy-Container eingebettet sein muss (siehe Abschnitt 3.5.2 auf Seite 32), kümmert sich der Container darum, dass das reelle Filterergebnis (evtl. in Kombination mit den Ergebnissen weiterer Filter im Container) mittels eines Schwellenwertes in eine boolesche Ja/Nein-Antwort umgewandelt wird.

Kapitel 6

Filtern von Trajektorien im zeitlichen Zusammenhang

Das Formulieren von Queries mit fest definierten Trajektorieneigenschaften, wie es bei der skizzenbasierten Modellierung geschieht, ist an sich sehr flexibel. Dennoch können damit keine Anfragen modelliert werden, die relative Beziehungen zwischen zwei Trajektorien definieren, die beide aus einer Datenquelle stammen. Dafür wurde ein sog. zeitabhängiger Filter entwickelt, mit dem solche Queries (mit einer zeitlichen Einschränkung) modelliert werden können.

Mit dem Filter lassen sich Trajektorien finden, die zu einem bestimmten Zeitpunkt sich in einer oder mehreren Eigenschaften möglichst ähnlich sind. Zum Beispiel sollen Trajektorien von Objekten gefunden werden, die sich zur selben Zeit am gleichen Ort befinden, sich in die gleiche Richtung bewegen oder die selbe Geschwindigkeit aufweisen. Dabei werden weder der Zeitpunkt noch Werte für die zu vergleichenden Eigenschaften vorgegeben. Ziel des Filters ist, zu einem bestimmten (aber unbekannten) Zeitpunkt eine möglichst hohe Ähnlichkeit anhand von vorgegebenen Ähnlichkeitsfunktionen zu finden. Durch die Ähnlichkeitsfunktionen und Fuzzy-Logik können auch Trajektorien mit einer definierten Distanz zwischen ihren Eigenschaften gefunden werden.

6.1 Zeitabhängigkeit und Caching-Problem

Im zeitabhängigen Filter sollen Trajektorien gefiltert werden werden, deren Objekte zum gleichen Zeitpunkt ähnliche Eigenschaften aufweisen. Da nach dem gleichen Zeitpunkt gefragt wird, kommt zum Vergleich zweier Trajektorien nur der Zeitraum in Frage, in dem sich die beiden Trajektorien zeitlich überlappen. Besteht eine solche Überlappung nicht, gibt es keinen Zeitpunkt, an dem beide Trajektorien gleichzeitig existieren. Insofern gibt es in diesem Fall auch keine Gemeinsamkeiten, ein Vergleich von sich zeitlich nicht überlappenden Trajektorien wäre überflüssig.

Da die Videotrajektorien miteinander verglichen werden, müssen bereits gefilterte Trajektorien in einem Cache gespeichert werden, damit diese mit später eintreffenden Trajektorien verglichen werden können. Der Filter hat dafür einen eigenen Trajektoriencache, der beim Filtern der Trajektorien gefüllt wird. Dabei müssen nur die Trajektorien im Cache gehalten werden, die mit der aktuell zu filternden Trajektorie eine zeitliche Überlappung haben. Andere Trajektorien können aus dem Cache entfernt werden. Dadurch ist auch sichergestellt, dass der Cache auf Dauer in seiner Größe beschränkt bleibt.

Das Caching-Problem

Nach dem Prinzip des Filters soll eine Trajektorie mit allen Trajektorien verglichen werden, die sich mit dieser zeitlich überlappen. Durch die Architektur der Filter und die Art der Verarbeitung der Trajektorien wird dieses Ziel allerdings durch zwei Kriterien eingeschränkt:

1. Die Architektur von Filtern im *VAC* sieht vor, dass für jede Trajektorie sofort das Filterergebnis berechnet und zurückgegeben wird, bevor die nächste Trajektorie verarbeitet werden kann. Da die Trajektorien in der zeitlichen Abfolge ihres Startzeitpunktes gefiltert werden, hat dies die Folge, dass das Filterergebnis einer Trajektorie nie von Trajektorien abhängig sein kann, die zeitlich später beginnen. Somit können zum Vergleich nur Trajektorien "aus der Vergangenheit" herangezogen werden.
2. Da der Filter Beziehungen zwischen Trajektorien bewertet, könnte man als Filterergebnis Trajektorienpaare erwarten. Dies wären jeweils zwei Trajektorien, die sich zu einem bestimmten Zeitpunkt in den zu vergleichenden Eigenschaften besonders ähnlich sind. Jedoch wird jede Trajektorie in ihrem Filtervorgang einzeln betrachtet, so dass die Markierung der Beziehungen zwischen sich ähnelnden Trajektorien als Filterergebnis nicht möglich ist. Zudem kann wegen der ersten Einschränkung immer nur die Trajektorie im Filter als "zu einer anderen Trajektorie ähnlich" bewertet werden, die von den beiden Trajektorien zum späteren Zeitpunkt anfängt. Die andere Trajektorie taucht u.U. gar nicht in der Ergebnismenge auf, da ihr Filterergebnis zu diesem Zeitpunkt bereits feststeht.

Letztendlich bezieht sich die Ausgabe des Filters auf Grund der genannten Einschränkungen auf Trajektorien, in deren Vergangenheit (bezüglich des Startzeitpunktes) zeitlich überlappende Trajektorien zu finden sind, die sich zu einem beliebigen, aber festen Zeitpunkt in den aktivierten Eigenschaften ähneln. Die Ähnlichkeit wird in jeder Eigenschaft durch eine Ähnlichkeitsfunktion definiert. Die Ausgabe des Filters für jede Trajektorie ist die *höchste Ähnlichkeit*, die die Trajektorie mit einer anderen Trajektorie im Cache zu einem bestimmten Zeitpunkt aufweist.

6.2 Merkmale und Algorithmus des Filters

Der Filter stellt als Auswahl drei verschiedene Merkmale zur Verfügung: *Position*, *Richtung* und *Geschwindigkeit*. Diese drei Merkmale sind – wie im skizzenbasierten Trajektorienfilter –

mit Einstellmöglichkeiten für Gewichtung und minimale Ähnlichkeit ausgestattet. Abbildung 6.1 zeigt die zur Auswahl stehenden Merkmale im Filterdialog.

Abbildung 6.1: Auswahl der Merkmale im Eigenschaftendialog des zeitabhängigen Trajektorienfilters. Aktiviert sind die Merkmale *Position* und *Richtung*.

Die zeitabhängige Distanzfunktion $D_{\mathcal{F}}(T, T', t)$ für jedes Merkmal $\mathcal{F}$ (*Position*, *Richtung* oder *Geschwindigkeit*) beschreibt jeweils die euklidische Distanz der *Position*, *Richtung* oder *Geschwindigkeit* zwischen den beiden Trajektorien T und T' zum Zeitpunkt t. Bei den Merkmalen *Richtung* und *Geschwindigkeit* wird der Wert zum Zeitpunkt t für jede Trajektorie über einen Bereich von zwei Sekunden um den Zeitpunkt gemittelt. $D_{\mathcal{F}}$ ist dabei nur für Zeitpunkte definiert, für die sowohl T als auch T' einen Wert besitzen. Der Zeitpunkt t muss also in dem Intervall liegen, das die zeitliche Überlappung der beiden Trajektorien darstellt.

Für jedes Merkmal lässt sich auch eine Ähnlichkeitsfunktion $\mathcal{A}_{\mathcal{F}}$ definieren, die die errechnete Distanz $D_{\mathcal{F}}$ auf einen Ähnlichkeitswert $A_{\mathcal{F}}$ abbildet:

$$A_{\mathcal{F}}(T, T', t) = \mathcal{A}_{\mathcal{F}}\left(D_{\mathcal{F}}(T, T', t)\right)$$

Als Ähnlichkeitsfunktionen stehen die selben Funktionstypen zur Verfügung, die auch für den skizzenbasierten Trajektorienfilter verwendet werden (siehe Abschnitt 5.2.2 auf Seite 51). Diese können genauso über den Eigenschaftendialog des Filters konfiguriert werden.

Unter Berücksichtigung der Merkmal-Gewichtungen $\Theta_{\mathcal{F}}$ und dem Kriterium für minimale Ähnlichkeiten $\Phi_{M_{\mathcal{F}}}$ (Abschnitte 5.3.1 und 5.3.2 auf Seite 56) lässt sich analog zur Berechnung des Ähnlichkeitswertes im skizzenbasierten Trajektorienfilter eine *zeitabhängige* Berechnungs-

vorschrift $Z_{M_\mathcal{F}}(T, T', t)$ aufstellen, die den Gesamtähnlichkeitswert der Trajektorien T und T' bezüglich der Menge der aktivierten Merkmale $M_\mathcal{F}$ zu einem Zeitpunkt t berechnet:

$$Z_{M_\mathcal{F}}(T, T', t) = \left(\sum_{\mathcal{F}_i \in M_\mathcal{F}} \left(A_{\mathcal{F}_i}(T, T', t) \cdot \Theta_{\mathcal{F}_i} \right) \right) \cdot \Phi_{M_\mathcal{F}}$$

Sei $I(T, T')$ das Zeitintervall, in dem sich T und T' überlappen. $M_t = \mathcal{S}_t(I(T, T'))$ ist dann die Menge der Zeitstempel in diesem Intervall, für die es in mindestens einer der beiden Trajektorien ein Sample gibt. $\mathcal{S}_t$ ist dabei die Segmentierungsfunktion, die die Samples beider Trajektorien in dem Abschnitt zurückgibt, in dem sich beide Trajektorien zeitlich überlappen. Die Gesamtähnlichkeit $A_{M_\mathcal{F}}(T, T')$ für diesen Filter kann in folgender Formel ausgedrückt werden:

$$A_{M_\mathcal{F}}(T, T') = \max_{t \in M_t} \left(Z_{M_\mathcal{F}}(T, T', t) \right)$$

Somit berechnet $A_{M_\mathcal{F}}(T, T')$ das Maximum der Ähnlichkeit zweier Trajektorien zu einem gemeinsamen Zeitpunkt. Existiert für T und T' keine zeitliche Überlappung, so ist die Menge der überlappenden Zeitstempel M_t leer. In diesem Fall ist die Ähnlichkeit $A_{M_\mathcal{F}} = 0$. Beim Trajektorienvergleich wird erwartet, dass die Trajektorien zwischen ihren Samples linear interpoliert sind und demnach dort keine Extrema besitzen. Somit genügt es, bei der Suche nach maximaler Ähnlichkeit die Trajektorien an den Zeitpunkten ihrer möglichen Extrema – den einzelnen Samples – zu vergleichen, um die minimale Distanz herauszufinden, da diese auf Grund der Annahme auf einem Sample liegen muss.

Kapitel 7

Validierung

In diesem Kapitel wird die Funktion der beiden vorgestellten Filter validiert. Dafür wird der Filtervorgang jeweils anhand eines Fallbeispiels näher erläutert und diskutiert. Abschnitt 7.1 beschäftigt sich mit der Validierung des skizzenbasierten Filters, Abschnitt 7.2 mit der des zeitabhängigen Filters.

7.1 Validierung des skizzenbasierten Filters

Zunächst wird die Funktion des in Kapitel 5 auf Seite 47 vorgestellten skizzenbasierten Filters anhand eines Fallbeispiels überprüft. In diesem Beispiel wird eine statische Analyse durchgeführt, d.h. ohne das Abspielen eines Videos in Echtzeit. Mittels des skizzenbasierten Trajektorienfilters sollen Trajektorien mit bestimmten Eigenschaften aus einem Datensatz gefunden werden. Dafür stehen allein die Trajektoriendaten – ohne Videobild – zur Verfügung. Lediglich ein statisches Hintergrundbild zur Darstellung des Kameraausschnitts ist verfügbar. Die Trajektoriendaten sind ein kleiner Ausschnitt aus der *Edinburgh Informatics Forum Pedestrian Database*[1] [Majo9], die die Bewegungsdaten von Personen enthält, die das Forum der *School of Informatics* der Universität Edinburgh durchquert haben. Der Ausschnitt beschränkt sich in diesem Beispiel auf ca. 50 Trajektorien innerhalb eines Zeitraums von einigen Minuten.

Als View zur Darstellung der Trajektorien wird *ISS* [HHWH11a] eingesetzt, dessen Funktionalität mit dieser Arbeit erweitert wurde. Die *ISS*-View ist eine View zum Einsatz in der statischen Analyse. Dabei werden die gesamten Trajektoriendaten zunächst in einen Cache geladen, um in gesamtem Umfang zur Verfügung zu stehen. Anschließend werden die geladenen Trajektorien gefiltert und in der View dargestellt. Dies geschieht im Gegensatz zu einer Echtzeitanalyse ohne Berücksichtigung der Zeitinformationen der Trajektorien, so dass das Filterergebnis aller Trajektorien so schnell wie möglich berechnet und zur Verfügung

[1] `http://homepages.inf.ed.ac.uk/rbf/FORUMTRACKING`

gestellt wird. Wird durch eine Änderung der Filterkriterien innerhalb von *ISS* oder im Filterbaum eine Neuberechnung der dargestellten Trajektorien gefordert, wird der Filtervorgang für alle Trajektorien von Neuem durchgeführt und das aktualisierte Ergebnis in der View dargestellt.

Bei der Auswahl eines oder mehrerer Cluster in der *ISS*-View wird die Anzeige auf die Trajektorien innerhalb der markierten Cluster beschränkt und neu dargestellt. Für diesen Schritt wird ein eigener Filter erstellt und in den Filterbaum eingehängt, der genau diese Reduzierung der Trajektorien vornimmt. Durch das erneute Filtern der Trajektorien mit dem aktualisierten Filterbaum erhält die View das gewünschte Ergebnis. Durch die Filterimplementierung kann die Interaktion innerhalb von *ISS* beliebig mit anderen Filtern, z.B. dem skizzenbasierten Filter, kombiniert werden, wie es in diesem Beispiel demonstriert wird.

Abbildung 7.1 zeigt die ungefilterten Trajektorien innerhalb der *ISS*-View. Diese sind anhand ihrer Position in verschiedene Cluster zusammengefasst. Der Kameraausschnitt zeigt das *Edinburgh Informatics Forum* aus der Top-Down-Perspektive, also senkrecht von oben.

Abbildung 7.1: Ungefilterte Trajektorien in *ISS*, nach ihrer Position in verschiedene Cluster zusammengefasst.

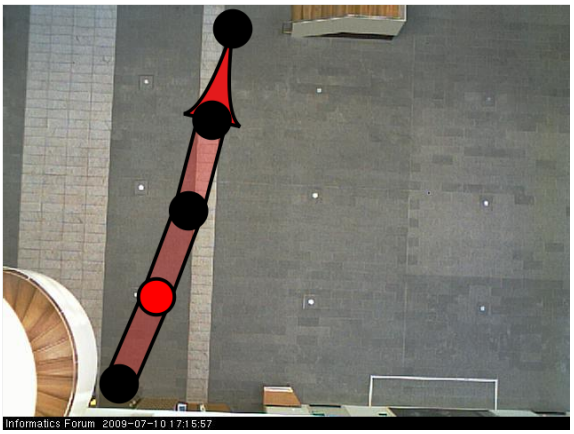

Abbildung 7.2: Erstellte Trajektorienskizze im Eigenschaftendialog des Filters.

7.1.1 Konfiguration des Filters

Nun wird der skizzenbasierte Filter eingesetzt, um die Trajektorien der Personen herauszufiltern, die vom Eingang der Halle (unten links) zur Treppe (oben in der Mitte) gegangen sind. Dafür wird ein neuer skizzenbasierter Filter erstellt und in den Filterbaum eingehängt. Da der skizzenbasierte Filter ein Fuzzy-Filter ist, muss dieser in einen Fuzzy-Container eingefügt werden, der über seinen Schwellenwert die letztendliche Filterentscheidung trifft. In diesem Fall wird ein Schwellenwert von 0,5 gewählt. Der Container lässt also alle Trajektorien passieren, für die der Fuzzy-Filter im inneren des Containers eine Ähnlichkeit von 0,5 oder

höher berechnet hat. Nach dem Einhängen kann der Filter über den Eigenschaftendialog konfiguriert und die Skizze erstellt werden. Abbildung 7.2 zeigt die angefertigte Skizze, mit der Trajektorien in der Menge gefunden werden sollen, die vom Eingang des Forums zur Treppe verlaufen. Der Verlauf der Trajektorie ist in der Skizze mit fünf Samples (schwarze Punkte) beschrieben. Der zeitliche Verlauf und somit die Bewegungsrichtung der Trajektorie ist durch die Pfeilspitze gekennzeichnet.

Da in diesem Fall die zeitlichen Aspekte der Trajektorien für das Filtern ohne Bedeutung sind, können diese beim Erstellen der Skizze vernachlässigt werden. Welche Trajektorieneigenschaften beim Vergleich zum Einsatz kommen, wird in den Merkmal-Einstellungen des Filters festgelegt. Hier sind die Position und Bewegungsrichtung der Trajektorien gefragt. Dafür werden im Filter die Merkmale *Position Coverage* (Positionsüberdeckung) und *Direction Mean* (Richtung im Mittel) aktiviert. Für die durchschnittliche Richtung wird eine Granularität von 2 gewählt, jede Trajektorie wird also für die Berechnung in zwei Segmente aufgeteilt, die separat miteinander verglichen werden. Somit wird sichergestellt, dass sowohl der vordere als auch der hintere Teil der Trajektorie im Mittel die gewünschte Richtung einhält. Als Ähnlichkeitsfunktion für *Direction Mean* wird eine asymmetrische Gaußkurve eingesetzt. Bis zu einer Winkelabweichung von 5 Grad der mittleren Trajektorienrichtung zur Skizze wird eine Ähnlichkeit von 1 definiert. Darüber fällt die Ähnlichkeit ab und verläuft gegen 0, bis sie ab einer Richtungsabweichung von ca. 30 Grad so gering ausfällt, dass sie keinen nennenswerten Beitrag mehr leistet.

Das Richtungsmerkmal wird zusätzlich in seiner Gewichtung etwa um die Hälfte herabgesetzt, um die größere Bedeutung der Position einer Trajektorie gegenüber ihrer Richtung hervorzuheben. Abbildung 7.3 illustriert die Merkmal-Einstellungen des Filters.

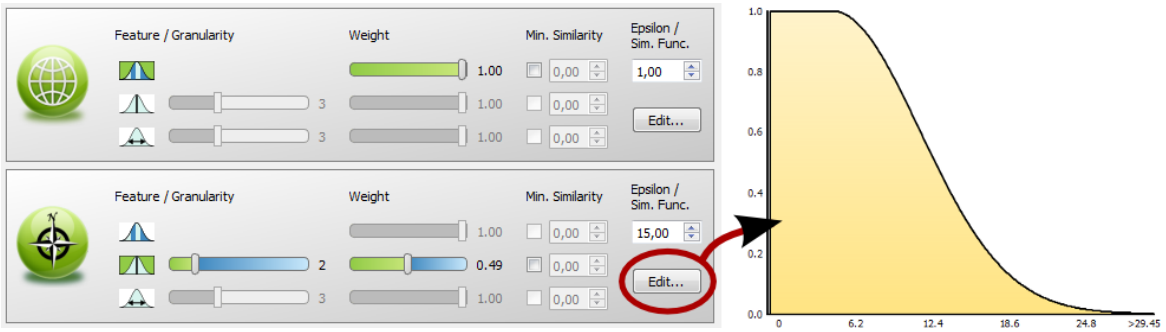

Abbildung 7.3: Einstellungen des Skizzenfilters: Aktiviert sind die Merkmale *Position Coverage* und *Direction Mean* mit Gauß-Ähnlichkeitskurve (Ähnlichkeit nach Richtungsabweichung in Grad). Die Gewichtung von *Direction Mean* ist im Verhältnis zu *Position Coverage* auf die Hälfte reduziert.

7.1.2 Filterergebnis

Als Ergebnis des Filtervorgangs bleiben von den ursprünglich ca. 50 Trajektorien noch fünf übrig, die – wie erwartet – ähnlich der Skizze zwischen Eingang und Treppe verlaufen. Abbildung 7.4 zeigt die gefilterten Trajektorien im Vergleich zur Skizze, einmal in Cluster zusammengefasst und einmal als einzelne Trajektorien. Diese Menge wäre jetzt klein genug, um sie manuell zu überprüfen.

Abbildung 7.4: Trajektorienskizze des Filters (links) im Vergleich mit dem Filterergebnis in Clusterrepräsentation (Mitte) bzw. als einzelne Trajektorien (rechts).

Die Clusterdarstellung in der Mitte zeigt jedoch, dass das Ergebnis Trajektorien enthält, die – entgegen der angegebenen Richtung in der Skizze – in die entgegengesetzte Richtung verlaufen. Dieses Ergebnis ist eine Folge der Unschärfe der verwendeten Fuzzy-Logik. Durch die höhere Gewichtung reicht eine hohe Übereinstimmung der Position aus, um allein den Schwellenwert von 0,5 zu erreichen, sodass der Beitrag der Richtung bedeutungslos wird. So kann auch eine Trajektorie, deren Ähnlichkeit bezüglich ihrer Richtung sehr gering ist, durch eine hohe Positionsüberdeckung den Filter passieren. Eine höhere Gewichtung der Richtung auf Kosten des Positionsgewichtes kann dieses Problem nicht vermeiden, sondern nur verlagern. Vermeiden ließe sich dieser Effekt durch einen höheren Schwellenwert im Container, was aber auch absichtlich gefilterte Trajektorien ausschließen könnte. Eine bessere Möglichkeit wäre das Einstellen einer minimalen Ähnlichkeit im Richtungsmerkmal, das auch Trajektorien ausschließt, die trotz hoher Positionsüberdeckung eine zu geringe Richtungsähnlichkeit aufweisen. Dies wäre bei den in die falsche Richtung verlaufenden Trajektorien der Fall.

7.2 Validierung des zeitabhängigen Filters

Der in Kapitel 6 auf Seite 59 vorgestellte zeitabhängige Filter ist vor allem zum Filtern von Datensätzen geeignet, deren Trajektorie zeitlich weit gestreut sind. Mit diesem Filter lassen sich Trajektorien finden, deren Objekte sich zur gleichen Zeit in einer Eigenschaft ähnlich sind, z.B. sich zur gleichen Zeit ungefähr am selben Ort befinden. Dadurch lässt

sich beispielsweise herausfinden, welche Objekte sich zu einem gemeinsamen Zeitpunkt mit anderen treffen oder kreuzen.

Für dieses Beispiel wurden Trajektorien aus GPS-Daten verwendet. Die Trajektorien beschreiben aufgezeichnete Bewegungen von Testpersonen in einem Umkreis von ca. 30 Kilometern und einem Zeitraum von etwa zwei Monaten. Dabei wurden ca. 2000 Trajektorien aufgezeichnet. Als View wurde eine Ansicht verwendet, die die Trajektorien im Kontext einer Landkarte darstellt[2].

7.2.1 Konfiguration des Filters

Der Filter soll Trajektorien finden, deren Objekt sich jeweils mit dem einer (beliebigen) anderen Trajektorie aus dem Datensatz zur gleichen Zeit ungefähr am selben Ort befindet. Dafür wird im Eigenschaftendialog des Filters das Merkmal *Position* aktiviert. Die Ähnlichkeitskurve wird so eingestellt, dass zwei Trajektorien beim Vergleich bis zu einem minimalen Abstand von 50 Metern zur selben Zeit mit einer Ähnlichkeit von 1 bewertet werden, die Standardabweichung der sich anschließenden (asymmetrischen) Gauß-Kurve beträgt 25 Meter. Abbildung 7.5 zeigt die Filtereinstellungen und die Ähnlichkeitskurve des Merkmals *Position*.

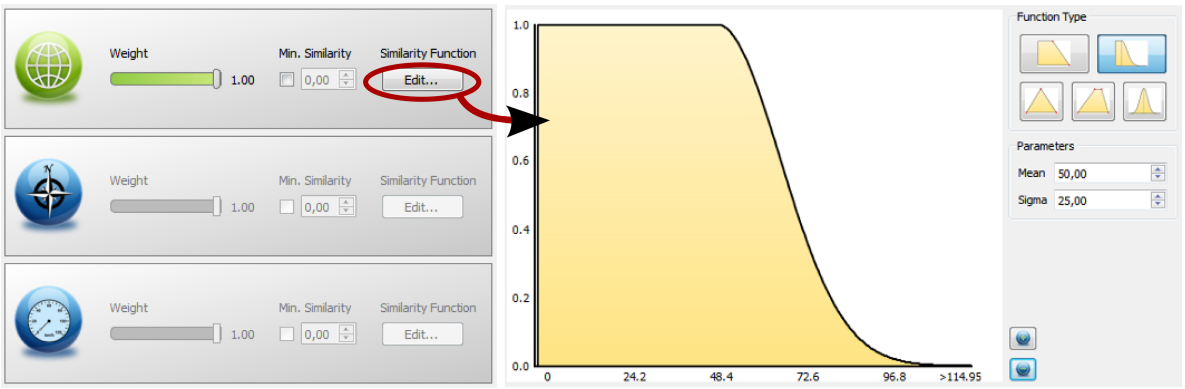

Abbildung 7.5: Filtereinstellungen des zeitabhängigen Filters: aktiviert ist das Merkmal *Position* mit Gauß-Ähnlichkeitskurve.

7.2.2 Filterergebnis

Das Ergebnis des Filtervorgangs ist eine Menge von ca. 10 Trajektorien, für deren Objekte der Filter jeweils eine genügend große Nähe zu einem anderen Objekt erkannt hat. Zu erwarten ist, dass die Ergebnismenge die Trajektorien aller Objekte enthält, die zu einem

[2]Genauere Angaben über die verwendeten Daten können aus lizenzrechtlichen Gründen nicht aufgeführt werden.

anderen Objekt irgendwann zur selben Zeit nicht weiter als etwa 50 Meter entfernt sind. Das Ergebnis ist hinsichtlich dieser Erwartung allerdings auf Grund der Einschränkungen in der Funktionsweise des Filters (siehe Abschnitt 6.1 auf Seite 59) unzureichend. Es können lediglich Trajektorien ermittelt werden, die sich mit einer beliebigen anderen Trajektorie treffen, die zeitlich davor liegt, also bereits gefiltert wurde und sich im Cache des Filters befindet. Ein Blick in die "Zukunft" ist nicht möglich. Zudem kommt diese andere Trajektorie nicht in der Ergebnismenge vor, wenn sie bei ihrem Filtervorgang zuvor ausgefiltert wurde.

Da aus dem Ergebnis des zeitabhängigen Filters heraus nicht bekannt ist, mit welcher anderen Trajektorie sich jede Trajektorie im Ergebnis trifft, ist eine visuelle Verifikation oder Interpretation der Daten ohne weitere Schritte nur schwer möglich. Die Trajektorien im Ergebnis können aber gut als Hinweis für eine genauere Betrachtung dienen, um Treffen oder andere Gemeinsamkeiten von Trajektorien zum gleichen Zeitpunkt herauszufinden. In der weiteren Analyse (z.B. mittels anderer Filter bzw. Views) könnte dann z.B. die nähere Umgebung der gefundenen Trajektorien untersucht werden.

Kapitel 8

Zusammenfassung

In dieser Arbeit wurde gezeigt, wie aus einem Überwachungsvideo (oder einer anderen Datenquelle) extrahierte Trajektorien anhand einer Skizze gesucht, bzw. gefiltert werden können. Dafür wurde eine Benutzeroberfläche zum Zeichnen der Skizze und Modellieren ihrer räumlichen und zeitlichen Eigenschaften entwickelt. Die Suche nach Trajektorien wurde durch einen Trajektorienfilter realisiert, der den Datenbestand mit der erstellten Skizze vergleicht. Die zu vergleichenden Trajektorieneigenschaften können im Filter ausgewählt und konfiguriert werden. Durch den Einsatz der Fuzzy-Logik können durch den Filter unscharfe Suchanfragen gestellt werden. Der Grad und die Art der Unschärfe kann durch Ähnlichkeitsfunktionen definiert werden. So können auch unsichere Kriterien zur Trajektoriensuche definiert werden, die ein sinnvolles Ergebnis liefern.

Ähnliche Eigenschaften von Trajektorien aus dem Datensatz können durch den Einsatz des zeitabhängigen Filters gefunden werden. Dabei werden Trajektorieneigenschaften zum selben Zeitpunkt miteinander verglichen. Hier wird das Prinzip der unscharfen Anfragen ebenso verwendet und erlaubt auch hier Anfragen unter Unsicherheit. Jedoch kann durch die genannten technischen Beschränkungen des Filterprinzips kein optimales Ergebnis erzielt werden. Eine Lösung, die auch Vergleiche mit Trajektorien "in der Zukunft" zulässt oder Beziehungen zwischen Trajektorien als Filterergebnis ausgeben kann, muss hier noch entwickelt werden.

Dieses Problem könnte etwa durch eine statische View gelöst werden, die den kompletten Trajektoriensatz zur Verfügung hat und nicht den Beschränkungen der Filterarchitektur unterliegt. Durch die View kann (ähnlich wie bei *ISS*) ein sog. "Whitelist-Filter" erstellt werden, der eine spezifisch vorgegebene Trajektorienmenge passieren lässt und somit das Ergebnis eines zeitabhängigen Filters ohne die gegebenen technischen Einschränkungen emuliert.

Die Möglichkeiten, Trajektorien mittels unscharfer Anfragen zu finden, sind mit den vorgestellten Filtern natürlich längst nicht ausgeschöpft. Durch das Heranziehen von anderen Trajektorieneigenschaften zum Vergleich und entsprechenden Vergleichsmethoden können je nach Anwendung neue Erkenntnisse über die analysierten Daten gewonnen werden. Auch

das Finden von Beziehungen zwischen Trajektorien bietet einiges an Potential, wodurch nicht nur Eigenschaften von bewegten Objekten, sondern auch Interaktionen zwischen ihnen in den Mittelpunkt rücken. Effizientere Vergleichsalgorithmen könnten hier zusätzlich die Berechnungszeit vor allem bei sehr großen Datenmengen deutlich verringern.

Das Ziel dieser Arbeit ist, den Visual-Analytics-Prozess bei der Analyse von Überwachungsvideos und anderen großen Datenmengen hinsichtlich ihrer enthaltenen Trajektorien weiter zu unterstützen. Die vorgestellten Ideen als auch evtl. nachfolgende Methoden sind somit ein Ansatz, diese Analyse einfacher und schneller zu gestalten, was auf Grund der immer weiter steigenden Menge an Überwachungsvideo- und Trajektoriendaten auch unbedingt notwendig ist.

Abbildungsverzeichnis

Literaturverzeichnis

[AAW07] G. Andrienko, N. Andrienko, S. Wrobel. Visual analytics tools for analysis of movement data. *SIGKDD Explor. Newsl.*, 9:38–46, 2007. doi:10.1145/1345448.1345455. (Zitiert auf Seite 16)

[BG73] R. Bellman, M. Giertz. On the analytic formalism of the theory of fuzzy sets. *Information Sciences*, 5(0):149 – 156, 1973. doi:10.1016/0020-0255(73)90009-1. (Zitiert auf Seite 24)

[Ble10] A. Bleicher. Eyes in the Sky That See Too Much [Update]. *Spectrum, IEEE*, 47(10):16, 2010. doi:10.1109/MSPEC.2010.5583451. (Zitiert auf Seite 7)

[BPB+10] M. Broilo, N. Piotto, G. Boato, N. Conci, F. De Natale. Object Trajectory Analysis in Video Indexing and Retrieval Applications. In D. Schonfeld, C. Shan, D. Tao, L. Wang, editors, *Video Search and Mining*, volume 287 of *Studies in Computational Intelligence*, pp. 3–32. Springer Berlin / Heidelberg, 2010. doi:10.1007/978-3-642-12900-1_1. (Zitiert auf den Seiten 17, 37 und 44)

[BSE08] R. Botchen, F. Schick, T. Ertl. Action-Based Multifield Video Visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 14(4):885 –899, 2008. doi:10.1109/TVCG.2008.40. (Zitiert auf den Seiten 17 und 34)

[CCM+97] S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram, D. Zhong. VideoQ: an automated content based video search system using visual cues. In *Proceedings of the fifth ACM international conference on Multimedia*, MULTIMEDIA '97, pp. 313–324. ACM, New York, NY, USA, 1997. doi:10.1145/266180.266382. (Zitiert auf Seite 19)

[CCM+98] S.-F. Chang, W. Chen, H. Meng, H. Sundaram, D. Zhong. A fully automated content-based video search engine supporting spatiotemporal queries. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(5):602 –615, 1998. doi:10.1109/76.718507. (Zitiert auf den Seiten 9, 18, 19 und 71)

[CPC09] S. Calderara, A. Prati, R. Cucchiara. Video surveillance and multimedia forensics: an application to trajectory analysis. In *Proceedings of the First ACM*

workshop on Multimedia in forensics, MiFor '09, pp. 13–18. ACM, New York, NY, USA, 2009. doi:10.1145/1631081.1631085. (Zitiert auf den Seiten 16 und 42)

[DP97] D. Dubois, H. Prade. *Using fuzzy sets in flexible querying: why and how?*, pp. 45–60. Kluwer Academic Publishers, Norwell, MA, USA, 1997. (Zitiert auf Seite 24)

[Fag96] R. Fagin. Combining fuzzy information from multiple systems (extended abstract). In *Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '96, pp. 216–226. ACM, New York, NY, USA, 1996. doi:10.1145/237661.237715. (Zitiert auf Seite 23)

[Fag98] R. Fagin. Fuzzy queries in multimedia database systems. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '98, pp. 1–10. ACM, New York, NY, USA, 1998. doi:10.1145/275487.275488. (Zitiert auf Seite 17)

[GKV+07] A. Girgensohn, D. Kimber, J. Vaughan, T. Yang, F. Shipman, T. Turner, E. Rieffel, L. Wilcox, F. Chen, T. Dunnigan. DOTS: support for effective video surveillance. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pp. 423–432. ACM, New York, NY, USA, 2007. doi:10.1145/1291233.1291332. (Zitiert auf Seite 16)

[GSTW07] A. Girgensohn, F. Shipman, T. Turner, L. Wilcox. Effects of presenting geographic context on tracking activity between cameras. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pp. 1167–1176. ACM, New York, NY, USA, 2007. doi:10.1145/1240624.1240801. (Zitiert auf Seite 16)

[HHW09] M. Höferlin, B. Höferlin, D. Weiskopf. Video Visual Analytics of Tracked Moving Objects. In *Proceedings of 3rd Workshop on Behaviour Monitoring and Interpretation*, volume 541, pp. 59–64. CEUR Workshop Proceedings, 2009. (Zitiert auf Seite 17)

[HHWH11a] M. Höferlin, B. Höferlin, D. Weiskopf, G. Heidemann. Interactive Schematic Summaries for Exploration of Surveillance Video. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ICMR '11, pp. 9:1–9:8. ACM, New York, NY, USA, 2011. doi:10.1145/1991996.1992005. (Zitiert auf den Seiten 7, 8, 20, 21, 25, 34, 39, 63, 71 und 74)

[HHWH11b] M. Höferlin, B. Höferlin, D. Weiskopf, G. Heidemann. Uncertainty-Aware Video Visual Analytics of Tracked Moving Objects. *Journal of Spatial Information Science (JOSIS)*, 2:87–117, 2011. doi:10.5311/JOSIS.2010.2.1. (Zitiert auf den Seiten 17, 22 und 71)

[HHWH12] M. Höferlin, B. Höferlin, D. Weiskopf, G. Heidemann. Interactive Schematic Summaries for Faceted Exploration of Surveillance Video. 2012. Bisher unveröffentlichte Erweiterung von *Interactive Schematic Summaries for Exploration of Surveillance Video* [HHWH11a]. (Zitiert auf den Seiten 25, 50, 54 und 56)

[HXF+07] W. Hu, D. Xie, Z. Fu, W. Zeng, S. Maybank. Semantic-Based Surveillance Video Retrieval. *Image Processing, IEEE Transactions on*, 16(4):1168 –1181, 2007. doi:10.1109/TIP.2006.891352. (Zitiert auf den Seiten 9, 18 und 45)

[HYC06] J.-W. Hsieh, S.-L. Yu, Y.-S. Chen. Motion-based video retrieval by trajectory matching. *Circuits and Systems for Video Technology, IEEE Transactions on*, 16(3):396 – 409, 2006. doi:10.1109/TCSVT.2006.869965. (Zitiert auf Seite 9)

[KAF+08] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, G. Melançon. Visual Analytics: Definition, Process, and Challenges. In A. Kerren, J. Stasko, J.-D. Fekete, C. North, editors, *Information Visualization*, volume 4950 of *Lecture Notes in Computer Science*, pp. 154–175. Springer Berlin / Heidelberg, 2008. doi:10.1007/978-3-540-70956-5_7. (Zitiert auf den Seiten 13, 14 und 15)

[KMS+08] D. Keim, F. Mansmann, J. Schneidewind, J. Thomas, H. Ziegler. Visual Analytics: Scope and Challenges. In S. Simoff, M. Böhlen, A. Mazeika, editors, *Visual Data Mining*, volume 4404 of *Lecture Notes in Computer Science*, pp. 76–90. Springer Berlin / Heidelberg, 2008. doi:10.1007/978-3-540-71080-6_6. (Zitiert auf Seite 13)

[KMT10] D. A. Keim, F. Mansmann, J. Thomas. Visual analytics: how much visualization and how much analytics? *SIGKDD Explor. Newsl.*, 11:5–8, 2010. doi:10.1145/1809400.1809403. (Zitiert auf den Seiten 13, 14 und 71)

[LDFW07] P. Laube, T. Dennis, P. Forer, M. Walker. Movement beyond the snapshot - Dynamic analysis of geospatial lifelines. *Computers, Environment and Urban Systems*, 31(5):481 – 501, 2007. doi:10.1016/j.compenvurbsys.2007.08.002. (Zitiert auf Seite 16)

[Maj09] B. Majecka. *Statistical models of pedestrian behaviour in the Forum*. Master's thesis, School of Informatics, University of Edinburgh, 2009. (Zitiert auf Seite 63)

[MK96] S. Mitaim, B. Kosko. What is the best shape for a fuzzy set in function approximation? In *Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on*, volume 2, pp. 1237 –1243 vol.2. 1996. doi:10.1109/FUZZY.1996.552354. (Zitiert auf Seite 51)

[Shn96] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pp. 336 –343. 1996. doi:10.1109/VL.1996.545307. (Zitiert auf Seite 15)

[TZZ79] U. Thole, H.-J. Zimmermann, P. Zysno. On the suitability of minimum and product operators for the intersection of fuzzy sets. *Fuzzy Sets and Systems*, 2(2):167 – 180, 1979. doi:10.1016/0165-0114(79)90023-X. (Zitiert auf Seite 24)

[Web83] S. Weber. A general concept of fuzzy connectives, negations and implications based on t-norms and t-conorms. *Fuzzy Sets and Systems*, 11(1–3):103 – 113, 1983. doi:10.1016/S0165-0114(83)80073-6. (Zitiert auf Seite 24)

[YNT02] C. Yajima, Y. Nakanishi, K. Tanaka. Querying Video Data by Spatio-Temporal Relationships of Moving Object Traces. In *Proceedings of the IFIP TC2/WG2.6 Sixth Working Conference on Visual Database Systems: Visual and Multimedia Information Management*, pp. 357–371. Kluwer, B.V., Deventer, The Netherlands, 2002. (Zitiert auf Seite 9)

[Zad65] L. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965. doi: 10.1016/S0019-9958(65)90241-X. (Zitiert auf den Seiten 23 und 24)

[Zad88] L. Zadeh. Fuzzy logic. *Computer*, 21(4):83 –93, 1988. doi:10.1109/2.53. (Zitiert auf Seite 23)

[ZHT06] Z. Zhang, K. Huang, T. Tan. Comparison of Similarity Measures for Trajectory Clustering in Outdoor Surveillance Scenes. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pp. 1135 –1138. 2006. doi: 10.1109/ICPR.2006.392. (Zitiert auf Seite 21)

Erklärung

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

(Johannes Engelhardt)