

Masterarbeit – GEO 511
Geographisches Institut der Universität Zürich
Abteilung für Geographische Informationssysteme

Algorithmen zur Detektion von Reaktionsbewegungsmustern

Autor: Michael Merki
Riedmatt 1
5610 Wohlen
Tel.: 079 / 768 30 60
E-Mail: michaelmerki@gmx.ch
Matrikel-Nr.: 06-713-333

Betreuung: Dr. Patrick Laube

Fakultätsvertreter: Prof. Dr. Robert Weibel

Abgabetermin: 28.04.2011

Zusammenfassung

Mobilität ist allgegenwärtig auf dieser Welt und das Verständnis der Bewegungen essentiell für viele Anwendungsbereiche. Aus diesem Grund hat der Forschungsbereich der Bewegungsmusteranalyse in den letzten Jahren einen grossen Aufschwung erlebt und bei immer mehr Forschern von verschiedensten Wissenschaften grosses Interesse geweckt. Aufgrund der verbesserten technischen Möglichkeiten und der zunehmenden Verbreitung von Lokalisierungstechnologien ist es heute möglich, detaillierte Bewegungspfade aufzuzeichnen. Für die Auswertung der grossen Datenmengen werden Techniken benötigt, um Informationen über die Bewegungen abzuleiten. Die GIScience versucht dazu einen Beitrag zu leisten und Methoden zur Bewegungsmusteranalyse zu entwickeln.

In dieser Arbeit werden Bewegungsmuster, welche als Folge von interagierenden und sich dabei gegenseitig beeinflussenden Punktobjekten von unterschiedlichen Arten entstehen (Reaktionsbewegungsmuster), untersucht. Dabei wird eine Übersicht der in der Tierverhaltensforschung bekannten Reaktionsbewegungsmuster vorgestellt. Zudem werden für die drei ausgewählten Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* die wichtigsten raum-zeitlichen Merkmale präsentiert und zu treffenden Formalisierungen zusammengefasst. Dabei wurden zwei unterschiedlichen Betrachtungsweisen eingesetzt. Während sich die individuelle Perspektive bei der Analyse der Bewegungsmuster auf die Individuen fokussiert, versucht die räumliche Perspektive die Muster aus Sicht der Räume (z.B. Strassensegmente), welche den Bewegungsmustern unterliegen, zu interpretieren. Die formalen Beschreibungen bilden die Grundlage für die Entwicklung der Algorithmen zur Detektion dieser drei Reaktionsbewegungsmuster in Bewegungsdaten. Um die Anwendbarkeit der Algorithmen zu testen, wurden Experimente mit simulierten Bewegungsdaten sowie mit einem realen Datensatz (Frequentie1550) durchgeführt.

Eine wichtige Erkenntnis dieser Arbeit betrifft die Entwicklung der Algorithmen. Zur Detektion von Reaktionsbewegungsmustern bewähren sich Algorithmen, welche sich zusammensetzen aus der Entdeckung von einzelnen Mustern (z.B. *Zusammensein*, *Folgen*) und der Überprüfung von weiteren Kriterien (z.B. Verzögerungszeit zwischen Teilmustern). Dies widerspiegelt sich auch in den guten Resultaten der Experimente. Für einige Algorithmen gilt es aber noch zusätzliche Überlegungen zu machen und weitere Kriterien zu bestimmen, um die Anwendbarkeit noch zu verbessern.

Aufgrund der Tatsache, dass sich Reaktionsbewegungsmuster (z.B. *Verfolgen/Entfliehen*) in ihrem raum-zeitlichen Abdruck nur unwesentlich von anderen Interaktionsformen (z.B. *Anführen/Folgen*, *Single File*) unterscheiden, ist das Finden von geeigneten Kriterien zur erfolgreichen Unterscheidung dieser Mustern massgeblich für den Erfolg der Algorithmen. Mit dem Parameter Verzögerungszeit wurde ein solches Kriterium gefunden. Dasselbe gilt für die Unterscheidung zwischen Reaktionsbewegungsmustern und Mustern, welche sich als Resultat von zufälligen Bewegungen ergeben und daher keine Bedeutung haben.

Eine weitere wichtige Aussage ergibt sich beim Vergleich zwischen den beiden Perspektiven. Dabei stellte sich bei der räumlichen Perspektive das Segment, als kleinste räumliche Einheit, als zu klein heraus im Vergleich zur Distanzbeziehung, welche im Falle der individuellen Perspektive analysiert wird. Prinzipiell ist die Anwendbarkeit der räumlichen Perspektive stark abhängig von der kleinsten unterteilbaren Raumeinheit (z.B. Segmentgrösse).

Summary

Mobility is everywhere present and the comprehension of movements essential for many application areas. For this reason, the research area of movement pattern analysis has expanded largely in recent years and has awoken the interest of many researchers from different fields. Because of the improved technical possibilities and the increasing extension of localization technologies, it is today possible to draw detailed movements paths. For the analysis of this huge amount of data techniques in order to derive information about the movements are used. The GIScience tries to make a contribution to this topic and develops methods for the analysis of movement patterns.

In this thesis, movement patterns which arise as a consequence of interacting point objects from different species are investigated. Thereby an overview of the known movement patterns in the animal behavior science is given. Additionally, for the three chosen reaction movement patterns *pursuit/escape*, *confrontation* and *avoidance* the most important space-time features are presented and the corresponding formalizations are summarized. In doing so, two different approaches are used. While the individual perspective is focusing on the analysis of movement patterns of individuals, the spatial perspective is trying to interpret the patterns in the view of spaces (e.g. street segments), which represent the basis of the movement patterns. The formal descriptions represent the fundament for the development of algorithms for the detection of reaction movement patterns. In order to test the applicability of the algorithms, different experiments with simulated movement data as well as with a real data set are conducted.

An important founding of this thesis regards the development of the algorithms. Algorithms which consist of the detection of single patterns (*being together*, *pursuit*) and of the examination of further criteria (e.g. delay time between partial patterns) are found to be optimal. This is also affirmed by the good results of the experiments. However, for a few algorithms additional considerations have to be made and further criteria in order to improve the applicability have to be determined.

Because of the fact that reaction movement patterns (e.g. *pursuit/escape*) are only slightly different in space and time to other interaction forms (e.g. *leadership/following*, *single file*), the detection of convenient criteria which differentiate these patterns is crucial for the success of the algorithms. The parameter delay time is such a criterion. The same is true for the differentiation between reaction movement patterns and patterns which are the result of random movements and therefore have no importance.

Another important conclusion arises by comparing the two approaches. Thereby it turned out that the segment (from the spatial perspective) as smallest unit is too small in comparison to the distance measure from the individual perspective. In general, the applicability strongly depends on the smallest divisible space unit (e.g. segment size).

Danksagung

An dieser Stelle möchte ich allen, welche mich während meiner Studienzeit am Geographischen Institut der Universität Zürich und im Speziellen im Verlaufe meiner Masterarbeit unterstützt haben, einen ganz herzlichen Dank aussprechen.

Ein herzliches Dankeschön gilt speziell meinem Betreuer während der Masterarbeit, Dr. Patrick Laube, welcher mir durch die vielen interessanten Diskussionen immer wieder neue Gedankengänge aufzeigen und hilfreiche Unterstützung bieten konnte. Die Zusammenarbeit mit ihm hat sehr viel Spass gemacht, vielen Dank dafür.

Bedanken möchte ich mich weiter bei Daniel Orellana (Universität Wageningen), Femke Broekhuis (Universität Oxford) und Dr. Jennifer Miller (Universität Texas) für die zur Verfügung gestellten Bewegungsdatensätze, sowie die diversen interessanten Anregungen und die hilfreichen Tipps.

Ein spezieller Dank richtet sich zudem an meine Freundin Rebekka Schibli, welche mich während meinem Studium stets begleitet hat und mich auch bei der Masterarbeit tatkräftig unterstützt hat. Ebenfalls bedanken möchte ich mich bei allen Studienkollegen für die tolle Studienzeit und natürlich bei meiner Familie sowie allen Verwandten für die Unterstützung. Dabei gilt es meine Mutter Susanne Merki herauszuheben, welche durch das Korrekturlesen einen wichtigen Beitrag zur Fertigstellung dieser Masterarbeit geleistet hat.

Wohlen, 28.04.2011

Michael Merki

Inhalt

Zusammenfassung	III
Summary	IV
Danksagung	V
Inhalt	VI
Abbildungen	VIII
Diagramme	X
Tabellen	XI
1 Einleitung	1
1.1 <i>Kontext und Motivation</i>	1
1.2 <i>Problemstellung und Zielsetzungen</i>	2
1.3 <i>Forschungsfragen</i>	2
1.4 <i>Aufbau der Arbeit</i>	3
2 Hintergrund	4
2.1 <i>Bewegungsmusteranalyse in der GIScience</i>	4
2.2 <i>Anwendungsbereiche</i>	7
2.3 <i>Übersicht Reaktionsbewegungsmuster</i>	12
2.4 <i>Räuber/Beute-Interaktionen</i>	15
2.5 <i>Territoriale Interaktion</i>	18
2.6 <i>Intraspezifische soziale Interaktionen</i>	23
2.7 <i>Forschungslücken</i>	24
3 Problemdefinition	27
3.1 <i>Beschreibung der Problemstellung</i>	27
3.2 <i>Eingrenzung des Untersuchungsgegenstandes</i>	27
4 Formale Beschreibung der Reaktionsbewegungsmuster	36
4.1 <i>Verfolgen/Entfliehen</i>	36
4.2 <i>Konfrontation</i>	44
4.3 <i>Meiden</i>	51
5 Entwicklung von Algorithmen	56
5.1 <i>Zusammensein / Getrenntsein</i>	57

5.2	<i>Konvergenz / Divergenz</i>	61
5.3	<i>Folgen</i>	62
5.4	<i>Intersektion (Kreuzen)</i>	63
5.5	<i>Verfolgen/Entfliehen</i>	65
5.6	<i>Konfrontation</i>	66
5.7	<i>Meiden</i>	67
6	Experimente	68
6.1	<i>Simulationsdaten des Steering Behavior-Modells</i>	68
6.2	<i>Frequentie1550-Datensatz</i>	73
7	Resultate	84
7.1	<i>Simulierten Bewegungsdaten</i>	84
7.2	<i>Frequentie1550-Daten</i>	92
8	Diskussion	108
8.1	<i>Forschungsfrage 1: Herleitung formaler Definitionen</i>	108
8.2	<i>Forschungsfrage 2: Entwicklung von Algorithmen</i>	110
8.3	<i>Forschungsfragen 3 und 4: Einfluss der Parameter und Vergleich der beiden Perspektiven</i>	111
8.4	<i>Einordnung der Erkenntnisse in den Forschungskontext</i>	124
9	Schlussfolgerungen	127
9.1	<i>Zusammenfassung</i>	127
9.2	<i>Wichtigste Erkenntnisse</i>	127
9.3	<i>Ausblick</i>	130
10	Literaturverzeichnis	132
11	Anhang	137
11.1	<i>Java- Klasse MovementPatternDetector</i>	137
11.2	<i>Persönliche Erklärung</i>	181

Abbildungen

Abbildung 2.1: REMO-Framework (Laube et al. 2004).....	5
Abbildung 2.2: Bewegungsmuster <i>Konvergenz</i> (Laube et al. 2004).....	6
Abbildung 2.3: Detektion von <i>Approximationen</i> (Orellana et al. 2009).....	7
Abbildung 2.4: Entdeckung von ungewöhnlichen Ereignissen in Trajektorien (Morris und Trivedi 2008).....	9
Abbildung 2.5: <i>Ausrauben</i> (Mahajan et al. 2004).....	9
Abbildung 2.6: Beispiele von <i>Kämpfen</i> zwischen Personen (Szlavik 2008).....	9
Abbildung 2.7: Klassifikation von Bewegungsmustern (Dodge et al. 2008).....	13
Abbildung 2.8: Übersicht Reaktionsbewegungsmuster.....	15
Abbildung 2.9: Trajektorien eines Kampfes (Blythe et al. 1996).....	17
Abbildung 2.10: Konfrontation zwischen Rothirschen (Creel et al. 2005).....	19
Abbildung 3.1: Verschiedene Raummodelle für die Analyse von Punktbewegungen (Laube 2009).....	29
Abbildung 3.2: Vergleich der individuellen und der räumlichen Perspektive.....	33
Abbildung 4.1: <i>Verfolgen/Entfliehen</i> aus individueller Perspektive.....	37
Abbildung 4.2: Definition der Frontregion nach Andersson et al. (2008).....	38
Abbildung 4.3: Ausgang <i>Verfolgen/Entfliehen</i> – Variante 1: erfolgreiche Jagd.....	39
Abbildung 4.4: Ausgang <i>Verfolgen/Entfliehen</i> – Variante 2: erfolgreiche Flucht.....	39
Abbildung 4.5: Auslösung von <i>Verfolgen/Entfliehen</i> aus räumlicher Perspektive.....	41
Abbildung 4.6: Auslösung von <i>Verfolgen/Entfliehen</i> aus räumlicher Perspektive auf dem Strassennetzwerk.....	41
Abbildung 4.7: Bewegung in dieselbe Richtung während dem <i>Verfolgen/Entfliehen</i>	42
Abbildung 4.8: Raum-zeitlicher Ablauf einer <i>Konfrontation</i> aus der individuellen Perspektive.....	44
Abbildung 4.9: Zusammentreffen zwischen zwei Punktobjekten.....	46
Abbildung 4.10: Raum-zeitlicher Ablauf einer <i>Konfrontation</i> aus der räumlichen Perspektive.....	48
Abbildung 4.11: Raum-zeitlicher Ablauf einer <i>Konfrontation</i> aus der Netzwerkperspektive.....	49
Abbildung 4.12: Voraussetzungen für die <i>Konvergenz</i> zwischen Punktobjekten auf einem Segment.....	50
Abbildung 4.13: Raum-zeitlicher Ablauf von <i>Meiden</i> individuelle Perspektive (1).....	52
Abbildung 4.14: Raum-zeitlicher Ablauf von <i>Meiden</i> individuelle Perspektive (2).....	53
Abbildung 4.15: Raum-zeitlicher Ablauf von <i>Meiden</i> räumliche Perspektive.....	54
Abbildung 4.16: Raum-zeitlicher Ablauf von <i>Meiden</i> Netzwerk-Perspektive.....	55
Abbildung 5.1: Algorithmen zur Detektion von Reaktionsbewegungsmustern sowie ihre wichtigsten Elemente.....	56
Abbildung 5.2: Detektion von <i>Zusammensein</i> (individuelle Perspektive).....	58
Abbildung 5.3: Detektion von <i>Zusammensein</i> (räumliche Perspektive).....	59
Abbildung 5.4: Detektion von <i>Getrenntsein</i> (individuelle Perspektive).....	60
Abbildung 5.5: Detektion von <i>Getrenntsein</i> (räumliche Perspektive).....	60
Abbildung 5.6: Detektion von <i>Konvergenz/Divergenz</i> (individuelle Perspektive).....	61
Abbildung 5.7: Detektion von <i>Konvergenz/Divergenz</i> (räumliche Perspektive).....	62
Abbildung 5.8: Detektion von <i>Folgen</i> (individuelle Perspektive).....	62
Abbildung 5.9: Detektion von <i>Folgen</i> (räumliche Perspektive).....	63
Abbildung 5.10: Detektion einer <i>Intersektion</i> (individuelle Perspektive).....	64
Abbildung 5.11: Detektion einer <i>Intersektion</i> (räumliche Perspektive).....	64

Abbildung 5.12: Detektion von <i>Verfolgen/Entfliehen</i> (individuelle und räumliche Perspektive)	65
Abbildung 5.13: Detektion von <i>Konfrontation</i> (individuelle und räumliche Perspektive).....	66
Abbildung 5.14: Detektion von <i>Meiden</i> (individuelle Perspektive)	67
Abbildung 6.1: Euklidischer Raum (links) und Netzwerkraum (rechts) im Steering Behavior-Modell	70
Abbildung 6.2: Netzwerkraum in Repast Symphony	72
Abbildung 6.3: Karte des spät-mittelalterlichen Amsterdams (Waag Society 2007)	74
Abbildung 6.4: Spielerin im Hauptquartier (Waag Society 2007)	74
Abbildung 6.5: Struktur des Frequentie1550-Datensatzes	75
Abbildung 6.6: Beispiele von Interaktionsformen (Orellana et al. 2009).....	76
Abbildung 6.7: Trajektorien der Bewegungsdaten des Frequentie1550-Projekts (Orellana et al. 2009).....	77
Abbildung 6.8: Bounding Box des Stadtzentrums von Amsterdam (OpenStreetMap 2010)	77
Abbildung 6.9: Strassennetzwerk von Amsterdam in Repast Symphony	79
Abbildung 6.10: Histogramm der Lücken in den vorprozessierten Frequentie1550-Daten	81
Abbildung 7.1: Positives Beispiel einer als <i>Verfolgen/Entfliehen</i> klassierten Situation.....	94
Abbildung 7.2: Negatives Beispiel einer als <i>Verfolgen/Entfliehen</i> klassierten Situation	95
Abbildung 7.3: Beispiel einer als <i>Verfolgen/Entfliehen</i> klassierten Situation	96
Abbildung 7.4: Beispiel von zwei richtigerweise detektierten <i>Verfolgen/Entfliehen</i>	97
Abbildung 7.5: Beispiel einer fälschlicherweise als <i>Meiden</i> klassierten Situation.....	100

Diagramme

Diagramm 7.1: Einfluss der Verzögerungszeit auf den Algorithmus VED_i	86
Diagramm 7.2: Einfluss des Frontbereichswinkels auf den Algorithmus VED_i	86
Diagramm 7.3: Einfluss der Verzögerungszeit auf den Algorithmus VED_r	86
Diagramm 7.4: Einfluss der kritischen Zeitdauer von <i>Folgen</i> auf den Algorithmus VED_r	86
Diagramm 7.5: Einfluss der Verzögerungszeit auf den Algorithmus MD_i	90
Diagramm 7.6: Einfluss des kritischen Richtungswechselwinkels auf den Algorithmus MD_i	90
Diagramm 7.7: Einfluss der Verzögerungszeit auf den Algorithmus MD_r	90
Diagramm 7.8: Einfluss des kritischen Richtungswechselwinkels auf den Algorithmus MD_r	90
Diagramm 7.9: Einfluss der kritischen Distanz von <i>Folgen</i> auf den Algorithmus VED_i	93
Diagramm 7.10: Einfluss der Verzögerungszeit auf den Algorithmus VED_i	93
Diagramm 7.11: Einfluss der Frontbereichswinkels auf den Algorithmus VED_i	93
Diagramm 7.12: Einfluss der Grösse der betrachteten Segmentnachbarschaft auf den Algorithmus VED_r	93
Diagramm 7.13: Einfluss der Verzögerungszeit auf den Algorithmus VED_r	93
Diagramm 7.14: Einfluss der Verzögerungszeit auf den Algorithmus MD_i	98
Diagramm 7.15: Einfluss der Verzögerungszeit auf die Algorithmen MD_r	98
Diagramm 7.16: Einfluss des kritischen Winkels des Richtungswechsels auf den Algorithmus MD_i	99
Diagramm 7.17: Einfluss der Minimaldistanz von <i>Meiden</i> auf den Algorithmus MD_i	99
Diagramm 7.18: Einfluss des kritischen Winkels des Richtungswechsels auf den Algorithmus MD_r	99
Diagramm 7.19: Einfluss der Segment-nachbarschaftsgrösse auf den Algorithmen MD_r	99
Diagramm 7.20: Einfluss der kritischen Zeitdauer von <i>Getrenntsein</i> auf den Algorithmus KD_i	103
Diagramm 7.21: Einfluss der Verzögerungszeit auf den Algorithmus KD_i	103
Diagramm 7.22: Einfluss der kritischen Zeitdauer von <i>Getrenntsein</i> auf den Algorithmus KD_r	103
Diagramm 7.23: Einfluss der Verzögerungszeit auf den Algorithmus KD_r	103
Diagramm 7.24: Einfluss der kritischen Intersektionsdistanz auf den Algorithmus KD_i	104
Diagramm 7.25: Einfluss der kritischen Zeitdauer der <i>Intersektion</i> auf den Algorithmus KDr	104
Diagramm 7.26: Einfluss der kritischen Distanz von <i>Annähern</i> auf den Ansatz von Orellana et al. (2009)	107

Tabellen

Tabelle 5.1: Übersicht der in dieser Arbeit verwendeten Kürzel der Algorithmen	57
Tabelle 7.1: Auflistung der idealen Parameter des Algorithmus VED_i	85
Tabelle 7.2: Auflistung der idealen Parameter des Algorithmus VED_r	88
Tabelle 7.3: Auflistung der idealen Parameter des Algorithmus MD_i	89
Tabelle 7.4: Auflistung der idealen Parameter des Algorithmus MD_r	91
Tabelle 7.5: Auflistung der idealen Parameter des Algorithmus KD_i	102
Tabelle 7.6: Auflistung der idealen Parameter des Algorithmus KD_r	105
Tabelle 7.7: Auflistung der idealen Parameter des Ansatzes von Orellana et al. (2009).....	106
Tabelle 8.1: Auflistung der Einflüsse der wichtigsten Parameter der Algorithmen	112

1 Einleitung

1.1 Kontext und Motivation

Mobilität ist ein Schlüsselement von vielen Prozessen und Aktivitäten im geographischen Raum und das Verständnis von Bewegung ist wichtig für viele wissenschaftliche Forschungszweige und diverse Anwendungsbereiche (Dodge et al. 2008). Einhergehend mit der zunehmenden Verbreitung von modernen Lokalisierungstechnologien ist auch der starke Anstieg des Potentials und der technischen Möglichkeiten um Daten über Bewegungen zu sammeln (Laube und Purves 2006). Heutzutage ist es möglich, detaillierte Trajektorien von Bewegungen von Menschen, Tieren, Fahrzeugen und anderen sich bewegenden Objekten aufzuzeichnen und daraus neue Erkenntnisse der Prozesse abzuleiten und deren Verständnis zu verbessern (Andersson et al. 2008). Diese Entwicklung führte dazu, dass zusehends mehr Forscher von den unterschiedlichsten Wissenschaftsbereichen grosses Interesse an der Bewegungsmusteranalyse zeigen (Laube et al. 2007). Laut Gudmundsson et al. (2008) ist plötzlich nicht mehr die Datenlage das Problem, sondern das Fehlen von adäquaten Methoden zur Analyse dieser Daten. Es werden also effiziente Analysetechniken (z.B. Algorithmen oder visuelle Analysetools) gesucht, mit deren Hilfe die relevanten und nützlichen Informationen der Bewegungen aus grossen Datensätzen extrahiert werden können (Dodge et al. 2008).

Das Thema dieser Masterarbeit ist die Analyse von Bewegungsmustern, welche als Resultat von Interaktionen und Reaktionen zwischen zwei verschiedenen Arten von sich bewegenden Punktobjekten entstehen. Im Gegensatz zu bisherigen Forschungsanstrengungen im Forschungsfeld der Bewegungsmusteranalyse innerhalb der GIScience, welche sich vorwiegend mit Verhaltensmustern mit Beteiligung einer Art beschäftigten, zielt diese Arbeit in Richtung der Analyse von Interaktions- und Reaktionsbewegungsmustern zwischen zwei unterschiedlichen Arten von sich bewegenden Punktobjekten. Beispiele für bisherige Forschungsschwerpunkte sind die Analyse und Beschreibung von Bewegungsmustern wie die *Herdenbildung (flock)* (z.B. Gudmundsson et al. 2007) oder das *Anführen einer Gruppe (leadership)* (z.B. Andersson et al. 2008, Gudmundsson et al. 2007) und die Entwicklung von Algorithmen zu deren Detektion in raum-zeitlichen Datensätzen.

Im Gegensatz zu den genannten Mustern geht es in dieser Arbeit also um interagierende Punktobjekte von zwei unterschiedlichen Arten, welche sich durch ihr Verhalten gegenseitig beeinflussen, was zu speziellen Interaktions- beziehungsweise Reaktionsmustern führt. Solche Reaktionsbewegungsmuster sind beispielsweise in der Literatur der ökologischen Verhaltensforschung (z.B. Cooper et al. 2008, Laporte et al. 2010), sowie in Arbeiten aus den Bereichen Sicherheit und Überwachung (z.B. Szlavik et al. 2008, Mahajan et al. 2004) beschrieben. Für diese beiden und noch weitere Anwendungsbereiche werden Methoden zur automatisierten Detektion solcher Muster äusserst interessant sein. Diese Ausführungen motivieren, in dieser Arbeit einen Beitrag zu einem besseren Verständnis von Reaktionsbewegungsmustern zu leisten und zudem für eine Auswahl dieser Muster automatisierte Verfahren zu deren Entdeckung in grossen Bewegungsdatensätzen zu entwickeln.

1.2 Problemstellung und Zielsetzungen

Untersuchungsgegenstand dieser Arbeit sind Trajektorien von zwei unterschiedlichen Typen von sich bewegendem Objekten (werden in dieser Arbeit als rote und blaue Punktobjekte bezeichnet), welche in einem vorgegebenen Raum aufeinandertreffen und miteinander interagieren. Dabei beeinflussen sich die beteiligten Punktobjekte gegenseitig in ihren Aktionen, was zu einer Abfolge von Aktions- und Reaktionsbewegungen führt. Eine Auswahl solcher Verhaltensweisen soll mit Hilfe von treffenden Formalisierungen und der Entwicklung von Algorithmen als Muster in Bewegungsdaten erfolgreich erkannt werden. Für die Formalisierung stehen zwei unterschiedliche Betrachtungsweisen zur Verfügung. Während sich die individuelle Perspektive bei der Analyse der Bewegungsmuster auf die Individuen fokussiert, versucht die räumliche Perspektive die Muster aus Sicht der Räume (z.B. Strassensegmente), welche den Bewegungsmustern unterliegen, zu interpretieren. Eine grosse Herausforderung dieser Arbeit ist, dass Reaktionsbewegungsmuster mit Hilfe der entworfenen Formalisierungen und Algorithmen von anderen Interaktionsformen unterschieden werden können.

Aus dieser Problemstellung ergeben sich die folgenden Zielsetzungen für diese Masterarbeit:

- Erstellung einer Übersicht der in der Literatur diskutierten Interaktions- und Reaktionsmuster zwischen zwei Arten von sich bewegendem Punktobjekten (z.B. zwischen zwei Tieren von unterschiedlichen Arten).
- Formalisierung der drei ausgewählten Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden*, wobei sowohl die individuelle als auch die räumliche Perspektive verwendet werden sollen.
- Entwicklung von Algorithmen zur Detektion der drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* in Bewegungsdaten, basierend auf den ausgearbeiteten Formalisierungen.
- Untersuchung der Anwendbarkeit der Algorithmen anhand von Experimenten mit simulierten und realen Bewegungsdaten. Speziell interessant ist dabei die Überprüfung, ob eine Abgrenzung zwischen den ausgewählten Reaktionsbewegungsmustern und anderen Interaktionsformen erreicht werden kann.

1.3 Forschungsfragen

Aus der Zielsetzung lassen sich die nachfolgenden Fragestellungen für diese Masterarbeit ableiten:

- FF1: Wie können, basierend auf den in der Literatur vorhandenen Definitionen, die Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* sowie *Meiden* formalisiert werden?
- FF2: Wie lassen sich diese Formalisierungen in Algorithmen umsetzen, um die definierten Reaktionsbewegungsmuster in raum-zeitlichen Datenreihen zu detektieren?

FF3: Welches sind die entscheidenden Parameter für eine erfolgreiche Detektion der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* und wie gross ist deren Einfluss für die Abgrenzung zu verwandten Bewegungsmustern?

FF4: Inwiefern unterscheiden sich die Resultate der Algorithmen der individuellen Perspektive von denjenigen der räumlichen Perspektive?

1.4 Aufbau der Arbeit

Der Aufbau dieser Arbeit richtet sich stark am Arbeitsablauf und an den durchgeführten Arbeitsschritten. Im nächsten Kapitel folgen Ausführungen zum wissenschaftlichen Hintergrund, wobei zuerst auf die bis anhin in der GIScience gemachten Anstrengungen im Bereich der Bewegungsmusteranalyse eingegangen wird und anschliessend mögliche Anwendungsbereiche diskutiert werden. Ebenfalls werden in Kapitel 2 die Erkenntnisse des Literaturstudiums zu den verschiedenen in der Disziplin der Tierverhaltensforschung beschriebenen Bewegungsmustern in Interaktions- und Reaktionsbeziehungen zwischen zwei unterschiedlichen Arten von beweglichen Punktobjekten aufgeführt. Dabei werden die verschiedenen Reaktionsbewegungsmuster sinnvoll gegliedert und in Form einer Übersicht kurz beschrieben. Im dritten Kapitel wird die Problemstellung im Detail beschrieben und definiert. Ziel dieser Ausführungen ist es primär, die zu untersuchenden Aspekte einzugrenzen und den Fokus der Arbeit festzulegen. In Kapitel 4 folgen abgeleitet aus den Beschreibungen zu den Reaktionsbewegungsmustern in Kapitel 2 die Formalisierungen der drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden*. Aufbauend auf diesen formalen Beschreibungen werden in Kapitel 5 für diese drei Reaktionsbewegungsmuster Algorithmen entwickelt, welche dann in Kapitel 6 mit Hilfe von Experimenten mit simulierten Bewegungsdaten und einem realen Datensatz auf deren Anwendbarkeit hin getestet werden. Die Resultate dieser Experimente werden in Kapitel 7 beschrieben und in Kapitel 8 diskutiert. Die Schlussfolgerungen dieser Masterarbeit sind in Kapitel 9 zu finden.

2 Hintergrund

In diesem Kapitel soll auf die wissenschaftlichen Hintergründe der in der Einleitung beschriebenen Aufgabenstellung und Zielsetzungen dieser Arbeit eingegangen werden. Dabei wird das Thema dieser Arbeit in einem ersten Abschnitt in den wissenschaftlichen Kontext und die Geschichte der Forschungsdisziplin der Bewegungsmusteranalyse in Raum-Zeit-Daten eingebettet und die jüngsten Entwicklungen und Erkenntnisse der Wissenschaft aufgezeigt, bevor exemplarisch zwei mögliche Anwendungsbereiche der in dieser Arbeit vorgestellten Methoden aufgegriffen und diskutiert werden. Anschliessend sollen die in der Literatur bekannten Reaktionsbewegungsmuster in Form einer Übersicht beschrieben werden. Die Ausarbeitung der wichtigsten Forschungslücken sowie die Diskussion der Forschungsfragen bilden den Abschluss dieses Kapitels, bevor dann in Kapitel 3 die Problemstellung genauer präzisiert und der Untersuchungsgegenstand der Arbeit eingegrenzt wird.

2.1 Bewegungsmusteranalyse in der GIScience

Die Analyse von Bewegungsmustern in raum-zeitlichen Daten ist ein relativ junges Forschungsfeld mit einer kurzen Geschichte (Gudmundsson et al. 2008). Dies ist auf zwei Hauptursachen zurückzuführen (Gudmundsson et al. 2008). Auf der einen Seite hatte die ganze Disziplin der Geographischen Informationswissenschaften (GIScience), welche sich aus der statischen Kartographie heraus entwickelte, lange mit grossen Problemen im Umgang mit Dynamik zu kämpfen (Gudmundsson et al. 2008). Als zweiten Grund für die späte Etablierung des Forschungsfeldes der Bewegungsmusteranalyse in Raum-Zeit-Daten als Subdisziplin der GIScience können die bis vor kurzem fehlenden technologischen Möglichkeiten zur Aufzeichnung von Bewegungen genannt werden (Gudmundsson et al. 2008). Mit dem Aufkommen von modernen Lokalisierungstechnologien haben sich das Potential und die technischen Möglichkeiten um Daten über raum-zeitliche Bewegungen zu sammeln jedoch stark und sprunghaft verbessert (Laube und Purves 2006). Es wurde plötzlich möglich, detaillierte Trajektorien von Bewegungen von Menschen, Tieren, Fahrzeugen und weiteren sich bewegenden Objekten aufzuzeichnen und daraus neue Erkenntnisse der involvierten Prozesse abzuleiten und deren Verständnis zu verbessern (Andersson et al. 2008). Dadurch wurde eine neue Ära der Bewegungsanalyse eingeläutet, an welcher Forschende von sehr vielen unterschiedlichen Wissenschaftsbereichen zusehend grosses Interesse zeigten (Laube et al. 2007). Dazu gehören laut Andersson et al. (2008) neben Forschenden aus der Geographie (z.B. Frank 2001) auch Wissenschaftler aus den Disziplinen der Datenbankforschung (z.B. Güting et al. 2000), Tierverhaltensforschung (z.B. Hulbert 2001), Überwachung und Sicherheit (z.B. Ng 2001, Thomas und Cook 2006), Transportanalysen (z.B. Kwan 2000) und Marktforschungen (z.B. Qu et al. 1998). Innerhalb weniger Jahre entwickelte sich die Lage also von einem notorischen Bewegungsdatendefizit hin zu einer Situation, in welcher man beinahe schon von einem Datenüberschuss sprechen kann (Gudmundsson et al. 2008). Plötzlich ist nicht mehr die Datenlage das Problem, sondern das Fehlen von adäquaten Analysemethoden (Gudmundsson et al. 2008). Dies hat vor allem damit zu tun, dass die

Bewegungsdaten von beweglichen Punktoobjekten in der Regel sehr umfangreich und komplex sind (Dodge et al. 2008). Aus diesem Grund gilt es heute, effiziente Data Mining Algorithmen und visuelle Analysetechniken zu entwickeln, um die relevanten und nützlichen Informationen über die Bewegungsmuster aus den Daten extrahieren zu können (Dodge et al. 2008). Im Bereich der explorativen Datenanalyse, welche versucht, die Kapazitäten von Computern mit den exzellenten Fähigkeiten des Menschen zur Entdeckung von vermuteten, aber auch unerwarteten Sachverhalten in graphischen Repräsentationen zu kombinieren (Gudmundsson et al. 2008), gibt es verschiedene mögliche Methoden für Bewegungsmusteranalysen. Diese werden hier nur kurz aufgeführt, da sich diese Arbeit auf die Entwicklung von Algorithmen zur Detektion von Reaktionsbewegungsmustern fokussiert und eine ausführliche Diskussion der bekannten Visualisierungstechniken den Rahmen der Arbeit sprengen würde. Zu diesen Visualisierungstechniken gehören zwei-dimensionale Karten mit Darstellungen von Trajektorien, Animationen von Bewegungsabläufen von Individuen und Gruppen und die Verwendung des so genannten Raum-Zeit-Aquariums als Erweiterung der zweidimensionalen Karte mit einer orthogonalen Zeitachse für die Entdeckung von Bewegungsmustern (Gudmundsson et al. 2008). Forschungsanstrengungen im Bereich Data Mining und Computational Geometry versuchen, mit Hilfe von spezifischen Algorithmen zur Extraktion von Mustern aus Daten einen wichtigen Beitrag im Prozess der Wissensgenerierung aus Datenbanken („Knowledge Discovery in Databases (KDD)“) zu leisten (Dodge et al. 2008). Als wichtiger Grundstein der Bewegungsmusteranalyse im Bereich Data Mining kann die Erarbeitung und die Implementierung des REMO-Framework (RElative MOtion) gesehen werden, welches verschiedene Bewegungsparameter (Geschwindigkeit, Beschleunigung und Bewegungsrichtung) vorschlägt, mit Hilfe deren ähnliche Verhaltensweisen innerhalb von Gruppen von Objekten analysiert werden können (Laube 2005, Laube und Imfeld 2002, Laube et al. 2005, Laube et al. 2004). Siehe dazu die Abbildung 2.1.

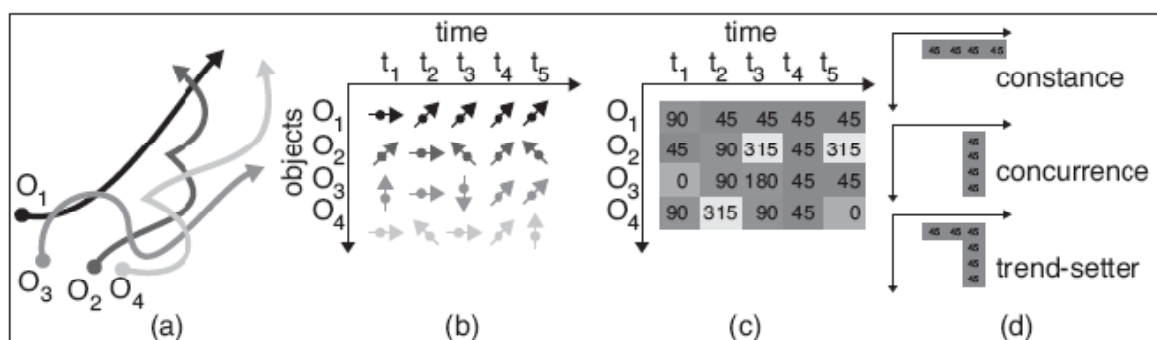


Abbildung 2.1: REMO-Framework (Laube et al. 2004)

Zur Veranschaulichung des REMO-Frameworks sind in dieser Abbildung die Trajektorien („geospatial lifelines“) von vier Punktoobjekten (a), die daraus abgeleiteten Bewegungsrichtungen (b), die REMO Analysematrix (c) sowie die zugeordneten generischen Bewegungsmuster (d) dargestellt.

Basierend auf diesem Framework konnten verschiedene raum-zeitliche Muster definiert und Algorithmen zu deren Detektion entwickelt werden (Laube et al. 2004). Dazu gehören die Bewegungsmuster *Herdenbildung (flock)*, *Führungsverhalten (leadership)*, *Konvergenz (convergence)* und *Zusammentreffen (encounter)* (Laube et al. 2004). Das Muster *Konvergenz* ist in der Abbildung

2.2 veranschaulicht. Die Definition dieser Bewegungsmuster lösten diverse weitere Forschungsarbeiten aus, wie beispielsweise

- das Bestreben diese vier Muster durch effiziente Approximationsalgorithmen zu detektieren (Gudmundsson et al. 2007),
- die Entwicklung von Algorithmen um *Herdenbildungen* in Gruppen von sich bewegenden Punktobjekten zu entdecken (Benkert et al. 2006, Gudmundsson und van Kreveld 2006),
- erste Ansätze um solche *Herdenbildungen* in Wireless Sensor Networks zu erkennen (Laube et al. 2008),
- oder ein Ansatz zur Erkennung von sich in einer Reihe bewegenden Punktobjekten (*Single File*) (Buchin et al. 2008).

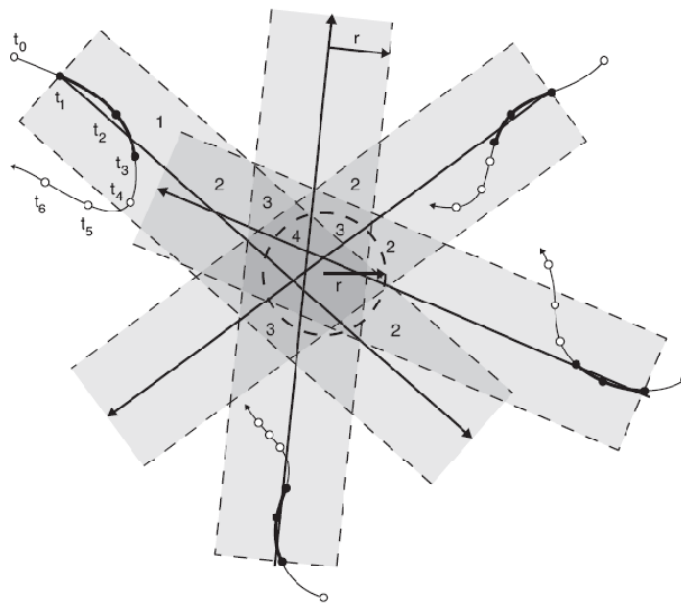


Abbildung 2.2: Bewegungsmuster *Konvergenz* (Laube et al. 2004)

Geometrische Detektion des Musters *Konvergenz*. Das dunkelgraue Polygon zeigt das Gebiet an, in welchem die Bewegungsvektoren der vier beweglichen Punktobjekte zusammentreffen und zwar in einem Umkreis mit einem kritischen Radius r .

Eine weitere wichtige Arbeit in der Forschungsgeschichte der Bewegungsmusteranalyse stellt die Taxonomie von Dodge et al. (2008) dar. Aufgrund der geringen Einigkeit betreffend der Definitionen von verschiedenen Bewegungsmustern haben Dodge et al. (2008) eine Taxonomie zur Klassifikation von verschiedenen Bewegungsverhalten von unterschiedlichen beweglichen Punktobjekten ausgearbeitet. Dabei wurden alle relevanten Bewegungsmuster klassiert und definiert, wobei auch Bewegungsmuster mit Beteiligung von zwei interagierenden Punktobjekten unterschiedlichen Typs genannt werden (Dodge et al. 2008). Interaktionen zwischen sich bewegenden Punktobjekten stehen auch in der Arbeit von Orellana und Renso (2010) im Fokus, wobei eine Ontologie von Interaktionen zur Charakterisierung von Bewegungsverhalten von Fußgängern vorgestellt wird. Unter Verwendung dieser Ontologie wird versucht, Bewegungsdaten mit Hilfe der Betrachtung der stattgefundenen Interaktionen zu interpretieren. Für die Analyse von Interaktionen sind verschiedenste Techniken von explorativen Analysetools bis hin zu Algorithmen zur Detektion von Bewegungsmustern vorgesehen.

Dabei gilt es anzumerken, dass bis anhin noch nicht für alle Interaktionsformen geeignete Methoden zur Verfügung stehen. In Orellana et al. (2009) werden aus diesem Grund Ansätze zur Entdeckung der Interaktionsmuster *Annäherung* (siehe Abbildung 2.3), *Anhalten* sowie *Anziehung* in Bewegungsdaten von Schülern, die an einem Freiluftspiel teilnahmen, präsentiert. *Anhaltebewegungen* werden zudem in Orellana und Wachowicz (2011) noch genauer unter die Lupe genommen und ein geeignetes Verfahren zu deren Detektion in Bewegungsdaten von Touristen in einem Nationalpark sowie von Kindern in einem Spiel vorgestellt.

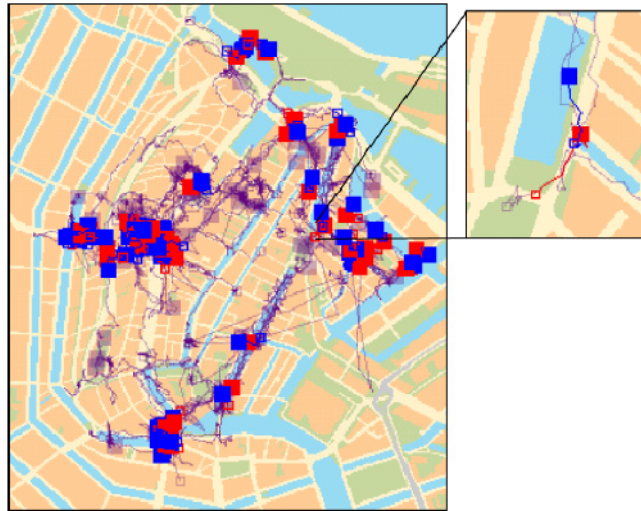


Abbildung 2.3: Detektion von *Approximationen* (Orellana et al. 2009)

Dargestellt in dieser Abbildung sind *Annäherungen* (*Approximationen*) zwischen zwei Teams (rot und blau), welche am Freiluftspiel Frequentie1550 teilnahmen (links). Eine solche *Annäherung* ist zudem in der Detailansicht dargestellt (rechts).

Dieser kurze geschichtliche Rückblick der Disziplin und der Überblick der bis anhin in der GIScience unternommenen Anstrengungen im Forschungsfeld der Bewegungsmusteranalyse zeigen, dass man sich bis heute grösstenteils mit der Analyse und Detektion von Mustern eines Typs von beweglichen Punktobjekten auseinandergesetzt hat. Diese Arbeit soll genau an diesem Punkt ansetzen und einen Schritt in Richtung der Analyse von Mustern mit Beteiligung von zwei Typen von sich bewegenden Punktobjekten machen. Interessant sein werden solche Methoden für die automatisierte Detektion von Reaktionsbewegungsmustern für verschiedene wissenschaftliche Disziplinen und Anwendungsbereiche. Im nächsten Abschnitt wird eine Übersicht solcher Anwendungsbereiche vorgestellt sowie exemplarisch zwei solche Bereiche aufgegriffen und kurz diskutiert.

2.2 Anwendungsbereiche

Die Ausführungen in diesem Abschnitt basieren auf den Erläuterungen von Gudmundsson et al. (2008). Die Analysen von Bewegungsmustern mit Beteiligung von sich bewegenden Punkten desselben Typs können für verschiedene andere Bereiche sehr interessant sein. Dazu gehören beispielsweise die Verhaltensforschung von Tieren, die Untersuchung von Menschenbewegungen, das Verkehrsmanagement, die Analyse von Sportszenen oder auch für die Nutzung zu Überwachungs- und Sicherheits-Zwecken. Im Falle der Tierverhaltensforschung kann mit Hilfe der Analyse von

Bewegungsmustern versucht werden, verschiedene Aspekte von Verhaltensweisen zu untersuchen und zu verstehen. Dabei ist es unter anderem möglich, die von Tieren bevorzugten Lebensräume zu bestimmen oder soziale Interaktionen, wie beispielsweise das Führungsverhalten in Tiergruppen, zu analysieren. Erkenntnisse aus Analysen von Menschenbewegungen können in die Planung von Städten und öffentlichen Infrastrukturen (z.B. Autobahnen) einfließen. Ebenfalls kann die Bewegungsmusteranalyse einen Beitrag leisten im Bereich des Verkehrsmanagements, wobei unvorhergesehene oder sogar gefährliche Konstellationen (z.B. Verkehrsstau) von sich bewegenden Objekten erkannt werden können. Im Bereich Überwachung und Sicherheit gibt es diverse Datenquellen, wie beispielsweise Videos von Überwachungskameras, GPS-Daten oder Koordinaten von Mobiltelefonen, welche Möglichkeiten zur Analyse von Bewegungsmustern bieten. Durch die Untersuchung dieser Bewegungsdaten kann versucht werden auffällige Verhaltensweisen zu entdecken und dadurch mögliche Verbrechen zu verhindern. Des Weiteren können Bewegungsmusteranalysen von Sportveranstaltungen zum Beispiel einem Trainer wichtige Hinweise über die gegnerischen Verhaltensweisen und Strategien liefern oder einem Schiedsrichter bei der Offsidefrage Unterstützung bieten.

Nun stellt sich die Frage, wie die Ausdehnung der Bewegungsmusteranalyse auf Bewegungsabläufe mit beteiligten Punktobjekten von zwei verschiedenen Arten in Anwendungsbereichen ausserhalb der GIScience genutzt werden können. In den folgenden Abschnitten werden die in Gudmundsson et al. (2008) aufgeführten Anwendungsbereiche Überwachung und Sicherheit sowie Tierverhaltensforschung exemplarisch aufgegriffen und hinsichtlich der Nutzung der Analyse von Reaktionsbewegungsmustern untersucht.

2.2.1 Überwachung und Sicherheit

Während Überwachungskameras zu Sicherheitszwecken immer billiger werden und dadurch eine weite Verbreitung aufweisen, bleibt der menschliche Einsatz zur Analyse der aufgenommenen Bilder weiterhin teuer (Ko 2008). Dies ist vor allem problematisch, da die Überwachungssysteme heute täglich grosse Mengen an Daten generieren, welche möglichst effizient analysiert werden sollen (Morris und Trivedi 2008). Überwachungskameras sollen aus diesem Grund in Zukunft nicht nur ein passives Werkzeug zur Aufzeichnung von Bewegungen sein, sondern ein wichtiges Hilfsmittel werden zur automatischen Detektion von Ereignissen und zur aktiven Planung von Massnahmen in Echtzeit, was ihren Einsatz im Sicherheitsbereich noch viel effizienter machen wird (Ko 2008). Das Ziel einer automatisierten visuellen Überwachung ist es also, eine Beschreibung der Geschehnisse in einem überwachten Gebiet zu generieren und basierend auf deren Interpretation adäquate Massnahmen zu treffen (Ko 2008). Dabei ist die Fähigkeit eines Überwachungssystems zur Analyse von menschlichen Bewegungen und Aktivitäten aus den Kamerabildern einer der zentralen Aspekte der automatischen visuellen Überwachung (Ko 2008). Das Erkennen und Verstehen von solchen Bewegungen involviert einerseits Beschreibungen von Aktionen von Objekten und Interaktionen zwischen diesen und andererseits die Analyse und Erkennung von Bewegungsmustern (Ko 2008). In automatischen visuellen

Überwachungssystemen werden solche künstlichen Intelligenztechniken angewendet, um Expertenentscheidungen zu ersetzen (Ko 2008). Für die Anwendung solcher Techniken sind diverse wichtige Vorbereitungsschritte notwendig, wie beispielsweise das Tracking, die Detektion und die Klassifikation von Objekten oder aber die Detektion und Extraktion von Bewegungsinformationen aus aufbereiteten und vorprozessierten Bildsequenzen (Ko 2008). Bei der Bestimmung der Bewegungsinformationen können grundsätzlich drei verschiedene Methoden zur Anwendung kommen, wobei neben Techniken zur Extraktion von optischen Flüssen in Bildsequenzen (optical flow objects) und regions- bzw. bildbasierten Objekten (region- or image-based features) auch Methoden zur Herausfilterung von Trajektorien angewendet werden können (Cedras und Shah 1995). Solche Trajektorien beschreiben den Bewegungspfad eines Objekts (Ko 2008) und erlauben die Analyse von Aktivitäten und Verhaltensformen sowie die Identifikation von interessanten Events (Morris und Trivedi 2008). Zu solchen Aktivitäten gehören auch ungewöhnliche Verhaltensweisen, welche vor allem aus Sicht der Sicherheitsüberwachung interessant sind und welche es zu erkennen gilt (Morris und Trivedi 2008). Solche ungewohnten Aktivitäten, wie beispielsweise eine 360 Grad-Drehung eines Autos auf einer Strasse (Siehe Abbildung 2.4), können mit Hilfe von sinnvoll gesetzten Grenzwerten in den Trajektorien entdeckt werden (Morris und Trivedi 2008).

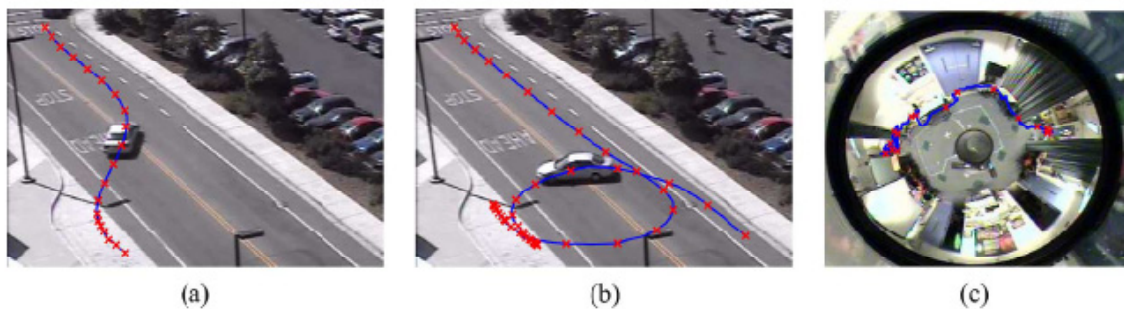


Abbildung 2.4: Entdeckung von ungewöhnlichen Ereignissen in Trajektorien (Morris und Trivedi 2008)

Dargestellt in dieser Abbildung sind ungewöhnliche Verhaltensweisen, welche in Trajektorien entdeckt werden können: Ein Auto, welches die Mittellinie überquert (a), eine 360°-Drehung eines Autos (b) sowie eine Person, welche den Wänden eines Raumes entlanggeht.



Abbildung 2.6: Beispiele von Kämpfen zwischen Personen (Szlavik 2008)

Dargestellt in dieser Abbildung sind Personen, welche sich bekämpfen. Solche Verhaltensweisen gilt es in Bildern von Überwachungskameras zu entdecken.



Abbildung 2.5: Ausrauben (Mahajan et al. 2004)

Dargestellt in dieser Abbildung ist eine Situation, in welcher die Person im weissen Hemd ein Rucksack stiehlt.

Weiter kann unter Verwendung von automatischen Analysen von Trajektorien versucht werden, Interaktionen zwischen Objekten aufzudecken (Morris und Trivedi 2008). Die Detektion von solchen Verhaltensweisen basiert auf Modellierungen und Klassifikationen von Aktivitäten und Berücksichtigung von bestimmten Regeln (Cohen et al. 2008). Typische Beispiele für Bewegungsmuster im Sicherheitsbereich sind *Kämpfen* (Szlavik et al. 2008) und *Ausrauben* (Mahajan et al. 2004). Diese beiden Verhaltensweisen sind in den Abbildungen 2.5 und 2.6 dargestellt. Weitere in Videos von Überwachungskameras entdeckte Verhaltensweisen von Menschen sind in der CASIA Behavior Database (CASIA 2005) aufgeführt.

Die Modellierung und Klassifikation von menschlichen Aktivitäten ist alles andere als leicht, nicht zuletzt aufgrund der komplexen Natur und der grossen Vielfalt von Bewegungen (Ko 2008). Je nach Ort, Umwelt und beteiligter Aktionspartner von Interaktionssituationen kommt es zu sehr unterschiedlichen Interaktionsformen (Morris und Trivedi 2008). Genau an dieser Stelle könnten die in dieser Arbeit vorgestellten Definitionen von Reaktionsbewegungsmustern, welche für den Sicherheitsbereich relevant sind, Anwendung finden. Dazu gehören beispielsweise das Interaktionsmuster von *Verfolgen/Entfliehen* oder das Verhalten in Kampfsituationen (*Konfrontationen*). Aufgrund der in Cedras (1995) beschriebenen Möglichkeit zur Generierung von Trajektorien aus Bildsequenzen von Überwachungskameras scheint es durchaus möglich, dass die in Kapitel 5 beschriebenen Algorithmen zur Detektion von Reaktionsbewegungsmustern in Trajektorien im Sicherheitsbereich eingesetzt werden könnten. Die Erkenntnisse dieser Arbeit und zukünftiger Studien zum Thema Reaktionsbewegungsmusteranalyse sind also interessant für die Sicherheitsbranche als ein breites Anwendungsgebiet mit vielen unterschiedlichen betroffenen Einsatzorten. Dazu zählen unter anderem die Überwachung von Häusern und Wohnquartieren, die Prävention von Kriminalität und Verbrechen, die Beobachtung der Sicherheit in öffentlichen Infrastrukturen, das Monitoring von Strassen und Parkgaragen und anderen öffentlichen Plätzen (Morris und Trivedi 2008).

2.2.2 Biologie – Verhaltensforschung von Tieren

Neben den bis anhin in der GIScience analysierten Bewegungsmustern wie *Herdenbildung (flock)*, *Führungsverhalten (leadership)*, *Konvergenz (convergence)* und *Zusammentreffen (encounter)* (Laube et al. 2004), können auch in der Literatur der ökologischen Verhaltensforschung Untersuchungen von Verhaltensmustern von interagierenden Punktobjekten gefunden werden. Darunter sind auch einige Muster mit Beteiligung von zwei unterschiedlichen Typen von Punktobjekten. In diesem Abschnitt sollen exemplarisch ein paar solche Reaktionsbewegungsmuster aufgegriffen werden und gezeigt werden, inwiefern automatisierte Methoden zu deren Detektion in Bewegungsdaten einen Mehrwert für das jeweilige Anwendungsgebiet schaffen können.

Cooper et al. (2008) erforschten mit Hilfe von GPS-Daten die Verbreitungs- und die Interaktionsmuster von Vieh und Reh. Erkenntnisse aus solchen Studien sind von grossem Interesse für die Bewirtschaftung von Weideland, in welcher in den letzten Jahren ein Paradigmenwechsel hin

zu einer möglichst naturnahen Produktion mit verschiedenen Tierarten auf derselben Fläche stattgefunden hat (Cooper et al. 2008). Im Speziellen für eine erfolgreiche Diversifikation der Artenvielfalt kombiniert mit einer nachhaltigen Nutzung sind neue Erkenntnisse zu Interaktionsmustern und deren Auswirkungen auf Verbreitungsgebiete von grosser Bedeutung (Cooper et al. 2008). Denn interspezifische Interaktionen haben einen grossen Einfluss auf die Flächennutzung, falls sich Tiere unterschiedlicher Arten um dieselben Ressourcen oder Habitate konkurrenzieren (Cooper et al. 2008). Die Untersuchungen von Cooper et al. (2008) konnten beispielsweise zeigen, dass Rehe die räumliche Nähe zu Vieh-Tieren *Meiden*, was eine starke zeitliche Separation zwischen den beiden Tieren zur Folge hat. Unter Berücksichtigung solcher Erkenntnisse kann mit Hilfe einer artenspezifischen Bewirtschaftung in den passenden Gebieten die jeweilige Tierproduktion maximiert werden (Cooper et al. 2008). Mit der Verwendung von automatisierten Methoden zur Detektion von solchen territorialen Interaktionen (z.B. *Meiden* zwischen Tieren) kann ein wichtiger Beitrag zu einer effizienteren Tierproduktion mit verschiedenen Arten auf der gleichen Fläche geleistet werden. Dadurch kann verhindert werden, dass Tierarten von anderen verdrängt werden.

Weiter kann mit Hilfe neuer Erkenntnisse der räumlichen Verbreitung und Interaktionen von sich in gleichen Territorien aufhaltenden und sich gegenseitig konkurrenzierenden Tierarten ein wichtiger Betrag zur Verbesserung von Tierschutzmassnahmen geleistet werden (Moorcroft 2008). In Afrika beispielsweise wurden Bestrebungen zum Schutze des afrikanischen Wildhundes gestört durch die kämpferischen Interaktionen mit Löwen und Hyänen (Moorcroft 2008). Dadurch wurde verhindert, dass sich die afrikanischen Wildhunde in den für sie vorgesehenen Lebensräumen etablieren konnten (Moorcroft 2008). Diese Erläuterungen zeigen, dass unter Verwendung von Methoden zur automatisierten Entdeckung von solchen kämpferischen Interaktionen zwischen Tieren der Erfolg von Massnahmen zur Artenerhaltung erhöht werden kann.

Ein letztes Beispiel betrifft eine Räuber/Beute-Interaktion und zwar diejenige zwischen Wölfen und Viehherden. Wie in Laporte et al. (2010) beschrieben überschneiden sich die Lebensräume von Wölfen häufig mit denjenigen des Viehs, was zu regelmässigen Angriffen auf die menschlichen Nutztiere führt. Anhand der Analyse der mit GPS-Sensoren ausgerüsteten Viehherden konnte gezeigt werden, dass Vieh-Tiere versuchen ihren Räubern auszuweichen (Laporte et al. 2010). Solche Antiräuberstrategien konnten auch durch die Analysen von Clark und Johnson (2009) bestätigt werden, was eine wertvolle Erkenntnis für die Viehhalter darstellt. Clark und Johnson (2009) konnten in den aufgezeichneten GPS-Daten auch Fluchtbewegungen des Viehs als sehr schnelle lineare Bewegungen erkennen. Mit Hilfe von automatisierten Methoden zur Detektion von solchen Strategien des Viehs können Viehhalter unterstützt werden bei der Planung von Schutzmassnahmen der Viehherden vor Räuberangriffen. Denkbar wäre beispielsweise die Entwicklung eines Frühwarnsystems, welches bei einem Angriff auf das Vieh anhand der Untersuchung der Bewegungspfade die Situation richtig interpretieren kann und Alarm gibt.

Dieser Abschnitt zeigt, dass ein besseres Verständnis von Reaktionsbewegungsmustern und die Entwicklung von Algorithmen zur Entdeckung von solchen Mustern in Bewegungsdaten auch für die biologische Verhaltensforschung sehr wertvoll sind. Im folgenden Abschnitt soll anhand einer Literaturübersicht der biologischen Verhaltensforschung aufgezeigt werden, welche Formen von Reaktionsbewegungsmustern bis anhin bekannt sind.

2.3 Übersicht Reaktionsbewegungsmuster

In der Folge werden die wichtigsten in der Literatur der Tierverhaltensforschung bekannten und diskutierten Reaktionsbewegungsmuster vorgestellt. Dabei gilt es festzuhalten, dass diese Auflistung keinesfalls den Anspruch nach Vollständigkeit hat. In den folgenden Abschnitten wird nur ein Überblick der in der Biologie bekannten Muster gegeben. Für eine vollständige Übersicht müssten auch noch andere Anwendungsbereiche (z.B. Überwachung und Sicherheit) unter die Lupe genommen werden. Da dies den Rahmen dieser Masterarbeit sprengen würde, wurde das Literaturstudium auf die Tierverhaltensforschung beschränkt. In den nachfolgenden Abschnitten werden die Reaktionsbewegungsmuster aus Sicht der Biologie beschrieben. In Kapitel 4 wird das Hauptaugenmerk dann auf der raum-zeitlichen Ausprägung der drei ausgewählten Bewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* liegen und es wird versucht, diese Merkmale in eine sinnvolle Formalisierung umzusetzen.

In der Forschung der Geographischen Informationswissenschaften sind, wie zu Beginn dieses Kapitels beschrieben, bis anhin wenig Anstrengungen unternommen worden hinsichtlich der Beschreibung und Definition von Bewegungsmustern zwischen sich bewegenden Punktobjekten unterschiedlicher Art. Eine der Ausnahmen bildet die Taxonomie von Dodge et al. (2008), welche in Form einer Übersichtsklassifikation neben den bekannten Bewegungsmustern wie beispielsweise die *Formation einer Herde (flock)* oder das *Anführen einer Gruppe (leadership)*, auch Interaktionsformen zwischen zwei unterschiedlichen Arten aufführt (siehe Abbildung 2.7).

Dazu gehören die Wechselbeziehung von *Verfolgen/Entfliehen (pursuit/evasion)*, die Bewegungsabfolge von *Angriff und Verteidigung in Kampfsituationen (fighting)*, das *Balzverhalten (courtship)* in Situationen mit Beteiligung von Männchen und Weibchen einer Spezies, sowie das *Spielen (play)* als intraspezifische Interaktionsform zum Erlernen von Verhaltensweisen (z.B. durch Jungtiere), welche alle in Blythe et al. (1996) eingehend diskutiert und beschrieben werden. Ebenfalls in der Taxonomie von Dodge et al. (2008) aufgeführt ist die *elterliche Verteidigung (parental protection)*.

Weiter ist in der Literatur der ökologischen Verhaltensforschung das *Meiden* von möglichen Interaktionspartnern (*avoidance*) bekannt (Cooper et al. 2008). Diese Verhaltensweise konnte anhand von Studien der Interaktion zwischen Vieh und Reh mit Hilfe der Analyse von GPS-Daten eingehend unter die Lupe genommen werden (Cooper et al. 2008).

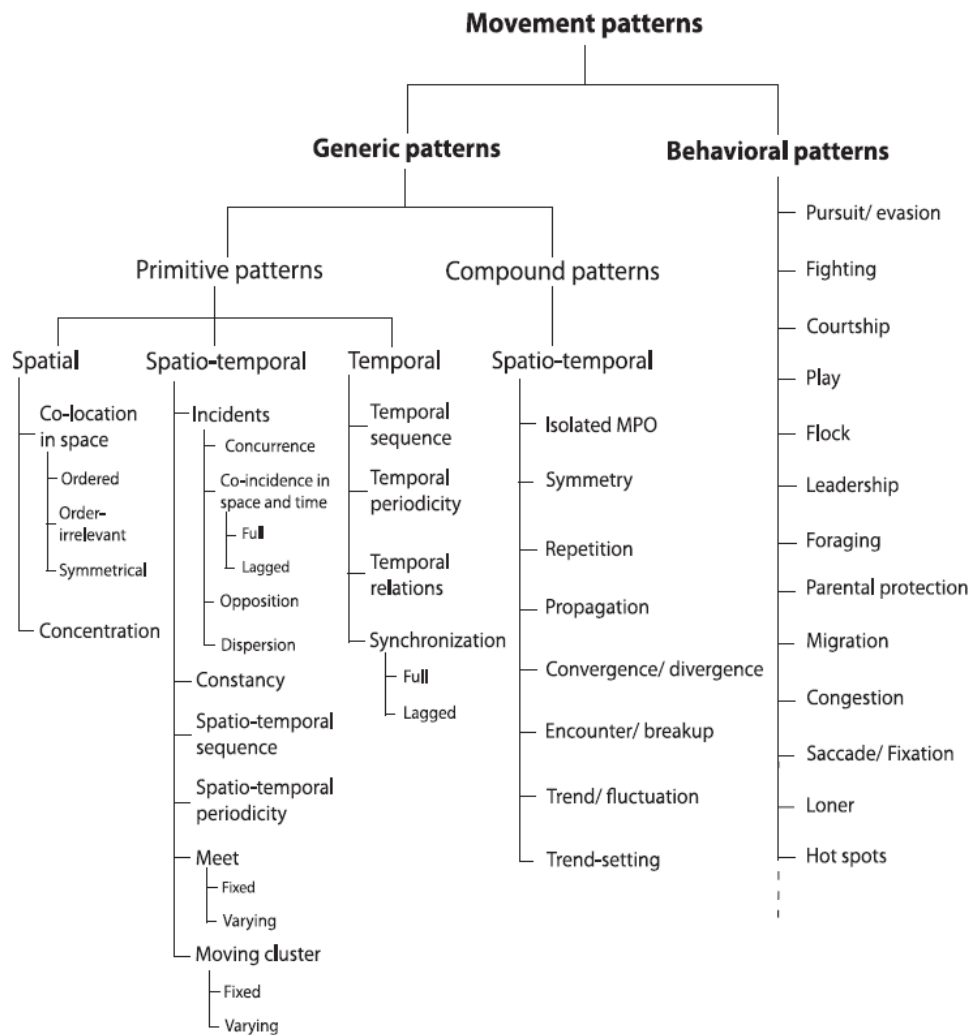


Abbildung 2.7: Klassifikation von Bewegungsmustern (Dodge et al. 2008)

Gliederung der bekannten Bewegungsmuster in generische Muster und Verhaltensmuster. Zu den Verhaltensmustern gezählt werden auch einige Reaktionsbewegungsmuster wie z.B. *Verfolgen/Entfliehen* oder *Kämpfen*.

Ebenfalls bekannt sind die Verhaltensmuster des Eindringens in ein Territorium, beziehungsweise der Verteidigung dessen (Barrett et al. 2005). Im Falle von Konkurrenz um Ressourcen in einem Lebensraum kann es beispielsweise zu einem Bewegungsmuster von *Eindringen und Verteidigen* eines Habitats kommen (Barrett et al. 2005).

Neben den Reaktionsmustern *Meiden* und *Verteidigen* gibt es auch die Möglichkeit, dass Punktobjekte auf Konkurrenzsituationen durch *Toleranz* auf ihrer Mitstreiter reagieren (Schoener 1983). *Toleranz* kann als Reaktion auf *Meiden* oder Besänftigung seitens des Konkurrenten erfolgen (Tanner und Adler 2009). Als Resultat wird dessen Anwesenheit ignoriert und eine gegenseitige Beeinflussung der Bewegungen bleibt aus (Tanner und Adler 2009).

Zu einem weiteren Reaktionsbewegungsmuster kommt es, wenn Punktobjekte unterschiedlicher Art im Gegensatz zu den Verhaltensweisen in Konkurrenzsituationen versuchen, miteinander zu kooperieren. *Kooperation* wird beschrieben als Verhalten, welches zu einem Vorteil für ein anderes Individuum (Empfänger) führt, wobei dieser Mehrwert für dieses Individuum den Auslöser beziehungsweise den Grund für diese Verhaltensweise darstellt (West et al. 2007).

Eine weitere Form der territorialen Interaktion ist der *Parasitismus*, welcher sich laut Connor (1995) dadurch auszeichnet, dass ein Individuum von den körperlichen Investitionen eines zweiten Individuums profitiert.

Eine weitere Verhaltensweise, welche in der Tierwelt beobachtet werden kann, ist unter dem Namen *Kleptoparasitismus* bekannt, wobei Punktobjekte versuchen, bereits von anderen Individuen gesammelte Nahrungsmittel zu stehlen, um dadurch nicht selber auf Futtersuche gehen zu müssen (Broom und Ruxton 2003). Eine ähnliche Interaktionsform ist in der Videoüberwachung und Sicherheit als Ausrauben bekannt (Mahajan et al. 2004).

Diese aufgezählten Reaktionsbewegungsmuster sind in der Abbildung 2.2 thematisch sinnvoll gegliedert dargestellt und werden in den folgenden Abschnitten jeweils kurz diskutiert und beschrieben. Diese Erläuterungen sollen als Übersicht über die in der Literatur bekannten und beschriebenen Reaktionsbewegungsmuster dienen. Dabei werden diese Muster primär aus der biologischen Perspektive betrachtet. Wie bereits eingangs dieses Kapitels erwähnt, haben die Ausführungen in diesem Kapitel keinen Anspruch auf Vollständigkeit, sondern haben vielmehr die Absicht, in Form eines Überblicks, die wichtigsten Aspekte einer Auswahl von Reaktionsbewegungsmustern zu nennen. In Kapitel 4 werden drei dieser Reaktionsbewegungsmuster, nämlich *Verfolgen/Entfliehen*, *Konfrontation* sowie *Meiden*, wieder aufgegriffen und aus einer raumzeitlichen Perspektive diskutiert.

Um die in den nächsten Unterkapiteln folgenden Ausführungen zu den Reaktionsbewegungsmustern etwas zu strukturieren, wurde versucht, etwas Ordnung in die in der Literatur der Tierverhaltensforschung beschriebenen Formen von Reaktionsverhalten zu bringen. Dabei wurden einerseits die Art der beteiligten Punktobjekte und andererseits die Auslöser und Ursachen für das Zustandekommen der spezifischen Verhaltensabfolgen unter die Lupe genommen. Dadurch konnten grundsätzlich drei unterschiedliche Arten von Reaktionsbewegungsmustern ausfindig gemacht werden:

- Bewegungsmuster mit Tieren unterschiedlicher Art oder derselben Spezies, welche die Rolle von Räuber und Beute einnehmen. Auslöser sind dabei Interessenskonflikte zwischen den beteiligten Räufern und ihrer Beute. Während die Räuber versuchen ihre Beute so schnell wie möglich zu fangen, ist es das oberste Ziel der bedrohten Tiere, sich das Überleben zu sichern.
- Reaktionsbewegungen von Punktobjekten, sowohl derselben Spezies, als auch unterschiedlicher Arten, wobei die Konkurrenz um Raum und Ressourcen (z.B. Nahrung, Paarungspartner) die Ursache für die Entstehung der Bewegungsabfolge darstellt.
- Intraspezifische Verhaltensformen mit Punktobjekten derselben Spezies, welche aufgrund differenzierender Rollenzuteilungen (z.B. Männchen vs. Weibchen) entstehen.

Diese drei Gruppen von Reaktionsbewegungsmustern sind in der Abbildung 2.8 dargestellt. Alle aufgeführten Interaktionsformen werden in den nachfolgenden Abschnitten, wie in dieser Übersicht geordnet, der Reihe nach diskutiert.

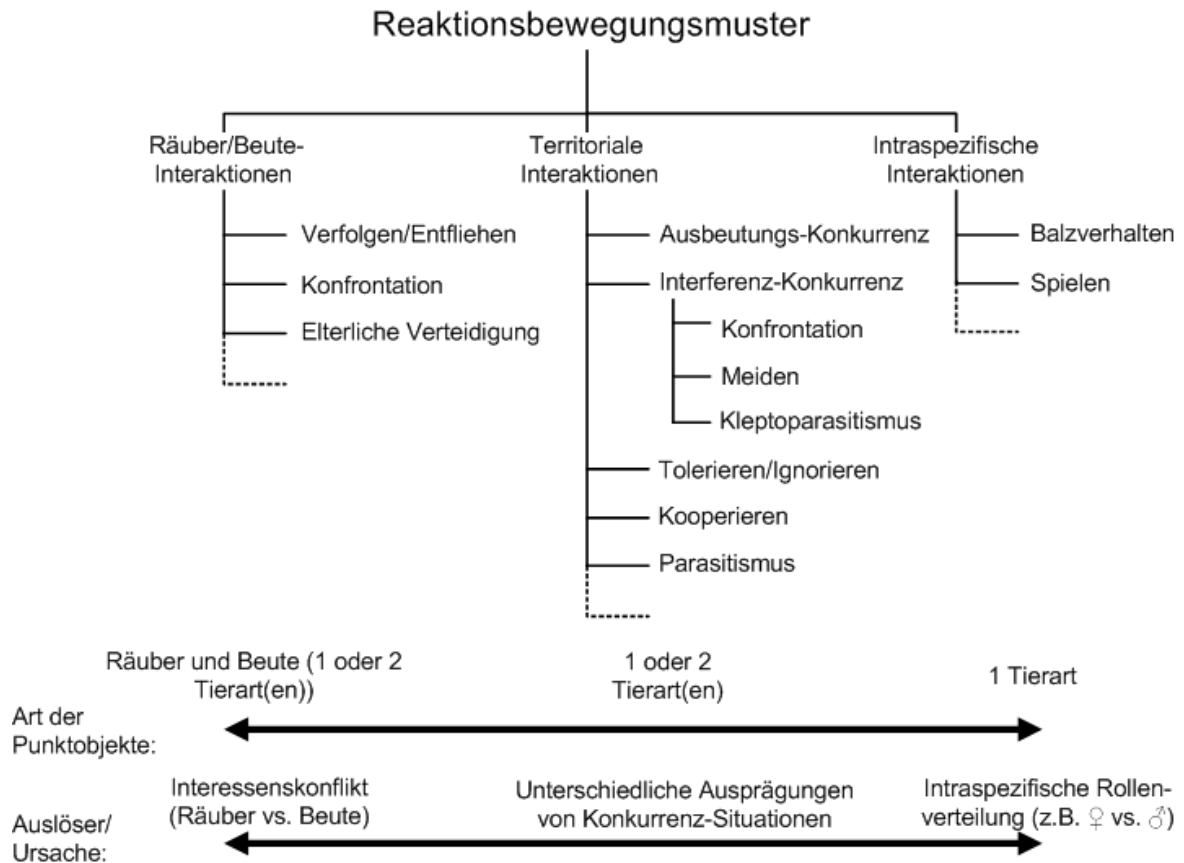


Abbildung 2.8: Übersicht Reaktionsbewegungsmuster

Einteilung der Reaktionsbewegungsmuster in die drei Klassen Räuber/Beute-Interaktionen, territoriale Interaktionen und intraspezifische Interaktionen.

2.4 Räuber/Beute-Interaktionen

In der Tierwelt können diverse Interaktionsformen ausfindig gemacht werden, in welchen eine bestimmte Anzahl von Tieren die Rolle von Räubern (z.B. Angreifer oder Verfolger) einnehmen und die restlichen Tiere als Beute (z.B. Verteidiger oder Fliehender) agieren. Solche Räuber/Beute-Interaktionen stehen in diesem Abschnitt im Fokus, bevor im nächsten territoriale Reaktionsbewegungsformen vorgestellt werden.

2.4.1 Verfolgen/Entfliehen

Ein Reaktionsbewegungsmuster, welches häufig in Räuber/Beute-Interaktionen anzutreffen ist, stellt die Bewegungsabfolge von *Verfolgen/Entfliehen* dar (Miller und Cliff 1994). Verfolgung bedeutet, dass sich ein Tier (z.B. ein Räuber) in Richtung eines Zielobjekts einer anderen Spezies (z.B. Beutetier) bewegt (Blythe et al. 1996). Das bedrohte Tier versucht als Reaktion auf den Angriff zu fliehen (Blythe et al. 1996). Bei der Betrachtung dieses Reaktionsbewegungsmusters gilt es zu berücksichtigen, dass Räuber erst ab einer gewissen Nähe zur Beute beginnen, die Verfolgung aufzunehmen (Furuichi 2002). Dasselbe gilt für Beutetiere, welche erst die Flucht ergreifen, wenn sich ein oder mehrere Verfolger in bedrohlicher Nähe befinden (Furuichi 2002). Dies lässt darauf schliessen, dass sowohl Verfolger als auch Fliehender eine gewisse Sichtlimitierung haben müssen

(Furuichi 2002). Diese Einschränkung des Sichtfeldes kann, je nach beteiligter Akteure und natürlichen Gegebenheiten, variieren (Furuichi 2002).

Während die Trajektorie eines Verfolgers eine mehr oder weniger gradlinige Bewegung hin zur beobachteten, erwarteten oder erinnerten Position des Zielobjekts widerspiegelt, sind die Bewegungsmuster von fliehenden Tieren komplexer (Miller und Cliff 1994). Verfolgte Tiere haben die Möglichkeit, auf zwei unterschiedliche Typen von Fluchtbewegungen zurückzugreifen (Furuichi 2002). Auf der einen Seite gradlinige, beziehungsweise bogenförmige Bewegungen, welche vor allem bei weit entfernten und langsamen Räubern erfolgsversprechend sind und auf der andern Seite Zickzack-Bewegungen für die Flucht vor schnellen und aus nächster Umgebung angreifenden Verfolgern (Furuichi 2002). Diese Aussagen werden unterstützt von Miller und Cliff (1994), welche das Fluchtverhalten von Tieren in drei nacheinander folgende Schritte unterteilen. Bei grosser Distanz zwischen Beute und Räuber ist die Fluchtbewegung in der Regel gradlinig (Miller und Cliff 1994). Mit zunehmendem Näherkommen des Räubers versucht das Fluchtobjekt zusehend den Verfolger durch unvorhersehbare Bewegungen (z.B. Zickzack-Bewegungen) zu irritieren (Miller und Cliff 1994). Falls es der Räuber trotz intensiven Fluchtversuchen der Beute schafft, das Zielobjekt einzuholen, folgen seitens der Beute die letzten Versuche, den Verfolger in die Irre zu führen und dadurch doch noch zu entkommen (Miller und Cliff 1994). Ein solches Mittel ist, sich tot zu stellen (Driver und Humphries 1988). Die beiden Taktiken des Zickzack-Gehens und des Todstellens sind Beispiele für adaptive Verhaltensweisen („protean behavior“) (Miller und Cliff 1994). Solche Verhaltensweisen können von Räubern nur schwer vorhergesagt werden und sind aus diesem Grund sehr effizient (Driver und Humphries 1988).

Für das Ende des Reaktionsbewegungsmusters zwischen Zielobjekt und Verfolger gibt es zwei mögliche Ausgänge (Furuichi 2002). Die erste Möglichkeit besteht darin, dass das Fluchtobjekt vom Verfolger eingeholt wird (Furuichi 2002). Dieser Ausgang des Verhaltensmusters kommt zustande, wenn die Distanz zwischen Verfolger und Fluchtobjekt eine gewisse Minimaldistanz unterschreitet (Furuichi 2002). Als zweite Möglichkeit des Ausgangs dieses Verhaltensmusters kommt die erfolgreiche Flucht des fliehenden Tieres vor dem Verfolger in Frage (Furuichi 2002). Dies ist der Fall, wenn der Verfolger die Jagd nach seiner Beute aufgibt, beispielsweise aufgrund ausgehender Energiereserven (Furuichi 2002).

2.4.2 Angreifen/Verteidigen in Konfrontationen

Neben dem Fliehen vor einem Angriff hat das bedrohte Tier auch die Möglichkeit, seinem Widersacher durch *kämpferisches Verteidigen* des eigenen Lebens zu begegnen (Eilam 2005). Diese Verteidigungsstrategie kommt vor allem in Situationen zum Einsatz, in welchen das angegriffene Tier keine Chance mehr hat, durch Fliehen zu entkommen oder sich durch Totstellen zu retten (Eilam 2005). Dabei steuert das bedrohte Tier auf seinen Widersacher zu und versucht dadurch, diesen doch noch vom Angriff abzuhalten und sich so das Überleben zu sichern (Eilam 2005). Kommt es zu einem Kampf zwischen Räuber und Beute, dann ist dieser geprägt durch schnelles Abwechseln von *Angriff*

und Verteidigung, beziehungsweise Verfolgung und Entfliehen durch die beteiligten Tiere (Blythe et al. 1996). Dabei zeichnen sich die Bewegungen der Tiere durch hohe Geschwindigkeiten, enge Wendungen und schnelle Drehungen aus (Blythe et al. 1996). Zudem kommt es aufgrund der geringen Nähe der beteiligten Tiere zu häufigem *Aufeinadertreffen*, was sich in schneidenden Trajektorien widerspiegelt (Blythe et al. 1996). Das Reaktionsbewegungsmuster von *Angriff und Verteidigung in Kampfsituationen* beinhaltet aus diesem Grund diverse generische Muster wie *Konvergenz*, *Divergenz*, *Zusammentreffen*, *Aufbrechen* oder *Anführen* (Dodge et al. 2008). Siehe dazu die simulierten Trajektorien eines Kampfes in der Abbildung 2.9. Für das Ende einer *Konfrontation* zwischen einem Räuber und seiner Beute kommen

bekanntlich zwei mögliche Ausgänge in Frage, wobei entweder der Räuber als Sieger hervorgeht und seine Beute töten kann oder das Beutetier gewinnt und als Folge entkommen und überleben kann.

Neben *Kämpfen* zwischen Räubern und ihrer Beute gibt es auch noch andere Auslöser für Kampfsituationen. Dabei können laut Moorcroft (2008) auch Konkurrenzsituationen um Lebensräume, Ressourcen oder Weibchen zu kämpferischem Verhalten zwischen Tieren führen. Diese Form der Konkurrenz wird im nächsten Unterkapitel 2.5 bei der Vorstellung der territorialen Interaktionsformen beschrieben.

2.4.3 Elterliche Verteidigung

Im Gegensatz zu dem im letzten Abschnitt beschriebenen Reaktionsbewegungsmuster von *Angriff und Verteidigung* sind bei der *elterlichen Verteidigung* neben den angreifenden und verteidigenden Tieren zusätzlich noch Jungtiere involviert. Dabei gibt es zwei unterschiedliche Ausprägungen des Bewegungsmusters der *elterlichen Verteidigung*. Auf der einen Seite ist der Schutz des Nachwuchses vor Räuberangriffen eine wichtige Form der *elterlichen Verteidigung* (Andersson et al. 1980). Dadurch kann die Überlebenschance der Jungen erhöht werden (Andersson et al. 1980). Neben der *elterlichen Verteidigung* im Falle einer Attacke von Räubern ist es je nach Grössenverhältnis zwischen Beute und Räuber auch möglich, dass es zu *Konterangriffen* seitens der bedrohten Tiere kommt (Magalhães et al. 2005). Daraus resultiert eine Abfolge von *Räuberangriff*, *Konterattacke* der Beute und *elterlichem Verteidigungsverhalten* durch die Räuber zum Schutz ihres eigenen Nachwuchses (Magalhães et al. 2005).

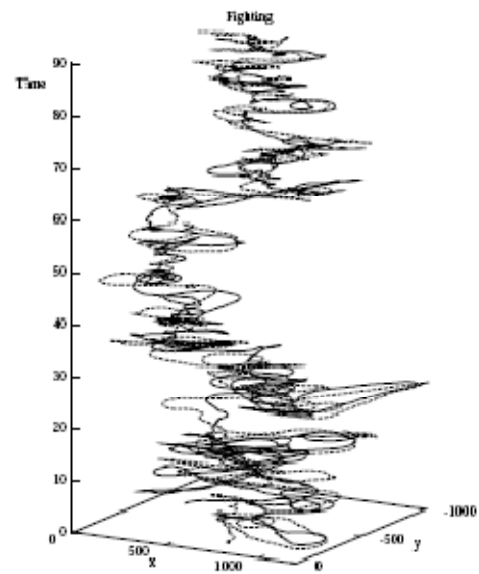


Abbildung 2.9: Trajektorien eines Kampfes (Blythe et al. 1996)

Trajektorien einer simulierten Situation mit kämpferischem Verhalten.

2.5 Territoriale Interaktion

Neben den in Kapitel 2.4 beschriebenen Verhaltensformen zwischen Räuber und Beute gibt es auch Bewegungsmuster, welche als Folge von territorialer Interaktion zwischen zwei oder mehreren Tieren entstehen. Solche Bewegungsmuster werden jeweils ausgelöst durch das Eindringen von Tieren in das Territorium von anderen Tieren. Unter Territorium versteht man dabei die Fläche, in der ein Tier normalerweise lebt, wobei Bewegungen wie Migration, Auswanderung oder andere lange Exkursionen nicht berücksichtigt werden (Moorcroft 2008).

In diesen Lebensräumen kann es aufgrund von limitierten Ressourcen (z.B. Nahrung oder Paarungspartner) zu Konkurrenzsituationen zwischen Tieren kommen (Moorcroft 2008). Konkurrenz kann dabei sowohl zwischen Individuen verschiedener Arten (interspezifische Konkurrenz), als auch zwischen Tieren derselben Art (intraspezifische Konkurrenz) stattfinden (Freudig und Sauermost 2004). Weiter können zwei unterschiedliche Formen von Konkurrenz, nämlich ausbeuterische („exploitative“) und behindernde („interference“) Konkurrenz, unterschieden werden (Park 1954). Je nachdem, ob sich die konkurrierenden Tiere begegnen oder nicht, spricht man von Interferenz-Konkurrenz (Konkurrenz durch Störung) oder von Ausbeutungs-Konkurrenz (exploitation competition) (Freudig und Sauermost 2004).

2.5.1 Ausbeutungs-Konkurrenz

Im Falle der Ausbeutung reagieren die Tiere auf die Verfügbarkeit einer Ressource, die durch die Aktivitäten von Konkurrenten gesenkt wird (Freudig und Sauermost 2004). Dabei ist jedes Individuum abhängig davon, welche Ressourcenmengen nach der Ausbeutung durch andere Individuen noch verfügbar sind (Freudig und Sauermost 2004). Die gemeinsame Nutzung derselben Ressourcen führt also zu einer indirekten negativen Interaktion (Case und Gilpin 1974). Ausbeutungs-Konkurrenz kann reduziert werden, wenn sich die beteiligten Individuen auf leicht unterschiedliche Ressourcen spezialisieren, auf verschiedene Habitate ausweichen oder zeitlich getrennt auf Nahrungssuche gehen (McGinley und Codella 2008). Im Falle der Spezialisierung auf verschiedene Ressourcen spricht man von *Meiden* von Ausbeutungs-Konkurrenzsituationen (Case und Gilpin 1974). Dadurch wird eine räumliche Koexistenz (sympatrisches Vorkommen) von Populationen verschiedener Arten ermöglicht, da die Arten Ressourcen aufteilen und dadurch unterschiedliche ökologische Nischen besetzen (Freudig und Sauermost 2004). Im Gegensatz dazu ist die räumliche und/oder zeitliche Separation der Nahrungsmittelaufnahme der Arten nicht alleine auf Basis der Ausbeutungs-Konkurrenz zu erklären, weshalb deren Entstehung nicht nur als Folge des *Meidens* von Konkurrenz um Ressourcen interpretiert werden darf (Case und Gilpin 1974). Für eine bessere Erklärung der Beweggründe der räumlichen und zeitlichen Separation muss die Interferenz-Konkurrenz beigezogen werden (Case und Gilpin 1974). Interferenz-Konkurrenz führt zu einem *Meiden* von direkten Interaktionen und als Folge davon zur Nutzung derselben Ressource an verschiedenen Orten und/oder zu verschiedenen Zeiten (Case und Gilpin 1974). Im Falle von einer Überlappung in der Ressourcennutzung kommt es in der Regel zu gegenseitiger interspezifischer

Interferenz, was zu einer räumlichen Separation innerhalb eines Habitats und somit zur Herausbildung von interspezifischer Territorialität führen kann (Case und Gilpin 1974). Auf die Interferenz-Konkurrenz wird im nachfolgenden Abschnitt noch näher eingegangen.

2.5.2 Interferenz-Konkurrenz

Im Gegensatz zu Verhaltensweisen, welche als Folge von Ausbeutungs-Konkurrenzsituationen entstehen, zeichnen sich Bewegungsmuster in Situationen mit Interferenz-Konkurrenz dadurch aus, dass die beteiligten Tiere direkt auf gegenseitige Anwesenheit reagieren, wobei zum Beispiel direkte Konkurrenz um Territorien oder Paarungspartner stattfindet (Freudig und Sauermost 2004). Dabei versuchen Tiere durch physische oder chemische Mechanismen sowie spezielle Verhaltensweisen anderen Tieren Schaden zuzufügen (Stewart et al. 2002), wobei die gemeinsam genutzten Ressourcen nicht unbedingt limitiert sein müssen (Odden et al. 2010). Solche Mechanismen werden beispielsweise eingesetzt bei direkten Attacken auf andere Tiere (Stewart et al. 2002). Zu solchen *aggressiven Verhaltensweisen* kommt es beispielsweise in Konkurrenzsituationen von zwei Räuberarten um dieselbe Beute („intraguild Predation“), welche wie der *Kleptoparasitismus* und das *Meiden* von Interaktionspartnern ebenfalls in Situationen mit Interferenz-Konkurrenz auftaucht (Creel et al. 2001). Diese Bewegungsmuster werden in der Folge kurz diskutiert.

2.5.3 Konfrontation

Im Falle von Konkurrenz um Lebensräume oder knappe Ressourcen kann es dazu kommen, dass Tiere in neue Territorien eindringen, während dort ansässige Tiere versuchen, durch *kämpferisches Verhalten* ihren Lebensraum zu verteidigen (Barrett et al. 2005). Eine solche Situation ist in der Abbildung 2.10 dargestellt. Zu den limitierten und aus diesem Grund umkämpften Ressourcen zählen zum Beispiel Nahrungsmittel oder Paarungspartner (Moorcroft 2008). *Kämpferisches Verteidigen* des eigenen Territoriums tritt vor allem dann auf, wenn die Verteidigung von relevanten und vor allem limitierten Ressourcen in diesem Gebiet aus Sicht des Individuums ökonomisch ist und daher ohne allzu grossen Krafteinsatz erfolgreich zu bewerkstelligen ist (Moorcroft 2008). Dies bedeutet, dass die Verteidigung zu einer verbesserten



Abbildung 2.10: Konfrontation zwischen Rothirschen (Creel et al. 2005)

Kämpferische Konfrontation zwischen zwei männlichen Rothirschen.

Fitness führen sollte im Vergleich zu alternativen Verhaltensweisen wie beispielsweise das *Meiden* oder *Ignorieren* von Konkurrenten (Moorcroft 2008). Schlüsselfaktoren für diesen Entscheid sind dabei die Verteilung der Ressourcen im Raum, deren Vorhersehbarkeit und die Anzahl der

Mitkonkurrenten (Moorcroft 2008). Entschliesst sich ein Tier für eine *kämpferische Verteidigung*, so resultiert in der Regel eine Verdrängung der unterlegenen Tiere (Moorcroft 2008). Wie im Falle von Kampfsituationen in Räuber/Beute-Interaktionen sind Kämpfe um Lebensräume und Ressourcen nach dem Eindringen der herausfordernden Tiere geprägt durch schnelles Abwechseln von *Angriff* und *Verteidigung*, beziehungsweise *Verfolgung* und *Entfliehen* der beteiligten Tiere (Blythe et al. 1996). Aus raum-zeitlicher Perspektive sind die Verhaltensweisen während einer Kampfsituation in territorialen Interaktionen also identisch mit denjenigen mit Beteiligung von Räufern und ihrer Beute und werden aus diesem Grund im nächsten Kapitel gemeinsam behandelt und zwar beschrieben unter dem Namen *Konfrontation*. An dieser Stelle gilt es noch anzumerken, dass es neben den bereits genannten aggressiv geführten *Zusammentreffen* zwischen Tieren auch sogenannte ritualisierte *Konfrontationen* gibt, bei welchen sich die Beteiligten nicht gegenseitig attackieren (Smith und Price 1973). Um ernsthafte Verletzungen zu verhindern, wird auf einen Kampf verzichtet (Smith und Price 1973). Der Sieger solcher *Konfrontationen* wird mit Hilfe anderer Kriterien bestimmt, zum Beispiel anhand eines Grössenvergleichs zwischen den beteiligten Tieren oder mit Hilfe der Zeitdauer, während derer die Tiere in Anwesenheit des anderen ausharren können ohne zu fliehen (Smith und Price 1973). Diese Ausführungen zeigen, dass *Konfrontationen* je nach Situation und Art der beteiligten Tiere stark unterschiedlich sein können.

2.5.4 Meiden

Neben *kämpferischem Verteidigungsverhalten* kann es auch aufgrund von *Meiden* allzu grosser Nähe zu Mitkonkurrenten zu einer Verdrängung kommen (Moorcroft 2008). Dabei gilt es zu beachten, dass *Meiden* wie aggressives Verhalten ebenfalls aus Interferenz-Konkurrenzsituationen heraus entstehen können (Creel et al. 2001). Dieses Phänomen konnte anhand der in grossen Teilen Asiens überlappend lebenden Tigern und Leoparden untersucht werden, wobei Odden et al. (2010) zeigen konnten, dass Leoparden das Zusammentreffen mit Tigern meiden. Als Resultat verschoben die Leoparden ihre Lebensräume an die Ränder der Territorien der Tiger (Odden et al. 2010). Dies bestätigten die Aussagen von Woodroffe und Ginsberg (2005), dass unterlegene Arten Interferenz-Konkurrenzsituationen meiden und sich als Folge davon aus den ursprünglichen Lebensräumen verdrängen lassen, um sich in Gebieten mit geringerem Interaktionspotential mit Widersachern anzusiedeln.

Ähnliche Beobachtungen konnten Cooper et al. (2008) bei der Untersuchung der Verbreitungsgebiete und Interaktionsformen von Vieh und Reh machen. Während Vieh-Tiere durch die Anwesenheit von Rehen nicht beeinflusst wurden, konnten im Verhalten der Rehe Ansätze von *Meiden* ihrer Interaktionspartner erkannt werden (Cooper et al. 2008). Rehe beschränkten die räumliche Nähe zu Vieh-Tieren auf eine Minimaldistanz von ungefähr 50 m und tendierten dazu, bei Annäherungsversuchen durch das Vieh, sich wieder zu entfernen (Cooper et al. 2008). Aufgrund dieses *Meidens* naher Kontakte zu Vieh-Tieren resultierte eine Separation der beiden Arten im Untersuchungsgebiet (Cooper et al. 2008).

Das Phänomen des *Meidens* konnte auch in einer Studie der Interaktion zwischen Räubern und Beute durch Laporte et al. (2010) beobachtet werden. Laporte et al. (2010) untersuchte dabei die Effekte von Wölfen auf Elche und Vieh. Die Reaktion von Vieh-Tieren auf die Anwesenheit von Wölfen war sehr variabel, was unter anderem auf fehlende Antiräuber-Verhaltensweisen aufgrund mangelnder Erfahrung im Umgang mit Räubern zurückzuführen war (Laporte et al. 2010). Im Gegensatz dazu stehen in der Wildnis lebende Elche, welche auf die Präsenz von Wölfen mit einer Verschiebung in Gebiete mit unebenem Terrain mit rauerer Oberflächen und grösserer Hangneigung reagierten (Laporte et al. 2010). Sowohl Vieh, als auch Elche erhöhten zudem die Anzahl der Bewegungsrichtungswechsel, was sich in ihren Bewegungspfaden widerspiegelte (Laporte et al. 2010).

Zusammenfassend kann ausgesagt werden, dass es je nach Perspektive, Betrachtungsmaßstab und beteiligter Akteure zu unterschiedlichen Ausprägungen von *Meiden* kommen kann. Dabei ist sowohl das *Meiden* von gemeinsamen Lebensräumen, wie im Falle von Leoparden und Tigern in Asien, als auch *Meiden* räumlicher Nähe (z.B. zwischen Elchen und Vieh) möglich. Im Falle einer Beteiligung von Räubern- und Beutetieren kann es zudem zu Lebensraumverlagerungen als Antiräuber-Reaktion kommen.

2.5.5 Kleptoparasitismus

Wie bereits angesprochen zählt neben der *aggressiven Verhaltensweise* und dem *Meiden* auch der *Kleptoparasitismus* zur Form der Interferenz-Konkurrenz (Creel et al. 2001). Dabei versuchen Tiere bereits von anderen Individuen gesammelte Nahrungsmittel zu stehlen, um dadurch nicht selber auf Futtersuche gehen zu müssen (Broom und Ruxton 2003). Das Erbeuten von Nahrung durch aggressive Verhaltensweisen ist aus Sicht der Räuber vor allem dann lohnenswert, wenn Nahrungsmittel versteckt oder rar sind und dadurch nur unter grossen Zeitinvestitionen ausfindig gemacht werden können (Sirot 2000). *Kleptoparasitismus* ist laut Broom und Ruxton (2003) bekannt bei Vögeln (z.B. Brockmann und Barnard 1979), Säugetieren (z.B. Carbone et al. 1997) und wirbellosen Tieren (Tso und Severinghaus 1998).

In vielen Fällen kommt es seitens der bestohlenen Individuen (Host) als Reaktion auf kleptoparasitische Verhaltensweisen zur Entwicklung von Verteidigungsstrategien, um deren negative Auswirkungen zu minimieren (Ridley und Raihani 2007). Solche defensive Verhaltensweisen konnten im Falle der Interaktion zwischen Elsterdrossling (*turdoides bicolor*) und ihrem potentiellen Kleptoparasiten Trauerdrongo (*dicurus adsimilis*) durch Ridley und Raihani (2007) beobachtet werden. Der Trauerdrongo folgt Gruppen von Elsterdrosslingen und agiert dabei als Wächter vor Räuberangriffen, wobei er Alarm gibt, um die Gruppe vor herannahenden Bedrohungen zu warnen. Der Trauerdrongo nutzt seine Rolle von Zeit zu Zeit aus und gibt falschen Alarm, um gesammelte Nahrungsmittel der Elsterdrosslingen zu stehlen. Um sich vor solchem *Kleptoparasitismus* zu schützen, haben vor allem grössere Gruppierungen der Drosslinge ein aggressives Verteidigungsverhalten gegenüber dem Drongo entwickelt, um diesen von der Gruppe fernzuhalten.

Dabei nehmen sie in Kauf, dass sie ohne ihr zusätzliches Warnsystem vor Räubern auskommen müssen, können sich dafür aber vor kleptoparasitischen Verhaltensweisen schützen und dadurch ihre Nahrung bewachen (Ridley und Raihani 2007).

Eine ähnliche Verhaltensweise, wie der *Kleptoparasitismus* bei Tieren, ist in den Bereichen Überwachung und Sicherheit unter dem Namen *Ausrauben* bekannt. Als Beispiel kann hier das Stehlen eines Rucksacks genannt werden, welches Mahajan et al. (2004) versuchten, mit Hilfe von Algorithmen in Bildern von Überwachungskameras zu detektieren. Dabei wurde das Stehlen als Bewegungsabfolge formalisiert mit den Schritten: *Ablegen* des Rucksacks, *Stehlen* dessen durch eine zweite Person und *Davonrennen* des Diebes (Mahajan et al. (2004).

2.5.6 Tolerieren/Ignorieren

Eine alternative Strategie in Konkurrenzsituationen ist neben der *kämpferischen Verhaltensweise* und dem *Meiden* das *Ignorieren* der Anwesenheit eines Mitstreiters (Moorcroft 2008). Dabei begegnen Tiere ihren Konkurrenten mit *Toleranz* (Schoener 1983). Eine tolerante Verhaltensweise kann als Reaktion auf *Meiden* oder Besänftigung seitens des Konkurrenten erfolgen (Tanner und Adler 2009). Als Resultat wird die Anwesenheit des Konkurrenten ignoriert und Reaktionen auf dessen Aktionen bleiben aus. Aus diesem Grund kann ausgesagt werden, dass eine gegenseitige Beeinflussung der Bewegungen fehlt (Tanner und Adler 2009).

Eine solche indifferente Verhaltensweise konnte im Falle von Vieh-Tieren gegenüber Rehen beobachtet werden (Loft et al. 1993). Mit Hilfe der Analyse der Bewegungen und Raumnutzungen von mit einem Radiosender ausgerüsteten Reh- und Viehtieren in der Sierra Nevada in Kalifornien konnte aufgezeigt werden, dass Viehtiere die Anwesenheit/Abwesenheit von Rehen ignorieren (Loft et al. 1993).

Tolerante Verhaltensweisen konnten auch zwischen zwei unterschiedliche Arten von Ameisen, nämlich *formica xerophila* und *formica integroides*, durch Tanner und Adler (2009) beobachtet werden.

2.5.7 Kooperieren

Neben Konkurrenzsituationen kann es in territorialen Interaktionen auch zu einer *Kooperation* zwischen den beteiligten Tieren kommen. Speziell interessant für diese Arbeit ist die *Kooperation* zwischen Individuen von zwei verschiedenen Arten, welche laut West et al. (2007) als *Mutualismus* bezeichnet wird. *Mutualismus* ist die Bezeichnung für eine Form der Wechselbeziehung zwischen artverschiedenen Organismen, bei der (im Gegensatz zur Konkurrenz, zum Räuber-Beute-Verhältnis oder zum *Parasitismus*) beide Partner ihren Nutzen ziehen können (Freudig und Sauermost 2004). Dieser Terminus wird häufig synonym zu Symbiose i.e.S. verwendet (Freudig und Sauermost 2004). Unter Symbiose i.w.S. wird jegliches Zusammenleben von artverschiedenen Organismen, einschliesslich des *Parasitismus*, verstanden (Freudig und Sauermost 2004). Im Gegensatz dazu

versteht man unter Symbiose i.e.S. eine gesetzmässige Vergesellschaftung artverschiedener Organismen, die für beide Symbiosepartner von Vorteil ist und grenzt somit die Symbiose gegen den *Parasitismus* ab (Freudig und Sauermost 2004).

2.5.8 Parasitismus

Eine weitere Form der territorialen Interaktion ist der *Parasitismus*, welcher sich laut Connor (1995) dadurch auszeichnet, dass ein Individuum von den körperlichen Investitionen eines zweiten Individuums profitiert. Beim *Parasitismus* handelt es sich eine Wechselbeziehung zweier Organismenarten in einem Parasit-Wirt System (Freudig und Sauermost 2004). Die Bildung des Parasit-Wirt-Systems vollzieht sich oft in vielen aufeinanderfolgenden, sich bedingenden Schritten, wobei die Partner im Allgemeinen räumlich und zeitlich koinzidieren müssen (Freudig und Sauermost 2004). Lokal ist oft das Auffinden des Wirtshabitats nötig (Freudig und Sauermost 2004). In ihm muss das Wirtsindividuum zufällig oder gezielt geortet werden (Wirtsfindung) (Freudig und Sauermost 2004).

2.6 Intraspezifische soziale Interaktionen

Neben Verhaltensformen zwischen Räubern und ihrer Beute oder territorialen Interaktionen können auch soziale Wechselbeziehungen zwischen Tieren derselben Art (intraspezifisch) zu Reaktionsbewegungsmustern führen. Dabei gilt es zu beachten, dass die intraspezifische Rollenverteilung (z.B. Männchen vs. Weibchen) zu einer Abfolge von Aktion und Reaktion führt. Zu intraspezifischen Interaktionsformen gehören beispielsweise das *Balzverhalten* zwischen Männchen und Weibchen derselben Art und das *Spielen* zwischen Jungtieren.

2.6.1 Balzverhalten

Aus dem Bereich der Fortpflanzung gilt das Reaktionsverhalten von *Werben* (*courting*) und *Umworben-Werden* (*being courted*) als wichtige vielfältige, intraspezifische Interaktionsform auf dem Weg hin zur Paarung (Barrett et al. 2005). Dabei interagieren Männchen und Weibchen in den Rollen des Werbers und der Umworbenen, was zu einer ähnlichen Abfolge von *Verfolgung* und *Entfliehen* wie im Falle von Räuber/Beute-Interaktionen führt, welche in Blythe et al. (1996) wie folgt beschrieben ist: Beim Werben bewegen sich Tiere (in der Regel männlich) auf Tiere des anderen Geschlechts zu (meist weiblich), mit welchen sie sich paaren möchten. Dabei gilt es sich von der besten Seite zu präsentieren, denn das umworbene Tier reagiert auf diesen Vorstoss durch sorgfältiges Prüfen der Bewerber, da die genetische Qualität des Paarungspartners einen grossen Einfluss auf die Fitness der Nachkommen hat. Die Präsentation der eigenen Stärke durch die Bewerber hat in genügender Nähe zu den umworbenen Tieren stattzufinden, ohne jedoch diesem dabei zu nahe zu treten und es dadurch zu verscheuchen. Die Umworbenen müssen während dessen in ihrem Verhalten versuchen, ein gutes Gleichgewicht zwischen Zeigen des Interesses und Zurückhaltung zu schaffen. Dadurch soll sichergestellt werden, dass den Bewerbern weder die Motivation geraubt wird noch dass

sich diese zu stark angezogen fühlen und es direkt zur Paarung kommt. Dadurch kommt es zu einer Wechselbeziehung von *Annähern* und *Distanzieren*, bis der finale Vorstoss durch den Bewerber erfolgt und es zur Begattung kommt.

2.6.2 Spielen

Viele der bis anhin genannten Reaktionsbewegungsmustern, wie beispielsweise *Verfolgen/Entfliehen*, *Kämpfen* oder das *Balzverhalten* werden in jugendlichem Alter durch interaktives *Spielen* geübt (Blythe et al. 1996). Dabei können Tiere die genannten Verhaltensweisen durch wiederkehrendes Abtauschen der Rollen (z.B. Wechsel zwischen Verfolger und Fliehender) üben (Blythe et al. 1996). Diese Erkenntnisse stammen von Fagen (1981), welcher das *Spielen* von Tieren in seinem Buch mit dem Titel „Animal Play Behavior“ ausführlich diskutiert.

2.7 Forschungslücken

In diesem Abschnitt sollen die in diesem Kapitel gemachten Ausführungen zum wissenschaftlichen Hintergrund dieser Masterarbeit diskutiert werden und daraus die in der Einleitung aufgelisteten Forschungsfragen abgeleitet werden.

Anhand der Betrachtung des wissenschaftlichen Backgrounds dieser Masterarbeit wird klar, dass bis anhin in der GIScience vorwiegend Methoden entwickelt worden sind für die Detektion von Mustern mit Beteiligung von einer Art von sich bewegenden Punktobjekten. Dazu gehören Algorithmen zur Entdeckung von Bewegungsmustern wie *Herdenbildung (flock)*, *Führungsverhalten (leadership)*, *Konvergenz (convergence)* und *Zusammentreffen (encounter)* (Laube et al. 2004), automatisierte Methoden für die Muster *Führungsverhalten* und *Folgen* (Andersson et al. 2008) oder aber ein Ansatz zur Detektion von sich schön in einer Reihe bewegenden Punktobjekten (*Single-File*) (Buchin et al. 2008). Dodge et al. (2008) entwickelte eine Taxonomie zur Klassifikation von Bewegungsmustern, wobei auch Muster mit Beteiligung von interagierenden Punktobjekten unterschiedlichen Typs genannt werden. Solche Interaktionen sind wichtige Bausteine der Ontologie von Orellana und Renso (2010). Diese Ontologie ist angewiesen auf Methoden zur automatisierten Detektion von Interaktionen (Orellana und Renso 2010). Für einige der bekannten Interaktionen fehlen bis anhin aber noch solche Methoden, weshalb in Orellana et al. (2009) Ansätze zur Entdeckung von *Annäherung*, *Anhalten* sowie *Anziehung* vorgestellt werden. Zu den Bewegungsmustern, für welche noch wenig automatisierten Verfahren bekannt sind, gehören auch die Reaktionsbewegungsmuster. Dies lässt den Schluss zu, dass bis anhin in der GIScience kaum Forschungsanstrengungen zur Entwicklung von automatisierten Verfahren von Reaktionsbewegungsmustern unternommen wurden.

Anhand der gemachten Ausführungen zum Forschungsbereich der Tierverhaltensforschung konnte zudem aufgezeigt werden, dass in der Biologie eine grosse Menge an Beschreibungen und Diskussionen von Reaktionsbewegungsmustern vorliegt, allerdings auch in diesem Forschungsbereich bis zum heutigen Zeitpunkt noch beinahe keine automatisierten Methoden entwickelt wurden. Durch die aufgezeigten möglichen Anwendungsbereiche wird zudem klar, dass solche Methoden sehr

hilfreich sein und grosse Verbreitung finden könnten. Um einen Beitrag zu leisten zur Schliessung dieser Forschungslücke, wird in dieser Arbeit die Forschungsfrage 1 diskutiert.

FF1: Wie können, basierend auf den in der Literatur vorhandenen Definitionen, die Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* sowie *Meiden* formalisiert werden?

In Kapitel 4 werden exemplarisch die drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* formalisiert. Dabei wird auf die vielen in der Literatur der Verhaltensbiologie gemachten Erläuterungen der Bewegungsmuster zurückgegriffen und versucht die wichtigsten raum-zeitlichen Merkmale abzuleiten und diese in einer formale Beschreibung zusammenzufassen.

Mit Hilfe der angesprochenen Formalisierung ist allerdings lediglich der erste Schritt in Richtung eines automatisierten Verfahrens getan. In einem nächsten Schritt müssen basierend auf den formalen Beschreibungen der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* Algorithmen zu deren Detektion in Bewegungsdatensätzen entwickelt werden. Dabei sollen für jedes der drei Reaktionsbewegungsmuster sowohl die individuelle als auch die räumliche Perspektive verwendet werden. *Konfrontation* wurde gewählt aufgrund des grossen Interesses von verschiedenen Forschungsrichtungen (z.B. Verhaltensbiologie, Überwachung und Sicherheit) an diesem Bewegungsmuster. Bei der Betrachtung der in diesem Kapitel beschriebenen Reaktionsbewegungsmuster fällt zudem auf, dass das Verhalten Kämpfen in *Konfrontationen* sowohl zwischen Räubern und deren Beute sowie zwischen Punktobjekten in Konkurrenzsituationen um Ressourcen (z.B. Nahrung oder Paarungspartner) auftreten kann. Dies bedeutet, dass *Konfrontationen* in der Realität sehr oft vorkommen. Aus Sicht der beteiligten Punktobjekte und in Anbetracht der auslösenden Umstände der Kampfsituationen haben diese beiden Fälle doch markant unterschiedliche Eigenschaften. Aus raum-zeitlicher Sicht sind diese Verhaltensweisen jedoch identisch. *Verfolgen/Entfliehen* wurde ausgewählt, da es sich dabei um den Klassiker schlechthin unter den Reaktionsbewegungsmustern handelt, welchem in der Literatur der Tierverhaltensforschung sehr grosse Aufmerksamkeit gewidmet wurde. Als Folge davon gibt es Unmengen an Literatur über dieses Bewegungsmuster, was eine sehr vielversprechende Grundlage darstellt für eine gute Formalisierung und einen gut funktionierenden Algorithmus. Um neben diesen beiden Räuber/Beute-Interaktionen auch noch eine territoriale Interaktionsform im Detail zu betrachten, wurde zudem das Reaktionsbewegungsmuster *Meiden* ausgewählt. Dieses Muster wurde bis zum heutigen Zeitpunkt in sehr vielen Forschungsanstrengungen der Tierverhaltensforschung untersucht, was darauf hindeutet, dass auch das Interesse an einem Algorithmus zur automatisierten Detektion gross ausfallen wird. Diese Ausführungen zeigen, dass in dieser Arbeit die Forschungsfrage 2 aufgegriffen und diskutiert wird.

FF2: Wie lassen sich diese Formalisierungen in Algorithmen umsetzen, um die definierten Reaktionsbewegungsmuster in raum-zeitlichen Datenreihen zu detektieren?

Die Ausführungen zur Taxonomie von Dodge et al. (2008) haben gezeigt, dass es sehr viele verschiedene Bewegungsmuster gibt, welche sich zum Teil entweder aus biologischer Sicht sehr ähnlich sind oder anhand raum-zeitlicher Kriterien grosse Ähnlichkeit aufweisen. Eine grosse Herausforderung der Bewegungsmusteranalyse ist demzufolge die Entwicklung von Algorithmen, welche einerseits die Detektion des formalisierten Musters zulassen und andererseits auch eine erfolgreiche Abgrenzung gegenüber ähnlichen Mustern ermöglichen. Für den Fall des Reaktionsbewegungsmusters *Verfolgen/Entfliehen* sind aus raum-zeitlicher Sicht beispielsweise die in Andersson et al. (2008) vorgestellten Muster *Führungsverhalten* und *Folgen* oder auch die Fortbewegung in einer Reihe (*Single-File*) (Buchin et al. 2008) durchaus vergleichbar. Ziel ist es also, mit Hilfe einer guten Formalisierung und den Parametern der Algorithmen eine erfolgreiche Abgrenzung gegenüber andern Interaktionsformen zu erreichen. Diesen Erläuterungen zu folge drängt sich in dieser Arbeit die Bearbeitung der Forschungsfrage 3 auf.

FF3: Welches sind die entscheidenden Parameter für eine erfolgreiche Detektion der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* und wie gross ist deren Einfluss für die Abgrenzung zu verwandten Bewegungsmustern?

Anhand diverser Experimente mit den entwickelten Algorithmen wird versucht die Forschungsfrage 3 zu beantworten. Dabei kann anhand der Berechnung von verschiedenen Gütekriterien der Erfolg der Algorithmen bei diesen Experimenten beziffert werden und Aussagen über die Stärken und Schwächen gemacht werden. Dadurch können die Grenzen der vorgeschlagenen Methoden ausgearbeitet und mögliche Verbesserungspotentiale aufgedeckt werden. Durch die Resultate der Experimente lassen sich wichtige Erkenntnisse gewinnen über die Einflüsse der einzelnen Parameter der Algorithmen und zudem können dadurch diejenigen Parameter bestimmt werden mit Hilfe deren eine gute Abgrenzung gegenüber anderen Bewegungsmuster gelingt. Dies ist von grosser Bedeutung, damit nicht Situationen fälschlicherweise als *Verfolgen/Entfliehen*, *Konfrontation* oder *Meiden* klassiert werden.

Mit der individuellen und der räumlichen Perspektive stehen in dieser Arbeit zwei unterschiedliche Betrachtungsweisen zu Verfügung, welche es zu vergleichen gilt. Mit Hilfe der Experimente mit den Algorithmen wird versucht für alle drei Reaktionsbewegungsmuster die wichtigsten Unterschiede auszuarbeiten und dadurch die Forschungsfrage 4 zu diskutieren.

FF4: Inwiefern unterscheiden sich die Resultate der Algorithmen der individuellen Perspektive von denjenigen der räumlichen Perspektive?

Als erstes folgen nun im nächsten Kapitel die genaue Problemdefinition und die Eingrenzung des Untersuchungsgegenstandes.

3 Problemdefinition

In diesem Kapitel sollen die Problemstellung genau beschrieben und die wichtigsten Begriffe und Konzepte definiert werden. Zudem sollen die für diese Arbeit relevanten Bereiche des Problems eingegrenzt werden, bevor dann in Kapitel 4 die Formalisierung der ausgewählten drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* beschrieben werden.

3.1 Beschreibung der Problemstellung

Untersuchungsgegenstand dieser Arbeit sind Trajektorien von zwei unterschiedlichen Typen von sich bewegenden Objekten (rote und blaue). Eine Auswahl dieser Objekte trifft in einem Raum aufeinander, wobei sich die interagierenden Objekte in ihren Aktionen gegenseitig beeinflussen. Dies führt zu einer Abfolge von Aktions- und Reaktionsbewegungen. Diese Bewegungen gilt es mittels formaler Beschreibungen und deren Umsetzung in räumlichen Algorithmen als Muster in Trajektorien zu erkennen. Eine wichtige Problemstellung ist zudem die Abgrenzung der Reaktionsbewegungsmuster mit Hilfe der Formalisierungen und Algorithmen gegenüber anderen Interaktionsformen, welche in Raum und Zeit einen ähnlichen Abdruck hinterlassen.

3.2 Eingrenzung des Untersuchungsgegenstandes

Diese doch sehr weit gefasste Problemdefinition soll hinsichtlich einer machbaren, aber zugleich anspruchsvollen und interessanten Aufgabenstellung für diese Masterarbeit sinnvoll eingegrenzt werden. Dabei sollen die Diskussionen und Erklärungen der wesentlichen Konzepte, welche dem Problemfeld zugrunde liegen, einen wichtigen Beitrag leisten zur sinnvollen Eingrenzung des Untersuchungsgegenstandes. Den Einstieg in diese schrittweise Einengung bildet die Diskussion der sich bewegenden Objekte, welche den eigentlichen Untersuchungsgegenstand darstellen.

3.2.1 Sich bewegende Objekte

Für die Beschreibung der beweglichen Objekte werden in dieser Arbeit Punkte (mit den Koordinaten x , y und der Zeit t) verwendet, wie es in Laube (2005) vorgeschlagen wird. Obwohl dadurch die räumliche Ausdehnung der Objekte (z.B. einer Zyklone) vernachlässigt wird, macht dies Sinn, da der Umgang mit Bewegungen in Raum und Zeit schon so eine genug grosse Herausforderung darstellt (Laube 2009). Dies bestätigt der Blick auf die bis anhin unternommenen Forschungsanstrengungen im Feld der Bewegungsmusteranalyse, welche beinahe alle bei der Beschreibung von beweglichen Objekten auf die Punktgeometrie zurückgegriffen haben (Laube 2009). In der euklidischen Geometrie wird ein Punkt als dimensionsloses (0-D) Objekt, welches eine Position im Raum ohne Ausdehnung aufweist, beschrieben (Laube 2005). Dabei wird die räumliche Position durch die Koordinaten x , y und allenfalls z spezifiziert (Laube 2005). Zu jeder gegebenen Zeit kann also die Position eines Punktobjekts als einfaches Tupel $(x,y,(z),t)$ fixiert werden, wobei neben den räumlichen Koordinaten x , y und z , eine weitere Koordinate t die jeweilige Zeit repräsentiert (Laube 2009).

3.2.2 Typen von Punktobjekten

Im Gegensatz zu den meisten bisherigen Forschungen zu Bewegungsmusteranalysen in der GIScience werden in dieser Arbeit interagierende Punktobjekte von zwei unterschiedlichen Typen untersucht. Dabei stellt sich die Frage, was denn eigentlich mit unterschiedlichen Typen gemeint ist. Da in dieser Arbeit die Bewegungen und das Verhalten im Fokus stehen, werden Typen von Punktobjekten dadurch definiert, dass sie in sozialen Interaktionssituationen unterschiedliche Verhaltensweisen an den Tag legen. Diese Verhaltensweisen müssen sich durch unverkennbare Eigenheiten kennzeichnen, welche als Regeln in eine formale Beschreibung integriert werden können. Aus diesen Erläuterungen ergeben sich die folgenden Typen von Interaktionspartnern:

- Punktobjekte von verschiedenen Spezies/Arten: Interspezifische Interaktionsformen, wie zum Beispiel das Verhalten zwischen Räubern und ihrer Beute oder die Bewegungsmuster zwischen Objekten zweier Arten, welche um dieselben Ressourcen konkurrieren.
- Punktobjekte verschiedener Interessensgruppierungen: Interaktionsmuster zwischen Punktobjekten mit unterschiedlichen Interessen, Absichten und Hintergründen, wie beispielsweise die Interaktion zwischen einem Passanten und einem Dieb oder das Aufeinandertreffen von verschiedenen Fanggruppierungen nach einem Fussballspiel.
- Punktobjekte mit unterschiedlichem Geschlecht: Zum Beispiel soziale Interaktion zwischen Männchen und Weibchen derselben Spezies während der Paarungszeit.

3.2.3 Anzahl der beteiligten Punktobjekte

Neben der Art der Punktobjekte stellt sich auch die Frage nach der Anzahl der beteiligten Punktobjekte. An den in dieser Arbeit formalisierten Reaktionsbewegungsmustern sind ein Punktobjekt eines ersten Typs und ein Punktobjekt eines zweiten beteiligt (1:1-Fall). Nach den Erläuterungen zur Repräsentationsweise, sowie zur Art und Anzahl der beteiligten Objekte folgen im nächsten Abschnitt Ausführungen zu den Räumen, welche die Rahmenbedingungen der Bewegungsmuster darstellen.

3.2.4 Modellierung des Raumes

Neben der Modellierung von sich bewegenden Objekten gilt es weiter, die den Bewegungsmustern zu Grunde liegenden Räume zu modellieren (Laube 2009). Dabei stehen diverse Modelle zur Verfügung, welche in Abbildung 3.1 dargestellt sind: Homogener euklidischer Raum (a), eingeschränkter euklidischer Raum (b), Raum-Zeit-Aquarium (c), heterogener feldbasierter Raum (d), irreguläre Tessellation (e) und Netzwerk-Raum (f). Eine ausführliche Diskussion aller dieser sechs verschiedenen Raummodelle ist in Laube (2009) zu finden. In der Folge wird nur auf die für diese Arbeit relevanten Modelle eingegangen. Dies sind der homogene euklidische Raum, der heterogene feldbasierte Raum, die irreguläre Tessellation sowie der Netzwerkraum. Die Ausführungen in diesem Abschnitt zu diesen Räumen basieren auf den Erläuterungen von Laube (2009). Dabei sollen in Form einer Übersicht diese vier Raummodelle kurz vorgestellt werden. Die genauen Beweggründe für die Verwendung dieser

Räume sowie die Erläuterungen zu deren Einsatz bei der Modellierung der individuellen und räumlichen Perspektive von Reaktionsbewegungsmustern folgt in Kapitel 3.2.7.

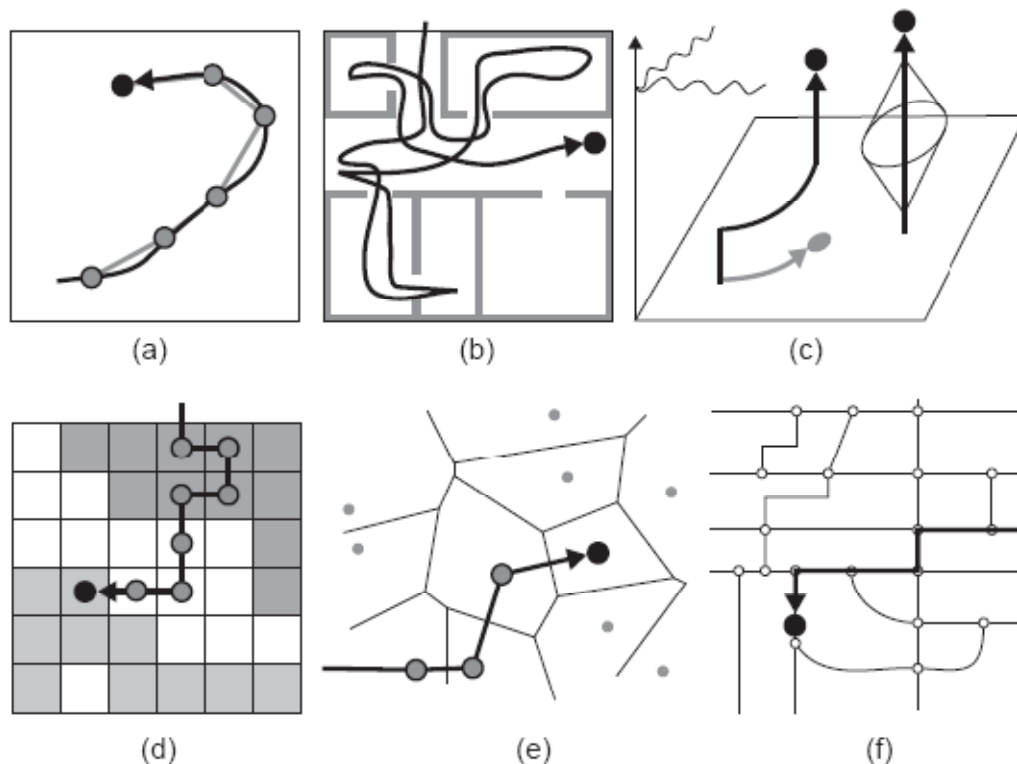


Abbildung 3.1: Verschiedene Raummodelle für die Analyse von Punktbewegungen (Laube 2009)

Darstellungen von sechs verschiedenen Raummodellen, welche verwendet werden können, um die den Bewegungen unterliegenden Räume abzubilden. Abgebildet sind Homogener euklidischer Raum (a), eingeschränkter euklidischer Raum (b), Raum-Zeit-Aquarium (c), heterogener feldbasierter Raum (d), irreguläre Tessellation (e) und Netzwerk-Raum (f).

Euklidischer Raum

Der homogene, zweidimensionale euklidische Raum R^2 ist die einfachste mögliche Modellierungsart. Darin können durch die sich bewegenden Punktoobjekte alle Positionen eingenommen werden. Denn die Bezeichnung homogen widerspiegelt die Annahme, dass der Raum ohne jegliche Hindernisse ausgestattet ist und dadurch die Bewegungen und Positionierungen der Punktoobjekte nicht beeinflusst werden. Dies bedeutet, dass der Kontext (z.B. Bodenbeschaffenheit, Nahrungsmittelvorkommen, etc.) der Bewegungen nicht berücksichtigt wird. Die Distanz in solchen Räumen wird als euklidische Distanz bezeichnet und repräsentiert die lineare Verbindung zwischen zwei Positionen im Raum. Der euklidische Raum eignet sich speziell gut für die Modellierung von Bewegung in Kombination mit dem Trajektorienmodell, welches im nächsten Abschnitt noch genauer erläutert wird. Das euklidische Raummodell wird häufig verwendet, falls die Distanzbeziehung zwischen Punktoobjekten im Fokus der Untersuchungen steht. Zudem kann diese Modellierungsart als sehr plausible Annäherung für viele räumliche Phänomene gesehen werden und eignet sich zudem sehr gut für die Implementierung von Analysen und Simulationen. Aus den genannten Gründen wird dieses Raummodell in den Forschungsbereichen Data Mining und Computational Geometry sehr häufig bei der Entwicklung von effizienten Algorithmen zur Detektion von Bewegungsmustern angewendet.

Der homogene euklidische Raum eignet sich auch sehr gut für die Analyse von Reaktionsbewegungsmustern und zwar speziell für die Betrachtung mit Hilfe der individuellen Perspektive, bei der die Distanzbeziehung zwischen den Punktobjekten im Zentrum der Untersuchungen steht. Genauere Erklärungen zur individuellen Perspektive folgen in Abschnitt 3.2.7.

Trotz der genannten Vorteile des homogenen euklidischen Raums gibt es auch Nachteile, wie beispielsweise der Umstand, dass in der realen Welt kaum je Bewegungen in objektlosen Räumen stattfinden. Daraus resultiert die Problematik, dass wichtige Umwelteinflüsse auf die untersuchten Bewegungen vernachlässigt werden und daher die Gefahr besteht, dass man die Einflüsse von Objektinteraktionen überbewertet. Die Umwelteinflüsse auf das Bewegungsverhalten werden als 1st-order Effekte, die gegenseitige Beeinflussung der Punktobjekte als 2nd-order Effekte bezeichnet. Bei der Verwendung des euklidischen Raummodells werden also die 1st-order Effekte vernachlässigt und alle sich abspielenden Bewegungsmuster als 2nd-order Effekte interpretiert. Dies gilt es bei der Verwendung dieses Modells zu berücksichtigen.

Während sich die an einem Reaktionsbewegungsmuster beteiligten Punktobjekte aus Sicht der individuellen Perspektive in einem leeren Raum bewegen, so interagieren sie aus der räumlichen Perspektive (siehe Erklärung in Kapitel 3.2.7) in inhomogenen Räumen. Um diese Betrachtungsweise zu modellieren, stehen der heterogene feldbasierte Raum, die irreguläre Tessellation sowie der Netzwerkraum zur Verfügung. Im Vergleich zum euklidischen Modell, bei welchem die den Bewegungen unterliegenden Räume als kontinuierlich angesehen werden, wird der Raum in diesen drei Ansätzen als diskret modelliert. In der Folge werden diese drei Raummodelle kurz erläutert.

Heterogener feldbasierter Raum

Für die Modellierung von heterogenen diskreten Räumen steht das feldbasierte konzeptionelle Datenmodell sowie die entsprechende Datenstruktur Raster zur Verfügung. Ein Beispiel für ein solches heterogenes Raster ist in der Abbildung 3.1 (d) dargestellt. Solche diskreten feldbasierten Räume eignen sich speziell gut für die Modellierung von Bewegungen, welche sich als Folge von Interaktionen und Reaktionen mit der inhomogenen Umwelt (z.B. Verfügbarkeit einer Ressource) ergeben. Aus diesem Grund findet dieses Raummodell grosse Anwendung im Forschungsbereich der biologischen Verhaltensforschung, wobei die den Tieren unterliegenden inhomogenen Habitate sehr gut abgebildet werden können. Aufgrund der fixen Zellengrösse des Rasters stösst allerdings auch dieses Modell an seine Grenzen. Die fixe Grösse der Rasterzellen führt dazu, dass sowohl die Schrittlängen der Bewegungen, als auch die Bewegungsrichtungen zwischen den Zellen eingeschränkt sind.

Irreguläre Tessellation

Eine weitere Möglichkeit um inhomogene diskrete Räume abzubilden, bietet das Raummodell der irregulären Tessellation. Diese Art der Raummodellierung wurde für das Abbilden von Bewegungsdaten von Mobiltelefonen entwickelt. Bei der Bewegung im Raum wechselt von Zeit zu Zeit der Mobilfunkturn, von welchem das Signal empfangen wird. Mit Hilfe der Abfolge der

verwendeten Mobilfunkantennen können die Bewegungspfade der Personen, welche sich mit einem Mobiltelefon bewegen, rekonstruiert werden. Für die Modellierung solcher Bewegungsdaten eignet sich die Unterteilung des Raumes aufgrund der Verteilung der Mobilfunktürme in eine irreguläre Tessellation. Dazu werden in der Regel Voronoi-Diagramme verwendet. Die Bewegungen sind daher modelliert als Verschiebung von einer Zelle zur nächsten, wobei sich die bewegenden Punktobjekte jeweils über ein gewisses Zeitintervall in derselben Zelle aufhalten. Aufgrund der in vielen Fällen doch sehr grossen Flächen dieser Zellen stösst die irreguläre Tessellation bei der Modellierung von exakten Bewegungspfaden an ihre Grenzen.

Netzwerkraum

Zum Abschluss der Vorstellung der Modellierungsmöglichkeiten von heterogenen Räumen wird hier noch kurz auf den Netzwerkraum eingegangen. Dieses Raummodell ist ein häufig verwendeter Ansatz zur Abbildung von menschlichen Bewegungen, denn die Mehrheit der Bewegungen von Menschen spielt sich in irgendeiner Art in einem Netzwerk ab. Beispiele dafür sind Strassennetze, öffentliche Verkehrsnetze oder Flugverkehrsnetze. Ein solches Strassennetz ist in der Abbildung 3.1 (f) dargestellt. Mit Hilfe des Netzwerkmodells können die genauen Positionen von Strassen und Kreuzungen detailliert modelliert werden.

3.2.5 Abbild von Bewegungen in Trajektorien

Um Bewegungen von Punktobjekten untersuchen zu können, muss das Konzept der Bewegung und die Möglichkeit zu dessen Repräsentation zuerst verstanden werden. Aus diesem Grund folgen in diesem Abschnitt eine kurze Definition von Bewegung und eine Erklärung, wie diese mit Hilfe von Trajektorien abgebildet werden kann.

Bewegung

In der GIScience kann bei der Definition von Bewegung die Konzeptualisierung von Veränderung (Wechsel) herbeigezogen werden (Laube 2005). Veränderung kann in zwei verschiedenen Ausprägungen vorkommen: Einerseits als Veränderung des Zustandes eines Objektes (z.B. Auftauchen oder Verschwinden eines Objekts) und andererseits als Bewegung eines Objekts bei gleichzeitiger Veränderung oder dem Gleichbleiben der geometrischen Form (Frank 2001). Bewegung wird in dieser Arbeit, wie es in Laube (2009) vorgeschlagen wird, als Veränderung der räumlichen Position eines Objekts über die Zeit definiert. Für die Betrachtung von Bewegung gibt es wiederum zwei mögliche Perspektiven, die Eulersche und die Lagrangesche Betrachtungsweise (Turchin 1998). Während die Eulersche Sichtweise Bewegungen als Veränderungen an einem fixen Punkt im Raum auffasst, bezeichnet der Lagrangesche Blickwinkel die Veränderungen der Position eines beweglichen Objekts entlang einer Trajektorie als Bewegungen (Turchin 1998). Beispiele für die Eulersche Perspektive sind die Vektorrepräsentation von Gletscherbewegungen oder geographische Diffusionsprozesse im Raum (Laube 2005). Die Aufzeichnungen von Bewegungen von Menschen, Tieren oder Fahrzeugen mit Hilfe von GPS sind Beispiele für das Lagrangesche Konzept von

Bewegung (Laube 2005). In dieser Arbeit steht die Analyse der Lagrangeschen Bewegung im Zentrum, wie es in Laube (2005) gemacht wurde. Mit Hilfe der räumlichen Perspektive (siehe Erklärung in Kapitel 3.2.7) werden in dieser Arbeit allerdings auch Aspekte der Eulerschen Betrachtungsweise berücksichtigt.

Trajektorien

Mit Hilfe der heute zur Verfügung stehenden grossen Palette an Lokalisierungstechnologien, wie beispielsweise GPS oder VHF Radio-Telemetrie, können die Bewegungen von Objekten aufgezeichnet werden (Laube 2005). Durch diese Aufzeichnung werden die Positionen von einer Anzahl von sich bewegenden Punktobjekten über eine bestimmte Zeitperiode aufgezeichnet, welche dann in Form von polygonalen Linien als Trajektorien für jedes einzelne Punktobjekt betrachtet werden können (Andersson et al. 2008). Der Pfad eines einzelnen Punktobjekts kann also als kontinuierliche Kurve in Raum und Zeit gesehen werden (Laube 2009). In der Realität wird aber nur eine diskrete Repräsentation dieses Bewegungspfades gespeichert, da Lokalisierungstechnologien nur in der Lage sind, einzelne diskrete Positionen zu speichern (Laube 2009). Dies bedeutet, dass diese Positionsangaben zuerst mit Hilfe von verbindenden geraden Linien zu Trajektorien diskretisiert werden müssen, welche dann als Approximation des „wahren“ Bewegungspfades gesehen werden können (Laube 2009). Dabei wird angenommen, dass sich ein Punktobjekt zwischen den Zeitpunkten t_i und t_{i+1} , an welchen Positionsangaben verfügbar sind, mit konstanter Geschwindigkeit auf einer geraden Linie zwischen den beiden Punkten bewegt (Andersson et al. 2008). Dies gilt es bei der Analyse, Formalisierung und Detektion von Reaktionsbewegungsmustern aus Trajektorien in dieser Arbeit zu berücksichtigen.

3.2.6 Bewegungsmuster

Es ist also das Ziel dieser Arbeit, Reaktionsbewegungsmuster basierend auf Trajektorien, zu analysieren. Doch was bedeutet eigentlich der Begriff Bewegungsmuster? Und was bedeutet er im Falle von reagierenden Punktobjekten? Laube (2009: 48) definiert in seiner Arbeit Bewegungsmuster als eine Beschreibung der Bewegungen von Individuen oder Gruppen von Individuen. Diese Beschreibung kann sich auf die den Bewegungen zugrunde liegenden Räume beziehen, muss sie aber nicht. In Dodge et al. (2008) werden unterschiedliche Typen von Bewegungsmustern ausgearbeitet, wobei grundsätzlich zwischen generischen Mustern und Verhaltensmustern unterschieden werden kann. Siehe dazu die Abbildung 2.1 im letzten Kapitel, in welcher die Taxonomie von Bewegungsmustern von Dodge et al. (2008) dargestellt ist. Die generischen Muster widerspiegeln die einzelnen Bausteine für alle komplexeren Muster von Punktobjekten, welche als Verhaltensmuster bezeichnet werden. Zu solchen Verhaltensmustern zählen zum Beispiel die *Herdenbildung* oder die Bewegungsabfolge von *Verfolgen/Entfliehen*. Schwerpunkt dieser Arbeit ist demzufolge die Analyse von Verhaltensmustern („behavioral patterns“). Aus der Kategorie der Verhaltensmuster speziell interessant für diese Arbeit sind Bewegungsmuster, welche als Folge von interagierenden, aufeinander

reagierenden und sich dabei gegenseitig beeinflussenden beweglichen Punktobjekten entstehen. Solche Verhaltensmuster werden mit dem Begriff Reaktionsbewegungsmuster beschrieben.

3.2.7 Räumliche vs. individuelle Perspektive

Für das Entstehen der im letzten Abschnitt definierten Bewegungsmuster gibt es, wie bereits angesprochen, zwei mögliche Betrachtungsweisen, die Eulersche und die Lagrangesche Betrachtungsweise (Turchin 1998). Diese beiden Betrachtungsweisen können auch auf die Interpretation von Reaktionsbewegungsmustern übertragen werden. Während die Lagrangesche Sichtweise die Entstehung von solchen Mustern auf der Ebene der interagierenden Individuen (individuelle Perspektive) beobachtet, steht bei der Eulersche Betrachtungsweise der Raum, in welchem die Bewegungen stattfinden, im Fokus (räumliche Perspektive). In dieser Arbeit werden die beiden Begriffe individuelle und räumliche Perspektive verwendet, da diese beiden Perspektiven nicht in allen Belangen mit den Betrachtungsweisen von Turchin (1998) übereinstimmen. Bewegungsmuster lassen sich sowohl aus der individuellen, als auch aus der räumlichen Perspektive in zwei unterschiedliche Phasen unterteilen. In einer ersten Phase wird durch eine Aktion eines beteiligten Punktobjektes der ganze Bewegungsablauf ausgelöst (Trigger, z.B. Annäherung). Als Antwort auf diese Aktion reagiert ein zweites Punktobjekt und führt ebenfalls eine Bewegung durch (Reaktion, Response, z.B. Fliehen). Dabei besteht zwischen der Aktion und der Reaktion eine zeitliche Beziehung. Aus Sicht der interagierenden Individuen (individuelle Perspektive) kommt der Distanz zwischen den Punktobjekten beim Auslösen der Reaktionsbewegung grosse Bedeutung zu. Aus einer räumlichen Perspektive kann das Reaktionsverhalten als Zeichen dafür gedeutet werden, dass sich Punktobjekte unterschiedlicher Arten auf derselben Fläche aufhalten wollen oder nicht.

Reaktionsbewegungsmuster Meiden

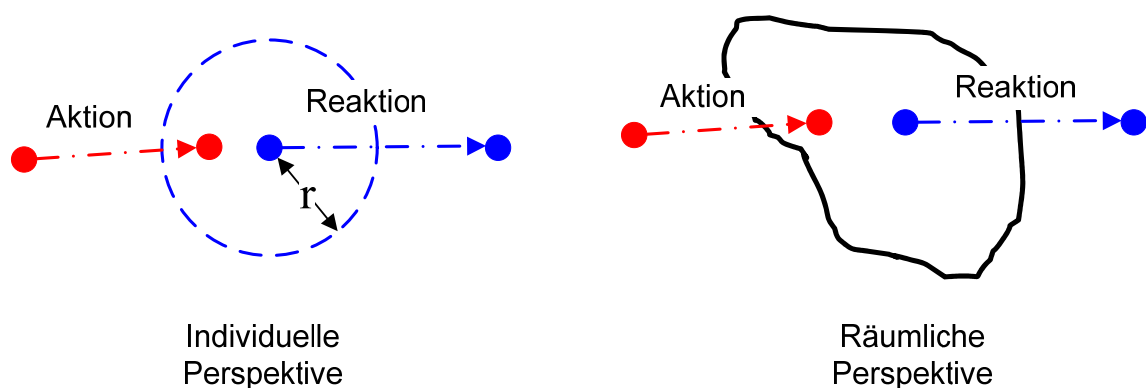


Abbildung 3.2: Vergleich der individuellen und der räumlichen Perspektive

Darstellung des raum-zeitlichen Abbildes von *Meiden* zwischen zwei Punktobjekten (rot und blau) aus individueller Perspektive (links) und aus räumlicher Perspektive (rechts).

Im Falle des Reaktionsbewegungsmusters *Meiden* ist aus individueller Perspektive die zu geringe Distanz zwischen den Punktobjekten verantwortlich für die Entstehung des Reaktionsverhaltens. Aus

räumlicher Sicht wird dagegen die gleichzeitige Nutzung eines umkämpften Territoriums als Ursache gesehen. Diese Sachverhalte sind in der Abbildung 3.2 dargestellt.

In dieser Arbeit sollen beide Betrachtungsweisen zur Anwendung kommen. Im Falle der individuellen Perspektive, bei der die Bewegungen primär mit Hilfe der Distanzbeziehungen zwischen den Punktobjekten analysiert werden, drängt sich die Modellierung des Raumes als homogenen euklidischen zweidimensionalen Raum auf. Bei der Verwendung der räumlichen Perspektive gilt es den homogenen euklidischen Raum mit einer Tessellation, einem Raster oder einem Netzwerk zu versehen, um die den Bewegungen zu Grunde liegenden Raumkompartimente abbilden zu können. Aus diesem Grund kann je nach Anwendungsfall zwischen den drei Modellen heterogener feldbasierter Raum, irreguläre Tessellation sowie Netzwerkraum ausgewählt werden. Während beispielsweise im Falle von territorialen Interaktionsformen zwischen Tieren eine irreguläre Tessellation am besten geeignet ist, so erweist sich das Netzwerkmodell bei Bewegungsdaten eines Freiluftspiels in einer Stadt als bestes Raummodell. Bei der Formalisierung der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* im nächsten Kapitel werden diese Raummodelle verwendet.

3.2.8 Betrachtungsmassstab

Bewegungen können sich auf sehr unterschiedlichen räumlichen und zeitlichen Skalen abspielen (Dodge et al. 2008). Die räumliche Betrachtungsweise reicht von der lokalen Skala im Bereich von Zentimetern (z.B. Augenbewegungen) bis hin zur globalen Skala (z.B. Flugzeugbewegungen) und die zeitliche Massstabspanne kennt sehr kurze bis sehr lange Bewegungen (Dodge et al. 2008). Bei der Betrachtung und Interpretation von Bewegungsmustern spielen die räumlichen und zeitlichen Betrachtungsmassstäbe eine entscheidende Rolle (Dodge et al. 2008). Da sich Reaktionsbewegungsmuster zwischen Punktobjekten durch kleinräumige Bewegungsabfolgen auszeichnen, werden in dieser Arbeit sowohl in zeitlicher wie auch in räumlicher Hinsicht die Bewegungen auf einer kleinen Skala analysiert.

3.2.9 Weitere Einflussfaktoren

Punktobjekte werden in ihren Bewegungen von diversen Faktoren beeinflusst, welche in Dodge et al. (2008) aufgeführt werden. Dazu gehören die intrinsischen Eigenschaften von Objekten (z.B. Maximalgeschwindigkeit), räumliche Einschränkungen (z.B. Barrieren, Netzwerke), Umwelteinflüsse (z.B. unterschiedliche Vegetation) und die gegenseitige Beeinflussung von Objekten. Das Hauptaugenmerk dieser Arbeit liegt auf Bewegungsmustern, welche durch Interaktion und Reaktion von Objekten ausgelöst werden. Diese Ursachen von Bewegungsmustern werden laut Laube (2009) als 2nd-order Effekte bezeichnet. Bei der gegenseitigen Beeinflussung der Punktobjekte spielen auch intrinsische Eigenschaften, wie zum Beispiel die Sichtweite von Objekten, eine Rolle. Die Umwelteinflüsse sowie räumliche Einschränkungen, welche als 1st-order Effekte bezeichnet werden, werden in dieser Arbeit nur bedingt einbezogen. Dies ist nur bei der räumlichen Perspektive der Fall,

bei der die den Bewegungen unterliegenden Raumkompartimente eine wichtige Rolle einnehmen. Dabei wird mit Hilfe der zur Auswahl stehenden Raummodelle heterogener feldbasierter Raum, irreguläre Tessellation sowie Netzwerkraum versucht, eine sinnvolle Unterteilung des Raumes abzubilden. Weitere Umwelteinflüsse wie zum Beispiel das Wetter werden in dieser Arbeit vernachlässigt.

3.2.10 Abgrenzung gegenüber anderen Interaktionsformen

Eine wichtige Problemstellung dieser Arbeit stellt die Abgrenzung der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* sowie *Meiden* gegenüber anderen sich ähnlich äussernden Interaktionsformen mit Hilfe geeigneter Formalisierungen dar. Ziel ist es, mit Hilfe der Ausarbeitung von einzigartigen Merkmalen dieser Reaktionsbewegungsmuster eine erfolgreiche Abgrenzung zu anderen Bewegungsmustern zu erreichen.

3.2.11 Definitive Aufgabenstellung

Aus den in diesem Unterkapitel 3.2 gemachten Ausführungen kann als Schlussfolgerung die folgende definitive Aufgabenstellung für diese Arbeit abgeleitet werden: Untersuchungsgegenstand sind Trajektorien von sich in einem zweidimensionalen, homogenen euklidischen Raum R^2 oder in einem zweidimensionalen, heterogenen diskreten Raum (Raster, Tessellation oder Netzwerk) bewegenden Punktobjekten (1 rotes und 1 blaues Objekt), welche unterschiedliche Verhaltensweisen aufgrund ihrer Art (Spezies), ihres Geschlechts oder ihres Interesses aufweisen. Mit Hilfe der individuellen und der räumlichen Perspektive sollen in einem ersten Schritt für die auf kleiner räumlicher und zeitlicher Skala stattfindenden Reaktionsverhalten *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden*, aufbauend auf der in der Literatur gemachten Beschreibungen, formale Definitionen ausgearbeitet werden (Kapitel 4). Bei dieser Formalisierung gilt es raum-zeitliche Regeln zu definieren, welche dann für die Entwicklung der Algorithmen zur Detektion dieser drei ausgewählten Reaktionsbewegungsmuster genutzt werden können (Kapitel 5). Ein wichtiges Problem dabei stellt die erfolgreiche Abgrenzung der Reaktionsbewegungsmuster gegenüber anderen Interaktionsformen dar. Dadurch soll sichergestellt werden, dass nicht zu viele systematische Fehldetektionen verzeichnet werden. Mit Hilfe von Experimenten (Kapitel 6) soll genau dies überprüft werden, um so eine Aussage über die Anwendbarkeit der entwickelten Algorithmen machen zu können (Kapitel 7-9).

4 Formale Beschreibung der Reaktionsbewegungsmuster

Von den Reaktionsbewegungsmustern, welche in Kapitel 2 kurz beschrieben wurden, sollen in diesem Kapitel die Muster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* im Detail analysiert werden. Während in der Literaturübersicht noch die biologische Sichtweise dieser Bewegungsmuster beschrieben wurde, soll der Fokus in diesem Kapitel auf die raum-zeitlichen Eigenheiten gelegt werden. Diese aus raum-zeitlicher Sicht typischen Merkmale der Reaktionsbewegungsmuster sollen die Grundlage bilden für die Ausarbeitung der formalen Definitionen. Dabei werden die drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* sowohl aus individueller als auch aus räumlicher Perspektive formalisiert. Um die Formalisierungen und Algorithmen an einem konkreten Beispiel mit Hilfe realer Raum-Zeit-Daten zu testen, wird in Kapitel 6 der Frequentie1550-Datensatz eingeführt. Dabei handelt es sich um Bewegungsdaten von Schülern, welche auf dem Strassennetzwerk von Amsterdam an einem Freiluftspiel teilnahmen. Für diesen konkreten Anwendungsfall wird die räumliche Perspektive so erweitert, dass die einzelnen Strassensegmente die Raumkompartimente darstellen, welche den Reaktionsbewegungsmustern zugrunde liegen. Die in diesem Kapitel ausgearbeiteten formalen Definitionen der individuellen und der räumlichen Perspektive werden anschliessend in Kapitel 5 bei der Entwicklung der Algorithmen genutzt.

4.1 Verfolgen/Entfliehen

4.1.1 Formalisierung individuelle Perspektive

Raum-zeitliche Beschreibung

Aus den in Kapitel 2 gemachten Ausführungen zum Reaktionsbewegungsmuster *Verfolgen/Entfliehen* lässt sich festhalten, dass das Reaktionsbewegungsmuster zwischen Verfolger und Fluchtobjekt nur zustande kommt, wenn sich die Interaktionspartner unter eine gewisse Minimaldistanz annähern. Dies verdeutlicht die Aussage von Furuichi (2002), dass sowohl Verfolger als auch Fliehender eine gewisse Sichtlimitierung haben und als Folge davon erst bei einer Unterschreitung einer kritischen Distanz auf die Anwesenheit des anderen reagieren. Dies führt dazu, dass Räuber ihre Beute nicht von überall her angreifen und Beuteobjekte ihrerseits nicht vor weit entfernten Räuber fliehen (Furuichi 2002). Bewegt sich also ein Punktobjekt (z.B. ein Räuber) in Richtung eines Zielobjekts einer anderen Spezies (z.B. Beutetier) und unterschreitet dabei das angesprochene Distanzkriterium, so wird das Reaktionsbewegungsmusters *Verfolgen/Entfliehen* grundsätzlich ausgelöst (Aktion) (Blythe et al. 1996). Um der genannten Einschränkung des Sichtfeldes gerecht zu werden, können bei der formalen Definition dieses Bewegungsmusters Umkreisradien um die beteiligten Punktobjekte miteinbezogen werden, welche die maximalen Sichtweiten widerspiegeln (Furuichi 2002). Dabei müssen die Sichtweiten und dementsprechend die Umkreisradien von zwei unterschiedlichen Punktobjekten nicht

immer gleich gross sein (Furuichi 2002). Taucht ein Beuteobjekt im Umkreis eines Räubers mit Radius r_{Vi} auf (Aktion), so begibt sich dieser auf die *Verfolgung* des Zielobjekts (Reaktion). Dies kann als Reaktionsbewegung auf die *Annäherung* der beiden Punktobjekte unter die kritische Distanz r_{Vi} bezeichnet werden. Tritt der Verfolger seinerseits in den Umkreis des gejagten Objekts mit Radius r_{Fj} , kommt es zu einem Fluchtversuch seitens des bedrohten Punktobjekts. Relevant für die Formalisierung ist primär die *Annäherung* der beiden Punktobjekte unter die kritische Sichtweite des Verfolgers, da ab diesem Zeitpunkt der Räuber seine Beute sehen und demzufolge die *Verfolgung* starten kann. Daher wird in der Abbildung 4.1 darauf verzichtet, den Umkreis des Fluchtobjekts darzustellen.

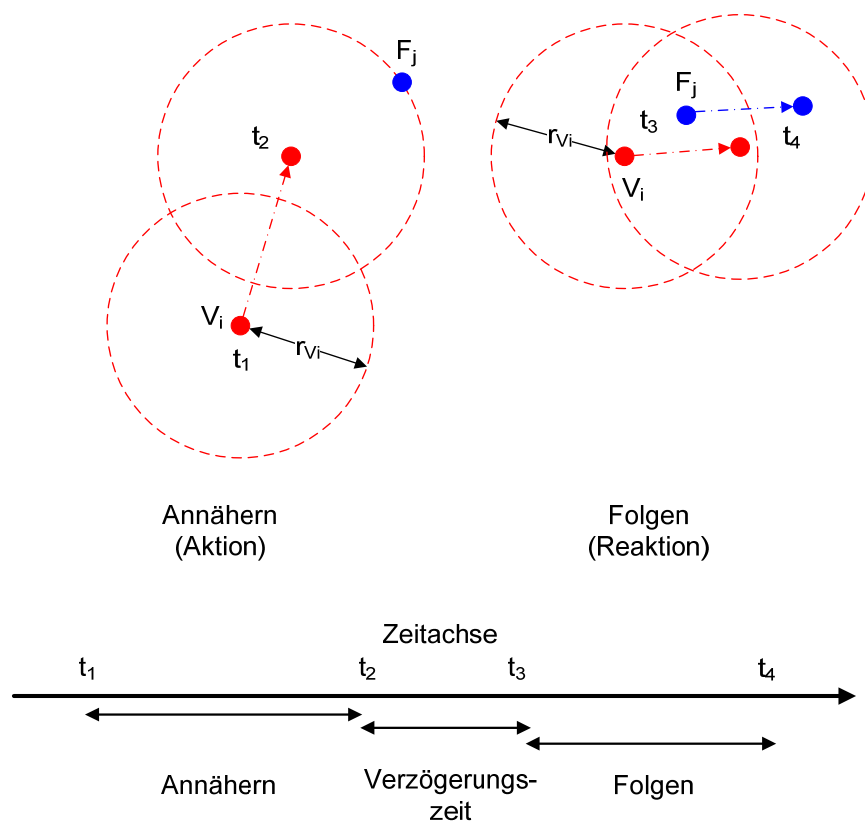


Abbildung 4.1: Verfolgen/Entfliehen aus individueller Perspektive

Raum-zeitliches Abbild der Situation bei der Auslösung des Reaktionsbewegungsmusters *Verfolgen/Entfliehen* aus der individuellen Perspektive. Die Unterschreitung der kritischen Distanz r_{Vi} zwischen Verfolger und Fluchtobjekt stellt dabei der Startpunkt dar (Aktion). Anschliessend an diese *Annäherung* beginnt die *Verfolgung*, wobei sich die beteiligten Punktobjekte in die gleiche Richtung bewegen (Reaktion).

Das Auftauchen eines Beuteobjekts im Sichtfeld des Jägers kann als Startpunkt eines Interaktionsmusters zwischen Verfolger und Fluchtobjekt gesehen werden. Wird die Distanz zwischen Räuber und seiner Beute kleiner als die kritische Distanz r_{Vi} , so sind die Bewegungen zwischen den beiden nicht mehr zufällig. Die Differenz zwischen dem Zeitpunkt des Beginns des *Zusammenseins* t_2 (Unterschreitung der kritischen Distanz) und dem Startpunkt der *Verfolgung* t_3 kann als *Verzögerungszeit* t_{delay} zwischen Aktion und Reaktion bezeichnet werden. Um sicherzustellen, dass es sich dabei um ein Reaktionsbewegungsmuster von *Verfolgen/Entfliehen* handelt, sollte die Verzögerungszeit zwischen der Unterschreitung der kritischen Distanz (Aktion) und dem Beginn der

Verfolgungsjagd (Reaktion) eine vordefinierte kritische Zeitdauer $t_{critiDelay}$ nicht überschreiten. Das Kriterium Verzögerungszeit wurde gewählt, da der Räuber unter grossem Zeitdruck steht. Will er seine Beute fangen, muss er so schnell wie möglich die *Verfolgung* aufnehmen und versuchen sein Ziel einzuholen. Somit ist zu erwarten, dass die zeitliche Verzögerung relativ kurz ausfällt wird.

Nach Beginn der Verfolgungsjagd versucht sich der Verfolger dem Zielobjekt anzunähern, während das verfolgte Punktobjekt seinerseits diverse Anstrengungen unternimmt, um sich von der Bedrohung zu entfernen (Furuichi 2002). Das Fluchtobjekt bewegt sich daher als ob es eine abstossende Kraft seitens seines Verfolgers verspüren würde (Furuichi 2002). Dasselbe gilt für das Verfolgerobjekt, welches sich verhält als ob es eine anziehende Kraft zur Beute verspüren würde (Furuichi 2002). Dies bedeutet, dass sich Räuber und Beute in dieselbe Richtung bewegen. Dabei gilt es zu berücksichtigen, dass neben den mehr oder weniger gleichen Bewegungsrichtungen der beteiligten Akteure auch die Ausrichtung des Kopfes in der realen Welt eine wichtige Rolle spielt bei der Wahl oder Änderung der Flucht- beziehungsweise Verfolgungsrouten (Furuichi 2002). Da es anhand der Trajektorien nicht möglich ist Rückschluss auf die Ausrichtung des Kopfes und daraus abgeleitet Aussagen über das genaue Blickfeld zu machen, wird das Sichtfeld wie in Andersson et al. (2008) bei der Detektion von *Leadership*-Mustern definiert. Dies ist durchaus zulässig, da die Bewegungen während einer Verfolgungsjagd sehr stark der Bewegungsabläufe in *Leadership* und *Trendsetting* gleicht, wobei das fliehende Objekt bei seiner Flucht den Verfolger anführt und mit seinem Verhalten dessen Bewegungen beeinflusst (Dodge et al. 2008). Die Frontregion $front(V_i)$ eines Verfolgers V_i wird definiert als Sektor des Umkreises mit Radius r_{V_i} um V_i mit einem Winkel von α_{V_i} (Andersson et al. 2008). Dabei haben die beiden Segmente, welche den Kreisbogen einschliessen, jeweils die Länge des Radius r_{V_i} , beginnen an der Position des Verfolgers V_i und haben im Vergleich zur Laufrichtung φ_{V_i} eine Orientierung von $\varphi_{V_i} \pm \alpha_{V_i}/2$ (Andersson et al. 2008). Siehe dazu die Abbildung 4.2.

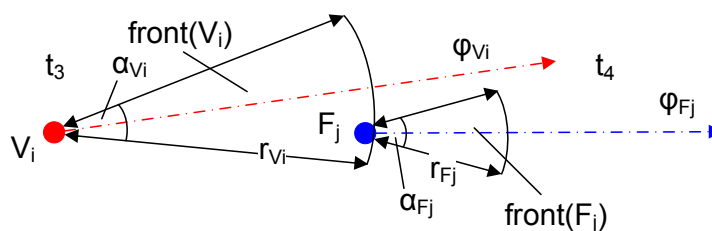


Abbildung 4.2: Definition der Frontregion nach Andersson et al. (2008)

Der Frontbereich $front(V_i)$ des Verfolgers V_i wird definiert wie in Anderson et al. (2008) vorgeschlagen, wobei sich ein Fluchtobjekt F_j bei einer *Verfolgung* über eine kritische Zeitdauer in diesem aufhalten muss.

Befindet sich ein Fluchtobjekte F_j zum Zeitpunkt t innerhalb des Frontbereiches $front(V_i)$ eines Verfolgers V_i , kann ausgesagt werden, dass V_i das Punktobjekt F_j zu diesem Zeitpunkt verfolgt (Andersson et al. 2008). Diese Bedingung muss über eine bestimmte minimale Zeitdauer t_{follow} eingehalten werden, damit vom Bewegungsmuster *Folgen* gesprochen werden kann. Im Vergleich zur Definition des Bewegungsmusters *Leadership* nach Andersson et al. (2008), welche einen Winkel β als maximale Abweichung zwischen den Bewegungsrichtungen φ_{V_i} und φ_{F_j} vorsieht, wird in der Definition des Bewegungsablaufes von *Verfolgen/Entfliehen* auf eine solche Beschränkung verzichtet,

da gerade bei einem Zickzack-Verlauf der Flucht (Miller und Cliff 1994) die Fortbewegungsrichtung der beteiligten Punktobjekte stark unterschiedlich sein kann.

Wie bereits bei der Literaturübersicht in Kapitel 2 beschrieben, gibt es zwei mögliche Ausgänge des Bewegungsmusters *Verfolgen/Entfliehen* (Furuichi 2002). Auf der einen Seite besteht die Möglichkeit, dass es dem Verfolger gelingt sein Zielobjekt einzuholen (Furuichi 2002). Dabei muss es der Jäger schaffen, sich bis auf eine gewisse Minimaldistanz an das verfolgte Punktobjekt anzunähern (Furuichi 2002). Ist diese Minimaldistanz in Form der Reichweite des Verfolgers $r_{catchVi}$ unterschritten, gibt es für das gejagte Punktobjekt kein Entrinnen mehr (Finney 1998). In Raum und Zeit äussert sich dies in einem Kreuzen der Bewegungspfade (*Intersektion*). Siehe dazu die Abbildung 4.3.

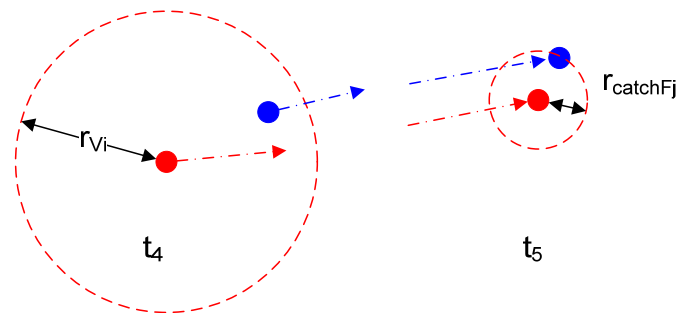


Abbildung 4.3: Ausgang *Verfolgen/Entfliehen* – Variante 1: erfolgreiche Jagd

Raum-zeitliches Abbild des Endes von *Verfolgen/Entfliehen* im Falle einer erfolgreichen Jagd, wobei es dem Verfolger gelingt, sich auf eine Minimaldistanz $r_{catchVi}$ an das Fluchtobjekt anzunähern und dieses keine Chance mehr hat zu entkommen.

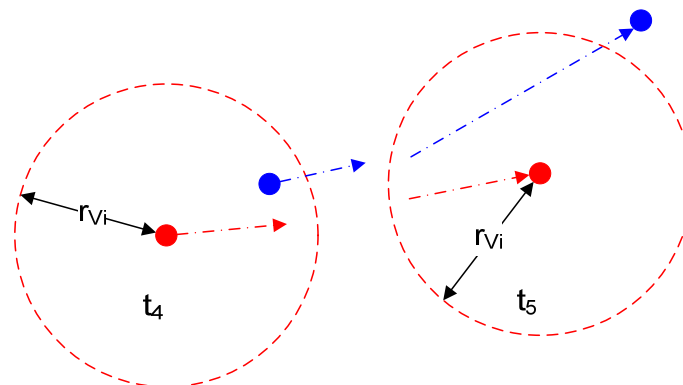


Abbildung 4.4: Ausgang *Verfolgen/Entfliehen* – Variante 2: erfolgreiche Flucht

Raum-zeitliches Abbild des Endes von *Verfolgen/Entfliehen* im Falle einer erfolgreichen Flucht des bedrohten Punktobjekts. Dem Fluchtobjekt gelingt es sich aus dem Sichtfeld des Verfolgers mit Radius r_{vi} zu bewegen und daher zu entkommen.

Auf der anderen Seite gibt es die Möglichkeit, dass das gejagte Objekt erfolgreich vor dem Verfolger fliehen kann (Furuichi 2002). In diesem Fall ist aus raum-zeitlicher Sicht relevant, dass sich das fliehende Punktobjekt nach einer bestimmten Zeit t_5 aus dem Blickfeld mit Radius r_{vi} der Verfolger bewegen kann, wie es in der Abbildung 4.4 zu sehen ist.

Definition von *Verfolgen/Entfliehen* individuelle Perspektive

Eine Anzahl potentieller Verfolger V_n und eine Menge von potentiellen Fluchtobjekten F_m bewegen sich in einem euklidischen Raum R^2 .

- Bedingung 1: Verkürzung der Distanz $d(V_i, F_j, t_2)$ zwischen zwei Punktobjekten $V_i, i \in n$ (potentieller Verfolger) und $F_j, j \in m$ (potentieller Flüchtling) unter die Sichtgrenze r_{Vi} von Punktobjekt V_i .
- Bedingung 2: Das Punktobjekt F_j befindet sich ab dem Zeitpunkt t_3 für eine minimale Zeitdauer t_{follow} in der Frontregion $front(V_i)$ von Punktobjekt V_i .
- Bedingung 3: Die zeitliche Verzögerung t_{delay} zwischen dem Zeitpunkt der Annäherung t_2 und dem Beginn der Verfolgung t_3 sollte kleiner sein als der kritische Wert $t_{critDelay}$.
- Bedingung 4: a) Verkürzung der Distanz $d(V_i, F_j, t_5)$ zwischen V_i und F_j unter die Fangweite bzw. Reichweite $r_{catchVi}$ von V_i (Beute ist gefangen).
b) Vergrößerung der Distanz $d(V_i, F_j, t_5)$ zwischen V_i und F_j über die Sichtweite r_{Vi} von V_i (Beute ist erfolgreich geflohen).

Formale Definition von *Verfolgen/Entfliehen* individuelle Perspektive

Bedingung 1: $\|V_i(x, y, t_2) - F_j(x, y, t_2)\| = d(V_i, F_j, t_2) < r_{Vi}$

Bedingung 2: $F_j \in front(V_i, t), t \in [t_3, t_4]$ und $t_4 - t_3 \geq t_{follow}$

Bedingung 3: $t_3 - t_2 = t_{delay} < t_{critDelay}$

Bedingung 4: a) $\|V_i(x, y, t_5) - F_j(x, y, t_5)\| = d(V_i, F_j, t_5) < r_{catchVi}$

b) $\|V_i(x, y, t_5) - F_j(x, y, t_5)\| = d(V_i, F_j, t_5) > r_{Vi}$

4.1.2 Formalisierung räumliche Perspektive

Raum-zeitliche Beschreibung

Im Vergleich zur individuellen Perspektive, bei der die Distanzbeziehung zwischen den Punktobjekten im Fokus der Untersuchung steht und die Unterschreitung der kritischen Distanz der Auslöser für das *Verfolgen/Entfliehen* darstellt, ist aus räumlicher Perspektive der Aufenthalt im gleichem Raum Gegenstand der Untersuchung. Sobald sich die Punktobjekte annähern und sich in demselben Kompartiment befinden (Aktion), wird mit einer zeitlichen Verzögerung das *Folgen* zwischen den Punktobjekten (Reaktion) ausgelöst. Eine solche Situation ist in der Abbildung 4.5 dargestellt.

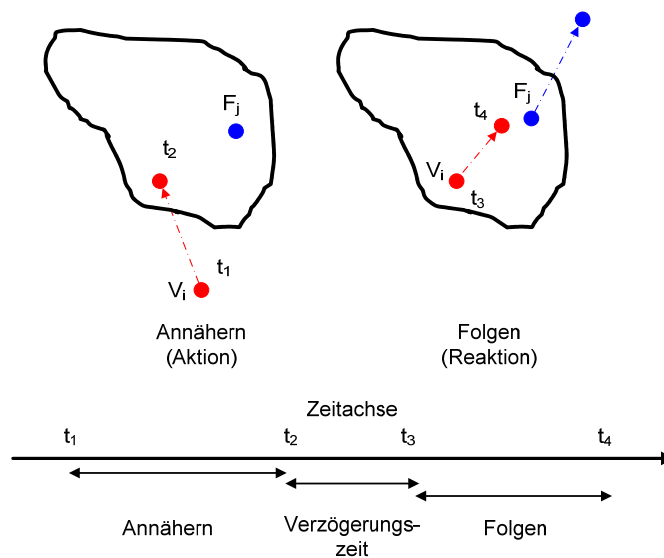


Abbildung 4.5: Auslösung von *Verfolgen/Entfliehen* aus räumlicher Perspektive

Raum-zeitliches Abbild der Auslösung von *Verfolgen/Entfliehen* aus der räumlichen Perspektive, falls sich ein Punktobjekt (Verfolger) in das Raumkompartiment eines zweiten (potentielle Beute) bewegt und die beiden sich als Folge davon im gleichen Raumkompartiment aufhalten (Aktion). Anschliessend beginnt die *Verfolgung* (Reaktion).

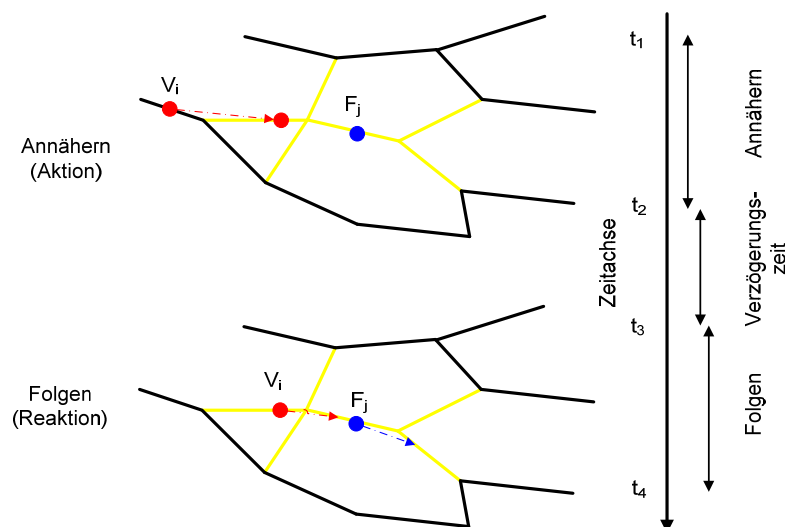


Abbildung 4.6: Auslösung von *Verfolgen/Entfliehen* aus räumlicher Perspektive auf dem Strassennetzwerk

Auslöser von *Verfolgen/Entfliehen* ist im Falle der Netzwerkperspektive das Eindringen des Verfolgers in die Segmentnachbarschaft des Fluchtobjekts (Aktion). Nachdem sich die beiden Objekte in derselben Nachbarschaft aufhalten beginnt die *Verfolgung* (Reaktion).

Wie bereits eingangs dieses Kapitels beschrieben, stellen für den konkreten Fall der Frequenzdaten die einzelnen Segmente des Strassennetzwerks die Kompartimente des Raumes dar, welche den Reaktionsbewegungsmustern zu Grunde liegen. Auslöser für den Start von *Verfolgen/Entfliehen* bildet aus räumlicher Perspektive je nach Definition des räumlichen Massstabs der Aufenthalt auf dem gleichen Strassensegment S_k oder in der gleichen Segmentnachbarschaft SN . Als Reaktion auf das *Zusammensein* in der gleichen Nachbarschaft begibt sich der Räuber nach einer zeitlichen Verzögerung auf die *Verfolgung* seiner Beute.

Während der *Verfolgung* müssen sich die Objekte im selben Raum in dieselbe Richtung bewegen. Im Falle der Segmentperspektive müssen sich die Punktobjekte auf den Strassensegmenten in dieselbe Richtung bewegen. Dabei kommen je nach Definition der räumlichen Ausdehnung der Segmentnachbarschaft die in der Abbildung 4.7 dargestellten drei Fälle vor. Im Falle, dass sich die Punktobjekte auf demselben Segment bewegen müssen (Nachbarschaftsgrösse 0), gilt die Situation wie in der Skizze 1 dargestellt nur als *Verfolgen/Entfliehen*, wenn sich die Punktobjekte auf dem gleichen Segment in dieselbe Richtung bewegen. Sind allerdings auch Bewegungen auf benachbarten Segmenten (Nachbarschaftsgrösse 1) oder sogar auf benachbarten der benachbarten Segmenten (Nachbarschaftsgrösse 2) zugelassen, so müssen die Bewegungsrichtungen durch die Orientierungen der Segmente korrigiert werden, damit bestimmt werden kann, ob sich die Punktobjekte in dieselbe Richtung bewegen und sich somit folgen.

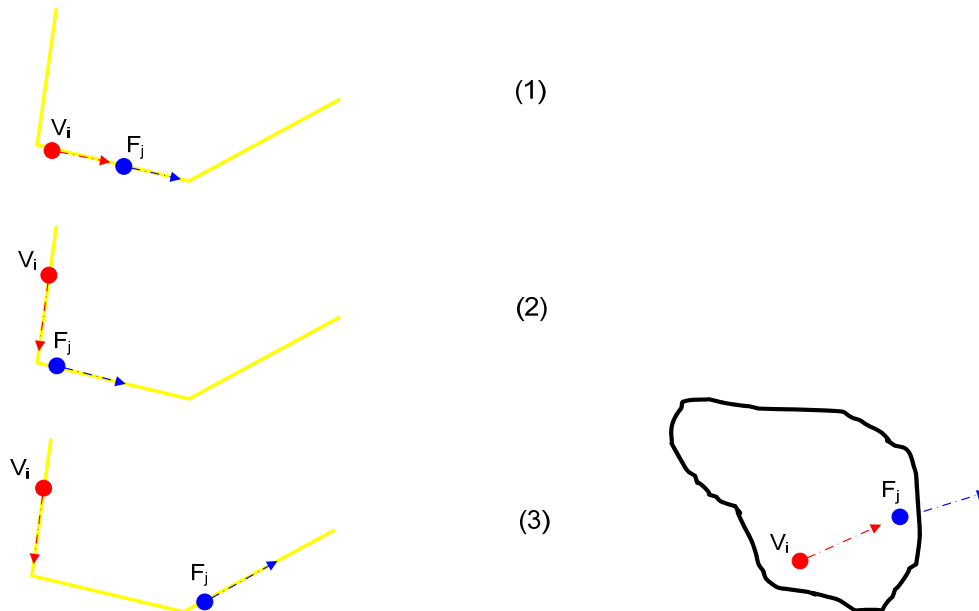


Abbildung 4.7: Bewegung in dieselbe Richtung während dem *Verfolgen/Entfliehen*

Im Falle der Netzwerkperspektive (links) müssen sich die an einer *Verfolgen/Entfliehen* beteiligten Punktobjekte auf dem Netzwerk in dieselbe Richtung bewegen und sich dabei in der vordefinierten Nachbarschaftsgrösse aufhalten. Die Fälle 1-3 zeigen Situationen mit einer *Verfolgung* bei unterschiedlicher Definition der Grösse der Segmentnachbarschaft. Für die herkömmliche Raumperspektive (rechts) müssen sich die Punkte während der *Verfolgung* in die gleiche Richtung bewegen und sich währenddessen im gleichen Raum aufhalten.

Das Ende des Reaktionsbewegungsmusters *Verfolgen/Entfliehen* hat, wie bereits bei der individuellen Perspektive angesprochen, zwei mögliche Ausgänge. Im Falle einer erfolgreichen Flucht gelingt es dem verfolgten Punktobjekte F_j sich aus der Segmentnachbarschaft SN_{V_i} des Verfolgers zu entfernen.

Für das erfolgreiche Einholen des Fluchtobjekts durch seinen Verfolger kann mit Hilfe der räumlichen Perspektive nur der Aufenthalt im gleichen kleinsten unterteilbaren Raumkompartiment herbeigezogen werden. Ob dies eine sinnvolle Formalisierung ergibt, ist primär abhängig von der Grösse dieser Raumeinheit. Ist die kleinste unterteilbare Raumeinheit wie im Falle der Frequentie1550-Daten ein Strassensegment mit einer durchschnittlichen Länge von 58 Metern, ist dies für die Formalisierung nicht ideal. Die Aussagekraft dieser Formalisierung hängt also stark mit der räumlichen Ausdehnung des kleinsten unterteilbaren Raumkompartiments zusammen. Abhilfe kann an dieser Stelle mit der im nächsten Abschnitt diskutierten Formalisierung von *Konfrontationen* gemacht werden. Da es bei einer erfolgreichen Jagd zu einer *Konfrontation* zwischen Räuber und seiner Beute kommt, kann auf die Formalisierung der *Konfrontation* ausgewichen werden. Bevor aber auf das Reaktionsbewegungsmuster *Konfrontation* eingegangen wird, soll in der Folge noch zusammenfassend die Definition von *Verfolgen/Entfliehen* aufgeführt werden.

Definition von *Verfolgen/Entfliehen* räumliche Perspektive

Eine Anzahl potentieller Verfolger V_n und eine Menge von potentiellen Fluchtobjekten F_m bewegen sich in einem Netzwerkraum NR^2 mit einer Menge von Segmenten S_l .

Bedingung 1: Aufenthalt eines Punktobjekts $V_i, i \in n$ (potentieller Verfolger) auf einem Segment $S_k, k \in l$, welches sich zum Zeitpunkt t_2 in der Segmentnachbarschaft $SN_{F_j} \in NR^2$ eines zweiten Punktobjekts $F_j, j \in m$ (potentieller Flüchtling) befindet.

Bedingung 2: Bewegungsrichtung $\varphi_{V_i}(t)$ von Punktobjekt V_i ist ab t_3 über eine minimale Zeitdauer t_{follow} gleich wie die Bewegungsrichtung $\varphi_{F_j}(t)$ von Punktobjekt F_j .

Bedingung 3: Die zeitliche Verzögerung t_{delay} zwischen dem Startzeitpunkt des Aufenthalts in derselben Segmentnachbarschaft t_2 und dem Beginn der *Verfolgung* t_3 sollte kleiner sein als der kritischer Wert $t_{critDelay}$.

Bedingung 4: a) Aufenthalt des Verfolgers V_i und des Flüchtlings F_j auf dem gleichen Segment S_k .
b) *Verlassen* der Segmentnachbarschaft SN_{V_i} durch das Fluchtobjekt F_j .

Formale Definition von *Verfolgen/Entfliehen* räumliche Perspektive

Bedingung 1: $V_i \in S_k, S_k \in SN_{F_i}$ und $SN_{F_j} \in NR^2$ zur Zeit t_2

Bedingung 2: $\varphi_{V_i}(t) = \varphi_{F_j}(t), t \in [t_3, t_4]$ und $t_4 - t_3 \geq t_{follow}$

Bedingung 3: $t_3 - t_2 = t_{delay} < t_{critDelay}$

Bedingung 4: a) $V_i, F_j \in S_k$ zur Zeit t_5

b) $F_j \in S_k, V_i \notin S_k$ und $S_k \notin SN_{V_i}$ zur Zeit t_5

4.2 Konfrontation

4.2.1 Formalisierung individuelle Perspektive

Raum-zeitliche Beschreibung

Wie in Kapitel 2 beschrieben können *Konfrontationen* sowohl in Form von Räuber/Beute-Interaktionen als auch in territorialen Konkurrenzsituationen vorkommen. Dabei weisen sie trotz den unterschiedlichen Beweggründen aus raum-zeitlicher Perspektive sehr ähnliche Verhaltensmuster auf. Zumindest für die grundsätzlichen Bewegungsabläufe kann eine einheitliche Formalisierung vorgenommen werden. Ausgelöst wird eine *Konfrontation* prinzipiell durch das *Annähern* zweier Punktobjekte. Aus den Ausführungen in Kapitel 2 geht zudem hervor, dass sich von kämpferischem Verhalten geprägte *Konfrontationen* zwischen zwei Punktobjekten durch schnelles Abwechseln von *Angriff* und *Verteidigung* beziehungsweise *Verfolgung* und *Entfliehen* der beteiligten Punktobjekte auszeichnen (Blythe et al. 1996). Anschliessend an *Konfrontationen*, welche aus territorialen Konkurrenzsituationen entstehen, kommt es in der Regel zu einer Verdrängung des Verlierers (Moorcroft 2008), was als Folge zu einer Separationsbewegung zwischen den Punktobjekten führt. Eine *Separation* ist auch im Falle einer Konkurrenz zwischen Räuber und Beute zu beobachten. Entweder aufgrund der erfolgreichen Flucht der Beute oder als Folge der Fortbewegung des Räubers nach dem Verzehr der Beute. Somit kann das grundsätzliche Abbild einer *Konfrontation* in Raum und Zeit aus der individuellen Perspektive als Abfolge der Bewegungsmuster *Annähern*, *Zusammentreffen* (*Intersektion*) und *Separieren* zusammengefasst werden, was in der Abbildung 4.8 dargestellt ist.

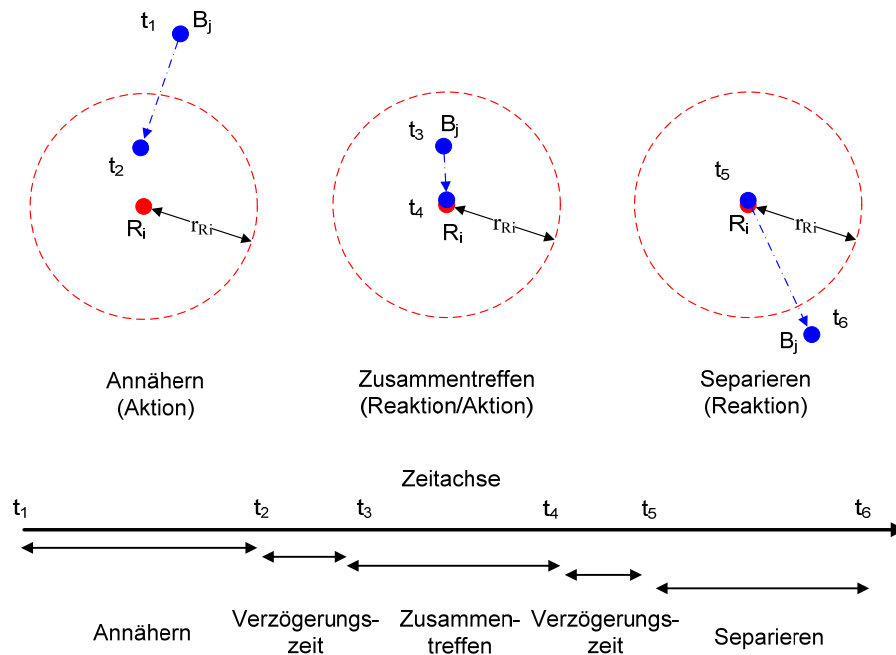


Abbildung 4.8: Raum-zeitlicher Ablauf einer Konfrontation aus der individuellen Perspektive

Aus individueller Perspektive wird eine *Konfrontation* ausgelöst durch das *Annähern* zweier Punktobjekte (blau und rot) unter eine kritische Distanz und das anschliessende *Zusammentreffen* dieser (Aktion). Nach dem *Zusammentreffen* erfolgt eine *Separation* (Reaktion).

Dabei muss angeführt werden, dass es sich bei dieser Formalisierung um eine starke Abstraktion der sich in der Realität abspielenden Bewegungspfade handelt. In Tat und Wahrheit sind neben der *Annäherung*, dem *Zusammentreffen* sowie der *Separation* in verschiedenen Ausprägungen von *Konfrontationen* (z.B. im Falle eines Kampfes) auch noch die in Blythe et al. (1996) beschriebenen schnellen Bewegungsabfolgen von *Angriff* und *Verteidigung* zu beobachten. Für solche sehr kleinräumigen Verhaltensmuster in Kämpfen ist es allerdings ungemein schwierig allgemeingültige Gesetzmässigkeiten zu entdecken. Nicht zuletzt, da es je nach Art und Zahl der beteiligten Punktobjekte zu stark unterschiedlichen Ausprägungen von Kampfsituationen kommt. Führt man sich beispielsweise den Kampf zwischen einem Löwen und einem Zebra sowie eine Schlägerei zwischen zwei Menschen vor Augen so wird unweigerlich klar, dass sich die kleinräumigen Bewegungsmuster während der *Konfrontation* zwischen diesen beiden Fällen stark unterscheiden können. Zudem gibt es neben der kämpferischen Auseinandersetzung auch *Konfrontationen*, welche sich durch zurückhaltende Verhaltensweisen der beteiligten Punktobjekte auszeichnen (Smith und Price 1973). Bei solchen *Konfrontationen* müsste die kleinräumige Formalisierung deutlich anders ausfallen als dies bei *kämpferischem Aufeinandertreffen* der Fall ist. Aufgrund der heute verfügbaren Qualität der Bewegungsdaten ist es zudem unsicher, ob Bewegungen, welche sich auf einer sehr kleinen Skala abspielen, auch wirklich in den aufgezeichneten Bewegungen erkennbar sind.

Aus diesen Gründen macht die in diesem Abschnitt vorgestellte, stark vereinfachte Formalisierung von *Konfrontationen* durchaus Sinn. Bei der Anwendung dieser Formalisierung und der daraus entwickelten Algorithmen (siehe Kapitel 5) ist es dringend notwendig, dass einem diese Abstraktion bewusst ist. Im Falle der Kenntnisse der genauen Bewegungsabläufe während einer *Konfrontation* kann die Formalisierung zudem beliebig erweitert werden. Daraus ergibt sich der Vorteil einer breiten Anwendbarkeit.

Dieser Vorteil zeigt sich bei der Analyse der Bewegungsdaten des Frequentie1550-Projekts. Dabei handelt es sich um einen Datensatz eines Freiluftspiels (Schnitzeljagd) in Amsterdam. Wie in Kapitel 6 bei der Beschreibung des Frequentie1550-Datensatzes noch ausführlicher diskutiert wird, war es laut Admiraal et al. (2007) die Aufgabe von Schülern in den Strassen von Amsterdam Checkpoints ausfindig zu machen und an diesen für ihr Team durch das erfolgreiche Lösen von Rätseln so viele Punkte wie möglich zu verdienen. Dabei war es auch möglich, andere Teams zu verfolgen, sich an diese anzunähern und diese anzurempeln, um eine *Konfrontation* zu starten (Admiraal et al. 2007). Bei einer solchen *Konfrontation* bekommt jeweils diejenige Mannschaft mit dem höheren Spielorden vom Konkurrenzteam Punkte zugesprochen (Admiraal et al. 2007). Dadurch, dass die in diesem Kapitel vorgestellte Formalisierung so allgemein gehalten wurde, kann sie auch auf die *Konfrontation* zwischen Spielern in diesem Freiluftspiel angewendet werden, was anhand der nachfolgenden Ausführungen gezeigt wird.

Auch im Falle des Frequentie1550-Spiels wird das Reaktionsbewegungsmuster *Konfrontation* aus Sicht der individuellen Perspektive ausgelöst durch das *Annähern* zweier Spieler unter eine kritische

Distanz r_{Ri} , welche im Normalfall der Sichtweite der beteiligten Punktobjekte entspricht. Ist dies geschehen, können sich die Punktobjekte sehen und nach einer bestimmten Verzögerungszeit r_{delay1} kommt es zu Reaktionsbewegungen. Falls sich beide Punktobjekte zu einer *Konfrontation* herausfordern möchten, kommt es mit grosser Wahrscheinlichkeit zu einem gegenseitigen Ansteuern und zu einem direkten *Ineinanderrennen* (Regel des Spiels). Diese Situation ist in der Abbildung 4.9 dargestellt. Das *Zusammentreffen* wird gekennzeichnet durch das Unterschreiten der Intersektionsdistanz $r_{Intersection}$ zwischen den beteiligten Punktobjekten.

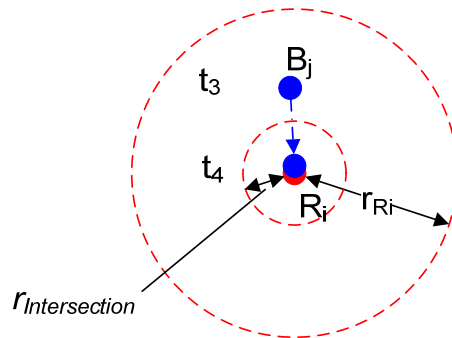


Abbildung 4.9: Zusammentreffen zwischen zwei Punktobjekten

Aus individueller Perspektive wird eine *Konfrontation* gekennzeichnet durch die Unterschreitung einer minimalen Intersektionsdistanz $r_{intersect}$ zwischen zwei Punktobjekten.

Allerdings ist durchaus auch denkbar, dass nur eines der Punktobjekte eine *Konfrontation* herbeiführen möchte und der zweite Beteiligte versucht einem Konflikt auszuweichen. Dies ist möglich, falls einem der beiden Punktobjekte eine Niederlage aufgrund ungleicher Kräfteverhältnisse (z.B. tieferer Orden, weniger Kraftreserven) droht. Diese beiden möglichen Fälle zeigen, dass die Verzögerungszeit t_{delay1} zwischen *Annäherung* und *Zusammentreffen* stark variabel ist. Weniger variabel erscheint die Verzögerungszeit t_{delay2} zwischen dem *Aufeinandertreffen* und der darauffolgenden Reaktion, der *Separation*. Da sich die beteiligten Punktobjekte im Freiluftspiel konkurrenzieren, haben sie weder die Absicht miteinander zu kooperieren noch sich aus Freundschaft zu treffen. Aus diesem Grund werden sie sich nach der Konfliktsituation wieder voneinander entfernen. Mit Hilfe dieser *Separation* und der Verzögerungszeit t_{delay2} zwischen dem *Zusammenprall* und der *Separation*, kann das Reaktionsbewegungsmuster *Konfrontation* von anderen Interaktionsformen wie das *Treffen* (*Meeting*) oder die *Kooperation* unterschieden werden.

Definition von *Konfrontation* individuelle Perspektive

Eine Anzahl Punktobjekt einer ersten Art (rot) R_n und eine Menge von potentiellen Interaktionspartnern einer zweiten Art (blau) B_m bewegen sich in einem euklidischen Raum R^2 .

Bedingung 1: Verkürzung der Distanz $d(R_i, B_j, t_2)$ zwischen zwei Punktobjekten $R_i, i \in n$ und $B_j, j \in m$ unter die Sichtgrenze r_{R_i} von Punktobjekt R_i .

Bedingung 2: Verkürzung der Distanz $d(R_i, B_j, t_4)$ zwischen den beiden Punktobjekten R_i und B_j unter die Intersektionsdistanz $r_{Intersection}$ ab t_4 über eine minimale Zeitdauer $t_{Intersection}$.

Bedingung 3: Die zeitliche Verzögerung t_{delay1} zwischen dem Zeitpunkt der *Annäherung* t_2 und dem Zeitpunkt des *Zusammentreffens* t_4 sollte kleiner sein als der kritische Wert $t_{critDelay1}$.

Bedingung 4: Erhöhung der Distanz $d(R_i, B_j, t_6)$ zwischen den zwei Punktobjekten R_i und B_j über die Sichtgrenze r_{R_i} von Punktobjekt R_i ab t_6 über eine minimale Zeitdauer $t_{Separation}$.

Bedingung 5: Die zeitliche Verzögerung t_{delay2} zwischen dem Zeitpunkt des *Zusammentreffens* t_4 und dem Zeitpunkt t_6 der Erhöhung der Distanz $d(R_i, B_j, t_6)$ auf einen Wert grösser/gleich r_{R_i} (Beginn *Getrenntsein*) sollte kleiner sein als der Wert $t_{critDelay2}$.

Formale Definition von *Konfrontation* individuelle Perspektive

Bedingung 1: $\|R_i(x, y, t_2) - B_j(x, y, t_2)\| = d(R_i, B_j, t_2) < r_{R_i}$

Bedingung 2: $\|R_i(x, y, t) - B_j(x, y, t)\| = d(R_i, B_j, t) < r_{Intersection}, t \in [t_4, t_5]$ und $t_5 - t_4 \geq t_{Intersection}$

Bedingung 3: $t_4 - t_2 = t_{delay1} < t_{critDelay1}$

Bedingung 4: $\|R_i(x, y, t) - B_j(x, y, t)\| = d(R_i, B_j, t) \geq r_{R_i}, t \in [t_6, t_7]$ und $t_7 - t_6 \geq t_{Separation}$

Bedingung 5: $t_6 - t_4 = t_{delay2} < t_{critDelay2}$

4.2.2 Formalisierung räumliche Perspektive

Raum-zeitliche Beschreibung

Betrachtet man das Reaktionsbewegungsmuster *Konfrontation* aus der räumlichen Perspektive, so tritt an Stelle der kritischen Distanz bei der *Annäherung* der Eintritt in dasselbe Raumkompartiment. Befinden sich also zwei Punktobjekte R_i und B_j in derselben Raumeinheit (*Zusammensein*), wird mit einer Verzögerungszeit t_{delay1} ein *Aufeinandertreffen* der beiden erwartet. Nach diesem *Zusammenprall* vergeht eine zweite Zeitverzögerung t_{delay2} bis sich die Punktobjekte wieder voneinander wegbewegen und sich als Folge nicht mehr im selben Raum aufhalten (*Getrenntsein*). Wählt man als Verzögerungszeit t_{delay2} zwischen *Zusammentreffen* und *Separation* einen nicht allzu grossen Wert, so kann das Reaktionsbewegungsmuster *Konfrontation* wie im Falle der individuellen Perspektive von anderen Interaktionsmustern wie *Kooperation* oder *Treffen*, welche als Abdruck in Raum und Zeit durchaus ähnlich sind, unterschiedenen werden.

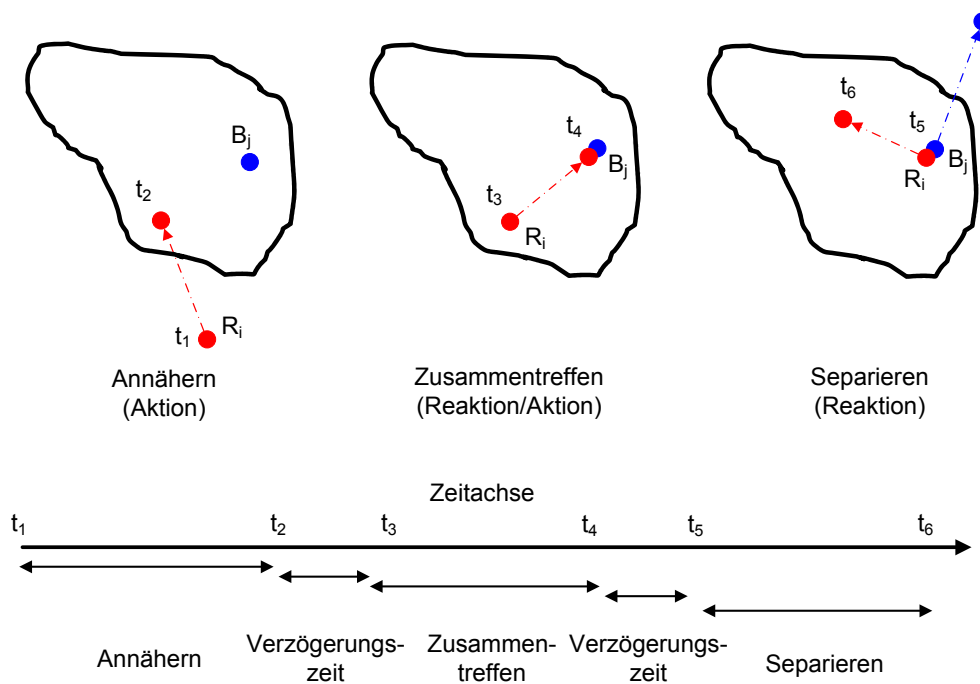


Abbildung 4.10: Raum-zeitlicher Ablauf einer Konfrontation aus der räumlichen Perspektive

Aus räumlicher Perspektive wird eine *Konfrontation* ausgelöst durch das Eintreten eines Punktobjekts in das Raumkompartiment eines zweiten, wobei es dabei zu einer Begegnung der beiden kommt (Aktion). Nach dem *Zusammentreffen* erfolgt eine *Separation* und eines der beiden Punktobjekte verlässt die Umgebung des anderen (Reaktion).

Transferiert man die gemachten Formalisierungen der räumlichen Perspektive auf ein Strassennetzwerk, so treten an Stelle fixer Raumausschnitte die Segmente, auf denen sich die Punktobjekte bewegen, sowie je nach Wahl der räumlichen Skala deren Nachbarsegmente. Siehe dazu die Abbildung 4.11. Das Reaktionsbewegungsmuster *Konfrontation* wird also ausgelöst durch den Eintritt eines Punktobjekts R_i in die Segmentnachbarschaft SN_{B_j} eines zweiten Punktobjekts. Ist dies geschehen, kommt es nach einer zeitlichen Verzögerung t_{delay1} zu einer Bewegung auf dem gleichen Segment aufeinander zu und anschliessend zu einem *Zusammentreffen* der Punktobjekte und somit zur Konfliktsituation. Vergeht wiederum eine gewisse Verzögerungszeit, ist wie bereits beim

allgemeinen räumlichen Fall beschrieben mit einer *Separation* zu rechnen. Dies führt dazu, dass sich die beteiligten Punktobjekte nicht mehr in derselben Segmentnachbarschaft befinden.

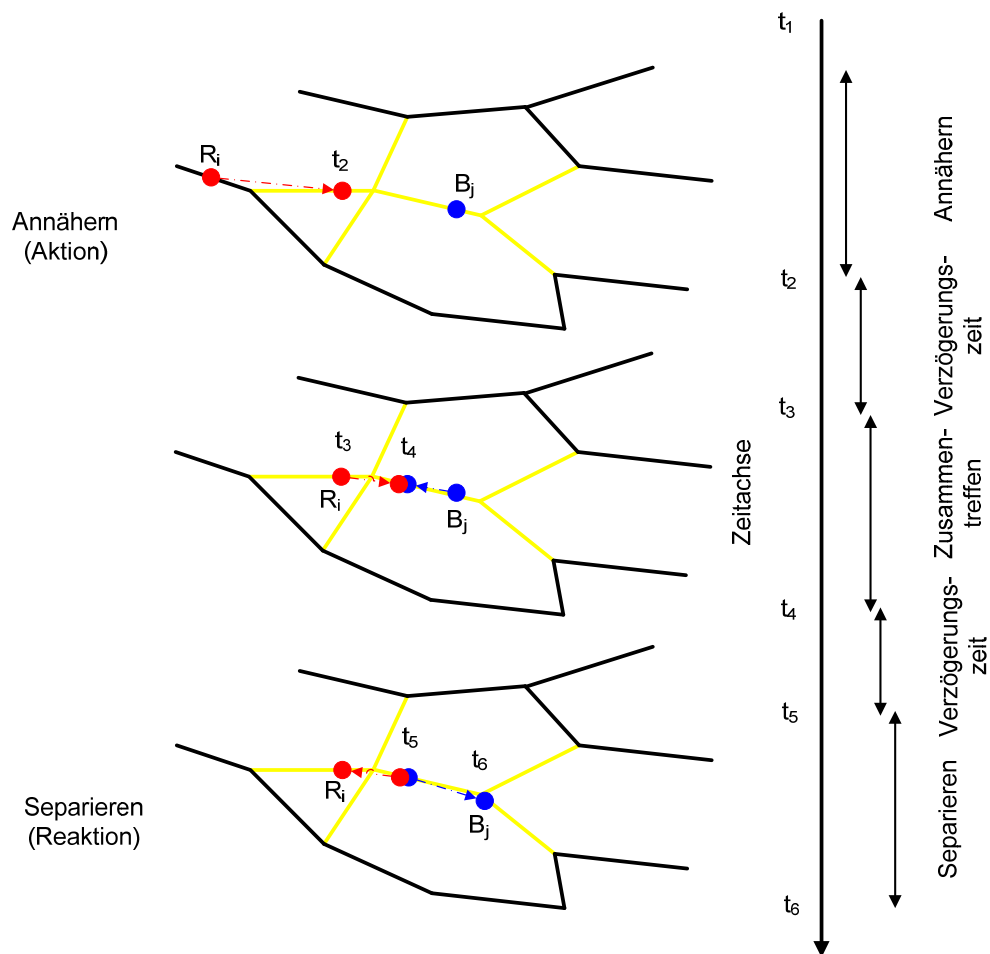


Abbildung 4.11: Raum-zeitlicher Ablauf einer Konfrontation aus der Netzwerkperspektive

Aus der Netzwerkperspektive wird eine *Konfrontation* ausgelöst durch das Eintreten eines Punktobjekts R_i in die Segmentnachbarschaft SN_{B_j} eines zweiten Objekts B_j (oben), wobei sich die Punktobjekte aufeinander zu bewegen und zusammentreffen (Mitte). Nach dem *Zusammentreffen* erfolgt eine *Separation* und eines der beiden Punktobjekte verlässt die Nachbarschaft des anderen (unten).

Der Unterschied zwischen der individuellen und der räumlichen Perspektive ist im Falle der *Konfrontation* neben den verschiedenen Raumbetrachtungen auch die Formalisierung des *Zusammentreffens*. Während bei der individuellen Perspektive die Punktobjekte eine minimale Intersektionsdistanz $r_{Intersection}$ unterschreiten müssen, wird das *Zusammentreffen* aus der räumlichen Perspektive gekennzeichnet durch die Bewegung aufeinander zu auf dem gleichen Segment während einer Zeitspanne $t_{Intersection}$. Da das Segment eine gerade Linie darstellt, können sich Punktobjekte nur in zwei Richtungen bewegen. Dies bedeutet, dass sich die Bewegungsrichtungen der Punktobjekte im Fall einer Bewegung aufeinander zu um 180 Grad unterscheiden müssen. Dabei gilt es zu beachten, dass trotz der durchgeführten Vorprozessierungsschritte die Bewegungen nicht immer perfekt auf dem Segment erfolgen und daher die Differenz zwischen den Bewegungsrichtungen der Punktobjekte etwas von 180 Grad abweichen kann. Um sicherzustellen, dass sich Punktobjekte nicht voneinander wegbewegen, werden die Orientierung der Segmente φ_{Sk} sowie die Distanzen $d(R_i, SK_{Start}, t_{3,4})$ bzw.

$d(B_j, SK_{Start}, t_{3,4})$ der Punktobjekte zum Startknoten des Segments SK_{Start} genutzt. Zwei Punktobjekte bewegen sich nur aufeinander zu, wenn dasjenige Punktobjekt, welches sich in Richtung der Segmentorientierung bewegt, sich näher beim Startpunkt SK_{Start} des Segments befindet. Dies ist in der Abbildung 4.12 dargestellt.

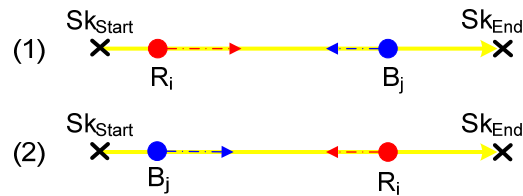


Abbildung 4.12: Voraussetzungen für die Konvergenz zwischen Punktobjekten auf einem Segment

Das rote Punktobjekt R_i befindet sich näher beim Startpunkt SK_{Start} des Segments S_k und bewegt sich in Richtung der Segmentorientierung von S_k (Abb. 1). Falls die Differenz zwischen den Bewegungsrichtungen der beiden Punktobjekte R_i und B_j 180° beträgt und die beiden Punktobjekte nicht stehen bleiben ($v_{R_i} > 0, v_{B_j} > 0$), müssen sich die beiden Punktobjekte aufeinander zu bewegen. Dasselbe gilt für den Fall, in dem das Punktobjekt B_j näher beim Startpunkt SK_{Start} des Segments ist (Abb. 2).

Definition von Konfrontation räumliche Perspektive

Eine Anzahl Punktobjekte einer ersten Art (rot) R_n und eine Menge von potentiellen Interaktionspartnern einer zweiten Art (blau) B_m bewegen sich in einem Netzwerkraum NR^2 mit einer Menge von Segmenten S_l .

Bedingung 1: Aufenthalt eines Punktobjekts $R_i, i \in n$ auf einem Segment $S_k, k \in l$, welches sich zum Zeitpunkt t_2 in der Segmentnachbarschaft $SN_{B_j} \in NR^2$ eines zweiten Punktobjekts $B_j, j \in m$ befindet.

Bedingung 2: Bewegung der beiden Punktobjekte R_i und B_j aufeinander zu, was bedeutet, dass die Bewegungsrichtungen der beiden Punktobjekte φ_{R_i} und φ_{B_j} einander entgegengesetzt sind vom Zeitpunkt t_3 über eine kritische Zeitdauer $t_{Intersection}$. Um sicherzustellen, dass sich Punktobjekte nicht voneinander wegbewegen, werden die Orientierung der Segmente φ_{S_k} sowie die Distanzen $d(R_i, SK_{Start}, t_{3,4})$ bzw. $d(B_j, SK_{Start}, t_{3,4})$ der Punktobjekte zum Startknoten des Segments SK_{Start} genutzt.

Bedingung 3: Die zeitliche Verzögerung t_{delay1} zwischen dem Zeitpunkt des Aufenthalts in der gleichen Segmentnachbarschaft t_2 und dem Beginn der Bewegung aufeinander zu t_3 sollte ein kritischer Wert $t_{critDelay1}$ nicht überschreiten.

Bedingung 4: Verlassen von R_i der Segmentnachbarschaft SN_{B_j} zum Zeitpunkt t_6 über eine minimale Zeitdauer $t_{Separation}$.

Bedingung 5: Die zeitliche Verzögerung t_{delay2} zwischen dem Ende der Bewegung aufeinander zu t_4 und dem Zeitpunkt des Verlassens t_6 sollte ein kritischer Wert $t_{critDelay2}$ nicht überschreiten.

Formale Definition von Konfrontation räumliche Perspektive

Bedingung 1: $R_i \in S_k, S_k \in SN_{B_j}$ und $SN_{B_j} \in NR^2$ zur Zeit t_2

Bedingung 2: $|\varphi_{R_i}(t) - \varphi_{B_j}(t)| \approx 180^\circ, t \in [t_3, t_4]$ und $t_4 - t_3 \geq t_{Intersection}$

$$a) \varphi_{R_i}(t) = \varphi_{S_k}, d(R_i, Sk_{Start}, t) < d(B_j, Sk_{Start}, t), t \in [t_3, t_4], t_4 - t_3 \geq t_{Intersection}, v_{R_i} > 0$$

$$b) \varphi_{B_j}(t) = \varphi_{S_k}, d(R_i, Sk_{Start}, t) > d(B_j, Sk_{Start}, t), t \in [t_3, t_4], t_4 - t_3 \geq t_{Intersection}, v_{B_j} > 0$$

Bedingung 3: $t_4 - t_2 = t_{delay1} < t_{critDelay1}$

Bedingung 4: $R_i \in S_k, B_j \notin S_k, S_k \notin SN_{B_j}$ zur Zeit $t \in [t_6, t_7], t_7 - t_6 \geq t_{Separation}$

Bedingung 5: $t_6 - t_4 = t_{delay2} < t_{critDelay2}$

4.3 Meiden**Raum-zeitliche Beschreibung**

Anhand der Übersicht des Reaktionsbewegungsmusters *Meiden* in Kapitel 2 konnte gezeigt werden, dass es je nach Perspektive, Betrachtungsmassstab und beteiligter Akteure zu unterschiedlichen Ausprägungen von *Meiden* kommen kann. Auf der einen Seite gibt es, wie Odden et al. (2010) anhand von Tigern und Leoparden in Asien aufzeigen konnte, die Möglichkeit, dass sich unterlegene Tiere aus den ursprünglichen Lebensräumen verdrängen lassen. Eine zweite mögliche Ausprägung von *Meiden* konnte durch Cooper et al. (2008) anhand der Interaktion zwischen Vieh und Reh beobachtet werden, wobei sich Rehe zwar im gleichen Lebensraum aufhalten wie Vieh-Tiere, dabei aber möglichst versuchen die räumliche Nähe zu diesen zu meiden. Ab einer gewissen kritischen Distanz zwischen den beiden Tieren, beginnt sich ein Reh von einem Vieh wegzubewegen (Cooper et al. 2008).

4.3.1 Formalisierung individuelle Perspektive

Aus Sicht der individuellen Perspektive von *Meiden* ist primär die letztere der beiden Ausprägung interessant. Durch das zur Verfügung stehende Distanzkriterium besteht eine ideale Möglichkeit die räumliche Nähe zwischen den beteiligten Punktobjekten abzubilden. Dabei wird bei der Unterschreitung eines Distanzkriteriums (abgeleitet aus Sichtlimitierung) als Folge der *Annäherung* zweier sich meidender Punktobjekte das Reaktionsbewegungsmuster *Meiden* ausgelöst. Dieser Sachverhalt ist in der Abbildung 4.11 dargestellt, wobei sich das blaue Punktobjekt unter die Distanz r_{R_i} an das rote annähert.

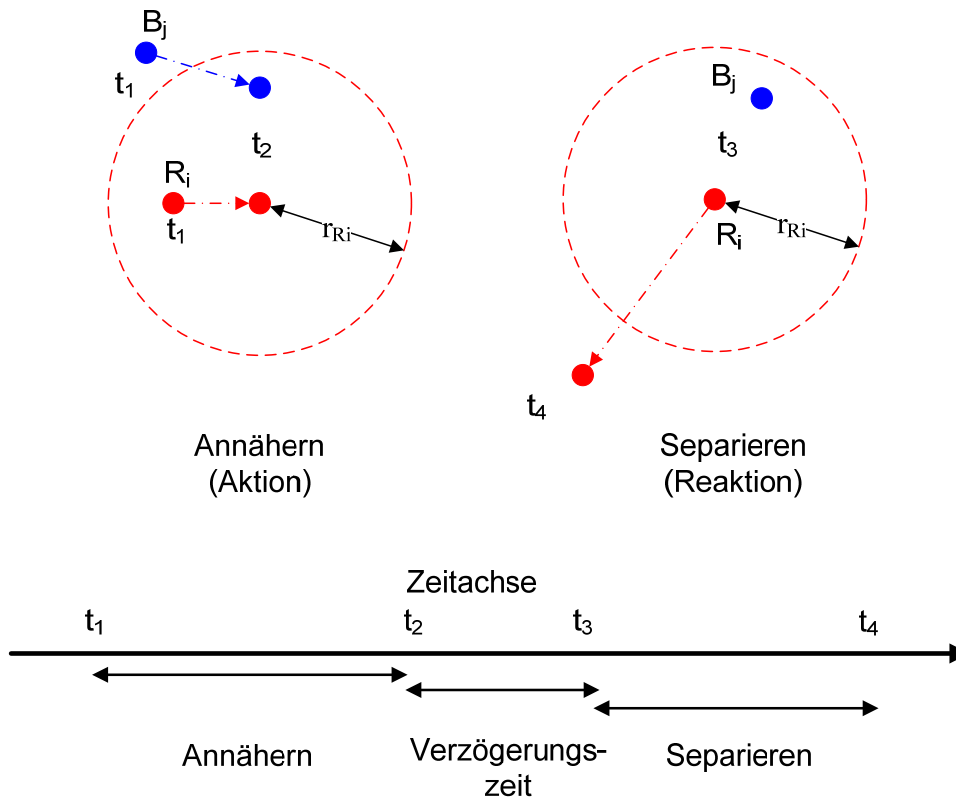


Abbildung 4.13: Raum-zeitlicher Ablauf von Meiden individuelle Perspektive (1)

Aus der individuellen Perspektive wird das Reaktionsbewegungsmuster *Meiden* als Abfolge von *Annäherung* der beiden Punktobjekte unter eine kritische Distanz r_{Ri} (Aktion) und *Separation* auf eine Distanz grösser als r_{Ri} (Reaktion) formalisiert.

Als Folge dieser *Annäherung* unter die kritische Distanz kommt es zu einer Reaktionsbewegung, wobei sich die meidenden Punktobjekte beginnen zu separieren. Je nachdem, ob sich die Punktobjekte gegenseitig meiden oder ob das *Meiden* nur von einem der beiden Objekte ausgeht, ist die raum-zeitliche Ausprägung dieses Reaktionsbewegungsmusters nicht identisch. Falls sich beide Punktobjekte nicht in der Nähe des anderen aufhalten wollen, so bewegen sich beide voneinander weg. Geht das *Meiden* allerdings nur von einem der beiden Punktobjekte aus, so wird die Separationsbewegung nur von diesem ausgeführt, während sich das zweite Objekt unbeeinflusst weiterbewegt.

Für den Beginn der *Separation* konnte ein weiteres raum-zeitlicher Merkmal von *Meiden* ausfindig gemacht werden und zwar ein abrupter Bewegungsrichtungswechsel. Mit einer zeitlichen Verzögerung t_{delay} nachdem sich zwei Punktobjekte unter die kritische Distanz r_{Ri} angenähert haben, ändern die meidenden Objekte plötzlich ihre Bewegungsrichtung und versuchen, sich vom anderen Objekt zu entfernen, was sich in der Separationsbewegung widerspiegelt. Dieser abrupte Bewegungsrichtungswechsel ist in der Abbildung 4.13 dargestellt, wobei der Winkel dieses Richtungswechsels mit α beschrieben wird.

Ein weiteres wichtiges Kriterium des Reaktionsbewegungsmusters *Meiden* aus der individuellen Perspektive betrifft die minimale Distanz von *Meiden*, welche ebenfalls in der Abbildung 4.13 dargestellt ist. Diese Distanz wird bei einem *Meiden* nie unterschritten.

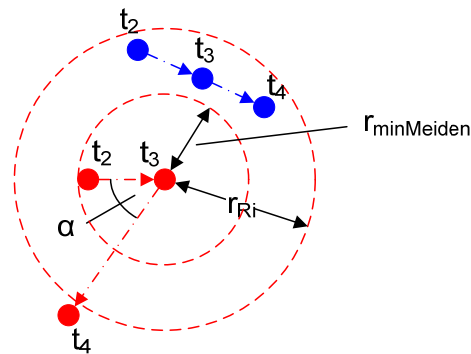


Abbildung 4.14: Raum-zeitlicher Ablauf von Meiden individuelle Perspektive (2)

Nach dem *Annähern* zwischen zwei Punktobjekten kommt es seitens des meidenden roten Objekts zu einem abrupten Wechsel der Bewegungsrichtung und anschliessend zu einer *Separation*. Dabei wird die minimale Distanz von *Meiden* $r_{\min\text{Meiden}}$ nie unterschritten.

Definition von *Meiden* individuelle Perspektive

Eine Anzahl Punktobjekt einer ersten Art (rot) R_n und eine Menge von potentiellen Interaktionspartnern einer zweiten Art (blau) B_m bewegen sich in einem euklidischen Raum R^2 .

Bedingung 1: Verkürzung der Distanz $d(R_i, B_j, t_2)$ zwischen zwei Punktobjekten $R_i, i \in n$ und $B_j, j \in m$ unter die Sichtgrenze r_{R_i} von Punktobjekt R_i .

Bedingung 2: Wechsel der Bewegungsrichtung von $\varphi_{R_i}(x, y, t_2)$ zu $\varphi_{R_i}(x, y, t_3)$ um mindestens φ_{krit} .

Bedingung 3: Keine Verkürzung der $d(R_i, B_j, t)$ unter die minimale Distanz von *Meiden* $r_{\min\text{Meiden}}$ während dem ganzen Reaktionsbewegungsmuster (von t_2 bis t_4).

Bedingung 4: Erhöhung der Distanz $d(R_i, B_j, t_4)$ zwischen den zwei Punktobjekten R_i und B_j über die Sichtgrenze r_{R_i} von Punktobjekt R_i ab dem Zeitpunkt t_4 über eine minimale Zeitdauer $t_{\text{Separation}}$.

Bedingung 5: Die zeitliche Verzögerung t_{delay} zwischen dem Zeitpunkt der Annäherung t_2 und des Verlassens t_4 sollte kleiner sein als der kritische Wert $t_{\text{critDelay}}$.

Formale Definition von *Meiden* individuelle Perspektive

Bedingung 1: $\|R_i(x, y, t_2) - B_j(x, y, t_2)\| = d(R_i, B_j, t_2) < r_{R_i}$

Bedingung 2: $|\varphi_{R_i}(t_2) - \varphi(t_3)| = \varphi_{R_i}(t_2, t_3) > \varphi_{\text{krit}}$

Bedingung 3: $\|R_i(x, y, t) - B_j(x, y, t)\| > r_{\min\text{Meiden}}, t \in [t_2, t_4]$

Bedingung 4: $\|R_i(x, y, t) - B_j(x, y, t)\| = d(R_i, B_j, t) \geq r_{R_i}, t \in [t_4, t_5], t_5 - t_4 \geq t_{\text{Separation}}$

Bedingung 5: $t_4 - t_2 = t_{\text{delay}} < t_{\text{critDelay}}$

4.3.2 Formalisierung räumliche Perspektive

Raum-zeitliche Beschreibung

Um die in Odden et al. (2010) angesprochene Variante von *Meiden*, bei welcher sich in territorialen Konkurrenzsituationen unterlegene Tiere aus ihren ursprünglichen Territorien verdrängen lassen, zu formalisieren, wird auf die räumliche Perspektive zurückgegriffen. Aus Sicht dieser Betrachtungsweise werden Situationen mit *Meiden* durch den Aufenthalt im gleichen Raumkompartiment ausgelöst. Für das Modellieren von *Meiden* zwischen Tieren kann ein Raster oder eine irreguläre Tessellation, welche die Lebensräume widerspiegeln, verwendet werden. Ein solches Habitat ist in der Abbildung 4.14 dargestellt. In diesem Fall kann klar von der Eulerschen Perspektive von *Meiden* gesprochen werden, da der Blick statisch auf ein Raumkompartiment fixiert wird und die sich darin abspielenden Bewegungen analysiert werden. Dabei wird das Reaktionsbewegungsmuster grundsätzlich ausgelöst durch das Eindringen eines Punktobjekts in das Raumkompartiment eines zweiten. Als Folge kommt es seitens des Punktobjekts, welches sich nicht im gleichen Raum mit einem zweiten Objekt aufhalten will, zu einem abrupten Wechsel der Bewegungsrichtung als Folge der *Separation*. Die *Separation* dauert solange an, bis sich die beiden Punktobjekte nicht mehr im gleichen Raumkompartiment aufhalten.

In Netzwerkperspektive tritt anstelle von fixen Raumkompartimenten die Segmentnachbarschaft. Diese ist variabel, da sie sich während der Bewegung der Punktobjekte durch das Netzwerk ständig ändert, was zwar den Grundsätzen der Eulerschen Perspektive widerspricht, aber eine erfolgreiche Formalisierung verspricht.

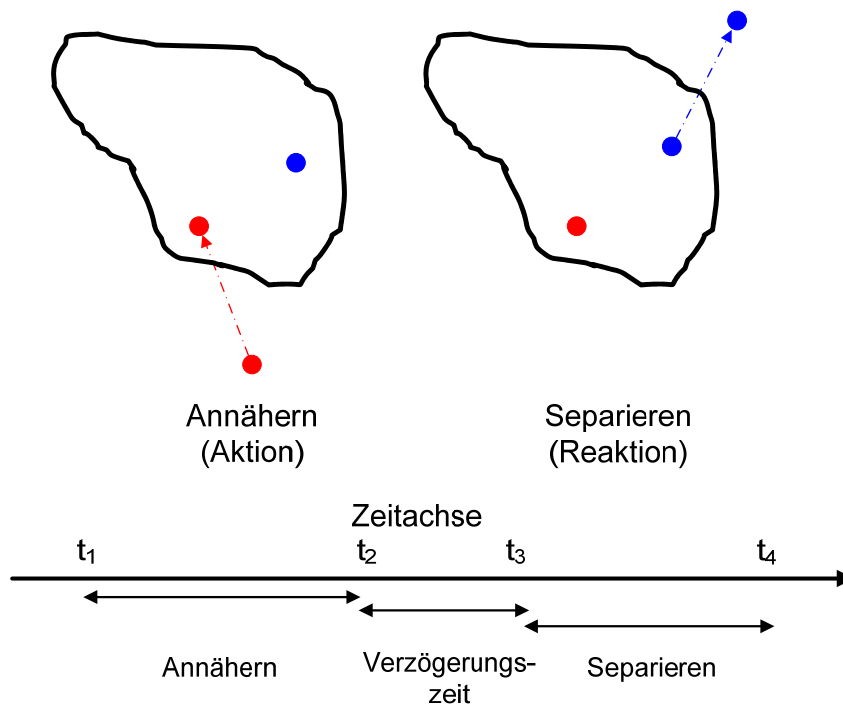


Abbildung 4.15: Raum-zeitlicher Ablauf von *Meiden* räumliche Perspektive

Nach dem *Annähern* zwischen zwei Punktobjekten kommt es seitens des meidenden roten Objekts zu einem abrupten Wechsel der Bewegungsrichtung und anschliessend zu einer *Separation*.

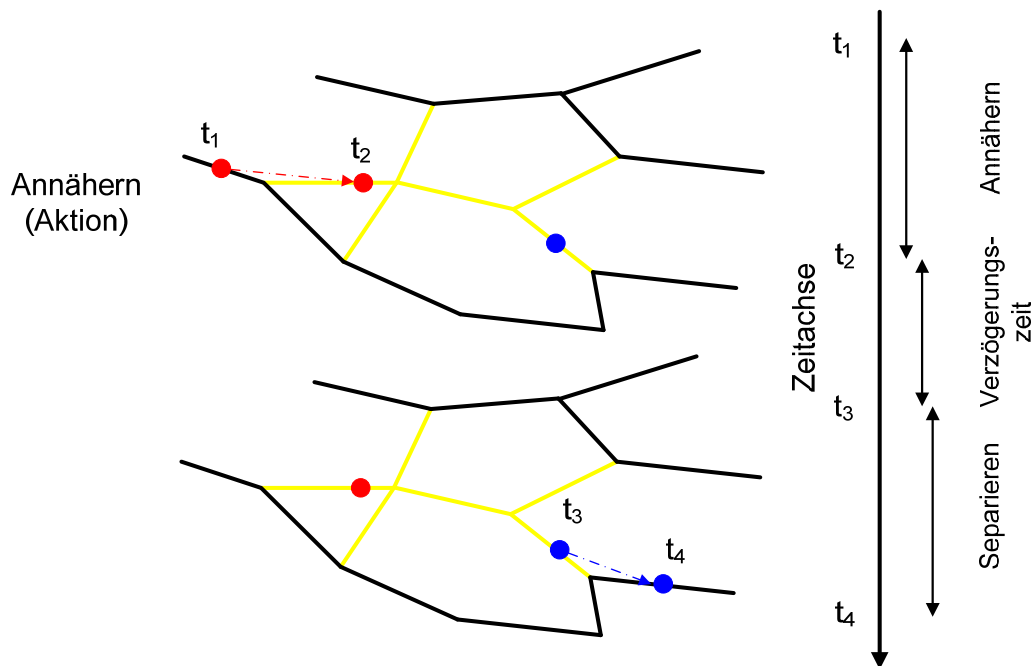


Abbildung 4.16: Raum-zeitlicher Ablauf von Meiden Netzwerk-Perspektive

Nach dem *Annähern* zwischen zwei Punktobjekten kommt es seitens des meidenden roten Objekts zu einem abrupten Wechsel der Bewegungsrichtung und anschliessend zu einer *Separation*.

Definition von *Konfrontation* räumliche Perspektive

Eine Anzahl Punktobjekte einer ersten Art (rot) R_n und eine Menge von Punktobjekten einer zweiten Art (blau) B_m bewegen sich in einem Netzwerkraum NR^2 mit einer Menge von Segmenten S_l .

Bedingung 1: Aufenthalt eines Punktobjekts R_i , $i \in n$ auf einem Segment S_k , $k \in l$, welches sich zum Zeitpunkt t_2 in der Segmentnachbarschaft $SN_{B_j} \in NR^2$ eines zweiten Punktobjekts B_j , $j \in m$ befindet.

Bedingung 2: Wechsel der Bewegungsrichtung von $\varphi_{R_i}(x, y, t_2)$ zu $\varphi_{R_i}(x, y, t_3)$ um mindestens φ_{krit} .

Bedingung 3: Verlassen von R_i der Segmentnachbarschaft SN_{B_j} zum Zeitpunkt t_4 .

Bedingung 4: Die zeitliche Verzögerung t_{delay} zwischen dem Startpunkt des Aufenthalts in der gleichen Segmentnachbarschaft t_2 und dem Verlassen t_4 sollte kleiner sein als der kritischer Wert $t_{critDelay}$.

Formale Definition von *Konfrontation* räumliche Perspektive

Bedingung 1: $R_i \in S_k, S_k \in SN_{B_j}$ und $SN_{B_j} \in NR^2$ zur Zeit t_2

Bedingung 2: $|\varphi_{R_i}(t_2) - \varphi_{R_i}(t_3)| = \varphi_{R_i}(t_2, t_3) > \varphi_{krit}$

Bedingung 3: $R_i \in S_k, B_j \notin S_k, S_k \notin SN_{B_j}$ zur Zeit $t \in [t_4, t_5]$ und $t_5 - t_4 \geq t_{separation}$

Bedingung 4: $t_4 - t_2 = t_{delay} < t_{critDelay}$

5 Entwicklung von Algorithmen

In diesem Kapitel werden die Algorithmen zur Detektion der drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* aufbauend auf den Formalisierungen von Kapitel 4 beschrieben. Die Ausführungen zur Formalisierung haben gezeigt, dass Reaktionsbewegungsmuster aus Abfolgen von einzelnen Bewegungsmustern bestehen. Zwischen diesen Mustern gibt es jeweils eine zeitliche Verzögerung. Die Algorithmen basieren somit auf der Detektion von einzelnen Bewegungsmustern und der Prüfung von zusätzlichen Kriterien (z.B. Verzögerungszeit). Im Falle von *Verfolgen/Entfliehen* beispielsweise werden zuerst die beiden Muster *Zusammensein* und *Folgen* gesucht und anschliessend die zeitliche Verzögerung zwischen diesen beiden untersucht. Die in den drei Reaktionsbewegungsmustern enthaltenen Bewegungsmuster sowie die zusätzlichen Kriterien, welche es zu prüfen gilt, sind in Abbildung 5.1 dargestellt.

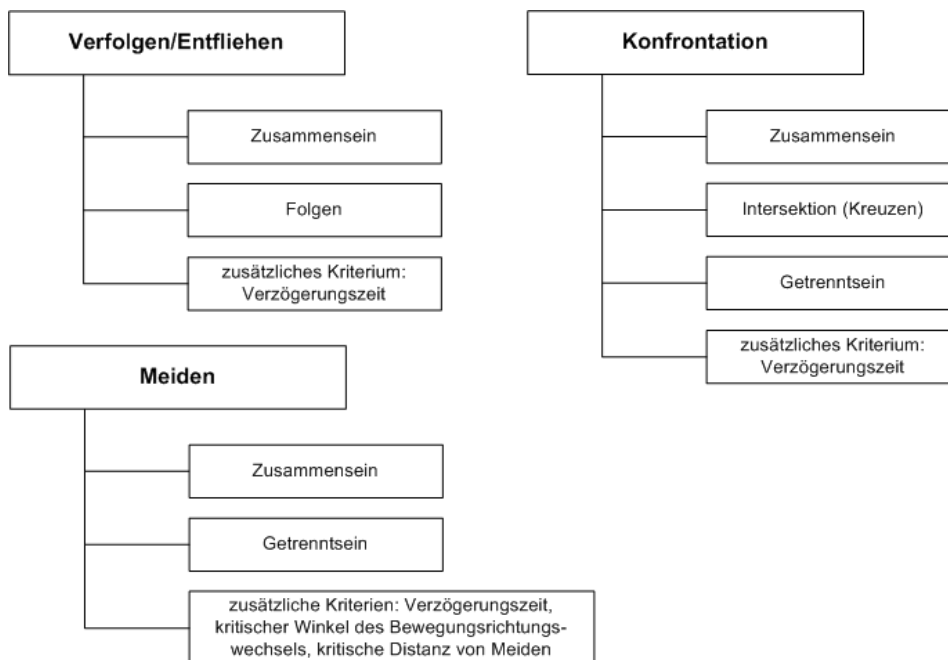


Abbildung 5.1: Algorithmen zur Detektion von Reaktionsbewegungsmustern sowie ihre wichtigsten Elemente

Die Algorithmen sind jeweils zusammengesetzt aus der Detektion von einzelnen Bewegungsmustern sowie der Überprüfung von zusätzlichen Kriterien wie beispielsweise der Verzögerungszeit.

Die Algorithmen wurden in der Programmiersprache Java entwickelt. Der gesamte Java-Code der Klasse „MovementPatternDetector“, in welcher alle Algorithmen zur Detektion der Reaktionsbewegungsmuster enthalten sind, ist im Anhang dieser Arbeit zu finden. In den Skizzen zu den einzelnen Algorithmen sind jeweils in roter Farbe die Namen der verwendeten Methoden aufgeführt. Um sich ein genaueres Bild der Funktionsweise dieser Methoden zu machen, können diese im Java-Code im Anhang betrachtet werden.

Die Algorithmen zur Detektion der drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* basieren alle auf paarweisen Vergleichen zwischen den n betrachteten Punktobjekten über alle Zeitschritte t . Als Beispiel kann die Methode „observeDistanceRelationships“

betrachtet werden. Mit Hilfe dieser Methode wird die Distanz zwischen allen n Punktobjekten über alle Zeitschritte t berechnet (Komplexität von $O(n^2t)$). Diese Methode wird beispielsweise bei der Detektion des Musters *Zusammensein* aus der individuellen Perspektive verwendet. Im Falle der räumlichen Perspektive wird mit der Methode „checkMovementsOnNeighbouringSegments“ durch einen paarweisen Vergleich über alle Zeitschritte geprüft, ob sich zwei Punktobjekte zu einem Zeitpunkt in derselben Segmentnachbarschaft aufhalten (Komplexität von $O(n^2t)$). Anstelle der Distanzbeziehung tritt also die Untersuchung des Aufenthalts in derselben Nachbarschaft. Das Bewegungsmuster *Zusammensein* ist ein Bestandteil aller drei Reaktionsbewegungsmuster. Da die Laufzeitkomplexität eines Algorithmus stets bestimmt wird von der Operation mit der grössten Anzahl Durchläufe (Saake und Sattler 2010), beträgt die Komplexität bei der Detektion der drei Reaktionsbewegungsmuster $O(n^2t)$.

In den nachfolgenden Abschnitten werden zuerst die Abläufe bei der Detektion der Bewegungsmuster *Zusammensein*, *Getrenntsein*, *Konvergenz*, *Divergenz*, *Folgen* und *Hineinrennen* skizziert. Anschliessend werden diese Muster zusammengesetzt zu den drei Reaktionsbewegungsmustern *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* diskutiert. In der Tabelle 5.1 aufgelistet sind die Kürzel der Reaktionsbewegungsmuster-Algorithmen. In diesem Kapitel sowie in den nachfolgenden werden jeweils diese Abkürzungen verwendet.

Abkürzung	Beschreibung des Algorithmus
VED _i	Algorithmus zur Detektion von <i>Verfolgen/Entfliehen</i> (individuellen Perspektive)
VED _r	Algorithmus zur Detektion von <i>Verfolgen/Entfliehen</i> (räumlichen Perspektive)
MD _i	Algorithmus zur Detektion von <i>Meiden</i> (individuelle Perspektive)
MD _r	Algorithmus zur Detektion von <i>Meiden</i> (räumliche Perspektive)
KD _i	Algorithmus zur Detektion von <i>Konfrontationen</i> (individuelle Perspektive)
KD _r	Algorithmus zur Detektion von <i>Konfrontationen</i> (räumliche Perspektive)

Tabelle 5.1: Übersicht der in dieser Arbeit verwendeten Kürzel der Algorithmen

Auflistung der Abkürzungen der Algorithmen zur Detektion von Reaktionsbewegungsmustern. Die individuelle Perspektive wird gekennzeichnet durch den Buchstaben i, die räumliche Perspektive durch den Buchstaben r.

5.1 Zusammensein / Getrenntsein

Die beiden Bewegungsmuster *Zusammensein* und *Getrenntsein* umschreiben den Zustand der Distanzbeziehungsweise Nachbarschaftsbeziehung zwischen zwei Punktobjekten. Im Falle von *Zusammensein* haben sich zwei Punktobjekte gegenseitig angenähert und verbleiben über eine gewisse Zeitspanne in nächster Nähe zueinander. Im Gegensatz dazu widerspiegelt das *Getrenntsein* das Resultat einer Bewegung voneinander weg (*Separation*). Als Folge dieser *Divergenz* sind die Punktobjekte nicht mehr zusammen und demzufolge räumlich getrennt. Die beiden Prozesse der *Konvergenz* (*Annäherung*) und *Divergenz* (*Separation*) werden in Kapitel 5.2 beschrieben. Aus den formalen Definitionen in Kapitel 4 geht hervor, dass primär die aus den beiden Prozessen *Konvergenz*

und *Divergenz* resultierenden Zustände *Zusammensein* und *Getrenntsein* relevant sind und weniger die Prozesse. Dies liegt grundsätzlich daran, dass es nicht so wichtig ist, wie sich zwei Punktobjekte angenähert haben. Dafür vielmehr dass sie sich unter einer kritischen Distanz angenähert haben und als Folge davon zusammen sind. Das gleiche gilt für das *Getrenntsein* und die *Separation*.

5.1.1 Zusammensein individuelle Perspektive

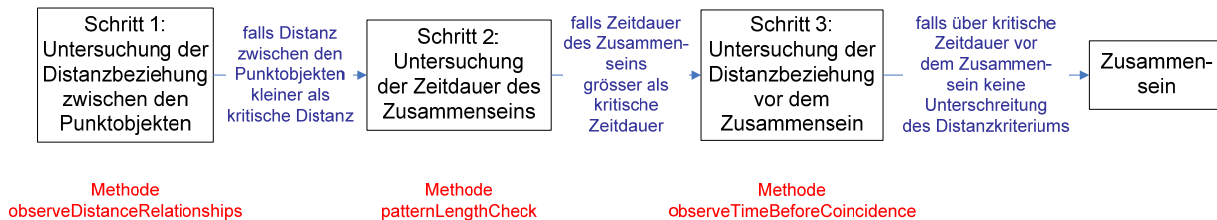


Abbildung 5.2: Detektion von *Zusammensein* (individuelle Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Zusammensein* mit der individuellen Perspektive. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Auflistung der Parameter: kritische Distanz *Zusammensein*, kritische Zeitdauer *Zusammensein*, Toleranz *Zusammensein*, kritische Zeitdauer ohne *Zusammensein*

Die genauen Schritte bei der Detektion von *Zusammensein* sind in der Abbildung 5.2 dargestellt. Aus individueller Perspektive gesehen, bedeutet das Bewegungsmuster *Zusammensein*, dass sich zwei Punktobjekte unter eine vordefinierte Distanz (Parameter kritische Distanz *Zusammensein*) angenähert haben und anschliessend über eine bestimmte Zeitdauer (Parameter kritische Zeitdauer *Zusammensein*) zusammenbleiben. Daher darf die Distanz zwischen den Punktobjekten während dieser kritischen Zeitdauer die kritische Distanz nicht überschreiten. Für die Untersuchung, ob die Distanz zwischen zwei Punktobjekten die kritische Distanz unterschreitet, wurde die Methode „observeDistanceRelationships“ entwickelt. Mit Hilfe dieser Methode wird die Distanz zwischen den Punktobjekten berechnet und mit dem Wert des Distanzparameters verglichen. Mit der Methode „patternLengthCheck“ wird in einem zweiten Schritt geprüft, ob die Einhaltung des Distanzparameters über eine genügend lange Zeitperiode aufrecht erhalten wird. In realen Bewegungsdatensätzen mit stark fehlerhaften Positionsangaben (z.B. Frequentie1550-Daten) ist es denkbar, dass sich Punktobjekte aufgrund von Sprüngen in den Daten für kurze Zeit wieder über die kritische Distanz entfernen. Um solche Unterbrüche in den Mustern zuzulassen, wurde ein Toleranzparameter (Parameter Toleranz *Zusammensein*) eingeführt. Falls also das Distanzkriterium über eine genügend grosse Zeitspanne (mit einer Anzahl von Unterbrüchen welche kleiner ist als der Toleranzparameter) eingehalten ist, kann von einem *Zusammensein* zwischen den Punktobjekten gesprochen werden. In gewissen Reaktionsbewegungsmustern kann es eine wichtige Voraussetzung sein, dass sich die Punktobjekte während einer kritischen Zeitdauer vor dem *Zusammentreffen* nicht bereits einmal getroffen haben. Dies ist zum Beispiel im Falle des Reaktionsbewegungsmusters *Meiden* von Bedeutung. Mit Hilfe der Methode „observeTimeBeforeCoincidence“ kann geprüft werden, dass sich zwei Punktobjekte über eine Zeitspanne vor dem *Zusammensein* (Parameter kritische Zeitdauer ohne

Zusammensein) nicht bereits einmal unter die kritische Distanz angenähert haben. Für die beiden Reaktionsbewegungsmuster *Verfolgen/Entfliehen* und *Konfrontation* spielt dieses Kriterium keine Rolle, weshalb auf die Verwendung dieses Parameters verzichtet wurde.

5.1.2 Zusammensein räumliche Perspektive

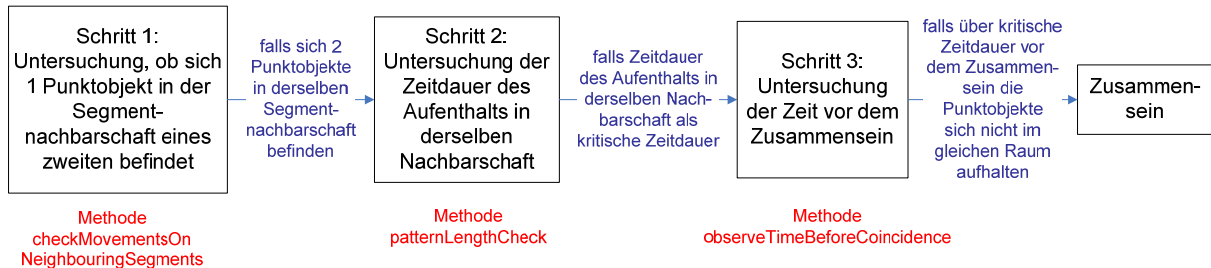


Abbildung 5.3: Detektion von *Zusammensein* (räumliche Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Zusammensein* mit der räumlichen Perspektive. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Auflistung der Parameter: Nachbarschaftsgrösse *Zusammensein*, kritische Zeitdauer *Zusammensein*, Toleranz *Zusammensein*, kritische Zeitdauer ohne *Zusammensein*

Der Ablauf bei der Detektion von *Zusammensein* mit der räumlichen Perspektive ist in der Abbildung 5.3 dargestellt. Dabei tritt an Stelle der Distanzbeziehung zwischen den Punktobjekten (individuellen Perspektive) der Aufenthalt im gleichen Raum. Im Falle eines Strassennetzwerks bedeutet dies das *Zusammensein* in der gleichen Segmentnachbarschaft. Die Grösse dieser Segmentnachbarschaft wird beschrieben durch den Parameter Nachbarschaftsgrösse *Zusammensein*. Während sich die Punktobjekte bei einer Grösse von 0 auf dem gleichen Segment bewegen müssen, ist im Falle einer Grösse von 1 der Aufenthalt auf demselben oder auf benachbarten Segmenten gefordert. Für die Nachbarschaftsgrösse 2 sind auch Bewegungen auf benachbarten der benachbarten Segmente erlaubt. Um zu bestimmen, ob sich zwei Punktobjekte in derselben Segmentnachbarschaft aufhalten, wurde die Methode „checkMovementsOnNeighbouringSegments“ entwickelt. Durch paarweise Vergleiche zwischen allen Punktobjekten werden diejenigen Zeitschritte gesucht, während denen sich zwei Punktobjekte in der gleichen Segmentnachbarschaft aufhalten. Nach der Bestimmung dieser Zeitpunkte folgt wie im Falle der individuellen Perspektive die Prüfung der Zeitdauer des Musters sowie die Untersuchung der Zeit vor dem *Zusammensein*. Wird das Bewegungsmuster *Zusammensein* über eine genügend lange Zeitperiode aufrechterhalten und erfolgte über eine kritische Zeitdauer vor dem Start keine *Annäherung* in die gleiche Segmentnachbarschaft, so kann von einem *Zusammensein* gesprochen werden.

5.1.3 Getrenntsein individuelle Perspektive

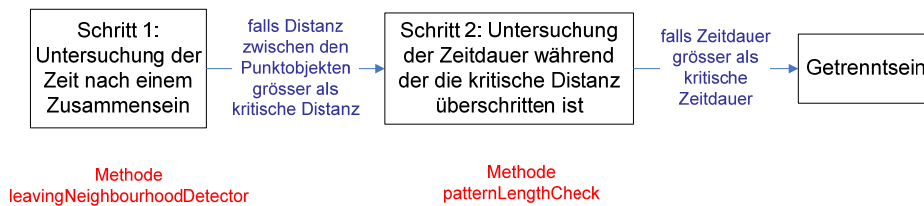


Abbildung 5.4: Detektion von *Getrenntsein* (individuelle Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Getrenntsein* mit der individuellen Perspektive. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Auflistung der Parameter: kritische Distanz *Getrenntsein*, kritische Zeitdauer *Getrenntsein*, Toleranz *Getrenntsein*

Aus Sicht der individuellen Perspektive bedeutet das Bewegungsmuster *Getrenntsein*, dass sich die Punktobjekte auf eine vordefinierte Distanz (Parameter kritische Distanz *Getrenntsein*) voneinander wegbewegt haben und sich über eine gewisse Zeitdauer (Parameter kritische Zeitdauer *Getrenntsein*) nicht mehr unter diese kritische Distanz annähern. Mit Hilfe der Methode „leavingNeighbourhoodDetector“ werden für alle Punktobjektpaare die Situationen nach einem *Zusammensein* ausfindig gemacht und untersucht, ob die Distanz zwischen den Punktobjekten grösser als die kritische Distanz ist. Anschliessend wird geprüft, wie lange sich die beiden Punktobjekte nach einem *Zusammensein* nicht mehr unter die kritische Zeitdauer annähern (Methode „patternLengthCheck“). Ist diese Zeitdauer genügend lange, so wird die Situation als Bewegungsmuster *Getrenntsein* deklariert. Siehe dazu auch die Abbildung 5.4. Für den Fall von fehlerhaften und sprunghaften Bewegungsdaten steht zudem ein Toleranzparameter zur Verfügung.

5.1.4 Getrenntsein räumliche Perspektive

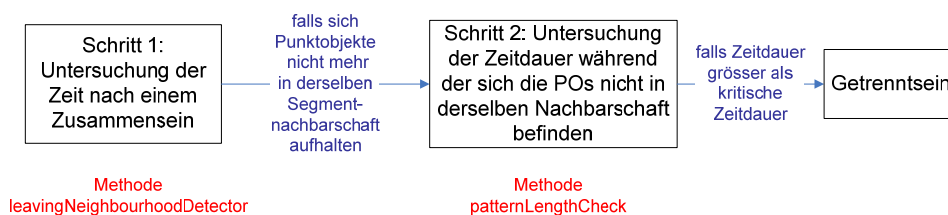


Abbildung 5.5: Detektion von *Getrenntsein* (räumliche Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Getrenntsein* mit der räumlichen Perspektive. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Auflistung der Parameter: Nachbarschaftsgrösse *Getrenntsein*, kritische Zeitdauer *Getrenntsein*, Toleranz *Getrenntsein*

Im Falle der räumlichen Perspektive tritt an Stelle der Distanzbeziehung zwischen den Punktobjekten das Strassennetzwerk, wie in der Abbildung 5.5 skizziert. Um von einem Bewegungsmuster *Getrenntsein* zu sprechen, müssen die Punktobjekte den Aufenthalt in der gleichen Segmentnachbarschaft beenden und dürfen diese für eine kritische Zeitspanne (Parameter kritische Zeitdauer *Getrenntsein*) nicht mehr betreten.

5.2 Konvergenz / Divergenz

Währenddem die beiden Bewegungsmuster *Zusammensein* und *Getrenntsein* den Zustand der räumlichen Distanz- bzw. Nachbarschaftsbeziehung zwischen Punktobjekten widerspiegeln, beschreiben die Muster *Konvergenz* (*Annäherung*) und *Divergenz* (*Separation*) die Prozesse, welche zu einem *Zusammensein* bzw. *Getrenntsein* führen. Aus der individuellen Perspektive wird zuerst geprüft, ob zwei Punktobjekte genügend Nahe beieinander sind. Ist dies der Fall, so werden Trends in den Distanzbeziehungen gesucht. Gibt es über eine genügend lange Zeitdauer eine stetige Verkürzung der Distanz zwischen den Punktobjekten, so wird die Situation als *Konvergenz* gewertet. Ist aber eine kontinuierliche Erhöhung der Distanzen zu beobachten, findet eine *Divergenz* statt. Siehe dazu die Abbildung 5.6.

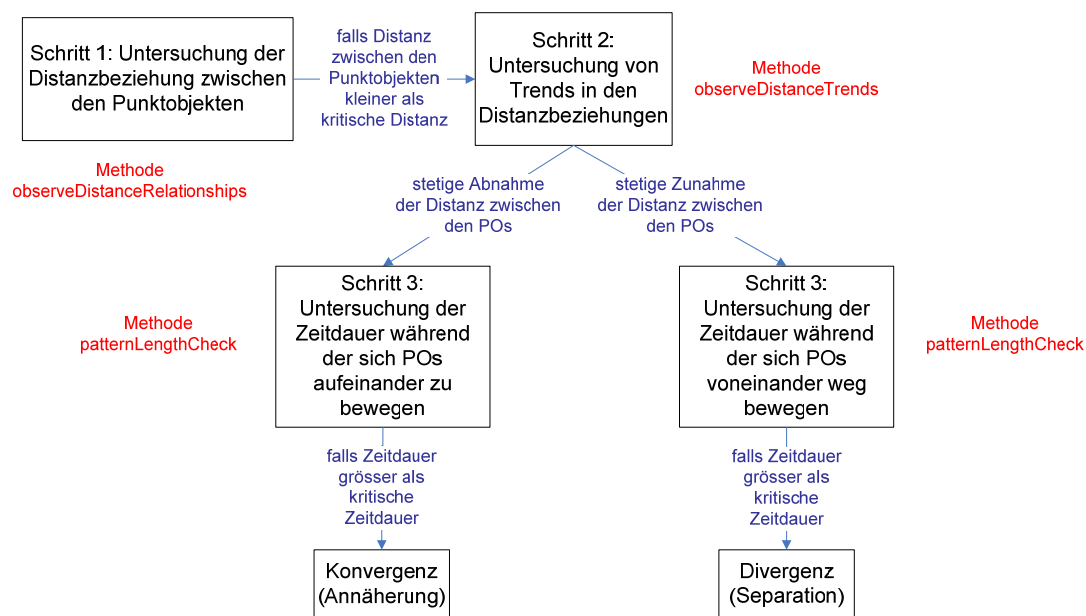


Abbildung 5.6: Detektion von *Konvergenz/Divergenz* (individuelle Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Konvergenz/Divergenz* mit der individuellen Perspektive. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Im Falle der räumlichen Perspektive wird zuerst untersucht, ob sich ein Punktobjekt in der Nachbarschaft eines zweiten aufhält. Anschliessend werden die Bewegungsrichtungen dieser Punktobjekte auf den Strassensegmenten analysiert und Bewegungen aufeinander zu sowie voneinander weg gesucht. Erstrecken sich diese Bewegungen über eine genügend lange Zeitdauer, so spricht man von *Konvergenz* beziehungsweise *Divergenz* (Siehe Abbildung 5.7).

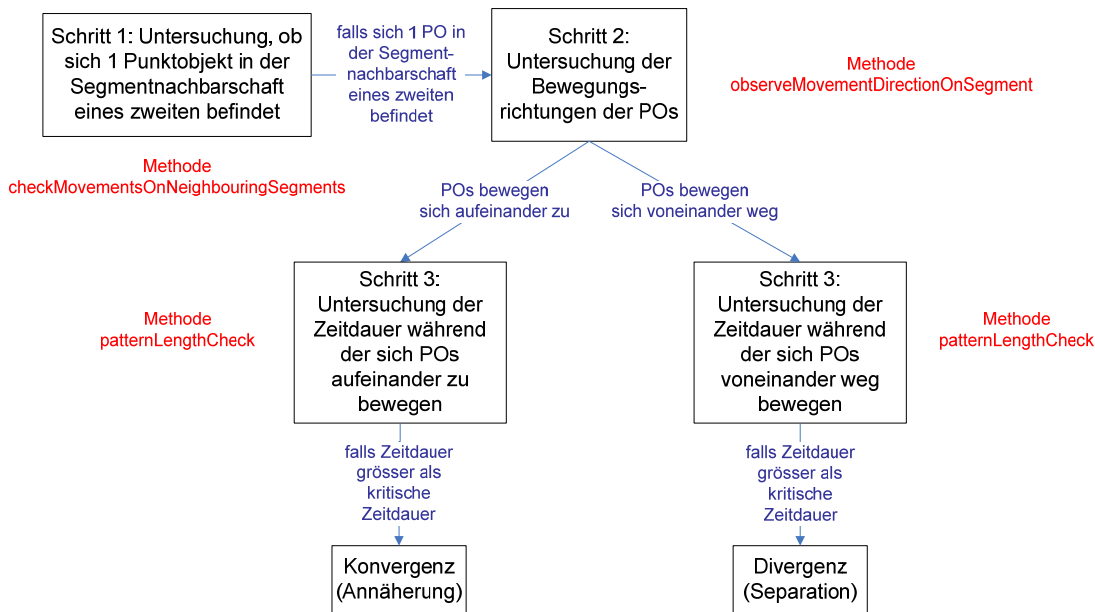


Abbildung 5.7: Detektion von Konvergenz/Divergenz (räumliche Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Konvergenz/Divergenz* mit der räumlichen Perspektive. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Diese Methoden wurden entwickelt und für die Detektion von Reaktionsbewegungsmustern getestet. Allerdings stellte sich heraus, dass bei der Entdeckung von Reaktionsbewegungsmustern weniger die Prozesse *Konvergenz/Divergenz* sondern vielmehr die daraus resultierenden Zustände *Zusammensein/Getrenntsein* entscheidend sind. Es ist nicht relevant, ob sich zwei Punktobjekte direkt und stetig oder in mehreren Etappen angenähert haben. Viel entscheidender ist die Tatsache, dass sie sich angenähert haben und als Folge davon über eine gewisse Zeitdauer in nächster Nähe zusammen bleiben. Aus diesem Grund wurden die beiden Muster *Konvergenz* und *Divergenz* nicht verwendet, weshalb auf eine Auflistung der Parameter verzichtet wird. Die einzige Ausnahme bildet *Konvergenz* im Falle des *Hineinrennens* zwischen Punktobjekten. Beim *Hineinrennen* aus der räumlichen Perspektive ist es eine wichtige Voraussetzung, dass sich zwei Punktobjekte auf dem gleichen Segment kontinuierlich annähern und sich schlussendlich treffen.

5.3 Folgen

5.3.1 Individuelle Perspektive

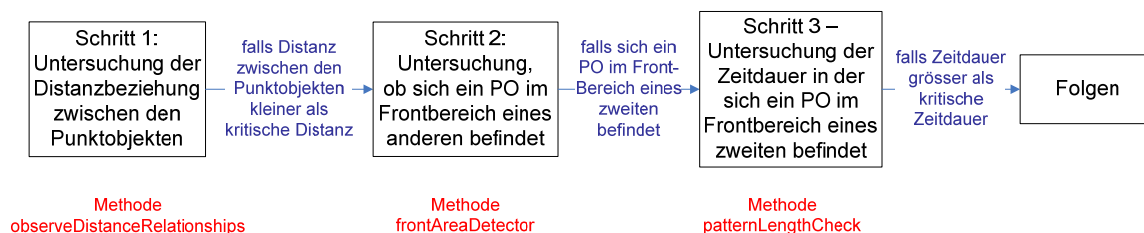


Abbildung 5.8: Detektion von Folgen (individuelle Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Folgen* mit der individuellen Perspektive. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Auflistung der Parameter: kritische Distanz *Folgen*, kritische Zeitdauer *Folgen*, kritischer Frontbereichswinkel, Toleranz *Folgen*

Um sicherzustellen, dass sich das Fluchtobjekt und sein Verfolger sehen können, wird mit Hilfe der Methode „observeDistanceRelationships“ die Distanzbeziehung zwischen den Punktobjekten untersucht. Ist die Distanz zwischen den Punktobjekten kleiner als der Parameter kritische Distanz von *Folgen*, wird mit der Methode „frontAreaDetector“ geprüft, ob sich eines der beiden Punktobjekte im Frontbereich des anderen befindet. Dabei ist die Grösse des Frontbereichs neben der kritischen Distanz von *Folgen* auch abhängig vom Parameter kritischer Frontbereichswinkel. Für die Überprüfung der Zeitdauer des *Folgens* wird die Methode „patternLengthCheck“ verwendet. Dabei gelten nur Situationen als *Folgen*, falls die oben beschriebenen Kriterien über eine vordefinierte Zeitspanne (Parameter kritische Zeitdauer von *Folgen*) aufrechterhalten werden. Aufgrund der Tatsache, dass der Verfolger für kurze Zeit sein Zielobjekt aus den Augen verlieren kann, wird ein Toleranzparameter eingeführt.

5.3.2 Räumliche Perspektive

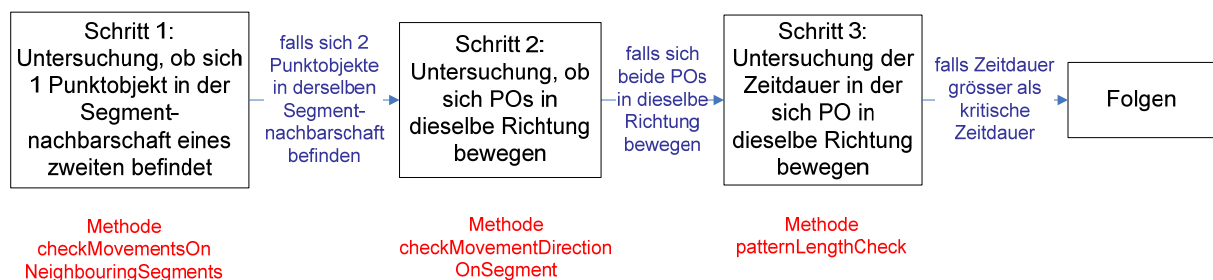


Abbildung 5.9: Detektion von *Folgen* (räumliche Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Folgen* mit der räumlichen Perspektive. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Auflistung der Parameter: Nachbarschaftsgrösse *Folgen*, kritische Zeitdauer *Folgen*, Toleranz *Folgen*

Im Vergleich zur individuellen Perspektive wird aus der räumlichen Perspektive anstelle der Distanzbeziehung der Aufenthalt in derselben Segmentnachbarschaft geprüft. Dies geschieht mit Hilfe der Methode „checkMovementsOnNeighbouringSegments“, wobei die Grösse dieser Nachbarschaft durch den Parameter Nachbarschaftsgrösse von *Folgen* definiert wird. Weiter wird untersucht, ob sich die Punktobjekte während dem Aufenthalt in derselben Nachbarschaft auf den Strassensegmente in die gleiche Richtung bewegen und ob sie dies über eine genügend lange Zeitdauer tun.

5.4 Intersektion (Kreuzen)

Das Bewegungsmuster *Intersektion* ist ein Bestandteil einer *Konfrontation* und beschreibt das Kreuzen der Bewegungspfade. Im Falle des Frequentie1550-Projekts wird eine *Intersektion* ausgelöst durch das gegenseitige *Hineinrennen* zwischen Spielern von unterschiedlichen Teams.

5.4.1 Individuelle Perspektive

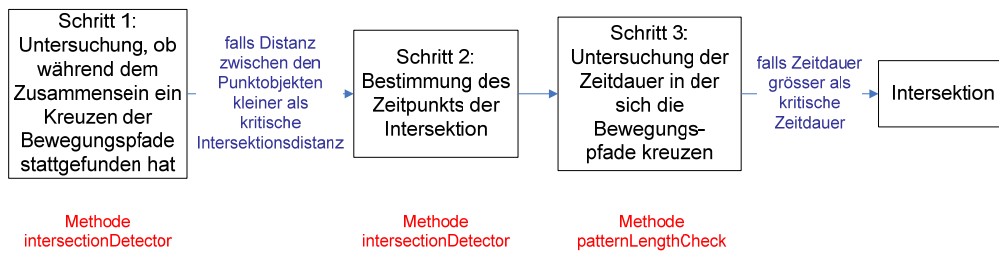


Abbildung 5.10: Detektion einer *Intersektion* (individuelle Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Intersektionen* mit der individuellen Perspektive. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Aus der individuellen Perspektive wurde das Kreuzen von Bewegungspfaden als Unterschreitung der Distanz zwischen zwei Punktobjekten unter den Wert des Parameters kritische Intersektionsdistanz formalisiert. Ob dies der Fall ist, wird mit Hilfe der Methode „intersectionDetector“ untersucht. Weiter wird der Zeitpunkt, bei dem die Punktobjekte sich am nächsten sind, bestimmt und dieser als Zeitpunkt der *Intersektion* deklariert. Die Überprüfung der Zeitdauer der *Intersektion* wird mit der Methode „patternLengthCheck“ vorgenommen.

Dieser Teil der *Konfrontation* kann im Falle von kämpferischem Verhalten zwischen den Punktobjekten noch beliebig erweitert werden. Für den Anwendungsfall der Frequentie1550-Daten war die Detektion einer *Intersektion* ausreichend, da die Punktobjekte lediglich ineinander hineinrennen mussten um eine *Konfrontation* auszulösen. Der Sieger der *Konfrontation* wurde anhand des Ordens der Teams im Spiel bestimmt.

Auflistung der Parameter: kritische Distanz *Intersektion*, kritische Zeitdauer *Intersektion*

5.4.2 Räumliche Perspektive

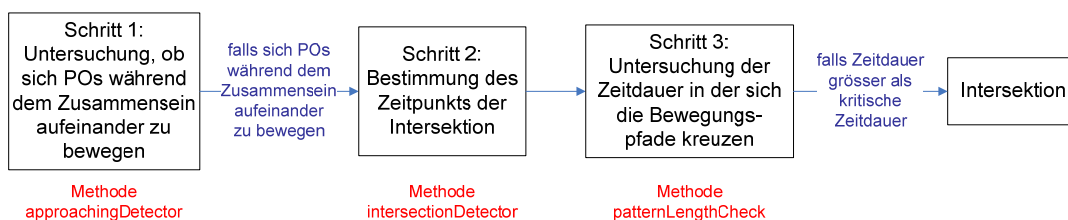


Abbildung 5.11: Detektion einer *Intersektion* (räumliche Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Intersektionen* mit der räumlichen Perspektive. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Auflistung der Parameter: kritische Zeitdauer *Intersektion*, Toleranz *Intersektion*

Aufgrund des Fehlens eines Distanzkriteriums wurde eine *Intersektion* im Falle der räumlichen Perspektive als Annäherung zwischen Punktobjekten während dem *Zusammensein* formalisiert. Die Bewegung aufeinander zu kann mit Hilfe der Methode „approachingDetector“ detektiert werden. Durch die Methode „intersectionDetector“ kann der Zeitpunkt der *Intersektion* bestimmt und anhand der Methode „patternLengthCheck“ die zeitliche Dauer überprüft werden. Der ganze Ablauf der Detektion mit allen Schritten ist in der Abbildung 5.11 dargestellt.

5.5 Verfolgen/Entfliehen

5.5.1 Individuelle Perspektive

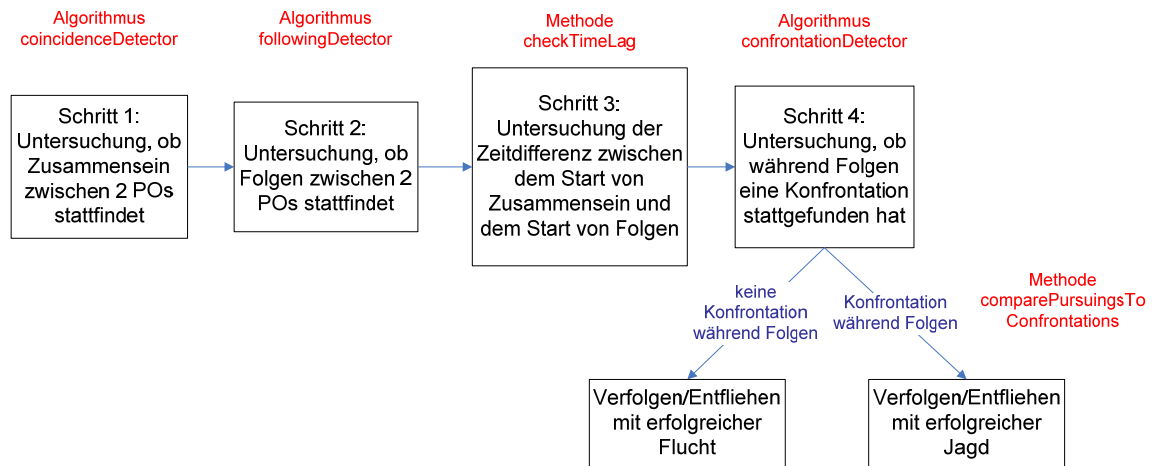


Abbildung 5.12: Detektion von *Verfolgen/Entfliehen* (individuelle und räumliche Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Verfolgen/Entfliehen*. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Der Algorithmus VED_i setzt sich zusammen aus der Detektion des Bewegungsmusters *Zusammensein* und der Entdeckung von *Folgen*. Zudem wird geprüft, ob die kritische Zeitverzögerung eingehalten werden kann. Um eine Aussage über das Ende der Verfolgungsjagd machen zu können, wird zudem noch der Algorithmus KD_i eingesetzt. Kann mit diesem Algorithmus eine *Konfrontation* detektiert werden, so wird die Situation als *Verfolgen/Entfliehen* mit erfolgreicher Jagd interpretiert. Ist allerdings keine *Konfrontation* auszumachen während der *Verfolgung*, so wird das Ende von *Verfolgen/Entfliehen* als erfolgreiche Flucht gewertet. Der ganze Aufbau des Algorithmus VED_i ist in der Abbildung 5.12 skizziert.

Auflistung der Parameter: kritische Distanz *Zusammensein*, kritische Zeitdauer *Zusammensein*, Toleranz *Zusammensein*, kritische Distanz *Folgen*, kritische Zeitdauer *Folgen*, kritischer Frontbereichswinkel, Toleranz *Folgen*, kritische Verzögerungszeit zwischen Beginn des *Zusammenseins* und Beginn von *Folgen*

5.5.2 Räumliche Perspektive

Im Vergleich zur individuellen Perspektive erfolgt die Detektion von *Verfolgen/Entfliehen* mit dem Algorithmus VED_r ebenfalls gemäss dem Ablauf in der Abbildung 5.12. Es werden lediglich anstelle der Methoden zur Detektion von *Zusammensein* und *Folgen* der individuellen Perspektive diejenigen für die räumliche Perspektive verwendet. Daraus ergeben sich auch leicht veränderte Parameter.

Auflistung der Parameter: Nachbarschaftsgrösse *Zusammensein*, kritische Zeitdauer *Zusammensein*, Toleranz *Zusammensein*, Nachbarschaftsgrösse *Folgen*, kritische Zeitdauer *Folgen*, Toleranz *Folgen*, kritische Verzögerungszeit zwischen Beginn des *Zusammenseins* und Beginn von *Folgen*

5.6 Konfrontation

5.6.1 Individuelle Perspektive

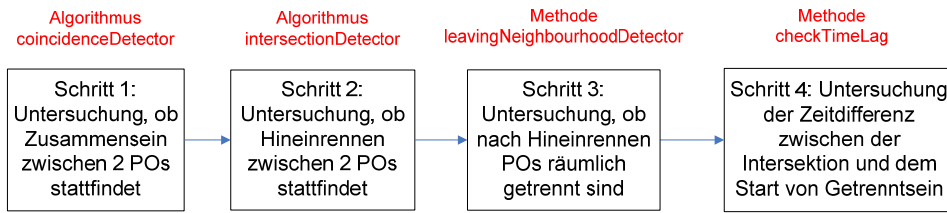


Abbildung 5.13: Detektion von *Konfrontation* (individuelle und räumliche Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Konfrontation*. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Der Algorithmus KD_i setzt sich zusammen aus der Detektion des Bewegungsmusters *Zusammensein* sowie der Entdeckung von *Intersektion* und *Getrenntsein*. Zudem wird geprüft, ob die kritische Zeitverzögerung zwischen dem Zeitpunkt der *Intersektion* und dem Startpunkt des *Getrenntseins* eingehalten werden kann. Diese Schritte sind in der Abbildung 5.13 skizziert. Auf eine Überprüfung der zeitlichen Verzögerung zwischen dem Beginn des *Zusammenseins* und der *Intersektion* wird verzichtet, da dieser Parameter stark unterschiedliche Werte annehmen kann. Je nach dem, ob sich die Punktobjekte vor der *Intersektion* noch verfolgen oder direkt ineinander hineinrennen, ist die zeitliche Verzögerung stark verschieden.

Auflistung der Parameter: kritische Distanz *Zusammensein*, kritische Zeitdauer *Zusammensein*, Toleranz *Zusammensein*, kritische Distanz *Intersektion*, kritische Zeitdauer *Intersektion*, kritische Distanz *Getrenntsein*, kritische Zeitdauer *Getrenntsein*, Toleranz *Getrenntsein*, kritische Verzögerungszeit zwischen *Intersektion* und Beginn *Getrenntsein*

5.6.2 Räumliche Perspektive

Der Algorithmus KD_r basiert ebenfalls auf den in der Abbildung 5.13 dargestellten Schritten. Einziger Unterschied zum Algorithmus KD_i ist die Verwendung der räumlichen Ausführungen der Methoden zur Detektion von *Zusammensein*, *Intersektion* und *Getrenntsein*. Dies widerspiegelt sich auch in den Parametern des Algorithmus KD_r .

Auflistung der Parameter: Nachbarschaftsgrösse *Zusammensein*, kritische Zeitdauer *Zusammensein*, Toleranz *Zusammensein*, kritische Zeitdauer *Intersektion*, Toleranz *Intersektion*, Nachbarschaftsgrösse *Getrenntsein*, kritische Zeitdauer *Getrenntsein*, Toleranz *Getrenntsein*, kritische Verzögerungszeit zwischen *Intersektion* und Beginn *Getrenntsein*

5.7 Meiden

5.7.1 Individuelle Perspektive

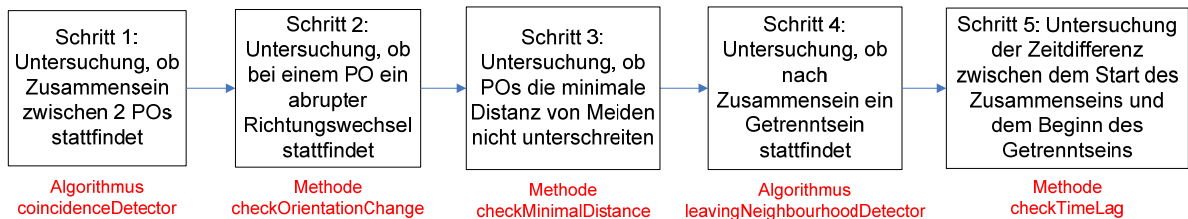


Abbildung 5.14: Detektion von *Meiden* (individuelle Perspektive)

Skizze der verschiedenen Schritte zur Detektion von *Meiden* mit der individuellen Perspektive. In roter Farbe aufgeführt sind die verwendeten Methoden, welche im Anhang dieser Arbeit als Java-Code zu finden sind.

Der Algorithmus MD_i setzt sich zusammen aus der Detektion von *Zusammensein* und der Entdeckung von *Getrenntsein*. Zudem wird untersucht, ob ein abrupter Richtungswechsel in den Bewegungen eines Punktobjekts zu beobachten ist und wie nahe sich die Punktobjekte maximal gekommen sind. Dabei darf die minimale Annäherungsdistanz nicht kleiner als der Wert des Parameters kritische Distanz von *Meiden* sein. Zum Abschluss wird noch die Verzögerungszeit zwischen dem Start des *Zusammenseins* und dem Beginn des *Getrenntseins* analysiert. Nur falls diese Zeitdifferenz die kritische Zeitverzögerung nicht überschreitet, wird von einem *Meiden* gesprochen.

Auflistung der Parameter: kritische Distanz *Zusammensein*, kritische Zeitdauer *Zusammensein*, Toleranz *Zusammensein*, kritische Zeitdauer ohne *Zusammensein*, kritische Distanz *Getrenntsein*, kritische Zeitdauer *Getrenntsein*, Toleranz *Getrenntsein*, kritische Distanz *Meiden*, kritischer Winkel des Bewegungsrichtungswechsels, kritische Verzögerungszeit zwischen Beginn des *Zusammenseins* und Start des *Getrenntseins*

5.7.2 Räumliche Perspektive

Im Vergleich zur individuellen Perspektive steht für den Algorithmus MD_r kein Parameter für die kritische Distanz von *Meiden* zur Verfügung, weshalb der Algorithmus nur aus 4 Hauptschritten besteht. Somit wird auf die Durchführung von Schritt 3 der Abbildung 5.14 verzichtet.

Auflistung der Parameter: Nachbarschaftsgrösse *Zusammensein*, kritische Zeitdauer *Zusammensein*, Toleranz *Zusammensein*, kritische Zeitdauer ohne *Zusammensein*, Nachbarschaftsgrösse *Getrenntsein*, kritische Zeitdauer *Getrenntsein*, Toleranz *Getrenntsein*, kritischer Winkel des Bewegungsrichtungswechsels, kritische Verzögerungszeit zwischen Beginn des *Zusammenseins* und Start des *Getrenntseins*

6 Experimente

Die im letzten Kapitel vorgestellten Algorithmen zur Detektion der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* wurden diversen Tests unterzogen, um deren Anwendbarkeit zu überprüfen. Dabei wurden einerseits simulierte Bewegungsdaten aus einem bereits bestehenden Modell (Steering Behavior von Schnellhammer und Feilkas 2001) und andererseits ein realer Raum-Zeit-Datensatz von Schülern, welche an einem Freiluftspiel in Amsterdam teilgenommen hatten (Frequentie1550-Projekt), verwendet. Diese Bewegungsdatensätze sowie ihre Verwendung bei den Tests der Algorithmen werden in diesem Kapitel diskutiert. Dabei gilt es zu beachten, dass alle Experimente in der agentenbasierten Simulationsumgebung Repast Symphony (Version 1.2.0, Java 6.0) durchgeführt wurden. Repast Symphony ist eine auf <http://repast.sourceforge.net/> kostenlos erhältliche Simulationsumgebung, welche in die Entwicklungsumgebung Eclipse integriert ist und auf Java und Groovy basiert. Die Resultate dieser Algorithmen-Tests werden dann in Kapitel 7 beschrieben und in Kapitel 8 diskutiert.

6.1 Simulationsdaten des Steering Behavior-Modells

Um einen ersten Eindruck zu erhalten, wie erfolgreich die entwickelten Algorithmen bei der Detektion von Reaktionsbewegungsmustern in Raum-Zeit-Daten sind, wurde das von Schnellhammer und Feilkas (2001) entwickelte Modell Steering Behaviors zur Simulation von Verhaltenskombinationen verwendet. Dieses Modell ist kostenlos für die Verwendung und Weiterentwicklung auf <http://www.steeringbehaviors.de> verfügbar. Das Steering Behavior-Modell wurde während einer Diplomarbeit erarbeitet, wobei alle in der Arbeit von Craig Reynold (1999) diskutierten Verhaltensformen, basierend auf der Sprache Java, implementiert wurden. Dies macht das Modell für diese Arbeit sehr wertvoll, da mit dessen Hilfe Verhaltensformen wie *Verfolgen*, *Entfliehen* oder *Separieren* simuliert werden können.

6.1.1 Beschreibung des Modells

Die in diesem Unterkapitel gemachten Erklärungen zur Funktionsweise des Steering Behaviors-Modell basieren alle auf den Ausführungen von Schnellhammer und Feilkas (2001) und können auf deren Website im Detail nachgelesen werden. Das Modell der Steering Behaviors ist eine Weiterentwicklung des in der Arbeit von Craig Reynolds (1987) vorgestellten „Boids“-Modell, welches für die Simulation von koordinierten Bewegungen, wie sie beispielsweise bei Fisch- oder Vogelschwärmen zu beobachten sind, verwendet werden kann. „Boids“ bezeichnet dabei eine einzelne simulierte Kreatur. Das ursprüngliche Modell von Reynolds (1987) basiert auf den drei Verhalten *Separation* (zum Abwenden von Kollisionen), *Angleichen* („Alignment“, zur Bewegung in dieselbe Richtung) und *Kohäsion* (für den Zusammenhalt innerhalb einer Gruppe). Mit Hilfe dieses Modells gelang es Reynolds, Bewegungen von Schwärmen erfolgreich zu simulieren. Im Jahr 1999 publizierte Reynolds eine Liste von zusätzlichen Verhalten, welche als wichtige Bausteine für die Steuerung von

Verhaltensweisen von autonomen Agenten dienen. Diese Bausteine bilden den sogenannten Baukasten für Systeme, in welchen die Agenten je nach zugewiesenen Steering Behaviors verschieden auf Gegebenheiten reagieren und somit komplexe Situationen unterschiedlich gut meistern können. Unter dem Begriff Steering Behavior werden also alle elementaren Reaktionsformen zusammengefasst, welche für die Steuerung der Verhaltensweisen von autonomen Charakteren in einem vorgegebenen System dienen. Zu diesen elementaren Verhaltensformen gehören beispielsweise *Suchen*, *Verfolgen*, *Fliehen*, *Ankommen*, *Wandern*, *Separation*, *Angleichen (Alignment)*, *Kohäsion*, *Herdenbildung (Flocking)*, *Enthalten sein* oder das *Meiden* von Hindernissen, welche alle im Steering Behaviors-Modell von Schnellhammer und Feilkas (2001) implementiert wurden. Für die Modellierung der autonomen Agenten verwendeten Schnellhammer und Feilkas (2001) ein vereinfachtes Fahrzeugmodell, wie es von Reynold (1999) vorgeschlagen wurde. Dabei wird das Fahrzeug als Punktmasse mit einer Position und einer Geschwindigkeit definiert. Mit Hilfe zweier Vektoren kann die Ausrichtung des Fahrzeugs abgebildet werden. Zu jedem Zeitpunkt der Simulation wird die Geschwindigkeit mit dem erzeugten Kraftvektor addiert und das Ergebnis zur aktuellen Position dazugerechnet, um die neue Position zu bestimmen. Die Kraftvektoren werden laufend neu berechnet und sind davon abhängig, welche Verhalten einem Fahrzeug zugeordnet sind und welche Gegebenheiten (andere Fahrzeuge, Hindernisse) sich in der nächsten Umgebung befinden.

6.1.2 Verwendung des Modells – Generierung von simulierten Bewegungsdaten

Wie im letzten Unterkapitel beschrieben, ist es mit Hilfe des Steering Behavior-Modells und den darin enthaltenen Verhaltensformen *Verfolgen*, *Entfliehen* und *Separation* möglich, die beiden Reaktionsbewegungsmuster *Verfolgen/Entfliehen* sowie *Meiden* zu simulieren. Um das Modell von Schnellhammer und Feilkas (2001) für den Test der entwickelten Algorithmen zur Detektion der beiden genannten Reaktionsbewegungsmuster nutzen zu können, waren aber noch einige Anpassungen notwendig. Da der dem Modell von Schnellhammer und Feilkas (2001) zugrunde liegende Code frei verfügbar ist, wurden mit Hilfe der Programmiersprache Java diese notwendigen Veränderungen vorgenommen. Dazu gehört eine Methode zur Ausgabe der Positionen der sich bewegenden Punktobjekte in ein File. Ebenfalls wurde eine Methode entwickelt für die Speicherung aller ablaufenden Verhaltensformen. Diese Liste mit Verhaltensformen enthält alle relevanten semantischen Informationen über die Bewegungsmuster, welche sich in den aufgezeichneten Bewegungsdaten abspielen und ist somit für die Bestimmung des Erfolges der Algorithmen bei der Detektion von grosser Wichtigkeit. Mit Hilfe des Vergleichs der detektierten Reaktionsbewegungsmuster und der tatsächlich ablaufenden Verhalten kann die Güte (z.B. Error of Omission, Error of Commission) der Algorithmen bestimmt werden. Des Weiteren wurde das Modell so angepasst, dass die Bewegungsvektoren an den Rändern der Spielzone gespiegelt werden. Dies war notwendig, um die Grösse des Spielfeldes zu begrenzen und dadurch die Anzahl der ablaufenden Reaktionsbewegungsmuster zu erhöhen.

6.1.3 Experimentdesign

Für die Evaluation der Algorithmen zur Detektion der beiden Reaktionsbewegungsmuster *Verfolgen/Entfliehen* und *Meiden*, wurden mit dem eigens erweiterten Simulationsmodell *Steering Behavior* von Schnellhammer und Feilkas (2001) sowohl für die individuelle, als auch für die räumliche Perspektive Bewegungsdatensätze generiert. Für den Fall der individuellen Perspektive wurden die Bewegungen jeweils von acht Punktoobjekten in einem begrenzten euklidischen Raum (640 x 480 Zellen) simuliert. Im Gegensatz dazu wurden diese acht Punktoobjekte für die Evaluation der Algorithmen der räumlichen Perspektive auf einem Strassennetzwerk laufen gelassen. Die Grösse des Netzwerkraumes ist aufgrund der Strassenschranken im Vergleich zum euklidischen Raum geringer, die Bounding Box des Strassennetzes beträgt aber ebenfalls 640 x 480 Zellen. Sowohl der euklidische als auch der Netzwerk-Raum sind in der Abbildung 6.1 dargestellt. In den folgenden Abschnitten werden die Einstellungen und Konstellationen im *Steering Behavior*-Modell der verschiedenen Algorithmestests beschrieben. Die Resultate dieser Tests sind in Kapitel 7 zu finden und die Diskussion folgt in Kapitel 8.

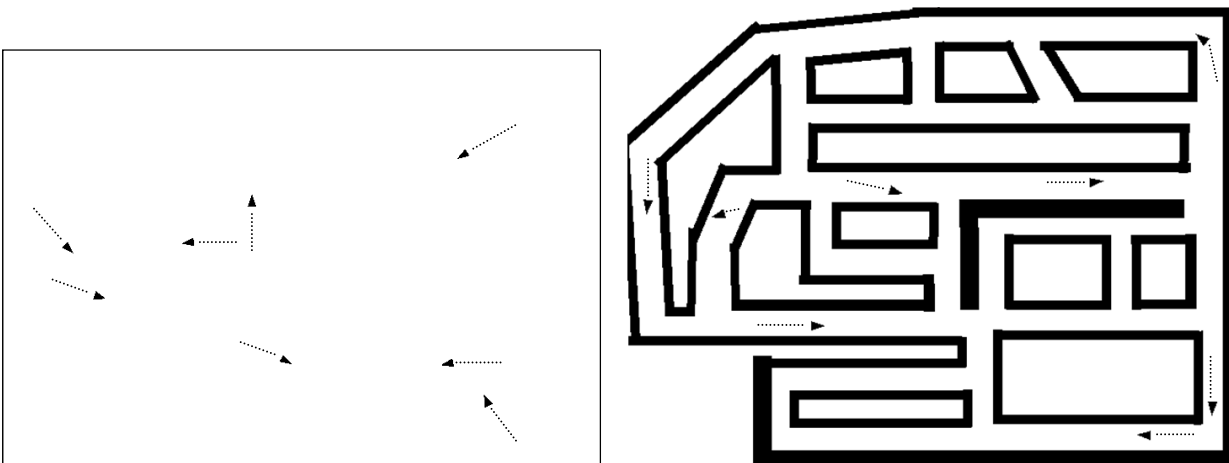


Abbildung 6.1: Euklidischer Raum (links) und Netzwerkraum (rechts) im Steering Behavior-Modell

In der linken Abbildung dargestellt sind Punktoobjekte, welche sich im euklidischen Raum des *Steering Behavior*-Modells frei bewegen können. Die Bewegungen der Punktoobjekte sind lediglich durch die Ränder des euklidischen Raums eingeschränkt. In der rechten Abbildung ist das Modell mit dem Strassennetzwerk sichtbar. Darin können sich die Punktoobjekte nur zwischen den schwarz dargestellten Strassenschranken bewegen.

Da im *Steering Behavior*-Modell das Verhalten *Konfrontation* nicht zur Verfügung steht, musste auf eine Evaluation der Algorithmen zur Detektion von *Konfrontationen* mit Hilfe dieses Modells verzichtet werden. Wie im nachfolgenden Kapitel beschrieben, konnten diese Algorithmen dafür mit Hilfe des *Frequentie1550*-Datensatz und den semantischen Zusatzinformationen zu den sich im Datensatz abspielenden *Konfrontationen* getestet werden.

Test des Algorithmus VED_i

Für den Test des Algorithmus VED_i wurden Bewegungsdaten von sechs Punktoobjekten, welche sich ganz zufällig im euklidischen Raum bewegen, generiert. Für diesen Zweck wurden im *Steering Behavior*-Modell sechs Punktoobjekte kreiert und mit dem Verhalten „Wandern“ ausgestattet. Zudem wurden die Positionen von zwei Punktoobjekten aufgezeichnet, welche sich verfolgen. Dabei wurde die

kritische Verfolgungsdistanz auf 75 Raumeinheiten eingestellt und die kritische Distanz, ab welcher das Fluchtpunktobjekt zu fliehen beginnt, auf 60. An dieser Stelle soll angemerkt werden, dass die Distanzangaben im Steering Behavior-Modell keine Masseinheit (z.B. Meter) besitzen. Während allen Ausführungen in diesem und in den nachfolgenden Kapiteln muss man sich bei der Betrachtung von Distanzangaben (z.B. Aktivdistanz eines Bewegungsmusters) also vor Augen führen, dass sich die Bewegungen in einem Raum mit einer Ausdehnung von 640 x 480 Raumeinheiten abspielen. Somit beträgt beispielsweise die Aktivdistanz von Verfolgen 11.7% der Breite des euklidischen Raums. Das gleiche gilt für die zeitliche Dimension, welche ebenfalls keine Masseinheit (z.B. Sekunden) besitzt. Um sich etwas besser vorstellen zu können, was die Zeiteinheit des Steering Behavior-Modells bedeutet, kann man die Geschwindigkeit, mit welcher sich die Punktobjekte bewegen, beiziehen. Die Punktobjekte legen, falls im Experimentdesign nichts anderes beschrieben, im Schnitt 2 Raumeinheiten pro Zeitschritt zurück. Somit bewegen sie sich pro Zeiteinheit um 0.3% der Raumbreite. Während den nachfolgenden Ausführungen zu den Einstellungen der einzelnen Experimenten sowie zu den Resultaten in Kapitel 7 wird darauf verzichtet, die Parameterwerte jeweils im Verhältnis der Grösse des euklidischen bzw. Netzwerkraums anzugeben.

Damit das fliehende Punktobjekt die Möglichkeit hat, sich wieder von seinem Verfolger zu entfernen, wurden dessen Werte der maximalen Kraft und der maximalen Geschwindigkeit im Vergleich zu den Werten des Verfolgerpunktobjekts leicht höher gewählt. Dies führt dazu, dass das Fluchtobjekt im Schnitt 2.3 Raumeinheiten pro Zeitschritt zurücklegt. Mit Hilfe dieser Einstellungen konnten für alle acht Punktobjekte Datensätze mit Bewegungen über 30'497 Zeitschritte erstellt werden. In diesen Daten lassen sich total 35 verschiedene Situationen mit *Verfolgen/Entfliehen* detektieren. Mit Hilfe von Experimenten konnten diese 35 beobachteten Ereignisse mit den Bewegungsmustern *Verfolgen/Entfliehen*, welche vom Algorithmus VED_i detektiert wurden, verglichen werden und daraus die idealen Parameter für diesen Algorithmus bestimmt werden.

Test des Algorithmus VED_r

Für die Überprüfung der Güte des Algorithmus VED_r wurden Positionen von acht verschiedenen Punktobjekten, welche sich auf dem in Abbildung 6.1 dargestellten Strassennetzwerk bewegen, gesammelt. Dabei gab es vier Punktobjekte, welche sich rein zufällig auf dem Netzwerk bewegten, sowie zwei Punktobjekt-Paare mit jeweils einem Verfolger und einem Fliehenden. Dies bedeutet, dass im Vergleich zu den Experimenten im euklidischen Raum sich zwei Punktobjekte mehr verfolgen. Dies wurde bewusst so gewählt, da sich im Netzwerkraum ansonsten zu wenige Situationen mit *Verfolgen/Entfliehen* abgespielt hätten. Aufgrund der stark eingeschränkten Bewegungen auf dem Netzwerk kommt es viel seltener zu einem Zusammentreffen zwischen Verfolger und Fluchtobjekt als im euklidischen Raum, weshalb zwei Verfolgern/Fliehender-Paare im Steering Behavior-Modell laufen gelassen wurden. Dadurch konnte sichergestellt werden, dass sich in den aufgezeichneten Bewegungsdaten eine für die Experimente genügend grosse Anzahl von *Verfolgen/Entfliehen* abspielt. Wie im Falle des euklidischen Raumes wurden die zufälligen Bewegungen durch das Verhalten

„Wandern“ gesteuert. Damit die Punktobjekte stets innerhalb der Strassenschranken blieben, wurde allen das Verhalten „Containment“ zugewiesen. Somit war jedes Punktobjekt über alle Zeitschritte den zwei Schranken des Strassensegments zugewiesen und konnte dadurch das Netzwerk nicht verlassen. Weiter gilt es zu beachten, dass die beiden Verfolger-Punktobjekte noch zusätzlich mit dem Verhalten „Pursuit“ mit einer Aktivdistanz von 50 ausgestattet wurden. Dies bedeutet, dass im Falle einer Annäherung auf 50 Distanzeinheiten an das Zielpunktobjekt die *Verfolgung* aufgenommen wird. Den beiden potentiell verfolgten Punktobjekten wurde das Verhalten „Evade“ zugewiesen mit einer Aktivdistanz von 45. Wie im Falle des euklidischen Raums wurden die Maximalwerte des Kraftvektors und der Geschwindigkeit für die beiden fliehenden Punktobjekte leicht höher gewählt als diejenigen Werte der beiden Verfolgerpunktobjekte. Dadurch konnte sichergestellt werden, dass es aufgrund der kleineren Aktivdistanz von „Evade“ im Vergleich zu „Pursuit“ überhaupt zu Verfolgungssituationen kommen konnte. Zudem konnte es aufgrund der erhöhten Maximalwerte für die beiden Fluchtobjekte zu einer erfolgreichen Flucht und daher zu mehreren zeitlich voneinander getrennten Situationen mit *Verfolgen/Entfliehen* kommen. In den Datensätzen, welche mit diesen Einstellungen generiert wurden, sind die zufälligen Bewegungen von vier Punktobjekten sowie 18 verschiedene Events mit *Verfolgen/Entfliehen* zwischen den anderen vier Punktobjekten enthalten. Um diese Reaktionsmuster mit Hilfe des Algorithmus VED_r zu detektieren, mussten die Bewegungsdaten noch mit Hilfe eines Map-Matching Algorithmus auf ein lineares Strassennetzwerk angepasst werden. Dieses lineare Strassennetzwerk enthält alle Mittellinien der in der Abbildung 6.1 dargestellten Strassensegmente. Ein Bild dieses Strassennetzwerks aus Repast Symphony zeigt die Darstellung 6.2.

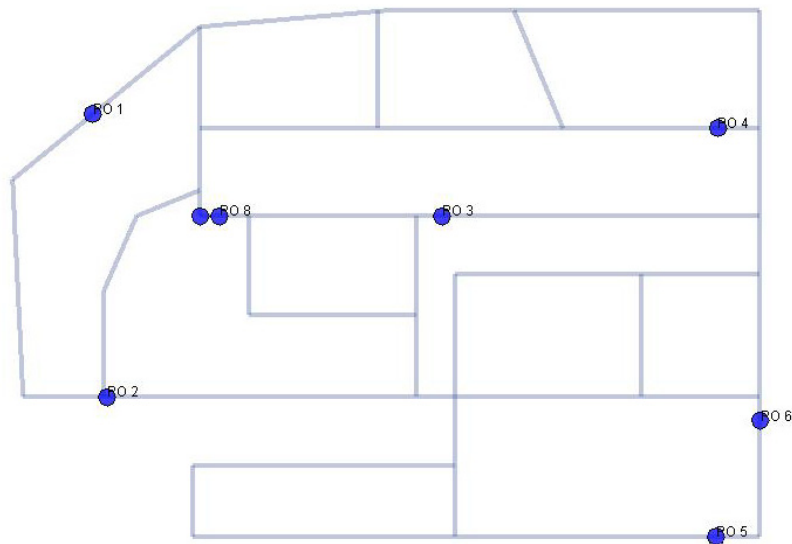


Abbildung 6.2: Netzwerkraum in Repast Symphony

Alle generierten Bewegungsdaten aus dem Steering Behavior-Modell wurden in Repast Symphony mit Hilfe eines einfachen Map-Matching Algorithmus auf dieses Strassennetzwerk projiziert und anschliessend für die Experimente der beiden Algorithmen VED_r und MD_r verwendet.

Aufgrund der geringen Abweichungen zwischen dem Netzwerk im Steering Behavior-Modell und demjenigen in Repast war die Verwendung eines sehr einfachen Map-Matching Algorithmus, welcher die Positionsangaben alleine aufgrund der Distanz zu den Strassensegmenten zuwies, ausreichend. Die

Koordinatenpaare der Bewegungen wurden dann jeweils auf die entsprechende Position auf den ihnen zugewiesenen Strassensegmenten gesetzt, damit die Bewegungen genau auf dem Netzwerk erfolgten.

Test des Algorithmus MD_i

Die Tests des Algorithmus MD_i wurden mit Hilfe der Bewegungsdaten von zwei Punktobjekten, welche die Anwesenheit in der nahen Umgebung von anderen Punktobjekten meiden sowie von sechs Punktobjekten, welche sich zufällig im Raum bewegen, durchgeführt. Für die Generierung dieser Daten wurden im Steering Behavior-Modell in einem euklidischen Raum die Bewegungen von sechs mit dem Verhalten „Wandern“ ausgestatteten Punktobjekten sowie von sechs Punktobjekten mit dem Verhalten „Separieren“ simuliert. Die Aktivdistanz von „Separieren“ wurde dabei auf 30 Einheiten eingestellt. Diese Einstellungen führten dazu, dass sobald sich eines der sechs zufällig im euklidischen Raum bewegenden Punktobjekte auf eine Distanz kleiner als 30 an eines der beiden anderen Punktobjekte annäherte, es zu einer Separationsbewegung seitens dieser beiden Objekte kam. In den Daten enthalten sind während einer Zeitspanne von 5'146 Zeitschritten insgesamt 36 Situationen mit einer Separation zwischen den Punktobjekten, welche mit Hilfe des Algorithmus MD_i gesucht werden können.

Test des Algorithmus MD_r

Für den Fall der räumlichen Perspektive (Algorithmus MD_r) wurden exakt die gleichen acht Punktobjekte mit den genau gleichen Verhalten und Einstellungen auf dem Netzwerk laufen gelassen und die Positionen der Bewegungen aufgezeichnet. Wie bei den Experimenten des Algorithmus VED_r wurden die mit dem Steering Behavior-Modell generierten Bewegungsdaten mit Hilfe eines einfachen Map-Matching Algorithmus den entsprechenden Strassensegmenten zugewiesen. In diesen Daten enthalten sind 33 Situationen mit *Meiden*, welche es in den Experimenten mit Hilfe des Algorithmus VED_r zu detektieren gilt.

6.2 Frequentie1550-Datensatz

Um die Algorithmen zur Detektion der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* neben deren Anwendbarkeit bei simulierten Daten ebenfalls auf deren Güte im Falle von realen Raum-Zeit-Daten zu testen, wurde ein Datensatz von Schülern bei einem Freiluftspiel (Frequentie1550-Projekt) verwendet. In diesem Unterkapitel folgen nun einige erläuternde Hintergrundinformationen zum Frequentie1550-Datensatz, eine Beschreibung der genauen Verwendung der Daten, Ausführungen zur den notwendig gewordenen Vorprozessierungsschritte sowie Erklärungen zu den durchgeführten Experimenten.

6.2.1 Beschreibung des Datensatzes

Die Erläuterungen in diesem Kapitel basieren auf den Ausführungen von Admiraal et al. (2007). Der Frequentie1550-Datensatz enthält Positionsangaben von niederländischen Schülern, welche in Amsterdam am Freiluft-Spiel „Schnitzeljagd“ teilnahmen. Dieses Spiel wurde von der Waag-Society



Abbildung 6.3: Karte des spät-mittelalterlichen Amsterdams (Waag Society 2007)

Karte des Zentrums von Amsterdam des Jahres 1550 mit den sechs verschiedenen Spielzonen des Freiluftspiels des Frequentie1550-Projekts

organisiert und an zehn verschiedenen Tagen im Juni 2007 durchgeführt. Ziel dieses Spiels war es, die Schüler ins Jahr 1550 zurückzusetzen und ihnen dabei das spätmittelalterliche Amsterdam näher zu bringen. Eine Karte von Amsterdam im Jahre 1550 ist in der Abbildung 6.3 dargestellt. Bevor die Schüler auf die Reise durch die Geschichte Amsterdams geschickt wurden, gab man ihnen einige Hintergrundinformationen in Form einer Rahmengeschichte auf den Weg. Durch diese Geschichte wurde den Schülern der Kontext des Spiels (Zeit und Ort, teilnehmende Charakteren, etc.) sowie ihre

Rolle darin klar gemacht. Die Schüler übernahmen dabei die Rolle von Pilgern, welche Amsterdam besuchen. Dabei war es ihr Ziel, die heilige Hostie (ein spezielles Relikt, als Mirakel von Amsterdam bekannt) zu besichtigen. Allerdings war das Relikt auf mysteriöse Art und Weise verschwunden und der Spielleiter, in Gestalt des Stadtverwalters, bot den Pilgern das Bürgerrecht von Amsterdam an, wenn sie als Gegenleistung mithalfen, das heilige Relikt ausfindig zu machen. Die Schüler begaben sich nach dieser Einführung in die mittelalterliche Welt des Jahres 1550 und versuchten in Teams von 4-5 Personen in den Strassen von Amsterdam, dem Mirakel auf die Schliche zu kommen. Dabei besetzten jeweils 2 Teammitglieder die Position im Hauptquartier, während die restlichen



Abbildung 6.4: Spielerin im Hauptquartier (Waag Society 2007)

Beispielsbild des Hauptquartiers eines Teams

Spieler der Gruppe, ausgerüstet mit Mobiltelefonen (mit GPS und Internet), im mittelalterlichen Amsterdam auf Punktejagd gingen. Die Spieler im Hauptquartier waren ihrerseits mit Laptop, Internet und Telefon ausgestattet (siehe Abbildung 6.4). Aufgrund der GPS-Empfänger wurde den Spielern auf ihren Mobiltelefonen immer ihre aktuelle Position auf einer digitalen Karte angezeigt. Dem Hauptquartier standen neben der Positionsangabe der eigenen Teammitglieder auch noch diejenigen der anderen Teams zur Verfügung. Durch dieses Equipment waren die Spieler unterwegs stets mit dem Hauptquartier in Verbindung und konnten sich wichtige Informationen über die geschichtlichen Hintergründe von Amsterdam oder über die Positionen von anderen Spielern senden lassen. Ziel des Spiels war es, in den Gassen von Amsterdam so viele Checkpoints wie möglich anzusteuern und an diesen die gestellten Aufgaben möglichst schnell und erfolgreich zu erfüllen, um dadurch so viele Punkte wie möglich zu sammeln. Dasjenige Team mit den meisten Punkten gewann das Spiel und somit das Bürgerrecht von Amsterdam. Während des Spiels konkurrierten sich also die

verschiedenen Schülergruppen, wobei sie sich gegenseitig Punkte wegschnappen konnten, indem sie andere Teams anrennpelten und diese zu einem Duell herausforderten. Der Orden der Teams (abgeleitet aus den gesammelten Punkten) entschied dabei, wer als Sieger aus dem Duell hervorging. Zudem hatten die Spieler unterwegs die Möglichkeit, sogenannte virtuelle Bomben auf die Strassen zu legen. Dadurch wurde dem nächsten Team, welches sich auf dieser Strasse bewegte, für kurze Zeit die Verbindung zum Hauptquartier unterbrochen. Weiter hatten die Teams die Möglichkeit, sich mit Hilfe sogenannter Mönchskutten für die anderen Spieler unsichtbar zu machen. Weitere Informationen zum genauen Ablauf des Spiels und den pädagogischen Hintergründen können auf den Webseiten <http://freq1550.waag.org/>, <http://www.waag.org/project/frequentie> und <http://www.frequentie1550.nl/> sowie in Admiraal et al. (2007) und Akkermann et al. (2008) nachgelesen werden.

Diese Ausführungen zeigen, dass der Frequentie1550-Datensatz hinsichtlich der Detektion von möglichen Reaktionsbewegungsmustern doch einiges an Potential aufweist. Dies im Speziellen aufgrund der Spielregel, dass sich die Teams gegenseitig konkurrenzieren und sich durch *Konfrontationen* Punkte wegnehmen konnten. Daher ist zu erwarten, dass es zu einigen Reaktionsverhaltensweisen wie *Verfolgen/Entfliehen* und *Konfrontationen* gekommen ist. Ebenfalls ist es sehr gut möglich, dass sich die Mannschaften gegenseitig gemieden haben, falls sie einer *Konfrontation* ausweichen wollten.

Der Datensatz enthält, wie in der folgenden Abbildung 6.5 sichtbar, die folgenden Angaben: Farbe des Teams, Position im WGS84-Koordinatensystem, Position auf der mittelalterlichen Karte, Zeit, Zone in der sich der Spieler aufhält, Name des Spielers und Datum.

Fields:

name: string, name of the team

lat: decimal, geographical latitude, decimal degrees, WGS84

lon: decimal, geographical longitude, decimal degrees, WGS84

rx: int, screen old map x coordinates

ry: int, screen old map y coordinates

TimeTxt: decimal, time of the day, UCT, hhmmss.ss

zonename: string, Game zone

label: string, player nickname

datetime: datetime, full datetime, UCT, MM/dd/yyyy hh:mm:ss

*****data*****

"name";"lat";"lon";"rx";"ry";"TimeTxt";"zonename";"label";"datetime"

"orange";52.372440;4.899937;"1794";"1927";"063752.148";"zone6";"jiwolo";6/1/2007 6:37:52

Abbildung 6.5: Struktur des Frequentie1550-Datensatzes

Eine Auflistung der verschiedenen Informationen sowie deren Datenformat des Frequentie1550-Datensatzes. Die letzten beiden Zeilen widerspiegeln die Reihenfolge der enthaltenen Daten sowie die erste Zeile des Datensatzes.

Bevor in den nächsten beiden Abschnitten erläutert wird, wie die Daten für den Test der entwickelten Algorithmen zur Detektion der ausgewählten Reaktionsbewegungsmuster verwendet werden, folgen in diesem Unterkapitel ein paar beispielhafte Erläuterungen, wie andere Forscher den Frequentie1550-

Datensatz in ihren Untersuchungen einsetzen konnten. Orellana et al. (2009) konzeptualisierten Bewegungen als Resultat von Interaktionen. Dabei sind Interaktionen zwischen Individuen,

	Individual	Environment	Collective
Individual	<u>Approximation</u> ^a Separation Following Crossing	Arrival Leaving Passing by	Flocking Detaching Queuing Guiding
Environment	Constriction Stop	Containment Inclusion	<u>Suspension</u> ^a Restriction <u>Attraction</u> ^a Repulsion
Collective	Following Evasion Collecting	Concentration Dispersion Stop	Fusion Division Disintegration

a. Underlined interactions were used in the implementation

Abbildung 6.6: Beispiele von Interaktionsformen (Orellana et al. 2009)

Auflistung von möglichen Interaktionsformen, welche sich aus Bewegungen von einzelnen oder Gruppen von Punktobjekten ergeben.

Verfahren, mit welchem sie *Stops* in den Bewegungen der Schüler (*Suspensions*) detektieren konnten. Der Ansatz von Orellana und Wachowicz (2011) basiert auf Bewegungsvektoren und deren räumlicher Einbettung („spatial association“). Um die räumliche Assoziation der Bewegungsvektoren zu berechnen, wurde ein Ansatz namens LISA (Local Index of Spatial Association), welcher eine lokale Version des Moran's I darstellt, verwendet. Mit Hilfe dieses Verfahrens können die aus den Bewegungsvektoren abgeleiteten Geschwindigkeiten der Bewegungen auf räumliche Autokorrelation und somit auf mögliche Clusters in der Werteverteilung hin untersucht werden und so Orte von *Anhalte-Bewegungen* detektiert werden. Für eine detaillierte Beschreibung des Verfahrens siehe Orellana und Wachowicz (2011).

6.2.2 Verwendung der Daten

Die Frequentie1550-Daten sind laut Orellana und Wachowicz (2011) gezeichnet von einer relativ schlechten Qualität des GPS-Signals. In den Datenreihen können diverse unnatürliche Sprünge in den Bewegungspfaden und Positionsangaben auf Häusern beobachtet werden. Dazu kommt, dass während des Spiels Störungen der Kommunikation zwischen den Spielern draussen und dem Hauptquartier auftraten, welche zu Verlusten von Positionsdaten geführt haben. Der von Orellana und Wachowicz (2011) neu entwickelte Ansatz zur Detektion von anhaltenden Personen hat allerdings den grossen Vorteil, dass er auch auf stark fehlerhafte Raum-Zeit-Daten angewendet werden kann. Im Gegensatz dazu war es für die Implementierung des Ansatzes von Orellana et al. (2009) notwendig, den Frequentie1550-Datensatz diversen Vorprozessierungsschritten zu unterziehen, um in diesen Daten Interaktionsmuster zu detektieren. Dabei wurden nur diejenigen Spieler berücksichtigt, über welche im

Gruppierungen und der Umwelt vorstellbar. Eine Zusammenfassung von möglichen Interaktionsformen zeigt die Abbildung 6.6. Orellana et al. (2009) fokussierten sich bei ihrer Implementierung der vorgestellten Metapher „Bewegung als Interaktion“ auf die in dieser Matrix unterstrichenen Interaktionen *Annäherung* (*Approximation*), *Anhalten* (*Suspension*) und *Anziehung* (*Attraction*) und versuchten diese drei Interaktionsformen in den Frequentie1550-Daten zu detektieren. Orellana und Wachowicz (2011) entwickelten wenig später ein neues

Minimum während 10 Minuten Positionsangaben zur Verfügung standen. Weiter wurden fehlerhafte Positionsangaben herausgefiltert. Um die für die Analyse der Interaktionsformen notwendigen Trajektorien zu generieren, wurde nach der Filterung zwischen den verfügbaren Positionsangaben der Spieler interpoliert. Das Resultat dieser Vorprozessierung waren kontinuierliche Bewegungspfade in Raum und Zeit, welche in der Abbildung 6.7 dargestellt sind. Da für die Detektion der in dieser Arbeit vorgestellten und formalisierten Reaktionsbewegungsmuster ebenfalls lineare Trajektorien mit einer hohen

zeitlichen Auflösung notwendig sind, wurden die Grundsätze dieses Vorprozessierungsverfahrens für diese Arbeit übernommen. Das genaue Verfahren wird in der Folge beschrieben.

Filterung

Beim Einlesen der Daten erfolgte bereits ein erster Filterschritt mit Hilfe einer Bounding Box um das Stadtzentrum von Amsterdam, in welchem sich die Spieler bewegten. Dadurch konnte verhindert



Abbildung 6.8: Bounding Box des Stadtzentrums von Amsterdam (OpenStreetMap 2010)

Bounding Box um das Stadtzentrum von Amsterdam, in welchem sich das Freiluftspiel Frequentie1550 abgespielt hat. Diese Bounding Box wurde verwendet für den ersten Filterschritt.

wurden alle Spieler mit Positionsangaben über eine Zeitspanne von weniger als 10 Minuten gelöscht (Filterschritt 2). Weiter wurden 2 Positionsangaben zur selben Zeit bereinigt (Filterschritt 3) und für alle Bewegungen die Geschwindigkeiten berechnet und geprüft, ob diese die kritische Geschwindigkeit von 4 m/s (=14.4 km/h) überschreiten. Dieser Wert wurde geschätzt und orientiert

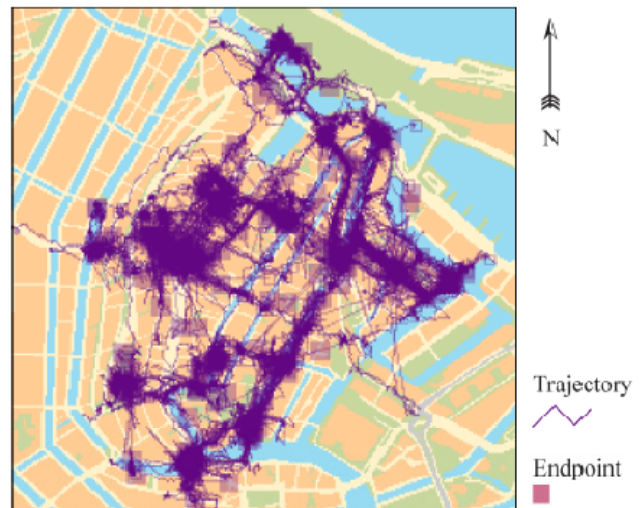


Abbildung 6.7: Trajektorien der Bewegungsdaten des Frequentie1550-Projekts (Orellana et al. 2009)

Darstellung der kontinuierlichen Bewegungspfade aller Spieler, für welche im Minimum während 10 Minuten Bewegungsdaten zur Verfügung standen.

sich an der realistischen maximalen Laufgeschwindigkeit eines Schülers im Alter von 12-14 Jahren. Der Wert wurde bewusst vorsichtig gewählt, um sicherzustellen, dass nicht zu viele Daten herausgefiltert wurden. Bei der Überschreitung dieses Grenzwertes wurden diese Positionsangaben entfernt (Filterschritt 4). Nach dieser Prozedur wurde der Filterschritt 2 erneut durchgeführt um sicherzustellen, dass die Spieldauer der Spieler durch die Schritte 3 und 4 nicht unter die kritische Dauer gesunken war. Am Schluss des ganzen Filterverfahrens standen die Bewegungsdaten von 260 Spielern für die Folgeschritte der Vorprozessierung zur Verfügung.

Map-Matching

Bevor nach den Filterschritten zwischen den Positionsangaben interpoliert werden konnte, galt es noch die Positionsfehler (zum Beispiel Position auf einem Gebäude) zu korrigieren. Dabei wurde auf das sogenannte Map-Matching Verfahren zurückgegriffen. Map-Matching bezeichnet den Prozess, bei dem jede Positionsangabe einem auf einer Karte enthaltenen Infrastrukturelement zugewiesen wird. In den meisten Fällen werden dabei GPS-Koordinaten einem entsprechenden Strassensegment zugewiesen (Schmid et al. 2011). Dies ist auch bei den Frequentie1550-Daten der Fall, da sich die Spieler auf dem Strassennetzwerk von Amsterdam bewegen. In Schmid et al. (2011) werden verschiedene Ansätze von Map-Matching Algorithmen unterschieden: (1) Punkt-zu-Punkt Ansatz: die Position wird einem Vertex des Strassennetzwerks zugewiesen, (2) Punkt-zu-Kurve Ansatz: die Position wird dem entsprechenden Knoten des Netzwerks zugeordnet und (3) Topologischer Ansatz: die Position wird zugewiesen unter Einbezug der topologischen Informationen des Netzwerks, von Fortbewegungseinschränkungen und von zusätzlichen Informationen (z.B. Angaben zur Orientierung, Geschwindigkeit). Da die topologischen Algorithmen bessere Ergebnisse liefern als die anderen beiden Ansätze (Schmid et al. 2011), wurde der Map-Matching Algorithmus von Greenfeld (2002) mit der Erweiterung von Brakatsoulas et al. (2005) verwendet für die Frequentie1550-Daten. Dabei muss beachtet werden, dass es noch andere topologische Algorithmen gibt (z.B. Quddus et al. (2003), Ochieng et al. (2004)), welche zum Teil noch verlässlichere Resultate ergeben, allerdings Zusatzinformationen, wie zum Beispiel über die Orientierung oder die Geschwindigkeit des sich bewegenden Punktobjektes, benötigen. Da solche Informationen für den Frequentie1550-Datensatz nicht vorliegen, war die Verwendung dieser Algorithmen nicht möglich. Der Algorithmus von Greenfeld (2002) benötigt als Eingabedaten nur die Koordinaten (x , y und die Zeit t) der Bewegungen sowie ein Strassennetzwerk und basiert im Wesentlichen auf den drei folgenden Kriterien, welche die Zuweisung eines GPS-Punktes zu einem Strassensegment bestimmen:

- 1) Distanz zwischen dem GPS-Punkt und dem potentiellen Segment,
- 2) Orientierung zwischen der Fahrtrichtung (vom letzten bis zum aktuellen Punkt) und der Ausrichtung des Segments und
- 3) Intersektionswinkel zwischen der Fahrtrichtung und der Ausrichtung des Segments im Falle einer vorliegenden Intersektion

Das Strassennetzwerk von Amsterdam für das Map-Matching Verfahren wurde mit Hilfe der Software ArcGIS (ESRI, Version 9.3.1) basierend auf der Strassenkarte von Bing Maps (Microsoft Corporation, Stand 2010) digitalisiert. Dieses Strassennetzwerk ist in der Abbildung 6.9 als Ausschnitt aus Repast Symphony dargestellt.



Abbildung 6.9: Strassennetzwerk von Amsterdam in Repast Symphony

Darstellung des Strassennetzwerks von Amsterdam für das Map-Matching Verfahren in Repast Symphony sowie der Positionen von fünf Schülerteams, welche am Freiluftspiel Frequentie1550 teilgenommen haben. Das Netzwerk wurde auf der Plangrundlage von Bing-Maps (Microsoft Corporation, Stand 2010) mit der Software ArcGIS (ESRI) digitalisiert.

Zu Beginn des Map-Matching Verfahrens wird ein Initialisierungsschritt durchgeführt, um dem ersten Punkt ein Segment zuzuweisen. Dabei wird einzig das Distanzkriterium angewendet, da es sich um den ersten Punkt handelt und daher die Fahrtrichtung nicht abgeleitet werden kann. Dieser Initialisierungsschritt wird immer dann wiederholt, wenn die kritische Zeitdauer von 21 Sekunden überschritten ist und es daher keinen Sinn mehr macht, die Fahrtrichtung zwischen dem letzten und dem aktuellen Punkt zu berechnen. Die Herleitung der kritischen Zeitdauer von 21 Sekunden folgt im nächsten Abschnitt „Interpolation“. Ist der Initialisierungsschritt erfolgreich durchgeführt worden, werden die benachbarten Segmente des zugewiesenen Strassenabschnitts sowie die benachbarten Abschnitte der benachbarten Abschnitte bestimmt. Für diese Untermenge der total verfügbaren Strassen werden für den nächsten Punkt die Gewichte für die oben diskutierten drei Kriterien berechnet. Dasjenige Strassensegment mit dem höchsten aufsummierten Wert wird ausgewählt, die Bestimmung der Nachbarschaften wird erneut durchgeführt und die Berechnung der Gewichte für den nächsten Punkt folgt. Dies geschieht solange, bis entweder das Verfahren beim letzten Punkt angekommen ist und der Map-Matching Prozess beendet ist oder die kritische Zeitdauer von 21 Sekunden überschritten wird und der Initialisierungsschritt erneut durchgeführt werden muss. Die Gewichte für

die einzelnen Kriterien sowie die Summe der Gewichte werden dabei nach den Formeln von Greenfeld (2002) berechnet:

$$(1) W = W_{AZ} + W_D + W_I$$

W = Totalgewicht,

W_{AZ} = Gewicht für die Orientierung,

W_D = Gewicht für die Distanz zwischen Punkt und Linie

W_I = Gewicht für den Intersektionswinkeln (falls es eine Intersektion gibt)

$$(2) W_{AZ} = C_{AZ} \cdot \cos^{n_{AZ}}(\Delta AZ)$$

$$(3) W_D = C_D - a \cdot D^{n_D}$$

$$(4) W_I = C_I \cdot \cos^{n_I}(\Delta AZ)$$

ΔAZ = Differenz zwischen dem Azimuth der Fahrtrichtung und des zu prüfenden Segments

C_i = Konstante, welche das maximale Gewicht bestimmt

n_i = Power Parameter

Für die Frequentie1550-Daten ergaben sich durch Experimentieren die folgenden Parameterwerte:

$C_{AZ} = 8$ (maximales Gewicht für die Orientierung (Azimuth))

$n_{AZ} = 4$ (Power Parameter für die Orientierung (Azimuth))

$C_D = 14$ (maximales Gewicht für die Distanz)

$n_D = 2.4$ (Power Parameter für die Distanz)

$a = 0.001$ (Parameter fuer die Distanz)

$C_I = 20$ (maximales Gewicht für den Intersektionswinkel)

$n_I = 1$ (Power Parameter für den Intersektionswinkel)

Um das Map-Matching Verfahren noch etwas zu optimieren, wurde aufbauend auf dem Algorithmus von Greenfeld (2002) noch zusätzlich die Erweiterung von Brakatsoulas et al. (2005) eines lokalen „Look-Ahead“ implementiert. Dabei handelt es sich um ein Fenster, welches neben dem aktuellen Punkt auch die Gewichte der nächsten beziehungsweise letzten Punkte miteinbezieht. Dadurch ergibt sich vor allem bei stark sprunghaften Positionsangaben eine klare Glättung des Bewegungspfades. Für den Frequentie1550-Datensatz stellte sich aufgrund der doch stark fehlerhaften Daten ein Fenster von 10 Punkten (5 nächste und 5 letzte Punkte) als ideal heraus.

Nach der Bestimmung des passenden Strassensegments wird der ursprüngliche Punkt senkrecht auf das Segment verschoben und die neuen Koordinaten abgespeichert. Nach der Durchführung des Map-Matching Verfahrens wird zwischen diesen neu generierten Positionen entlang des Strassennetzwerks linear interpoliert, um kontinuierlich Bewegungspfade zu erhalten (für jede Sekunde eine Positionsangabe). Das Interpolationsverfahren wird im nächsten Abschnitt beschrieben und diskutiert.

Interpolation

Bei der Interpolation zwischen zwei sich auf dem Strassensegment befindenden Positionen gibt es drei mögliche Fälle, welche unterschieden werden müssen und für die es eigene Interpolationsverfahren zu entwickeln galt:

- (1) Die Punkte befinden sich auf dem gleichen Segment,
- (2) auf zwei benachbarten Segmenten oder
- (3) nicht auf benachbarten Segmenten (weder 1. noch 2.)

Im ersten Fall ist die Interpolation einfach entlang des Strassensegments zu vollziehen. Sind die Punkte auf benachbarten Segmenten, so wird zuerst auf dem ersten Segment bis zum Schnittpunkt der Segmente interpoliert, bevor auf dem zweiten Abschnitt fortgesetzt wird. Ist weder der Fall 1 noch der Fall 2 zutreffend, muss zuerst mit Hilfe eines Shortest-Path Algorithmus die kürzeste Route zwischen den Positionsangaben bestimmt werden. Entlang dieses kürzesten Pfades kann in der Folge interpoliert werden. Für die Bestimmung der kürzesten Verbindung zwischen den Positionsangaben wurde der Algorithmus zur Berechnung des „Shortest-Path“ nach Dijkstra (1959) implementiert.

Kommen wir nun noch zur Herleitung der kritischen Zeit zwischen zwei Positionsangaben, ab welcher eine lineare Interpolation keinen Sinn mehr macht. In den nachfolgenden Histogrammen (siehe Abbildung 9.10) sind alle zeitlichen Lücken zwischen zwei Positionsangaben aufgezeichnet. Daraus geht hervor, dass die grosse Mehrheit der Lücken nur unwesentlich von der Samplingrate der GPS-Geräte, welche 10 Sekunden betrug, abweicht. Die Ausnahmen bilden einige Ausreisser. Während 40'080 Lücken eine Zeitdauer von 10 Sekunden aufweisen, sind es nur noch 98 Lücken mit einer Zeitspanne von 22 Sekunden. Aus dem Histogramm ablesbar ist die Tatsache, dass es ab 21 Sekunden nur noch wenige Werte gibt. Aus diesem Grund wurde ein Grenzwert für die Interpolation von 21 Sekunden gewählt. Dies bedeutet, dass bei einer zeitlichen Differenz zwischen zwei Positionsangaben von mehr als 21 Sekunden auf eine Interpolation verzichtet wird und während dieser Zeitspanne keine Informationen zur Bewegung der Teams vorliegen und somit während dieser Lücken auch keine Reaktionsbewegungsmuster gefunden werden können.

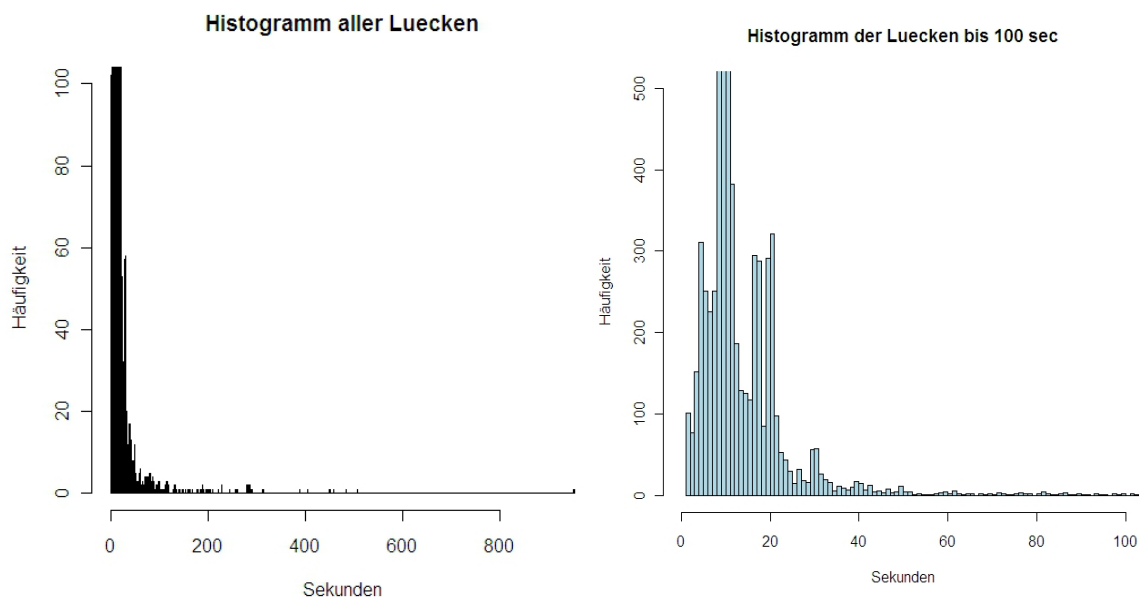


Abbildung 6.10: Histogramm der Lücken in den vorprozessierten Frequentie1550-Daten

Darstellung der Häufigkeitsverteilung der Lücken zwischen zwei Positionsangaben in den vorprozessierten Frequentie1550-Daten. Das Histogramm links zeigt für alle Zeitdauern der Lücken die Häufigkeiten bis zu einem Wert von 100. Im Histogramm rechts dargestellt sind die Zeitspannen der Lücken von 0 bis 100 sowie deren Häufigkeiten (bis 500).

6.2.3 Experimentdesign

Im Zentrum der Experimente mit den Frequentie1550-Daten steht der Test, ob die entwickelten Algorithmen auf reale Bewegungsdaten anwendbar sind.

Test der Algorithmen zur Detektion von *Konfrontationen*

Für die Evaluation der beiden Algorithmen KD_i und KD_r konnten die semantischen Informationen der während der Durchführung stattgefundenen Events beigezogen werden. Auf der Website des Frequentie1550-Projekts test.frequentie1550.nl/results wurden während der zehn Spieltage alle wichtigen Events (z.B. Ort und Zeitpunkt der durch die Teams gelösten Rätsel sowie die aktuellen Punktestände) aufgeschaltet. Weiter wurden alle *Konfrontationen* zwischen den verschiedenen Schülergruppen auf der Homepage nachgeführt. Dabei wurden jeweils die beiden beteiligten Teams, der Ort und der Zeitpunkt der *Konfrontation* sowie die Sieger aufgeführt. Mit Hilfe eines in Java geschriebenen HTML-Parsers konnten die relevanten Angaben zu den *Konfrontationen* von der Frequentie1550-Website extrahiert und gespeichert werden. Diese Informationen konnten in der Folge dazu verwendet werden, um die detektierten *Konfrontationen* mit den tatsächlich stattgefundenen zu vergleichen. Dafür galt es allerdings noch einige Vorbereitungsschritte durchzuführen. Aufgrund der im vorangegangenen Abschnitt diskutierten Unterbrüchen in den Bewegungsdatenreihen der Teams wurden nur diejenigen *Konfrontationen* für die Evaluation verwendet, welche auch in den Daten enthalten sind. Dies wurde anhand eines Vergleichs zwischen den Spielzeiten der Teams und den beobachteten *Konfrontationen* sichergestellt. Bei der genaueren Betrachtung der Informationen zu den *Konfrontationen* wurden für einzelne Teams fehlerhafte Angaben aufgedeckt. Dazu gehören falsche Uhrzeiten sowie Angaben zu *Konfrontationen*, welche so nicht stattgefunden haben können. Denn für einzelne Teams gab es Auflistungen von *Konfrontationen*, welche zeitgleich an unterschiedlichen Orten mit verschiedenen Konfrontationspartnern stattgefunden haben sollen. Weiter gab es *Konfrontationen*, welche innerhalb von sehr kurzer Zeit an weit entfernten Orten über die Bühne gegangen sein sollen. Solche fehlerhaften *Konfrontationen* wurden von der Untersuchung ausgeschlossen. Übrig blieben 33 *Konfrontationen* während der 10 Spieltage, mit denen die Evaluation der Algorithmen zur Detektion von *Konfrontationen* durchgeführt werden konnte. Mit Hilfe einer zusätzlichen Methode wurden alle detektierten *Konfrontationen* mit den beobachteten verglichen und im Falle einer zeitlichen Abweichung von weniger als 5 Minuten als korrekt detektierte Muster deklariert. Die Zeitmarge von 5 Minuten wurde gewählt, da sich sowohl die Bewegungsdaten als auch die Zusatzangaben zu den *Konfrontationen* der Frequentie1550-Website aufgrund der gemachten Ausführungen doch als stark fehlerhaft herausstellten. Anhand des Vergleichs zwischen detektierten und beobachteten *Konfrontationen* konnten die idealen Parameter herausgefunden werden, sowie die räumliche und die individuelle Variante des Algorithmus miteinander verglichen werden.

Test der Algorithmen zur Detektion von *Verfolgen/Entfliehen*

Für den Test der Algorithmen zur Detektion von *Verfolgen/Entfliehen* musste aufgrund von fehlenden semantischen Informationen auf stichprobenartige Untersuchungen ausgewichen werden. Mit Hilfe

der in Repast Symphony animierten Bewegungsdaten konnten einige qualitative Überprüfungen der Plausibilität der detektierten Muster durchgeführt werden. Zudem konnte der Einfluss einiger wichtiger Parameter, deren Werte bei diversen Experimenten variiert wurden, anhand der unterschiedlichen Anzahl von detektierten Bewegungsmustern aufgezeigt werden. Allerdings gilt es festzuhalten, dass die dabei als *Verfolgen/Entfliehen* klassierten Situationen aufgrund der fehlenden semantischen Informationen nicht differenziert werden konnten und somit keine Gütekriterien berechnet werden konnten.

Test der Algorithmen zur Detektion von *Meiden*

Auch für den Test der Algorithmen zur Detektion von *Meiden* konnten aufgrund von fehlenden semantischen Informationen nicht die gleichen systematischen Untersuchungen durchgeführt werden wie im Falle der Algorithmen zur Detektion von *Konfrontationen*. Um einen Eindruck zu erhalten, wie erfolgreich die Algorithmen in der Lage sind das Reaktionsbewegungsmuster *Meiden* in realen Bewegungsdaten zu detektieren, kann mit Hilfe der semantischen Informationen der *Konfrontationen* allerdings trotzdem ein erster Test durchgeführt werden. Und zwar dürfen während der beobachteten Konfrontationen keine Situationen mit *Meiden* zwischen denselben Teams stattfinden. Durch diesen Test können zwar keine exakten Aussagen über die Güte der Algorithmen gemacht werden, jedoch kann abgeschätzt werden, wie viele der Bewegungsmuster fehlerhaft detektiert wurden (Error of Commission), falls sie sich gleichzeitig mit einem beobachteten *Konfrontationsereignis* abspielen. Des Weiteren werden wie im Falle der Algorithmen zur Detektion von *Verfolgen/Entfliehen* stichprobenartige Untersuchungen mit Hilfe der in Repast Symphony animierten Bewegungen der Teams durchgeführt und die auf dem Display gesehenen Bewegungsmuster mit den durch die Algorithmen detektierten Situationen mit *Meiden* zwischen zwei Teams verglichen.

7 Resultate

In diesem Kapitel werden die Resultate der im letzten Kapitel beschriebenen Experimente aufgeführt und beschrieben. Im ersten Teil des Kapitels folgen die Erläuterungen zu den Experimenten mit den simulierten Bewegungsdaten aus dem Steering Behavior-Modell bevor im zweiten Teil die Resultate der Experimente mit den Frequentie1550-Daten beschrieben werden. An dieser Stelle soll noch einmal darauf hingewiesen werden, dass die Distanzangaben der simulierten Daten aus dem Steering Behavior-Modell keine Masseinheit besitzen. Die Bewegungen spielen sich in einem Raum mit einer Ausdehnung von 640 x 480 Raumeinheiten ab und die Punktobjekte bewegen sich im Durchschnitt pro Zeiteinheit (ebenfalls ohne Masseinheit) um zwei Raumeinheiten, was 0.3% der Raumbreite entspricht. Auch in diesem Kapitel wird darauf verzichtet, die Parameterwerte jeweils im Verhältnis zur Grösse des euklidischen bzw. Netzwerkraums anzugeben.

7.1 Simulierten Bewegungsdaten

7.1.1 Algorithmus VED_i

Die simulierten Bewegungsdaten für den Test des Algorithmus zur Detektion von *Verfolgen/Entfliehen* aus der individuellen Perspektive (VED_i) enthalten 35 verschiedene, zeitlich voneinander getrennte Situationen mit *Verfolgen/Entfliehen* zwischen zwei Punktobjekten (Verfolger und Fluchtobjekt) sowie zufällige Bewegungen der restlichen sechs Punktobjekte. Das Ziel der Experimente war es, mit Hilfe von idealen Werten für die Parameter des Algorithmus VED_i eine möglichst grosse Anzahl dieser 35 Verfolgungssituationen zu detektieren und somit die Anzahl der nicht gefunden Bewegungsmuster (Error of Omission, Fehler 2.Art) zu minimieren, ohne dabei allzu viele Fehldetektionen (Error of Commission, Fehler 1.Art) in Kauf nehmen zu müssen. Das Experimentieren mit verschiedenen Werten für die Parameter des Algorithmus VED_i ergaben die in der nachfolgenden Tabelle 7.1 aufgelisteten idealen Parameterwerte und die Gütemasse, deren Bedeutung und Berechnungsweise in der Folge kurz erläutert werden. Das Gütekriterium Producer's Accuracy zeigt, wie viele Bewegungsmuster aller in den Referenzdaten enthaltenen Muster vom Algorithmus richtig detektiert wurden (in Prozent, Siehe Gleichung 1). Dabei werden die fehlerhaft gefundenen Muster nicht berücksichtigt. Die Anzahl der detektierbaren Bewegungsmuster entspricht in diesem Experiment immer der Anzahl der Verfolgungssituationen im Datensatz von 35.

$$\text{Producer's Accuracy} = \frac{\text{richtig detektierte Bewegungsmuster}}{\text{total detektierbare Bewegungsmuster in Referenzdaten}} \cdot 100 \quad (1)$$

Die fehlerhaft detektierten Bewegungsmuster spielen im Gegensatz dazu im Falle der User's Accuracy eine wichtige Rolle, denn dieses Gütemass zeigt die Wahrscheinlichkeit an, dass es sich bei einem detektierten Bewegungsmuster auch tatsächlich um ein richtig detektiertes handelt. Berechnet wird die User's Accuracy mit Hilfe der Gleichung 2.

$$User's Accuracy = \frac{\text{richtig detektierte Bewegungsmuster}}{\text{total detektierte Bewegungsmuster}} \cdot 100 \quad (2)$$

Das letzte verwendete Gütemass ist die Inclass Accuracy, welche sowohl die EOC (Error of Commission) als auch die EOO (Error of Omission) in einem Schritt berücksichtigt. Die Inclass Accuracy wird mit Hilfe der Gleichung 3 berechnet. Ein hoher Wert weist dabei auf einen optimalen Kompromiss zwischen EOC und EOO hin.

$$Inclass Accuracy = \frac{\text{richtig detektierte Bewegungsmuster}}{\text{zuwenig} + \text{zuviel detektierte Bewegungsmuster}} \quad (3)$$

An dieser Stelle wird noch darauf hingewiesen, dass in den Tabellen für die EOC und EOO jeweils in Klammern noch die normalisierten Werte angegeben sind. Die Normalisierung erfolgt dabei mit Hilfe einer Division durch die Anzahl der detektierbaren Bewegungsmuster aus den Referenzdaten. In den Diagrammen ist jeweils der Verlauf der normalisierten EOC und EOO abgebildet. In der folgenden Tabelle 7.1 sind nun die idealen Parameterwerte sowie die Gütemasse für den Algorithmus VED_i aufgelistet.

kritische Distanz <i>Zusammensein</i>	75	Anzahl richtig detektierte Muster	34
kritische Zeitdauer <i>Zusammensein</i>	75	Anzahl detektierbare Muster	35
Toleranz <i>Zusammensein</i>	0	Producer's Accuracy	97.1
kritische Distanz <i>Folgen</i>	75	User's Accuracy	94.4
kritische Zeitdauer <i>Folgen</i>	65	Error of Omission	1 (0.03)
Toleranz <i>Folgen</i>	20	Error of Commission	2 (0.06)
kritischer <i>Frontbereichswinkel</i> [Grad]	55	Inclass Accuracy	11.3
kritische Verzögerungszeit	40		

Tabelle 7.1: Auflistung der idealen Parameter des Algorithmus VED_i

Übersicht der ermittelten idealen Parameterwerte des Algorithmus VED_i (link Spalte) sowie der Gütemasse aus dem Experiment mit den simulierten Bewegungsdaten unter Verwendung der idealen Werte (rechte Spalte).

Aufgrund der gewählten Aktivdistanz von *Verfolgen* im Steering Behavior-Modell von 75 erstaunt die ideale kritische Distanz von 75 für das *Zusammensein* zwischen zwei Punktobjekten sowie der optimale Wert für die kritische Distanz von 75 für das Bewegungsmuster *Folgen* wenig. Im Falle von realen Bewegungsdaten müssten die Werte dieser beiden Parameter ebenfalls durch Experimentieren ermittelt oder anhand von Expertenbefragungen (z.B. Biologen) bestimmt werden. Mit Hilfe der Zeitdauer des *Zusammenseins* von 75 sowie der Zeitdauer von *Folgen* von 65 Zeitschritten, ergab sich eine ideale Unterscheidung zwischen Mustern, welche als Folge von zufälligen Bewegungen in die gleiche Richtung entstehen und dem Reaktionsbewegungsmuster *Verfolgen/Entfliehen*. Für den Toleranzparameter von *Folgen* ergaben die Experimente im Falle eines Werts von 20 ein ideales Gleichgewicht zwischen EOO und EOC.

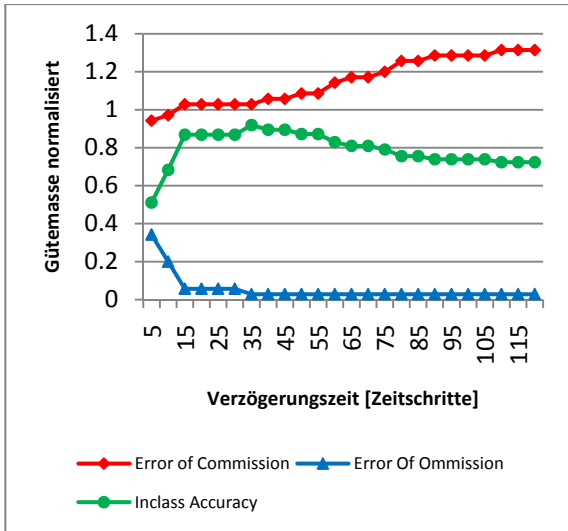


Diagramm 7.1: Einfluss der Verzögerungszeit auf den Algorithmus VED_i

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierender Verzögerungszeit zwischen den Bewegungsmustern *Zusammentreffen* und *Folgen*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.1 gehalten. Ausnahme bildet einzig der Wert von 120 Grad für den Parameter kritischer Frontbereichswinkel, welcher absichtlich höher gewählt wurde, um den Einfluss der Verzögerungszeit zu verdeutlichen.

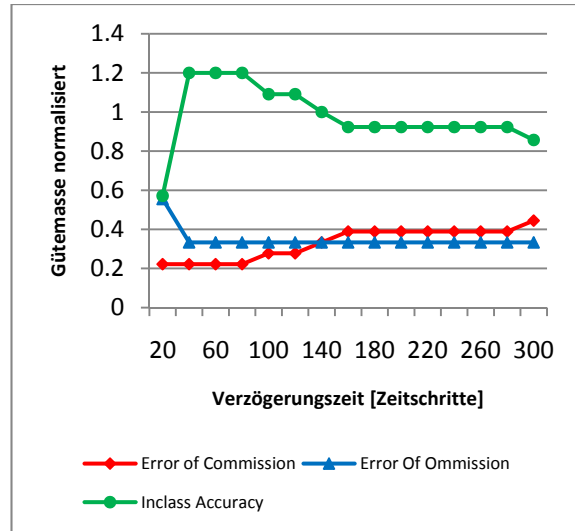


Diagramm 7.3: Einfluss der Verzögerungszeit auf den Algorithmus VED_r

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy bei variierender Verzögerungszeit. Alle anderen Parameter wurden konstant gehalten (Werte der Tabelle 7.2).

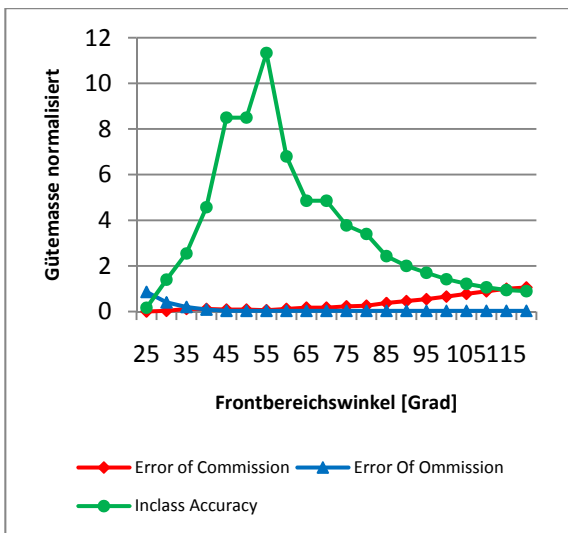


Diagramm 7.2: Einfluss des Frontbereichswinkels auf den Algorithmus VED_i

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy bei variierendem Frontbereichswinkel des Bewegungsmusters *Folgen*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.1 gehalten.

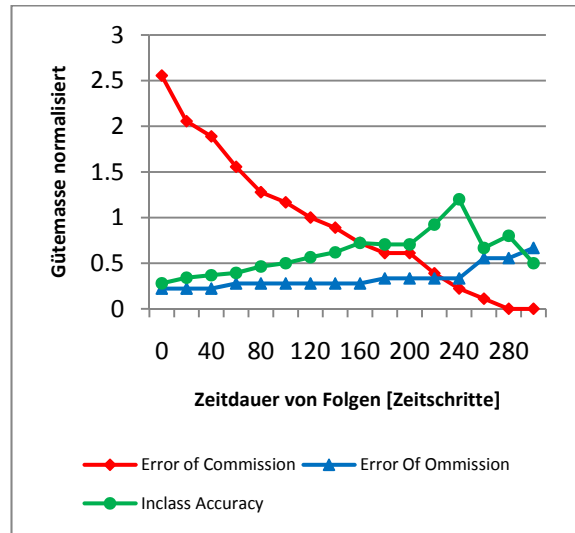


Diagramm 7.4: Einfluss der kritischen Zeitdauer von Folgen auf den Algorithmus VED_r

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy bei variierender kritischer Zeitdauer des Bewegungsmusters *Folgen*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.2 gehalten, mit Ausnahme der kritischen Zeitdauer des *Zusammenseins* von 120, welcher bewusst tief gehalten wurde, um die grosse Abhängigkeit vom Parameter Zeitdauer von *Folgen* zu verdeutlichen.

Um den Wert der EOC zu reduzieren, war eine gute Abgrenzung zwischen Situationen mit *Verfolgen/Entfliehen* und solchen, bei welchen sich Punktobjekte rein zufällig in dieselbe Richtung bewegen, notwendig. Entscheidend für diese Abgrenzung waren die Herleitung des kritischen Winkels des Frontbereichs sowie die Bestimmung der optimalen Verzögerungszeit. Im Falle der Verzögerungszeit zwischen dem Start des *Zusammentreffens* und dem Beginn des *Folgens* wird aus dem Diagramm 7.1 deutlich, dass mit Hilfe einer Reduktion der erlaubten Zeitverzögerung die Anzahl der fehlerhaft detektierten Bewegungsmuster stark reduziert werden kann. Während bei einer kritischen Verzögerungszeit von 120 noch ein Wert für die EOC von 1.31 (46 fälschlicherweise detektierte Muster) registriert wird, ist es bei einer zeitlichen Verzögerung von 5 Zeitschritten nur noch ein Wert von 1.03 (36 Error of Commission). Allerdings steigen währenddessen auch die nicht gefundenen Bewegungsmuster. Eine ideale Balance zwischen EOC und EOO, welche sich im Mass Inclass Accuracy widerspiegelt, wird bei einer Verzögerungszeit im Bereich von 40 Zeitschritten erreicht. In diesem Wertebereich steht ein Wert der EOC von 1.03 einem sehr tiefen EOO-Wert von 0.03 gegenüber. Weiter gilt es zu beachten, dass auch bei einer Reduktion der Verzögerungszeit auf 15 Zeitschritte der Wert der EOO nur auf 0.06 ansteigt und der EOC-Wert weiter gesenkt werden kann.

Diese Ausführungen zeigen, dass mit Hilfe der Verzögerungszeit Situationen mit *Verfolgen/Entfliehen* gegenüber Konstellationen, in denen sich Punktobjekte zufällig über einen längeren Zeitraum folgen, relativ gut abgegrenzt werden können. Der Wert der EOC von 1.03 bei einer Verzögerungszeit im Bereich von 40 Zeiteinheiten zeigt jedoch auf, dass die Verzögerungszeit für eine erfolgreiche Abgrenzung zu anderen Bewegungsmustern alleine nicht ausreicht. Vielmehr muss mit Hilfe des kritischen Winkels des Frontbereichs des Verfolgers versucht werden, die fehlerhaft detektierten Muster noch weiter einzugrenzen. Dies gelingt, wie das Diagramm 7.2 gut aufzuzeigen vermag, durch eine Verkleinerung des Frontbereichswinkels auf 55 Grad. Dadurch wird verhindert, dass parallel zueinander laufende Punktobjekte als Verfolger und Fluchtobjekt klassiert werden. Die Punktobjekte müssen sich also hintereinander bewegen.

7.1.2 Algorithmus VED_r

Der Datensatz, welcher für den Test des Algorithmus zur Detektion von *Verfolgen/Entfliehen* aus der räumlichen Perspektive VED_r generiert wurde, enthält wie bereits im letzten Kapitel eingeführt die Bewegungsdaten von 18 zeitlich verschiedenen Situationen mit *Verfolgen/Entfliehen*. Auf der Suche nach den idealen Parametern für den Algorithmus VED_r wurde dieser Datensatz analysiert und die Bewegungsmuster, welche durch diesen Algorithmus detektiert wurden, mit den Informationen dieser 18 Fällen von *Verfolgen/Entfliehen* verglichen. Die Resultate der Herleitung der idealen Parameter sind der nachfolgenden Tabelle 7.2 zu entnehmen. Anstelle der kritischen Distanz zwischen den Punktobjekten galt es für die räumliche Perspektive eine optimale Grösse der Nachbarschaft zu finden. Durch Experimentieren mit verschiedenen Werten wurde sowohl für das *Zusammentreffen*, wie auch für das *Folgen* die Nachbarschaftsordnung 1 als ideale Grösse definiert. Diese bedeutet, dass sich die

an einer Verfolgung beteiligten Punktobjekte auf direkt benachbarten Segmenten aufhalten müssen, damit vom Reaktionsbewegungsmuster *Verfolgen/Entfliehen* gesprochen werden kann.

Nachbarschaftsgrösse <i>Zusammensein</i>	1	Anzahl richtig detektierte Muster	12
kritische Zeitdauer <i>Zusammensein</i>	250	Anzahl detektierbare Muster	18
Toleranz <i>Zusammensein</i>	14	Producer's Accuracy	66.6
Nachbarschaftsgrösse <i>Folgen</i>	1	User's Accuracy	75.0
kritische Zeitdauer <i>Folgen</i>	240	Error of Omission	6 (0.33)
Toleranz <i>Folgen</i>	35	Error of Commission	4 (0.22)
kritische Verzögerungszeit	35	Inclass Accuracy	1.2

Tabelle 7.2: Auflistung der idealen Parameter des Algorithmus VED_r

Übersicht der ermittelten idealen Parameterwerte des Algorithmus VED_r (link Spalte) sowie der Gütemasse aus dem Experiment mit den simulierten Bewegungsdaten unter Verwendung der idealen Werte (rechte Spalte).

Im Vergleich zur individuellen Perspektive fielen zudem die Werte für die kritischen Zeitspannen der beiden Muster *Zusammentreffen* und *Folgen* wesentlich grösser aus. Dies ist darauf zurückzuführen, dass die Bewegungsfreiheit der vier sich zufällig bewegenden Punktobjekte durch das Netzwerk doch stark eingeschränkt war und es somit häufig zu ungewolltem und somit zufälligem Folgen dieser Punktobjekte kam. Dieser Umstand wird noch verstärkt, da es den Punktobjekten in den Simulationen des Steering Behavior-Modells nicht möglich ist anzuhalten. Um diese eigentlich bedeutungslosen Muster von Bewegungen in dieselbe Richtung von wahren Events mit *Verfolgen/Entfliehen* abgrenzen zu können, mussten also sehr hohe Werte für die kritischen Zeitspannen gewählt werden. Dieser Einfluss der kritischen Zeitdauer ist aus den Resultaten eines Experiments, bei welchem die kritische Zeitdauer von *Folgen* über den Wertebereich von 0 bis 300 variiert wurde, gut abzulesen. Die Resultate sind im Diagramm 7.4 dargestellt. Der Maximalwert der Inclass Accuracy wurde bei einer kritischen Zeitdauer von *Folgen* von 240 Zeiteinheiten erreicht.

Die Verzögerungszeit zeigte sich bei den Experimenten als wichtiger Parameter, wie im Falle der individuellen Perspektive, jedoch nicht als der entscheidende. Diese Aussage wird untermauert durch die im Diagramm 7.2 dargestellten Resultate eines weiteren Experiments, bei welchem die Verzögerungszeit über einen Wertebereich von 0 bis 300 Zeitschritte hinweg variiert wurde. Mit steigender Grösse der Verzögerungszeit nimmt auch der Wert der EOC zu, wobei die Zunahme allerdings nicht so gross ist wie im Falle einer Reduktion der kritischen Zeitdauer des Bewegungsmusters *Folgen*. Die Diskussion dieses Zusammenhangs folgt in Kapitel 8.

7.1.3 Algorithmus MD_i

Das Experimentieren mit den verschiedenen Parameterwerten des Algorithmus zur Detektion von *Meiden* (individuelle Perspektive) ergab die in der Tabelle 7.3 dargestellten idealen Werte, mit welchen das beste Resultat erzielt werden konnte. Die kritischen Distanzen für die Bewegungsmuster *Zusammentreffen* und *Getrenntsein* wurden von der im Steering Behavior-Modell eingestellten Aktivdistanz der *Separation* von 30 Raumeinheiten abgeleitet. Durch Experimentieren erwies sich die

kritische Distanz von 35 als idealer Wert. Durch diese Experimente konnten auch die kritischen Zeitdauern und die Toleranzen der Muster *Zusammentreffen* und *Getrenntsein* bestimmt werden.

kritische Distanz <i>Zusammensein</i>	35	Anzahl richtig detektierte Muster	32
kritische Zeitdauer <i>Zusammensein</i>	6	Anzahl detektierbare Muster	33
Toleranz <i>Zusammensein</i>	0	Producer's Accuracy	97.0
kritische Zeitdauer ohne <i>Zusammensein</i>	60	User's Accuracy	97.0
kritische Distanz <i>Getrenntsein</i>	35	Error of Omission	1 (0.03)
kritische Zeitdauer <i>Getrenntsein</i>	15	Error of Commission	1 (0.03)
Toleranz <i>Getrenntsein</i>	0	Inclass Accuracy	16.0
kritischer Richtungswechselwinkel [Grad]	19		
kritische Distanz <i>Meiden</i>	19		
kritische Verzögerungszeit	45		

Tabelle 7.3: Auflistung der idealen Parameter des Algorithmus MD_i

Übersicht der ermittelten idealen Parameterwerte des Algorithmus MD_i (link Spalte) sowie der Gütemasse aus dem Experiment mit den simulierten Bewegungsdaten unter Verwendung der idealen Werte (rechte Spalte).

Weiter wurde die erlaubte Minimaldistanz zwischen den Punktobjekten während des Reaktionsbewegungsmusters *Meiden* durch Experimentieren ausfindig gemacht. Mit Hilfe dieses Parameters kann verhindert werden, dass Ereignisse, bei denen sich Punktobjekte sehr nahe kommen und sich daher unmöglich meiden können, als *Meiden* bezeichnet werden. Für die Beurteilung der Güte des Algorithmus zur Detektion von *Meiden* sind die beiden Parameter Bewegungsrichtungswechsel sowie die Verzögerungszeit, welche in der Folge genauer unter die Lupe genommen werden, speziell interessant. Der Parameter Verzögerungszeit war im Falle des Reaktionsbewegungsmusters *Verfolgen/Entfliehen* entscheidend für das Abschneiden der Algorithmen bei den Tests. Für den Algorithmus zur Detektion von *Meiden* stellte sich die Verzögerungszeit während den Experimenten allerdings als weit weniger wichtig heraus, als dies noch bei *Verfolgen/Entfliehen* der Fall war. Das Diagramm 7.5 zeigt diesen Sachverhalt anhand der geringen Variabilität des Wertes der EOC relativ deutlich. Trotz starker Reduktion der Verzögerungszeit gelingt es nicht, die EOC weiter zu verringern, ohne dabei eine starke Zunahme der EOO in Kauf nehmen zu müssen. Die Ursachen für den geringen Einfluss der Verzögerungszeit in den Experimenten mit den simulierten Bewegungsdaten werden in Kapitel 8 diskutiert. Weitaus wichtiger ist der Parameter Bewegungsrichtungsänderung. Während sich die Punktobjekte bei einem zufälligen Aufeinandertreffen in ihren Bewegungen nicht beeinflussen, kommt es im Falle von *Meiden* zu einer Separationsbewegung, sobald sich die Punktobjekte zu nahe kommen. Durch diese *Separation* kommt es zu einer abrupten Änderung der Fortbewegungsrichtung, mit Hilfe derer das Reaktionsbewegungsmuster *Meiden* von Mustern von zufälligen Ereignissen unterschieden werden kann. Dieser Sachverhalt ist aus dem Diagramm 7.6 gut abzulesen. Je höher die kritische Bewegungsrichtungsänderung, desto geringer ist der Wert der EOC. Dabei ist die Reduktion zwischen 10 und 20 Grad besonders effektiv. Das ideale Gleichgewicht zwischen EOC und EOO und somit ein maximaler Wert der Inclass Accuracy kann bei 19 Grad erreicht werden.

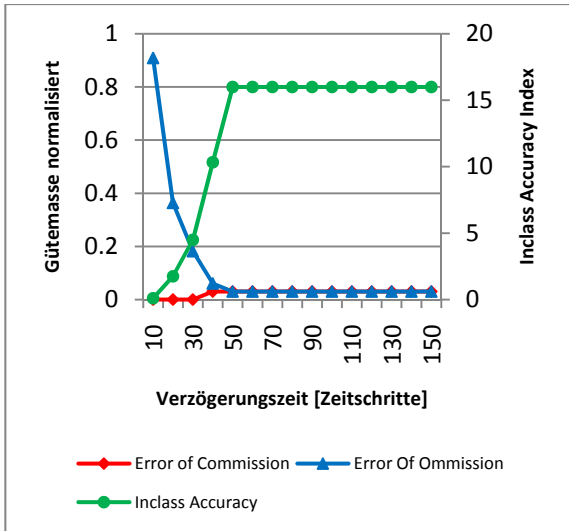


Diagramm 7.5: Einfluss der Verzögerungszeit auf den Algorithmus MD_i

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierender Verzögerungszeit zwischen den Bewegungsmustern *Zusammentreffen* und *Getrenntsein*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.3 gehalten.

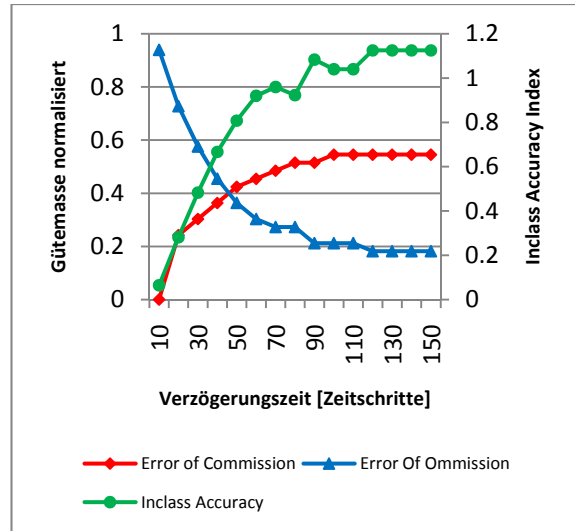


Diagramm 7.7: Einfluss der Verzögerungszeit auf den Algorithmus MD_r

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierender Verzögerungszeit zwischen den Bewegungsmustern *Zusammentreffen* und *Getrenntsein*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.4 gehalten.

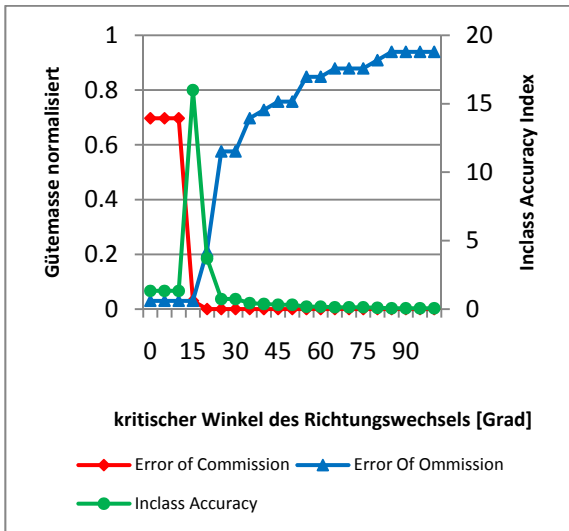


Diagramm 7.6: Einfluss des kritischen Richtungswechselwinkels auf den Algorithmus MD_i

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierendem kritischem Richtungswechselwinkel. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.3 gehalten.

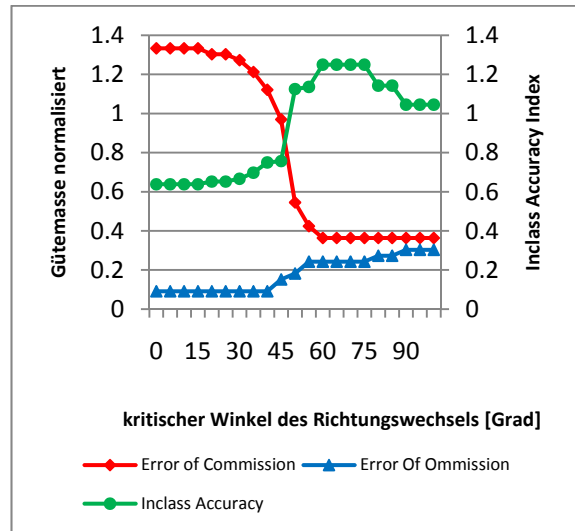


Diagramm 7.8: Einfluss des kritischen Richtungswechselwinkels auf den Algorithmus MD_r

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierendem kritischem Richtungswechselwinkel. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.4 gehalten.

7.1.4 Algorithmus MD_r

Durch das Experimentieren mit den verschiedenen Parametern des Algorithmus zur Detektion von *Meiden* (räumliche Perspektive) ergaben sich die in der Tabelle 7.4 aufgeführten idealen Parameterwerte, mit welchen das beste Ergebnis erzielt werden konnte. Die beiden Parameter der Segmentnachbarschaft (Nachbarschaftsgrösse *Zusammensein*, Nachbarschaftsgrösse *Getrenntsein*) wurden mit 0 gewählt, was bedeutet, dass nur Bewegungen auf dem gleichen Segment unter die Lupe genommen werden. Durch Experimentieren mit diesem Parameter stellte sich eine Nachbarschaft von 0 als bester Wert heraus, da dadurch vor allem die EOC stark reduziert werden konnten. Im Falle einer Nachbarschaftsgrösse von 1 oder 2 werden durch den Algorithmus MD_r viel mehr Situationen fälschlicherweise als *Meiden* deklariert. Eine Nachbarschaft von 0 macht durchaus Sinn, führt man sich die Aktivdistanz des Verhaltens *Separation* im Steering Behavior-Modell von 30 Raumeinheiten, sowie die durchschnittliche Segmentlänge des Strassennetzwerks von 119.1 Einheiten vor Augen. Dies bedeutet, dass das Verhalten *Separation* nur aktiv sein kann, falls sich zwei Punktobjekte auf dem gleichen Strassensegment aufhalten.

Nachbarschaftsgrösse <i>Zusammensein</i>	0	Anzahl richtig detektierte Muster	27
kritische Zeitdauer <i>Zusammensein</i>	9	Anzahl detektierbare Muster	33
Toleranz <i>Zusammensein</i>	0	Producer's Accuracy	81.8
kritische Zeitdauer ohne <i>Zusammensein</i>	60	User's Accuracy	60.0
Nachbarschaftsgrösse <i>Getrenntsein</i>	0	Error of Omission	6 (0.18)
kritische Zeitdauer <i>Getrenntsein</i>	13	Error of Commission	18 (0.54)
Toleranz <i>Getrenntsein</i>	0	Inclass Accuracy	1.1
kritischer Richtungswechselwinkel [Grad]	50		
kritische Verzögerungszeit	120		

Tabelle 7.4: Auflistung der idealen Parameter des Algorithmus MD_r.

Übersicht der ermittelten idealen Parameterwerte des Algorithmus MD_r (link Spalte) sowie der Gütemasse aus dem Experiment mit den simulierten Bewegungsdaten unter Verwendung der idealen Werte (rechte Spalte).

Mit Hilfe weiterer Experimente konnten auch die kritischen Zeitdauern der Bewegungsmuster *Zusammensein* und *Getrenntsein* ausfindig gemacht werden. Interessant scheinen jedoch vor allem die beiden Parameter Verzögerungszeit und Richtungswechselwinkel. Der Einfluss des Parameters Verzögerungszeit soll durch das Diagramm 7.7 veranschaulicht werden. Aus diesem Diagramm geht hervor, dass es anhand der Verringerung der zugelassenen Verzögerungszeit zwischen dem Beginn des *Zusammenseins* und dem *Getrenntsein* nicht möglich ist, den Wert der EOC zu reduzieren ohne gleichzeitig den EOO-Wert zu erhöhen. Dies widerspiegelt sich in den parallel verlaufenden Linien der EOC (rot) und der korrekt detektierten Bewegungsmustern (dunkelgrün). Somit gelingt, wie beim Algorithmus MD_i gesehen, die Abgrenzung zwischen Mustern von zufälligem Aneinandervorbeigehen und dem Bewegungsmuster *Meiden* mit Hilfe des Parameters Verzögerungszeit nur ungenügend. Als Folge davon spielt die Verzögerungszeit nur eine untergeordnete Rolle. Der Wert von 120, welcher im Vergleich zur Verzögerungszeit von 45 Zeiteinheiten des Algorithmus MD_i doch viel höher ist, ist mit

den grösseren Distanzparametern zu erklären. Während bei der individuellen Perspektive die kritische Distanz des Zusammenseins bei 35 Raumeinheiten liegt, ist die durchschnittliche Segmentlänge von 119.1 ebenfalls viel grösser. Dies führt dazu, dass es länger dauert, bis sich zwei Punktobjekte wieder voneinander entfernt haben und sich als Folge davon nicht mehr auf dem gleichen Segment aufhalten.

Von grosser Bedeutung für den Algorithmus MD_i ist, wie im Falle der individuellen Perspektive, der Parameter Richtungswechselwinkel. Das Diagramm 7.8 zeigt den grossen Einfluss dieses Parameters auf den Erfolg des Algorithmus. Durch eine Erhöhung des zugelassenen Winkels bei einem Richtungswechsel kann die Anzahl der fehlerhaft detektierten Bewegungsmuster stark reduziert werden. Der Verlauf der Kurve der EOC zeigt allerdings, dass die fehlerhaft als *Meiden* deklarierten Muster ab einem kritischen Abbiegewinkels von 60 bis auf 100 Grad nicht mehr weiter verringert werden können. In Kapitel 8 wird versucht eine Erklärung für diesen Umstand zu finden und zwar mit Hilfe der Betrachtung der verwendeten Daten.

7.2 Frequentie1550-Daten

7.2.1 Algorithmus VED_i

Wie in Kapitel 8 angesprochen, stehen für den Frequentie1550-Datensatz nur für das Reaktionsbewegungsmuster *Konfrontation* semantische Informationen zur Verfügung. Dies bedeutet, dass in den Bewegungsdaten des Frequentie1550-Projekts zwar Situationen mit *Verfolgen/Entfliehen* detektiert werden können, allerdings nicht überprüft werden kann, ob sich dieses Reaktionsbewegungsmuster auch tatsächlich abgespielt hat. Es ist lediglich möglich, stichprobenartig die detektierten Muster anhand von Animationen der Bewegungen in *Repast Symphonie* zu betrachten und so eine Aussage über die Plausibilität der gefundenen Reaktionsbewegungsmuster *Verfolgen/Entfliehen* zu machen. Zudem kann die unterschiedliche Anzahl der detektierten Muster bei Variieren von interessanten Parametern miteinander verglichen werden. Als erstes untersucht wurde der Einfluss des Parameters Frontbereichswinkel auf die Anzahl der detektierten Events mit *Verfolgen/Entfliehen*. In den Experimenten mit den simulierten Bewegungsdaten zeigte sich der Frontbereichswinkel als wichtiger Parameter. Speziell für die Reduktion der EOC war dieser Parameter der entscheidende. Die Wichtigkeit des Frontbereichswinkels für den Algorithmus VED_i wurde auch für den Anwendungsfall der Frequentie1550-Daten überprüft. Die Resultate dieses Experiments sind im Diagramm 7.11 dargestellt. Daraus geht hervor, dass der Frontbereichswinkel auch bei der Analyse des Frequentie1550-Datensatz eine wichtige Rolle spielt. Bei Erhöhung des Winkels steigt die Anzahl der detektierten Bewegungsmuster von *Verfolgen/Entfliehen* stetig an und erreicht bei einem Winkel von 140 Grad mit einer Anzahl von 18 das Maximum. Die Abnahme der Anzahl der detektierten Muster zwischen 140 Grad und 180 Grad ist darauf zurückzuführen, dass sich bei sehr grossen Frontbereichswinkeln mehr Punktobjekte gegenseitig im Blickfeld haben. Eine solche Situation wird vom Algorithmus als gegenseitiges Folgen interpretiert, was per Definition kein *Verfolgen/Entfliehen* sein kann.

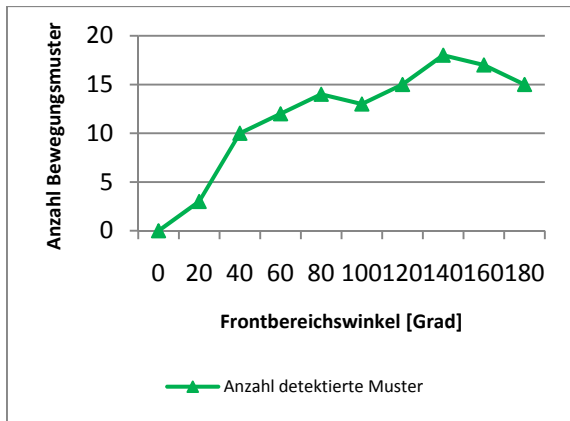


Diagramm 7.11: Einfluss der Frontbereichswinkels auf den Algorithmus VED_i

Verlauf der Anzahl detektierter Bewegungsmuster bei variierendem Frontbereichswinkel von *Folgen*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.1 gehalten. Ausnahme bildet einzig der Wert von 35 für den Toleranzparameter des Bewegungsmusters *Folgen*. Dieser Parameter musste erhöht werden, da in den trotz Vorprozessierung noch immer stark sprunghaften Daten ansonsten beinahe keine Bewegungsmuster *Verfolgen/Entfliehen* hätten detektiert werden können.

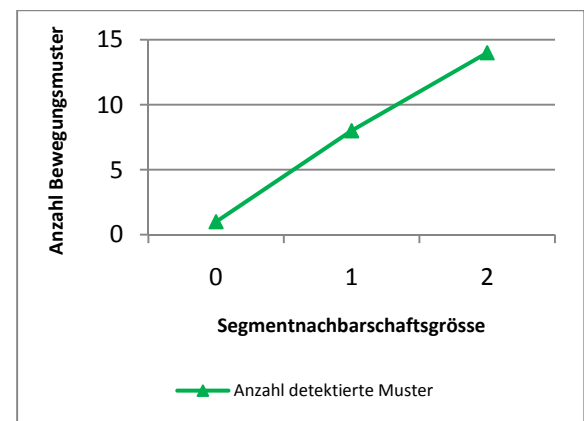
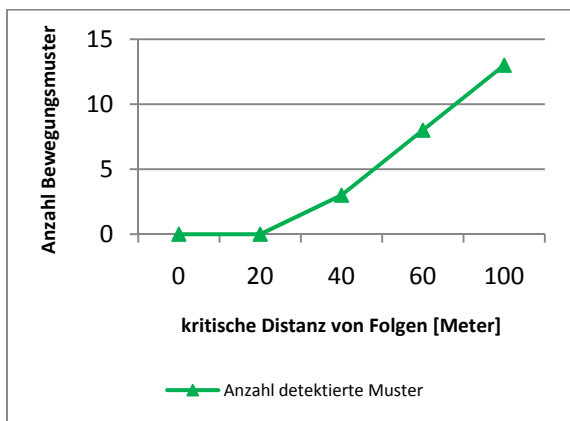


Diagramm 7.9: Einfluss der kritischen Distanz von Folgen auf den Algorithmus VED_i

Verlauf der Anzahl detektierter Bewegungsmuster bei variierender kritischer Distanz von *Folgen*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.1 gehalten. Ausnahme bildet einzig der Wert von 35 für den Toleranzparameter von *Folgen*, welcher aufgrund der schlechten Datenqualität höher gewählt wurde.

Diagramm 7.12: Einfluss der Grösse der betrachteten Segmentnachbarschaft auf den Algorithmus VED_i

Verlauf der Anzahl detektierter Bewegungsmuster bei variierender Grösse der Segmentnachbarschaft. Alle anderen Parameter wurden konstant gehalten (kritische Zeitdauer *Zusammensein* = 75 sec, Toleranz *Zusammensein* = 0, kritische Zeitdauer *Folgen* = 65 sec, Toleranz *Folgen* = 35, kritische Verzögerungszeit = 35 sec).

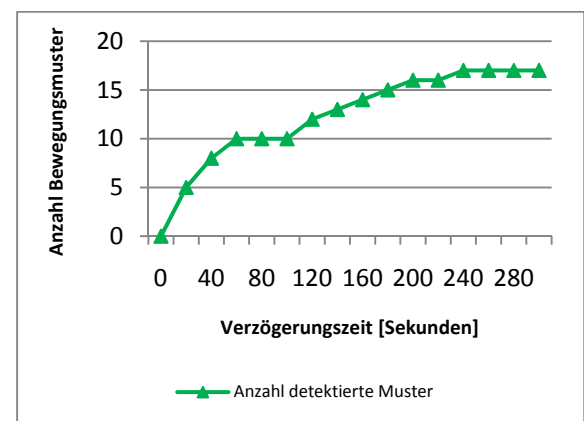
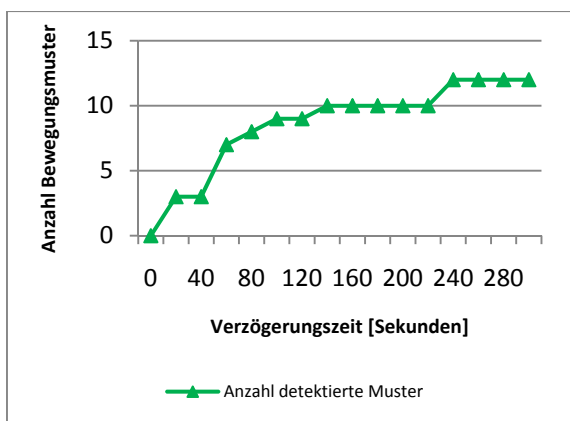


Diagramm 7.10: Einfluss der Verzögerungszeit auf den Algorithmus VED_i

Verlauf der Anzahl detektierter Bewegungsmuster bei variierender kritischer Distanz von *Folgen*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.1 gehalten. Ausnahme bilden einzig der Wert von 35 für den Toleranzparameter von *Folgen* sowie der kritische Frontbereichswinkel mit einem Wert von 20 Grad.

Diagramm 7.13: Einfluss der Verzögerungszeit auf den Algorithmus VED_r

Verlauf der Anzahl detektierter Bewegungsmuster bei variierender Grösse der Segmentnachbarschaft. Alle anderen Parameter wurden konstant gehalten (Nachbarschaftsgröße *Zusammensein/Folgen* = 1, kritische Zeitdauer *Zusammensein* = 75 sec, Toleranz *Zusammensein* = 0, kritische Zeitdauer *Folgen* = 65 sec, Toleranz *Folgen* = 35).

Der Anstieg von drei detektierten Mustern bei einem Winkel von 20 Grad bis auf eine Anzahl von 18 bei einem Winkel von 140 Grad zeigt, dass der Einfluss dieses Parameters auch im Falle der Frequentie1550-Daten nicht zu unterschätzen ist. Speziell bei einer Erhöhung des Frontbereichswinkels von 20 auf 40 Grad steigt die Anzahl der detektierten Muster sprunghaft an. Der Einfluss dieser Zunahme des Frontbereichswinkels von lediglich 20 Grad soll durch die nachfolgenden Abbildungen verdeutlicht werden. In der ersten Abbildung 7.1 dargestellt ist eine Situation mit *Verfolgen/Entfliehen*, welche mit einem Parameterwert von 20 Grad detektiert wurde. Die beiden Punktobjekte verfolgen sich schön auf zwei geraden Strassensegmenten.

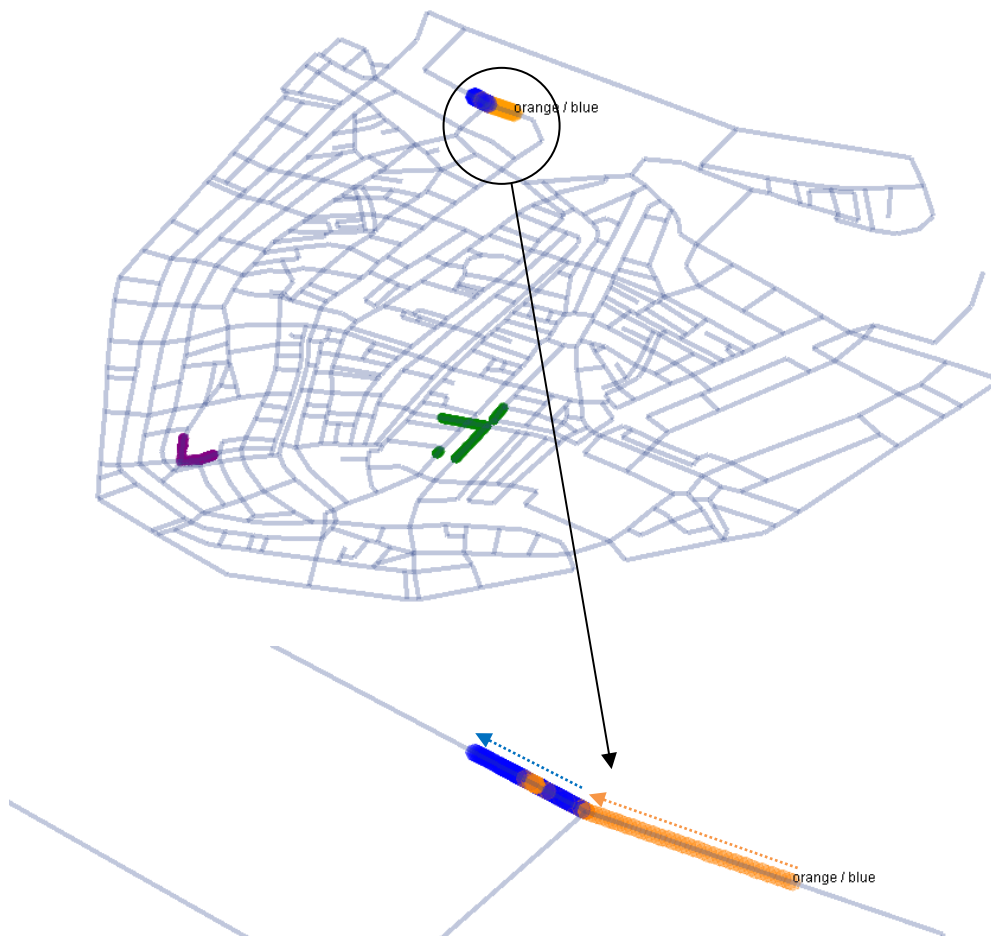


Abbildung 7.1: Positives Beispiel einer als *Verfolgen/Entfliehen* klassierten Situation

Beispiel einer Situation mit *Verfolgen/Entfliehen* zwischen den Teams orange (Verfolger) und blau (Fliehende). Auf der Übersichtsgraphik sind zudem die Bewegungen der zum Zeitpunkt der Verfolgung ebenfalls spielenden Teams violett und grün dargestellt. Dieses Beispiel zeigt auf, wie sich das orange Team hinter dem blauen her bewegt.

Mit zunehmendem Frontbereichswinkel werden allerdings auch immer mehr Situationen fehlerhaft als *Verfolgen/Entfliehen* interpretiert. Als Veranschaulichung eines solchen Falls soll die in der Abbildung 7.2 dargestellte Situation dienen, welche vom Algorithmus aufgrund des erhöhten Parameters Frontbereichswinkel von 40 als *Verfolgen/Entfliehen* interpretiert wurden, obwohl sich diese beiden Punktobjekte aufgrund der Häuser entlang der Strassen gar nicht sehen konnten. Dieses Beispiel soll veranschaulichen, dass man durch eine Erhöhung des Frontbereichswinkels Gefahr läuft, dass Situationen fälschlicherweise als *Verfolgen/Entfliehen* interpretiert werden.

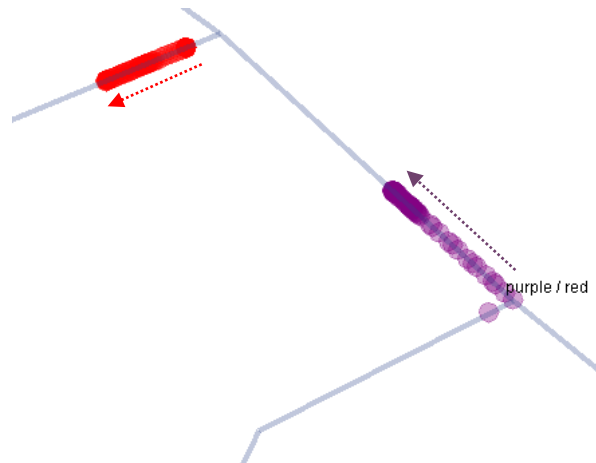


Abbildung 7.2: Negatives Beispiel einer als *Verfolgen/Entfliehen* klassierten Situation

Beispiel einer Situation mit *Verfolgen/Entfliehen* zwischen den Teams violett (Verfolger) und rot (Fliehende). Dabei bewegen sich die Teams auf unterschiedlichen Segmenten. Je nach dem, was für Sichtverhältnisse auf diesem Strassennetzwerk vorherrschen, können sich die Teams in dieser Situation gar nicht sehen.

Neben der Wahl eines kleinen Frontbereichswinkels gibt es zudem einen zweiten Ansatz, um solche Fehldetektion zu verhindern. Und zwar Mittels einer Anpassung der kritischen Distanz von *Verfolgen/Entfliehen* an die Sichtverhältnisse auf dem Strassensegment. Mit Hilfe eines Experiments mit variabler Distanz von *Folgen*, dessen Resultate im Diagramm 7.9 dargestellt sind, konnte gezeigt werden, dass bei einer Vergrößerung des Wertes dieses Parameters auch die Anzahl der detektierten Bewegungsmuster zunimmt. Durch eine Erhöhung der kritischen Distanz werden dabei auch unweigerlich die EOC zunehmen, da sich zwei weitentfernte Punktobjekt nicht mehr auf dem gleichen Segment aufhalten und sich aufgrund der Sichteinschränkungen in den engen Strassen von Amsterdam gar nicht sehen können. Bei einer sinnvollen Wahl der kritischen Distanz von *Folgen* sollte man sich also an der Länge der Strassensegmente orientieren.

Die Tests des Algorithmus zur Detektion von *Verfolgen/Entfliehen* unter Verwendung der Simulationsdaten haben gezeigt, dass auch die Verzögerungszeit ein relevanter Parameter ist. Allerdings war die Wichtigkeit dieses Parameters nicht so gross wie dies der Parameter Frontbereichswinkel war. Für die Überprüfung des Einflusses des Parameters Verzögerungszeit im Falle der Frequentie1550-Daten wurde ein Experiment durchgeführt, bei welchem einzig die Verzögerungszeit variiert wurde. Dabei kamen die im Diagramm 7.10 dargestellten Zusammenhänge zwischen der Verzögerungszeit und der Anzahl der detektierten Bewegungsmuster *Verfolgen/Entfliehen* zu Tage. Aus dieser Graphik ablesbar ist der grosse Einfluss der Verzögerungszeit auf die Anzahl der detektierten *Verfolgen/Entfliehen*. Mit zunehmender erlaubter kritischer Verzögerungszeit steigt auch die Anzahl der Detektionen, wobei bei einer Zeitverzögerung von 300 Sekunden das Maximum mit 12 detektierten Bewegungsmustern erreicht wird. Daraus lässt sich ableiten, dass der Einfluss des Parameters Verzögerungszeit höher ist, als es noch in den Experimenten mit den simulierten Bewegungsdaten der Fall war. Eine ausführliche Diskussion der Ursachen dieses Zusammenhangs sowie der Erkenntnisse zu den beiden Parametern kritische Distanz von *Folgen* und Frontbereichswinkel ist in Kapitel 8 zu finden.

7.2.2 Algorithmus VED_r

Bei der Analyse der Anwendbarkeit des Algorithmus VED_r auf den Frequentie1550-Datensatz stellten sich die mittels der Experimente mit den Simulationsdaten des Steering Behavior-Modells gefundenen idealen Parameterwerte als zu hohe Hürden heraus. Unter der Anwendung dieser Parameter wurde in den Frequentie1550-Daten kein Bewegungsmuster *Verfolgen/Entfliehen* gefunden. Dies liegt primär an den sehr hohen Werten der Zeitparameter der beiden Bewegungsmuster *Zusammensein* (250 sec) und *Folgen* (240 sec), was für den trotz Vorprozessierung immer noch stark fehlerbehafteten Datensatz eine zu hohe Anforderung darstellt. Aufgrund der vielen Vorwärts- und Rückwärtssprüngen in den Positionsangaben ist es nicht realistisch, dass sich zwei Punktoobjekte über solch grosse Zeitspannen auf benachbarten Segmenten gleichmässig in dieselbe Richtung bewegen. Ohne den Wert für den Toleranzparameter drastisch zu erhöhen, findet der Algorithmus in den Frequentie1550-Daten also kein *Verfolgen/Entfliehen*. Aus diesem Grund wurden die Parameter auf die Werte des Algorithmus VED_i (kritische Zeitdauer *Zusammensein* = 75 sec / kritische Zeitdauer *Folgen* = 65 sec) angepasst und einige Experimente durchgeführt, bei welchen einzelne Parameter variiert wurden. Dabei gilt es darauf hinzuweisen, dass an Stelle der Distanzparameter der Bewegungsmuster *Zusammensein* und *Folgen* der individuellen Perspektive nun für die räumliche Perspektive die Grösse der Segmentnachbarschaft tritt. Die Parameter der Segmentnachbarschaft für das *Zusammensein* und das *Folgen* wurden in einem Experiment über den Wertebereich von 0 bis 2 variiert und dabei der Einfluss auf die Anzahl der detektierten Bewegungsmuster untersucht. Das Resultat dieser Untersuchung ist im Diagramm 7.12 dargestellt. Wie erwartet nimmt mit zunehmender Nachbarschaftsgrösse auch die Anzahl der detektierten Bewegungsmuster zu. Die Zunahme der detektierten Bewegungsmuster birgt allerdings immer auch die Gefahr von fehlerhaft detektierten Bewegungsmustern. Aufgrund der fehlenden semantischen Informationen kann auch in diesem Fall der mit der Vergrösserung der Segmentnachbarschaft einhergehende Anstieg des Wertes der EOC nicht genau beziffert werden. Die Abbildung 7.3 zeigt ein Beispiel, in dem sich Punktoobjekte auf nicht benachbarten Segmenten in dieselbe Richtung bewegen (Segmentnachbarschaft mit Grösse 2).

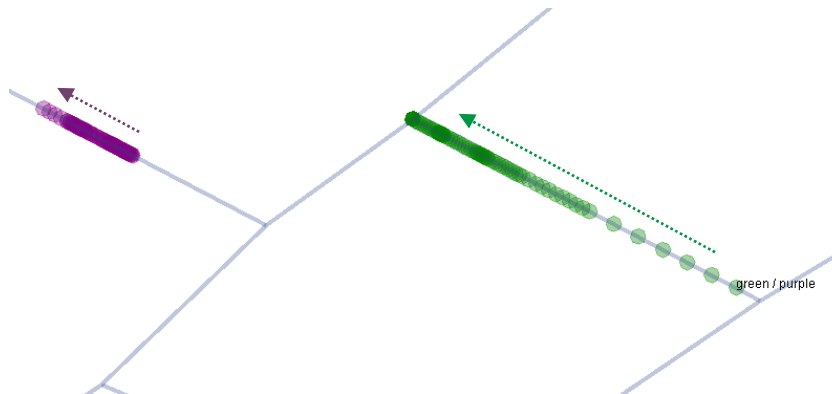


Abbildung 7.3: Beispiel einer als *Verfolgen/Entfliehen* klassierten Situation

Beispiel einer Situation mit *Verfolgen/Entfliehen* zwischen den Teams grün (Verfolger) und violett (Fliehende). Wie in der Abbildung 7.2 sind auch hier die Gegebenheiten auf dem Strassennetzwerk entscheidend, ob wirklich von einer *Verfolgungssituation* gesprochen werden kann, da sich die Punktoobjekte auf nicht benachbarten Segmenten bewegen.

Bei der Betrachtung dieses Beispiels wird klar, dass diese Situation unter Umständen durchaus als *Verfolgen/Entfliehen* klassiert werden könnte. Und zwar falls sich zwischen den Strassen keinerlei Hindernisse, welche die Sicht einschränken, befänden. Bei der Wahl der angemessenen Segmentnachbarschaftsgrösse kommt somit dem Hintergrundwissen über die Gegebenheiten und Rahmenbedingungen, welche während der Aufnahme der Bewegungsdaten vorherrschten, grosse Bedeutung zu. Im Falle der Frequenz1550-Daten macht eine Nachbarschaftsgrösse von 0 oder 1 aufgrund der starken Sichteinschränkungen durch die Häuser entlang der schmalen Gassen wohl mehr Sinn. Dies bedeutet, dass sich die Punktobjekte auf dem gleichen, beziehungsweise auf benachbarten Segmenten aufhalten müssen. Für die Nachbarschaftsgrösse von 1 sollen in der Folge exemplarisch noch zwei Situationen aufgezeigt werden, in denen *Verfolgen/Entfliehen* richtig detektiert wurden (Siehe Abbildung 7.4). Dabei sieht man gut, wie sich die Punktobjekte auf benachbarten Segmenten in die gleiche Richtung bewegen und sich aufgrund der in einer Linie verlaufenden Strassen immer im Blickfeld haben.



Abbildung 7.4: Beispiel von zwei richtigerweise detektierten *Verfolgen/Entfliehen*

Beispiel einer Situation mit *Verfolgen/Entfliehen* zwischen den Teams rot (Verfolger) und violett (Fliehende). Die beiden Teams bewegen sich aufgrund der Segmentorientierungen immer sehen.

Neben der Segmentnachbarschaftsgrösse wurden zudem noch Experimente mit einem zweiten interessanten Parameter durchgeführt, und zwar mit der Verzögerungszeit. Dabei wurde der Parameter Verzögerungszeit über einen Wertebereich von 0 bis 300 variiert, während alle anderen Parameter konstant gehalten wurden. Die Resultate dieses Experiments sind im Diagramm 7.13 zusammengefasst. Der Verlauf der Kurve zeigt, dass die Anzahl der detektierten Bewegungsmuster mit steigender Verzögerungszeit zunimmt. Der Verlauf der Kurve ist mit derjenigen der individuellen Perspektive (Diagramm 7.10) zu vergleichen. Sie zeigt auf, dass die zeitliche Verzögerung zwischen *Zusammentreffen* und *Folgen* im Vergleich zu den Experimenten mit den simulierten Bewegungsdaten an Bedeutung gewinnt. Dieser Zusammenhang wird in Kapitel 8 ausführlich diskutiert.

7.2.3 Algorithmus MD_i

Wie bei den Tests der Algorithmen zur Detektion von *Verfolgen/Entfliehen* stehen auch für die Tests der Algorithmen zur Detektion von *Meiden* keine semantischen Informationen zur Verfügung mit denen systematische Untersuchungen angestellt und daraus die idealen Parameterwerte abgeleitet werden könnten. Nichts desto trotz wurden einige interessante Parameter ausgewählt und mit diesen einige Experimente durchgeführt. Zu diesen Parametern gehört die Verzögerungszeit. Wie im vorangegangenen Kapitel beschrieben, war die Verzögerungszeit in den Experimenten mit den simulierten Bewegungsdaten nicht der einflussreichste Parameter. Um diese Aussage zu prüfen, wurde ein Experiment durchgeführt, bei welchem lediglich die Verzögerungszeit variiert wurde und alle anderen Parameter konstant gehalten wurden. Dabei wurden die detektierten Situationen mit *Meiden* mit den semantischen Informationen der *Konfrontationen* verglichen, um eine Aussage über die EOC machen zu können. Diese Untersuchung macht Sinn, da sich während dem *Meiden* zwischen zwei Punktobjekten per Definition keine *Konfrontation* zwischen denselben abspielen kann. Die Resultate dieses Experiments sind im Diagramm 7.14 dargestellt, wobei der Verlauf der Anzahl der detektierten Muster sowie der EOC bei variierender Verzögerungszeit abgebildet ist. Aus dieser Darstellung kann abgelesen werden, dass bei einer Zunahme der Verzögerungszeit sowohl die Anzahl der detektierten Bewegungsmuster, als auch der Wert der EOC ansteigt.

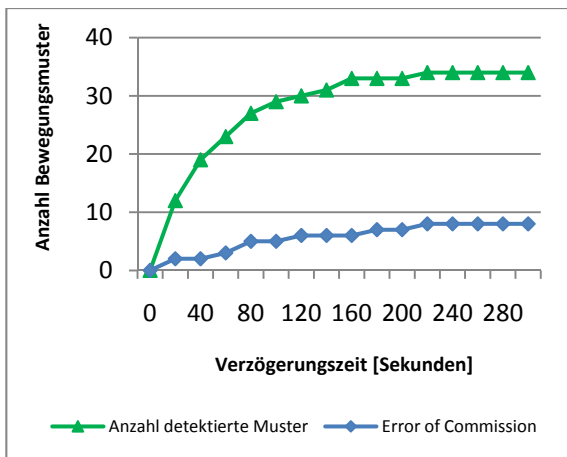


Diagramm 7.14: Einfluss der Verzögerungszeit auf den Algorithmus MD_i

Verlauf der Anzahl detektierter Reaktionsbewegungsmuster *Meiden* (grün) sowie der Error of Commission (blau) bei variierender Verzögerungszeit zwischen den Bewegungsmustern *Zusammentreffen* und *Getrenntsein*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.3 gehalten, mit Ausnahme der Distanzparameter von *Zusammensein* und *Getrenntsein*, welche mit einem Wert von 60 m aufgrund der Datenqualität höher gewählt wurde.

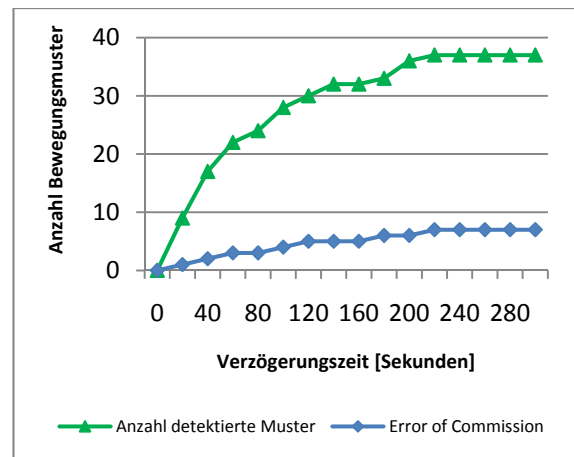


Diagramm 7.15: Einfluss der Verzögerungszeit auf die Algorithmen MD_r

Verlauf der Anzahl detektierter Reaktionsbewegungsmuster *Meiden* (grün) sowie der Error of Commission (blau) bei variierender Verzögerungszeit zwischen den Bewegungsmustern *Zusammentreffen* und *Getrenntsein*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.4 gehalten, mit Ausnahme der Segmentnachbarschaftsgröße von *Zusammentreffen* und *Getrenntsein*, für welche ein Wert von 1 gewählt wurde.

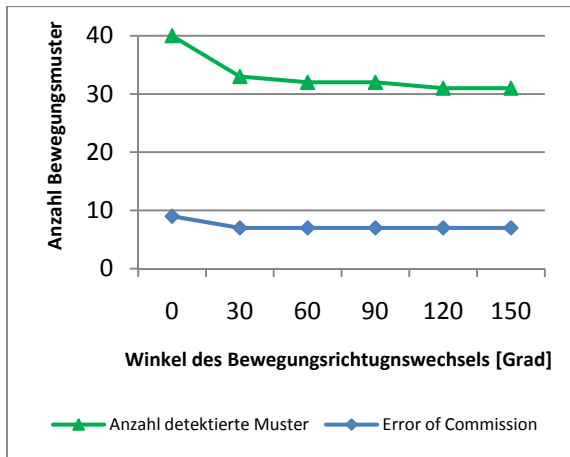


Diagramm 7.16: Einfluss des kritischen Winkels des Richtungswechsels auf den Algorithmus MD_i

Verlauf der Anzahl detektierter Reaktionsbewegungsmuster *Meiden* (grün) sowie der Error of Commission (blau) bei variierendem kritischem Winkel des Richtungswechsels. Alle anderen Parameter wurden konstant gehalten (kritische Distanz *Zusammensein* = 60 m, kritische Zeitdauer *Zusammensein* = 6 sec, Toleranz *Zusammensein* = 0, kritische Distanz *Getrenntsein* = 60 m, kritische Zeitdauer *Getrenntsein* = 15 sec, Toleranz *Getrenntsein* = 0, Minimaldistanz *Meiden* = 20 m, kritische Verzögerungszeit = 210 sec)

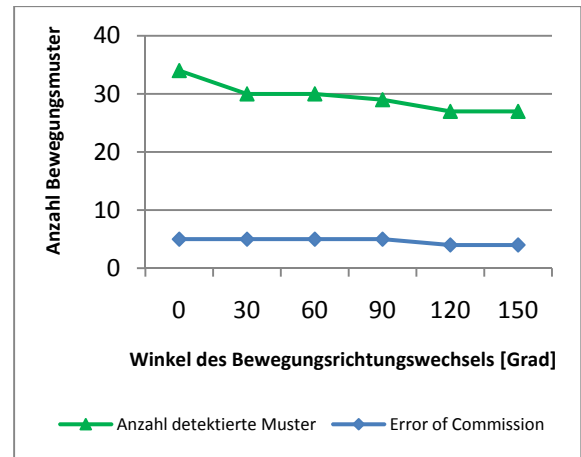


Diagramm 7.18: Einfluss des kritischen Winkels des Richtungswechsels auf den Algorithmus MD_r

Verlauf der Anzahl detektierter Reaktionsbewegungsmuster *Meiden* (grün) sowie der Error of Commission (blau) bei variierendem kritischem Winkel des Richtungswechsels. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.4 gehalten, mit Ausnahme der Segmentnachbarschaftsgröße von *Zusammentreffen* und *Getrenntsein*, für welche ein Wert von 1 gewählt wurde.

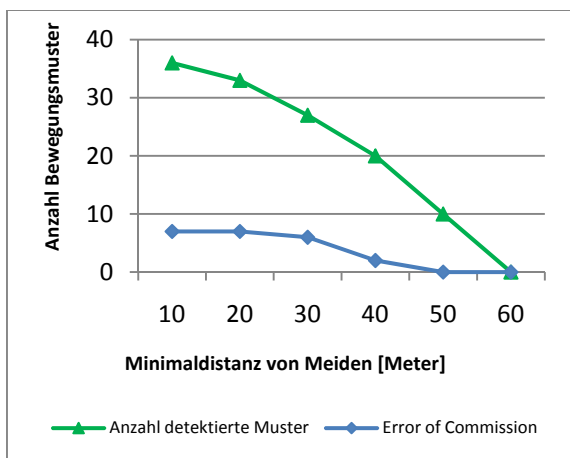


Diagramm 7.17: Einfluss der Minimaldistanz von Meiden auf den Algorithmus MD_i

Verlauf der Anzahl detektierter Reaktionsbewegungsmuster *Meiden* (grün) sowie der Error of Commission (blau) bei variierender Minimaldistanz von *Meiden*. Alle anderen Parameter wurden konstant gehalten (kritische Distanz *Zusammensein* = 60 m, kritische Zeitdauer *Zusammensein* = 6 sec, Toleranz *Zusammensein* = 0, kritische Distanz *Getrenntsein* = 60 m, kritische Zeitdauer *Getrenntsein* = 15 sec, Toleranz *Getrenntsein* = 0, kritischer Winkel des Richtungswechsel = 19 Grad, kritische Verzögerungszeit = 210 sec)

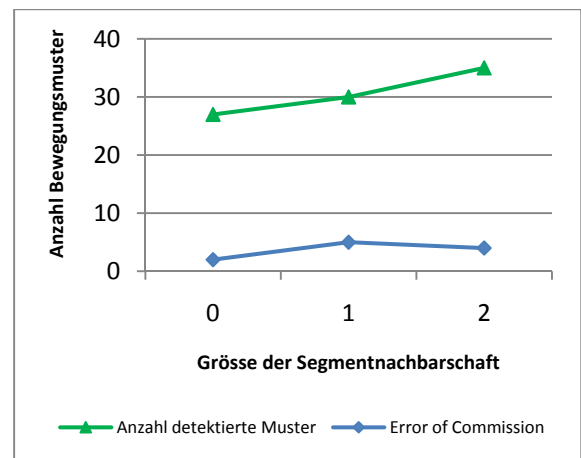


Diagramm 7.19: Einfluss der Segmentnachbarschaftsgröße auf den Algorithmen MD_r

Verlauf der Anzahl detektierter Reaktionsbewegungsmuster *Meiden* (grün) sowie der Error of Commission (blau) bei variierender Grösse der Segmentnachbarschaft. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.4 gehalten.

Dieser Anstieg der EOC bedeutet, dass der Algorithmus bei einer grösseren Verzögerungszeit zwischen *Zusammentreffen* und *Getrenntsein* auch Situationen als *Meiden* deklariert, in welchen sich in der Realität *Konfrontationen* zwischen diesen Punktobjekten abgespielt haben. Die Punktobjekte dürfen sich also länger nahe beieinander aufhalten, bis die Separationsbewegung erfolgt, wobei sie in dieser Zeit Interaktionsformen, wie eben zum Beispiel eine *Konfrontation*, durchführen können. Eine tiefe Verzögerungszeit ist also in diesem Fall enorm wichtig für eine erfolgreiche Abgrenzung von *Meiden* gegenüber anderen Bewegungsmustern wie die Herdenbildung oder *Konfrontationen*. Zur Veranschaulichung einer durch den Algorithmus bei einer maximal erlaubten Verzögerungszeit von 20 Sekunden fälschlicherweise als *Meiden* detektierten Situation, in welcher eigentlich eine *Konfrontation* stattgefunden hat (siehe schwarze Markierung) dient die folgende Abbildung 7.5.

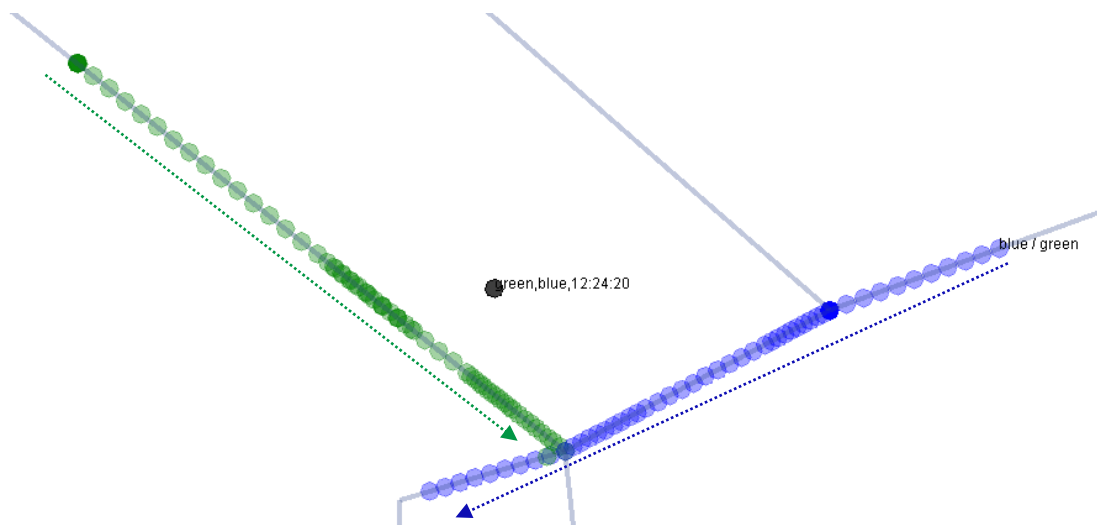


Abbildung 7.5: Beispiel einer fälschlicherweise als *Meiden* klassierten Situation

Beispiel einer Situation, welche als *Meiden* deklariert wurde, obwohl laut den semantischen Zusatzinformationen eine *Konfrontation* (gekennzeichnet durch schwarzen Punkt) stattgefunden hat. Diese Fehldetektion ist auf den zu hoch gewählten Wert für den Parameter Verzögerungszeit zurückzuführen.

Neben der Verzögerungszeit kann zudem mit Hilfe der minimal erlaubten Distanz zwischen den beteiligten Punktobjekten versucht werden, die Reaktionsbewegungsmuster *Meiden* und *Konfrontation* gegeneinander abzugrenzen und damit die EOC zu verringern. Dies gelingt sehr gut, wie das Diagramm 7.17 zeigt. Durch die Erhöhung der minimalen Annäherungsdistanz zwischen zwei Punktobjekten während des *Meidens* kann Schritt für Schritt der Wert der EOC verringert werden. Dies erscheint logisch, da die Punktobjekte im Falle einer *Konfrontation* ineinander hineinrennen und sich somit sehr nahe kommen.

Zum Abschluss der Betrachtung des Algorithmus MD_i soll das Augenmerk noch auf den Parameter Winkel des Bewegungsrichtungswechsels gelegt werden. Dieser Parameter erwies sich während den Experimenten mit den Simulationsdaten als der entscheidende für die Abgrenzung zwischen *Meiden* und zufälligem Aneinandervorbeigehen. Anhand eines Experiments wurde der Einfluss dieses Parameters bei der Analyse der Frequentie1550-Daten geprüft. Dabei wurde der Winkel über einen

Bereich von 0 bis 180 Grad variiert und dabei alle anderen Parameter konstant gehalten. Die Resultate dieses Experiment sind im Diagramm 7.16 zusammengefasst, wobei deutlich erkennbar ist, dass eine Variation der kritischen Bewegungsrichtungsänderung kaum einen Einfluss auf das Abschneiden des Algorithmus hat. Diese Erkenntnis lässt sich mit Hilfe des Strassennetzwerks erklären, was im Kapitel 8 eingehend diskutiert wird.

7.2.4 Algorithmus MD_r

Wie im Falle des Algorithmus MD_i wurde auch für den Algorithmus MD_r ein Experiment mit variierender Verzögerungszeit unter Verwendung der Frequentie1550-Daten durchgeführt, um die Wichtigkeit dieses Parameters herauszufinden. Dabei wurden die während dem Experiment unter Verwendung der idealen Parameter detektierten Situationen mit *Meiden* mit den semantischen Informationen der *Konfrontationen* verglichen. Da während eines Ereignisses mit *Meiden* sicherlich keine *Konfrontation* stattfinden kann, ist es mit Hilfe dieses Vergleichs möglich, die EOC zu benennen. Klar ist, dass neben diesen Fehldetektionen noch weitere EOC unter den detektierten Mustern sein können, wobei diese aufgrund fehlender semantischer Informationen zu den Ereignissen mit *Meiden* nicht nachgewiesen werden können. Immerhin kann mit diesem Experiment ein erster Eindruck über die Bedeutung des Parameters Verzögerungszeit gewonnen werden. Der Verlauf der Kurven in Diagramm 7.15 zeigt, dass mit zunehmender Verzögerungszeit die Anzahl der detektierten Bewegungsmuster sowie der Wert der EOC steigen. Dies bedeutet, dass mit grösserer Verzögerungszeit auch zusehends *Meiden* detektiert wird, obwohl eigentlich eine Konfrontation stattgefunden hat.

In einem zweiten Experiment wurde ein weiterer wichtiger Parameter, nämlich der kritische Winkel des Bewegungsrichtungswechsels, unter die Lupe genommen. Bereits in den Experimenten mit den Simulationsdaten wurde dieser Parameter untersucht, wobei sich dieser Parameter zwar als wichtig herausstellte, jedoch auch die Grenzen seines Einflusses bei der Reduktion der EOC ab einem Winkel von 60 Grad aufgezeigt werden konnten. Im Falle der Frequentie1550-Daten ist die Bedeutung dieses Parameters noch geringer, was aus den mehr oder weniger horizontal verlaufenden Kurven im Diagramm 7.18 ablesbar ist. Durch eine Erhöhung des kritischen Winkels kommt es nur zu einer minimalen Abnahme der Anzahl der detektierten Reaktionsbewegungsmuster *Meiden* sowie der EOC. Dieser Umstand ist mit den verwendeten Bewegungsdaten, welche sich auf einem Netzwerk abspielen, erklärbar, was in Kapitel 8 diskutiert wird.

Um den Einfluss des räumlichen Massstabs auf die Resultate des Algorithmus MD_r zu testen, wurde ein Experiment durchgeführt, bei dem die Grösse der Segmentnachbarschaft der beiden Bewegungsmuster *Zusammentreffen* und *Getrenntsein* variiert wurde. Die Resultate dieses Experiments sind im Diagramm 7.19 dargestellt. Wie erwartet nimmt mit zunehmender Segmentnachbarschaftsgrösse auch die Anzahl der detektierten Bewegungsmuster zu.

7.2.5 Algorithmus KD_i

Die Analyse der Güte des Algorithmus KD_i wurde mit Hilfe der semantischen Zusatzinformationen durchgeführt. Bei diesen Tests stellte sich heraus, dass der Algorithmus KD_i nicht sehr gut abschneidet. Dies ist aus den in der Tabelle 7.5 aufgelisteten idealen Parametern und den verschiedenen Gütemassen ablesbar. Die Werte für die beiden Parameter kritische Distanz des *Zusammentreffens* (50 Meter) sowie die kritische Zeitspanne (30 Sekunden) wurden gewählt, wie es in Orellana et al. (2009) für die Detektion von Approximationen vorgeschlagen wurde. Dies vor allem mit dem Hintergedanken, dadurch einen Vergleich mit dem Ansatz von Orellana et al. (2009) durchführen zu können.

kritische Distanz <i>Zusammensein</i> [Meter]	50	Anzahl richtig detektierte Muster	13
kritische Zeitdauer <i>Zusammensein</i> [Sekunden]	30	Anzahl detektierbare Muster	33
Toleranz <i>Zusammensein</i>	0	Producer's Accuracy	39.4
kritische Distanz <i>Getrenntsein</i> [Meter]	50	User's Accuracy	50.0
kritische Zeitdauer <i>Getrenntsein</i> [Sekunden]	15	Error of Omission	20 (0.6)
Toleranz <i>Getrenntsein</i>	0	Error of Commission	13 (0.39)
kritische Intersektionsdistanz [Meter]	50	Inclass Accuracy	0.4
kritische Zeitdauer <i>Intersektion</i> [Sekunden]	1		
kritische Verzögerungszeit [Sekunden]	100		

Tabelle 7.5: Auflistung der idealen Parameter des Algorithmus KD_i

Übersicht der ermittelten idealen Parameterwerte des Algorithmus KD_i (link Spalte) sowie der Gütemasse aus dem Experiment mit den Frequenzdaten unter Verwendung der idealen Werte (rechte Spalte).

Die Diskussion der genauen Ursachen für das schlechte Abschneiden des Algorithmus KD_i folgt in Kapitel 8. In den folgenden Abschnitten werden die Resultate der Experimente mit den drei wichtigsten Parametern dieses Algorithmus aufgezeigt. Zu diesen drei Parametern gehören die kritische Zeitdauer des Bewegungsmusters *Getrenntsein*, die Verzögerungszeit sowie die kritische Intersektionsdistanz. Der Anfang macht der Parameter kritische Zeitdauer des Bewegungsmusters *Getrenntsein*. Laut der Definition des Reaktionsbewegungsmusters *Konfrontation* müssen sich die beteiligten Punktobjekte, nach dem sie ineinander hineingerannt sind, wieder voneinander entfernen. Dies ist notwendig, um dieses Reaktionsbewegungsmuster von anderen Bewegungsmustern, wie zum Beispiel der *Herdenbildung*, abzugrenzen. Während die Punktobjekte bei einer Herdenbildung die Nähe zueinander suchen, haben diese im Falle einer *Konfrontation* keinerlei Beweggründe, um sich nach dem *Zusammentreffen* noch länger in der Nähe aufzuhalten und als Folge davon beginnen sie, sich zu separieren. Für die Analyse des Bewegungsmusters *Getrenntsein* wurde ein Experiment durchgeführt, bei welchem die kritische Zeitdauer variiert wurde. Die Resultate dieses Experiments sind im nachfolgenden Diagramm 7.20 dargestellt. Der Verlauf der Kurve der EOC zeigt, dass mit zunehmender vorgeschriebener Zeitdauer des Bewegungsmusters *Getrenntsein* die fälschlicherweise als *Konfrontation* klassierten Bewegungsmuster reduziert werden können. Wie die Kurve der EOO

allerdings zeigt, kann diese Reduktion der EOC nicht ohne eine gleichzeitige Reduktion der richtig detektierten Bewegungsmuster erzielt werden.

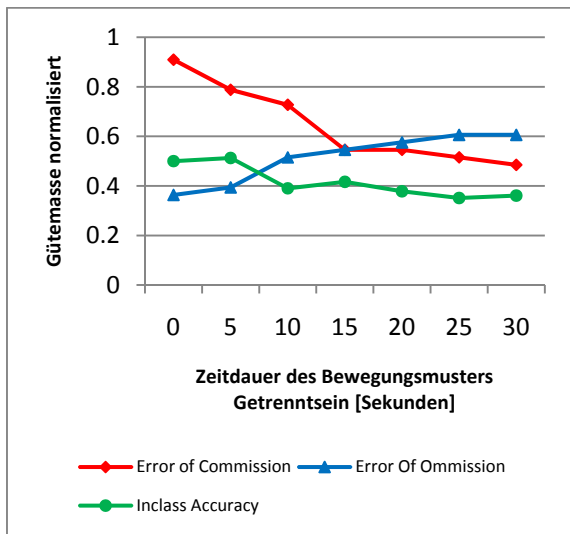


Diagramm 7.20: Einfluss der kritischen Zeitdauer von *Getrenntsein* auf den Algorithmus KD_i

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierender kritischer Zeitdauer des Bewegungsmusters *Getrenntsein*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.5 gehalten. Die Ausnahme bildet einzig der Wert von 600 für die Verzögerungszeit, welcher höher gewählt wurde, um den Einfluss der Zeitdauer von *Getrenntsein* zu verstärken.

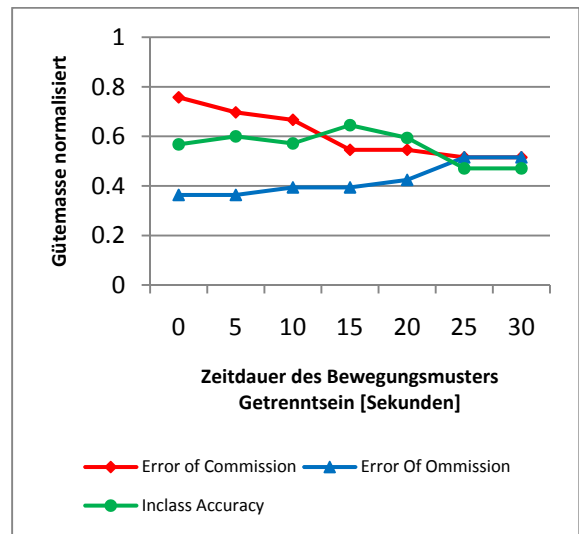


Diagramm 7.22: Einfluss der kritischen Zeitdauer von *Getrenntsein* auf den Algorithmus KD_r

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierender Größe der Segmentnachbarschaft. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.6 gehalten. Die Ausnahme bildet einzig der Wert von 600 sec für die Verzögerungszeit, welcher höher gewählt wurde, um den Einfluss der Zeitdauer von *Getrenntsein* zu verstärken.

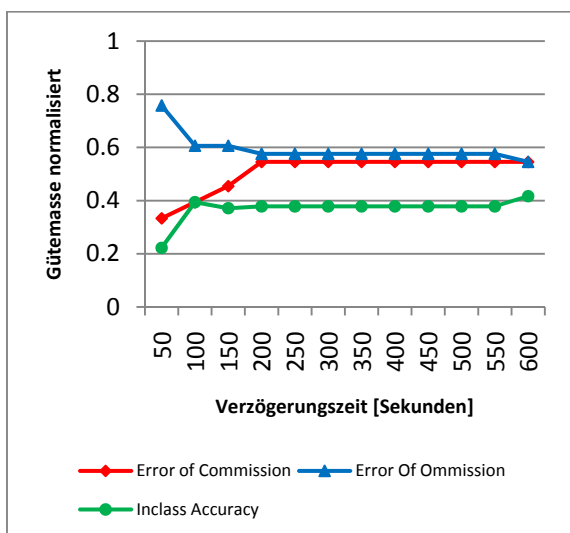


Diagramm 7.21: Einfluss der Verzögerungszeit auf den Algorithmus KD_i

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierender Verzögerungszeit. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.5 gehalten.

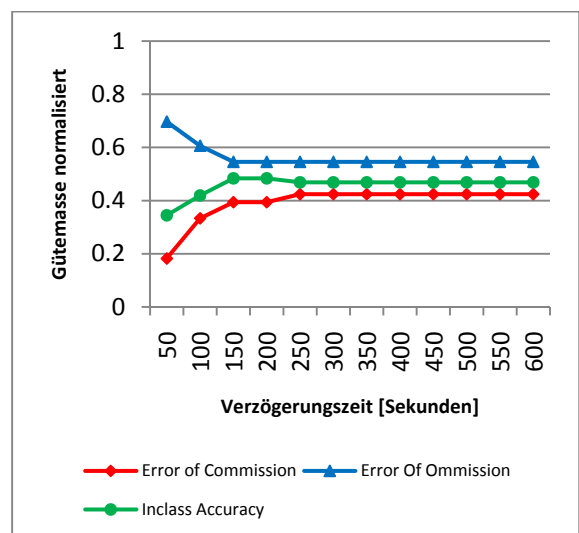


Diagramm 7.23: Einfluss der Verzögerungszeit auf den Algorithmus KD_r

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierender Verzögerungszeit. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.6 gehalten.

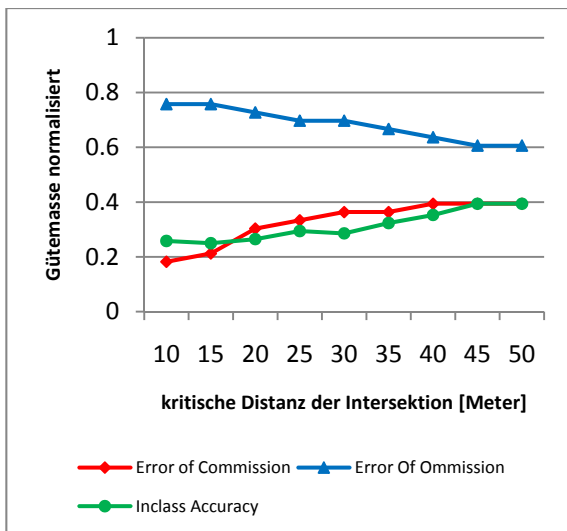


Diagramm 7.24: Einfluss der kritischen Intersektionsdistanz auf den Algorithmus KD_i

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierender kritischer Intersektionsdistanz. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.5 gehalten.

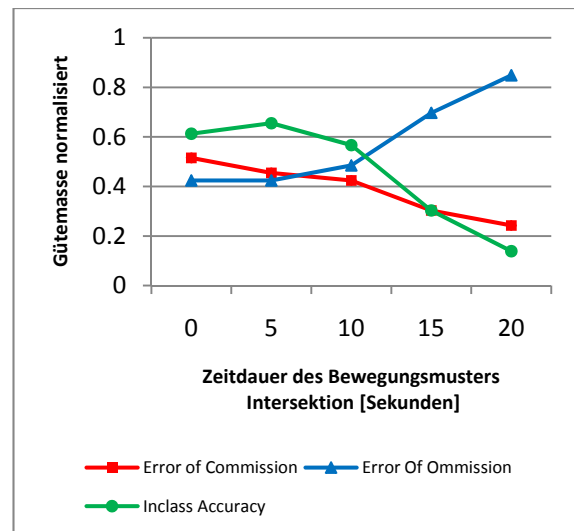


Diagramm 7.25: Einfluss der kritischen Zeitdauer der Intersektion auf den Algorithmus KD_r

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierender kritischer Zeitdauer der *Intersektion*. Alle anderen Parameter wurden konstant auf dem idealen Wert der Tabelle 7.6 gehalten.

Eine detaillierte Erläuterung von möglichen Erklärungsversuchen für diesen Zusammenhang folgt in Kapitel 8. Ein zweiter interessanter Parameter ist die Verzögerungszeit zwischen dem Zeitpunkt der *Intersektion* von zwei Punktobjekten und der *Separation* zwischen denselben. Um die Bedeutung dieses Parameters zu ergründen, wurde ebenfalls ein Experiment durchgeführt, bei dem die Verzögerungszeit variiert wurde. Die Resultate dieses Experiments, welche im Diagramm 7.21 dargestellt sind, zeigen, dass der Wert der EOC mit Hilfe einer Reduktion der Verzögerungszeit zwischen 200 und 50 Sekunden kontinuierlich verringert werden kann. Allerdings kommt es auch in diesem Fall gleichzeitig zu einer Erhöhung des Werts der EOO und somit kann ausgesagt werden, dass auch mit Hilfe des Parameters Verzögerungszeit keine ideale Abgrenzung zwischen dem Reaktionsbewegungsmuster *Konfrontation* und anderen Interaktionsformen möglich ist. Nicht anders sieht es im Falle des Parameters Intersektions-Distanz aus, was bei der Betrachtung des Diagramms 7.24 klar wird. Eine detaillierte Diskussion dieser Sachverhalte folgt in Kapitel 8.

7.2.6 Algorithmus KD_r

Der Algorithmus KD_r hat im Vergleich zu seinem Pendant der individuellen Perspektive (Algorithmus KD_i) bei den Experimenten etwas besser abgeschnitten. Auch in diesem Fall wurde der Wert der kritischen Zeitdauer des Bewegungsmusters *Zusammensein* dem in Orellana et al. (2009) vorgeschlagenen Wert von 30 Sekunden gleichgesetzt. Anstelle der kritischen Distanz von 50 Metern tritt im Falle des Algorithmus KD_r die Größe der Segmentnachbarschaft, welche aufgrund der Experimentresultate gleich 1 gesetzt wurde. Eine Auflistung aller idealen Parameterwerte und der Gütemasse ist in Tabelle 7.6 zu sehen.

Nachbarschaftsgrösse <i>Zusammensein</i>	1	Anzahl richtig detektierte Muster	20
kritische Zeitdauer <i>Zusammensein</i> [Sekunden]	30	Anzahl detektierbare Muster	33
Toleranz <i>Zusammensein</i>	0	Producer's Accuracy	60.6
Nachbarschaftsgrösse <i>Getrenntsein</i>	1	User's Accuracy	55.6
kritische Zeitdauer <i>Getrenntsein</i> [Sekunden]	15	Error of Omission	13 (0.39)
Toleranz <i>Getrenntsein</i>	0	Error of Commission	16 (0.48)
kritische Zeitdauer <i>Intersektion</i> [Sekunden]	5	Inclass Accuracy	0.7
Toleranz <i>Intersektion</i>	0		
kritische Zeitverzögerung [Sekunden]	120		

Tabelle 7.6: Auflistung der idealen Parameter des Algorithmus KD_r .

Übersicht der ermittelten idealen Parameterwerte des Algorithmus KD_r (link Spalte) sowie der Gütemasse aus dem Experiment mit den Frequentie1550-Daten unter Verwendung der idealen Werte (rechte Spalte).

Wie im Falle des Algorithmus KD_i folgt auch für den Algorithmus KD_r eine Diskussion der Schwächen und der Gründe für das nicht optimale Abschneiden bei den Experimenten in Kapitel 8. In den folgenden Abschnitten soll nur kurz auf die Resultate der Experimente, bei denen die wichtigsten Parameter kritische Zeitdauer des *Getrenntseins*, Verzögerungszeit sowie Zeitdauer der *Intersektion* variiert wurden, eingegangen werden. Die Experimente mit den ersten beiden Parametern ergaben die gleichen Schlussfolgerungen wie beim Algorithmus KD_i , und zwar, dass mit Hilfe dieser beider Parameter keine ideale Abgrenzung zwischen dem Reaktionsbewegungsmuster *Konfrontation* und anderen Interaktionsformen möglich ist. Dies zeigen auch die Kurvenverläufe in den Diagrammen 7.22 und 7.23. Im Gegensatz zum Algorithmus KD_i wurde für den Algorithmus KD_r aufgrund des Fehlens eines kritischen Distanzparameters der *Intersektion* ein Experiment mit der kritischen Zeitdauer der *Intersektion* durchgeführt. Im Falle der räumlichen Perspektive bedeutet die *Intersektion*, dass sich zwei Punktobjekte über diese kritische Zeitspanne aufeinander zu bewegen. Je grösser der Wert dieses Parameters ist, desto höher sind die Anforderungen an eine *Konfrontation*. Dies widerspiegelt sich auch im Verlauf der EOC und EOO, welche bei steigender kritischer Zeitdauer der *Intersektion* gegensätzlich verlaufen (Siehe Diagramm 7.25). Während der Wert der EOC kontinuierlich zunimmt, steigen die EOO stetig an. Die Quintessenz der Experimente mit diesem Parameter bleibt also die gleiche wie bei den beiden vorangegangenen und zwar, dass auch mit Hilfe der Intersektionszeitdauer keine optimale Abgrenzung erzielt werden kann.

7.2.7 Ansatz zur Detektion von Annäherung nach Orellana et al. (2009)

Um die beiden in dieser Arbeit entwickelten Algorithmen zur Detektion von *Konfrontationen* mit einem Ansatz aus der Literatur vergleichen zu können, wurde zusätzlich noch die in Orellana et al. (2009) vorgestellte Methode zur Bestimmung von *Approximationen* zwischen sich bewegenden Punktobjekten implementiert und anhand von Experimenten mit den Frequentie1550-Daten getestet. Orellana et al. (2009) schlagen in ihrer Arbeit die Verwendung einer kritischen Distanz von 50 Metern sowie einer kritischen Zeitspanne von mindestens 30 Sekunden vor, um eine Situation als *Approximation* zu klassieren. Mit Hilfe dieser Parameterwerte finden Orellana et al. (2009) bei einem

Experiment mit den Frequentie1550-Daten 68 Situationen mit einer *Approximation* zwischen zwei Punktobjekten. Der eigens implementierte Ansatz von Orellana et al. (2009) wurde mit den vorgeschlagenen Parameterwerten von 50 Metern und 30 Sekunden anhand eines Experiments mit den vorprozessierten Frequentie1550-Daten getestet. Dabei wurden die detektierten *Approximationen* mit den semantischen Informationen der *Konfrontationen* verglichen. Klar ist, dass es nicht die Absicht von Orellana et al. (2009) war, mit Hilfe der vorgeschlagenen Methode möglichst viele *Konfrontationen* zu detektieren, sondern *Annäherungen* zwischen Punktobjekten zu finden. Trotzdem scheint es durchaus interessant herauszufinden, wie gut diese Methode für die Detektion von *Konfrontationen* angewendet werden kann. Die Testresultate sind in der Tabelle 7.7 dargestellt.

kritische Distanz <i>Zusammensein</i> [Meter]	50	Anzahl richtig detektierte Muster	27
kritische Zeitdauer <i>Zusammensein</i> [Sekunden]	30	Anzahl detektierbare Muster	33
Toleranz <i>Zusammensein</i>	0	Producer's Accuracy	81.8
		User's Accuracy	44.3
		Error of Omission	6 (0.18)
		Error of Commission	34 (1.03)
		Inclass Accuracy	0.7

Tabelle 7.7: Auflistung der idealen Parameter des Ansatzes von Orellana et al. (2009)

Übersicht der in Orellana et al. (2009) vorgeschlagenen Parameterwerte (link Spalte) sowie der Gütemasse aus dem Experiment mit den Frequentie1550-Daten unter Verwendung dieser Werte (rechte Spalte).

Somit konnten mit Hilfe dieser Methode insgesamt 61 Situationen mit *Approximationen* gefunden werden, was im Vergleich zur totalen Anzahl von 68 Detektionen von Orellana et al. (2009) eine Differenz von 7 *Approximationen* ergibt. Diese Differenz ist darauf zurückzuführen, dass in dieser Arbeit eine andere Vorprozessierung der Frequentie1550-Daten durchgeführt wurde. Die in der Tabelle 7.7 aufgelisteten Werte für die Gütekriterien zeigen, dass das Verfahren durchwegs sehr erfolgreich bei der Klassierung von *Konfrontationen* ist, was sich in einem geringen Wert der EOC und einem hohen Wert der Producer's Accuracy von 81.8 widerspiegelt. Der hohe Wert der EOC sowie der tiefe Wert der User's Accuracy von 44.3 zeigen, dass dieser Ansatz allerdings Probleme bekundet bei der Abgrenzung von *Konfrontationen* gegenüber anderen Interaktionsformen, bei welchen sich zwei Punktobjekte über eine Zeitspanne von mindestens 30 Sekunden unter eine kritische Distanz von 50 Metern annähern. An dieser Stelle soll noch einmal darauf hingewiesen werden, dass dies auch nicht das Ziel der Methode von Orellana et al. (2009) ist. Mit Hilfe eines Experiments, dessen Resultate im Diagramm 7.26 dargestellt sind, konnte der Einfluss der kritischen Distanz des Bewegungsmusters *Approximation* gezeigt werden. Wie erwartet steigt mit zunehmender Distanz der Wert der EOC an, wobei gleichzeitig der Wert der EOO sinkt.

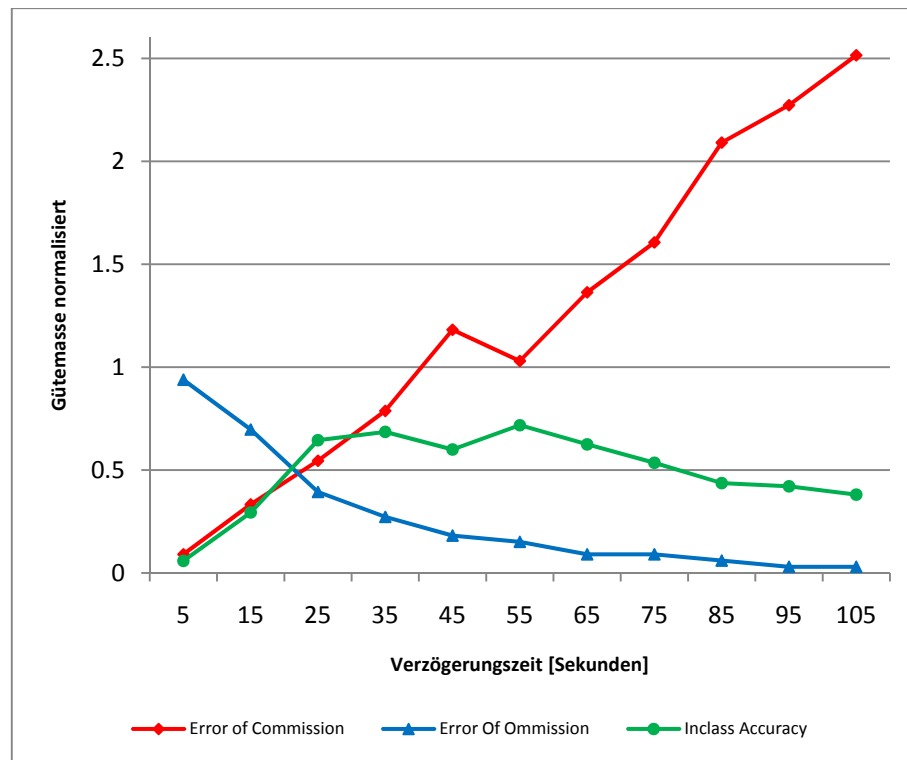


Diagramm 7.26: Einfluss der kritischen Distanz von *Annähern* auf den Ansatz von Orellana et al. (2009)

Verlauf der normalisierten Masse der Error of Commission (rot) und Error of Omission (blau) sowie die Kurve der Inclass Accuracy (grün) bei variierender kritischer Distanz des Bewegungsmusters *Approximation*. Die beiden anderen Parameter Zeitdauer der *Approximation* (= 30 Sekunden) sowie Toleranz der *Approximation* (= 0) wurden konstant gehalten.

8 Diskussion

In einem ersten Teil der Diskussion werden die Forschungsfragen dieser Masterarbeit noch einmal aufgegriffen. Dabei wird zuerst auf die Forschungsfragen 1 und 2 eingegangen, welche die beiden Arbeitsschritte der Formalisierung und der Algorithmenentwicklung betreffen. Anschliessend werden für alle in Kapitel 5 vorgestellten Algorithmen zur Detektion der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* die beiden Forschungsfragen 3 und 4 diskutiert. Dabei wird sowohl für die individuelle, als auch für die räumliche Perspektive der Algorithmen zuerst auf die Stärken und Schwachpunkte sowie auf die Einflüsse der wichtigsten Parameter eingegangen (FF 3), bevor anschliessend die beiden Perspektiven miteinander verglichen werden (FF 4). Bei diesem Vergleich soll ein spezielles Augenmerk auf das Abschneiden bei den Experimenten gelegt werden. Zu dieser Diskussion der Resultate gilt es anzumerken, dass die, mit Hilfe der Experimente bestimmten, idealen Parameterwerte nicht allgemeingültig sind. Diese Werte erwiesen sich lediglich für die verwendeten Daten als ideal. Sie sind stark abhängig von den jeweiligen Rahmenbedingungen, welche bei der Aufnahme der Bewegungsdaten vorherrschten sowie von der Art der beteiligten Punktobjekte. Die vorgeschlagenen Parameterwerte sind also mit Vorsicht zu geniessen und müssen im Falle einer Anwendung der Algorithmen für einen anderen Datensatz neu gewählt werden. Dabei können die Parameterwerte durch Experimente oder mit Hilfe von Expertenwissen (z.B. Verhaltensbiologen) bestimmt werden.

In einem zweiten Teil dieser Diskussion sollen die Erkenntnisse, welche durch diese Arbeit gewonnen werden konnten, in den Forschungskontext eingeordnet werden. Dabei werden die am Ende des zweiten Kapitels genannten Forschungslücken wieder aufgegriffen und es soll diskutiert werden, inwiefern diese Arbeit einen Beitrag zur Schliessung dieser Lücken leisten kann. Bei diesen Ausführungen soll zudem auch eine Abgrenzung zu Arbeiten anderer Forscher diskutiert werden.

8.1 Forschungsfrage 1: Herleitung formaler Definitionen

FF1: Wie können, basierend auf den in der Literatur vorhandenen Definitionen, die Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* sowie *Meiden* formalisiert werden?

Die Resultate der Formalisierung der drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* sowohl aus der individuellen, als auch aus der räumlichen Perspektive sind in Kapitel 4 beschrieben. In den folgenden Abschnitten sollen die wichtigsten Erkenntnisse, welche während der Ausarbeitung der Formalisierung erzielt werden konnten, noch kurz diskutiert werden.

8.1.1 Probleme/Schwierigkeiten bei der Formalisierung

Die Schwierigkeit im Falle der Formalisierung des Reaktionsbewegungsmusters *Konfrontation* war der Umgang mit den vielen, in der Literatur beschriebenen, möglichen Ausprägungen dieses Musters.

Je nach Art und Anzahl der beteiligten Objekte können *Konfrontationen* sehr unterschiedliche Merkmale aufweisen, was eine allgemeingültige Formalisierung stark erschwert. Vergleicht man beispielsweise ein Kampf zwischen einem Löwen und einem Zebra mit einer Schlägerei zwischen Fanggruppierungen nach einem Fussballspiel so wird unweigerlich klar, wie unterschiedlich *Konfrontationen* sein können. Zudem gilt es anzumerken, dass für die kleinräumigen Bewegungen während einer *Konfrontation* (z.B. in einem Kampf) nur sehr schwer universell gültige Bewegungsabfolgen definiert werden können. Aus diesen Gründen wurde dieses Reaktionsbewegungsmuster stark abstrahiert als Abfolge von *Annäherung*, *Zusammentreffen* und *Separation* formalisiert. Eine solche allgemeine Formalisierung birgt allerdings die Gefahr, dass bei vielen verschiedenen ähnlichen Interaktionsformen in den Bewegungsdaten eine Abgrenzung von *Konfrontationen* nicht gewährleistet ist. Während das Bewegungsmuster *Meiden* bei der Formalisierung keine grösseren Probleme verursachte, war im Falle von *Verfolgen/Entfliehen* die Unterscheidung zu anderen ähnlichen Bewegungsmustern eine Herausforderung. Dies ist darauf zurückzuführen, dass andere Interaktionsformen, wie beispielsweise die Bewegungen in *Anführen/Folgen* oder *Single-File*, sich aus raum-zeitlicher Sicht nur unwesentlich von *Verfolgen/Entfliehen* unterscheiden. Mit der geringen Verzögerungszeit zwischen dem *Annähern* und dem Start der *Verfolgung* konnte aber ein Merkmal gefunden werden, mit welchem *Verfolgen/Entfliehen* gegenüber anderen Bewegungsmustern abgegrenzt werden kann. Eine ausführliche Diskussion des Einflusses dieses Kriteriums folgt in den nächsten Abschnitten bei der Interpretation der Experimentresultate und bei der Besprechung der Forschungsfrage 3.

8.1.2 Verwendete Perspektiven

Während der Formalisierung der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* wurde sowohl die individuelle, als auch die räumliche Perspektive verwendet. Dabei wurde versucht, die in Turchin (1998) vorgeschlagene Lagrangesche und Eulersche Sichtweise umzusetzen. Im Falle der individuellen Perspektive konnten die Ideen der Lagrangeschen Betrachtungsweise, bei welcher die beweglichen Punktobjekte im Fokus der Betrachtung stehen, eingebunden werden. Bei der Formalisierung mit Hilfe der räumlichen Perspektive wurde versucht, die Grundsätze des Eulerschen Blickwinkels umzusetzen. Für den klassisch räumlichen Fall wurden dabei die den Bewegungen unterliegenden Räume in Raumkompartimente unterteilt. Diese Betrachtungsweise widerspiegelt genau die Eulersche Perspektive, bei der die Bewegungen durch einen statischen Blick auf fixe Räume analysiert werden. Für den spezifischen Anwendungsfall der Frequentie1550-Daten stellten die Strassensegmente die kleinste unterteilbare Raumeinheit dar. Für die Formalisierung von Bewegungen auf einem Strassennetzwerk wurde die Segmentnachbarschaft eines Punktobjekts eingeführt. Da sich die Punktobjekte auf den Segmenten bewegen, kommt es zu einer ständigen Veränderung der Segmentnachbarschaft und als Folge davon kann nicht mehr von einer statischen Betrachtung des Raumes gesprochen werden. Dies ist also eine Abweichung zur Eulerschen Perspektive, welche für eine gute Formalisierung von Reaktionsbewegungsmustern im Falle von Bewegungen auf einem

Netzwerk nötig war. Um die Eulersche Perspektive eins zu eins auf ein Netzwerk anzuwenden, müssten entweder alle Segmente einzeln betrachtet werden, die Segmente zu fixen Zonen zusammengefasst oder ein Raster mit einer vordefinierten Maschenweite verwendet werden.

8.2 Forschungsfrage 2: Entwicklung von Algorithmen

FF2: Wie lassen sich diese Formalisierungen in Algorithmen umsetzen, um die definierten Reaktionsbewegungsmuster in raum-zeitlichen Datenreihen zu detektieren?

Eine ausführliche Beschreibung der Algorithmen zur Detektion der drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* ist in Kapitel 5 zu finden. In diesem Abschnitt werden die wichtigsten Erfahrungen bei der Entwicklung der Algorithmen noch einmal aufgegriffen und diskutiert.

8.2.1 Aufbau der Algorithmen

Die Algorithmen zur Detektion der drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* sind jeweils zusammengesetzt aus der Entdeckung von einzelnen Bewegungsmustern sowie der Überprüfung von zusätzlichen Kriterien (z.B. Verzögerungszeit zwischen den einzelnen Mustern). Als Beispiel kann die *Konfrontation* genannt werden, welche als Abfolge von *Annähern*, *Zusammentreffen* und *Separation* formalisiert wurde. Der Algorithmus testet demzufolge das Vorhandensein dieser drei Muster sowie die Verzögerungszeiten zwischen diesen. Dabei werden jeweils paarweise Vergleiche zwischen allen Punktobjekten über alle Zeitschritte durchgeführt, was einer Laufzeitkomplexität von $O(n^2t)$ gleichkommt. Um diese Komplexität zu verringern müssten zusätzliche Anstrengungen unternommen werden, was in Kapitel 9 noch thematisiert wird.

8.2.2 Zusammensein/Getrenntsein vs. Konvergenz/Divergenz

Eine wichtige Erkenntnis der Algorithmenentwicklung betrifft die Detektion der beiden Muster *Zusammensein*, welches als Folge einer *Konvergenz* (*Annäherung*) zustande kommt, und *Getrenntsein*, welches das Resultat der *Divergenz* (*Separation*) darstellt. Bei der Entwicklung der Algorithmen wurden für alle vier Muster verschiedene Methoden entwickelt. Tests mit diesen Methoden haben aber gezeigt, dass für die Detektion von Reaktionsbewegungsmustern primär *Zusammensein* und *Getrenntsein* als Beschreibung des Zustandes der Distanz- beziehungsweise Nachbarschaftsbeziehung relevant sind. *Konvergenz* und *Divergenz* sind von untergeordneter Bedeutung, da es nicht relevant ist, ob sich zwei Punktobjekte auf direktem Weg oder mit Unterbrüchen annähern/separieren. Wichtig ist nur, dass sich zwei Punktobjekte angenähert/separiert haben und danach über eine gewisse Zeitdauer zusammen/getrennt bleiben.

8.2.3 Anwendbarkeit der Algorithmen

Die letzte nennenswerte Erkenntnis betrifft die Anwendbarkeit der entwickelten Algorithmen auf reale Bewegungsdaten. Bewegungsdatensätze können geprägt sein durch fehlerhafte Positionsangaben und Sprünge in den Daten, wie anhand der Frequentie1550-Daten in dieser Arbeit gezeigt werden konnte. Um die Verwendung der Algorithmen zur Detektion von Reaktionsbewegungsmustern auch für die Analyse von realen Daten zu gewährleisten, wurde ein Toleranzparameter eingeführt. Dieser Parameter erlaubt Unterbrüche in den Bewegungsmustern. Solange die Anzahl Unterbrüche den gewählten Wert des Toleranzparameters nicht überschreitet, so kann eine Situation mit einem Reaktionsbewegungsmuster immer noch richtig detektiert werden. Dies ist zum Beispiel bei der Anwendung des Algorithmus VED_i der Fall, wenn sich die Beute für kurze Zeit aus dem Frontbereich des Verfolgers bewegt. Mit Hilfe der Verwendung des Toleranzparameters kann demzufolge die Anwendbarkeit der Algorithmen auf fehlerhafte Bewegungsdatensätze sichergestellt werden.

8.3 Forschungsfragen 3 und 4: Einfluss der Parameter und Vergleich der beiden Perspektiven

- FF3: Welches sind die entscheidenden Parameter für eine erfolgreiche Detektion der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* und wie gross ist deren Einfluss für die Abgrenzung zu verwandten Bewegungsmustern?
- FF4: Inwiefern unterscheiden sich die Resultate der Algorithmen der individuellen Perspektive von denjenigen der räumlichen Perspektive?

Wie bereits eingangs dieses Kapitels angesprochen, werden die beiden Forschungsfragen 3 und 4 zusammen untersucht. Dabei wird jeweils für die beiden Algorithmen eines Reaktionsbewegungsmuster (z.B. Algorithmus VED_i und VED_r) die Forschungsfrage 3 diskutiert, bevor anschliessend die beiden Perspektiven, welche den Algorithmen zu Grunde liegen, einander gegenüber gestellt werden.

Die wichtigsten Erkenntnisse der Diskussion der Forschungsfrage 3 sind als Übersicht in der Tabelle 8.1 aufgelistet. Dabei sind für alle sechs in dieser Arbeit vorgeschlagenen Algorithmen die wichtigsten Parameter aufgelistet und deren Einfluss bei den Experimenten mit den simulierten Bewegungsdaten und den Frequentie1550-Daten aufgeführt. Der Einfluss der Parameter ist jeweils in die beiden Kategorien gering und gross klassiert. Zudem ist für jede Beurteilung in Klammern eine kurze Begründung angegeben.

Eine ausführliche Diskussion der genauen Zusammenhänge der verschiedenen Parameter, sowie die Vergleiche zwischen der räumlichen und der individuellen Perspektive (Forschungsfrage 4) sind in den nachfolgenden Abschnitten zu finden.

Algorithmus	Wichtige Parameter	Einfluss in den Experimenten mit simulierten Daten	Einfluss in den Experimenten mit Frequentie1550-Daten
VED _i	Verzögerungszeit	gering (keine Abgrenzung zu anderen Interaktionen notwendig)	gross (Abgrenzung zu anderen Interaktionen)
	Frontbereichswinkel	gross (Sicherstellen, dass sich POs in gleiche Richtung bewegen)	gross (Abgrenzung zu Bew. auf benachbarten Segmenten)
	Distanz <i>Folgen</i>	gross (Sicherstellen, dass sich POs sehen können)	gross (Sicherstellen, dass sich POs sehen können)
	Zeitdauer <i>Folgen</i>	gross (Abgrenzung zu Zufallsbewegungen in gleiche Richtung)	gross (Abgrenzung zu Zufallsbewegungen in gleiche Richtung)
VED _r	Verzögerungszeit	gering (keine Abgrenzung zu anderen Interaktionen notwendig)	gross (Abgrenzung zu anderen Interaktionen)
	Nachbarschaftsgrösse	gross (Sicherstellen, dass sich POs sehen können)	gross (Sicherstellen, dass sich POs sehen können)
	Zeitdauer <i>Folgen</i>	gross (Abgrenzung zu Zufallsbewegungen in gleiche Richtung)	gross (Abgrenzung zu Zufallsbewegungen in gleiche Richtung)
MD _i	Verzögerungszeit	gering (keine Abgrenzung zu anderen Interaktionen notwendig)	gross (Abgrenzung zu anderen Interaktionen)
	Bewegungsrichtungswechsel	gross (abrupte Richtungsw. beim <i>Meiden</i> im euklidischen Raum)	gering (Abbiegen auf Netzwerk ebenfalls Richtungswechsel)
	Minimaldistanz <i>Meiden</i>	gross (Abgrenzung zu Kreuzen der Bewegungspfade zwischen POs)	gross (Abgrenzung zu anderen Interaktionen)
MD _r	Verzögerungszeit	gering (keine Abgrenzung zu anderen Interaktionen notwendig)	gross (Abgrenzung zu anderen Interaktionen)
	Bewegungsrichtungswechsel	gering (Abbiegen auf Netzwerk ebenfalls Richtungswechsel)	gering (Abbiegen auf Netzwerk ebenfalls Richtungswechsel)
KD _i	Verzögerungszeit	keine Experimente durchgeführt	gross (Abgrenzung zu anderen Interaktionen)
	Zeitdauer <i>Getrenntsein</i>		gering (fehlende <i>Separation</i> nach einer <i>Konfrontation</i>)
	Distanz <i>Intersektion</i>		gering (Bewegungsdaten zu fehlerhaft)
KD _r	Verzögerungszeit	keine Experimente durchgeführt	gross (Abgrenzung zu anderen Interaktionen)
	Zeitdauer <i>Getrenntsein</i>		gering (fehlende <i>Separation</i> nach <i>Konfrontation</i>)
	Zeitdauer <i>Intersektion</i>		gering (Bewegungsdaten zu fehlerhaft)

Tabelle 8.1: Auflistung der Einflüsse der wichtigsten Parameter der Algorithmen

Übersicht über die experimentell bestimmten Einflüsse der wichtigsten Parameter der Algorithmen VED_i, VED_r, MD_i, MD_r, KD_i und KD_r. Dabei sind die Einflüsse der Parameter bei den Experimenten mit den simulierten Bewegungsdaten und mit den Frequentie1550-Daten in gering und gross klassiert. Zudem sind jeweils die wichtigsten Gründe für dieses Resultat aufgeführt.

8.3.1 Algorithmus VED_i

Die Kennzahlen des Tests des Algorithmus VED_i mit den Simulationsdaten unter Verwendung der idealen Parameter, welche alle in der Tabelle 7.1 aufgelistet sind, weisen hohe Werte auf. Im Speziellen die Inclass Accuracy mit einem Wert von über 11 fiel sehr hoch aus. Zudem sind die Werte der EOC (Error of Commission) und EOO (Error of Omission) sehr tief. Nur gerade eine Situation mit *Verfolgen/Entfliehen* wurde nicht gefunden. Dabei gilt es zu beachten, dass dieses *Verfolgen/Entfliehen* nur gerade 43 Zeitschritte lang aufrechterhalten wurde und somit vom Algorithmus aufgrund des Zeitkriteriums gar nicht gefunden werden konnte. Denn laut den idealen Parametern von *Folgen* (siehe Tabelle 7.1) muss eine Verfolgungssituation im Minimum 65 Sekunden dauern, um als *Verfolgen/Entfliehen* bezeichnet zu werden. Dieses eine Reaktionsbewegungsmuster könnte nur dann gefunden werden, wenn man die kritische Zeitdauer von *Folgen* stark reduzieren würde. Dabei würde man allerdings Gefahr laufen, die Anzahl der zusätzlich fehlerhaft detektierten Bewegungsmuster stark zu erhöhen. Somit muss also an dieser Stelle darauf hingewiesen werden, dass der Algorithmus VED_i Probleme mit der Detektion von sehr kurzen Ereignissen bekundet. Solche Situationen mit *Verfolgen/Entfliehen* über sehr kurze Zeiträume sind in der Realität durchaus denkbar, falls sich das Fluchtobjekt sehr schnell von seinem Verfolger entfernt und dieser als Konsequenz davon aufgibt und die Verfolgung beendet. Alles in allem schneidet der Algorithmus VED_i bei den Experimenten mit den simulierten Bewegungsdaten des Steering Behavior-Modells aber sehr gut ab.

Verzögerungszeit und Frontbereichswinkel

Kommen wir nun zur Diskussion der Forschungsfrage 3 bei der die Abgrenzung des Reaktionsbewegungsmusters *Verfolgen/Entfliehen* zu anderen sich in Raum und Zeit ähnlich äussernden Bewegungsmustern (z.B. *Anführen einer Gruppe* oder zufällig in dieselbe Richtung gehen) im Zentrum steht. Überraschend erscheint bei der Betrachtung der Resultate der Experimente der immense Einfluss des kritischen Frontbereichswinkels und die im Verhältnis dazu geringe Auswirkung der Verzögerungszeit. Bei genauerer Betrachtung der analysierten Bewegungsdaten wird jedoch schnell klar, dass der geringe Einfluss der Verzögerungszeit darauf zurückzuführen ist, dass aufgrund der Konstellationen in den Simulationen im Steering Behavior-Modell keinerlei andere Interaktionsformen zwischen Punktobjekten entstehen konnten. Somit ist eine Abgrenzung von *Verfolgen/Entfliehen* gegenüber anderen Typen von Interaktionen nicht nötig, weshalb die Verzögerungszeit eine untergeordnete Rolle spielt. Ganz im Gegensatz dazu der Parameter Frontbereichswinkel, welcher sich als sehr wichtig erweist bei der Abgrenzung von *Verfolgen/Entfliehen* gegenüber Mustern von zufällig in dieselbe Richtung gehenden Punktobjekten. Durch die Wahl eines kleinen Frontbereichswinkels kann sichergestellt werden, dass sich die Punktobjekte exakt in dieselbe Richtung bewegen müssen, um als *Verfolgen/Entfliehen* bezeichnet zu werden. Die Verzögerungszeit wäre weitaus wichtiger, falls andere Bewegungsmuster wie zum Beispiel *Anführen/Folgen* (Andersson et al. 2008), *Single-File* (Buchin et al. 2008) oder *Herdensbildungen (flock)* (Laube et al. 2004) zwischen den Punktobjekten in den Bewegungsdaten

enthalten wären. Diese Muster weisen in Raum und Zeit alle sehr ähnliche Merkmale wie *Verfolgen/Entfliehen* auf. Bei der Betrachtung der Beschreibungen der Muster *Single-File* in Buchin et al. (2008) und *Anführen/Folgen* in Andersson et al. (2008) wird allerdings klar, dass diese primär für den Fall von interagierenden Punktobjekten derselben Art formalisiert wurden. Dabei wurde auf die Unterteilung des Musters in Aktion und Reaktion und die Einführung eines Parameters Verzögerungszeit verzichtet. Dies mit gutem Grund, da die beteiligten Objekte der gleichen Spezies miteinander interagieren möchten und dabei nicht unter Zeitdruck stehen, wie dies während einer Verfolgungsjagd zwischen einem Räuber und seiner Beute der Fall ist. Möchte ein Räuber seine Beute nicht entkommen lassen, so muss er nach der *Annäherung* sofort reagieren und die *Verfolgung* aufnehmen. Somit sollte die zeitliche Verzögerung zwischen der *Annäherungsbewegung* und dem *Folgen* in der Regel kurz ausfallen. Im Gegensatz dazu wird erwartet, dass die Verzögerungszeit in Interaktionssituationen mit kooperierenden Punktobjekten grösser ausfällt. Als Beispiel kann eine Schafsherde genannt werden, welche sich über einen längeren Zeitraum zusammen am gleichen Ort aufhält und von Zeit zu Zeit unter der Führung des Leitschafes den Weideplatz wechselt.

In den Bewegungsdaten des Frequentie1550-Projekts sind neben *Verfolgen/Entfliehen* noch andere Interaktionen zwischen den Teams möglich. Dazu gehören beispielsweise *Konfrontationen* oder *Kooperationen* zwischen den Teams. Mit Hilfe der Experimente konnte gezeigt werden, dass die Wichtigkeit des Parameters Verzögerungszeit bei den Experimenten mit den Frequentie1550-Daten im Vergleich zu den Experimenten mit den Simulationsdaten aus diesem Grund grösser ist. Je grösser die gewählte kritische Verzögerungszeit, desto mehr Bewegungsmuster wurden detektiert. Aufgrund der fehlenden semantischen Informationen zu den Interaktionen ist es allerdings nicht möglich, diese Bewegungsmuster zu differenzieren. Klar ist aber, dass die Gefahr einer Fehldetektion mit zunehmender Verzögerungszeit ansteigt, da erwartet wird, dass die an *Verfolgen/Entfliehen* beteiligten Punktobjekte sich kurze Zeit nach dem *Zusammentreffen* zu verfolgen beginnen. Für die Herleitung des idealen Wertes für den Parameter Verzögerungszeit sind semantische Informationen jedoch unerlässlich. Eine andere Möglichkeit wäre die Verwendung von Expertenwissen. Im Falle von interagierenden Tieren könnten zum Beispiel Verhaltensbiologen beigezogen werden, um den entsprechenden Grenzwert der erlaubten Verzögerungszeit zu wählen.

Frontbereichswinkel und kritische Distanz von *Folgen*

Weiter konnte der grosse Einfluss der beiden Parameter Frontbereichswinkel und kritische Distanz von *Folgen* durch Experimente mit den Frequentie1550-Daten gezeigt werden. Im Falle des kritischen Frontbereichswinkels konnte durch ein Experiment herausgefunden werden, dass mit zunehmender Grösse des Winkels die Anzahl der detektierten Bewegungsmuster zunimmt. Allerdings läuft man durch eine Erhöhung des Frontbereichs zudem Gefahr, dass Situationen fälschlicherweise als *Verfolgen/Entfliehen* interpretiert werden. Dies ist zum Beispiel bei einem kritischen Frontbereichswinkel von 40 Grad der Fall, falls sich Punktobjekte auf nicht benachbarten und nicht in einer Linie befindlichen Segmenten bewegen und sich, beispielsweise aufgrund der Häuser entlang der

engen Strassen von Amsterdam, gar nicht sehen können. Bei einem solchen Fall kann kaum von willentlichem *Verfolgen/Entfliehen* gesprochen werden. Vielmehr bewegen sich die Punktobjekte zufällig in dieselbe Richtung. Dasselbe Phänomen konnte auch im Falle der kritischen Distanz von *Folgen* gezeigt werden. Bei der Wahl eines Werts für diese beiden Parameter sollte man sich demzufolge an den Rahmenbedingungen und Gegebenheiten, welche bei der Aufzeichnung der Bewegungen vorherrschten, orientieren. Im Falle der Frequenzdaten macht zum Beispiel ein Distanzparameter in der Größenordnung der mittleren Strassensegmentlänge (ca. 58 Meter) Sinn.

Zusammengefasst kann festgehalten werden, dass der Algorithmus VED_i diverse einflussreiche Parameter besitzt, um eine erfolgreiche Abgrenzung zu anderen Bewegungsmustern vorzunehmen. Die Ausführungen haben zudem gezeigt, dass der Algorithmus VED_i sowohl auf simulierte als auch auf reale Bewegungsdaten angewendet werden kann. Einzig die Detektion von sehr kurzen Ereignissen mit *Verfolgen/Entfliehen* bereitet dem Algorithmus einige Mühe.

8.3.2 Algorithmus VED_r

Die in der Tabelle 7.2 aufgelisteten Gütemasse des Tests des Algorithmus VED_r mit den Simulationsdaten des Netzwerkraumes unter Verwendung der idealen Parameter weisen darauf hin, dass der Algorithmus VED_r bei den Experimenten nicht so hohe Werte aufweist wie der Algorithmus VED_i . In der Folge sollen für dieses Resultat Gründe genannt werden. Die Werte der Gütekriterien sind im Wesentlichen darauf zurückzuführen, dass der Algorithmus VED_r Probleme bekundet bei der Unterscheidung zwischen Bewegungsmustern von sich tatsächlich verfolgenden Punktobjekten (laut den semantischen Zusatzinformationen der Simulationsdaten) und Mustern, welche als Folge von zufällig in dieselbe Richtung gehenden Objekten zustandekommen (ohne Semantik). Aufgrund der Verwendung des Strassennetzwerks im Steering Behavior-Modell sind die Bewegungen der Punktobjekte stark eingeschränkt, was dazu führt, dass sich auch die zufällig im Raum bewegendes Punktobjekte gezwungenermassen von Zeit zu Zeit folgen müssen. Dies widerspiegelt sich in einem höheren Wert der EOC. Weiter ist aufgrund der Einstellungen im Steering Behavior-Modell das *Anhalten* der Punktobjekte nicht möglich, was die Abgrenzung weiter erschwert.

Zeitdauer des *Folgens*

Die einzige Möglichkeit ist die Wahl eines sehr grossen Werts für den Parameter kritische Zeitdauer des *Folgens*. Als Konsequenz davon steigen allerdings automatisch auch die nicht gefundenen Bewegungsmuster und somit der Wert der EOC an. Kurze Ereignisse mit *Verfolgen/Entfliehen* können durch den Algorithmus aufgrund des idealen Wertes für den Zeitparameter von *Folgen* von 250 Zeitschritten (siehe Tabelle 7.2) gar nicht gefunden werden. Dies wird bei der Betrachtung der nicht gefundenen Bewegungsmuster bestätigt, welche sich laut den semantischen Zusatzinformationen der Simulationsdaten über eine Zeitdauer von 6 bis 192 Zeitschritten erstrecken. Im Vergleich zum euklidischen Raum ist es also bei Bewegungen auf einem Strassennetzwerk schwieriger, Muster von zufälligen Bewegungen in dieselbe Richtung von Situationen mit *Verfolgen/Entfliehen* abzugrenzen.

Verzögerungszeit

Kommen wir nun zur Diskussion des Parameters Verzögerungszeit. Dieser Parameter ist laut den Ausführungen in Kapitel 7 wichtig, nimmt im Vergleich zur kritischen Zeitdauer von *Folgen* aber eher eine untergeordnete Rolle ein. Wie bereits beim Algorithmus VED_i beschrieben, ist dies primär auf die fehlenden Interaktionen zwischen den Punktobjekten in den Bewegungsdaten des Steering Behaviors zurückzuführen. Folglich ist keine Abgrenzung zu anderen Interaktionstypen notwendig. Ein weiterer möglicher Grund, welcher im Falle des Algorithmus VED_i noch nicht diskutiert wurde, ist die Tatsache, dass die Punktobjekte im Steering Behavior-Modell nicht anhalten. Dies führt dazu, dass sich Punktobjekte nach dem Annähern entweder in dieselbe oder in die entgegengesetzte Richtung bewegen. Die Verzögerungszeit ist also in jedem Fall kurz. Wären auch *Anhaltebewegungen* möglich, könnte eines der beiden Punktobjekte nach dem zufälligen *Annähern* kurz seine Geschwindigkeit reduzieren, für einen Moment stehen bleiben und erst danach sich rein zufällig in die gleiche Richtung wie das andere Punktobjekt bewegen. Dadurch ergäbe sich eine längere Verzögerungszeit und somit die Möglichkeit einer erfolgreichen Abgrenzung. Dem gegenüber beginnt bei *Verfolgen/Entfliehen* auf den Verfolger (z.B. Räuber) bei der Unterschreitung der Aktivdistanz sofort eine anziehende Kraft zu wirken, was mit einer geringen Verzögerungszeit zwischen *Annäherung* und *Verfolgung* einhergeht. Somit müsste der Einfluss der Verzögerungszeit im Falle von realen Bewegungsdaten zunehmen. Dies konnte anhand der Experimente mit den Frequentie1550-Daten, in welchen sowohl *Anhalten*, als auch andere Interaktionsformen zwischen den Spielern vorkommen, bestätigt werden.

Grösser der Segmentnachbarschaft

Anstelle des Parameters der kritischen Distanz von *Folgen* des Algorithmus VED_i tritt im Falle des Algorithmus VED_r die Grösse der Segmentnachbarschaft. Wie mit Hilfe der in Kapitel 7 beschriebenen Experimente und dargestellten Beispiele gut aufgezeigt werden konnte, spielt die Segmentnachbarschaftsgrösse für den Erfolg des Algorithmus VED_r eine entscheidende Rolle. Dabei muss der Wert dieses Parameters unbedingt unter Berücksichtigung der örtlichen Gegebenheiten (z.B. Sichtverhältnisse) gewählt werden. Für den Anwendungsfall der Frequentie1550-Daten macht aufgrund der aufragenden Häuser entlang der engen Strassen in Amsterdam ein Wert von 1 für den Parameter Segmentnachbarschaftsgrösse Sinn. Unter Umständen wäre eine Erweiterung des Algorithmus möglich, mit deren Hilfe die Orientierungsdifferenz zwischen den benachbarten Segmenten geprüft werden könnte. Mit einer solchen Erweiterung könnte sichergestellt werden, dass sich die Punktobjekte im Blickfeld haben und sich nicht nur zufällig in dieselbe Richtung bewegen.

Zusammenfassend kann ausgesagt werden, dass der Algorithmus VED_r sowohl auf simulierte als auch auf reale Bewegungsdaten angewendet werden kann. Die Grenzen des Algorithmus im Falle der simulierten Daten sind vor allem auf die Einstellungen im Steering Behavior-Modell zurückzuführen. Bei der Analyse der Frequentie1550-Daten, in denen sowohl andere Interaktionsformen, als auch *Anhaltebewegungen* enthalten sind, zeigte sich die Anwendbarkeit der Algorithmus VED_r auf reale Bewegungsdaten. Vorsicht ist aber bei der Detektion von kurzen *Verfolgen/Entfliehen* geboten.

8.3.3 Vergleich der Algorithmen VED_i vs. VED_r

Nach der Diskussion der Forschungsfrage 3 für die beiden Algorithmen VED_i und VED_r folgt nun in diesem Abschnitt der Vergleich zwischen diesen beiden Methoden, wobei hinsichtlich der Beantwortung der Forschungsfrage 4 die Unterschiede zwischen den beiden Perspektiven ausgearbeitet werden sollen. Die oben gemachten Ausführungen zeigen, dass sowohl der Algorithmen VED_i als auch der Algorithmus VED_r einige Mühe bekunden bei der Detektion von sehr kurzen Ereignissen mit *Verfolgen/Entfliehen*. Diese Probleme verstärken sich im Falle des Algorithmus VED_r , was sich in den leicht tieferen Werten der Gütekriterien widerspiegelt. Ein grosser Teil dieser Unterschiede ist allerdings darauf zurückzuführen, dass es im Falle der Bewegungen auf einem Netzwerk schwieriger ist, Muster von zufälligen Bewegungen in dieselbe Richtung gegenüber *Verfolgen/Entfliehen* abzugrenzen. Dies ist wiederum die Hauptursache, dass die Bewegungen der Punktobjekte durch das Netzwerk stark eingeschränkt sind und es somit öfters und über längere Zeiträume zu zufälligen Bewegungen in dieselbe Richtung kommt. Für einen adäquaten Vergleich zwischen den beiden Algorithmen müsste auch der Algorithmus VED_i in Experimenten mit den simulierten Bewegungsdaten des Netzwerkraums getestet werden. Alles in allem kann festgehalten werden, dass die Algorithmen bei der Analyse von Netzwerkbewegungsdaten noch Verbesserungspotential aufweisen.

Weiter kann ausgesagt werden, dass die räumliche Variante den Nachteil hat, dass die kleinste unterscheidbare Raumeinheit ein Strassensegment ist. Im Vergleich zur Distanz der individuellen Perspektive, welche viel feinere Justierungen zulässt, sind mit der Variation der Segmentnachbarschaftsgrösse nur grobe Anpassungen möglich. Dies zeigt sich auch aus dem Vergleich zwischen den beiden Diagrammen 7.10 und 7.11, welche den Einfluss der kritischen Distanz von Folgen beziehungsweise der Segmentnachbarschaftsgrösse auf die Anzahl der detektierten Bewegungsmuster des Algorithmus VED_i respektive VED_r aufzeigen. Aus diesem Vergleich geht hervor, dass mit einer Erhöhung der Grösse der Segmentnachbarschaft die Anzahl der detektierten *Verfolgen/Entfliehen* sprunghaft ansteigt, während mit Hilfe der kritischen Distanz ein kontinuierlicher Anstieg der detektierten Bewegungsmuster erreicht werden kann. Die räumliche Perspektive bietet dafür den Vorteil, dass Bewegungen auf parallelen, nahe zueinander liegenden Segmenten ausgeschlossen werden können. Im Falle der individuellen Perspektive besteht die Gefahr, dass solche Ereignisse aufgrund der geringen Distanz zwischen den Punktobjekten als *Verfolgen/Entfliehen* bezeichnet werden. Mit Hilfe des kritischen Frontbereichswinkels besitzt der Algorithmus VED_i allerdings noch ein zusätzliches Kriterium, mit welchem solche Situationen richtig interpretiert werden können.

8.3.4 Algorithmus MD_i

Betrachtet man die Gütemasse des durchgeführten Experiments mit den simulierten Bewegungsdaten und mit den idealen Parametern für den Algorithmus MD_i (siehe Tabelle 7.3), so kann eine durchaus

positive Bilanz gezogen werden. Sowohl die EOO als auch die EOC sind verschwindend klein, was sich in den Massen der Producer's Accuracy und User's Accuracy widerspiegelt. Beide Gütemasse zeigen sehr hohe Werte. Ebenfalls ein sehr hoher Wert weist die Inclass Accuracy auf, was als Folge der guten Balance zwischen der Anzahl der erfolgreich detektierten Muster und den EOO und EOC zustande gekommen ist.

Verzögerungszeit

Nun stellt sich noch die Frage nach den entscheidenden Parametern für die Abgrenzung von *Meiden* gegenüber anderen Bewegungsmustern mit Hilfe des Algorithmus MD_i (Forschungsfrage 3). Die Erläuterungen der Experimentresultate in Kapitel 7 haben gezeigt, dass mit Hilfe der Verzögerungszeit während den Experimenten mit den simulierten Bewegungsdaten des Steering Behavior-Modells keine optimale Lösung erreicht werden kann. Trotz starker Reduktion der Verzögerungszeit gelingt es nicht, den Wert der EOC weiter zu verringern, ohne dabei eine starke Erhöhung der EOO in Kauf nehmen zu müssen. Der geringe Einfluss der Verzögerungszeit ist allerdings nicht weiter verwunderlich, da in den verwendeten Simulationsdaten keinerlei Interaktionsformen enthalten sind, welche abgegrenzt werden müssten. Neben Situationen mit *Meiden* war während der Simulation nur noch zufälliges *Kreuzen von Bewegungspfaden* möglich. Da sich solche zufälligen Annäherungssituationen aufgrund der fehlenden Interaktionen auch über relativ kurze Zeiträume abspielen, ist eine Abgrenzung mit der Zeitverzögerung zwischen *Annäherung* und *Separation* nur schwer möglich. Bei den Tests mit den Frequentie1550-Daten konnte dagegen gezeigt werden, dass die Verzögerungszeit sehr entscheidend sein kann. Mit Hilfe der semantischen Informationen der *Konfrontationen* konnte nachgewiesen werden, dass bei einer Erhöhung der erlaubten Verzögerungszeit zwischen *Zusammentreffen* und *Separieren* zusehends mehr Situationen als *Meiden* deklariert wurden, obwohl in Tat und Wahrheit eine *Konfrontation* stattgefunden hat. Mit steigender Verzögerungszeit stieg daher der Wert der EOC stetig an. Mit Hilfe dieses Experiments konnte die Bedeutung des Parameters Verzögerungszeit bei der Abgrenzung zwischen *Meiden* und *Konfrontation* untermauert werden.

Winkel der Bewegungsrichtungsänderung

Für die Abgrenzung der beschriebenen Situationen mit zufälligem *Kreuzen der Bewegungspfade* in den simulierten Daten des Steering Behavior-Modells kann der kritische Winkel der Bewegungsrichtungsänderung herbeigezogen werden. Da sich meidende Punktobjekte bei einem *Zusammentreffen* abstossen, kommt es zu einer abrupten Bewegungsrichtungsänderung. Mit Hilfe dieses plötzlichen Richtungswechsels kann das Reaktionsbewegungsmuster *Meiden* gegenüber zufälligem *Aufeinandertreffen*, bei welchem sich die beteiligten Punktobjekte in ihren Bewegungen nicht beeinflussen, sehr effizient abgegrenzt werden. Der Einfluss des Parameters Richtungswechselwinkel wurde auch in Experimenten mit den Frequentie1550-Daten geprüft. Dabei hat die Variation der kritischen Bewegungsrichtungsänderung im Falle des Frequentie1550-Datensatzes kaum einen Einfluss auf die Resultate des Algorithmus MD_i. Dies ist darauf zurückzuführen, dass die Punktobjekte während den Bewegungen auf dem Strassennetzwerk zu

Richtungswechseln gezwungen werden. Somit kann der Richtungswechselwinkel nur im Falle von Bewegungen in einem euklidischen Raum wirklich zur Abgrenzung des Reaktionsbewegungsmusters *Meiden* beitragen.

Minimaldistanz von *Meiden*

Der letzte Parameter, welcher genauer unter die Lupe genommen wird, ist die Minimaldistanz von *Meiden*. Wie in Kapitel 7 anhand der Resultate der Experimente mit den Frequentie1550-Daten aufgezeigt werden konnte, ist dieser Parameter sehr effektiv bei der Abgrenzung zwischen *Meiden* und *Konfrontationen*. Denn bei *Konfrontationen* müssen die Spieler ineinander hineinrennen und sich dadurch unweigerlich sehr nahe kommen. Mit einer Erhöhung der erlaubten Minimaldistanz von *Meiden* kann verhindert werden, dass Situationen, in denen eine *Konfrontation* stattfindet, fälschlicherweise als *Meiden* deklariert werden. Somit kann mit Hilfe der Minimaldistanz von *Meiden* der Ausfall des Parameters Richtungswechselwinkel im Falle von Netzwerkdaten kompensiert werden. Dies führt zum Fazit, dass der Algorithmus sowohl auf simulierte als auch auf reale Bewegungsdaten angewendet werden kann.

8.3.5 Algorithmus MD_r

Der Algorithmus MD_r wurde anhand von Experimenten mit simulierten Bewegungen in einem Netzwerkraum im Steering Behavior-Modell getestet. Die Resultate des Durchlaufs mit den idealen Parametern ergeben im Vergleich zum Algorithmus MD_i tiefere Werte für alle Gütekriterien. Grund dafür ist die ungenügende Abgrenzung von Situationen mit *Meiden* gegenüber Mustern von zufälligen Bewegungen auf dem Netzwerk. In der Folge soll die Diskussion über die genauen Zusammenhänge der wichtigsten beiden Parameter Verzögerungszeit und Bewegungsrichtungsänderung Aufschluss geben über die gewonnen Erkenntnisse. Dabei sollen die Ursachen der tiefen Werte der Gütekriterien erörtert werden.

Verzögerungszeit

Der Parameter Verzögerungszeit hat im Falle der simulierten Bewegungsdaten des Steering Behavior-Modells erwartungsgemäss keinen grossen Einfluss aufgrund der bis anhin mehrfach genannten Ursache, dass keine Interaktionsformen durchgeführt wurden. Wie es bei allen bis anhin betrachteten Algorithmen der Fall war, so nimmt die Bedeutung dieses Parameters auch für den Algorithmus MD_r bei der Analyse der Frequentie1550-Daten zu.

Winkel der Bewegungsrichtungsänderung

Der kritische Winkel der Bewegungsrichtungsänderung stellte sich bei den Experimenten mit den simulierten Bewegungsdaten als wichtig heraus. Wie die Resultate der Experimente in Kapitel 7 zeigen, hat allerdings auch der Einfluss dieses Parameters bei Bewegungen auf einem Netzwerk seine Grenzen. Dies aufgrund der Tatsache, dass es durch das Abbiegen an Segmentübergängen schnell einmal zu abrupten Richtungsänderungen kommt. Solche Wechsel in der Bewegungsrichtung werden

vorwiegend durch die eingeschränkte Bewegungsfreiheit auf einem Netzwerk hervorgerufen. Dieselben Zusammenhänge zeigten sich bei den Experimenten mit den Frequentie1550-Daten. Diese Ausführungen zeigen, dass die vorgeschlagene Methode im Falle von Bewegungen auf einem Strassennetzwerk noch nicht wunschgemäss funktioniert. Weitere Überlegungen sind notwendig, um den Algorithmus auch im Falle eines Netzwerks anwendbar zu machen. Eine Idee ist der Einbezug der Orientierungsdifferenzen zwischen den Segmenten. Sobald ein Punktobjekt sich von einem Segment auf ein anderes bewegt, könnte der Abbiegewinkel mit Hilfe der Differenz der Orientierungen der beiden Segmente korrigiert werden. Diese Idee wurde mit einer Erweiterung des Algorithmus geprüft. Mit deren Hilfe wird bei einer Überschreitung des kritischen Winkels der Bewegungsrichtungsänderung untersucht, ob ein Übergang zwischen zwei Segmenten stattgefunden hat. Dadurch kann der Wert der EOC unter Verwendung der idealen Parameter reduziert werden. Es muss allerdings in Kauf genommen werden, dass im Gegenzug der Wert der EOO ansteigt. Daraus wird ersichtlich, dass der Abgrenzungsversuch zwischen tatsächlichen *Meiden*-Ereignissen und Abbiegen auf dem Strassennetzwerk nicht so einfach ist. Die Zunahme der EOO ist darauf zurückzuführen, dass es im Falle von *Meiden* häufig dazu kommt, dass sich separierende Punktobjekte an Kreuzungen auf andere Segmente verdrängen lassen. Immerhin kann mit dieser Erweiterung des Algorithmus die Anzahl der fehlerhaft detektierten Bewegungsmuster ein wenig reduziert werden. Die verbleibenden EOC kommen zu Stande, da aufgrund des sehr einfachen Map-Matching Algorithmus der Segmentwechsel mit dem Richtungswechsel nicht in jedem Fall gleichzeitig stattfindet. Die Verwendung eines besseren Map-Matching Verfahrens (z.B. das Verfahren von Greenfeld (2002)) würde die EOC weiter reduzieren können. Diese Erläuterungen zeigen, dass noch weitere Überlegungen angestellt werden müssen, um die Methode weiter auszureifen.

8.3.6 Vergleich der Algorithmen MD_i vs. MD_r

Für die Bearbeitung der Forschungsfrage 4 werden die beiden Algorithmen zur Detektion von *Meiden* der individuellen und der räumlichen Perspektive miteinander verglichen. Der wichtigste Unterschied der beiden Perspektiven betrifft die Anwendbarkeit auf Bewegungsdaten, welche sich auf einem Netzwerk abspielen. Anhand der Frequentie1550-Daten konnte gezeigt werden, dass mit Hilfe der beiden Parameter Verzögerungszeit sowie Minimaldistanz von *Meiden* das Reaktionsbewegungsmuster *Meiden* durch den Algorithmus MD_i sehr erfolgreich gegenüber *Konfrontationen* abgegrenzt werden konnte. Mit Hilfe des Algorithmus MD_r gelingt eine solche Abgrenzung nur ungenügend und zwar mit dem Parameter Verzögerungszeit. Zudem ist der Parameter kritischer Richtungswechselwinkel aufgrund der genannten Ursachen ebenfalls nicht auf die Bewegungen auf einem Netzwerk anwendbar. Dazu kommt, dass ein vergleichbarer Parameter wie die Minimaldistanz von *Meiden* beim Algorithmus MD_r nicht zur Verfügung steht, da das Strassensegment die kleinste mögliche Raumeinheit darstellt. Als Schlussfolgerung kann ausgesagt werden, dass der Algorithmus MD_r in dieser Form noch nicht die gewünschten Resultate liefert. Im Vergleich dazu erwies sich der Algorithmus MD_i als sehr guter Ansatz, um sowohl in

Bewegungsdaten eines euklidischen Raums, als auch in Bewegungen auf einem Netzwerk das Reaktionsbewegungsmuster *Meiden* zu detektieren.

8.3.7 Algorithmus KD_i

Wie in Kapitel 7 bei der Beschreibung der Resultate bereits ausgeführt, schneidet der Algorithmus KD_i relativ schlecht ab. Dies widerspiegelt sich in den tiefen Werten der Gütekriterien. In den folgenden Abschnitten sollen die Schwächen des Algorithmus KD_i sowie ihre Ursachen diskutiert werden. Den Anfang bilden die Erklärungen für den hohen Wert der EOO.

Fehlerhafte Bewegungsdaten

Eine wichtige Ursache ist der Umstand, dass sich die Punktobjekte in den fehlerhaften Bewegungsdaten nicht genug annähern und somit die Daten die Realität nicht zureichend abbilden. Dies konnte anhand eines Experiments mit dem Ansatz von Orellana et al. (2009) zur Detektion von *Approximationen*, bei welchem die kritische Distanz variiert wurde, gezeigt werden. Aus den in Diagramm 7.26 dargestellten Resultaten dieses Experiments geht hervor, dass sogar bei einer kritischen Distanz von über 100 Metern noch nicht alle *Konfrontationen* gefunden werden (eine der *Konfrontationen* wird erst ab einer Distanz von 250 Metern gefunden). Weiter kann festgehalten werden, dass sich Punktobjekte in sechs *Konfrontationen* laut den Bewegungsdaten nicht über eine Zeitspanne von 30 Sekunden unter die kritische Distanz von 50 Meter (Parameterwerte laut Orellana et al. (2009)) annähern. Somit kann ein Teil der EOO begründet werden.

Zeitdauer des *Getrenntseins*

Für die Begründung der restlichen Fehler wird in der Folge der Parameter Zeitdauer des Bewegungsmusters *Getrenntsein* genauer unter die Lupe genommen. Zu diesem Zweck wurde ein Experiment durchgeführt, bei welchem dieser Parameter variiert wurde. Als Quintessenz dieses Experiments geht hervor, dass die Abgrenzung des Reaktionsbewegungsmusters *Konfrontation* gegenüber anderen Interaktionsformen nicht wunschgemäss funktioniert. Dies ist darauf zurückzuführen, dass sich die Punktobjekte nach einer *Konfrontation* nicht wie erwartet voneinander wegbewegen und somit die in Kapitel 4 vorgestellte Formalisierung dieses Musters nicht optimal ist. Jedoch ist auch in diesem Fall ein Teil der EOO auf die schlechte Qualität der Bewegungsdaten zurückzuführen. In manchen Fällen standen nach einer *Konfrontation* für eines der beiden beteiligten Punktobjekte keine Bewegungsdaten mehr zur Verfügung, was bedeutet, dass eine mögliche Separationsbewegung in den Daten nicht abgebildet ist und somit die *Konfrontation* nicht als solche detektiert werden kann. Dieser Umstand zeigt ein Experiment mit einem leicht abgeänderten Algorithmus, bei welchem auch Situationen während denen keine Bewegungsdaten zur Verfügung stehen als *Getrenntsein* deklariert werden. Mit Hilfe dieses abgeänderten Algorithmus wurden drei *Konfrontationen* mehr richtig detektiert. Somit sind die EOO aber mehrheitlich darauf zurückzuführen, dass sich die Punktobjekte nach einer *Konfrontation* nicht voneinander wegbewegen und somit die in dieser Arbeit vorgestellte Formalisierung von *Konfrontationen* unpassend ist.

Verzögerungszeit und Intersektionsdistanz

Eine weitere Möglichkeit besteht darin, das Abschneiden des Algorithmus mit Hilfe der Verzögerungszeit und der kritischen Intersektionsdistanz zu verbessern. Um den Erfolg dieser beiden Ansätze zu prüfen, wurde je ein Experiment durchgeführt, bei welchen einer dieser beiden Parameter variiert wurde. Für beide Parameter konnte gezeigt werden, dass keine ideale Abgrenzung zwischen *Konfrontationen* und anderen Bewegungsmustern gelingt. Im Falle der Verzögerungszeit kann dieser Umstand mit der fehlenden Separationsbewegung erklärt werden. Fehlt eine solche *Separation*, so kann die erlaubte Verzögerungszeit noch so gross sein, ohne dass eine *Konfrontation* detektiert werden kann. Für den Parameter Intersektionsdistanz kann wie im Falle der kritischen Distanz des *Zusammenseins* mit der schlechten Qualität der Daten argumentiert werden. Aufgrund des Umstandes, dass sich die Punktobjekte in den Bewegungsdaten des Frequentie1550-Projekts während vielen *Konfrontationen* nicht genug annähern, ist die Verwendung einer tiefen Intersektionsdistanz auch nicht erfolgreich.

Formalisierung

Wie diese Ausführungen zeigen, gibt es viele Erklärungsansätze für das schlechte Abschneiden des Algorithmus KD_i bei den durchgeführten Experimenten. Die mangelhafte Datenqualität sowie die ungenügende Formalisierung des Reaktionsbewegungsmusters *Konfrontation* sind dabei die Hauptursachen. Im Falle einer besseren Datenqualität wären die Distanzparameter sicherlich wesentlich einflussreicher bei der Abgrenzung gegenüber anderen Bewegungsmustern. Die ausbleibenden Separationsbewegungen könnten aber auch mit Hilfe von besseren Daten nicht erzwungen werden. Vielmehr scheint die Formalisierung von *Konfrontationen* im Falle des Frequentie1550-Projekts nicht immer zutreffend zu sein. In anderen Anwendungsbereichen, wie beispielsweise der Tierverhaltensforschung, scheint die *Separation* zweier Tiere nach einer *Konfrontation* logisch, da sich zwei Feinde nach einer *Konfrontation* selten zusammen aufhalten wollen. Im Falle der Schüler in einem Freiluftspiel ist es denkbar, dass sich einzelne Teams primär auf *Konfrontationen* konzentrieren und sich weniger für das Suchen von Checkpoints und das Lösen von Rätseln begeistern lassen. Dies konnte laut Admiraal et al. (2007) bei gewissen Teams beobachtet werden. Daraus folgen mehrere nacheinander folgende *Konfrontationen* ohne Separationsbewegung, welche mit dem Algorithmus KD_i nicht gefunden werden können. Zudem ist es auch denkbar, dass sich Teams nach einer *Konfrontation* auf eine *Kollaboration* einigen können, um schneller neue Checkpoints zu finden. In einem Spiel sind diese Schüler zwar Gegner, müssen bei einem Zusammentreffen aber nicht gegenseitig um ihr Leben fürchten, weshalb gemeinsame Bewegungen in Form von Gruppen nicht auszuschliessen sind. Daraus ergibt sich die Schlussfolgerung, dass gerade im Falle von Tierbewegungsdaten der Algorithmus noch bessere Resultate liefern könnte, da die gemachte Formalisierung besser auf die Verhältnisse in der Natur passt. Um diese Vermutung zu belegen, müssten aber zuerst noch Experimente mit Datensätzen von Tierbewegungen durchgeführt werden, was im Anhang dieser Arbeit thematisiert wird.

An dieser Stelle soll darauf hingewiesen werden, dass auch im Falle einer verbesserten Datenqualität noch einige Probleme zu bewältigen wären. Denn neben dem Reaktionsbewegungsmuster *Konfrontation* gibt es noch viele andere Interaktionsformen, bei denen sich Punktobjekte sehr nahe kommen. Somit wird die Abgrenzung durch eine bessere Qualität der Bewegungsdaten nicht unbedingt einfacher. Aus diesem Grund müssten die genauen Gesetzmässigkeiten der kleinräumigen Bewegungen von *Konfrontationen* noch genauer studiert und formalisiert werden. Als Beispiel können die Verhaltensweisen von Tieren in einem Kampf genannt werden, welche zu formalisieren sind, um solche *Konfrontationen* zu anderen Interaktionsformen abzugrenzen. Im Falle des Freiluftspiels des Frequentie1550-Projekts wäre dieses Vorhaben schwierig, da sich die *Konfrontationen* lediglich dadurch auszeichnen, dass Spieler ineinander hineinrennen. Ob sich *Konfrontationen* nur mit diesem Merkmal gegenüber anderen Interaktionsformen abgrenzen lassen, ist ungewiss.

8.3.8 Algorithmus KD_r

Um sich einen Eindruck der Schwächen und Stärken des Algorithmus KD_r zu machen, können die in der Tabelle 9.6 aufgelisteten Werte für die Gütekriterien herbeigezogen werden. Bei der Betrachtung dieser Werte wird klar, dass der Algorithmus KD_r wie schon sein Pendant der individuellen Perspektive in den Experimenten nicht sehr gut abschneidet. Die unbefriedigenden Resultate des Algorithmus KD_r haben ihre Ursachen primär in der ungenügenden Abgrenzung zwischen *Konfrontationen* und anderen Interaktionsformen, was sich im hohen Wert der EOC widerspiegelt.

Verzögerungszeit und Zeitdauer von *Getrenntsein*

Somit sind wir nun bei der Diskussion der Einflüsse der wichtigsten Parameter bei der Abgrenzung von *Konfrontationen* und daher bei der Beantwortung der Forschungsfrage 3 angelangt. Für den Algorithmus KD_r ergeben sich aus den Resultaten der durchgeführten Experimente im Grossen und Ganzen die gleichen Erkenntnisse wie im Falle des Algorithmus KD_i , weshalb die Diskussion an dieser Stelle kurz ausfällt. Die Diagramme in Kapitel 7 zeigen, dass sowohl der Parameter Verzögerungszeit, als auch die kritische Zeitdauer von *Getrenntsein* keine erfolgreiche Abgrenzung von *Konfrontationen* gegenüber anderen Interaktionsformen erlauben. Dies ergibt also die gleiche Erkenntnis wie für den Algorithmus KD_i .

Zeitdauer der *Intersektion*

Interessant ist allerdings der Parameter Zeitdauer der *Intersektion*, welcher anstelle des nicht zur Verfügung stehenden Parameters kritische Intersektionsdistanz des Algorithmus KD_i untersucht wurde. Während beim Algorithmus KD_i das *Zusammentreffen* mit Hilfe der Unterschreitung einer Intersektionsdistanz gekennzeichnet wird, wird im Falle des Algorithmus KD_r die Bewegung auf dem gleichen Segment aufeinander zu verwendet. Wie die Ausführungen in Kapitel 7 zeigen, konnte aber auch mit der Zeitdauer der *Intersektion* keine erfolgreiche Abgrenzung erzeugt werden. Dies ist primär auf die schlechte Datenqualität zurückzuführen. Wären die Bewegungsdaten nicht so sprunghaft, so würde mit diesem Parameter ein gutes Kriterium zur Verfügung stehen. Die Wahl des sehr tiefen

idealen Werts für diesen Parameter von 5 zeigt, dass dieses Kriterium im Falle der Frequentie1550 nicht wirklich angewendet werden kann.

Die Gründe für diese mangelhafte Trennung zwischen *Konfrontationen* und anderen Typen von Interaktionen wurden bei der Diskussion des Algorithmus KD_i ausführlich besprochen, weshalb sie an dieser Stelle nur noch zusammengefasst werden: Die Hauptgründe sind die schlechte Qualität der Frequentie1550-Daten sowie die in manchen Fällen fehlenden Separationsbewegungen zwischen den beteiligten Punktobjekten.

8.3.9 Vergleich der Algorithmen KD_i vs. KD_r

Beim Vergleich der beiden Algorithmen zur Detektion von *Konfrontationen* fällt auf, dass beide während den Experimenten keine überzeugenden Resultate erzielen konnten. Das etwas bessere Abschneiden des Algorithmus KD_r ist möglicherweise darauf zurückzuführen, dass der räumliche Massstab mit einer Segmentnachbarschaftsgrösse von 1 doch deutlich höher gewählt wurde, als dies beim Algorithmus KD_i mit einer kritischen Distanz von 50 Metern der Fall ist. Dadurch konnten mehr *Konfrontationen* erkannt werden, allerdings ergaben sich als Folge davon auch leicht mehr Fehldetektionen.

Ein weiterer Unterschied zwischen den beiden Algorithmen betrifft die Detektion von *Intersektionen*. Dabei wurden zwei unterschiedliche Ansätze gewählt, welche es zu diskutieren gilt. Auf der einen Seite wurde im Fall der individuellen Perspektive für die *Intersektion* ein Distanzkriterium gewählt, wobei bei der Unterschreitung dieser kritischen Distanz von einem gegenseitigen *Ineinanderlaufen* gesprochen wird. Dieses Kriterium wird aufgrund der schlechten Datenqualität sehr selten eingehalten. Auf der anderen Seite konnten für die räumliche Perspektive keine Distanzbeziehungen untersucht werden. Zudem ist der Aufenthalt auf dem gleichen Strassensegment nicht gerade ein guter Indikator für eine *Konfrontation*, wenn man sich vor Augen führt, dass die durchschnittliche Segmentlänge in der Innenstadt von Amsterdam 58 Meter beträgt. Für die Detektion von *Intersektionen* wurde aus diesem Grund auf einen anderen Ansatz ausgewichen. Und zwar wurde eine *Intersektion* als eine Bewegung aufeinander zu formalisiert, da die Spieler im Freiluftspiel Frequentie1550 ineinander hineinrennen mussten, um eine *Konfrontation* zu starten. Diese unterschiedlichen Ansätze könnten auch zu den leichten Differenzen in den Resultaten geführt haben. Da aber weder das Distanzkriterium beim Algorithmus KD_i oft eingehalten wurde, noch das *Annäheren* auf dem gleichen Segment oft stattgefunden hat, ist der Unterschied zwischen den Algorithmen äusserst gering.

8.4 Einordnung der Erkenntnisse in den Forschungskontext

Wie bereits in den einleitenden Erläuterungen dieses Kapitels angesprochen, soll zum Abschluss der Diskussion die in dieser Arbeit errungenen neuen Erkenntnisse in den Forschungskontext eingeordnet werden. In einem ersten Abschnitt soll dabei kurz auf die Abgrenzung zu anderen Forschungen in der GIScience zur automatisierten Detektion von Bewegungsmustern eingegangen werden. In einem

zweiten Abschnitt werden die in Kapitel 2.8 beschriebenen Forschungslücken noch einmal aufgegriffen und dabei der Mehrwert, der durch diese Arbeit geschaffen wurde, diskutiert.

8.4.1 Vergleich mit anderen Forschungsarbeiten

Wie das Kapitel 2 anhand der Ausführungen zu den wissenschaftlichen Hintergründen dieser Arbeit sehr gut aufzeigen konnte, wurden im Bereich der GIScience bis anhin vorwiegend Methoden zur Detektion von Bewegungsmustern mit Beteiligung von einer Art von sich bewegenden Punktobjekten entwickelt. Ein solches Beispiel ist die Arbeit von Laube et al. (2004), in welcher Algorithmen zur Detektion von *Herdenbildung (flock)*, *Führungsverhalten (leadership)*, *Konvergenz (convergence)* und *Zusammentreffen (encounter)* vorgeschlagen werden. Im Gegensatz zu solchen Bestrebungen wurde in der vorliegenden Arbeit versucht, Ansätze zur Detektion von Interaktionen zwischen zwei unterschiedlichen Arten von Punktobjekten zu entwickeln. Für diesen Zweck wurden unterschiedliche Parameter vorgeschlagen und geprüft, um eine erfolgreiche Abgrenzung zu den herkömmlichen Bewegungsmustern sicherzustellen. Beispielsweise konnte anhand des diskutierten Parameters Verzögerungszeit eine erfolgreiche Abgrenzung zwischen *Verfolgen/Entfliehen* und anderen Interaktionsformen, wie beispielsweise das *Führungsverhalten* und *Folgen* (Andersson et al. 2008) oder die *Bewegungen in einer Reihe (Single-File)* (Buchin et al. 2008), sichergestellt werden.

Ebenfalls gegenüber den Ansätzen von Orellana et al. (2009), welche primär auf explorativen Visualisierungstechniken basieren, konnte eine klare Abgrenzung gemacht werden. Zu diesem Zweck wurde die in Orellana et al. (2009) vorgeschlagene Methode zur Erkennung von *Approximationen* in einem separaten Algorithmus umgesetzt und mit Hilfe der Frequentie1550-Daten einigen Experimenten unterzogen. Die Resultate dieser Tests sind in der Tabelle 7.7 und dem Diagramm 7.26 dargestellt. Beim Vergleich dieser Resultate mit denjenigen der beiden in dieser Arbeit vorgeschlagenen Algorithmen zur Erkennung von *Konfrontationen* wird klar, dass der Ansatz von Orellana et al. (2009) zwar beinahe alle in den semantischen Zusatzdaten enthaltenen *Konfrontationen* findet, dabei allerdings einen viel höheren Wert der EOC in Kauf nehmen muss. Somit kann ausgesagt werden, dass diese Methode grösste Mühe hat bei der Abgrenzung von *Konfrontationen* gegenüber Events, bei denen sich Punktobjekte annähern. An dieser Stelle soll noch einmal darauf hingewiesen werden, dass dies auch nicht das Ziel der Methode von Orellana et al. (2009) ist. Orellana et al. (2009) versucht mit Hilfe eines *Approximations*-Detektors lediglich Interaktionen zwischen Punktobjekten zu finden, ohne dabei Aussagen über die genauen Verhaltensformen zu machen. Orellana et al. (2009) vermerken am Ende ihrer Arbeit, dass mit weiteren Analysen der genauen Bewegungen nach einer *Approximation* das Verständnis der genauen Interaktion zwischen den Punktobjekten verbessert werden könnte. Dabei könnte beispielsweise unterschieden werden, ob eine *Konfrontation* oder eine *Kollaboration* stattgefunden hat. Diese Ausführungen zeigen, dass die in dieser Arbeit vorgestellten Algorithmen weitergehen und versuchen, die Art der Interaktionen zwischen den sich annähernden Punktobjekten zu bezeichnen. Mit Hilfe der Algorithmen zur Detektion von *Verfolgen/Entfliehen*,

Konfrontation und *Meiden* kann versucht werden, eine Aussage über die Interaktionsformen zwischen interagierenden Punktobjekten zu machen. Somit bestehen klare Unterschiede zwischen den bis anhin entwickelten Ansätzen zur Erkennung von Interaktionen und der in dieser Arbeit vorgeschlagenen Methoden zur Detektion von Reaktionsbewegungsmustern.

8.4.2 Aufgreifen der Forschungslücken

Am Ende des Kapitels 2 im Abschnitt zu den Forschungslücken konnte festgehalten werden, dass bis anhin sowohl in der GIScience, als auch in den bekannten Anwendungsbereichen (z.B. Biologie, Überwachung und Sicherheit) noch kaum automatisierte Verfahren zur Detektion von Bewegungsmustern mit Beteiligung von verschiedenen Arten von sich bewegenden Punktobjekten entwickelt worden sind. Ein erster Schritt in Richtung der Schliessung dieser Forschungslücke wurde in dieser Arbeit mit Hilfe von Algorithmen zur Detektion der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* gemacht. Die Experimente mit den Algorithmen haben gezeigt, dass die Algorithmen für die Bewegungsmuster *Verfolgen/Entfliehen* und *Meiden* erfolgreich auf reale Daten angewendet werden können. Im Falle der Algorithmen zur Detektion von *Konfrontationen* sind noch zusätzliche Überlegungen nötig, um eine tadellose Anwendbarkeit sicherzustellen. Weiter konnte gezeigt werden, dass alle Algorithmen einflussreiche Parameter besitzen, mit Hilfe derer eine erfolgreiche Abgrenzung zu anderen Bewegungsmustern möglich ist. Als Fazit kann also ausgesagt werden, dass diese Arbeit im Bestreben einen ersten Beitrag zur Schliessung der diskutierten Forschungslücken zu machen, erfolgreich war.

9 Schlussfolgerungen

9.1 Zusammenfassung

Zu Beginn dieser Arbeit wurden die wichtigsten in der Literatur der Tierverhaltensforschung bekannten Reaktionsbewegungsmuster in Form eines Überblicks vorgestellt. Aus dieser Liste wurden die drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Meiden* und *Konfrontation* ausgewählt, auf ihre raum-zeitlichen Merkmale hin untersucht und diese in einer formalen Beschreibung der Bewegungsabläufe zusammengefasst. Bei der Formalisierung wurden sowohl die individuelle als auch die räumliche Perspektive verwendet. Diese erarbeiteten Formalisierungen bildeten die Grundlage für die Entwicklung der Algorithmen zur Detektion der Reaktionsbewegungsmuster *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden*. Dabei wurden für jedes der drei Muster ein Algorithmus für die individuelle und einer für die räumliche Perspektive ausgearbeitet. Diese Algorithmen setzen sich aus der Detektion von Teilmustern zusammen, wobei zusätzlich noch weitere Kriterien (z.B. die zeitliche Verzögerung zwischen den einzelnen Mustern) geprüft werden. Als Beispiel kann der Algorithmus zur Detektion von *Verfolgen/Entfliehen* genannt werden, welcher sich zusammensetzt aus der Entdeckung der beiden Bewegungsmuster *Zusammensein* und *Folgen*. Um sicherzustellen, dass sich diese beiden Muster zu einem Reaktionsbewegungsmuster *Verfolgen/Entfliehen* zusammensetzen, wird zudem die Verzögerungszeit zwischen dem Start des *Zusammenseins* und dem Beginn des *Folgens* untersucht. Die entwickelten Algorithmen wurden anschliessend durch Experimente mit simulierten Bewegungsdaten und mit einem realen Datensatz eines Freiluftspiels in Amsterdam (Frequentie1550-Projekt) auf deren Anwendbarkeit hin überprüft. Dabei war es das Ziel, die Grenzen der in dieser Arbeit vorgestellten Algorithmen ausfindig zu machen und zudem die Einflüsse der verschiedenen Parameter zu testen. Insbesondere wurde untersucht, inwiefern die Parameter einen Beitrag zu einer erfolgreichen Unterscheidung zwischen den Reaktionsbewegungsmustern und anderen Interaktionsformen leisten können. Für diese Untersuchungen standen sowohl für die Simulationsdaten (für *Verfolgen/Entfliehen* und *Meiden*), als auch für die Frequentie1550-Daten (für *Konfrontationen*) semantische Zusatzinformationen über die sich im Datensatz abspielenden Reaktionsbewegungsmuster zur Verfügung.

9.2 Wichtigste Erkenntnisse

Eine erste wichtige Aussage dieser Arbeit ist die Erkenntnis, dass für eine erfolgreiche Differenzierung von verschiedenen Reaktionsbewegungsmustern zwischen interagierenden Punktobjekten die Bewegungen auf feiner zeitlicher und räumlicher Skala untersucht werden müssen. Dies konnte anhand der Ausführungen über das Reaktionsbewegungsmuster *Konfrontation* gezeigt werden. Im Falle von kämpferisch geprägten *Konfrontationen* kommt es zu schnellen Angriffs- und Verteidigungsbewegungen. Solche Bewegungen spielen sich auf einer kleinen räumlichen und zeitlichen Skala ab. Will man eine Unterscheidung von *Konfrontation* zu anderen Interaktionsformen

(z.B. *elterliche Verteidigung, Spielen, Kooperation*) sicherstellen, so ist die Formalisierung dieser Bewegungen wichtig. Führt man sich aber vor Augen, dass es sehr viele unterschiedliche Ausprägungen von *Konfrontationen* gibt (z.B. Schlägerei zwischen Menschen, verschiedenste Formen von Kämpfen zwischen Tieren), wird unweigerlich klar, dass die Ableitung von allgemeingültigen Gesetzmässigkeiten in solchen Bewegungen nicht einfach ist. Aus diesem Grund wurde das Reaktionsbewegungsmuster *Konfrontation* stark abstrahiert als Abfolge von *Annähern, Zusammentreffen* und *Separieren* beschrieben. Somit wurden bei der Formalisierung bewusst die kleinräumigen Bewegungen vernachlässigt und nur die universell gültigen Gesetzmässigkeiten miteinbezogen. Dies führt zu einer grösseren Anwendbarkeit dieser Formalisierung, welche aber im Falle von vielen ähnlichen Mustern in einem Datensatz an ihre Grenzen stösst. Somit wird der Trade-off zwischen breiter Anwendbarkeit und erfolgreicher Differenzierung zwischen verschiedenen Mustern klar. Je mehr kleinräumige Bewegungen in einer Formalisierung enthalten sind, desto besser gelingt die Unterscheidung zwischen möglichen Interaktionsformen. Im gleichen Zug leidet aber auch die Allgemeingültigkeit der Formalisierung.

Einhergehend mit der Formalisierung der Bewegungen auf immer feineren Skalen sind auch die zunehmenden Anforderungen an die Datenqualität. Die Ansprüche an die Bewegungsdaten bei der Analyse von Reaktionsbewegungsmustern sind sehr hoch, da sich verschiedene Interaktionsverhalten in Raum und Zeit nur auf einer sehr kleinen Skala unterscheiden lassen. Dies führt dazu, dass die Bewegungsdaten sowohl zeitlich wie auch räumlich eine sehr hohe Auflösung haben müssen. Dies konnte anhand der Experimente mit den Frequentie1550-Daten gezeigt werden. Viele *Konfrontationen* konnten nicht gefunden werden, da die Annäherungsbewegungen zwischen den Punktobjekten durch die Daten nicht genügend gut abgebildet waren.

Die Zusammensetzung der Algorithmen zur Detektion der drei Reaktionsbewegungsmuster *Verfolgen/Entfliehen, Konfrontation* und *Meiden* aus einer Abfolge von Methoden zur Entdeckung von elementaren Bewegungsmustern, hat sich bewährt. Wichtig für die Entwicklung von erfolgsversprechenden Algorithmen ist das Finden von Kriterien, welche eine Unterscheidung zwischen den formalisierten Reaktionsbewegungsmustern und anderen Interaktionsformen zulassen. Denn viele Reaktionsbewegungsmuster weisen aus raum-zeitlicher Perspektive ähnliche Merkmale auf wie Interaktionsformen zwischen Punktobjekten derselben Art. Ein Beispiel dafür ist das Muster *Verfolgen/Entfliehen*, welches einen sehr ähnlichen Abdruck in Raum und Zeit hinterlässt wie die Bewegungsabfolgen in *Anführen* und *Folgen (Leadership)* oder das *Fortbewegen in einer Reihe (Single-File)*. Im Falle von *Verfolgen/Entfliehen* konnte mit Hilfe des Parameters Verzögerungszeit ein Kriterium gefunden werden, mit Hilfe dessen eine erfolgreiche Unterscheidung erreicht werden kann. Die Verzögerungszeit zeigt sich auch für die beiden anderen formalisierten Reaktionsbewegungsmuster *Konfrontation* und *Meiden* als wichtiger Parameter.

Neben der Abgrenzung zu Interaktionen ist zudem die Trennung zwischen Reaktionsbewegungsmustern, welche auch tatsächlich eine Semantik besitzen, und bedeutungslosen

Mustern, welche als Resultat von zufälligen Bewegungen im Raum entstehen, eine Herausforderung. Anhand der raum-zeitlichen Betrachtung der Verhaltensweisen während einer Situation mit *Verfolgen/Entfliehen* und der zufälligen Bewegungen von Punktobjekten in dieselbe Richtung wird klar, dass sich diese beiden Verhalten als Muster in einen Datensatz nur unwesentlich unterscheiden. Somit ist es das Ziel, Algorithmen zu entwickeln, welche mit Hilfe von einflussreichen Kriterien diese beiden Verhaltensformen in den Bewegungsdaten unterscheiden können. Der Erfolg dieser Algorithmen kann mit Hilfe von semantischen Zusatzinformationen über die in einem Datensatz enthaltenen Muster bestimmt werden. Für das Reaktionsbewegungsmuster *Verfolgen/Entfliehen* können im Falle des Algorithmus VED_i der Frontbereichswinkel, sowie die Distanz und die Zeitdauer von *Folgen* für eine erfolgreiche Unterscheidung verwendet werden. Mit Hilfe solcher Parameter kann die Wahrscheinlichkeit erhöht werden, dass sich zwei Punktobjekte im Blickfeld haben und sich willentlich verfolgen. Dabei müssen die Werte dieser Parameter unter Berücksichtigung der Gegebenheiten und Rahmenbedingungen (z.B. Sichtverhältnisse und Hindernisse auf einem Strassennetzwerk), welche bei der Aufzeichnung der Bewegungsdaten vorherrschten, gewählt werden. Zudem können Experten (z.B. Tierverhaltensforscher) mithelfen, die idealen Parameterwerte zu bestimmen. Folglich würde die Wahl der Parameter anders ausfallen bei Tierbewegungen auf einer vegetationslosen Fläche als bei Bewegungen von Fussgängern auf einem Strassennetzwerk.

Eine weitere wichtige Erkenntnis dieser Arbeit ist die Aussage über die Anwendbarkeit der entwickelten Algorithmen. Während die Algorithmen VED_i und VED_r sowie MD_i laut den Experimentresultaten sowohl auf simulierte Daten, als auch auf reale Bewegungsdaten anwendbar sind, müssen die Algorithmen MD_r , KD_i und KD_r noch erweitert werden, um eine problemlose Anwendung auf reale Daten zu garantieren. Im Falle des Algorithmus MD_r verlor der Parameter kritischer Bewegungsrichtungswechselwinkel im Vergleich zum euklidischen Raum an Einfluss. Dies ist darauf zurückzuführen, dass Objekte auf einem Netzwerk an Segmentübergängen zu einem Abbiegen und daher zu einem Bewegungsrichtungswechsel gezwungen sind. Deshalb müssen noch zusätzliche Überlegungen angestellt werden, um eine Unterscheidung zwischen Abbiegen auf dem Netzwerk und Richtungswechsel aufgrund von *Meiden*, sicherzustellen. Die tiefen Werte der Gütekriterien der beiden Algorithmen KD_i und KD_r bei den Experimenten ist primär auf die Qualität der Daten sowie auf die für diese Bewegungsdaten unpassende Formalisierung zurückzuführen.

Weiter konnten in dieser Masterarbeit zwei unterschiedliche Ansätze zur Raummodellierung miteinander verglichen werden, und zwar die individuelle und die räumliche Perspektive. Anhand der durchgeführten Experimente stellte sich das Segment als kleinste räumliche Einheit im Falle der räumlichen Perspektive als zu grob heraus. Dies zeigte sich speziell für das Muster *Meiden*, für welches aus räumlicher Perspektive kein geeignetes Kriterium zur Abbildung der maximal erlaubten *Annäherung* zur Verfügung steht, da Strassensegmente als kleinste Einheit nicht weiter unterteilt werden können. Prinzipiell kann ausgesagt werden, dass die Anwendbarkeit der räumlichen Perspektive stark abhängig ist von der kleinsten unterteilbaren Raumeinheit. Dabei ist die Wahl der

Segmentgrösse, Rasterweite oder Ausdehnung der Zellen in einer Tesselation entscheidend. Ein weiterer Unterschied zwischen den beiden Perspektiven zeigte sich bei der Analyse der Einflüsse der Parameter. Während bei der individuellen Perspektive bei Bewegungen in einem euklidischen Raum alle Parameter einen Beitrag zur erfolgreichen Abgrenzung leisten, sind gewisse Parameter bei der räumlichen Perspektive und den Bewegungen auf dem Netzwerk viel weniger einflussreich. Als Beispiel kann der Parameter Richtungswechsel im Falle von *Meiden* genannt werden. Da die Punktobjekte während ihren Bewegungen zu Richtungswechseln auf dem Netzwerk gezwungen sind, ist dieser Parameter kein geeignetes Kriterium. Somit büssen gewisse Parameter aufgrund der eingeschränkten Bewegungen auf dem Netzwerk ihren Einfluss ein, was primär auf das den Bewegungen unterliegende Raummodell zurückzuführen ist.

9.3 Ausblick

Die in dieser Arbeit vorgestellten Algorithmen weisen, wie die Diskussion der Experimentresultate gezeigt hat, noch einige Verbesserungsmöglichkeiten auf, welche in diesem Abschnitt kurz aufgelistet werden.

Die Algorithmen der räumlichen Perspektive könnten noch mit einem Test erweitert werden, bei dem geprüft wird, ob sich Punktobjekte auf benachbarten Segmenten sehen können. Dabei könnten die Orientierung der Strassensegmente oder zusätzliche Kontextinformationen (z.B. Hindernisse) miteinbezogen werden. Im Falle von Punktobjekten, welche sich auf benachbarten, in einer Linie befindenden Segmenten ohne grössere Hindernisse (z.B. Bäume, Bauten) bewegen, steigt die Wahrscheinlichkeit, dass die Punktobjekte miteinander interagieren. Durch diese Erweiterung wäre eine bessere Differenzierung der Bewegungsmuster möglich.

Ebenfalls weist der Algorithmus MD_r Verbesserungspotential auf, wobei versucht werden könnte, den Einfluss des Parameters Richtungsänderung zu erhöhen. Beispielsweise könnte getestet werden, ob ein abrupter Richtungswechsel aufgrund des Netzwerks oder aber als Folge von *Meiden* zustande gekommen ist. Zusätzliche Ideen für Erweiterungen wären das Untersuchen von *Meiden* einer fixen Zone (Tesselation) oder einer Rasterzelle, was die Eulersche Perspektive widerspiegeln würde.

Für die beiden Algorithmen zur Detektion von *Konfrontation* ist ebenfalls Verbesserungspotential auszumachen, wobei primär mit Hilfe einer tieferen Granularität der Formalisierung die Güte des Algorithmus verbessert werden könnte. Je nach Anwendungsfall könnte die in dieser Arbeit vorgestellte allgemeingehaltene Formalisierung einer *Konfrontation* noch erweitert werden.

Bei der Entwicklung der Algorithmen von *Verfolgen/Entfliehen* wurde darauf verzichtet, die in Kapitel 4 formalisierten Varianten des Ausgangs dieses Musters (erfolgreiche Jagd / erfolgreiche Flucht) umzusetzen. Dies primär aus dem Grund, dass mit dem *Konfrontations*-Algorithmus eine Methode entwickelt wurde, um zusammentreffende Objekte zu detektieren. Für das Ende von *Verfolgen/Entfliehen* wäre aber auch die Anwendung der Formalisierung in Kapitel 4 denkbar.

Eine mögliche Erweiterung für alle Formalisierungen und Algorithmen wäre die Ausweitung auf den Fall mit Beteiligung von n Objekten einer ersten Art und m Objekten einer zweiten ($n:m$ -Fall). Für dieses Unterfangen wären allerdings noch zusätzliche Literaturstudien notwendig, um die genauen Abläufe im Falle von mehreren beteiligten Punktobjekten zu verstehen und diese in formalen Definitionen umzusetzen.

Zudem wäre eine Verbesserung der Effizienz der Algorithmen denkbar. Die in dieser Arbeit vorgestellten Algorithmen basieren auf einem Vergleich zwischen allen n Punktobjekten während allen Zeitschritten t , was einer Laufzeitkomplexität von $O(n^2t)$ gleichkommt. Mit Hilfe von Techniken der Computational Geometry (z.B. Approximationsalgorithmen) könnte die Komplexität der Algorithmen noch verringert werden.

Neben *Verfolgen/Entfliehen*, *Konfrontation* und *Meiden* gilt es in Zukunft noch weitere Reaktionsbewegungsmuster zu formalisieren und in Algorithmen umzusetzen und zudem die in dieser Arbeit entwickelten Algorithmen auf andere Bewegungsdaten anzuwenden. Dabei sollten unbedingt auch Tierbewegungsdaten verwendet werden. Ideal wären Bewegungsdaten mit zusätzlichen Beobachtungsdaten der sich in den Daten abspielenden Muster (semantische Zusatzinformationen). Dadurch könnte beispielsweise die in Kapitel 8 diskutierte Vermutung der verbesserten Anwendbarkeit der Algorithmen zur Detektion von *Konfrontation* bei der Analyse von Tierbewegungsdaten getestet werden.

10 Literaturverzeichnis

- Admiraal, W., Raessens, J. und van Zeijts, H., 2007. Technology enhanced learning through mobile technology in secondary education. In: P. Cunningham und M. Cunningham eds. *Expanding the knowledge economy. Issues, applications, case studies (Part 2)*. Amsterdam: IOS Press, 1241–1248.
- Akkerman, S., Admiraal, W. und Huizenga, J., 2009. Storification in History Education: A Mobile Game in and about Medieval Amsterdam. *Computers & Education*, 52 (2), 449-459.
- Andersson, M., Gudmundsson, J., Laube, P. und Wolle, T., 2008. Reporting leaders and followers among trajectories of moving point objects. *GeoInformatica*, 12 (4), 497-528.
- Andersson, M., Wiklund, G. und Rundgren, H., 1980. Parental defence of offspring: a model and an example. *Animal Behaviour*, 28 (2), 536-542.
- Barrett, H.C., Todd, P.M., Miller, G.F. und Blythe, P.W., 2005. Accurate judgments of intention from motion cues alone: A crosscultural study. *Evolution & Human Behavior*, 26, 313-331.
- Benkert, M., Gudmundsson, J., Hübner, F. und Wolle T., 2006. Reporting flock patterns. In: Y. Azar und T. Erlebach, eds. *Proceedings of the 14th European Symposium on Algorithms (ESA 2006), volume 4168 of Lecture Notes in Computer Science*. Berlin: Springer, 660-671.
- Blythe, P.W., Miller, G.F. und Todd, P.M., 1996. Human Simulation of Adaptive Behavior: Interactive Studies of Pursuit, Evasion, Courtship, Fighting, and Play. In: P. Maes, M.J. Mataric, J.-A. Meyer, J. Pollack und S.W. Wilson, eds. *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. Cambridge: MIT Press/Bradford Books, 13-22.
- Brakatsoulas, S., Pfoser, D., Salas, R. und Wenk, C., 2005. On map-matching vehicle tracking data. In: K. Böhm, C.S. Jensen, L.M. Haas, M.L. Kersten, P.Å. Larson und B.C. Ooi, eds. *Proceedings of the 31st international conference on Very large data bases*. New York: VLDB Endowment, 853-864.
- Brockmann, H.J. und Barnard, C.J., 1979. Kleptoparasitism in birds. *Animal Behaviour*, 27, 487–514.
- Broom, M. und Ruxton, G.D., 2003. Evolutionarily stable kleptoparasitism: consequences of different prey types. *Behavioral Ecology*, 14, 23–33.
- Buchin, K., Buchin, M. und Gudmundsson, J. 2008. Detecting Single File Movement. In: *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*. Irvine, California: ACM, 288-297.
- Carbone, C., Du Toit J.T. und Gordon, I.J., 1997. Feeding success of African wild dogs: does kleptoparasitism by spotted hyenas influence hunting group success? *Journal of Animal Ecology*, 66, 318–326.
- Case, T.J. und Gilpin, M.E., 1974. Interference competition and niche theory. *Proceedings of the National Academy of Sciences*, 71, 3073–3077.
- CASIA, 2005. CASIA action database for recognition [online]. Center for Biometrics and Security Research, Institute of Automation Chinese Academy of Sciences, Peking. Verfügbar unter: <http://www.cbsr.ia.ac.cn/english/Action%20Databases%20EN.asp> [Zugriff am 13.10.2010].
- Cedras, C. und Shah, M., 1995. Motion-Based Recognition: A Survey. *Image and Vision Computing*, 13 (2), 129-155.
- Clark, P. und Johnson, D.E., 2009. Wolf-Cattle Interactions in the Northern Rocky Mountains. In: *Range Field Data 2009 Progress Report. Special Report 1092. June 2009*. Corvallis, OR: Oregon State University, Agricultural Experiment Station, 1-7.
- Cohen, C., Morelli, F. und Scott, K., 2008. A Surveillance System for Recognition of Intent within Individuals and Crowds. In: *IEEE Conference on Technologies for Homeland Security*. Waltham, MA: IEEE, 559-565.

- Connor, R.C., 1995. The benefits of mutualism: a conceptual framework. *Biological Reviews*, 70, 427-457.
- Cooper, S.M., Perotto-Baldivieso, H.L., Owens, M.K., Meek, M.G. und Figueroa-Pagán, M., 2008. Distribution and interaction of white-tailed deer and cattle in a semi-arid grazing system. *Agriculture, Ecosystems and Environment*, 127, 85-92.
- Creel, S., Spong, G. und Creel, N.M., 2001. Interspecific competition and the population biology of extinction-prone carnivores. In: J. Gittleman, D. Macdonald, S. Funk und R. Wayne, eds. *Conservation of Carnivores*. Cambridge: Cambridge University Press, 35-60.
- Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 269-271.
- Dodge, S., Weibel, R. und Lautenschütz, A.-K., 2008. Towards a Taxonomy of Movement Patterns. *Journal of Information Visualization*, 7, 240-252.
- Driver, P.M. und Humphries, D.A., 1988. Protean behavior. *The biology of unpredictability*. Oxford: University Press.
- Eilam, D., 2005. Die hard: a blend of freezing and fleeing as a dynamic defense-implications for the control of defensive behaviour. *Neuroscience and Biobehavioral Reviews*, 29, 1181-1191.
- Fagen, R., 1981. *Animal Play Behavior*. New York: Oxford University Press.
- Finney, L., Vinson, J. und Zaba, D., 1998. A Three-Phase Model for Predator-Prey Analysis. *Journal of Undergraduate Mathematics and its Applications*, 18 (3), 277-292.
- Frank, A.U., 2001. Socio-economic units: Their life and motion. In: A.U. Frank, J. Raper, und J.P. Cheylan, eds. *Life and Motion of Socio-economic Units, Volume 8 of GISDATA*. London: Taylor & Francis, 21-34.
- Freudig, D. und Sauermost, R., 2004. *Lexikon der Biologie*. Heidelberg: Spektrum Akademischer Verlag.
- Furuichi, N., 2002. Dynamics between a predator and a prey switching two kinds of escape motions. *Journal of Theoretical Biology*, 217, 159-166.
- Greenfeld, J. 2002. Matching gps observations to locations on a digital map. In: *Proceedings of the 81th Annual Meeting of the Transportation Research Board*, 2002.
- Gudmundsson, J. und Van Kreveld, M., 2006. Computing longest duration flocks in trajectory data. In: R.A. de By und S. Nittel, eds. *Proceedings of the 14th ACM Symposium on Advances in GIS*. New York: Association for Computing Machinery (ACM), 35-42.
- Gudmundsson, J., Van Kreveld, M.J. und Speckmann, B., 2007. Efficient Detection of Patterns in 2D Trajectories of Moving Points. *GeoInformatica*, 11 (2), 195-215.
- Gudmundsson, J. Laube, P. und Wolle, T., 2008. Movement Patterns in Spatio-Temporal Data. In: S. Shekar und H. Xiong, eds. *Encyclopedia of GIS*. Heidelberg: Springer, 726-732.
- Gütting, R., Boehlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N., Nardelli, E., Schneider, M. und Vazirgiannis, M., 2000. A foundation for representing and querying moving objects. *ACM Transactions on Database Systems (TODS)*, 2520 (1), 1-42.
- Hulbert, I.A.R., 2001. GPS and its use in animal telemetry: The next five years. In: A.M. Sibbald und I.J. Gordon, eds. *Proceedings of the Conference on Tracking Animals with GPS*. Aberdeen, UK: Macaulay Institute, 51-60.
- Ko, T., 2008. A survey on behavior analysis in video surveillance for homeland security applications. In: *37th IEEE Applied Imagery Pattern Recognition Workshop*. Washington DC: IEEE, 84-91.
- Kwan, M.P., 2000. Interactive geovisualization of activity-travel patterns using three dimensional geographical information systems: A methodological exploration with a large data set. *Transportation Research Part C*, 8 (1-6), 185-203.
- Laporte, I., Muhly, T.B., Pitt, J.A., Alexander, M. und Musiani, M., 2010. *Effects of Wolves on Elk and Cattle Behaviors: Implications for Livestock Production and Wolf Conservation* [online]. PLoS

One 5 (8). Verfügbar unter:

<http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0011954> [Zugriff am 13.10.2010].

- Laube, P., 2005. *Analysing point motion – Spatio-temporal data mining of geospatial lifelines*. Thesis (PhD). Department of Geography, University of Zurich, Switzerland.
- Laube, P., 2009. Progress in Movement Pattern Analysis. In: B. Gottfried und H. Aghajan, eds. *Behaviour Monitoring and Interpretation – Ambient Assisted Living*. Amsterdam: IOS Press, 43-71.
- Laube, P., Duckham, M., und Wolle, T., 2008. Decentralised Movement Pattern Detection Among Mobile Geosensor Nodes. In: T.J. Cova, H.J. Miller, K. Beard, A.U. Frank und M.F. Goodchild, eds. *Geographic Information Science, GIScience 2008*, Lecture Notes in Computer Science, 5266. Heidelberg: Springer: 132-144.
- Laube, P., Dennis, T., Forer, P. und Walker, M., 2007. Movement Beyond the Snapshot - Dynamic Analysis of Geospatial Lifelines. *Computers, Environment and Urban Systems*, 31 (5), 481-501.
- Laube, P. und Imfeld, S., 2002. Analyzing relative motion within groups of trackable moving point objects. In: M.J. Egenhofer und D.M. Mark, eds. *Geographic Information Science, volume 2478 of lecture Notes in Computer Science*. Berlin-Heidelberg, DE: Springer, 132-144.
- Laube, P., Imfeld, S. und Weibel, R., 2005. Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographic Informaiton Science*, 19 (6), 639-668.
- Laube, P., Van Kreveld, M. und Imfeld, S., 2004. Finding REMO – detecting relative motion patterns in geospatial lifelines. In: P.F. Fisher, ed. *Developments in Spatial Data Handling, Proceedings of the 11th International Symposium on Spatial Data Handling*. Heidelberg: Springer, 201-214.
- Laube, P. und Purves, R.S., 2006. An approach to evaluating motion pattern detection techniques in spatio-temporal data. *Computers, Environment and Urban Systems*, 30 (3), 347-374.
- Loft, E.R., Kie, J.G. und Menke, J.W., 1993. Grazing in the Sierra Nevada: home range and space use patterns of mule deer as influenced by cattle. *California Fish and Game*, 79 (4), 145–166.
- Magalhães, S., Janssen, A., Montserrat, M. und Sabelis, M.W., 2005. Prey attack and predators defend: counterattacking prey trigger parental care in predators. *Proceedings of the Royal Society of London Series B*, 272, 1929–1933.
- Mahajan, D., Kwatra, N., Jain, S., Kalra, P. und Banerjee, S., 2004. A framework for activity recognition and detection of unusual activities. In: B. Chanda, S. Chandran und L.S. Davis, eds. *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*. Kolkata, India: Allied Publishers Private Limited, 1-7.
- McGinley, M. und Codella S.G., 2008. Exploitative competition [online]. In: J. Cutler, ed. *Encyclopedia of Earth*. Environmental Information Coalition, National Council for Science and the Environment. Verfügbar unter: http://www.eoearth.org/article/Exploitative_competition [Zugriff am 05.10.2010].
- Miller, G. und Cliff, D., 1994. *Co-evolution of pursuit and evasion I: biological and game-theoretic foundations*. School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK.
- Moorcroft, P.R., 2008. Animal Home Ranges. In: S.E. Jørgensen, ed. *Encyclopedia of Ecology*. New York: Elsevier, 174-180.
- Morris, B. und Trivedi, M., 2008. A Survey of Vision-Based Trajectory Learning and Analysis for Surveillance. *IEEE Trans. on Circuits and Systems for Video Technology*, 18 (8), 1114-1127.
- Ng, R.T., 2001. Detecting outliers from large datasets. In: H.J. Miller und J. Han, eds. *Geographic Data Mining and Knowledge Discovery*. London: Taylor & Francis, 218–235.
- Ochieng, W.Y., Quddus, M.A. und Noland, R.B., 2004. Map-matching in complex urban road networks. *Brazilian Journal of Cartography (Revista Brasileira de Cartografia)*, 55 (2), 1-18.
- Odden, M., Wegge, P. und Fredriksen, T., 2010. Do tigers displace leopards? If so, why? *Ecological Research*, 25 (4), 875-881.

- Orellana, D. und Renso, C., 2010. Developing an interactions ontology for characterizing pedestrian movement behavior. In: M. Wachowicz, ed. *Movement-Aware Applications for Sustainable Mobility: Technologies and Approaches*. Hershey, PA: IGI Global Publishing, 62-86.
- Orellana, D. und Wachowicz, M., 2011. Exploring patterns of movement suspension in pedestrian mobility (noch nicht veröffentlicht).
- Orellana, D., Wachowicz, M., Andrienko, N. und Andrienko, G., 2009. Uncovering interaction patterns in mobile outdoor gaming. In: S. Dragicevic, ed. *International Conference on Advanced Geographic Information Systems & Web Services: GEOWS '09, Cancun, Mexico, February 1-7, 2009*. Piscataway, NJ: GEOWSIEEE, 177-182.
- Park, T., 1954. Experimental studies of interspecies competition II. Temperature, humidity, and competition of two species of *Tribolium*. *Physiological Zoology*, 27 (3), 177-238.
- Qu, Y., Wang, C. und Wang, X.S., 1998. Supporting fast search in time series for movement patterns in multiple scales. In: *Seventh International Conference on Information and Knowledge Management*. Bethesda, MD: ACM Press, 251-258.
- Quddus, M.A., Ochieng, W.Y., Zhao, L. und Noland, R.B., 2003. A general map matching algorithm for transport telematics applications. *GPS solutions*, 7 (3), 157-167.
- Reynolds, C.W., 1987. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, 21 (4), 25-34.
- Reynolds, C. W., 1999. Steering Behaviors For Autonomous Characters. In: *The proceedings of Game Developers Conference*. San Francisco, California: Miller Freeman Game Group, 763-782.
- Ridley, A.R. und Raihani, N.J., 2007. Facultative response to a kleptoparasite by the cooperatively breeding pied babbler. *Behavioral Ecology*, 18, 324-330.
- Saake, G. und Sattler, K.U., 2010. *Algorithmen und Datenstrukturen*. 4.Auflage. Heidelberg: dpunkt.Verlag.
- Schoener, T.W., 1983. Field experiments on interspecific competition. *American Naturalist*, 122, 240-285.
- Schmid, F., Richter, K.-F., Laube, P. (2011). Compressing Trajectories with Network Semantics (noch nicht veröffentlicht).
- Schnellhammer, C. und Feilkas, T., 2002. *Steering Behaviors*. Diplomarbeit. Fachhochschule Regensburg.
- Sirost, E., 2000. An evolutionarily stable strategy for aggressiveness in feeding groups. *Behavioral Ecology*, 11, 351-356.
- Smith, J. M., Price, G. R. 1973. The logic of animal conflict. *Nature*, 246, 15-18.
- Stewart, K.M., Bowyer, R.T., Kie, J.G., Cimon, N.J. und Johnson, B.K., 2002. Temporospatial distributions of elk, mule deer, and cattle: resource partitioning and competitive displacement. *Journal of Mammalogy*, 83, 229-244.
- Szlávik, Z., Kovács, L., Havasi, L., Benedek, Cs., Petrás, I., Utasi, Á., Licsár, A., Czúni, L. und Szirányi, T., 2008. Behavior and event detection for annotation and surveillance. *Proceedings of International Workshop on Content-Based Multimedia Indexing 2008*. London, UK: IEEE, 117-124.
- Tanner C.J. und Adler, F.R., 2009. To fight or not to fight: context-dependent interspecific aggression in competing ants. *Animal Behaviour*, 77, 297-305.
- Thomas, J.J. und Cook, K.A., 2006. A visual analytics agenda. *IEEE Computer Graphics and Applications*, Vol. 26(1), 10-13.
- Tso I.M., Severinghaus L.L., 1998. Silk stealing by *Argyrodes lanyuensis* (Araneae: Theridiidae): a unique form of kleptoparasitism. *Animal Behaviour*, 56, 219-225.
- Turchin, P., 1998. *Quantitative Analysis of Movement: measuring and modeling population redistribution in plants and animals*. Sunderland, MA: Sinauer Associates.

- Waag Society., 2007. *Frequency1550 Mobile history city game* [online]. Waag Society. Verfügbar unter: <http://www.waag.org/project/frequentie> [Zugriff am 20.1.2011].
- West, S.A., Griffin, A.S. und Gardner, A., 2007. Social semantics: altruism, cooperation, mutualism, strong reciprocity and group selection. *Journal of Evolutionary Biology*, 20, 415–432.
- Woodroffe, R. und Ginsberg, J.G., 2005. King of the beasts? Evidence for guild redundancy among large mammalian carnivores. In: J. Ray, J. Berger, K.H. Redford, R. Steneck, eds. *Large carnivores and biodiversity: does saving one conserve the other?* New York: Island Press, 154–176.

11 Anhang

11.1 Java- Klasse MovementPatternDetector

```
package frequentiel1550;

import java.util.*;
import java.util.ArrayList;

public class MovementPatternDetector {

public class MovementPatternDetector {

    //Deklaration der Variablen
    private double minimalDistance; //minimale Distanz, damit Bewegungsmuster auftreten kann
    private double maximalDistance; //maximale Distanz, damit Bewegungsmuster auftreten kann
    private double criticalAngle; //kritischer Winkel des Frontbereiches
    private double criticalTime; //kritische Zeitdauer für Definition eines Bewegungsmusters
    private double criticalTimeLag; //maximal erlaubte Verzögerungszeit zwischen Aktion und Reaktion
    private PointObjectGroup pObjGr; //Gruppe von Punktobjekten, welche analysiert werden sollen
    private StreetGroup strGr; //Gruppe mit Strassen, auf welchen sich die Punktobjekte bewegen
    private int containmentCounter; //Zähler für die Anzahl steps in welcher 1 anderes Objekt im Frontbereich ist
    private double patternTolerance; //Anzahl erlaubte Unterbrüche im Bewegungsmuster (Toleranz)
    private int toleranceChecker; //Hilfsvariable um Ueberschreitung der Toleranz zu prüfen
    private int startChecker; //Hilfsvariable fuer Start eines Bewegungsmusters
    private int nextStart; //Hilfsvariable, welche das Suchfenster fuer den followingDetector zurueckstellt
    private int helper = 1; //wird nach jedem Durchlauf der Distanzberechnung um 1 erhöht
    //damit zwei Punktobjekte nicht zweimal verglichen werden

    public static ArrayList <String>
    movementPatternArray = new ArrayList(); // Array fuer die Speicherung der die gefundenen Bewegungsmuster
    private String reactionPatternType = ""; //Variable für den Namen des Bewegungsmusters

    //Konstruktor
    public MovementPatternDetector() {

    }
```

```
//Algorithmen zur Detektion von Reaktionsbewegungsmustern

/**
 * Algorithmus zur Detektion von Verfolgen/Entfliehen mit Hilfe der individuellen Perspektive
 * @param pObjGr Gruppe mit Punktobjekten
 * @param criticalDistanceCoincidence Parameter kritische Distanz Zusammensein
 * @param criticalTimeCoincidence Parameter kritische Zeitdauer Zusammensein
 * @param coincidenceTolerance Toleranzparameter Zusammensein
 * @param criticalTimeLag Parameter kritische Verzögerungszeit
 * @param criticalDistanceFollowing Parameter kritische Distanz Folgen
 * @param criticalTimeFollowing Parameter kritische Zeitdauer Folgen
 * @param followingTolerance Toleranzparameter Folgen
 * @param criticalAngle kritischer Winkel des Frontbereichs
 * @return void
 **/

public void PEDetectorIndividualPerspective(PointObjectGroup pObjGr, double criticalDistanceCoincidence,
double criticalTimeCoincidence, double coincidenceTolerance, double criticalTimeLag,
double criticalDistanceFollowing, double criticalTimeFollowing, double followingTolerance,
double criticalAngle) throws Exception {

    this.pObjGr = pObjGr;
    int reactionPatternNr = 0;
    int reactionPatternArrayNr = 0;
    reactionPatternType = "Pursuit/Evasion";

    //Aktion = Zusammentreffen (Coincidence)
    //1.Schritt: Untersuchung, ob sich POs getroffen haben
    coincidenceDetectorIndPersp(criticalDistanceCoincidence,criticalTimeCoincidence,coincidenceTolerance,0,0);

    //2.Schritt: Untersuchen ob sich POs ueber eine kritische Zeitdauer folgen (Reaktion auf Annaeherung)
    followingDetectorIndPersp(criticalDistanceFollowing,criticalAngle,criticalTimeFollowing, followingTolerance);

    //3.Schritt: Untersuchung der Zeitverzoegerung zwischen Aktion und Reaktion
    checkTimeLag(criticalTimeLag, 1, 4, 4, 2, reactionPatternNr, reactionPatternArrayNr);

    //4.Schritt: Untersuchung, ob waehrend Verfolgung eine Konfrontation stattgefunden hat
    //Verwendung der idealen Parameter
    reactionPatternType = "Konfrontation";
    confrontationDetectorIndividualPerspective(pObjGr, 50, 30, 0, 50, 50, 15, 0, 100);
    comparePursuingsToConfrontations("Verfolgen/Entfliehen(ind.Persp.)");
}

```

```

/**
 * Algorithmus zur Detektion von Verfolgen/Entfliehen mit Hilfe der räumlichen Perspektive
 * @param pObjGr           Gruppe mit Punktobjekten
 * @param strGr           Gruppe mit allen relevanten Strassensegmenten
 * @param neighbourhoodCoincidence  Parameter Nachbarschaftsgrösse Zusammensein
 * @param criticalTimeCoincidence  Parameter kritische Zeitdauer Zusammensein
 * @param patternToleranceCoincidence  Toleranzparameter Zusammensein
 * @param neighbourhoodFollowing  Parameter Nachbarschaftsgrösse Folgen
 * @param criticalTimeFollowing  Parameter kritische Zeitdauer Folgen
 * @param patternToleranceFollowing  Toleranzparameter Folgen
 * @param criticalTimeLag  Parameter kritischer Verzögerungszeit
 * @return void
 **/

public void PEDetectorSpacePerspective(PointObjectGroup pObjGr, StreetGroup strGr,
    int neighbourhoodCoincidence, double criticalTimeCoincidence, double patternToleranceCoincidence,
    int neighbourhoodFollowing, double criticalTimeFollowing, double patternToleranceFollowing,
    double criticalTimeLag) throws Exception {

    int reactionPatternNr = 1;
    int reactionPatternArrayNr = 0;
    this.pObjGr = pObjGr;
    this.strGr = strGr;
    reactionPatternType = "Pursuit/Evasion";

    //1.Schritt: Analyse der Aktion -> Untersuchung, ob sich die Punktobjekte annähern (sich auf dem gleichen Segment
    aufhalten)
    coincidenceDetectorSpacePersp(neighbourhoodCoincidence, criticalTimeCoincidence, patternToleranceCoincidence, 0);

    //2.Schritt: Analyse der Reaktion -> Untersuchung, ob ein Punktobjekt nach dem Annähern ein anderes verfolgt
    followingDetectorSpacePersp(neighbourhoodFollowing, criticalTimeFollowing, patternToleranceFollowing);

    //3.Schritt: Untersuchung der Zeitverzögerung zwischen Aktion und Reaktion
    checkTimeLag(criticalTimeLag, 1, 4, 4, 3, reactionPatternNr, reactionPatternArrayNr);

    //4.Schritt: Überprüfen, ob während der Verfolgung eine Konfrontation stattgefunden hat
    //mit den idealen Parametern
    confrontationDetectorSpacePerspective(pObjGr, strGr, 1, 30, 0, 5, 0, 1, 15, 0, 120);
    comparePursuitsToConfrontations("Verfolgen/Entfliehen(räuml.Persp.)");
}

```

```

/**
 * Algorithmus zur Detektion von Meiden (Avoidance) mit Hilfe der individuellen Perspektive
 * @param pObjGr Gruppe mit Punktobjekten
 * @param criticalDistanceCoincidence Parameter kritische Distanz Zusammensein
 * @param criticalTimeCoincidence Parameter kritische Zeitdauer Zusammensein
 * @param coincidenceTolerance Toleranzparameter Zusammensein
 * @param coincidenceFreeTime Parameter kritische Zeitdauer ohne Zusammensein
 * @param criticalDistanceLeaving Parameter kritische Distanz Getrenntsein
 * @param criticalTimeLeaving Parameter kritische Zeitdauer Getrenntsein
 * @param leavingTolerance Toleranzparameter Getrenntsein
 * @param criticalDistanceAvoidance Parameter Minimaldistanz von Meiden
 * @param criticalDirectionChange Parameter kritischer Winkel Bewegungsrichtungsänderung
 * @param criticalTimeLag Parameter kritische Verzögerungszeit
 * @return void
 **/

public void avoidanceDetectorIndividualPerspective(PointObjectGroup pObjGr, double criticalDistanceCoincidence,
double criticalTimeCoincidence, double coincidenceTolerance, double coincidenceFreeTime,
double criticalDistanceLeaving, double criticalTimeLeaving, double leavingTolerance,
double criticalDistanceAvoidance, double criticalDirectionChange,
double criticalTimeLag) throws Exception {

    this.pObjGr = pObjGr;
    int reactionPatternNr = 2;
    int reactionPatternArrayNr = 1;

    //1.Schritt: Bestimmung ab wann sich 2 Punktobjekte angenaehert haben (Aktion)
    coincidenceDetectorIndPersp(criticalDistanceCoincidence, criticalTimeCoincidence,
        coincidenceTolerance, coincidenceFreeTime, criticalDistanceAvoidance);

    //2.Schritt: Untersuchung wie Nahe sich Punktobjekte maximal angenähert haben
    checkMinimalDistance(criticalDistanceAvoidance);

    //3.Schritt: Untersuchung ob es einen abrupten Richtungswechsel gegeben hat
    checkOrientationChange(criticalDirectionChange, 1, 5);

    //4.Schritt: Bestimmung ab wann sich 1 PO aus der Nachbarschaft eines anderen POs bewegt hat
    leavingNeighbourhoodDetector(6, criticalTimeLag, criticalTimeLeaving, leavingTolerance);

    //5.Schritt: Untersuchung der Zeitverzögerung zwischen Aktion und Reaktion
    checkTimeLag(criticalTimeLag, 1, 6, 3, 2, reactionPatternNr, reactionPatternArrayNr);
}

```

```

/**
 * Algorithmus zur Detektion von Meiden (Avoidance) mit Hilfe der räumlichen Perspektive
 * @param pObjGr Gruppe mit Punktobjekten
 * @param strGr Gruppe mit allen relevanten Strassensegmenten
 * @param neighbourhoodCoincidence Parameter Nachbarschaftsgrösse Zusammensein
 * @param criticalTimeCoincidence Parameter kritische Zeitdauer Zusammensein
 * @param coincidenceTolerance Toleranzparameter Zusammensein
 * @param coincidenceFreeTime Parameter kritische Zeitdauer ohne Zusammensein
 * @param neighbourhoodLeaving Parameter Nachbarschaftsgrösse Getrenntsein
 * @param criticalTimeLeaving Parameter kritische Zeitdauer Getrenntsein
 * @param leavingTolerance Toleranzparameter Getrenntsein
 * @param criticalDirectionChange Parameter kritischer Winkel der Bewegungsrichtungsänderung
 * @param criticalTimeLag Parameter kritische Verzögerungszeit
 * @return void
 */
public void avoidanceDetectorSpacePerspective(PointObjectGroup pObjGr, StreetGroup strGr,
    int neighbourhoodCoincidence, double criticalTimeCoincidence, double coincidenceTolerance,
    double coincidenceFreeTime, int neighbourhoodLeaving, double criticalTimeLeaving, double leavingTolerance, double
    criticalDirectionChange, double criticalTimeLag) throws Exception {

    this.pObjGr = pObjGr;
    this.strGr = strGr;
    int reactionPatternNr = 3;
    int reactionPatternArrayNr = 1;

    //1.Schritt: Bestimmung ab wann sich 2 Punktobjekte angenaehert haben (Aktion)
    coincidenceDetectorSpacePersp(neighbourhoodCoincidence, criticalTimeCoincidence,
        coincidenceTolerance, coincidenceFreeTime);

    //2.Schritt: neu Untersuchung ob es einen abrupten Richtungswechsel gegeben hat
    checkOrientationChange(criticalDirectionChange, 1, 4);

    //3.Schritt: Bestimmung ab wann sich 1 Punktobjekt aus der Nachbarschaft eines anderen bewegt hat
    leavingNeighbourhoodDetector(5, criticalTimeLag, criticalTimeLeaving, leavingTolerance);

    //4.Schritt: Untersuchung der Zeitverzögerung zwischen Aktion und Reaktion
    checkTimeLag(criticalTimeLag, 1, 5, 3, 2, reactionPatternNr, reactionPatternArrayNr);
}

```

```

/**
 * Algorithmus zur Detektion von Konfrontationen (Confrontations) mit Hilfe der individuellen Perspektive
 * @param pObjGr Gruppe mit Punktobjekten
 * @param criticalDistanceCoincidence Parameter kritische Distanz Zusammensein
 * @param criticalTimeCoincidence Parameter kritische Zeitdauer Zusammensein
 * @param coincidenceTolerance Toleranzparameter Zusammensein
 * @param criticalDistanceIntersection Parameter kritische Distanz Intersection
 * @param criticalDistanceLeaving Parameter kritische Distanz Getrenntsein
 * @param criticalTimeLeaving Parameter kritische Zeitdauer Getrenntsein
 * @param leavingTolerance Toleranzparameter Getrenntsein
 * @param criticalTimeLag Parameter kritische Verzögerungszeit
 * @return void
 **/

public void confrontationDetectorIndividualPerspective(PointObjectGroup pObjGr, double criticalDistanceCoincidence,
double criticalTimeCoincidence, double coincidenceTolerance, double criticalDistanceIntersection,
double criticalDistanceLeaving, double criticalTimeLeaving, double leavingTolerance,
double criticalTimeLag) throws Exception {

    this.pObjGr = pObjGr;
    int reactionPatternNr = 4;
    int reactionPatternArrayNr = 2;

    //Aktion = Coincidence -> POs haben sich unter bestimmte Distanz angenaehert und verbleiben so ueber
    //kritische Zeitdauer
    //1.Schritt: Untersuchung, ob sich POs unter eine kritische Distanz angenaehert haben
    coincidenceDetectorIndPersp(criticalDistanceCoincidence,criticalTimeCoincidence,coincidenceTolerance,0,0);

    //Reaktion 1 = POs rennen ineinander hinein (Konfrontation)
    //2.Schritt: Untersuchung, ob sich die Trajektorien der POs schneiden
    intersectionDetectorIndPersp(criticalDistanceIntersection, 1);

    //Reaktion 2 = Verlassen der Nachbarschaft
    //3.Schritt: Untersuchung, ob eines der Punktobjekte die Nachbarschaft eines zweiten Verlassen hat
    leavingNeighbourhoodDetector(4, 100000, criticalTimeLeaving, leavingTolerance);

    //Verzoegungszeit = Zeitdifferenz zwischen Encounter und Leaving
    //4.Schritt: Test, ob Verzoegerungszeit die kritische Zeitdauer nicht ueberschreitet
    checkTimeLag(criticalTimeLag, 5, 2, 3, 2, reactionPatternNr, reactionPatternArrayNr);
}

```

```

/**
 * Algorithmus zur Detektion von Konfrontationen (Confrontations) mit Hilfe der individuellen Perspektive
 * @param pObjGr Gruppe mit Punktobjekten
 * @param neighbourhoodCoincidence Parameter Nachbarschaftsgrösse Zusammensein
 * @param criticalTimeCoincidence Parameter kritische Zeitdauer Zusammensein
 * @param coincidenceTolerance Toleranzparameter Zusammensein
 * @param criticalTimeIntersection Parameter kritische Zeitdauer Intersektion
 * @param intersectionTolerance Toleranzparameter Intersektion
 * @param neighbourhoodLeaving Parameter Nachbarschaftsgrösse Getrenntsein
 * @param criticalTimeLeaving Parameter kritische Zeitdauer Getrenntsein
 * @param leavingTolerance Toleranzparameter Getrenntsein
 * @param criticalTimeLag Parameter kritische Verzögerungszeit
 * @return void
 **/

public void confrontationDetectorSpacePerspective(PointObjectGroup pObjGr, StreetGroup strGr,
    int neighbourhoodCoincidence, double criticalTimeCoincidence, double coincidenceTolerance,
    double criticalTimeIntersection, double intersectionTolerance, double neighbourhoodLeaving,
    double criticalTimeLeaving, double leavingTolerance, double criticalTimeLag) throws Exception {

    this.pObjGr = pObjGr;
    this.strGr = strGr;
    int reactionPatternNr = 5;
    int reactionPatternArrayNr = 2;

    //Aktion = Meeting -> POs haben sich auf benachbarten Segmenten angenaehert
    //1.Schritt: Untersuchung, ob sich POs getroffen haben
    coincidenceDetectorSpacePersp(neighbourhoodCoincidence, criticalTimeCoincidence, coincidenceTolerance, 0);

    //Reaktion = POs rennen ineinander hinein (Konfrontation)
    //2.Schritt: Untersuchung, ob sich die Trajektorien der POs schneiden
    intersectionDetectorSpacePersp(neighbourhoodCoincidence, criticalTimeIntersection, intersectionTolerance);

    //Reaktion 2 = Verlassen der Nachbarschaft
    //3.Schritt: Untersuchung, ob eines der Punktobjekte die Nachbarschaft eines zweiten Verlassen hat
    leavingNeighbourhoodDetector(4, 1000, criticalTimeLeaving, leavingTolerance);

    //Verzoegungszeit = Zeitdifferenz zwischen Meeting(Aktion) und Intersection (Reaktion)
    //4.Schritt: Test, ob Verzoegerungszeit die kritische Zeitdauer nicht ueberschreitet
    checkTimeLag(criticalTimeLag, 5, 2, 3, 2, reactionPatternNr, reactionPatternArrayNr);
}

```

```
/**
 * Algorithmus zur Detektion von Approximationen nach Orellana et al. (2009)
 * @param pObjGr Gruppe mit Punktobjekten
 * @param criticalDistanceCoincidence Parameter kritische Distanz Zusammensein
 * @param criticalTimeCoincidence Parameter kritische Zeitdauer Zusammensein
 * @param coincidenceTolerance Toleranzparameter Zusammensein
 * @return void
 */

public void confrontationDetectorIndividualPerspectiveDanielOrellana(PointObjectGroup pObjGr,
    double criticalDistanceCoincidence,
    double criticalTimeCoincidence, double coincidenceTolerance) throws Exception {

    this.pObjGr = pObjGr;
    int reactionPatternNr = 4;
    int reactionPatternArrayNr = 2;

    //Untersuchung, ob sich POs unter eine kritische Distanz angenaehert haben
    coincidenceDetectorIndPersp(criticalDistanceCoincidence, criticalTimeCoincidence, coincidenceTolerance, 0, 0);
    double criticalTimeLag = 1;
    checkTimeLag(criticalTimeLag, 1, 4, 1, 4, reactionPatternNr, reactionPatternArrayNr);
}

//Methoden zur Detektion von Bewegungsmustern

/**
 * Methode zur Detektion von Annäherung (Konvergenz) individuelle Perspektive
 * @param criticalDistanceApproaching Parameter kritische Distanz Annäherung
 * @param criticalTimeApproaching Parameter kritische Zeitdauer Annäherung
 * @param approachingTolerance Toleranzparameter Annäherung
 * @return void
 */

public void approachingDetectorIndPersp(double criticalDistanceApproaching,
    double criticalTimeApproaching, double approachingTolerance) throws Exception {

    int patternNr = 0;
    int patternSign = 1;
    this.maximalDistance = criticalDistanceApproaching;
    this.minimalDistance = 0;
    this.criticalTime = criticalTimeApproaching;
    this.patternTolerance = approachingTolerance;
}
```

```

//1.Schritt: Untersuchung der Distanzbeziehung
observeDistanceRelationships(patternNr);

//2.Schritt: Untersuchung von Trends in den Distanzen
//-> ob sich die Distanz zwischen 2 POs stetig verkleinert (=Annaeherung)
observeDistanceTrends(patternNr);

//3.Schritt: Untersuchung der Dauer der Annaeherung
patternLengthCheck(patternNr, patternSign);
}

/**
 * Methode zur Detektion von Annaeherung (Konvergenz) räumliche Perspektive
 * @param neighbourStreetOrder Parameter Nachbarschaftsgrösse Annaeherung
 * @param criticalTimeApproaching Parameter kritische Zeitdauer Annaeherung
 * @param criticalTimeApproaching Toleranzparameter Annaeherung
 * @return void
 */

public void approachingDetectorSpacePersp(int neighbourStreetOrder,
    double criticalTimeApproaching, double approachingTolerance) throws Exception {

    int patternNr = 0;
    int patternSign = 2;
    this.criticalTime = criticalTimeApproaching;
    this.patternTolerance = approachingTolerance;

    //1.Schritt: Ueberpruefung, ob sich 2 POs auf gleichem bzw. benachbarten Segmenten bewegen
    checkMovementsOnNeighbouringSegments(patternNr, neighbourStreetOrder);

    //2.Schritt: Testen, ob sich POs auf den benachbarten Segmenten aufeinander zubewegen (=Annaeherung)
    checkMovementDirectionOnSegment(patternNr);

    //3.Schritt: Testen, ob sich POs genügend lange auf benachbarten Segmenten in die gleiche Richtung bewegen
    patternLengthCheck(patternNr, patternSign);
}

/**
 * Methode zur Detektion von Zusammensein (Coincidence) individuelle Perspektive
 * @param pObjGr Gruppe mit Punktobjekten
 * @param criticalDistanceCoincidence Parameter kritische Distanz Zusammensein
 * @param criticalTimeCoincidence Parameter kritische Zeitdauer Zusammensein

```

```

*   @param coincidenceTolerance      Toleranzparameter Zusammensein
*   @param coincidenceFreeTime      Parameter kritische Zeitdauer ohne Zusammensein
*   @param minimalDistanceCoincidence Parameter minimale erlaubte Distanz während Zusammensein
*   @return void
**/

public void coincidenceDetectorIndPersp(double criticalDistanceCoincidence, double criticalTimeCoincidence,
    double coincidenceTolerance, double coincidenceFreeTime, double minimalDistanceCoincidence)
    throws Exception {

    this.maximalDistance = criticalDistanceCoincidence;
    this.minimalDistance = minimalDistanceCoincidence;
    this.criticalTime = criticalTimeCoincidence;
    this.patternTolerance = coincidenceTolerance;
    int patternNr = 1;
    int patternSign = 1;

    //1.Schritt: Untersuchung der Distanzbeziehungen zwischen den Punktobjekten
    observeDistanceRelationships(patternNr);

    //2.Schritt: Ueberpruefung ob Bewegungsmuster genuegend lange aufrechterhalten wird
    patternLengthCheck(patternNr, patternSign);

    //3.Schritt: Sicherstellen, dass sich die POs vor dem Aufeinandertreffen angenaehert
    // -> um sicherzustellen, dass sich POs nicht bereits kurze Zeit vorher getroffen haben
    // -> falls sich POs kurze Zeit vorher getroffen haben, wird Aufeinandertreffen herausgefiltert
    observeTimeBeforeCoincidence(coincidenceFreeTime, patternNr, patternSign+1);

}

/**
*   Methode zur Detektion von Zusammensein (Coincidence) räumliche Perspektive
*   @param pObjGr      Gruppe mit Punktobjekten
*   @param neighbourStreetOrder Parameter Nachbarschaftsgrösse Zusammensein
*   @param criticalTimeCoincidence Parameter kritische Zeitdauer Zusammensein
*   @param coincidenceTolerance Toleranzparameter
*   @param coincidenceFreeTime Parameter kritische Zeitdauer ohne Zusammensein
*   @return void
**/

public void coincidenceDetectorSpacePersp(int neighbourStreetOrder, double criticalTimeCoincidence,
    double coincidenceTolerance, double coincidenceFreeTime) throws Exception {

    this.criticalTime = criticalTimeCoincidence;

```

```

    this.patternTolerance = coincidenceTolerance;
    int patternNr = 1;
    int patternSign = 1;

    //1.Schritt: Untersuchung der Distanzbeziehungen zwischen den Punktobjekten
    checkMovementsOnNeighbouringSegments(patternNr, neighbourStreetOrder);

    //2.Schritt: Ueberpruefung ob Bewegungsmuster genuegend lange aufrechterhalten wird
    patternLengthCheck(patternNr, patternSign);

    //3.Schritt: Sicherstellen, dass sich die POs vor dem Aufeinandertreffen angenaehert
    // -> um sicherzustellen, dass sich POs nicht bereits kurze Zeit vorher getroffen haben
    // -> falls sich POs kurze Zeit vorher getroffen haben, wird Aufeinandertreffen herausgefiltert
    observeTimeBeforeCoincidence(coincidenceFreeTime, patternNr, (patternSign+1));
}

/**
 * Methode zur Detektion von Separation (Divergenz) individuelle Perspektive
 * @param criticalDistanceSeparation Parameter kritische Distanz Separation
 * @param criticalTimeSeparation Parameter kritische Zeitdauer Separation
 * @param separationTolerance Toleranzparameter Separation
 * @param minimalDistance Parameter minimale erlaubte Distanz Separation
 * @return void
 */

public void separationDetectorIndPersp(double criticalDistanceSeparation, double criticalTimeSeparation,
    double separationTolerance, double minimalDistance) throws Exception {

    int patternNr = 2;
    int patternSign = 1;
    this.minimalDistance = minimalDistance;
    this.maximalDistance = criticalDistanceSeparation;
    this.criticalTime = criticalTimeSeparation;
    this.patternTolerance = separationTolerance;

    //1.Schritt: Untersuchung der Distanzbeziehung
    observeDistanceRelationships(patternNr);

    //2.Schritt: Untersuchung von Trends in den Distanzen
    //-> ob sich die Distanz zwischen 2 POs stetig vergrößer (=Separation)
    observeDistanceTrends(patternNr);

    //3.Schritt: Untersuchung der Dauer des Annaehern
    patternLengthCheck(patternNr, patternSign);
}

```

```
/**
 * Methode zur Detektion von Separation (Divergenz) räumliche Perspektive
 * @param criticalTimeSeparation   Parameter kritische Zeitdauer Separation
 * @param separationTolerance     Toleranzparameter Separation
 * @return void
 */

public void separationDetectorSpacePersp(double criticalTimeSeparation, double separationTolerance)
    throws Exception {

    int patternNr = 2;
    int patternSign = 2;
    int neighbourStreetOrder = 2;
    this.criticalTime = criticalTimeSeparation;
    this.patternTolerance = separationTolerance;

    //1.Schritt: Ueberpruefung, ob sich 2 POs auf gleichem bzw. benachbarten Segmenten bewegen
    checkMovementsOnNeighbouringSegments(patternNr, neighbourStreetOrder);

    //2.Schritt: Testen, ob sich POs auf den benachbarten Segmenten voneinander wegbewegen
    checkMovementDirectionOnSegment(patternNr);

    //3.Schritt: Testen, ob sich POs genügend lange auf benachbarten Segmenten in die gleiche Richtung bewegen
    patternLengthCheck(patternNr, patternSign);

}

/**
 * Methode zur Detektion von räumlicher Trennung zwischen Punktobjekten (=Getrenntsein)
 * @param coincidenceSign         Kennzeichen fuer das Bewegungsmuster Zusammensein
 * @param criticalTimeLagCoincidenceVsLeaving   kritische Zeitverzoegerung zwischen Zusammensein/Getrenntsein
 * @param criticalTimeLeaving     Parameter kritische Zietdauer Getrenntsein
 * @param leavingTolerance        Toleranzparameter Getrenntsein
 * @return void
 */

public void leavingNeighbourhoodDetector(int coincidenceSign, double criticalTimeLagCoincidenceVsLeaving,
    double criticalTimeLeaving, double leavingTolerance) {

    int patternNr = 3;
    int patternSign = 1;
    this.criticalTime = criticalTimeLeaving;
    this.patternTolerance = leavingTolerance;
}
```

```

helper = 1;
for(int i=0; i<pObjGr.getGroupSize(); i++) {
    //System.out.println("    ... Punktobjekt "+(i+1));
    for(int j=helper; j<pObjGr.getGroupSize(); j++) {
        //System.out.println("        ... im Vergleich mit Punktobjekt "+(j+1));
        for(int k=0; k<pObjGr.getPO(i).getAmountOfPoints(); k++) {
            int meetingStepCounter = 0;
            int leavingStepCounter = 0;
            while(k<pObjGr.getPO(i).getAmountOfPoints()
                && pObjGr.getPO(i).movementPatterns[j][k][1] == coincidenceSign) {
                meetingStepCounter++;
                k++;
            }
            //Untersuchung der Zeitdifferenz zwischen dem Start von Coincidence und dem Start von Leaving
            if(meetingStepCounter != 0 && meetingStepCounter < criticalTimeLagCoincidenceVsLeaving) {
                //falls Coincidence zu Ende
                int l = (int) (k-(pObjGr.getPO(j).getPoint(0).getTime()
                    - pObjGr.getPO(i).getPoint(0).getTime()));
                while(l<pObjGr.getPO(j).getAmountOfPoints() && k<pObjGr.getPO(i).getAmountOfPoints()
                    && pObjGr.getPO(i).movementPatterns[j][k][1] == 0) {
                    //jedoch nur dann Leaving wenn auch beide POs noch spielen
                    //Coincidence kann auch zu Ende sein weil das zweite PO gar nicht mehr spielt
                    if(pObjGr.getPO(i).getPoint(k).getTime()
                        == pObjGr.getPO(j).getPoint(l).getTime()
                        && pObjGr.getPO(i).getPoint(k).getX() > 0
                        && pObjGr.getPO(j).getPoint(l).getX() > 0) {

                        pObjGr.getPO(i).movementPatterns[j][k][patternNr] = 1.0;
                        leavingStepCounter++;
                    }
                    k++;
                    l++;
                }
                k--;
            }
        }
    }
    helper++;
}
patternLengthCheck(patternNr, patternSign);
}

```

```
/**
 * Methode zur Detektion von Folgen individuelle Perspektive
 * @param criticalDistanceFollowing Parameter kritische Distanz Folgen
 * @param criticalAngle Parameter kritischer Frontbereichswinkel
 * @param criticalTimeFollowing Parameter kritische Zeitdauer Folgen
 * @param followingTolerance Toleranzparameter Folgen
 * @return void
 */
public void followingDetectorIndPersp(double criticalDistanceFollowing, double criticalAngle,
    double criticalTimeFollowing, double followingTolerance) throws Exception {

    int patternNr = 4;
    int patternSign = 1;
    this.maximalDistance = criticalDistanceFollowing;
    this.minimalDistance = 0;
    this.criticalAngle = criticalAngle;
    this.criticalTime = criticalTimeFollowing;
    this.patternTolerance = followingTolerance;

    //1.Schritt: Untersuchung der Distanzbeziehungen zwischen den Punktobjekten
    // && 2.Schritt: Ueberpruefung ob sich ein Objekt im Frontbereich des anderen befindet
    observeDistanceRelationships(patternNr);

    //3.Schritt: Ueberpruefung ob Bewegungsmuster genuegend lange aufrechterhalten wird
    patternLengthCheck(patternNr, patternSign);

}

/**
 * Methode zur Detektion von Folgen räumliche Perspektive
 * @param neighbourStreetOrder Parameter Nachbarschaftsgrösse Folgen
 * @param criticalTimeFollowing Parameter kritische Zeitdauer Folgen
 * @param followingTolerance Toleranzparameter Folgen
 * @return void
 */
public void followingDetectorSpacePersp(int neighbourStreetOrder, double criticalTimeFollowing,
    double followingTolerance) throws Exception {

    int patternNr = 4;
    int patternSign = 2;
    this.criticalTime = criticalTimeFollowing;
```

```

    this.patternTolerance = followingTolerance;

    //1.Schritt: Ueberpruefung, ob sich 2 POs auf gleichem bzw. benachbarten Segmenten bewegen
    checkMovementsOnNeighbouringSegments(patternNr, neighbourStreetOrder);

    //2.Schritt: Testen, ob sich POs in die gleiche Richtung bewegen auf den benachbarten Segmenten
    checkMovementDirectionOnSegment(patternNr);

    //3.Schritt: Testen, ob sich POs genügend lange auf benachbarten Segmenten in die gleiche Richtung bewegen
    patternLengthCheck(patternNr, patternSign);
}

/**
 * Methode zur Detektion von Intersektionen (Kreuzen der Bewegungspfade) individuelle Perspektive
 * @param intersectionDistance Parameter kritische Intersektionsdistanz
 * @param intersectionDuration Parameter kritische Zeitdauer Intersektion
 * @return void
 */

public void intersectionDetectorIndPersp(double intersectionDistance, double intersectionDuration)
    throws Exception {

    int patternNr = 5;
    int patternSign = 1;
    this.criticalTime = intersectionDuration;
    this.patternTolerance = 0;

    //1.Schritt: Bestimmen des Zeitpunkts, an dem Bewegungspfade kreuzen -> Distanz minimal
    helper = 1;
    for(int i=0; i<pObjGr.getGroupSize(); i++) {
        for(int j=helper; j<pObjGr.getGroupSize(); j++) {
            double currentMinDistance = 1000000;
            int currentMinDistanceIndex = -1;
            for(int k=0; k<pObjGr.getPO(i).getAmountOfPoints(); k++) {
                while(k<pObjGr.getPO(i).getAmountOfPoints() && pObjGr.getPO(i).movementPatterns[j][k][1] == 4) {
                    int l = (int) (k - (pObjGr.getPO(j).getPoint(0).getTime()
                        - pObjGr.getPO(i).getPoint(0).getTime()));
                    double testDistance = pObjGr.getPO(i).getPoint(k).
                        distanceInMeter(pObjGr.getPO(j).getPoint(l));
                    if(testDistance < currentMinDistance) {
                        currentMinDistance = testDistance;
                        currentMinDistanceIndex = k;
                    }
                }
            }
            k++;
        }
    }
}

```

```

        }
        if(currentMinDistance != 1000000) {
            if(currentMinDistance < intersectionDistance) {
                pObjGr.getPO(i).movementPatterns[j][currentMinDistanceIndex][patternNr] = 1.0;
            }
            currentMinDistance = 1000000;
            currentMinDistanceIndex = -1;
            k--;
        }
    }
    helper++;
}

//2.Schritt: Ueberpruefung ob Bewegungsmuster genuegend lange aufrechterhalten wird
patternLengthCheck(patternNr, patternSign);
}

/**
 * Methode zur Detektion von Intersektionen (Kreuzen der Bewegungspfade) räumliche Perspektive
 * @param neighbourStreetOrder Parameter Nachbarschaftsgrösse Intersektion
 * @param criticalTimeIntersection Parameter kritische Zeitdauer Intersektion
 * @param intersectionTolerance Toleranzparameter Intersektion
 * @return void
 */

public void intersectionDetectorSpacePersp(int neighbourStreetOrder, double criticalTimeIntersection,
    double intersectionTolerance) throws Exception {

    int patternType = 5;
    int patternSign = 1;
    this.criticalTime = criticalTimeIntersection;
    this.patternTolerance = intersectionTolerance;

    //1.Schritt: Untersuchung, ob sich POs aufeinander zubewegen (Approach) waehrend sie sich in
    //der gleichen Strassennachbarschaft bewegen
    approachingDetectorSpacePersp(neighbourStreetOrder, criticalTimeIntersection, intersectionTolerance);

    //2.Schritt: Bestimmen des Confrontation-Zeitpunkts (nach Approach)
    helper = 1;
    for(int i=0; i<pObjGr.getGroupSize(); i++) {
        for(int k=helper; k<pObjGr.getGroupSize(); k++) {
            for(int j=0; j<pObjGr.getPO(i).getAmountOfPoints(); j++) {
                boolean encounterTester = false;

```

```

        int encounterIndex = -1;
        while (j < pObjGr.getPO(i).getAmountOfPoints()
            && pObjGr.getPO(i).movementPatterns[k][j][0] == 3.0
            && pObjGr.getPO(i).movementPatterns[k][j][1] == 4.0) {

            encounterIndex = j;
            encounterTester = true;
            j++;

        }
        if (encounterTester) {
            pObjGr.getPO(i).movementPatterns[k][encounterIndex][patternType] = 1.0;
            encounterTester = false;
            j--;
        }
    }
    helper++;
}

//3.Schritt: Ueberpruefung ob Bewegungsmuster genuegend lange aufrechterhalten wird
this.criticalTime = 1;
patternLengthCheck(patternType, patternSign);
}

//Hilfsmethoden

/**
 * Methode zur Untersuchung der Distanzbeziehungen zwischen den Punktobjekten
 * @param patternType Typ des Bewegungsmusters
 * @return void
 */

public void observeDistanceRelationships(int patternType) throws Exception {
    helper = 1;
    double distance = 1000000;
    for (int i=0; i<pObjGr.getGroupSize(); i++) {
        //System.out.println(" ... Punktobjekt "+pObjGr.getPO(i).getName());
        for (int k=helper; k<pObjGr.getGroupSize(); k++) {
            //Ueberpruefung nur durchfuehren, falls sich die Bewegungsdaten zweier Punktobjekte zeitlich ueberlappen
            if ((pObjGr.getPO(k).getPoint(0).getTime() >= pObjGr.getPO(i).getPoint(0).getTime()
                && pObjGr.getPO(k).getPoint(0).getTime()
                <= pObjGr.getPO(i).getPoint(pObjGr.getPO(i).getAmountOfPoints()-1).getTime())
                || (pObjGr.getPO(i).getPoint(0).getTime() >= pObjGr.getPO(k).getPoint(0).getTime()
                && pObjGr.getPO(i).getPoint(0).getTime()

```

```

        <= pObjGr.getPO(k).getPoint(pObjGr.getPO(k).getAmountOfPoints()-1).getTime()) {
    int l=0;
    for(int j=0;j<pObjGr.getPO(i).getAmountOfPoints(); j++) {
        distance = 1000000;
        if(l<pObjGr.getPO(k).getAmountOfPoints()-1){
            Point p1 = pObjGr.getPO(i).getPoint(j);
            Point p2 = pObjGr.getPO(k).getPoint(l);
            //falls Zeitpunkt bereits vorbei, wird zum nächsten Punkt gesprungen
            while(p1.getTime()-p2.getTime()>0) {
                l++;
                p2 = pObjGr.getPO(k).getPoint(l);
            }
            try {
                if(p1.getTime()-p2.getTime() == 0) {
                    if(p1.getX() > 0 && p2.getX() > 0) {
                        //nur falls fuer POs zu diesem Zeitpunkt
                        //Bewegungsdaten verfuegbar sind
                        distance = p1.distanceInMeter(p2);
                    }
                    l++;
                }
            }
            catch (Exception e){
                distance = 1000000;
            }
            if(distance<maximalDistance && distance>minimalDistance) {
                if(patternType==4) { //Bewegungsmuster 4 = Folgen
                    //2.Schritt des PEDetectors Individual Perspective: Ueberpruefung,
                    //ob sich ein Objekt im Frontbereich des anderen befindet
                    frontAreaDetector(pObjGr.getPO(i), pObjGr.getPO(k), j, (l-1));
                }
                else if(patternType==0 || patternType==2){
                    //Bewegungsmuster 0 = Annahern, Bewegungsmuster 2 = Separieren
                    pObjGr.getPO(i).movementPatterns[k][j][patternType]=distance;
                }
                else {
                    //Bewegungsmuster 1 = Coincidence, 3 = Leaving oder 5 = Intersektion
                    pObjGr.getPO(i).movementPatterns[k][j][patternType]=1.0;
                    //pObjGr.getPO(k).movementPatterns[i][l][patternType]=1.0;
                }
            }
        }
    }
}

```

```

    }
    helper++;
}

/**
 * Methode zur Untersuchung ob sich ein Punktobjekt im Frontbereich eines anderen befindet
 * @param po1      Punktobjekt 1
 * @param po2      Punktobjekt 2
 * @param stepPO1  Zeitschritt von PO1
 * @param stepPO2  Zeitschritt von PO2
 * @return void
 */

public void frontAreaDetector(PointObject po1, PointObject po2, int stepPO1, int stepPO2) {
    double movementDirectionPO1;
    double movementDirectionPO2;
    //Test ob eines der beiden Punktobjekte im Frontbereich des anderen liegt
    try {
        movementDirectionPO1 = po1.getPoint(stepPO1-1).azimuth(po1.getPoint(stepPO1));
    }
    catch (Exception e) {
        movementDirectionPO1 = po1.getPoint(stepPO1).azimuth(po1.getPoint(stepPO1+1));
    }
    try {
        movementDirectionPO2 = po2.getPoint(stepPO2-1).azimuth(po2.getPoint(stepPO2));
    }
    catch (Exception e) {
        movementDirectionPO2 = po2.getPoint(stepPO2).azimuth(po2.getPoint(stepPO2+1));
    }
    double angleBetweenPO1PO2 = po1.getPoint(stepPO1).azimuth(po2.getPoint(stepPO2));
    double angleBetweenPO2PO1 = po2.getPoint(stepPO2).azimuth(po1.getPoint(stepPO1));
    boolean tester1 = false;
    boolean tester2 = false;
    if((angleBetweenPO1PO2 <= (movementDirectionPO1+(criticalAngle/2))
    && (angleBetweenPO1PO2 >= (movementDirectionPO1-(criticalAngle/2)))) {
        //PO2 befindet sich im Frontbereich von PO1
        tester1 = true;
    }
    if((angleBetweenPO2PO1 <= (movementDirectionPO2+(criticalAngle/2))
    && (angleBetweenPO2PO1 >= (movementDirectionPO2-(criticalAngle/2)))) {
        //PO1 befindet sich im Frontbereich von PO2
        tester2 = true;
    }
}

```

```

//nur falls nicht beide im Frontbereich des anderen -> keine Verfolgung sondern aufeinander zu bewegen
if(!tester1 || !tester2) {
    if(tester1) {
        po1.movementPatterns[(po2.getPointObjectNr())][stepPO1][4]=1.0;
    }
    else if(tester2) {
        po2.movementPatterns[(po1.getPointObjectNr())][stepPO2][4]=1.0;
    }
}

}

/**
 * Methode um zu ueberpruefen, ob das Bewegungsmuster genuegend lange aufrechterhalten wird
 * @param patternType Typ des Bewegungsmusters
 * @param patternSign Kennzeichen des Bewegungsmuster im Bewegungsmusterarray
 * @return void
 */

public void patternLengthCheck(int patternType, int patternSign) {
    int helper = 1;
    for(int i=0; i<pObjGr.getGroupSize(); i++) {
        containmentCounter = 0;
        toleranceChecker = 0;
        for(int k=0; k<pObjGr.getGroupSize(); k++) {
            containmentCounter = 0;
            toleranceChecker = 0;
            for(int j=0; j<pObjGr.getPO(i).getAmountOfPoints(); j++) {
                if(pObjGr.getPO(i).movementPatterns[k][j][patternType]==patternSign) {
                    startChecker=1;
                    containmentCounter++;
                    if(containmentCounter>=criticalTime){
                        for(int l=(j-(containmentCounter-1)); l<=j; l++) {
                            //minus 1, da j noch nicht um 1 erhoeht, containmentCounter aber schon!
                            int m = (int) (l-(pObjGr.getPO(k).getPoint(0).getTime()-
                                pObjGr.getPO(i).getPoint(0).getTime()));
                            pObjGr.getPO(i).movementPatterns[k][l][patternType]=patternSign+1.0;
                            if(patternType == 1 && reactionPatternType.equals("Pursuit/Evasion")) {
                                //l=Folgen -> ansonsten kann unter Umstaenden
                                //Approach nicht einem Folgen zugeordnet werden
                                pObjGr.getPO(k).movementPatterns[i][m][patternType]=patternSign+1.0;
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

else {
    //keine Verfolgung zu diesem Zeitpunkt registriert
    // -> Unterbruch im Muster oder Abbruch
    if(startChecker > 0) {
        //bereits ein Zeitpunkt mit Verfolgung gefunden
        if(toleranceChecker <= patternTolerance) {
            //nur Unterbruch, Toleranz noch eingehalten
            if(toleranceChecker == 0) {
                //beim ersten Unterbruch wird naechster Startpunkt bestimmt
                nextStart = j;
            }
            toleranceChecker++;
            containmentCounter++;
        }
        else {
            //Abbruch, da Toleranz ueberschritten
            j=nextStart;
            containmentCounter = 0;
            toleranceChecker=0;
            startChecker = 0;
        }
    }
    else {
        //naechster Punkt
    }
}
}

//Ausgabe der gefundenen Bewegungsmuster
collectAndPrintOutPatterns(i, patternType, patternType, (int)(patternSign+1.0), 2);
helper++;
}
}

/**
 * Methode um zu ueberpruefen, ob sich die POs aufeinander zu bewegen oder sich voneinander entfernen
 * @param patternType Nummer des Bewegungsmusters: 0=Annaherung, 2=Separation
 * @return void
 */

public void observeDistanceTrends(int patternType) {
    //Untersuchung des Verlaufs (Trends) der Distanzen zwischen den Punktobjekten
    helper=1;
    for(int i=0; i<pObjGr.getPointObjectVector().size(); i++) {

```

```

    for(int j=helper; j<pObjGr.getPointObjectVector().size(); j++) {
        PointObject pol = pObjGr.getPO(i);
        for(int k=0; k<pol.getAmountOfPoints()-1; k++) {
            double distanceDifference = pol.movementPatterns[j][k+1][patternType]
                -pol.movementPatterns[j][k][patternType];

            if(distanceDifference == 0) {
                //1.Fall: Punktobjekte haben Distanz zueinander nicht veraendert
                pol.movementPatterns[j][k][patternType] = 0.0;
            }
            else if(distanceDifference < 0) {
                //2.Fall: Punktobjekte haben sich angenaehert
                if(patternType == 0) {
                    //Bewegungsmuster 0 = Annaeherung
                    pol.movementPatterns[j][k][patternType] = 1.0;
                }
                else {
                    pol.movementPatterns[j][k][patternType] = 0.0;
                }
            }
            else if(distanceDifference > 0) {
                //Punktobjekte haben sich voneinander entfernt
                if(patternType == 2) {
                    //Bewegungsmuster 2 = Separation
                    pol.movementPatterns[j][k][patternType] = 1.0;
                }
                else {
                    pol.movementPatterns[j][k][patternType] = 0.0;
                }
            }
        }
    }
}

/**
 * Methode um zu prüfen, ob der Winkel der Bewegungsrichtungsänderung den kritischen Winkel ueberschreitet
 * @param criticalDirectionChange Parameter kritischer Winkel der Bewegungsrichtungsänderung
 * @param patternType Typ des Bewegungsmusters
 * @param patternSign Kennzeichen des Bewegungsmuster im Bewegungsmusterarray
 * @return void
 */

public void checkOrientationChange(double criticalDirectionChange, int patternNr, int patternSign) {
    int helper = 1;

```

```

for(int i=0; i<pObjGr.getGroupSize(); i++) {
    containmentCounter = 0;
    toleranceChecker = 0;
    for(int k=helper; k<pObjGr.getGroupSize(); k++) {
        containmentCounter = 0;
        toleranceChecker = 0;
        for(int j=1; j<pObjGr.getPO(i).getAmountOfPoints(); j++) {
            boolean abruptDirectionChangeTester = false;
            int stepCounter = 0;
            double maxDirectionChange1 = 0;
            double maxTime1 = 0;
            double maxDirectionChange2 = 0;
            double maxTime2 = 0;
            while(j<pObjGr.getPO(i).getAmountOfPoints()-1
                && pObjGr.getPO(i).movementPatterns[k][j][patternNr]==patternSign) {

                int m = (int) (j-(pObjGr.getPO(k).getPoint(0).getTime()
                    -pObjGr.getPO(i).getPoint(0).getTime()));

                stepCounter++;
                //Bestimmung der Bewegungsrichtungen von 2 nacheinanderfolgenden Schritten
                double movementDirection1A;
                if(m>0) {
                    movementDirection1A = pObjGr.getPO(i).getPoint(j-1).
                        azimuth(pObjGr.getPO(i).getPoint(j));
                }
                else {
                    // falls erstes Koordinatenpaar
                    movementDirection1A = -1;
                }
                double movementDirection1B = pObjGr.getPO(i).getPoint(j).
                    azimuth(pObjGr.getPO(i).getPoint(j+1));
                if(movementDirection1A < 0) {
                    //falls keine Bewegung stattgefunden hat werden Bewegungsrichtungen angeglichen
                    movementDirection1A = movementDirection1B;
                }
                if(movementDirection1B < 0) {
                    //falls keine Bewegung stattgefunden hat werden Bewegungsrichtungen angeglichen
                    movementDirection1B = movementDirection1A;
                }
                double directionChange1 = Math.abs(movementDirection1A-movementDirection1B);
                if(directionChange1 > 180) {
                    //falls Bewegungsrichtungswechsel groesser als 180 Grad
                    directionChange1 = 360-directionChange1;
                }
            }
        }
    }
}

```

```

double movementDirection2A;
if(m>0) {
    movementDirection2A = pObjGr.getPO(k).getPoint(m-1).
        azimuth(pObjGr.getPO(k).getPoint(m));
}
else {
    //erste Positionsangabe dieses Punktobjekts
    movementDirection2A = -1;
}
double movementDirection2B = pObjGr.getPO(k).getPoint(m).
    azimuth(pObjGr.getPO(k).getPoint(m+1));

if(movementDirection2A < 0) {
    //falls keine Bewegung stattgefunden hat werden Bewegungsrichtungen angeglichen
    movementDirection2A = movementDirection2B;
}
if(movementDirection2B < 0) {
    //falls keine Bewegung stattgefunden hat werden Bewegungsrichtungen angeglichen
    movementDirection2B = movementDirection2A;
}
double directionChange2 = Math.abs(movementDirection2A-movementDirection2B);
if(directionChange2 > 180) {
    //falls Bewegungsrichtungswechsel groesser als 180 Grad
    directionChange2 = 360-directionChange2;
}
if(directionChange1 > maxDirectionChange1) {
    //falls Bewegungsrichtungswechsel von PO1 groesser als der bis anhin maximale Wert
    maxDirectionChange1 = directionChange1;
    maxTime1 = j;
}
if(directionChange2 > maxDirectionChange2) {
    //falls Bewegungsrichtungswechsel von PO2 groesser als der bis anhin maximale Wert
    maxDirectionChange2 = directionChange2;
    maxTime2 = j;
}
if(directionChange1 > criticalDirectionChange
    || directionChange1 < -criticalDirectionChange
    || directionChange2 > criticalDirectionChange
    || directionChange2 < -criticalDirectionChange) {
    //falls maximaler Bewegungsrichtungswechsel von PO1 oder PO2
    //groesser als der kritische Wert
    int windowleft = 0; //mit einem Fenster vor und nach dem Zeitpunkt des
    int windowright = 0; //maximalen Bewegungsrichtungswechsel kann geprueft
    int segmentChangeCounter1 = 0; //werden, ob ein Abbiegen zwischen 2 Segmenten stattfindet

```

```

        for(int q=(j-windowleft); q<j>windowright; q++) {
            if(q<pObjGr.getPO(i).getAmountOfPoints()-1
                && pObjGr.getPO(i).getPoint(q).getAssignedRoadSegment()
                != pObjGr.getPO(i).getPoint(q+1).getAssignedRoadSegment()) {
                //Uebergang zwischen zwei Segmenten durch PO1
                segmentChangeCounter1++;
            }
        }
        int segmentChangeCounter2 = 0;
        for(int q=(j-windowleft); q<j>windowright; q++) {
            if(q<pObjGr.getPO(k).getAmountOfPoints()-1
                && pObjGr.getPO(k).getPoint(q).getAssignedRoadSegment()
                != pObjGr.getPO(k).getPoint(q+1).getAssignedRoadSegment()) {
                //Uebergang zwischen zwei Segmenten durch PO2
                segmentChangeCounter2++;
            }
        }
        if(segmentChangeCounter1 == 0 && segmentChangeCounter2 == 0) {
            //nur falls kein Abbiegen stattgefunden hat, ist Bewegungsrichtungswechsel
            // als Folge von Meiden zwischen 2 POs hervorgerufen worden
            abruptDirectionChangeTester = true;
        }
    }
    j++;
}
if(abruptDirectionChangeTester) {
    //Winkel des Bewegungsrichtungswechsels groesser als kritischer Winkel && kein Abbiegen
    j--;
    for(int q=j; q>(j-stepCounter); q--) {
        pObjGr.getPO(i).movementPatterns[k][q][patternNr] = patternSign+1.0;
    }
}
}
}
helper++;
}
}
}

```

```

/**
 * Methode um zu prüfen, ob eine minimale Distanz unterschritten wurde
 * @param criticalDistance minimale erlaubte Distanz
 * @return void
 */

public void checkMinimalDistance(double criticalDistance) {
    for(int i=0; i<pObjGr.getPointObjectVector().size(); i++) {
        for(int j=0; j<pObjGr.getPointObjectVector().size(); j++) {
            for(int k=0; k<pObjGr.getPO(i).getAmountOfPoints(); k++) {
                double minimalDistance = 1000000;
                int startIndex = 0;
                while(k < pObjGr.getPO(i).getAmountOfPoints()
                    && pObjGr.getPO(i).movementPatterns[j][k][1]==4) {
                    if(minimalDistance == 1000000) {
                        startIndex = k;
                    }
                    int m = (int) (k - (pObjGr.getPO(j).getPoint(0).getTime()
                        - pObjGr.getPO(i).getPoint(0).getTime()));
                    double testDistance
                        = pObjGr.getPO(i).getPoint(k).distanceInMeter(pObjGr.getPO(j).getPoint(m));
                    if(testDistance < minimalDistance) {
                        //Distanz kleiner als Minimaldistanz
                        minimalDistance = testDistance;
                    }
                    k++;
                }
                if(minimalDistance != 1000000) {
                    k--;
                    if(minimalDistance >= criticalDistance) {
                        //Minimaldistanz groesser/gleich kritische Distanz
                        for(int q=startIndex; q<=k; q++) {
                            pObjGr.getPO(i).movementPatterns[j][q][1]=5.0;
                        }
                    }
                }
            }
        }
    }
}

```

```

/**
 * Methode um zu ueberpruefen, wie gross die Verzoegerungszeit zwischen Aktion und Reaktion ist
 * und ob diese die erlaubte Zeit ueberschreitet
 * @param criticalTimeLag          kritische Verzoegerungszeit
 * @param actionPatternType       Nummer des Aktionsmusters
 * @param actionPatternSign       Kennzeichen des Aktionsbewegungsmusters
 * @param reactionPatternType     Nummer des Reaktionsmusters
 * @param reactionPatternSign     Kennzeichen des Reaktionsbewegungsmusters
 * @param reactionPatternNr       Nummer des Reaktionsbewegungsmusters
 * @param reactionPatternArrayNr  Index des Reaktionsbewegungsmusters im Array mit allen Reaktionsbewegungsmustern
 * @return void
 */

public void checkTimeLag(double criticalTimeLag, int actionPatternType, int actionPatternSign,
    int reactionPatternType, int reactionPatternSign, int reactionPatternNr, int reactionPatternArrayNr) {
    //Testen, wie gross die Zeitdifferenz zwischen Aktionsbewegung und Reaktionsbewegung ist
    this.criticalTimeLag = criticalTimeLag;
    helper=1;
    for(int i=0; i<pObjGr.getPointObjectVector().size(); i++) {
        for(int j=0; j<pObjGr.getPointObjectVector().size(); j++) {
            double startTimeActionPattern = 0;
            int startIndexActionPattern = 0;
            double endTimeActionPattern = 0;
            int endIndexActionPattern = 0;
            double startTimeReactionPattern = 0;
            int endIndexReactionPattern = 0;
            int startIndexReactionPattern = 0;
            double timeLag = 100000;
            PointObject pol = pObjGr.getPO(i);
            PointObject po2 = pObjGr.getPO(j);
            boolean approachStartTester = false;
            boolean approachEndTester = false;
            for(int k=0; k<pol.getAmountOfPoints(); k++) {
                startIndexReactionPattern = 0;
                endIndexReactionPattern = 0;
                while(k < pol.getAmountOfPoints()
                    && pol.movementPatterns[j][k][actionPatternType]==actionPatternSign) {
                    //Aktion findet statt
                    if(!approachStartTester) {
                        startTimeActionPattern=pol.getPoint(k).getTime();
                        startIndexActionPattern = k;
                        approachStartTester = true;
                    }
                }
                endTimeActionPattern = pol.getPoint(k).getTime();
            }
        }
    }
}

```

```

        endIndexActionPattern = k;
        approachEndTester = true;
        //falls Reaktionsmuster ebenfalls schon begonnen hat
        if(pol.movementPatterns[j][k][reactionPatternType]==reactionPatternSign
            && startIndexReactionPattern == 0) {
            startIndexReactionPattern = k;
            startTimeReactionPattern = pol.getPoint(k).getTime();
        }
        if(pol.movementPatterns[j][k][reactionPatternType]==reactionPatternSign
            && startIndexReactionPattern != 0) {
            endIndexReactionPattern = k;
        }
        k++;
    }
    if(approachEndTester) {
        k--;
        approachEndTester = false;
        approachStartTester = false;
    }
    //falls Aktion und Reaktion nacheinander ablaufen
    if(pol.movementPatterns[j][k][reactionPatternType]==reactionPatternSign
        && startIndexReactionPattern == 0.0) {
        startTimeReactionPattern = pol.getPoint(k).getTime();
        startIndexReactionPattern = k;
        timeLag = startTimeReactionPattern - endTimeActionPattern;
        if(timeLag < this.criticalTimeLag) {
            //falls kritische Zeitverzögerung zwischen Aktion und Reaktion
            //nicht ueberschritten ist
            //Reaktionsbewegungsmuster 4/5 = Konfrontation
            if(reactionPatternNr == 4 || reactionPatternNr == 5) {
                //nur Zeitpunkt der Konfrontation wird herausgeschrieben
                pol.reactionMovementPatterns[j][startIndexActionPattern]
                    [reactionPatternArrayNr] = 1.0;
                while(k < pol.getAmountOfPoints()
                    && pol.movementPatterns[j][k][reactionPatternType]
                        ==reactionPatternSign) {
                    k++;
                }
            }
            //Reaktionsbewegungsmuster 0/1 = Folgen
            else if(reactionPatternNr == 0 || reactionPatternNr == 1) {
                //Zeitschritte von Folgen werden herausgeschrieben
                while(k < pol.getAmountOfPoints()
                    && pol.movementPatterns[j][k][reactionPatternType]

```



```

    }
    //Reaktionsbewegungsmuster 0/1 = Folgen
    else if(reactionPatternNr == 0 || reactionPatternNr == 1) {
        //Zeitschritte von Folgen werden herausgeschrieben
        int startIndex = startIndexReactionPattern;
        int endIndex = k;
        for(int l=startIndex; l<=endIndex; l++) {
            pol.reactionMovementPatterns[j][l][reactionPatternArrayNr] = 1.0;
        }
        while(k < pol.getAmountOfPoints()
            && pol.movementPatterns[j][k][reactionPatternType]
                ==reactionPatternSign) {
            pol.reactionMovementPatterns[j][k][reactionPatternArrayNr] = 1.0;
            k++;
        }
    }
    else {
        //Meiden
        int startIndex = startIndexActionPattern;
        int endIndex = endIndexActionPattern;
        for(int l=startIndex; l<=endIndex; l++) {
            pol.reactionMovementPatterns[j][l][reactionPatternArrayNr] = 1.0;
        }
        int counter = 0;
        while(k < pol.getAmountOfPoints()
            && pol.movementPatterns[j][k][reactionPatternType]
                ==reactionPatternSign) {
            if(counter < criticalTime) {
                pol.reactionMovementPatterns[j][k][reactionPatternArrayNr]=1.0;
                counter++;
            }
            k++;
        }
    }
}
}
//falls Aktion und Reaktionsmuster parallel laufen und Reaktion vor der Aktion fertig ist
//kommt zum Beispiel im Fall von Meeting und Folgen oder Meeting und Konfrontation vor
else if(startIndexReactionPattern != 0 ) {
    int startIndex = startIndexActionPattern;
    int endIndex = endIndexActionPattern;
    timeLag = startTimeReactionPattern - startTimeActionPattern;
    if(timeLag < this.criticalTimeLag) {
        if(reactionPatternNr == 0 || reactionPatternNr == 1) {

```



```

    }
}
for(int i=0; i<pursuitArray.size(); i++) {
    String[] lineArray1 = pursuitArray.get(i).split(" ");
    for(int j=0; j<confrontationArray.size(); j++) {
        String[] lineArray2 = confrontationArray.get(j).split(" ");
        if(confrontationArray.get(j).contains(lineArray1[11])
            && confrontationArray.get(j).contains(lineArray1[12])
            && confrontationArray.get(j).contains(lineArray1[14])
            && confrontationArray.get(j).contains(lineArray1[15])) {

            if((Time.transformTimeToDouble(lineArray2[9]) >= Time.transformTimeToDouble(lineArray1[9]) &&
                Time.transformTimeToDouble(lineArray2[9]) <= Time.transformTimeToDouble(lineArray1[17])) ||
                (Time.transformTimeToDouble(lineArray2[17]) >= Time.transformTimeToDouble(lineArray1[9]) &&
                Time.transformTimeToDouble(lineArray2[17]) <= Time.transformTimeToDouble(lineArray1[17]))) {
                String movementPatternDescription = "";
                for(int k=0; k<lineArray1.length-1; k++) {
                    movementPatternDescription = movementPatternDescription + lineArray1[k]+" ";
                }
                movementPatternDescription = movementPatternDescription + " mit Confrontation (detektiert)";
                movementPatternArray.add(movementPatternDescription);
                System.out.println(movementPatternDescription);
                confrontationArray.remove(j);
            }
        }
    }
}

/**
 * Methode zur Untersuchung der Zeit vor einem Zusammensein
 * -> dabei wird geprüft, ob vor einem Zusammensein bereits einmal ein Zusammensein stattgefunden hat
 * @param coincidenceFreeTime    kritische Zeitdauer vor einem Zusammensein,
 *                               in welcher kein anderes Zusammensein stattfinden darf
 * @param patternType            Typ des Bewegungsmusters
 * @param patternSign            Index des Bewegungsmuster im Bewegungsmusterarray
 * @return void
 */

public void observeTimeBeforeCoincidence(double coincidenceFreeTime, int patternNr, int patternSign) {
    // -> um sicherzustellen, dass sich POs nicht bereits kurze Zeit vorher getroffen haben
    // -> falls sich POs kurze Zeit vorher getroffen haben, wird Aufeinandertreffen herausgefiltert
    helper = 1;
    for(int i=0; i<pObjGr.getGroupSize(); i++) {

```

```

for(int j=helper; j<pObjGr.getGroupSize(); j++) {
    for(int k=pObjGr.getPO(i).getAmountOfPoints()-1; k>=0; k--) {
        int meetingStepCounter = 0;
        int arrivingStepCounter = 0;
        while(k>=0 && pObjGr.getPO(i).movementPatterns[j][k][1] == 2) {
            meetingStepCounter++;
            k--;
        }
        if(meetingStepCounter != 0) {
            while(k>0 && pObjGr.getPO(i).movementPatterns[j][k][1] != 2) {
                arrivingStepCounter++;
                k--;
            }
            if(arrivingStepCounter > coincidenceFreeTime-1) { //immer 1 zuwenig
                //falls Zeitdauer ohne Zusammensein groesser als der kritische Wert
                for(int l=k+arrivingStepCounter+meetingStepCounter; l>k+arrivingStepCounter; l--) {
                    pObjGr.getPO(i).movementPatterns[j][l][1][patternNr] = patternSign+1.0;
                }
            }
        }
        if(meetingStepCounter != 0) {
            k++;
        }
    }
    helper++;
}
//Ueberpruefung ob Bewegungsmuster genuegend lange aufrechterhalten wird
patternSign++;
patternLengthCheck(patternNr, patternSign);
}

/**
 * Methode zur Ueberpruefung, ob 2 POs sich auf gleichem bzw. benachbartem Segment bewegen
 * @param patternNr          Nummer des Bewegungsmusters
 * @param neighbourStreetOrder Ordnung der Strassennachbarschaft
 *
 * -> 0 = nur assignedRoadSegment wird einbezogen
 * -> 1 = nur benachbarte Segmente werden einbezogen
 * -> 2 = auch benachbarte der benachbarten Segmente werden einbezogen
 *
 * @return void
 */

public void checkMovementsOnNeighbouringSegments(int patternNr, int neighbourStreetOrder) {

```

```

//Paarweiser Vergleich zwischen den POs ob sie sich in der gleichen Nachbarschaft aufhalten
String neighbourStreets = "";
StreetGroup neighbourSegments = new StreetGroup();
helper = 1;
for(int i=0; i<pObjGr.getGroupSize(); i++) {
    for(int k=helper; k<pObjGr.getGroupSize(); k++) {
        int lastAssignedSegment1 = -1;
        //Ueberpruefung nur durchfuehren, falls sich die Bewegungsdaten zweier Punktobjekte zeitlich ueberlappen
        if((pObjGr.getPO(k).getPoint(0).getTime() >= pObjGr.getPO(i).getPoint(0).getTime()
            && pObjGr.getPO(k).getPoint(0).getTime()
                <= pObjGr.getPO(i).getPoint(pObjGr.getPO(i).getAmountOfPoints()-1).getTime())
            || (pObjGr.getPO(i).getPoint(0).getTime() >= pObjGr.getPO(k).getPoint(0).getTime()
            && pObjGr.getPO(i).getPoint(0).getTime()
                <= pObjGr.getPO(k).getPoint(pObjGr.getPO(k).getAmountOfPoints()-1).getTime())) {
            int l=0;
            for(int j=0; j<pObjGr.getPO(i).getAmountOfPoints(); j++) {
                if(l<pObjGr.getPO(k).getAmountOfPoints()-1){
                    Point p1 = pObjGr.getPO(i).getPoint(j);
                    Point p2 = pObjGr.getPO(k).getPoint(l);
                    //falls Zeitpunkt bereits vorbei, wird zum naechsten Punkt gesprungen
                    while(p1.getTime()-p2.getTime()>0) {
                        l++;
                        p2 = pObjGr.getPO(k).getPoint(l);
                    }
                    try {
                        if(p1.getTime()-p2.getTime() == 0) {
                            int assignedSegment1 = p1.getAssignedRoadSegment();
                            int assignedSegment2 = p2.getAssignedRoadSegment();
                            if(neighbourStreetOrder > 0) {
                                //Bestimmung der Segmentnachbarschaft von PO1
                                if(lastAssignedSegment1 != assignedSegment1) {
                                    neighbourSegments = strGr.streetGroup.get(assignedSegment1).
                                        getNeighbourStreets(neighbourStreetOrder);
                                    lastAssignedSegment1 = assignedSegment1;
                                }
                                //Untersuchung, ob assignedSegment2
                                //in der Segmentnachbarschaft von PO1 enthalten ist
                                for(int m=0; m<neighbourSegments.streetGroup.size(); m++) {
                                    if(neighbourSegments.streetGroup.get(m).getstr_id()
                                        == assignedSegment2) {
                                        pObjGr.getPO(i).movementPatterns[k][j][patternNr]=1.0;
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```



```
double movementDirectionPO2 = 0;
double differenceOfMovementDirection = 0;
if (pol.movementPatterns[j][k][patternNr]==1) {
    l=(int) (k-startTimeDifference);
    if (pol.getPoint(k).getX() > 0 && po2.getPoint(l).getX() > 0) {
        //um zu verhindern, dass Punkte mit keinen Koordinaten verglichen werden
        try {
            movementDirectionPO1 = pol.getPoint(k-1).azimuth(pol.getPoint(k));
        }
        catch (Exception e) {
            movementDirectionPO1 = pol.getPoint(k).azimuth(pol.getPoint(k+1));
        }
        try {
            movementDirectionPO2 = po2.getPoint(l-1).azimuth(po2.getPoint(l));
        }
        catch (Exception e) {
            movementDirectionPO2 = po2.getPoint(l).azimuth(po2.getPoint(l+1));
        }

        if (movementDirectionPO1 >= 0) {
            movementDirectionPO1PrevStep = movementDirectionPO1;
        }
        if (movementDirectionPO2 >= 0) {
            movementDirectionPO2PrevStep = movementDirectionPO2;
        }
        //falls sich eines der POs nicht bewegt,
        //wird Bewegungsrichtung des letzten Zeitschritts verwendet
        if (movementDirectionPO1 < 0) {
            movementDirectionPO1 = movementDirectionPO1PrevStep;
        }
        if (movementDirectionPO2 < 0) {
            movementDirectionPO2 = movementDirectionPO2PrevStep;
        }

        //1.Fall: POs bewegen sich auf dem gleichen Segment
        if (pol.getPoint(k).getAssignedRoadSegment()==po2.getPoint(l).getAssignedRoadSegment()) {
            //Differenz zwischen den Bewegungsrichtungen wird berechnet
            differenceOfMovementDirection =
                Math.abs(movementDirectionPO1-movementDirectionPO2);
            //falls Differenz der Bewegungsrichtungen kleiner ist als
            //Toleranz von 5 Grad, bewegen sich die POs in dieselbe Richtung
            if (patternNr == 4 && differenceOfMovementDirection < directionTolerance
                && differenceOfMovementDirection > -directionTolerance) {
                Street road = strGr.streetGroup.get(pol.getPoint(k)).
            }
        }
    }
}
```

```

getAssignedRoadSegment());
double segmentOrientation = road.getStPoint().azimuth(road.getEndPoint());
double segVsMovOrientation =
    Math.abs(segmentOrientation - movementDirectionPO1);
if(segVsMovOrientation > 175 && segVsMovOrientation < 185) {
    //Punktobjekte bewegen sich entgegengesetzt zur Segmentorientierung
    double distPO1SegEndPoint = pol.getPoint(k).
        distanceInMeter(road.getEndPoint());
    double distPO2SegEndPoint = po2.getPoint(l).
        distanceInMeter(road.getEndPoint());
    if(distPO1SegEndPoint > distPO2SegEndPoint) {
        //PO1 befindet sich weiter weg vom Segmentendpunkt als PO2
        //-> PO2 verfolgt PO1
        pObjGr.getPO(j).movementPatterns[i][l][patternNr]=2.0;
    }
    else {
        //PO1 befindet sich naeher beim Segmentendpunkt als PO1
        //-> PO1 verfolgt PO2
        pObjGr.getPO(i).movementPatterns[j][k][patternNr]=2.0;
    }
}
else {
    //Punktobjekte bewegen sich in Richtung der Segmentorientierung
    double distPO1SegEndPoint = pol.getPoint(k).
        distanceInMeter(road.getEndPoint());
    double distPO2SegEndPoint = po2.getPoint(l).
        distanceInMeter(road.getEndPoint());
    if(distPO1SegEndPoint < distPO2SegEndPoint) {
        //PO1 befindet sich naeher beim Segmentendpunkt als PO2
        //-> PO2 verfolgt PO1
        pObjGr.getPO(j).movementPatterns[i][l][patternNr]=2.0;
    }
    else {
        //PO1 befindet sich weiter weg vom Segmentendpunkt als PO2
        //-> PO1 verfolgt PO2
        pObjGr.getPO(i).movementPatterns[j][k][patternNr]=2.0;
    }
}
}
else if(patternNr != 4 && differenceOfMovementDirection > 175) {
    //Punktobjekte bewegen sich aufeinander zu oder voneinander weg
    Street street = strGr.streetGroup.get(pol.getPoint(k).
        getAssignedRoadSegment());
    double segmentOrientation = street.getStPoint().

```

```

                                azimuth(street.getEndPoint());
double diffPO1Seg = segmentOrientation - movementDirectionPO1;
double diffPO2Seg = segmentOrientation - movementDirectionPO2;
double distanceDiffPO1 = street.getStPoint().
                                distanceInMeter(po1.getPoint(k));
double distanceDiffPO2 = street.getStPoint().
                                distanceInMeter(po2.getPoint(l));

if(diffPO1Seg < 5 && diffPO1Seg > -5) {
    //PO1 bewegt sich in Segmentrichtung
    //in diesem Fall muss PO2 naeher beim Segmentstartpunkt liegen,
    //damit sich POs separieren
    if(patternNr == 2 && distanceDiffPO2 < distanceDiffPO1) {
        //POs bewegen sich voneinander weg
        pObjGr.getPO(i).movementPatterns[j][k][patternNr]=2.0;
    }
    else if (patternNr == 0 && distanceDiffPO1 < distanceDiffPO2){
        //POs bewegen sich aufeinander zu
        pObjGr.getPO(i).movementPatterns[j][k][patternNr]=2.0;
    }
}
else if(diffPO2Seg < 5 && diffPO2Seg > -5) {
    //PO2 bewegt sich in Segmentrichtung
    //in diesem Fall muss PO1 naeher beim Segmentstartpunkt liegen,
    //damit sich POs separieren
    if(patternNr == 2 && distanceDiffPO1 < distanceDiffPO2) {
        //POs bewegen sich voneinander weg
        pObjGr.getPO(i).movementPatterns[j][k][patternNr]=2.0;
    }
    else if (patternNr == 0 && distanceDiffPO2 < distanceDiffPO1){
        //POs bewegen sich aufeinander zu
        pObjGr.getPO(i).movementPatterns[j][k][patternNr]=2.0;
    }
}
}
}

//2.Fall: POs bewegen sich nicht auf dem gleichen, auf benachbarten Segmenten
else {
    Point p1 = po1.getPoint(k);
    Point p2 = po2.getPoint(l);
    //ShortestPath-Berechnung wird nur durchgefuehrt, falls sich die
    //POs nicht auf den gleichen Segmenten wie beim vorherigen Schritt befinden
    if(p1.getAssignedRoadSegment() != assignedRoadSegment1

```

```

        || p2.getAssignedRoadSegment() != assignedRoadSegment2) {
    assignedRoadSegment1 = p1.getAssignedRoadSegment();
    assignedRoadSegment2 = p2.getAssignedRoadSegment();
    Interpolator ip = new Interpolator();
    PointObject poNew = new PointObject();
    int start = 0;
    int ende = 1000;
    //Berechnung des kuerzesten Pfades zwischen den POs
    shortestPath = ip.shortestPath(p1, p2, poNew, strGr, start, ende);
}
double diffSum = 0;
Double[] movDirVsSegOrient = new Double[2];

//2.Fall Variante A: POs bewegen sich auf benachbarten Segmenten
//(falls Anzahl Segmente = 2)
//wichtig: in shortestPath-Vector befinden sich noch
//2 Zusatzangaben fuer die Interpolation -> darum Groesse = 4
if(shortestPath.size()==4) {
    int str1 = po1.getPoint(k).getAssignedRoadSegment();
    int str2 = po2.getPoint(l).getAssignedRoadSegment();
    Street street1 = strGr.streetGroup.get(str1);
    Street street2 = strGr.streetGroup.get(str2);
    //Berechnung der Differenz zwischen Bewegungsrichtung
    //und SegmentOrientierung
    double movDirVsSegOrient1 = Math.abs(movementDirectionPO1
        - street1.getSegmentOrientation(street2));
    movDirVsSegOrient[0] = movDirVsSegOrient1;
    double movDirVsSegOrient2 = Math.abs(movementDirectionPO2
        - street2.getSegmentOrientation(street1));
    movDirVsSegOrient[1] = movDirVsSegOrient2;
    diffSum = movDirVsSegOrient1+movDirVsSegOrient2;
}
//2.Fall Variante B: POs bewegen sich auf im 2.Grad benachbarten Segmenten
else {
    int counter = 0;
    Double[] movementDirections = new Double[2];
    movementDirections[0] = movementDirectionPO1;
    movementDirections[1] = movementDirectionPO2;
    for(int s=0; s<shortestPath.size()-3; s++) {
        int str1 = Integer.parseInt(shortestPath.get(s));
        int str2 = Integer.parseInt(shortestPath.get(s+1));
        Street street1 = strGr.streetGroup.get(str1);
        Street street2 = strGr.streetGroup.get(str2);
        if(s==0) {

```

```

        //Berechnung der Differenz zwischen Bewegungsrichtung
        //und Segmentorientierung
        movDirVsSegOrient[counter] =
            Math.abs(movementDirections[counter]
                - street1.getSegmentOrientation(street2));
        counter++;
    }
    else if(s==shortestPath.size()-4) {
        //Berechnung der Differenz zwischen Bewegungsrichtung
        //und Segmentorientierung
        movDirVsSegOrient[counter] =
            Math.abs(movementDirections[counter]
                - street2.getSegmentOrientation(street1));
        counter++;
    }
}
for(double diff : movDirVsSegOrient) {
    diffSum = diffSum + diff;
}
}
//falls die Differenzen zwischen Bewegungsrichtung und Segmentorientierung
//gleich (180 oder 0) sind, bewegen sich beide POs auf Knotenpunkt hin bzw.
// von diesem weg -> nur falls sich movDirVsSegOrient1 & movDirVsSegOrient2
//unterscheiden bewegen sich die POs in dieselbe Richtung bzw. folgen sich
if(diffSum - 180 < directionTolerance
    && diffSum - 180 > -directionTolerance && patternNr == 4) {
    //POs folgen sich
    if(movDirVsSegOrient[0] < directionTolerance
        && movDirVsSegOrient[0] > -directionTolerance) {
        //PO2 verfolgt PO1
        pObjGr.getPO(j).movementPatterns[i][1][4]=2.0;
    }
    else {
        //PO1 verfolgt PO2
        pObjGr.getPO(i).movementPatterns[j][k][4]=2.0;
    }
}
//Separation falls Differenz zwischen Bewegungsrichtungen
//und Segmentorientierungen gleich 0
//bedeutet, dass sich beide POs vom Knotenpunkt wegbewegen
else if(diffSum < directionTolerance
    && diffSum > -directionTolerance && patternNr == 2) {
    //POs separieren sich
    pObjGr.getPO(i).movementPatterns[j][k][2]=2.0;
}

```



```
//Reaktionsbewegungsmuster 0 = Verfolgen/Entfliehen individuelle Perspektive
if(patternCategory == 1) {
    if(patternNr == 0) {
        movementPatternArray.add("    - "+pObjGr.getPO(i).getDate()
        +": Verfolgen/Entfliehen(ind.Persp.) zum Zeitpunkt "+timeText
        +" zwischen "+pObjGr.getPO(i).getNameNr()+" (" +pObjGr.getPO(i).getTeam()
        +") und "+pObjGr.getPO(j).getNameNr()+" (" +pObjGr.getPO(j).getTeam()+") bis ");
        System.out.print("    - Verfolgen/Entfliehen (individuelle Perspektive) zum
        Zeitpunkt "+timeText+" zwischen "+pObjGr.getPO(i).getNameNr()
        +" (" +pObjGr.getPO(i).getTeam()+", Verfolger) und "+pObjGr.getPO(j).getNameNr()
        +" (" +pObjGr.getPO(j).getTeam()+", Fliehender) bis ");
    }
    //Reaktionsbewegungsmuster 1 = Verfolgen/Entfliehen raeumliche Perspektive
    else if(patternNr == 1) {
        movementPatternArray.add("    - "+pObjGr.getPO(i).getDate()
        +": Verfolgen/Entfliehen(raeuuml.Persp.) zum Zeitpunkt "+timeText
        +" zwischen "+pObjGr.getPO(i).getNameNr()+" (" +pObjGr.getPO(i).getTeam()
        +") und "+pObjGr.getPO(j).getNameNr()+" (" +pObjGr.getPO(j).getTeam()+") bis ");
        System.out.print("    - Verfolgen/Entfliehen (raeumliche Perspektive) zum
        Zeitpunkt "+timeText+" zwischen "+pObjGr.getPO(i).getNameNr()
        +" (" +pObjGr.getPO(i).getTeam()+", Verfolger) und "+pObjGr.getPO(j).getNameNr()
        +" (" +pObjGr.getPO(j).getTeam()+", Fliehender) bis ");
    }
    //Reaktionsbewegungsmuster 2 = Meiden individuelle Perspektive
    else if(patternNr == 2) {
        movementPatternArray.add("    - "+pObjGr.getPO(i).getDate()+
        ": Meiden(ind.Persp.) zum Zeitpunkt "+timeText+" zwischen "
        +pObjGr.getPO(i).getNameNr()+" (" +pObjGr.getPO(i).getTeam()
        +") und "+pObjGr.getPO(j).getNameNr()+" (" +pObjGr.getPO(j).getTeam()+") bis ");
        System.out.print("    - Meiden (individuelle Perspektive) zum Zeitpunkt "
        +timeText+" zwischen "+pObjGr.getPO(i).getNameNr()+" (" +pObjGr.getPO(i).getTeam()
        +") und "+pObjGr.getPO(j).getNameNr()+" (" +pObjGr.getPO(j).getTeam()+") bis ");
    }
    //Reaktionsbewegungsmuster 3 = Meiden raeumliche Perspektive
    else if(patternNr == 3) {
        movementPatternArray.add("    - "+pObjGr.getPO(i).getDate()
        +": Meiden(raeuuml.Persp.) zum Zeitpunkt "+timeText
        +" zwischen "+pObjGr.getPO(i).getNameNr()+" (" +pObjGr.getPO(i).getTeam()
        +") und "+pObjGr.getPO(j).getNameNr()+" (" +pObjGr.getPO(j).getTeam()+") bis ");
        System.out.print("    - Meiden (raeumliche Perspektive) zum Zeitpunkt "+
        timeText+" zwischen "+pObjGr.getPO(i).getNameNr()+" (" +pObjGr.getPO(i).getTeam()+
        ") und "+pObjGr.getPO(j).getNameNr()+" (" +pObjGr.getPO(j).getTeam()+") bis ");
    }
    //Reaktionsbewegungsmuster 4 = Konfrontation individuelle Perspektive
```

```

else if(patternNr == 4) {
    movementPatternArray.add("    - "+pObjGr.getPO(i).getDate()+
        ": Konfrontation(ind.Persp.) zum Zeitpunkt "+timeText
        +" zwischen "+pObjGr.getPO(i).getNameNr()+" ("+pObjGr.getPO(i).getTeam()
        +" ) und "+pObjGr.getPO(j).getNameNr()+" ("+pObjGr.getPO(j).getTeam()+") bis ");
    System.out.print("    - Konfrontation (individuelle Perspektive) zum Zeitpunkt "
        +timeText+" ("+pObjGr.getPO(i).getDate()+") zwischen "+pObjGr.getPO(i).getNameNr()
        +" ("+pObjGr.getPO(i).getTeam()+") und "+pObjGr.getPO(j).getNameNr()
        +" ("+pObjGr.getPO(j).getTeam()+") bis ");
}
//Reaktionsbewegungsmuster 5 = Konfronation raeumliche Perspektive
else if(patternNr == 5) {
    movementPatternArray.add("    - "+pObjGr.getPO(i).getDate()
        +": Konfrontation(raeuml.Persp.) zum Zeitpunkt "+timeText+" zwischen "
        +pObjGr.getPO(i).getNameNr()+" ("+pObjGr.getPO(i).getTeam()+") und "
        +pObjGr.getPO(j).getNameNr()+" ("+pObjGr.getPO(j).getTeam()+") bis ");
    System.out.print("    - Konfrontation (raeumliche Perspektive) zum Zeitpunkt "
        +timeText+" ("+pObjGr.getPO(i).getDate()+") zwischen "+pObjGr.getPO(i).getNameNr()
        +" ("+pObjGr.getPO(i).getTeam()+") und "+pObjGr.getPO(j).getNameNr()
        +" ("+pObjGr.getPO(j).getTeam()+") bis ");
}
}
else {
    //Bewegungsmuster 0 = Annaeherung
    if(patternNr == 0) {
        System.out.print("    - Approach zum Zeitpunkt "+timeText+" zwischen "
            +pObjGr.getPO(i).getNameNr()+" ("+pObjGr.getPO(i).getTeam()
            +" ) und "+pObjGr.getPO(j).getNameNr()+" ("+pObjGr.getPO(j).getTeam()+") bis ");
    }
    //Bewegungsmuster 1 = Coincidence
    else if(patternNr == 1) {
        System.out.print("    - Coincidence zum Zeitpunkt "+timeText+" zwischen "
            +pObjGr.getPO(i).getNameNr()+" ("+pObjGr.getPO(i).getTeam()+") und "
            +pObjGr.getPO(j).getNameNr()+" ("+pObjGr.getPO(j).getTeam()+") bis ");
    }
    //Bewegungsmuster 2 = Separation
    else if(patternNr == 2) {
        System.out.print("    - Separation zum Zeitpunkt "+timeText+" zwischen "+
            pObjGr.getPO(i).getNameNr()+" ("+pObjGr.getPO(i).getTeam()+") und "
            +pObjGr.getPO(j).getNameNr()+" ("+pObjGr.getPO(j).getTeam()+") bis ");
    }
    //Bewegungsmuster 3 = Getrenntsein
    else if(patternNr == 3) {
        System.out.print("    - Leaving zum Zeitpunkt "+timeText+" zwischen "

```


11.2 Persönliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und die den verwendeten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Wohlen, 28.04.2011

Michael Merki