

Skript zur Vorlesung
Managing and Mining Multiplayer Online Games
im Sommersemester 2015

Kapitel 9: Räumliche Verhaltensmodelle

Skript © 2012 Matthias Schubert

http://www.dbs.ifi.lmu.de/cms/VO_Managing_Massive_Multiplayer_Online_Games

Kapitelüberblick

- Spatial Data Mining in Games
- Visual Analytics und Heat Maps
- Spatial Prediction
- Spatial Outliers
- Trajektorien Darstellungen und Vergleiche
- Mustersuche in Trajektorien

Spatial Data Mining und Spiele

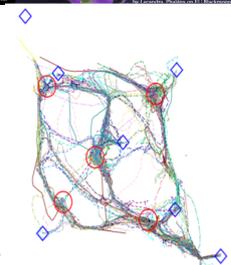
- Viele Spiele finden in einer virtuellen 2D-/3D- Welt statt.
- Bewegung und Positionierung ist häufig ein wichtiger Teil des Game Plays.
- Aufbau der Spielwelt ist relevant für das Balancing.
- Analyse von räumlichen und räumlich-zeitlichen Abläufen wird unter dem Begriff *Spatial Data Mining* zusammengefasst.



3

Aufgaben Spatial Game Analytics

- Finde Exploitation-Spots
 - Extraktion von Spielzügen und Bewegungsstrategien
 - Erkennen von Encountern (Open PVP)
 - Sub-Team Erkennung
 - Dynamisches Anpassen von Respawn-Raten
 - Erkennen von Bots und Multi-Boxing
 - Erkennen von Bewegungs- und Teleportations-Hacks
- ⇒ Suche bestimmte Orte
(Heatmaps, Spatial Prediction, Spatial Outlier)
- ⇒ Suche nach Bewegungsmustern (Trajectory Mining)



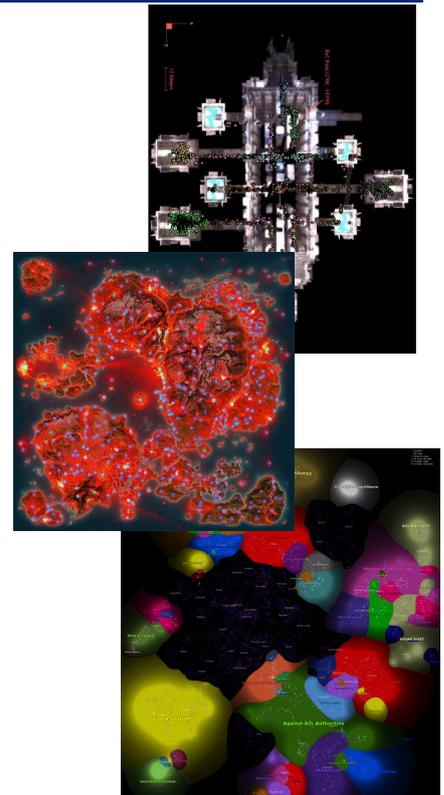
4

Räumliche Daten und Visualisierung

- Räumliche Daten bestehen aus Objektbeschreibung und Position.
(Beispiel: Marine, 43,56)
- Um besondere Orte zu finden, werden die Objektbeschreibungen bzgl. der Position aggregiert.
(z.B. Anzahl der Kills an einer Position, Spawn-Häufigkeit eines Monsters an einem Ort)
- Räumliche Kontinuität: i.d.R. geht man davon aus, dass sich benachbarte Positionen ähnlich verhalten.

⇒ Darstellung von aggregierten Informationen über 2D Histogramme (Bin Counting)

⇒ Darstellung der räumlichen Kontinuität über Glättungsansätze (Kerndichteschätzer)



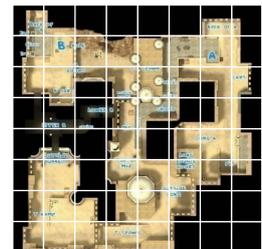
5

Heat Maps

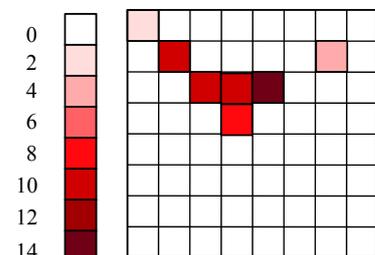
- Visualisierung der Verteilung der Ereignisse über die X-,Y- Koordinaten einer Karte.
- Darstellung der Verteilung als 2D-Dichteverteilung.
- Die Höhe der Bins wird durch Farbe kodiert.

Einfacher Algorithmus: Bin Counting

1. Lege uni-distantes Grid über die Karte
2. Für jedes Ereignis
 1. Bestimme die Gridzelle
 2. Erhöhe Zähler der Gridzelle um 1
3. Zeichne das Grid und färbe jede Zelle mit einer Farbe, die der Zahl in der Zelle entspricht.



3					
	10				5
		11	11	14	
			9		

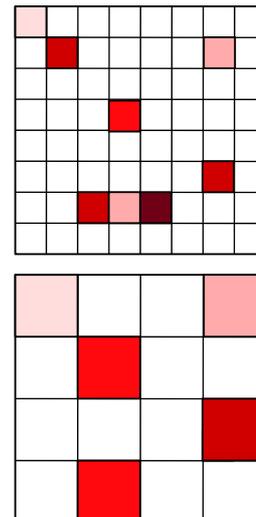


6

Heat Maps

Probleme bei Bin Counting:

- Einstellung der Grid-Größe:
 - zu klein: zerrissene Darstellung, wenige dichte Bereiche
 - zu groß: grobe Darstellung, wenig Unterscheidung
- Position des Grids beeinflusst Ergebnis
- räumliche Kontinuität kann schlecht erkennbar sein



Abhilfe: Glätten der Kurve mit Kerndichteschätzung

Abschätzung der Objektdichte über Summe von Kernfunktionen

- ⇒ Kontinuierliche und geglättete Dichtefunktion
- ⇒ Rasterung der Daten erst beim Zeichnen

7

Kerndichteschätzer

- Verfahren zur Abschätzung einer kontinuierlichen Dichtefunktion aus einer Samplmenge X .
- Betrachte Dichte $p(t)$ als Mixture-Model von $|X|$ Verteilungen, die alle mit der Kernfunktion $k(t)$ verteilt sind:

$$p(x) = \frac{1}{|X|} \sum_{t \in X} k(t - x)$$

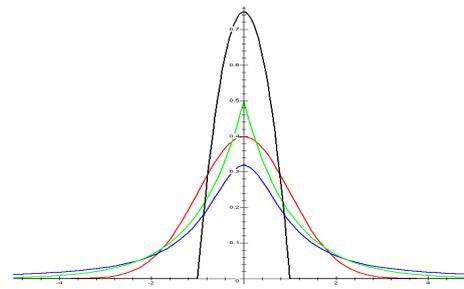
- Gängige Kernfunktion:

- Gaußkern : $k(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2}$

- Cauchy-Kern: $k(t) = \frac{1}{\pi(1+t^2)}$

- Picard-Kern : $k(t) = \frac{1}{2} e^{-|t|}$

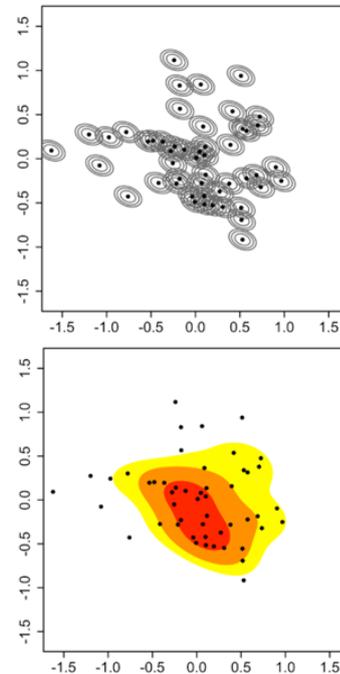
- Epanechnikow-Kern: $k(t) = \begin{cases} \frac{3}{4}(1-t^2), & \text{falls } t \in [-1;1] \\ 0 & \text{sonst} \end{cases}$



8

Heatmaps mit Kerndichteschätzern

- Kerne im 2D-Raum unter der Annahme unabhängiger Dimensionen:
$$p(t) = \left(\frac{1}{|X|} \sum_{x \in X} k(t_1 - x_1) \right) \cdot \left(\frac{1}{|X|} \sum_{x \in X} k(t_2 - x_2) \right)$$
- Jedes Bin entspricht einem Pixel
- Für jedes Pixel P wird $p(m)$ am Pixelmittelpunkt m berechnet
- Zur effizienten Berechnung:
 - Gehe über alle Punkte x :
 - Gehe über alle Pixel p :
 - Gehe über beide Dimensionen:
 - Erhöhe den Wert von p um $k(x-p_m)$ mit p_m Mittelpunkt von p



9

Spatial Data Mining

- Spezialbereich des Data Minings, der sich mit räumlichen Objekten beschäftigt.
- Objekt O besteht aus einer räumlichen Komponente $p \in IR^2/IR^3$ und einer Objektbeschreibung $v \in F$. (F ist ein beliebiger Feature-Raum)
- Spezielle Tasks im Spatial Data Mining:
 - **Spatial Outlier Detection:** Finde Orte, bei denen die Feature-Beschreibung deutlich von der Beschreibung der räumlich nahen Objekte abweicht.
(Beispiel: Exploitation Spots, bei denen man nicht getroffen werden kann.)
 - **Spatial Prediction:** Vorhersage von Orten, an denen bestimmte Phänomene häufig auftreten.
(Beispiel: Berechne die Wahrscheinlichkeit, ob an einer bestimmten Stelle, ein bestimmtes Verhalten beobachtet werden kann.)
 - **Spatial Clustering:** Clustering, das sowohl die räumliche Nähe als auch die Ähnlichkeit im Feature-Raum verwendet, um Cluster zu bilden bzw. voneinander abzugrenzen.
(Beispiel: Werden beliebige Aktivitäten häufig an bestimmten Stellen der Karte unternommen.)
 - **Spatial Rule Mining:** Ableiten von Assoziationsregeln auf Basis häufiger räumlicher Muster. (Beispiel: 80% der Städte, die innerhalb von 50 km der Siedlung eines anderen Spielers gebaut werden, überleben nicht bis zum Ende des Spiels.)

10

Spatial Prediction

- Supervised Learning auf räumlichen Daten.
- Spatial Auto Regression (SAR): Erweitern von Regressionsmodellen zur Berücksichtigung der Zielwerte naher Objekte (hier Matrixschreibweise):

$$y = \rho \cdot W \cdot y + X \cdot \beta + \varepsilon$$

- y : Vektor der Zielwerte
 - W : Matrix, die die räumliche Nähe der Objekte darstellt
 - X : Datenmatrix, die aus den Trainingsvektoren gebildet wird
 - ε : Normalverteilter Fehler/Rauschen
 - ρ : Gewichtungsfaktor für räumliche Komponente
 - β : Gewichtungsvektor für inhaltliche Komponente
- Umformung für die Berechnung: $(1 - \rho \cdot W) \bar{y} = X\beta + \varepsilon$

$$y = (1 - \rho \cdot W)^{-1} X\beta + (1 - \rho \cdot W)^{-1} \varepsilon$$

- $(1 - \rho \cdot W)^{-1}$ kann als räumliche Glättung des Feature-Raums aufgefasst werden.
- Bestimmen von ρ und β mit Maximum Likelihood Schätzern oder Markov-Chain-Monte-Carlo Abschätzung.

11

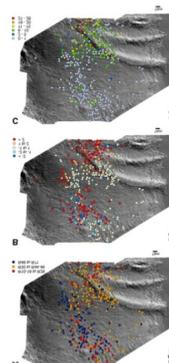
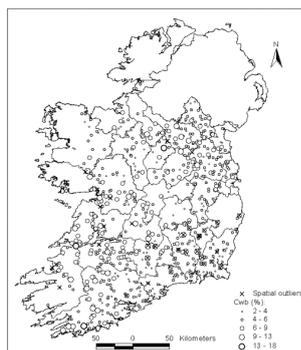
Spatial Outlier Detection

Gegeben: Eine Menge DB von räumlichen Objekte $O = (p, v)$.

Gesucht: Objekte, die in ihrer räumlichen Umgebung ungewöhnlich sind.

Allgemeines Vorgehen:

1. Bestimme für jedes Objekt O eine räumliche Nachbarschaft N .
(z.B. N besteht aus den räumlich k -nächsten Nachbarn von O).
2. Vergleiche die Feature-Beschreibungen $O.v$
mit der Verteilung der Feature-Beschreibungen in N .

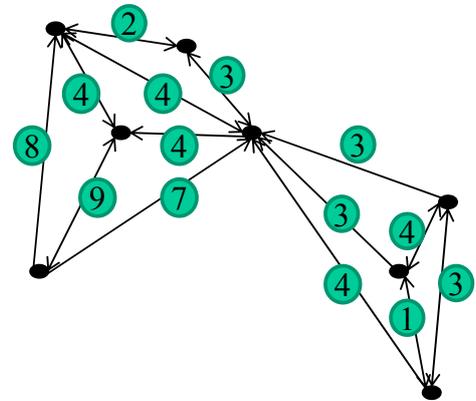


12

Spatial Outlier Detection

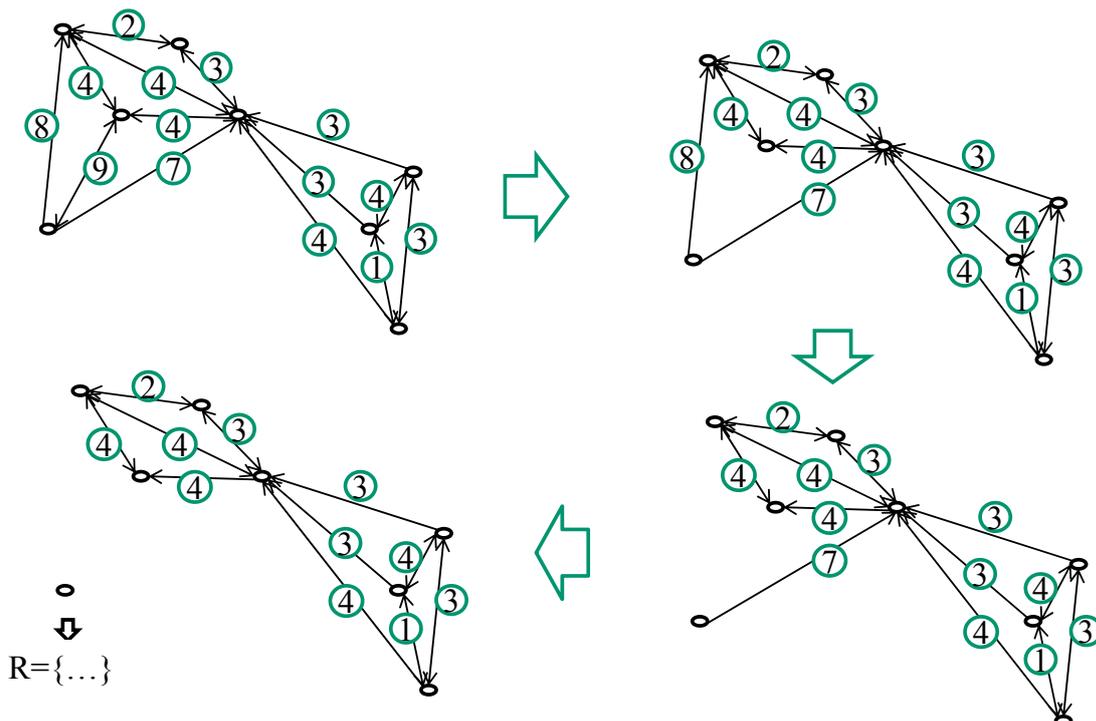
Point Outlier Detection (POD):

1. Aufstellen eines Nearest Neighbor Graphen $G(DB, E)$ auf den räumlichen Positionen.
 $E := \{(o_i, o_j) \mid o_i, o_j \in DB \wedge o_j \in NN_k(o_i)\}$
 Gewichtungsfunktion:
 $w(o_i, o_j) = ||o_i.v - o_j.v||$
2. Sortiere E absteigend nach $w(o_i, o_j)$
3. Solange $|R| < m$
 (noch keine m Outlier gefunden)
 1. Entferne die Kante (o_i, o_j) mit max. Gewicht $w(o_i, o_j)$
 2. Falls o_i jetzt isoliert ist
 Füge o_i in das Ergebnis R ein



13

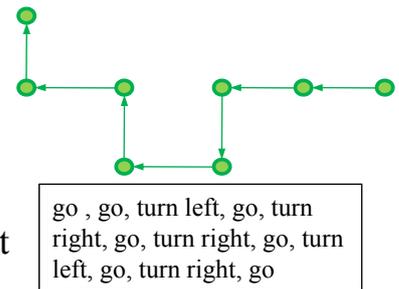
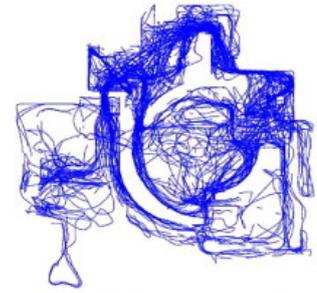
Beispiel POD



14

Trajektorien

- Trajektorien beschreiben eine Bewegung durch den Raum (Zeitreihen über räumliche Positionen)
- **Räumliche Trajektorie:** $Q=(x_1, \dots, x_l) \in \mathbb{R}^2 \times \dots \times \mathbb{R}^2$ heißt räumliche Trajektorie der Länge l über \mathbb{R}^2 .
- **Räumlich-Zeitliche Trajektorie:** Sei T eine Domäne zur Darstellung der Zeit, dann heißt $Q=((x_1, t_1), \dots, (x_l, t_l)) \in (\mathbb{R}^2 \times T) \times \dots \times (\mathbb{R}^2 \times T)$ räumlich-zeitliche Trajektorie der Länge l über \mathbb{R}^2 .
- Alternativ können Trajektorien auch relativ zu einer Startposition beschrieben werden.
- Bewegung ist kontinuierlich: Um einen kontinuierlichen Pfad zu erhalten, wird i.d.R. angenommen, dass die Bewegung zwischen 2 Positionen linear und mit konstanter Geschwindigkeit zurückgelegt wird.



Distanzmaße für Trajektorien

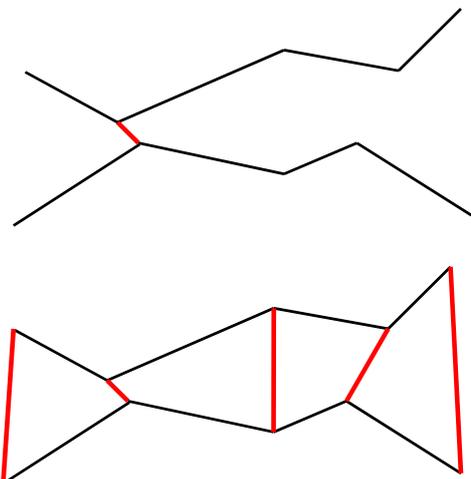
- **Punkt zu Trajektorie:** Gegeben $p \in \mathbb{R}^2$ und Trajektorie $Q=((x_1, t_1), \dots, (x_l, t_l))$: $D(p, Q) = \min_{(x, t) \in Q} d(p, x)$
- **Trajektorie zu Trajektorie:** Gegeben $Q=((x_1, t_1), \dots, (x_l, t_l))$ und $P= ((y_1, t'_1), \dots, (y_l, t'_l))$:

Closest Pair Distanz:

$$CPD(Q, P) = \min_{(x_i, t_i) \in Q, (y_j, t'_j) \in P} d(x_i, y_j)$$

Sum-of-Pairs:

$$SPD(Q, P) = \sum_{i=1}^n d(x_i, y_i)$$

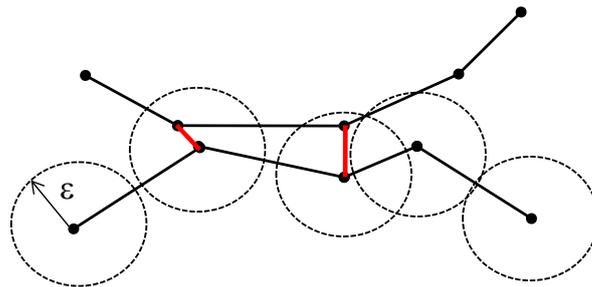


Abstandsmaße für Trajektorien

- bei unterschiedlicher Länge: DTW (Siehe Kapitel 8)
Aber: DTW ist anfällig für Ausreißer.
- Längste gemeinsame Subsequenz (Ähnlichkeitsmaß!)
LCSS (Longest Common SubSequenz):

$$LCSS(Q, P) = \begin{cases} 0, & \text{falls } n = 0 \vee m = 0 \\ 1 + LCSS(\text{Rest}(Q), \text{Rest}(P)), & \text{falls } d(\text{Head}(Q), \text{Head}(P)) \leq \varepsilon \wedge |n - m| < \delta \\ \max(LCSS(\text{Rest}(Q), P), LCSS(Q, \text{Rest}(P))), & \text{sonst} \end{cases}$$

- ε : Grenzwert für Positionsmatching, δ max. Verschiebung
- Berechnung durch Rekursion



17

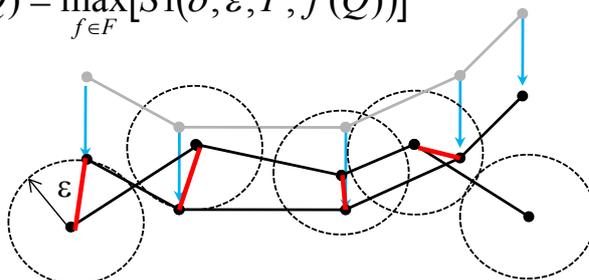
LCSS Ähnlichkeit

- $LCSS(P, Q)$ zählt bis jetzt nur die Länge der größten gemeinsamen Subsequenz, ist aber nicht normiert:

$$S1(\delta, \varepsilon, P, Q) = \frac{LCSS(P, Q)}{\min(|P|, |Q|)}$$

- Ähnlichkeit berücksichtigt noch nicht die Translation von Trajektorien (Translation: Verschiebung aller Positionen um einen festen Vektor):
Sei F die Menge aller Translationen und $f(Q) \in F$ eine Translation:

$$S2(\delta, \varepsilon, P, Q) = \max_{f \in F} [S1(\delta, \varepsilon, P, f(Q))]$$



18

Kompression von Trajektorien

Eigenschaften von Trajektorien in Spielen:

- hohe Auflösung (ca. 20-30 Punkte/s)
- keine Positionsmessfehler (Position ist exakt hinterlegt)
- Geschwindigkeit ist häufig fest abgestuft und Bewegung ist häufig linear.

Probleme: Auflösung ist häufig zu hoch und redundant

- Speicherbedarf ist extrem hoch
- Vergleiche werden sehr teuer (alle DTW basierten Maße sind quadratisch)

Lösungsansatz: Reduktion der Wegpunkte

⇒ Kompression durch Weglassen von Wegpunkten

⇒ Gute Verfahren minimieren Approximationsfehler

19

Douglas-Peucker Algorithmus

Gegeben: Eine Trajektorie $Q = ((x_1, t_1), \dots, (x_n, t_n))$ der Länge l .

Gesucht: Q' mit $|Q'| \ll l$ und Approximationsfehler ist kleiner als δ .

Algorithmus:

DP(Q, δ)

$Q' = ((x_1, t_1), (x_n, t_n))$

FOR ALL (x_i, t_i) in Q

IF $Error(x_i, Q') > \delta$ THEN

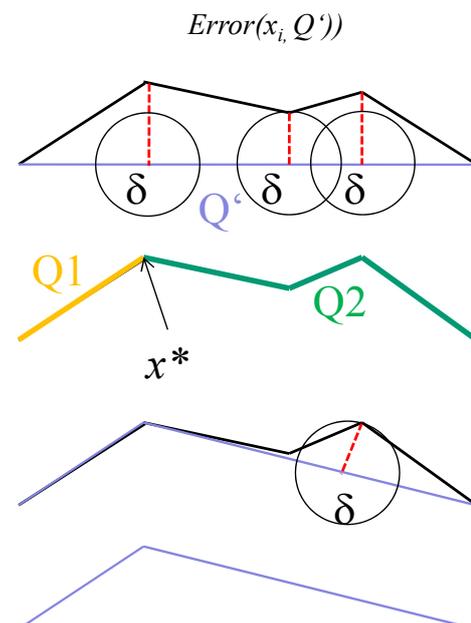
bestimme x^* mit $\max(Error(x_i, Q'))$

$(Q1, Q2) = split(Q, x^*)$

RETURN $DP(Q1, \delta) \circ DP(Q2, \delta)$

ENDFOR

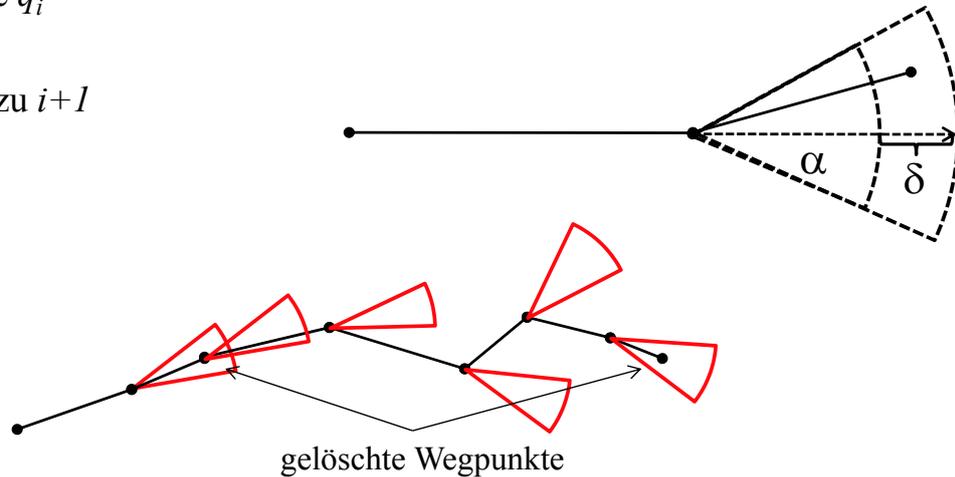
RETURN Q'



20

Kompression mit Geschwindigkeit und Richtung

- Betrachte letzte 2 Wegpunkte q_{i-2} , q_{i-1} und berechne Bewegungsrichtung $d_i = \frac{q_{i-2} - q_{i-1}}{\|q_{i-2} - q_{i-1}\|}$ und Geschwindigkeit $v_i = \frac{\|q_{i-2} - q_{i-1}\|}{t_{i-2} - t_{i-1}}$
- Extrapoliere den nächsten Wegpunkt $q_{i-1} + d_i v_i (t_{i+1} - t_i)$ und teste:
Wenn $|v_i(t_i - t_{i-1}) - (q_i - q_{i-1})| \leq \delta$ und $\frac{\langle d_i, q_i - q_{i-1} \rangle}{\|d_i\| \cdot \|q_i - q_{i-1}\|} \leq \alpha$
lösche q_i
sonst
gehe zu $i+1$



21

Mustersuche in Trajektorien

- Trajektorien können wie andere Objekte auch mittels distanzbasiertem Data Mining (z.B. OPTICs) und entsprechenden Distanzmaßen (LCSS) analysiert werden.
- Die resultierenden Muster bestehen aber aus global ähnlichen Trajektorien.
- Viele interessante Muster auf Trajektorien basieren aber nur auf einem verhältnismäßig kleinen Teil der Trajektorie.
- Interessante Muster haben häufig bestimmte räumliche Nebenbedingungen.

=> Spezielle Mustersuche für Trajektorien

22

Kontinuierliche Flocks

Idee: Finde Objekte die für eine gewisse Zeit einen gemeinsamen Weg hatten.

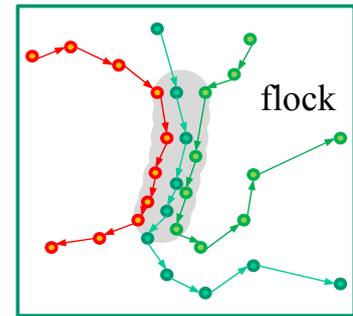
Beispiel: Subteams in Spielen, Convoys, Verbände, ...

Definition: *Kontinuierlicher (m,k,r) -Flock*

Sei DB eine Menge von Trajektorien der Länge l , ein Flock im Zeitintervall $I=[t_i, t_j]$ mit $j-i+1 \geq k$ besteht aus mindestens m Objekten, so dass es für jeden Zeitpunkt in I eine Scheibe mit Radius r gibt, die alle m Objekte umschließt.

Bemerkung: Berechnung des Flocks mit der längsten Dauer und Berechnung des Flocks mit dem größten Subset sind NP-harte Probleme.

=> Lösungen sind aufwändig oder nur approximativ



23

Flocks mit diskreter Zeit

Definition: *Diskreter (m,k,r) -Flock*

Sei DB eine Menge von Trajektorien der Länge l , ein Flock im $I=[t_i, t_j]$ mit $j-i+1 \geq k$ besteht aus mindestens m Objekten, so dass es für jeden diskreten Zeitpunkt t_l mit $i \leq l \leq j$ eine Scheibe mit Radius r existiert, die alle m Objekte umschließt.

- **Lemma:** Wenn sich die Objekte mit konstanter Geschwindigkeit und auf einer direkten Linie zwischen den Wegpunkten bewegen, sind diskrete und kontinuierliche Flocks äquivalent.
- **Vorteil:** Man kann aus dem kontinuierlichen Problem ein diskretes machen.
Aber: Die Komplexität bleibt und steckt in der Kombinatorik der möglichen Teilmengen. Mögliche Anzahl von Flocks mit m Elementen:

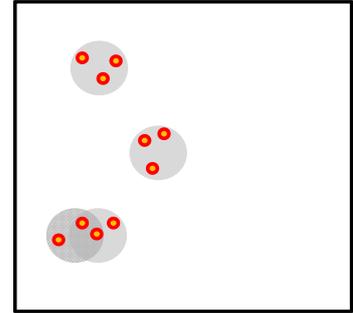
$$\binom{|DB|}{m} \cdot (l - k + 1)$$

24

Suche nach Flocks

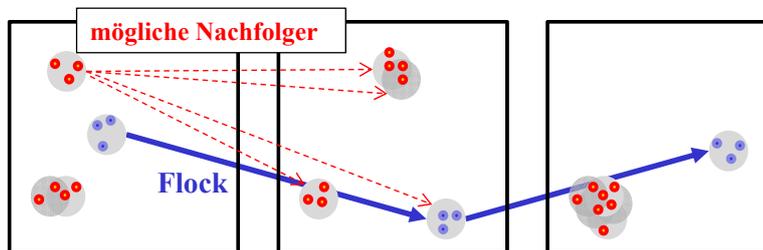
Vorgehen umfasst 2 Teilaufgaben:

1. Finde alle Scheiben mit Radius r , die mindestens m Punkte zum Zeitpunkt t_i enthalten.
 \Rightarrow Sequenz aus Teilmengen von DB
 \Rightarrow Eine Trajektorie kann auch in mehreren Teilmengen enthalten ein.



2. Finde Sequenz $(S(t_i), \dots, S(t_j))$ von Scheiben $S(t_i)$ für die Zeitpunkte t_i mit $i \leq l \leq j$ für die gilt:

$$\left| \bigcap_{i \leq l \leq j} S(t_l) \right| \geq m$$



25

Finden aller Scheiben zum Zeitpunkt t

Disks(t_i)

Generiere Grid-Index I über DB_i

FOR ALL non-empty cells $gx \in I$ **DO**

$Pr = gx$

$P_s = NeighborCells(gx)$

IF $|P_s| \geq m$ **THEN**

FOR EACH $pr \in Pr$ **DO**

$H = Range(pr, 2r)$

FOR each $pj \in H$ **DO**

IF not computed $\{pr, pj\}$ **THEN**

compute disks $\{c1, c2\}$ from $\{pr, pj\}$

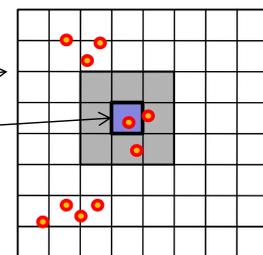
FOR EACH disk $ck \in \{c1, c2\}$ **DO**

$c = ck \cap H$

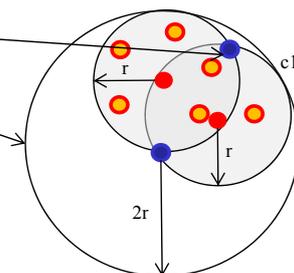
IF $|c| \geq m$ **THEN**

$C.add(c)$

RETURN C



c2



26

Finden von (m,k,r)-Flocks

Continuous Refinement Evaluation (CRE)

CRE(DB,k)

FOR EACH point in time t_i **DO**

L: Trajectories in time interval t_{i-k} to t_i

$C^1 = \text{Disks}(L[t_{i-k}])$ // alle Scheiben an denen Trajektorien aus L zu t_{i-k} beteiligt sind

$F = \{\}$ // Ergebnis

FOR EACH $c1 \in C^1$ **DO** // Für jede Startscheibe

$L'[1] = \text{trajectories in } c1$

$F^1 = c1, F^t = \{\}$

FOR $t = 2$ to k **DO** // Für die nächsten k-1 Zeitpunkte

$C^t = \text{Disks}(L'[t])$

$F^t = \{\}$

FOR EACH $c \in C^t$ **DO** // Für alle Scheiben zum Zeitpunkt t

FOR EACH $f \in F^{t-1}$ **DO** // Für alle bisher gültigen Flocks

IF $|c \cap f| \geq m$ **THEN**

$F^t = F^t \cup \{c \cap f\}$ // Erweitern des Flocks um einen Zeitpunkt

IF $|F^t| = 0$ **THEN**

BREAK

$F = F \cup F^t$

RETURN F

27

Meets (Encounter)

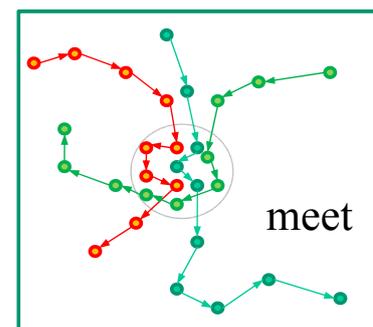
Idee: Finde Objekte, die sich für eine gewisse Zeit zusammen an einem Ort aufhalten.

Beispiel: Encounter, Kämpfe.

Definition: (m,k,r)-Meet

Sei DB eine Menge von Trajektorien der Länge l , ein Meet im Zeitintervall $I=[t_i, t_j]$ mit $j-i+1 \geq k$ besteht aus mindestens m Objekten, so dass für jeden Zeitpunkt in I alle m Objekte in einer Scheibe mit Radius r und Mittelpunkt M liegen.

Bemerkung: Die Berechnung von Meets ist einfacher als die Berechnung von Flocks, da bei 2 aufeinanderfolgenden Zeitpunkten nur die Positionen der Scheiben aber nicht deren Trajektorien untersucht werden müssen.



28

Lernziele

- Anwendungen für Spatial Game Analytics
- Heat Maps mit Bin Counting und Kerndichteschätzer
- Tasks im Spatial Data Mining
- Spatial Prediction mit Autoregression
- Spatial Outlier Detection mit POD
- Trajektorien, relative und absolute Trajektorien
- Vergleiche zwischen Trajektorien (LCSS)
- Kompression von Trajektorien (Douglas-Peucker, geschwindigkeits- und richtungsbasierte Komp.)
- Mustersuche in Trajektorien
 - Definition von Flocks
 - Berechnung von Flocks
 - Definition von Meets

29

Literatur

- Marcos R. Vieira, Petko Bakalov, and Vassilis J. Tsotras. 2009. **On-line discovery of flock patterns in spatio-temporal data**. In *Proc of the 17th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (GIS '09)*. ACM, New York, NY, USA, 286-295.
- Yu Zheng, Xiaofang Zhou: **Computing with Spatial Trajectories**, Springer, 2011.
- Marc Benkert, Joachim Gudmundsson, Florian Hübner, and Thomas Wolle. **Reporting flock patterns**. *Comput. Geom. Theory Appl.* 41, 3 (November 2008), 111-125.
- Anders Drachen, Alessandro Canossa : **Evaluating Motion: Spatial User Behavior in Virtual Environments** International Journal of Arts and Technology, 4(3): 1--21, 2011.
- H.K. Pao, K.T. Chen, H.C. Chang: **Game Bot Detection via Avatar Trajectory Analysis** Computational Intelligence and AI in Games, IEEE Transactions on, 2(3): 162--175, 2010.
- Jehn-Ruey Jiang, Ching-Chuan Huang, Chung-Hsien Tsai: **Avatar Path Clustering in Networked Virtual Environments** In Proceedings of the 2010 IEEE 16th International Conference on Parallel and Distributed Systems, 2010.
- C. Thureau, C. Bauckhage, G. Sagerer: **Learning human-like movement behavior for computer games**, In From animals to animats 8: Proceedings of the 8th International Conference on Simulation of Adaptive Behavior, 2004.
- Yufeng Kou, Chang-Tien Lu, Raimundo F. Dos Santos: **Spatial Outlier Detection: A Graph-Based Approach**, 19th IEEE International Conference on Tools with Artificial Intelligence , pp. 281-288, Vol.1 (ICTAI 2007), 2007.
- Shekhar, Shashi and Schrater, Paul and Vatsavai, Ranga Raju and Wu, Wei Li and Chawla, Sanjay. **Spatial Contextual Classification and Prediction Models for Mining Geospatial Data**. *IEEE Transactions on Multimedia*. 4(2):174-188, 2002.

30