

Lecture Notes for
Managing and Mining Multiplayer Online Games
Summer semester 2017

Chapter 9: Spatial behavior models

Lecture Notes © 2012 Matthias Schubert

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

Chapter Overview

- spatial data mining in games
- visual analytics and heat maps
- spatial outliers
- trajectories: representation and similarity
- pattern search on trajectory data

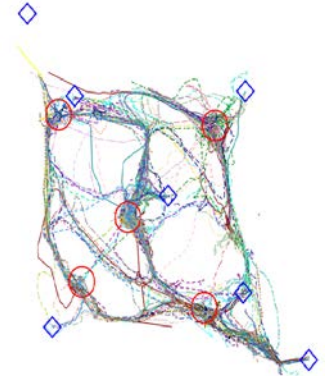
Spatial Data Mining and Games

- many games take place in a virtual 2D-/3D-World
- movement and position is often an important part of game play
- game world design is relevant for balancing
- analysis of spatial and spatial-temporal information is referred to as *Spatial Data Mining*



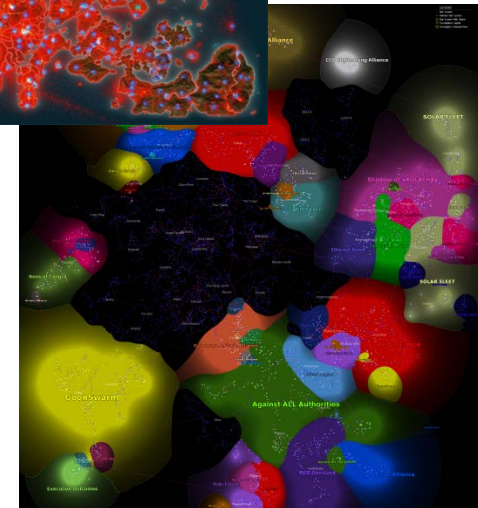
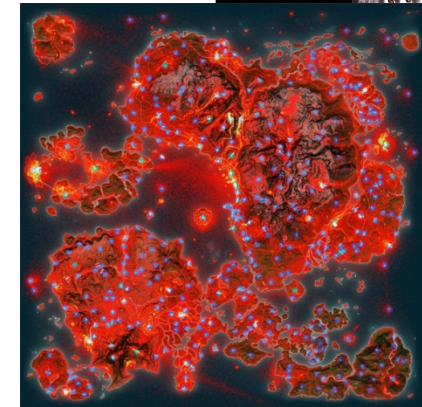
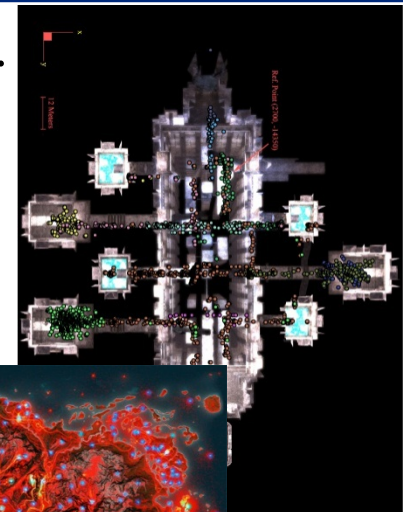
Tasks of Spatial Game Analytics

- find exploitation spots
 - extract game moves and movement strategies
 - encounter detection (open PVP)
 - sub team recognition
 - dynamic adjustment of respawn times
 - detect bot and multiboxers
 - detect movement and teleportation hacks
- ⇒ find specific places
(heat-maps, spatial outliers)
- ⇒ find movement patterns (trajectory mining)



Spatial Data and Visualization

- spatial data consists of object descriptions and positions.
(Example: Marine, 43,56)
 - to find special places, object descriptions are aggregated w.r.t. positions
(e.g. number of kills at a position, monster's spawn frequency at a place)
 - spatial continuity: usually one assumes adjacent positions to behave in a similar fashion.
- ⇒ presentation of aggregated information in 2D histograms (bin counting)
- ⇒ presentation of spatial continuity with smoothing approach (kernel density estimation)



Heat Maps

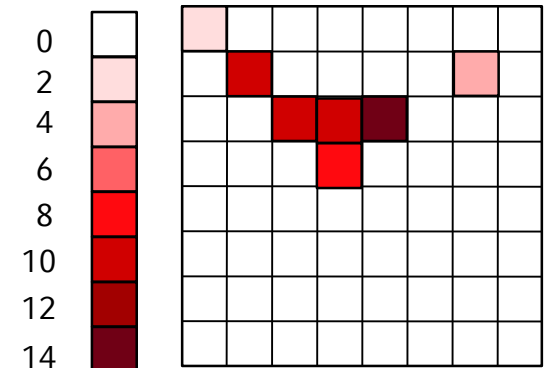
- visualizing the distribution of events on X-,Y-coordinates of a map.
- displaying the distribution as a 2D-Density distribution.
- a bin's height is encoded with it's color.

simple algorithm: Bin Counting

1. place uni-distance Grid overlay on the map
2. for every event
 1. determine grid cell
 2. increase grid cell counter by 1
3. draw the grid and color each cell matching the number within.



3							
	10					5	
		11	11	14			
			9				

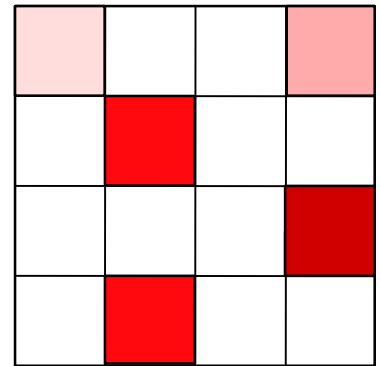
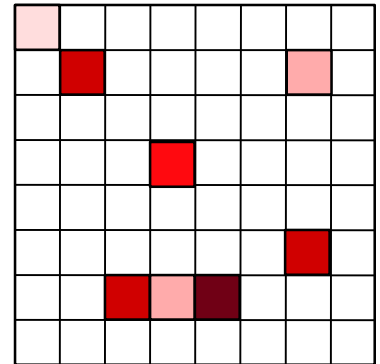


Heat Maps

Problems with bin counting:

- setting grid-size:
 - too small: torn view, few dense areas
 - too big: rough view, few differences
- grid position influences result
- spatial continuity may be hardly discernible

Remedy: smooth curves with kernel density estimation
estimate density with the sum of kernel functions
⇒ continuous and smoothed density function
⇒ discretization of data only for drawing



Kernel density estimator

- method to estimate a continuous density function from a sample set X .
- consider density $p(t)$ as mixture model of $|X|$ distributions, all of them distributed with kernel function $K(t)$:

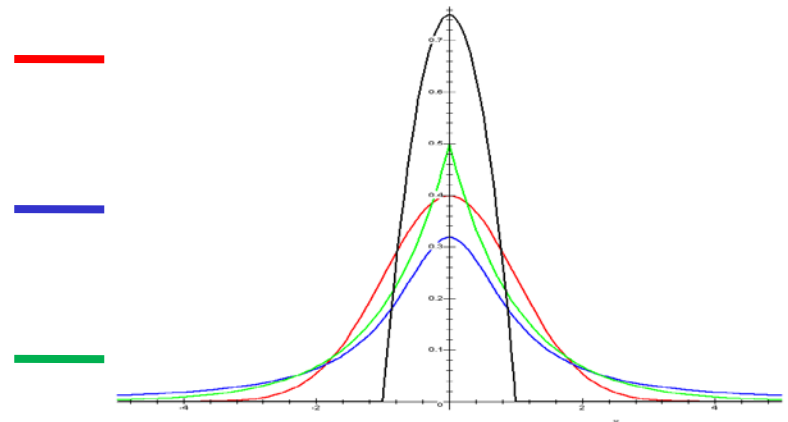
- common kernel functions: $p(x) = \frac{1}{|X|} \sum_{t \in X} K(t - x)$

- Gauss-kernel : $K(t) = \frac{1}{\sqrt{2\pi}} e^{\left(-\frac{1}{2}t^2\right)}$

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

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

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



Heatmaps with kernel density estimators

- kernels in 2D-Space assuming independent dimensions:
- every bin corresponds to one pixel
- for every pixel P , $p(m)$ is calculated based on pixel center m
- for efficient calculation:

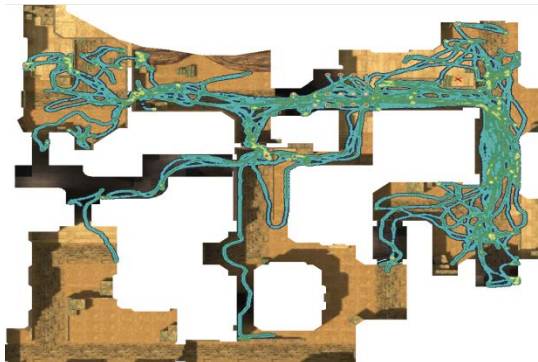
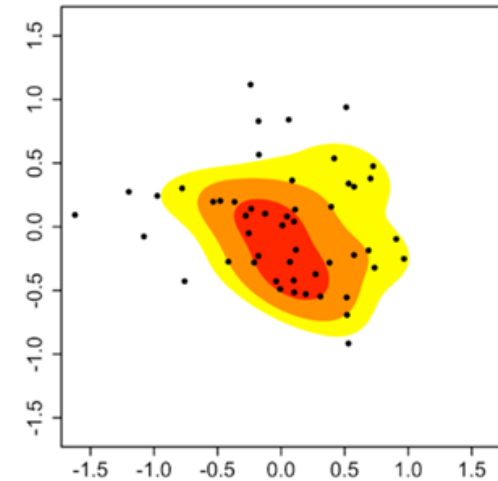
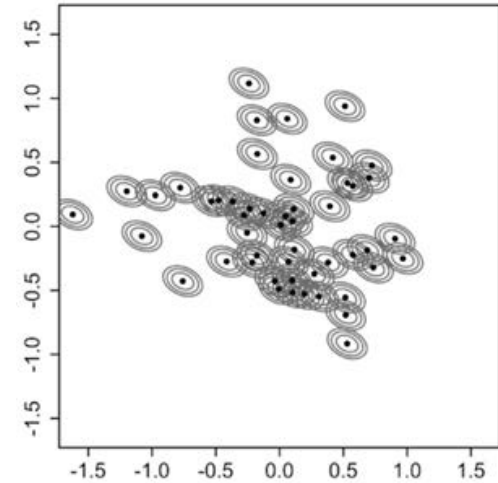
for all points x :

for all pixel p :

for both dimensions:

increase the value of p
by $K(x-p_m)$ with p_m center of p

$$p(t) = \left(\frac{1}{|X|} \sum_{x \in X} K(t_1 - x_1) \right) \cdot \left(\frac{1}{|X|} \sum_{x \in X} K(t_2 - x_2) \right)$$



Spatial Data Mining

- particular data mining methods for spatial objects.
- object O consists of a spatial component $p \in \mathbb{R}^2/\mathbb{R}^3$ and an object description $v \in F$. (F is an arbitrary feature space)
- special tasks in spatial data mining:
 - **Spatial Outlier Detection:** find places where the feature descriptions significantly varies from the object description of close objects.
(Example: exploitation spots where you can not be hit.)
 - **Spatial Prediction:** prediction of areas where certain phenomena are more frequent. (Example: calculate the probability of a certain behavior occurring at a certain spot.)
 - **Spatial Clustering:** Clustering using proximity as well as similarities of the feature space to create or differentiate clusters.
(Example: Are any actions frequently taken at certain areas of the map?)
 - **Spatial Rule Mining:** Derivation of association rules based on frequent spatial patterns. (Example: 80% of cities built within 50 km of another players settlement do not survive until the end of the game.)

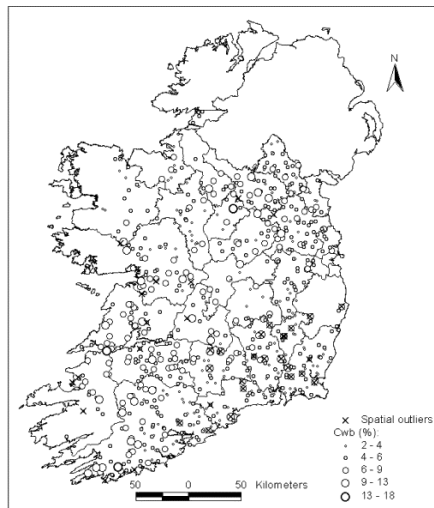
Spatial Outlier Detection

Given: A set DB of spatial objects $O = (p, v)$.

Searched: Objects that are unusual for their neighborhood.

General procedure:

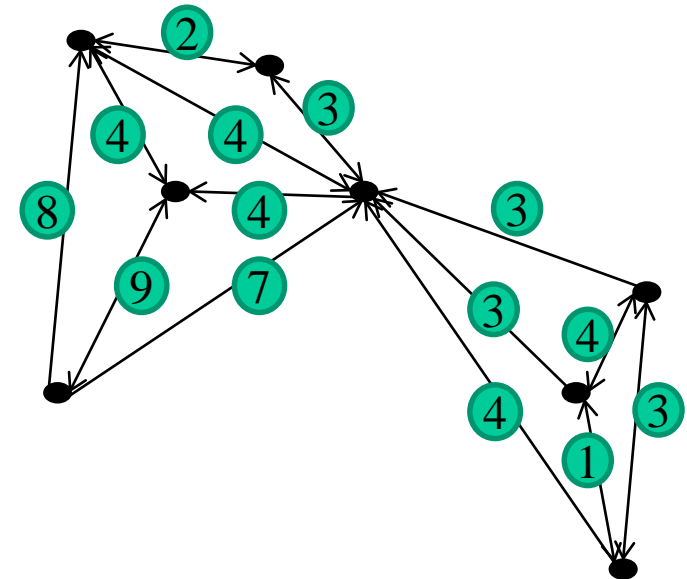
1. Determine neighborhood N for every object O .
(e.g N consists of k closest neighbors of O).
2. Compare the feature description of $O.v$ with the distribution of feature descriptions in N .



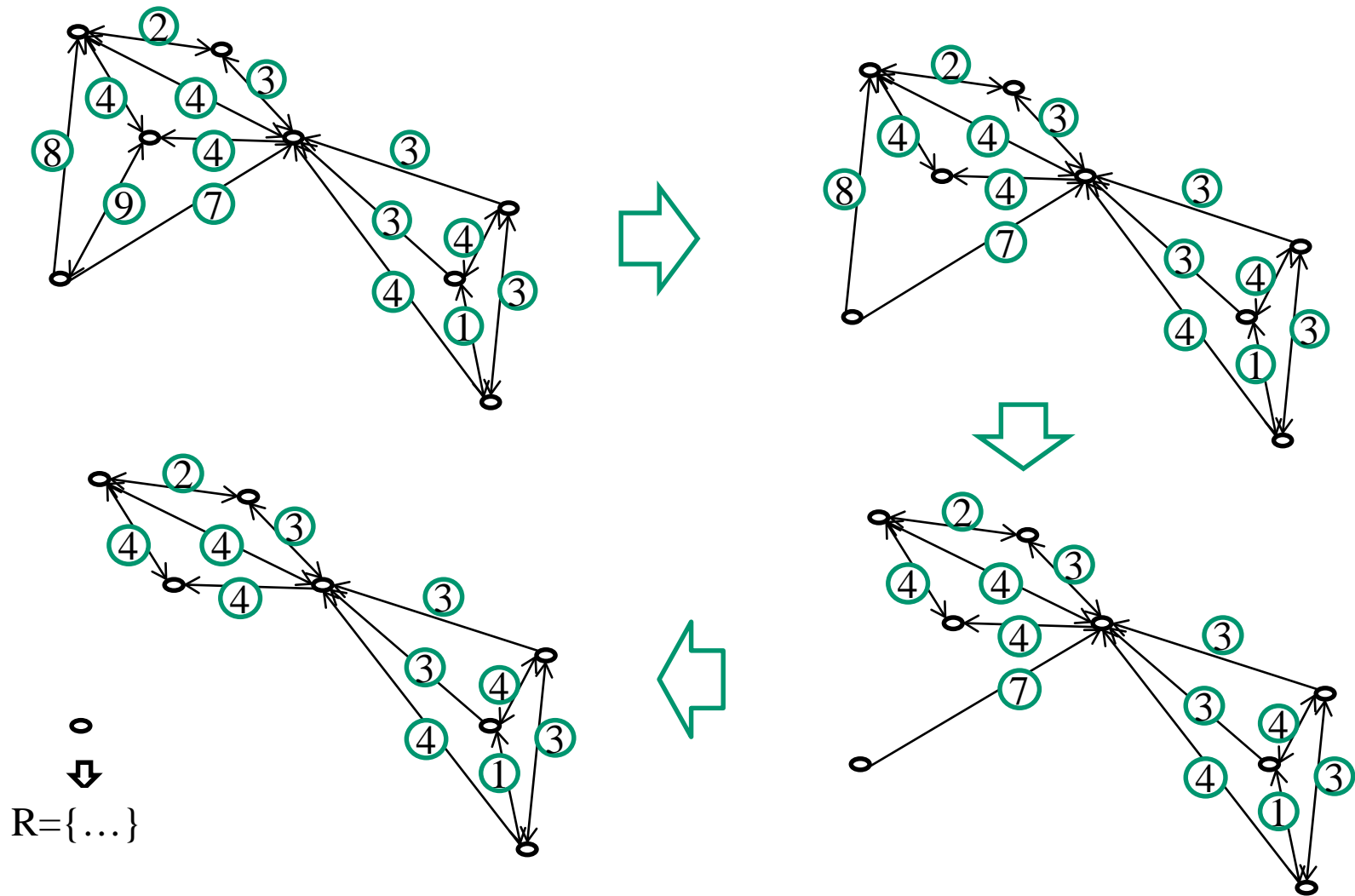
Spatial Outlier Detection

Point Outlier Detection (POD):

1. set up a nearest neighbor graph $G(DB, E)$ for spatial positions.
 $E := \{(o_i, o_j) \mid o_i, o_j \in DB \wedge o_j \in NN_k(o_i)\}$
weighting function:
 $w(o_i, o_j) = ||o_i.v - o_j.v||$
2. sort E by $w(o_i, o_j)$ in descending order
3. while $|R| < m$
(m outliers not found yet)
 1. remove the edge (o_i, o_j) with max. weight $w(o_i, o_j)$
 2. if o_i is isolated, insert o_i into the result R

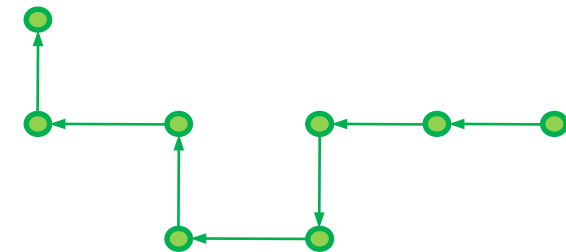


Example POD



Trajectories

- trajectories describe a movement through space (time series of spatial positions)
- spatial trajectory:** $Q=(x_1, \dots, x_l) \in \mathbb{R}^2 \times \dots \times \mathbb{R}^2$ is known as spatial trajectory of length l over \mathbb{R}^2 .
- spatial-temporal trajectory:** Let T be a domain to present time, then $Q=((x_1, t_1), \dots, (x_l, t_l)) \in (\mathbb{R}^2 \times T) \times \dots \times (\mathbb{R}^2 \times T)$ is a spatial-temporal trajectory of length l over \mathbb{R}^2 .
- alternatively trajectories can be described relatively to a starting position.
- movement is continuous: to get a continuous path, the movement between two positions is assumed to be linear and to be traversed with constant speed.



go , go, turn left, go, turn
right, go, turn right, go,
turn left, go, turn right,
go

Distance Measure for Trajectories

- **point to trajectory:** Given $p \in \mathbb{R}^2$ and trajectory $Q = ((x_1, t_1), \dots, (x_l, t_l))$:

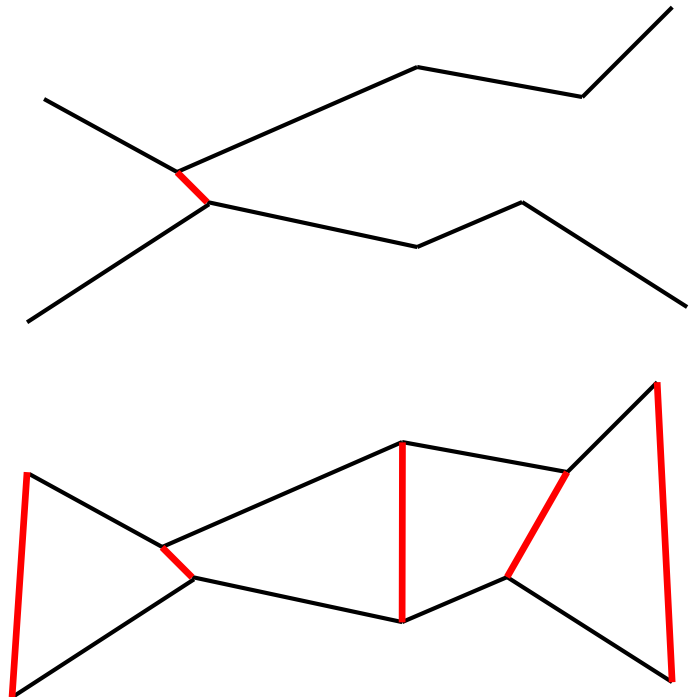
$$D(p, Q) = \min_{(x, t) \in Q} d(p, x)$$
- **trajectory to trajectory:** Given $Q = ((x_1, t_1), \dots, (x_l, t_l))$ and $P = ((y_1, t'_1), \dots, (y_l, t'_l))$:

Closest Pair Distance:

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

Sum-of-Pairs:

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

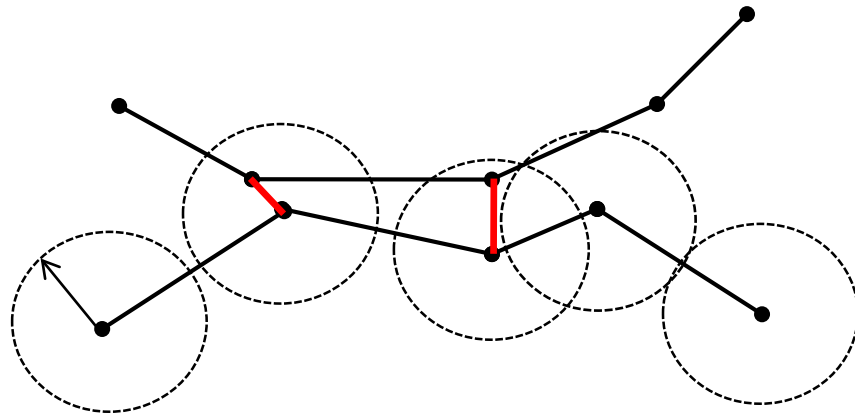


Distance Measures for Trajectories

- for different lengths: DTW (See Chapter 8)
but: DTW is susceptible to outliers.
- longest common sub-sequence (similarity measure!)
LCSS (Longest Common Sub-Sequence):

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

- ε : threshold for position matching, δ max. shift
- calculation by recursion



LCSS Similarity

- LCSS(P,Q) only counts the length of the longest common subsequence up to now, but is not normalized yet:

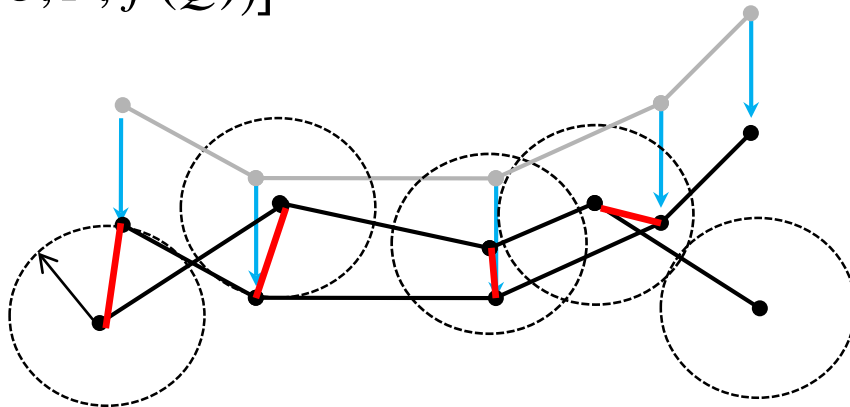
$$S1(\delta, \varepsilon, P, Q) = \frac{LCSS(P, Q)}{\min(|P|, |Q|)}$$

- similarity does not yet take the translation of trajectories into account

(translation: Shifting all positions by a fixed vector):

Let F be the set of all translations and $f(Q)$ F one translation:

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



Compressing trajectories

characteristics of trajectories in games:

- high resolution (ca. 20-30 points/s)
- no measuring errors for positions
- velocity gradation is usually steady and movement is often linear.

problems: resolution is often too high and redundant

- extremely high memory requirement
- comparisons become very expensive
(e.g., all DTW based measures are square)

spproach: reduce waypoints

⇒ compression by omitting waypoints

⇒ good methods minimize approximation errors

Douglas-Peucker Algorithm

Given: A trajectory $Q=((x_1, t_1), \dots, (x_l, t_l))$ of l length.

Searched: Q' with $|Q'| \ll l$ and approximation error smaller than δ .

Algorithm:

$DP(Q, \delta)$

$Q' = ((x_1, t_1), (x_l, t_l))$

FOR ALL (x_i, t_i) in Q

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

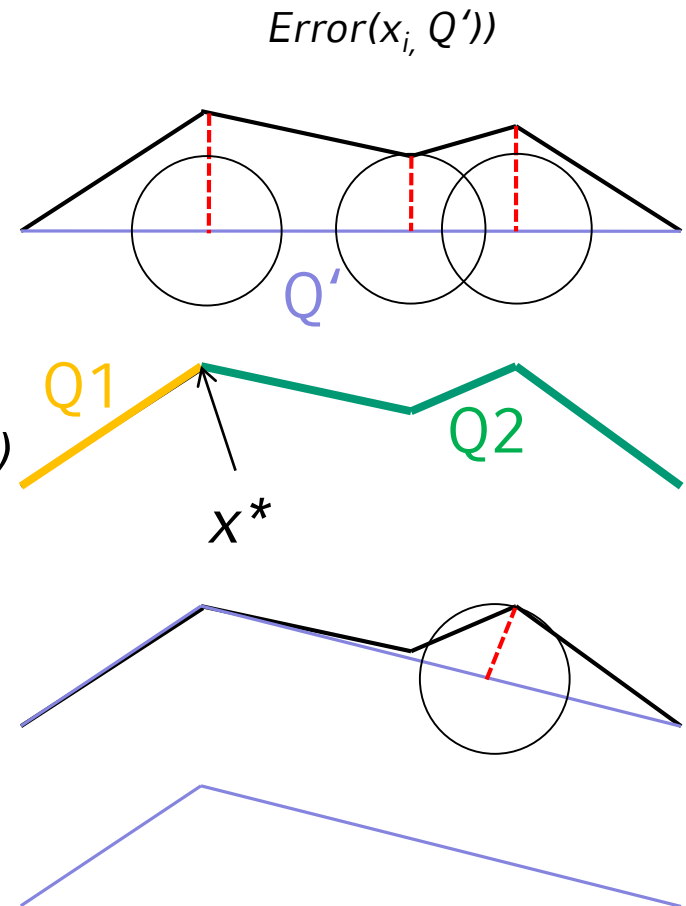
determine x^* with $\max(Error(x_i, Q'))$

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

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

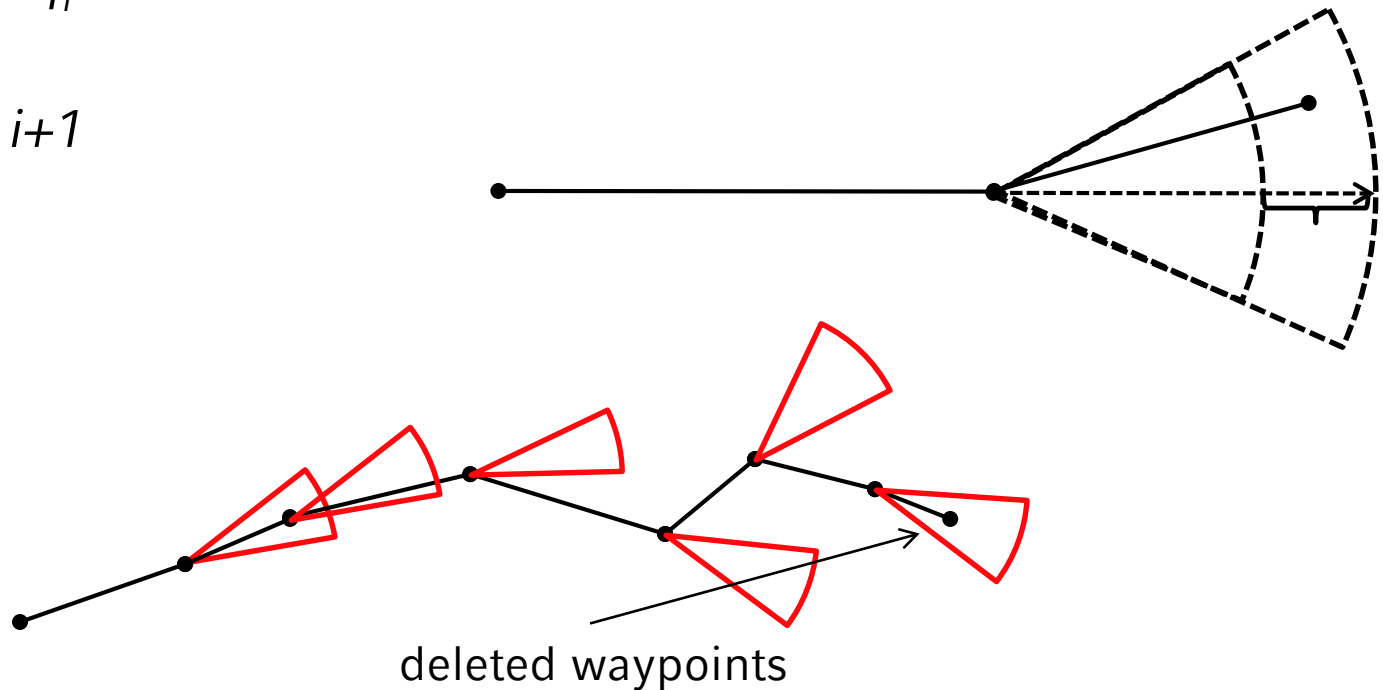
ENDFOR

RETURN Q'



Compressing with Speed and Direction

- Consider last 2 waypoints q_{i-2}, q_{i-1} and calculate movement direction $d_i = \frac{q_{i-2} - q_{i-1}}{\|q_{i-2} - q_{i-1}\|}$ and speed $v_i = \frac{\|q_{i-2} - q_{i-1}\|}{t_{i-2} - t_{i-1}}$
- extrapolate next waypoint $q_{i-1} + d_i v_i (t_{i+1} - t_i)$ and test:
 If $|v_i(t_i - t_{i-1}) - (q_i - q_{i-1})|$ and $\frac{\langle d_i, q_i - q_{i-1} \rangle}{\|d_i\| \cdot \|q_i - q_{i-1}\|} \leq \alpha$
 delete q_i
 else
 go to $i+1$



Pattern Search in Trajectories

- like other objects, trajectories can be analyzed with distance based data mining (z.B. OPTICs) and corresponding distance measures (LCSS).
- but resulting patterns consist of globally similar trajectories.
- many interesting trajectory patterns rest on a relative small part of the trajectory.
- interesting patterns usually have spatial constraints.

=> special pattern search methods for trajectories

Continuous Flocks

Idea: Find objects that share a path for a certain time interval.

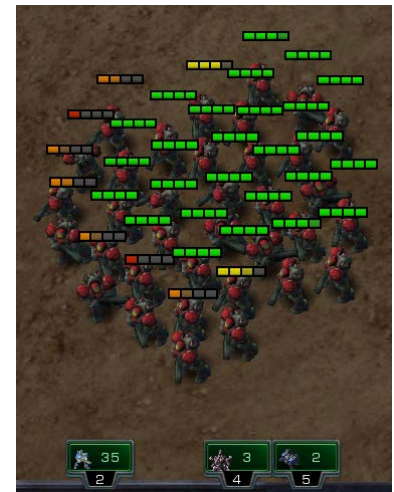
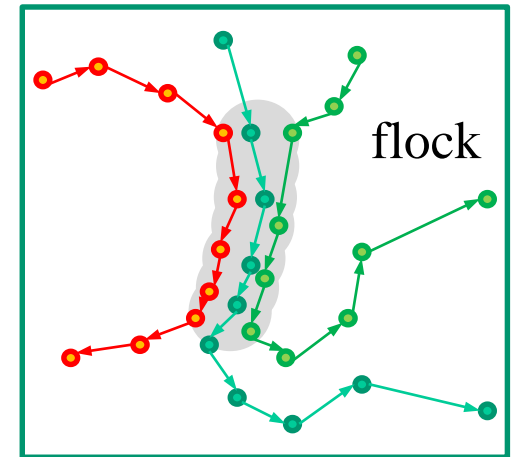
Example: subteams in games, convoys,...

Definition: Continuous (m,k,r) -Flock

Let DB be a set of trajectories of length l , a Flock within the time interval $I=[t_i, t_j]$ where $j-i+1 \geq k$ consists of at least m objects, so that a disc with radius r , enclosing all m objects, exists in I .

Remark: Calculating the flock with the longest duration and the flock with the largest subset are NP-hard problems.

=> solutions are complex or only approximate



Flocks with discreet Time

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

Let DB be a set of trajectories of l length, a Flock in $l=[t_i, t_j]$ with $j-i+1 \geq k$ consists of at least m objects, so that a disc with radius r , enclosing all m objects, exists for each discrete time t_l where $i \leq l \leq j$.

- **Lemma:** If objects move with constant speed and on a direct line between waypoints, discrete and continuous flocks are equivalent.
- **Advantage:** Turning a continuous problem to a discrete one.
But: Complexity remains unchanged and comes from the combination of possible subsets.

The possible number of flocks with m elements is: $\binom{|DB|}{m} \cdot (l - k + 1)$

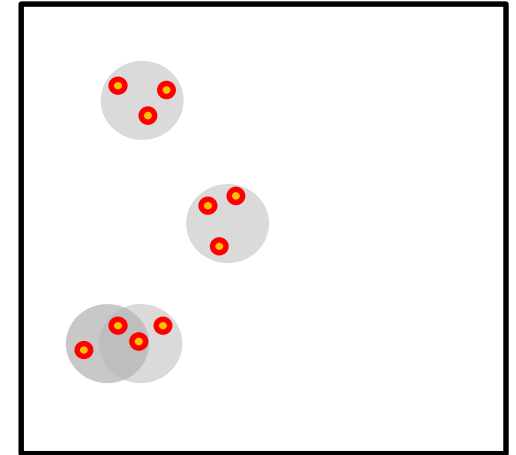
Searching for Flocks

Procedure encompasses 2 subtasks:

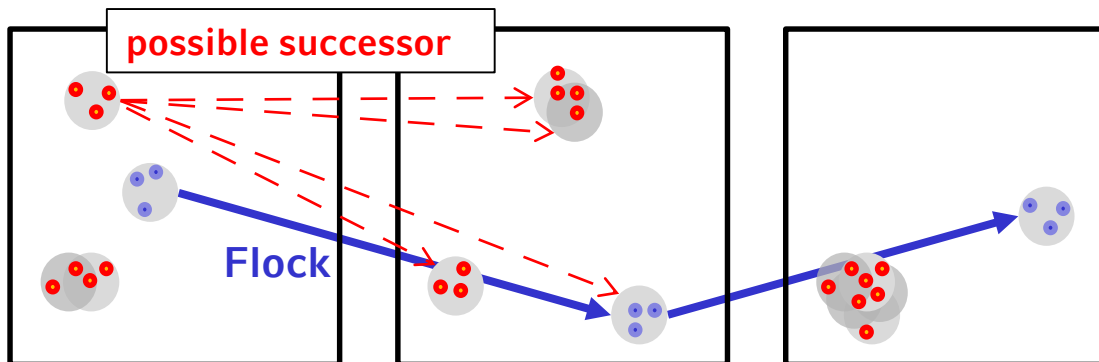
1. Find all discs of radius r , containing at least m points for time t_j .

=> sequence of subsets of DB

=> one trajectory may be present in several subsets.



2. Find sequence $(S(t_i), \dots, S(t_j))$ of discs $S(t_l)$ for the points in time t_l with $i \leq l \leq j$ for which the following condition holds: $\left| \bigcap_{i \leq l \leq j} S(t_l) \right| \geq m$



Find all Discs for the Point in Time t

Discs(t_i)

build grid index I for DB_i

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

$Pr = gx$

$Ps = \text{NeighborCells}(gx)$

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

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

$H = \text{Range}(pr, 2r)$

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

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

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

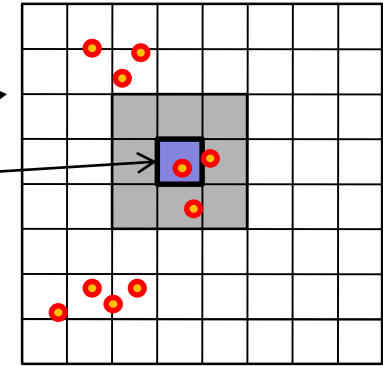
FOR EACH disk $ck \in \{c1, c2\}$ **DO**

$c = ck \cap H$

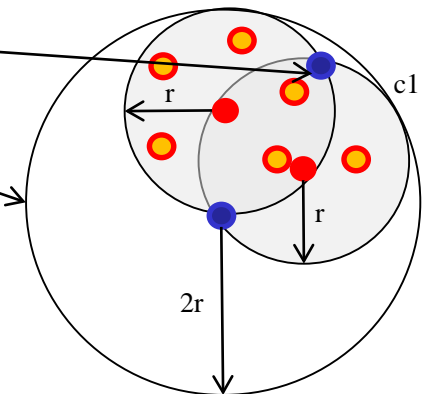
IF $|c| \geq m$ **THEN**

$C.add(c)$

RETURN C



$c2$



Finding (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

$C^1 = \text{Disks}(L[t_{i-k}])$ // all containing trajectories in L at t_{i-k}

$F = \{\}$ // results flocks

FOR EACH $c1 \in C^1$ **DO** // for each start disc

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

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

FOR $t = 2$ **to** k **DO** // for the next k-1 times

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

$F^t = \{\}$

FOR EACH $c \in C^t$ **DO** // for all disc at time t

FOR EACH $f \in F^{t-1}$ **DO** // for currently valid flocks

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

$F^t = F^t \cup \{c \cap f\}$ // extend the flock by one point in time

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

BREAK

$F = F \cup F^t$

RETURN F

Meets (Encounter)

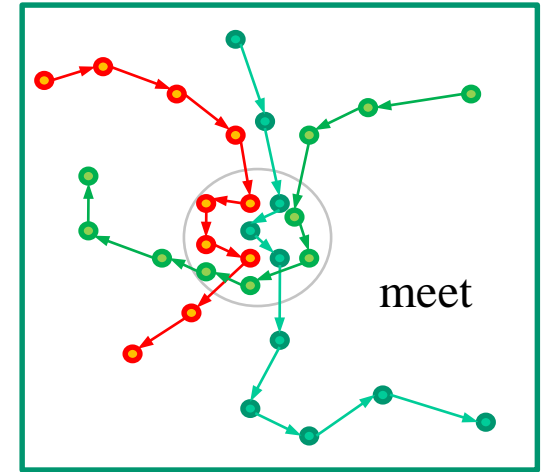
Idea: Find objects that stay together in an area for a certain time.

Examples: Encounter, Combat.

Definition: (m,k,r) -Meet

Let DB be a set of trajectories of length l , a meet within the time interval $I=[t_i, t_j]$ with $j-i+1 \geq k$ consists of at least m objects, so that for every point in time $t_i \in I$ all m objects lie within a disc of radius r and center point M .

Remarks: Calculating meets is easier than calculating flocks because for two consecutive points in time only the discs positions, not their trajectories, must be analyzed.



Encounter Detection

Idea: To find out where a team succeeded /failed and find the decisive moments in a game.

- in Dota2 defeating enemy heroes grants the biggest advantage in gold/XP
- find situations where this was possible or succeeded
=> Encounters

Encounter characteristics

- encounters represent only a portion of the game
- encounters can happen simultaneously
- often only sub teams are involved in encounters

Defining Encounters

Idea: Fights happen when opponents can influence each other.

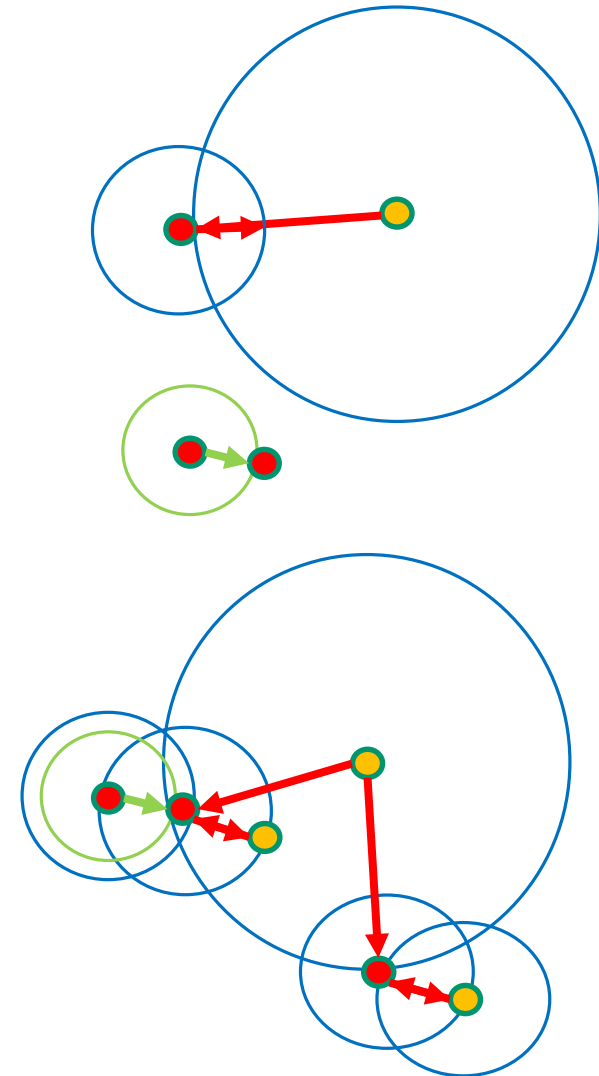
- opponents have to be in fighting range
- each hero unit might have an individual attack range
- heroes can support (e.g. heal) a friendly unit

Which kind of information is necessary?

- Spatial position and unit type for each controlled hero unit
- Attack and support ranges for all units types

Encounter Situations

- **Combat link:** 2 hero units from different teams A and B. Either A can attack B or vice versa
- **Support link:** 2 hero units from the same team A and B. Either A can support B or vice versa
- Each hero type has individual **attack and support ranges** (Ranges are mean values plus to standard deviations)
- **Component Graph:** Connected Graph build by Combat/support Links



Encounter Situations

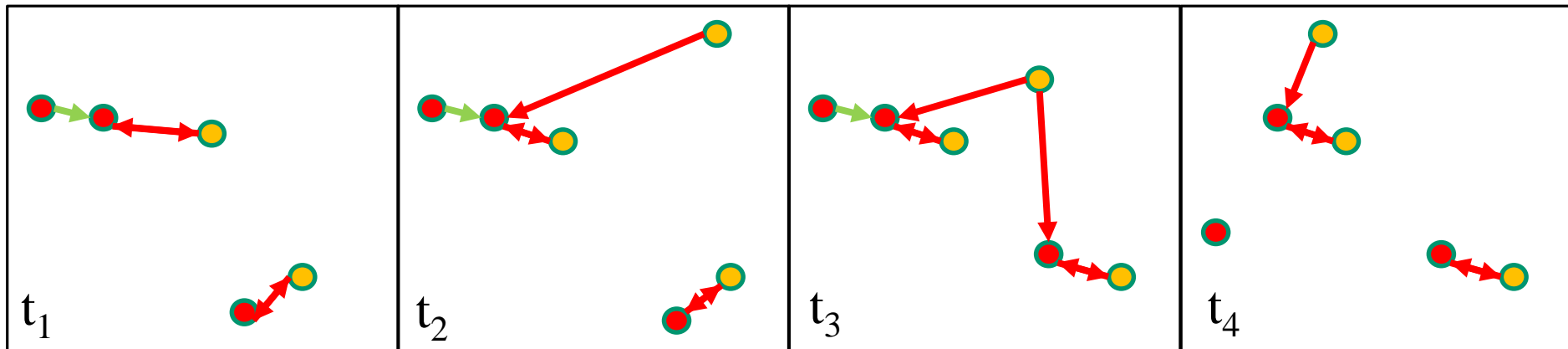
Formally...

Definition: Combat Component

- units U and the union $E_d = CL \cup SL$ of combat links CL and support links SL between the units in U .
- $E_u = \{(u_i, u_j) \mid (u_i, u_j) \in E_d \vee (u_j, u_i) \in E_d\}$
- situation graph $G(U, E_u)$.
- combat component C : connected subgraph $G(\bar{U}, \bar{E})$ of $G(U, E_u)$ where $\bar{U} \subseteq U, \bar{E} \subseteq \bar{U} \times \bar{U}$
and $\forall u_1, u_l \in \bar{U}: \exists (u_1, u_2, \dots, u_l)$
where $i \in \{1, \dots, l\}: (u_i, u_{i+1}) \in \bar{E}$
and $\exists u_i, u_j \in \bar{U}: u_1.team \neq u_2.team$.

Defining Encounters

- Component Graphs describe an Encounter at tick t
- An encounter usually lasts multiple consecutive ticks
- Hero Units can join encounters
- Hero Units might be defeated or leave
- Encounters can split
- Encounters can join



Defining Encounters

Formally...

Definition: Successor

Given a set of components $CS_t = \{ C_{1,t}, \dots, C_{l,t} \}$ describing encounter E at tick t . Let τ be a timeout threshold. A component $C_{t+\Delta t}$ is a successor of CS_t denoted as $CS_t \rightarrow C_{t+\Delta t}$ if the following conditions hold:

- $\Delta t \leq \tau$
- $\exists u_1, u_2 \in C_{t+\Delta t} : \exists C_{i,t} \in CS_t : u_1 \in C_{i,t} \wedge C_{j,t} \in CS_t : u_2 \in C_{j,t} \wedge u_1.team \neq u_2.team$

Defining Encounters

Formally....

Definition: Encounter

An encounter is a sequence (CS_0, \dots, CS_l) of lists of components CS_i where the following condition holds:
 $\forall C_{i,t} \in CS_t: CS_{t-1} \rightarrow C_{i,t}$ with $t \in \{1, \dots, l\}$.

Encounter Detection

What is the input data ?

- hero type (combat range, support range), team
- time series of position updates (one at a time)

Algorithm:

- initialize hero information
- stream over position updates and update distances
- for each player movement process the impact to the current component graphs
- keep lists of open encounters
- move encounters to a closed set if they time out

The Algorithm

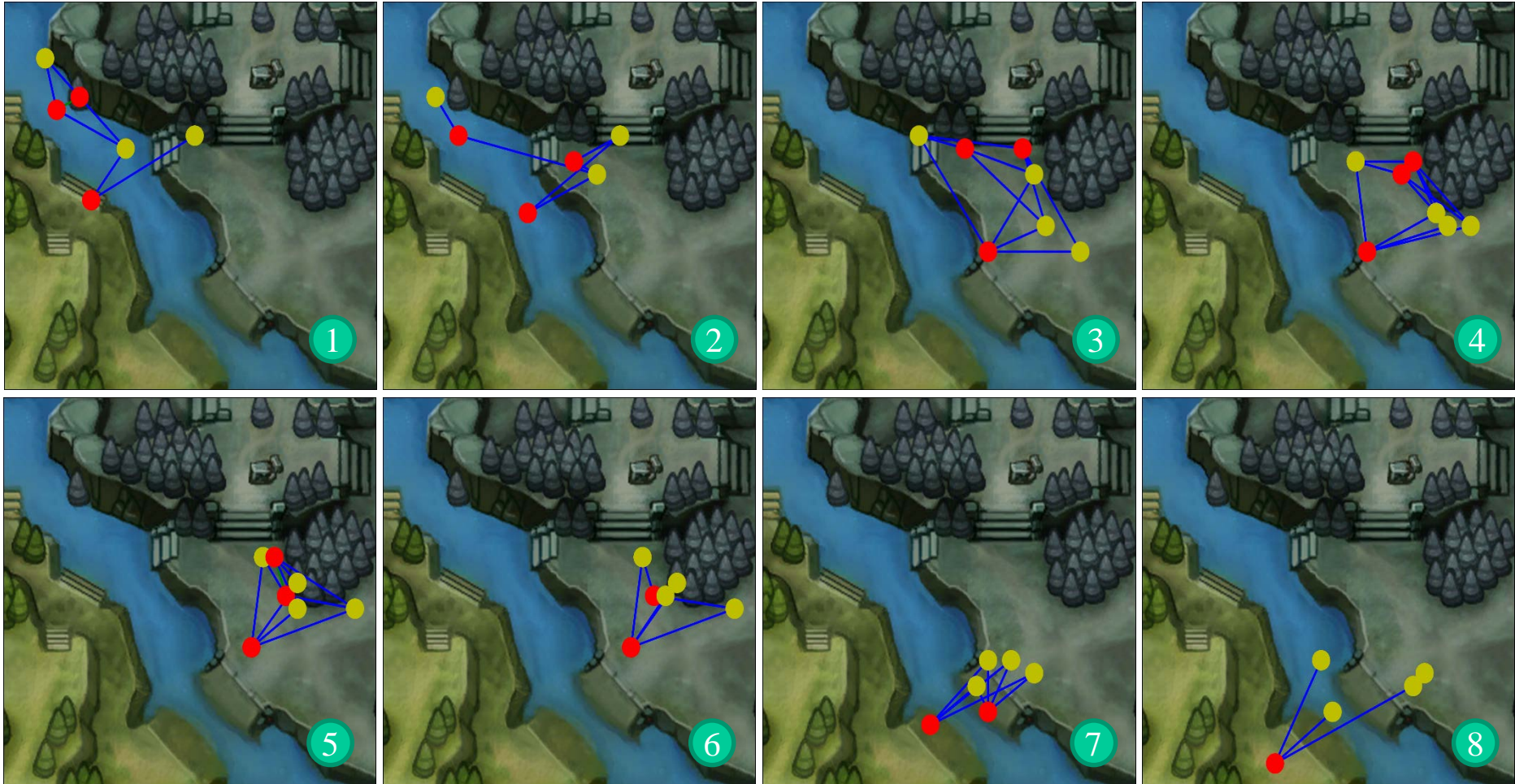
Encounter Detection (position_stream)

```
While position_stream.hasNext():  
    component = build_component(unit,distance_table)  
    If component is combat component:  
        compute predecessors(component, open_encounters)  
        If predecessors.size() == 0:  
            open_encounters.add(new Encounter(component))  
        If predecessors.size() == 1:  
            predecessors.get(1).update(component)  
        If predecessors.size() >1:  
            open_encounters.join(predecessors,component)  
        For encounter in open_encouters:  
            If encounter has timeout:  
                move encounter from open_encounter to closed_encounters  
For encounter in open_encouters:  
    move encounter from open_encounter to closed_encounters  
return closed_encounters
```

An Example Encounter



An Example Encounter (Detailed View)



Learning Goals

- use cases for spatial game analytics
- heat maps with bin counting and kernel density estimation
- tasks of spatial data mining
- spatial outlier detection with POD
- trajectories, relative and absolute trajectories
- comparing trajectories (LCSS)
- compressing trajectories
- pattern search in trajectories
 - definition of flocks
 - calculation of flocks
 - definition of meets
 - encounter detection

Literature

- 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.