

Consiglio Nazionale
delle Ricerche

Mobility Patterns

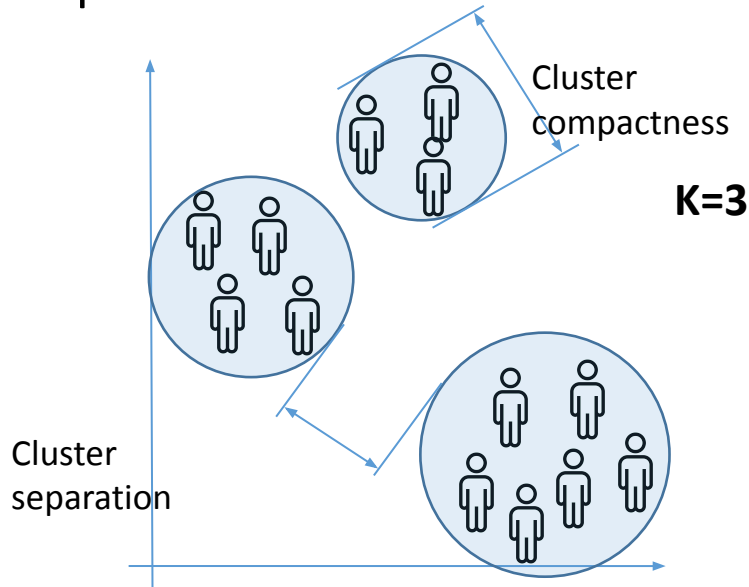
Content of this lesson

- Global patterns
 - Trajectory distances
 - Trajectory clustering
- Local patterns
 - Flocks, Convoys & Swarms
 - Moving clusters
 - T-Patterns

Global Patterns

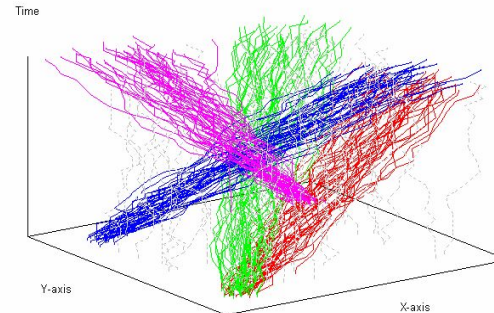
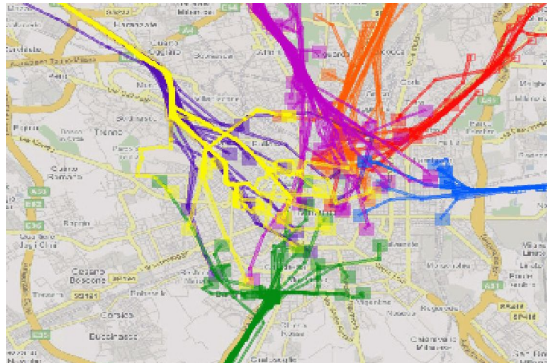
Clustering (sample K-means family)

- Find k subgroups that form compact and well-separated clusters



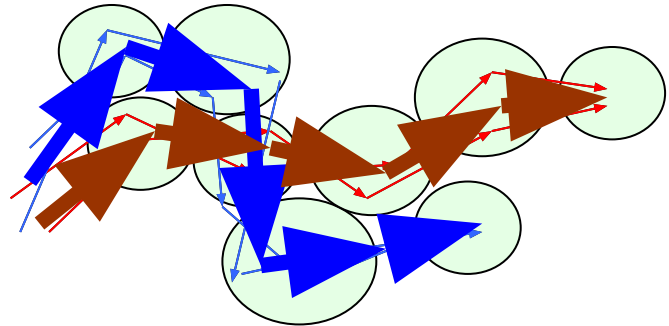
Trajectory clustering

- Trajectories are grouped based on similarity



Trajectory Clustering

- Questions:
 - Which distance between trajectories?
 - Which kind of clustering?
 - What is a cluster 'mean' in our case?
 - A representative trajectory?



Trajectory Distances

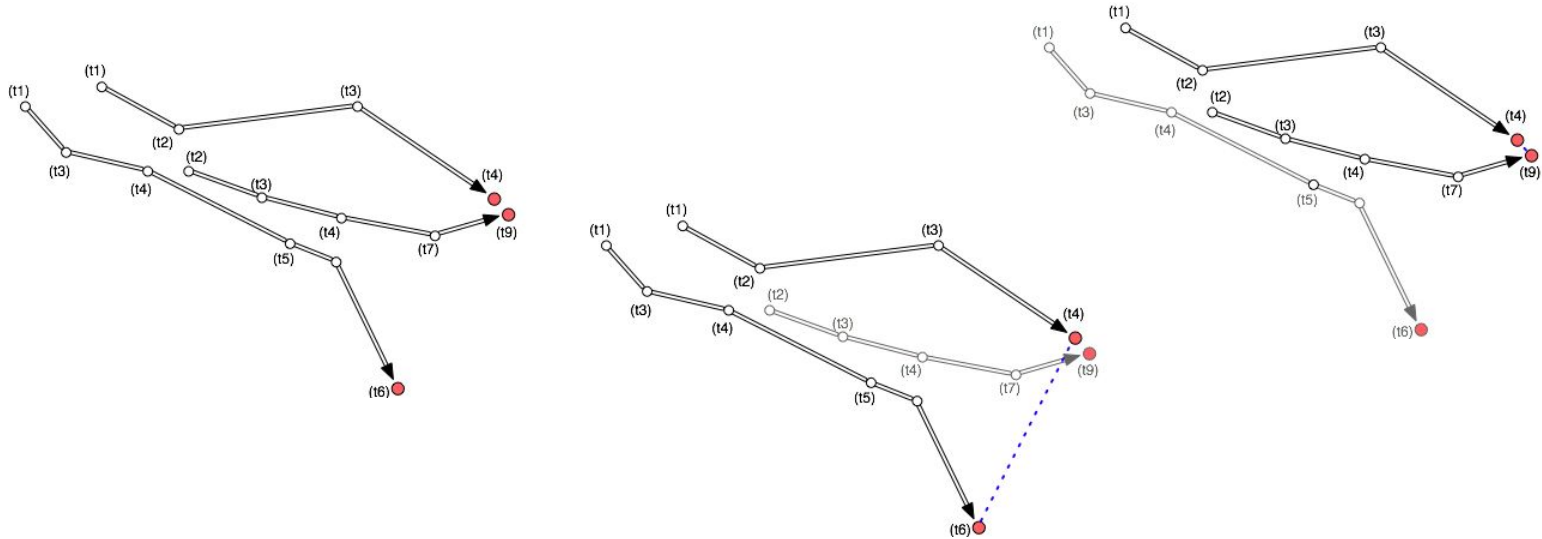
Families of Trajectory Distances

- Trajectory as **set** of points
 - Single-point approaches
 - Hausdorff distance
- Trajectory as **sequence** of points
 - Fréchet distance
 - Time series distances: Euclidean, DTW & LCSS
- Trajectory as **time-stamped sequence** of points
 - Average Euclidean distance

Trajectory as set of points

Common Destination

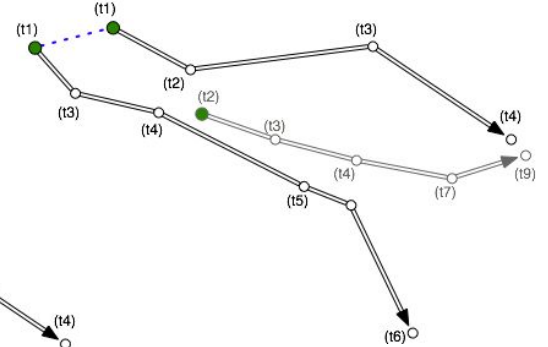
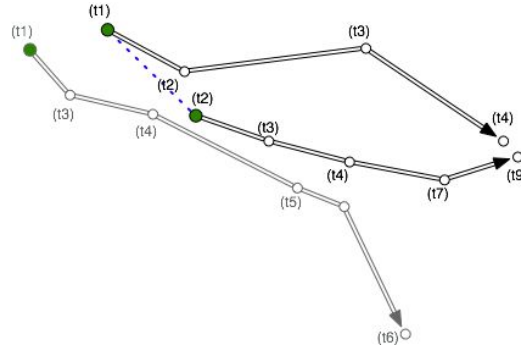
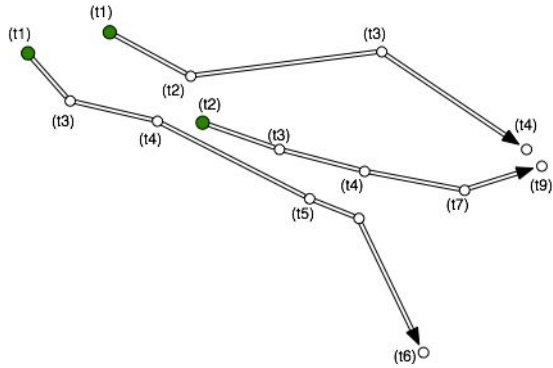
- ❑ Select last point P_{last} for each trajectory
- ❑ $D(T, T') = \text{Euclidean}(P_{last}, P'_{last})$



Trajectory as set of points

Common Origin

- ❑ Select first point *Pfirst* for each trajectory
- ❑ $D(T, T') = \text{Euclidean}(P_{\text{first}}, P'_{\text{first}})$



Trajectory as set of points

Hausdorff distance

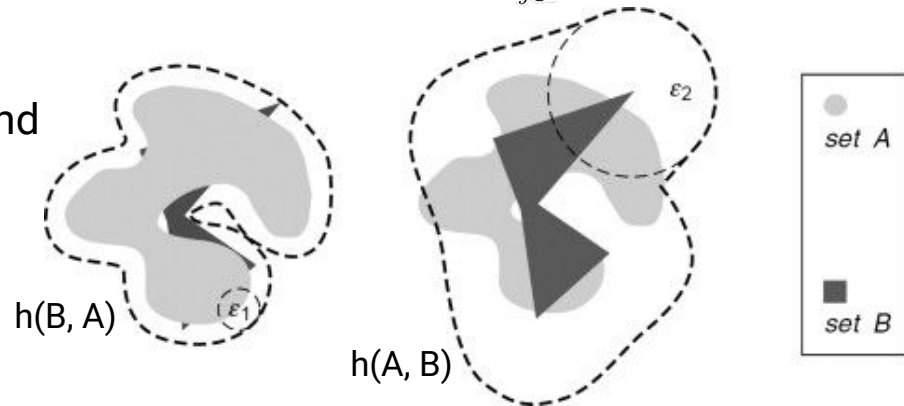
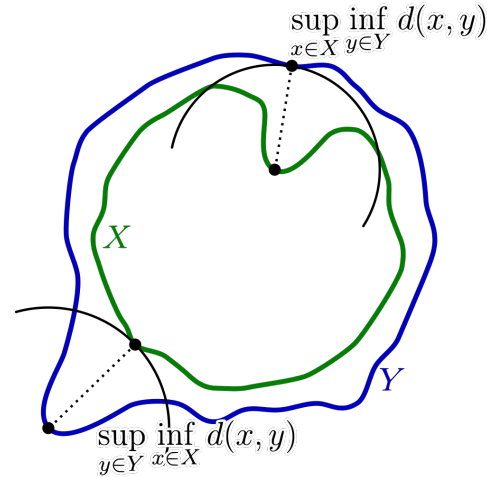
- Intuition: two sets are close if every point of either set is close to some point of the other set

- Formally, given sets A and B: $d_H(X, Y) = \max \left\{ \sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y) \right\}$.

- $r(x, B) = \inf \{d(x, b) : b \in B\}$
- $h(A, B) = \sup \{r(a, B) : a \in A\}$
- $d_H(A, B) = \max \{h(A, B), h(B, A)\}$

- Equivalently:

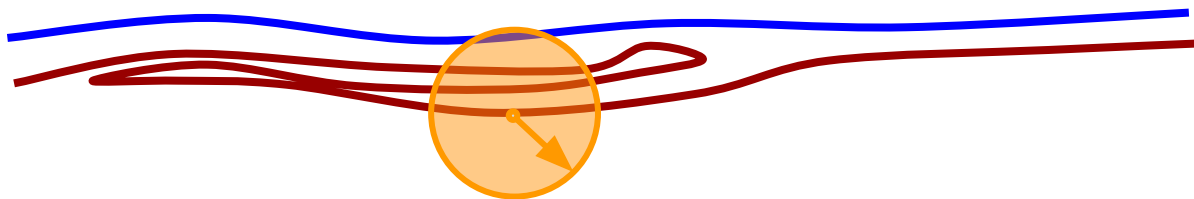
- $h(A, B) =$ minimum buffer radius around B that fully contains A
- $d_H(A, B) =$ symmetric version of $h()$



Trajectory as sequence of points

From Hausdorff to Fréchet distance

- Applied to trajectories, sometimes Hausdorff distance yields counter-intuitive results
- How far are these?



- Reasonable in a set-oriented view
- Wrong in terms of moving objects

Trajectory as sequence of points

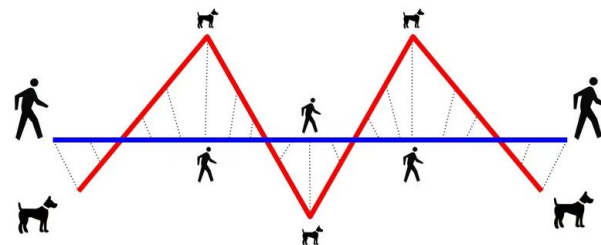
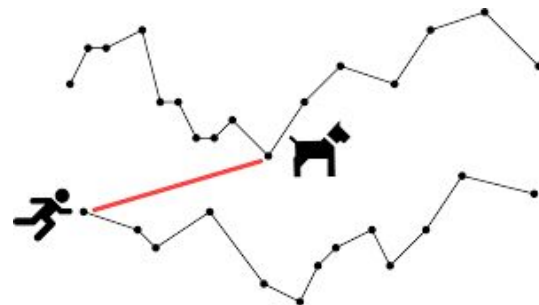
Fréchet distance

- Intuition: equivalent of Dynamic Time Warping on continuous curves
- Formally:

$$F(A, B) = \inf_{\alpha, \beta} \max_{t \in [0,1]} \left\{ d\left(A(\alpha(t)), B(\beta(t))\right) \right\}$$

α and β are non-decreasing mappings from $[0,1]$ to the points along A and B in forward order

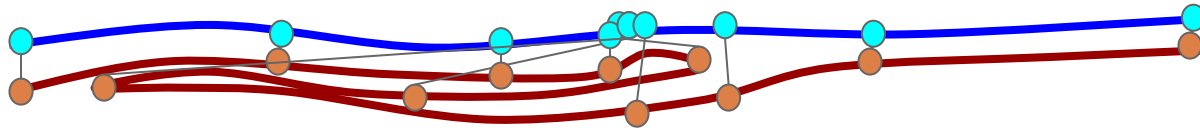
- Also described as “minimum leash length”:
 - What is the minimum length of a leash needed to stroll around the dog, given the owner’s and the dog’s trajectories?



Trajectory as sequence of points

Fréchet distance

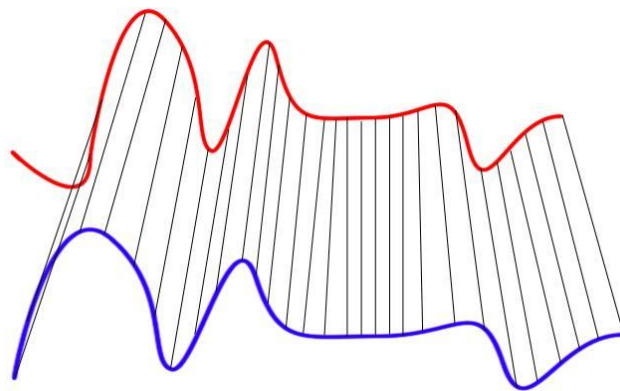
- Back to our example



Trajectory as sequence of points

Time series distances

- Just replace “difference of two values” with “spatial distance of two points”
- IMPORTANT: most methods in this class assume constant sampling rates
- Examples:
 - Dynamic Time Warping
 - Edit Distance with Real values
 - Similar to DTW, but can remove points

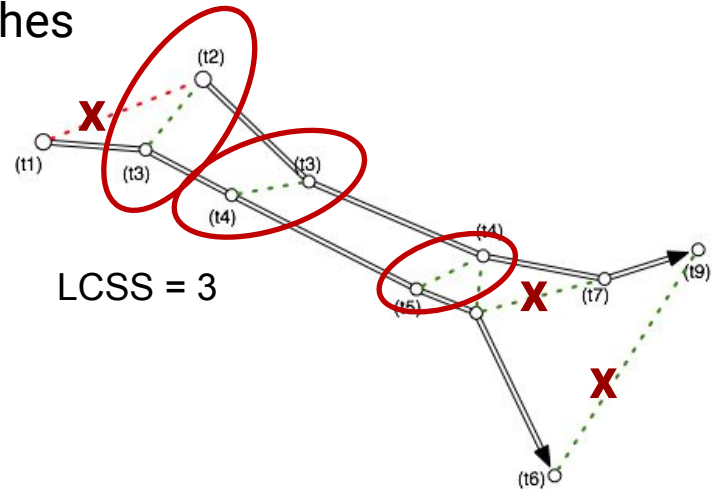


Dynamic Time Warping Matching

Trajectory as sequence of points

Time series distances

- Longest Common SubSequence
 - Define a maximum radius
 - Match points from the two trajectories if $\text{dist}() < \text{radius}$
 - Find contiguous subsequences of matches
 - LCSS = length of the best match




Trajectory as time-stamped sequence of points

Average Euclidean distance

- The trajectory is seen as a continuous spatio-temporal curve
- Positions between input points (the GPS fixes) linearly interpolated

$$D(\tau_1, \tau_2) |_T = \frac{\int_T d(\tau_1(t), \tau_2(t)) dt}{|T|}$$

distance between
moving objects τ_1
and τ_2 at time t



- “Synchronized” behaviour distance
 - Similar objects = almost always in the same place at the same time
- Computed on the whole trajectory

Clustering Algorithms

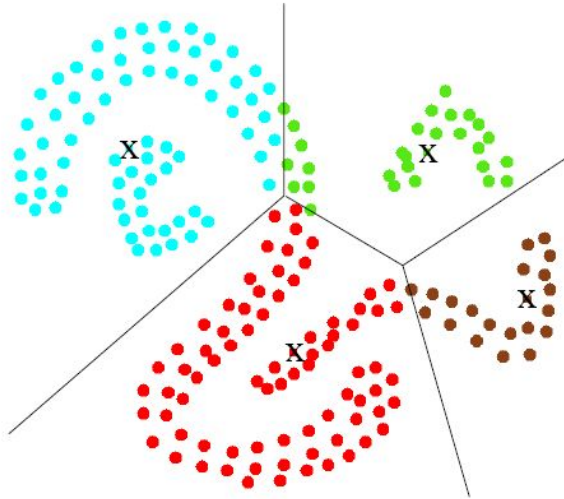
Which kind of clustering method?

- In principle, any distance-based algorithm
- General requirements:
 - Non-spherical clusters should be allowed
 - E.g.: A traffic jam along a road = “snake-shaped” cluster
 - Tolerance to noise
 - Low computational cost
 - Applicability to complex, possibly non-vectorial data
- A suitable candidate: Density-based clustering
 - OPTICS (Ankerst et al., 1999)
 - Evolution of standard DBSCAN

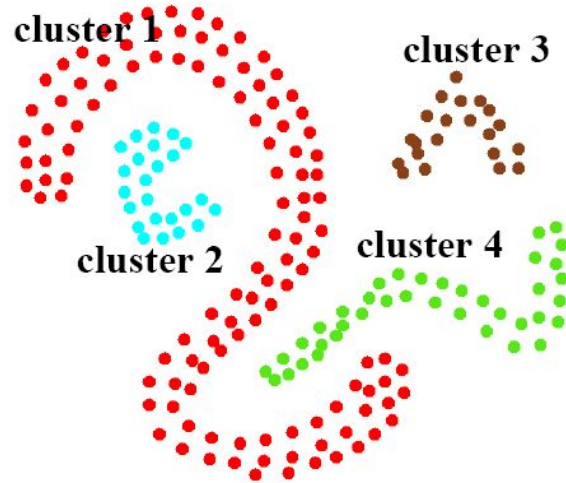
Density Based Clustering

A refresher

K-means



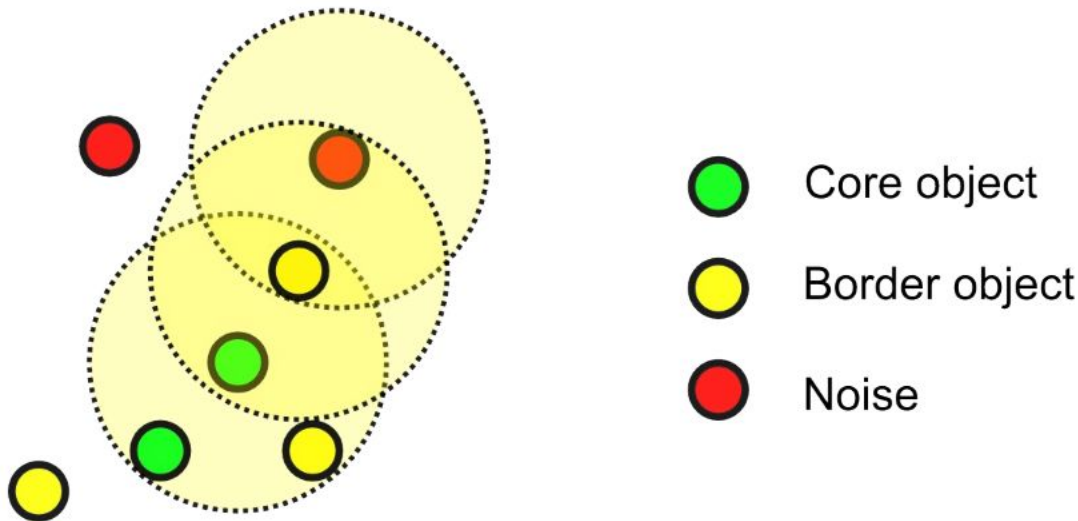
Density-based



Density Based Clustering

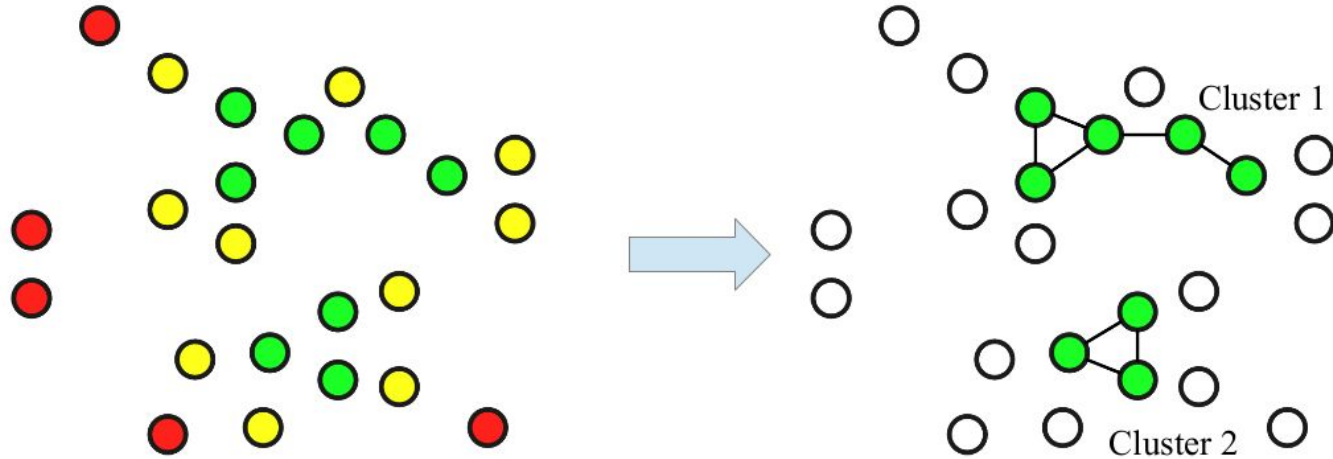
Step 1: label points as core (dense), border and noise

- Based on thresholds R (radius of neighborhood) and min_pts (min number of neighbors)



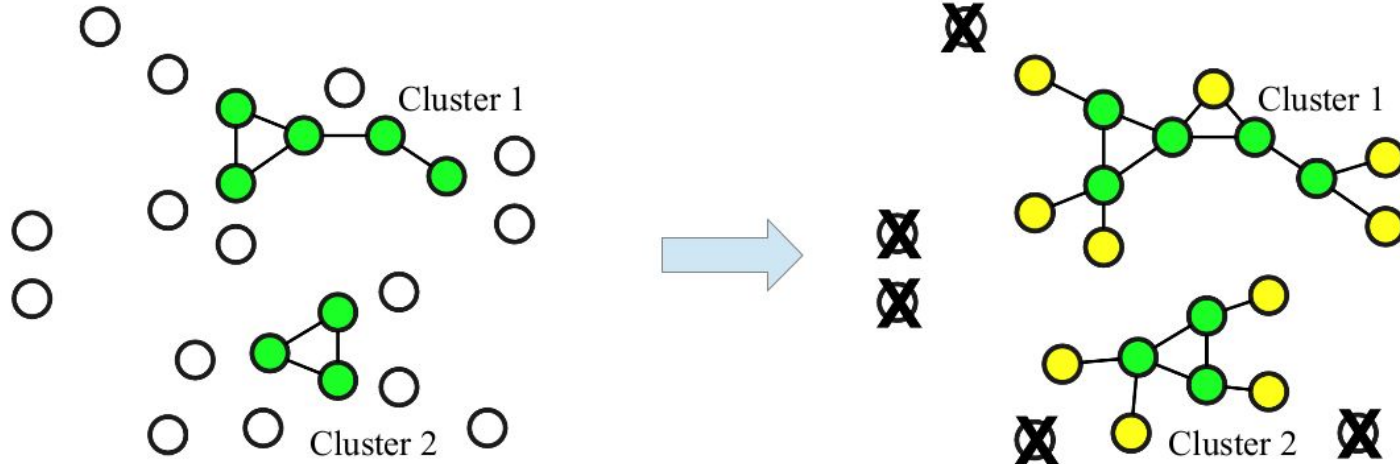
Density Based Clustering

Step 2: connect core objects that are neighbors, and put them in the same cluster



Density Based Clustering

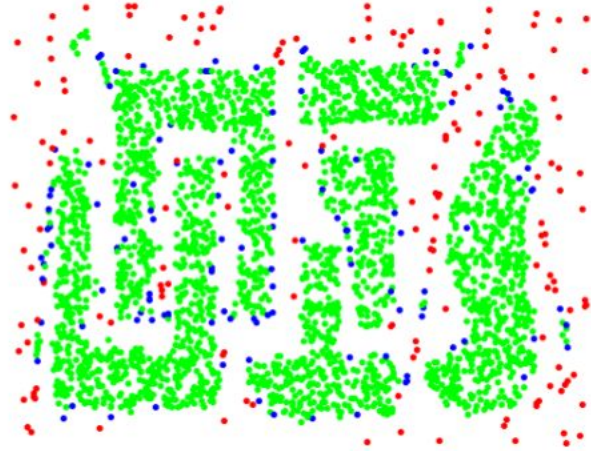
Step 3: associate border objects to (one of) their core(s), and remove noise



Density Based Clustering

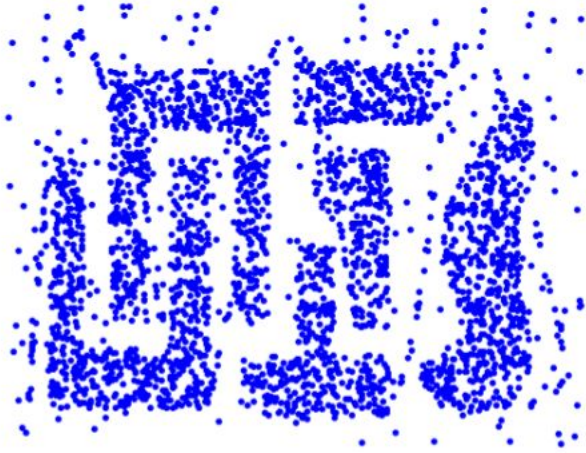


Original Points

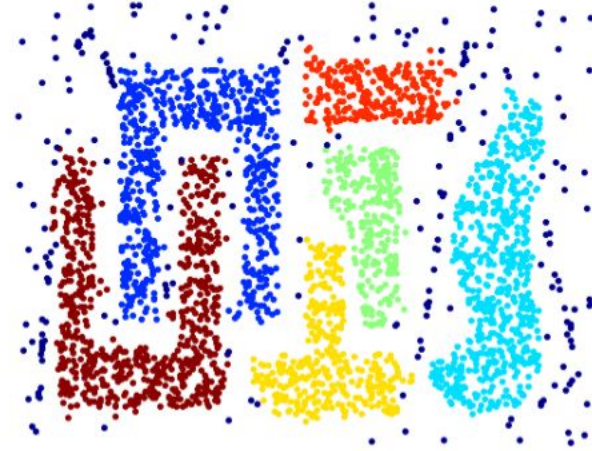


Point types: **core**,
border and **noise**

Density Based Clustering



Original Points

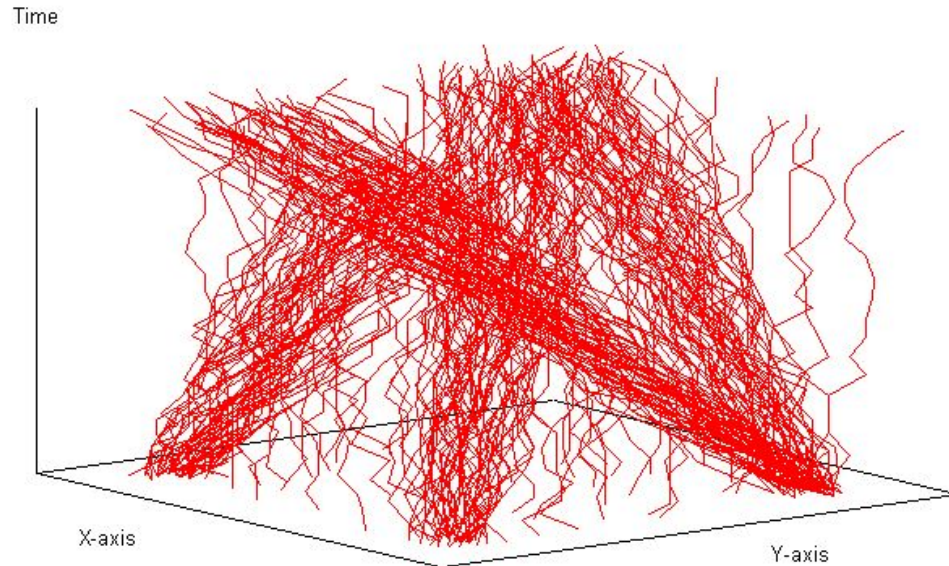


Clusters

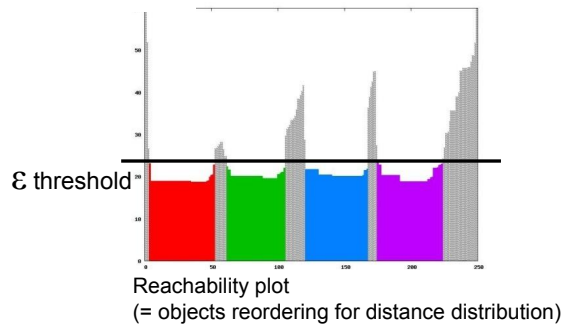
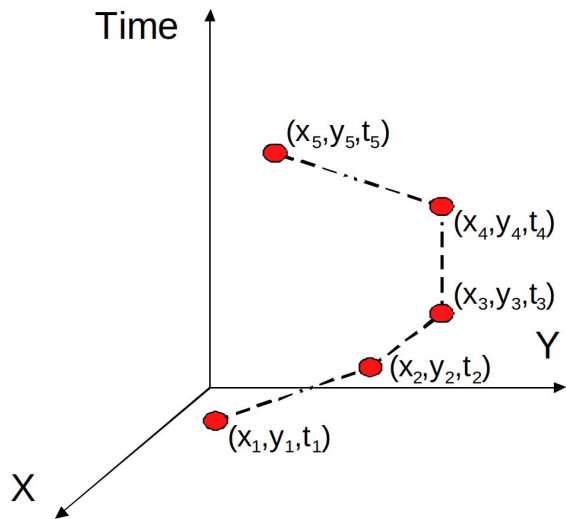
- Resistant to Noise
- Can handle clusters of different shapes and sizes

A sample dataset

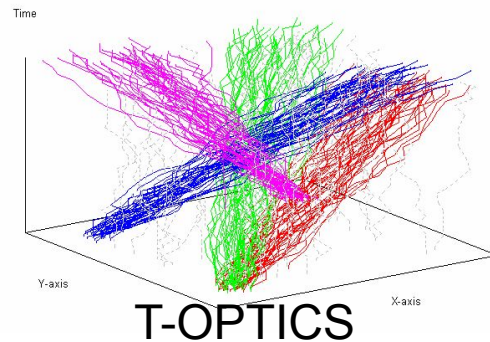
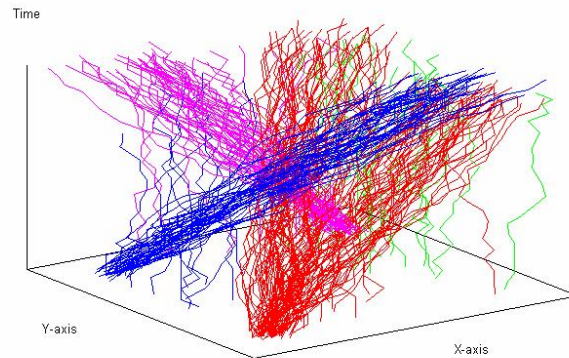
- A set of trajectories forming 4 clusters + noise (synthetic)



T-OPTICS vs. K-means



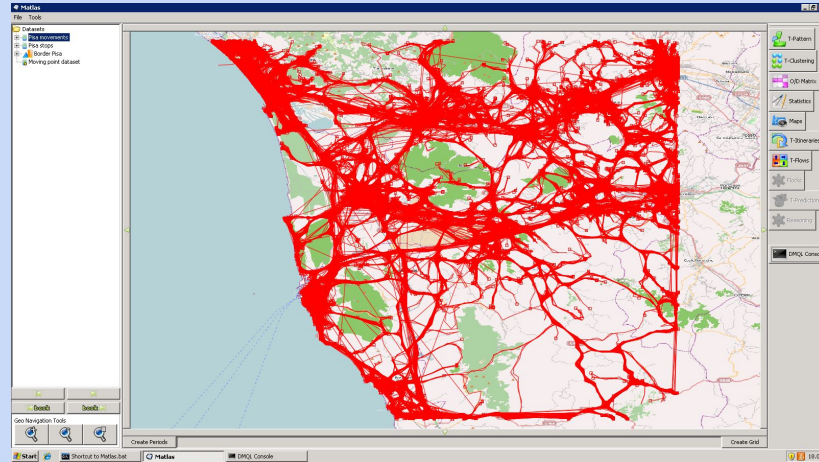
K-means



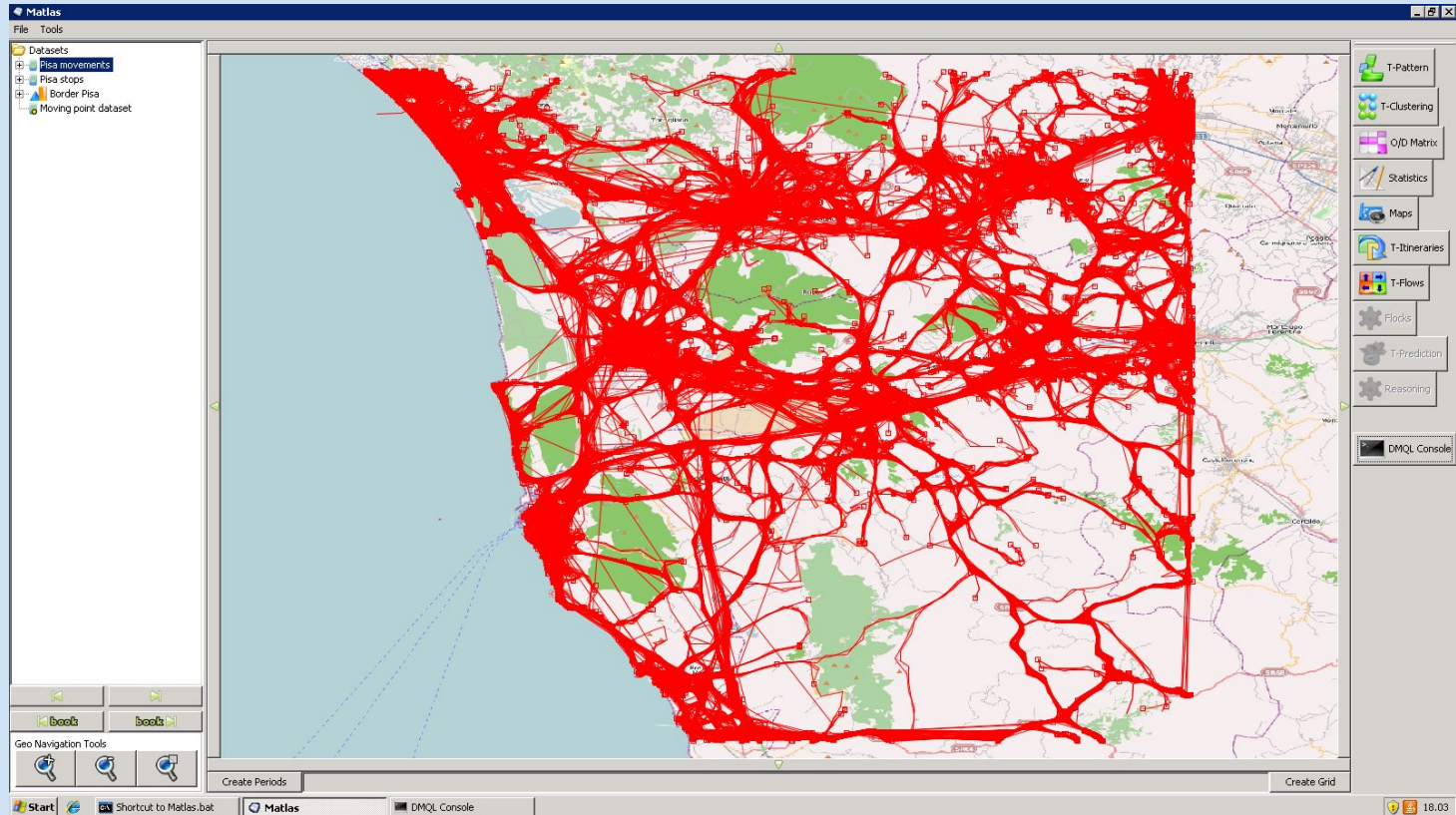
INTERVALLO

What's the source of traffic in Pisa?

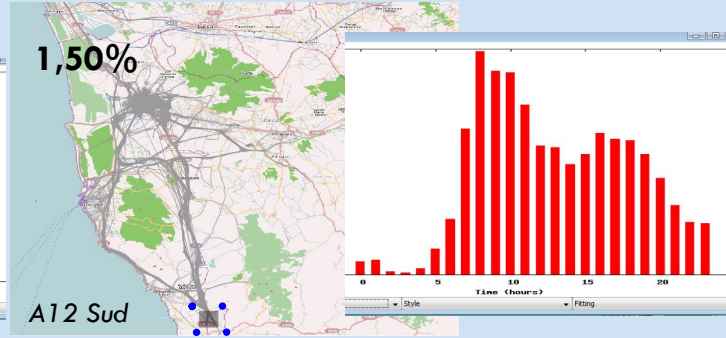
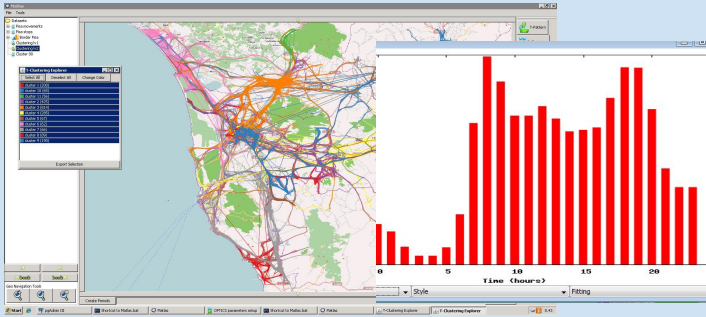
Trajectory clustering at work



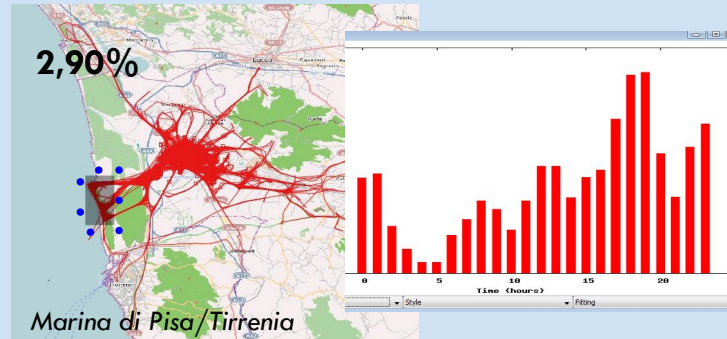
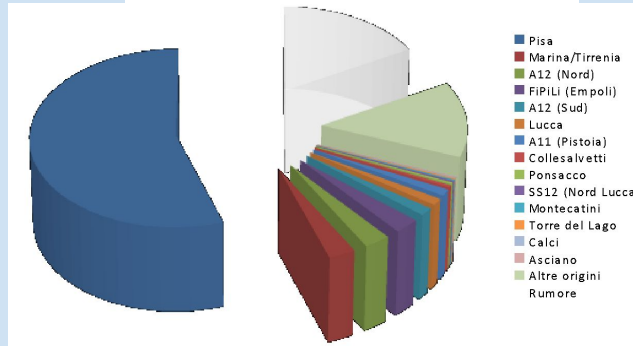
Access patterns using T-clustering



Characterizing the access patterns: origin & time



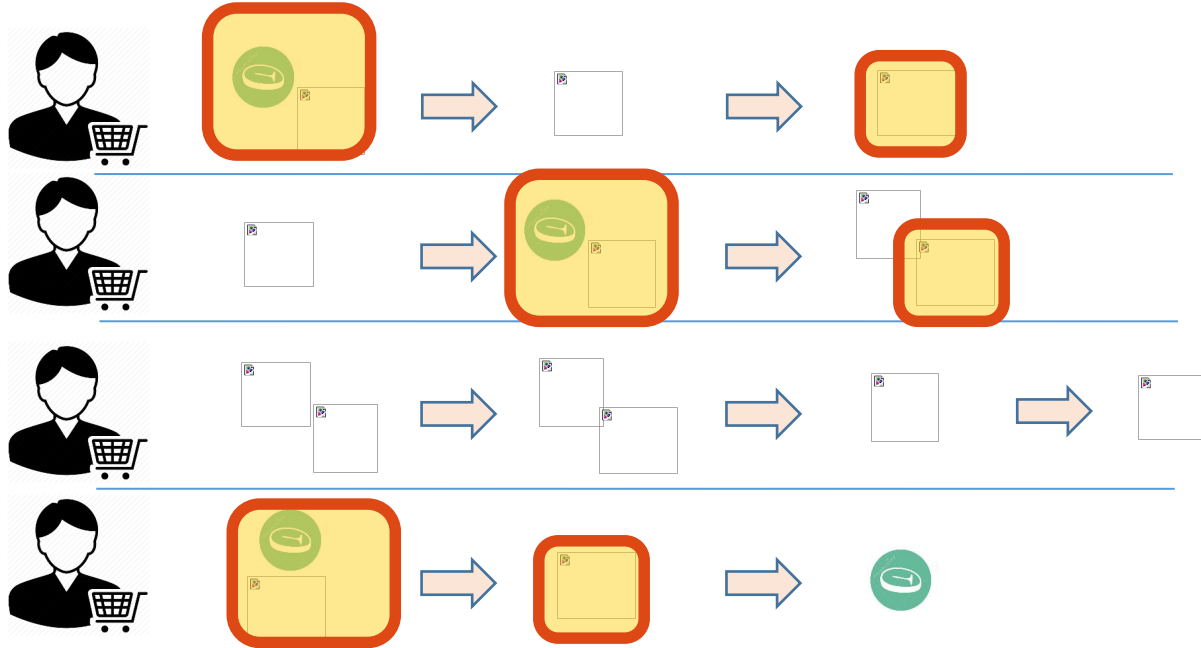
Origin distribution



Local Trajectory Patterns

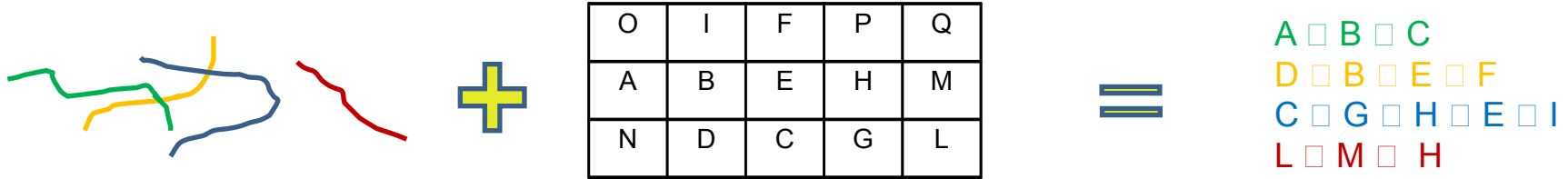
Frequent patterns in sequences

- Frequent sequences (a.k.a. Sequential patterns)
- Input: sequences of events (or of groups)



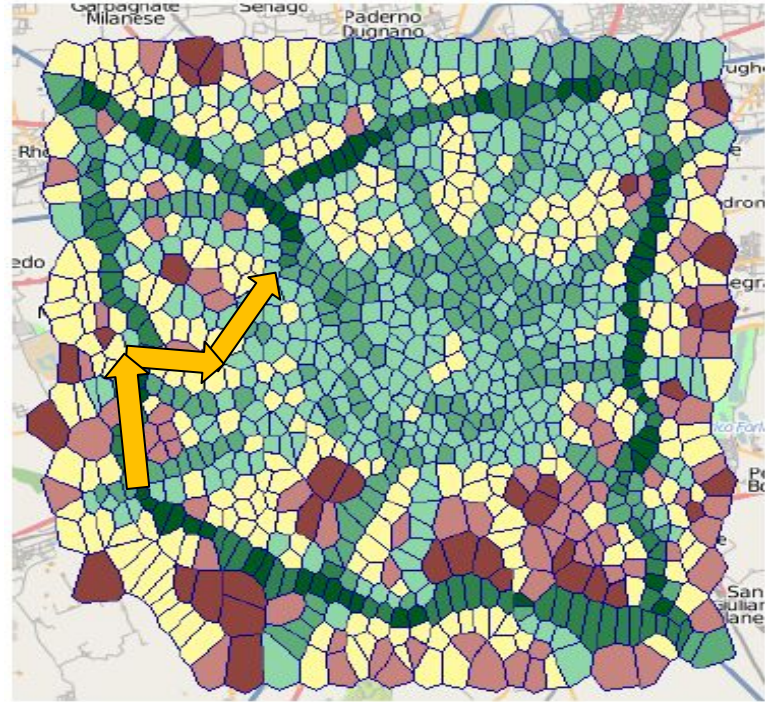
From trajectories to sequential patterns: the easy way

- Map each trajectory to a sequence of areas
 - Predefined or driven by data

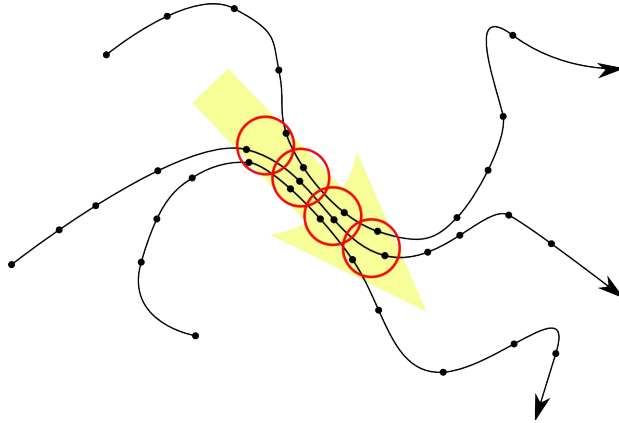


From trajectories to sequential patterns: the easy way

- A “Trajectory frequent pattern” can be defined as sequential pattern over traversed areas



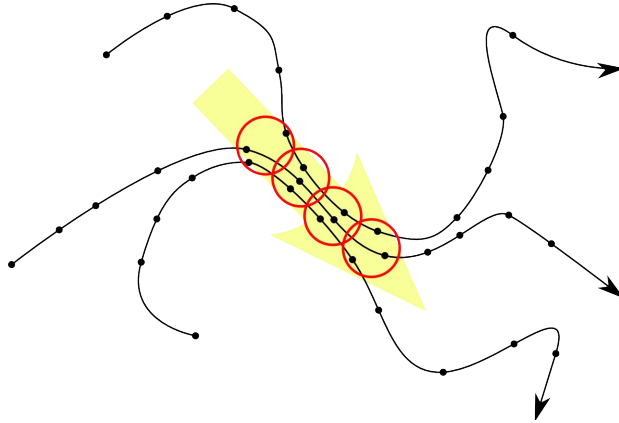
Moving Trajectory Flocks



- Group of objects that move together (close to each other) for a time interval

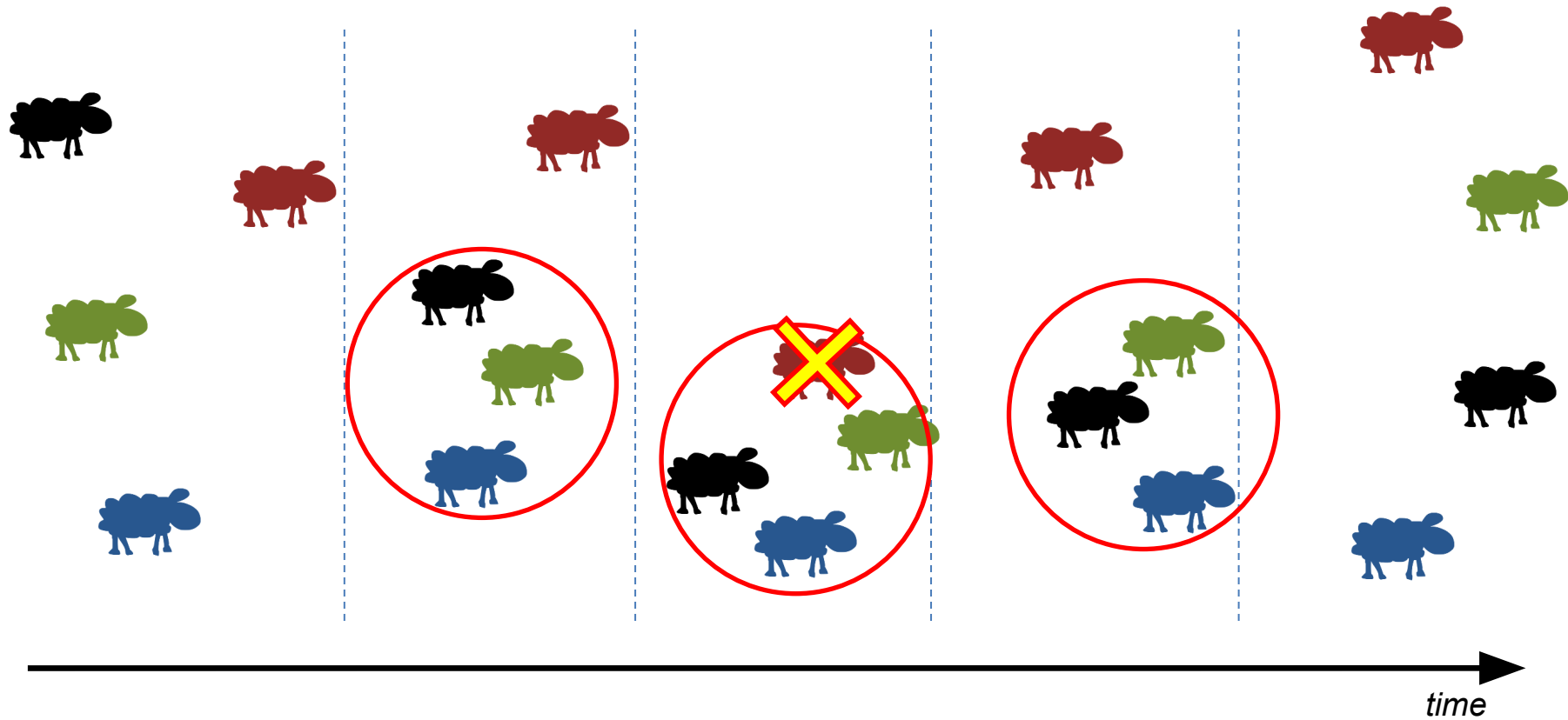


Moving Trajectory Flocks



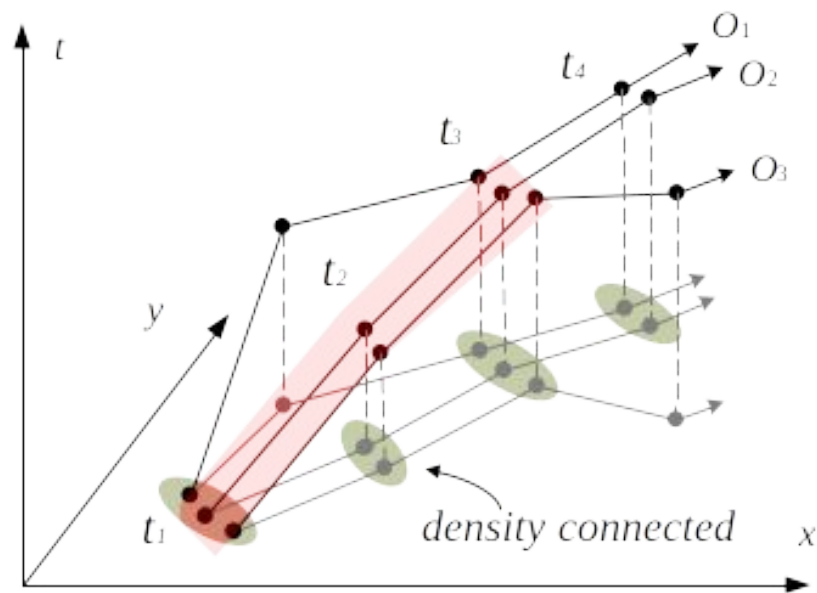
- Group of objects that move together (close to each other) for a time interval
- Discover all possible:
 - sets of objects O , with $|O| > \text{min_size}$ and
 - time intervals T , with $|T| > \text{min_duration}$
- such that for all timestamps $t \in T$ the points in $O|t$ are contained in a circle of radius r

Moving Trajectory Flocks



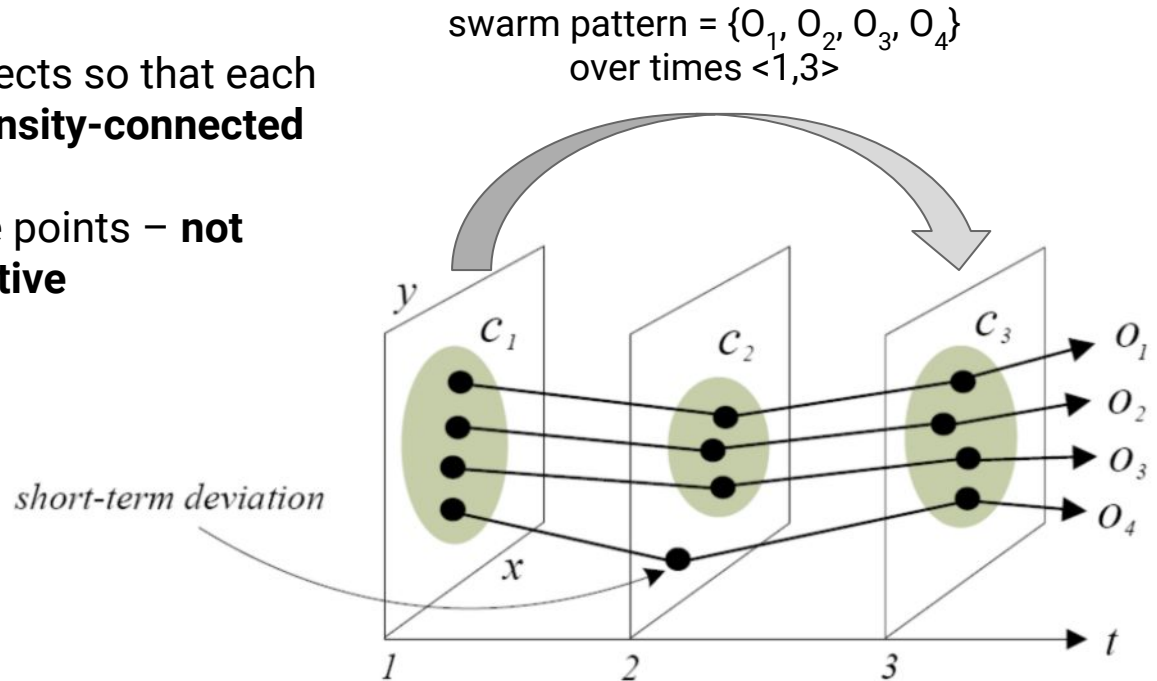
From Flocks to Convoys

- Given radius r , size m , and time threshold k
 - find all groups of objects so that each group consists of **density-connected objects** w.r.t. r and m
 - during at least k consecutive time points
- Basically replace circles with DBSCAN clusters



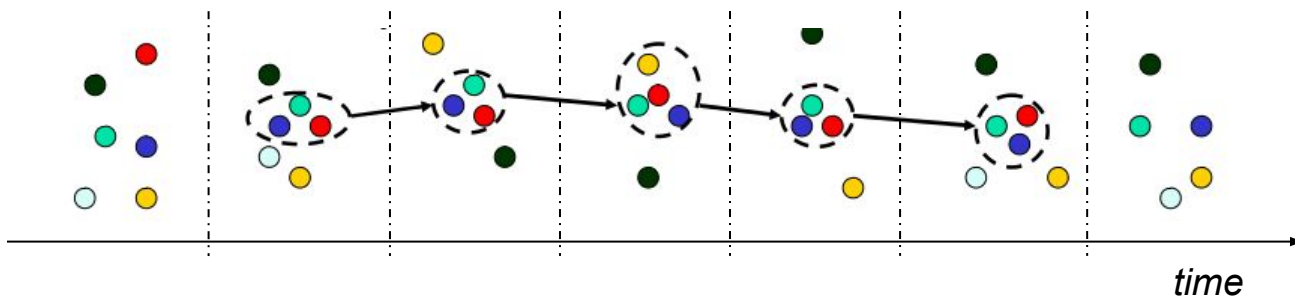
From Convoys to Swarms

- Given radius r , size m , and time threshold k
 - find all groups of objects so that each group consists of **density-connected objects** w.r.t. r and m
 - during at least k time points – **not necessarily consecutive**



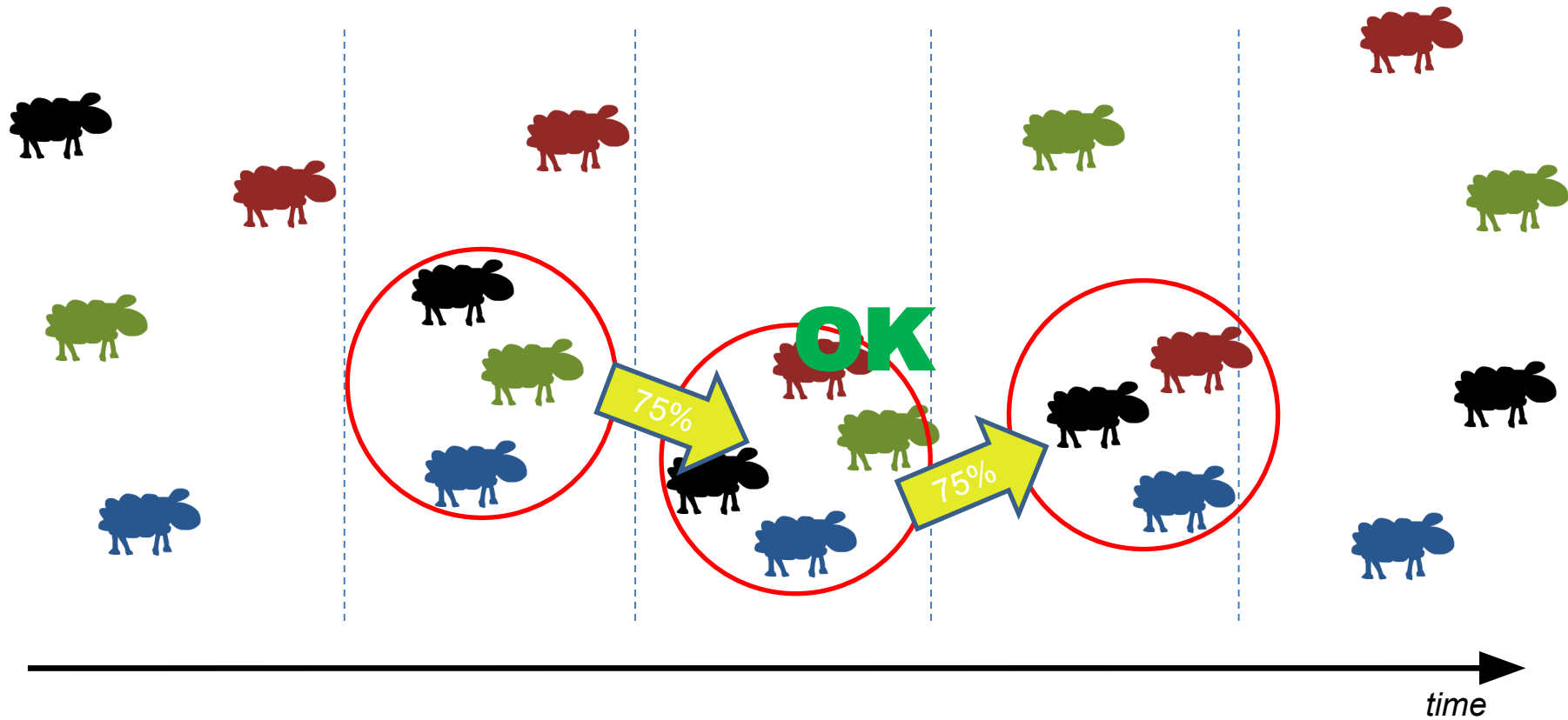
Moving Clusters

- A **moving cluster** is a set of objects that move close to each other for a long time interval



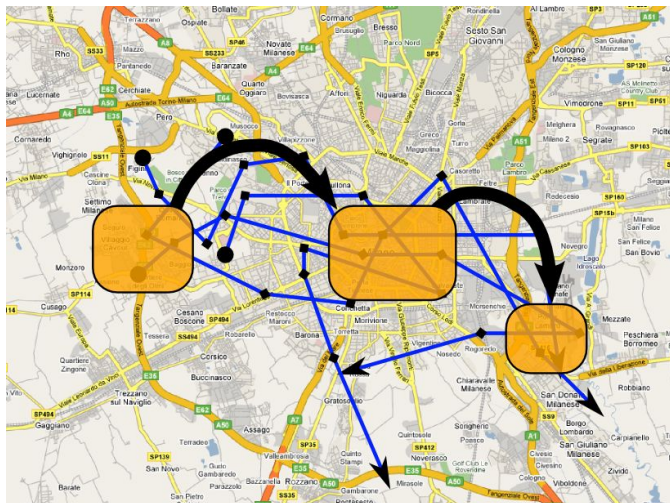
- Formal Definition [Kalnis et al., SSTD'05]:
 - A **moving cluster** is a sequence of (snapshot) clusters c_1, c_2, \dots, c_k such that for each timestamp i ($1 \leq i < k$),
$$|c_i \cap c_{i+1}| / |c_i \cup c_{i+1}| \geq \theta \quad (0 < \theta \leq 1)$$

Moving Clusters



T-Patterns

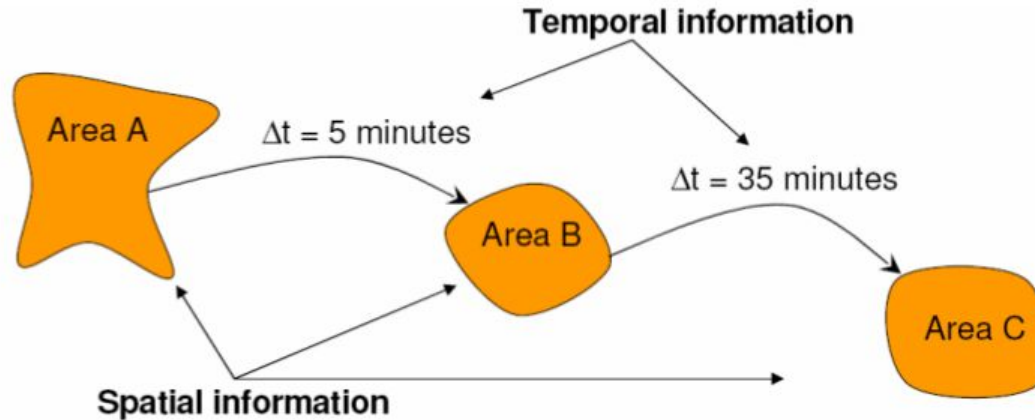
- A sequence of visited regions, **frequently** visited in the **specified order** with **similar transition times**



T-Patterns

$$A_0 \xrightarrow{t_1} A_1 \xrightarrow{t_2} \dots A_{n-1} \xrightarrow{t_n} A_n$$

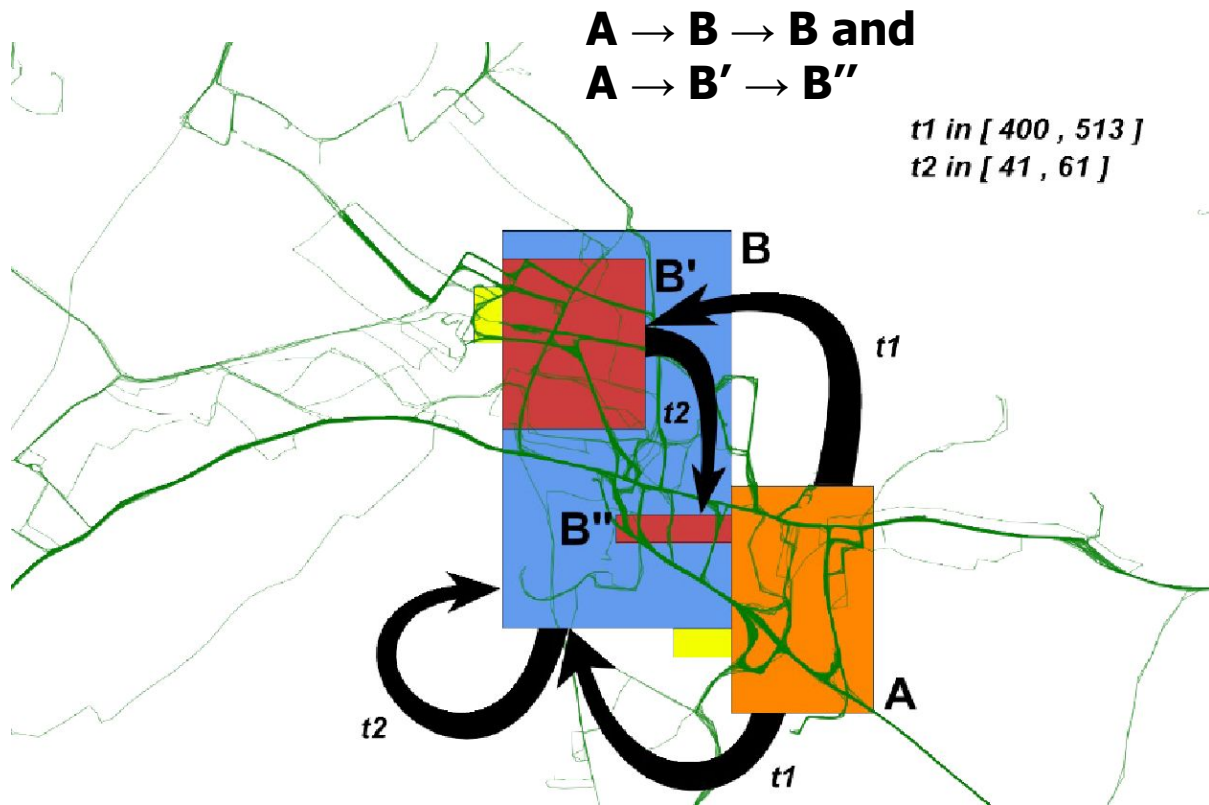
- t_i = transition time, A_i = spatial region



$$\textit{Station} \xrightarrow{20 \text{ min.}} \textit{Castle} \xrightarrow{65 \text{ min.}} \textit{Museum}$$

Sample Trajectory Pattern

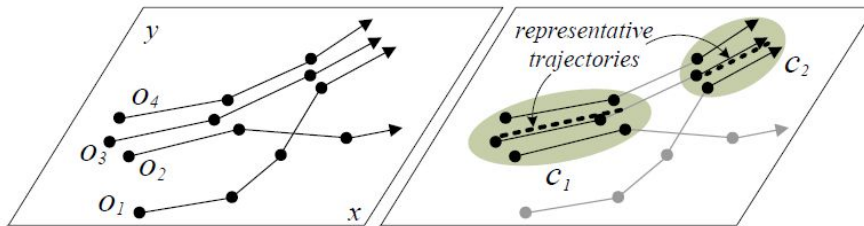
Data Source: Trucks in Athens (273 trajectories)



**Local or
Global ?**

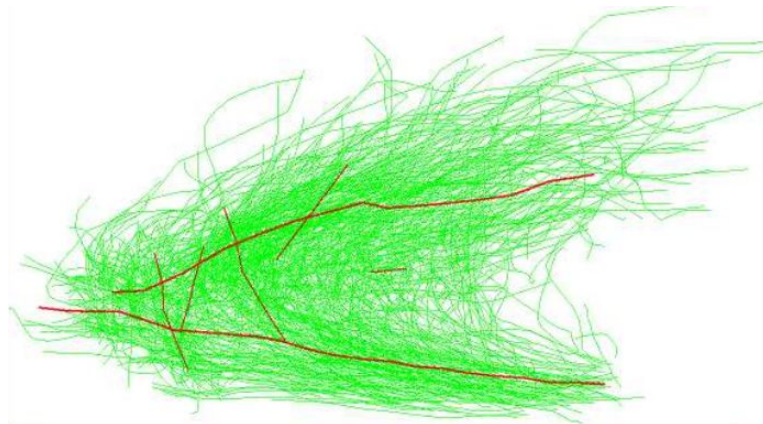
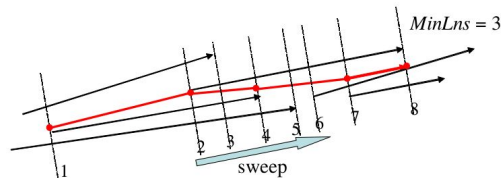
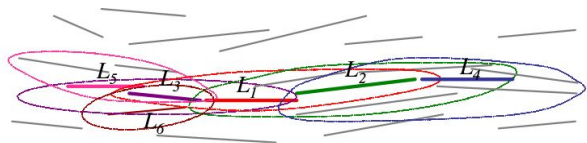
Clustering partial trajectories

- TRACLUS = TRAjectory CLUStering
 - Clustering of density-connected trajectory segments.
 - Time is not considered.
- Procedure
 1. Partition a trajectory into sub-trajectories.
 2. DBSCAN clustering is done on the sub-trajectories.
 3. Represent a cluster by a representative (sub-)trajectory



Clustering partial trajectories

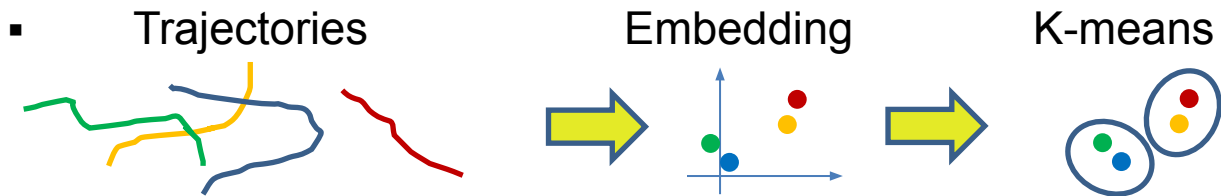
- TRACLUS
 - Connect representative segments to form “trajectories”
 - The results is similar to “moving clusters”



A quick peek into Deep Learning

Deep Learning approaches

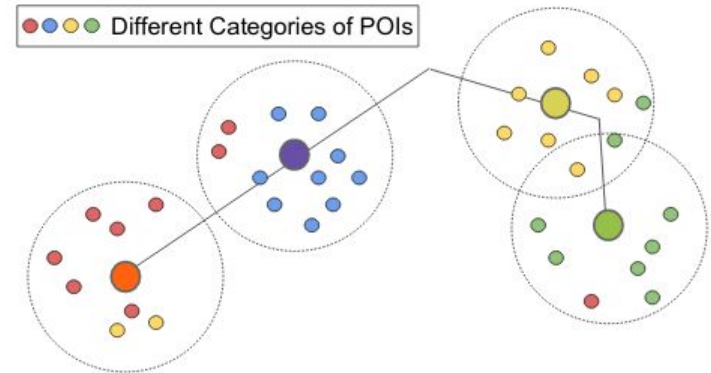
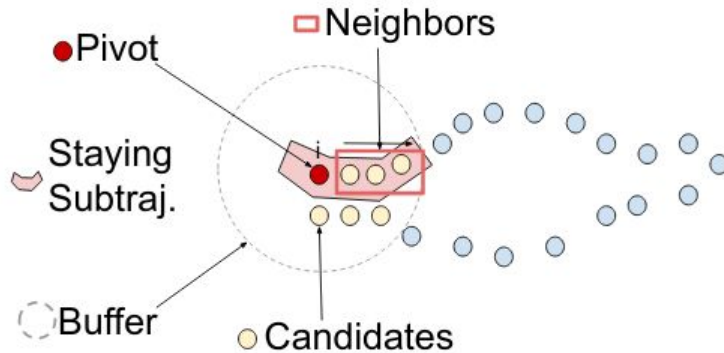
- Sample approach: DETECT: Deep Trajectory Clustering for Mobility-Behavior Analysis
- Basic idea:



- Integrate the clustering step in the learning of embeddings
- Three steps:
 - Enrich trajectories with context
 - LSTM-based embedding of trajectories
 - Clustering on embeddings

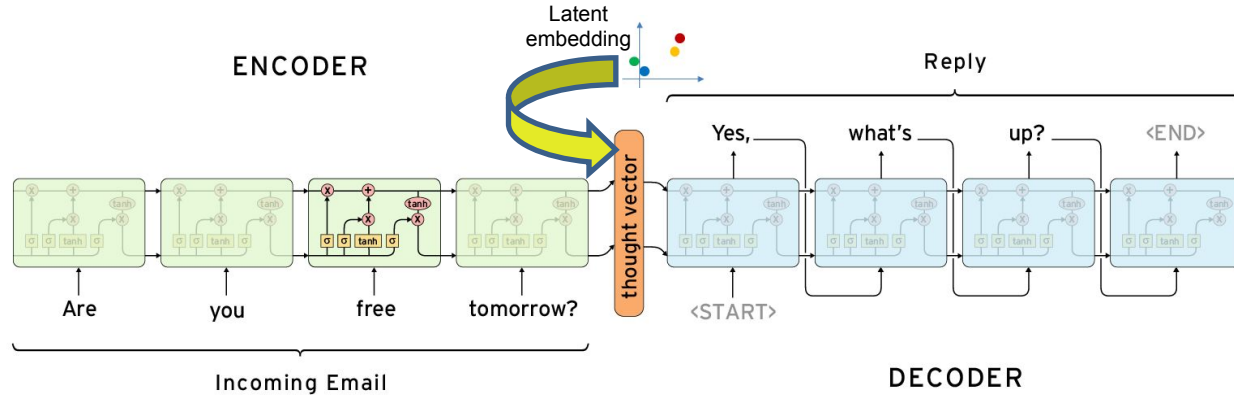
DETECT / 1

- Enrich trajectories with context
 - Identify stay areas = segment of trajectory where there is no movement, basically a stop
 - Create a buffer around the area
 - Select all points-of-interest located there (hotels, shops, etc.)
 - Compute a feature vector, one feature per PoI category
- Output
 - $Traj = \langle (x,y,[f_1,\dots, f_n]), (x',y',[f'_1,\dots, f'_n]), \dots \rangle$



DETECT / 2

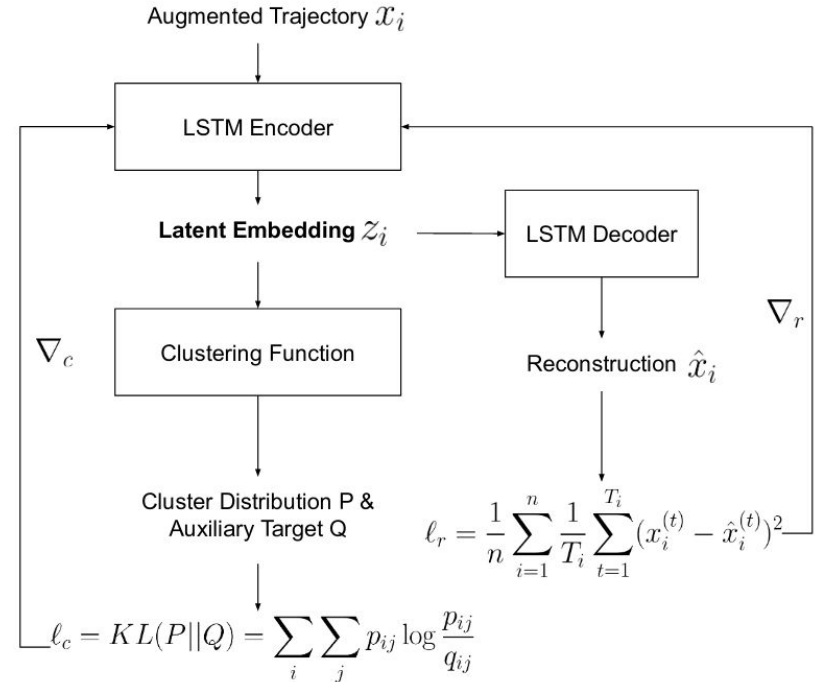
- LSTM-based embedding of trajectories
 - Apply a encoder-decoder schema to the enriched trajectories
 - Use LSTM as basic mechanism



- Objective: minimize the difference between the encoder input and the decoder output

DETECT / 3

- Clustering on embeddings
- Clustering error becomes one term of the overall loss function
- P & Q = points distribution
 - P = real data (embedded)
 - Q = clusters (Student t-distribution around centers)



Homeworks

to be delivered by Friday, October 28th 2022



Homework 6.1

Implement a simple (non-optimized) discrete version of Hausdorff distance for trajectories, i.e. considering only the GPS points and not the segments connecting them:



- Apply it to a set of taxi trips: randomly pick 10 trajectories as “query objects”; find for each of them the trips of the dataset having $d_H(\cdot) < 500$ mt; show them (query + result) on the map.
- Submit a (well commented) python notebook, where d_H is defined as a function

Homework 6.2

Define a simple “embedding” of trajectories, e.g. as trajectory length, main direction, average latitude, etc. (you decide the number of features to use); then cluster the embeddings (you decide the clustering algorithm); finally, show on a map the different clusters.

- Apply it to a (sub)set of taxi trips, e.g. SF.
- Submit a (well commented) python notebook

Homework 6.3

Mimicking TraClus

Strongly simplify a dataset D of trajectories (output = D'), then build a second dataset D'' containing, for each trip in D' , all its segments. Then, cluster the segments in D'' using the coordinates of start and end as attributes for clustering (4 attributes per segment), and show results on a map. You decide the clustering algorithm to use.

- Apply it to a (sub)set of taxi trips, e.g. SF.
- Submit a (well commented) python notebook