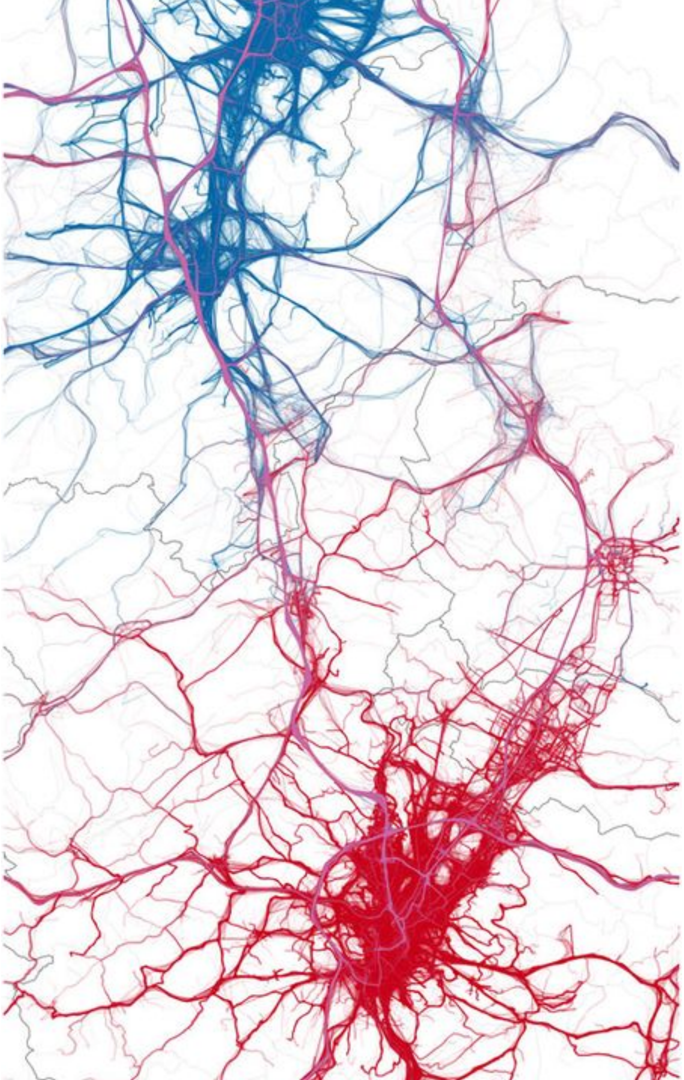


Preprocessing Mobility Data

Consiglio Nazionale delle Ricerche

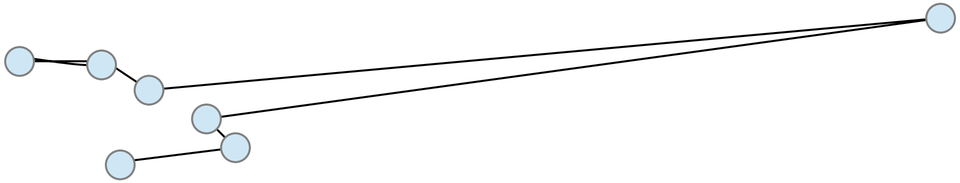
Content of this lesson

- Preprocessing trajectories
  - trajectory filtering
  - point map matching
  - route reconstruction
  - trajectory compression
  - Semantic enrichment
    - stop detection / trajectory segmentation
    - home location detection (GPS & MobPhones)
    - activity recognition (POI-based)

Trajectory filtering

- Data points are sometimes affected by errors
- Errors can have huge effects on results

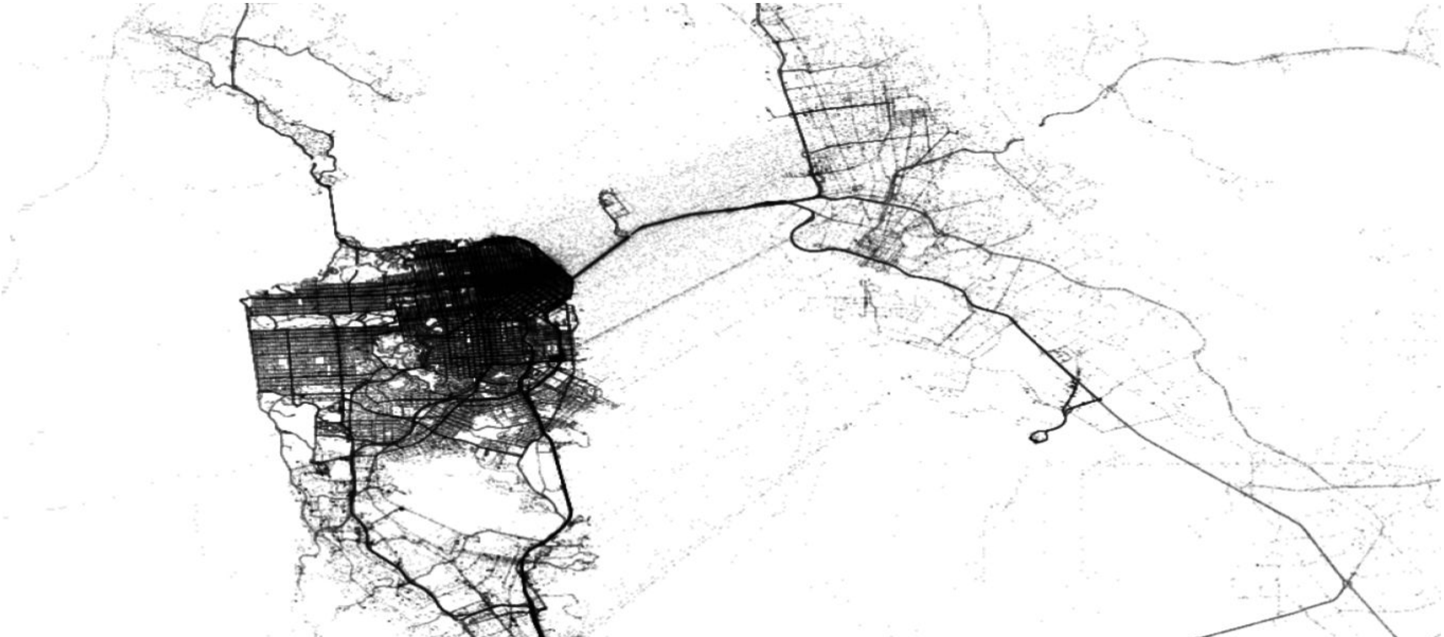
What is the real length of this trip?



- Two families of approaches:
  - Context-based filtering
  - Movement-based filtering

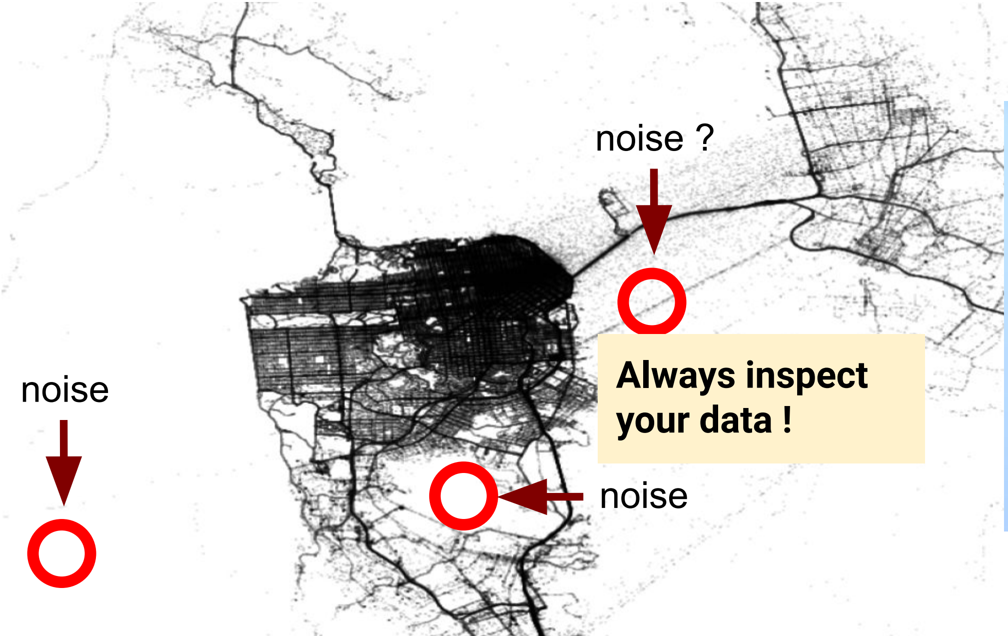
Context-based filtering

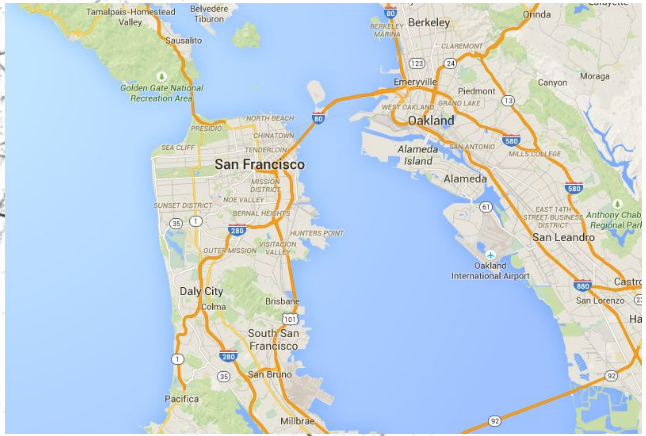
- Single points might contain errors of various kinds



Context-based filtering

- Single points might contain errors of various kinds
- **Map-based detection**: cars on the water or out of roads are noise
    - Caution: do you trust 100% your map?





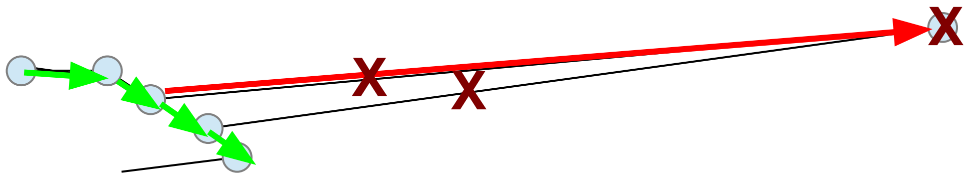
Movement-based filtering

- No context is used, just the geometry / dynamics of movement

- **Speed-based** noise filtering approach:
  - The first point of the trajectory is set as valid
  - Scan all remaining points "p" of the trajectory (time order)
    - Compute "v" = average straight-line speed between point "p" and the previous valid one
    - If "v" is huge (e.g. larger than 400 km/h)
      => remove "p" from trajectory ("p" will not be used next to estimate speeds…)

      else

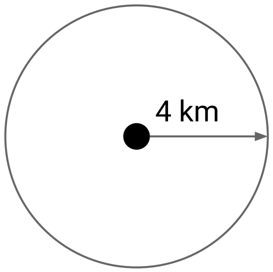
      => set "p" as valid

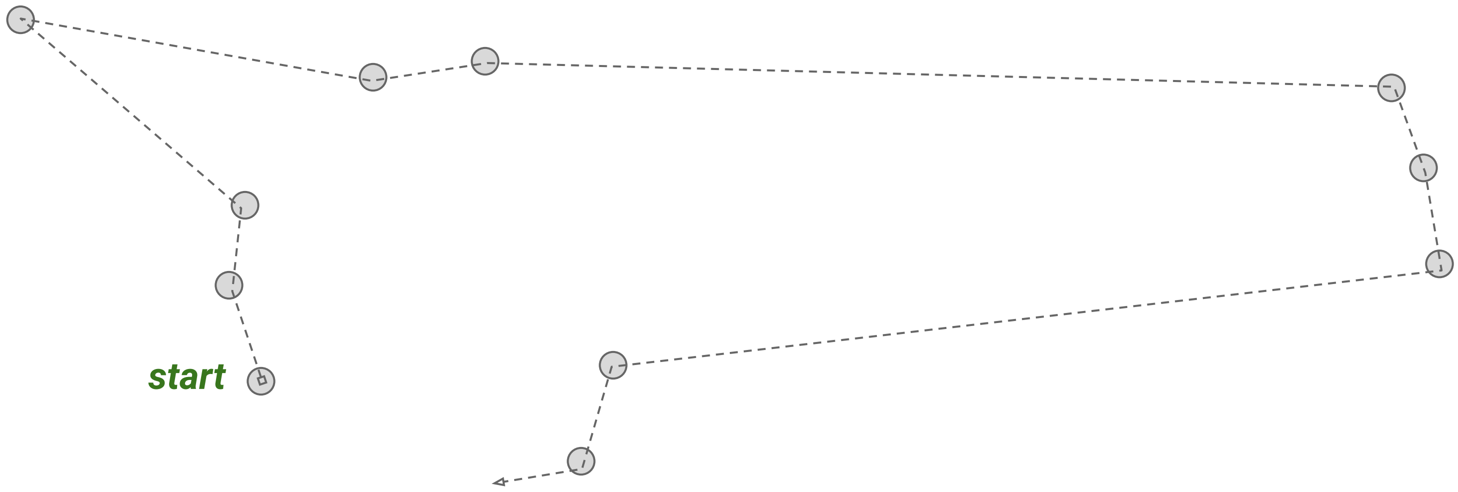


Movement-based filtering

Exercise

- What happens in this situation? (Multiple noisy points)
  - Assume constant sampling rate 1 minute
  - Speed threshold = 240 km/h (= 4 km/minute)

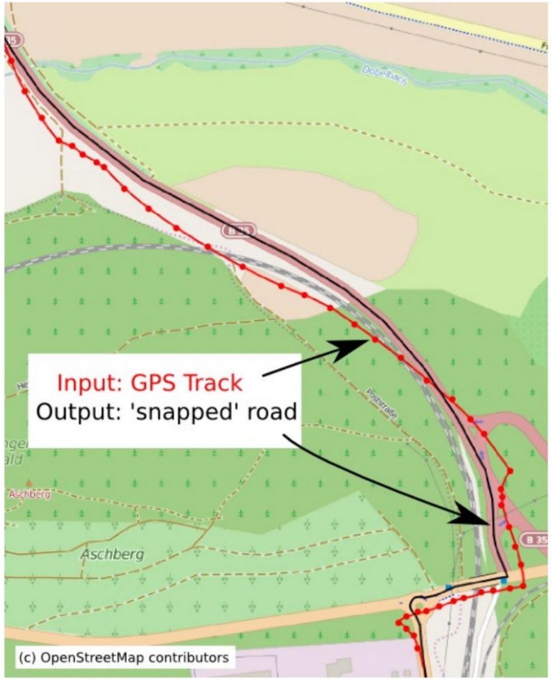




Point map matching

- Points can be aligned to the road network
  - Objective 1: improve accuracy of position
  - Objective 2: remove extreme cases (ref. filtering)
  - Objective 3: translate trajectories to sequences of road IDs

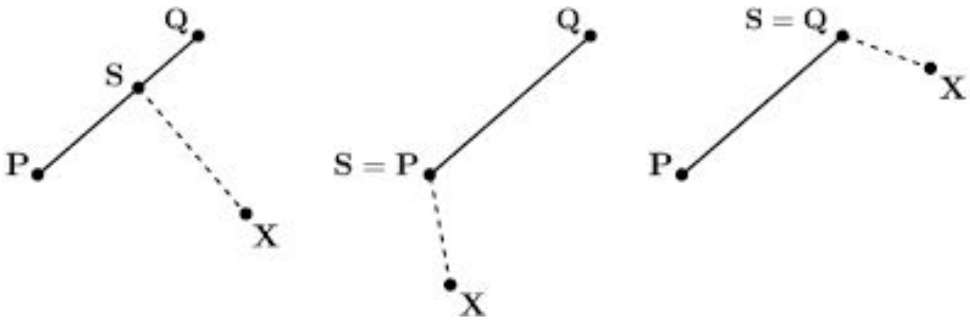
- Idea: project the point to the close location in the network
  - Usually there is a maximum threshold
  - Points farther than the threshold from any road are removed as noise



Point map matching

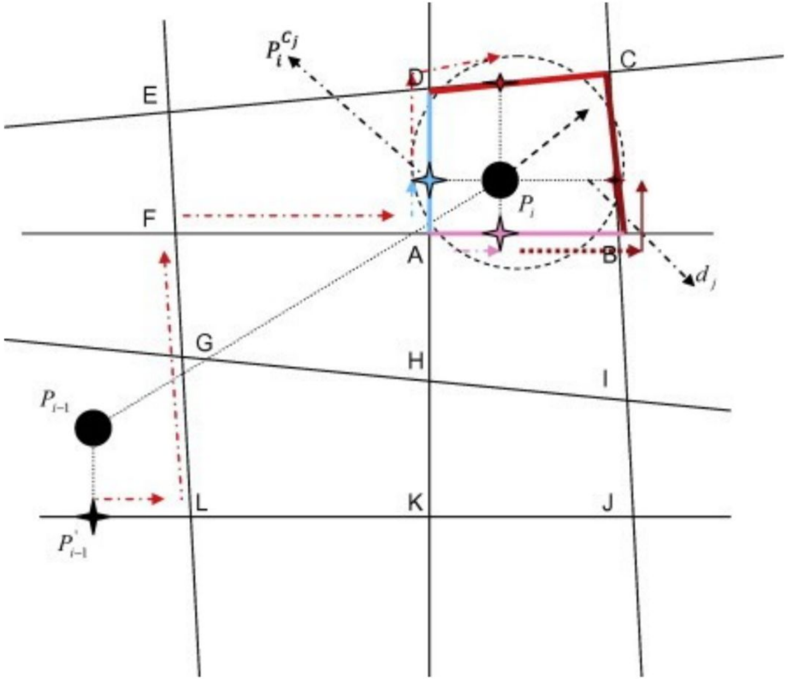
- Point projection
  - Requires to compare each point to each road segment

- Refresher on point-to-segment distance computation



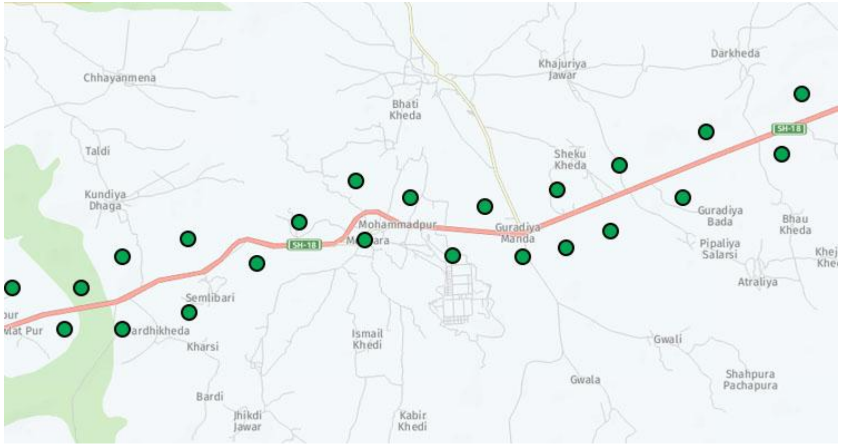
Point map matching

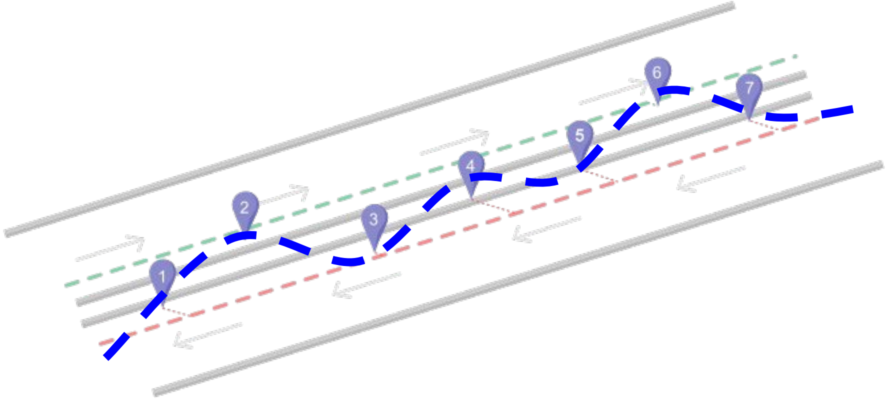
- In some contexts there can be multiple choices at reasonable distance
  - Simply taking the closest one is “risky”



Point map matching

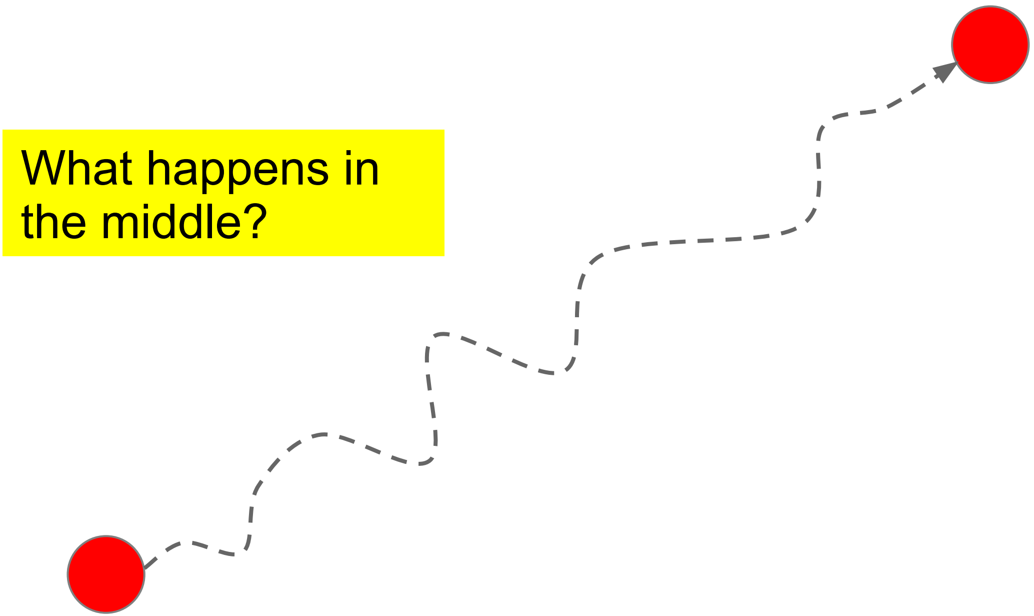
- Matching points separately can lead to inconsistent results
  - Mainly road-dense areas with position accuracy comparable to road separation
- Need a trajectory-level matching
  - Linked to route reconstruction





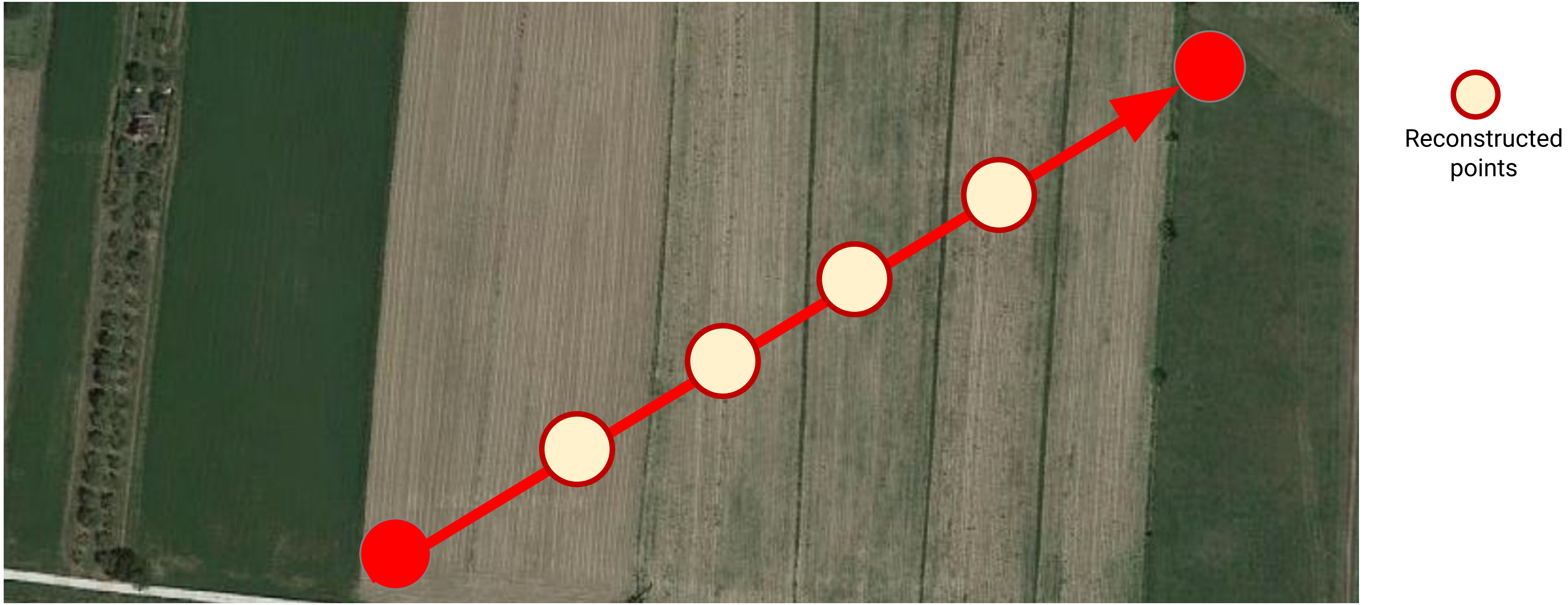
Route reconstruction

- Sometimes the space/time gap between consecutive points is significant



Route reconstruction

- Typical solutions:
  - Free movement => straight line, uniform speed



Route reconstruction

- Typical solutions:
  - Constrained movement => shortest path

Reconstructed points

Route reconstruction

Shortest paths can be replaced by alternative “optimal paths”

- Based on a notion of path cost
- Typical ones: path **length**, path **duration** (requires to know typical traversal times of roads)
- Alternative ones: fuel consumption, EV battery consumption, CO2 emissions, mixed costs

Algorithms applied are standard graph path optimization methods:

- Dijkstra's algorithm → efficient, requires that costs are non-negative
- Bellman-Ford algorithm → less efficient, can work with negative weights (but no cycles)

See *method* parameter of *shortest_path* function of NetworkX

Refresher: Dijkstra’s minimum cost algorithm

- Simple and efficient: O(M + N log N) time complexity (M = |edges|, N = |nodes|)

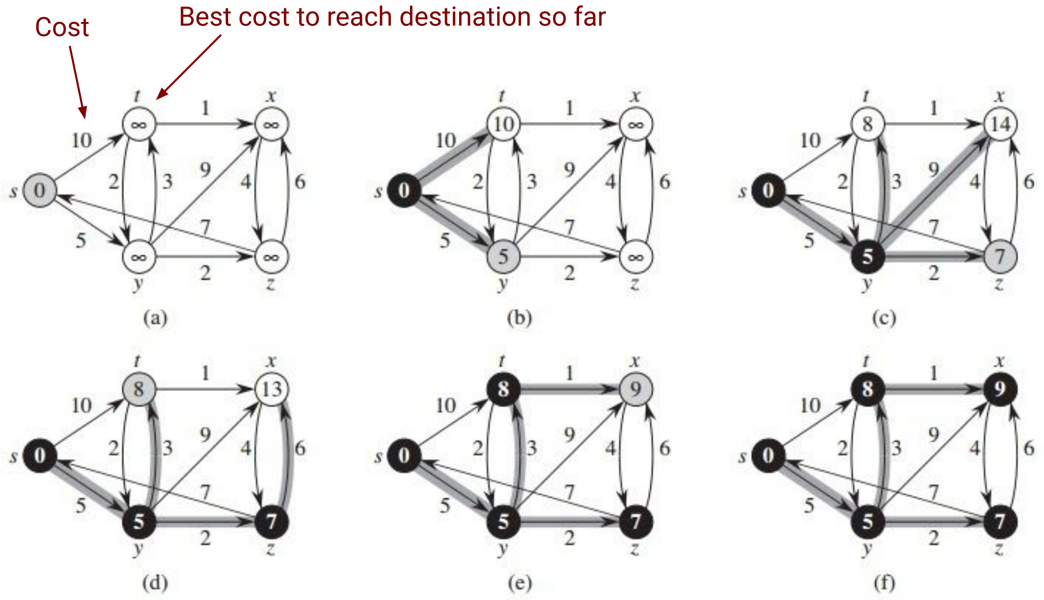
```
function Dijkstra(Graph, source):

    for each vertex v in Graph.Vertices:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]
        remove u from Q

        for each neighbor v of u still in Q:
            alt ← dist[u] + Graph.Edges(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```



Trajectory Map Matching

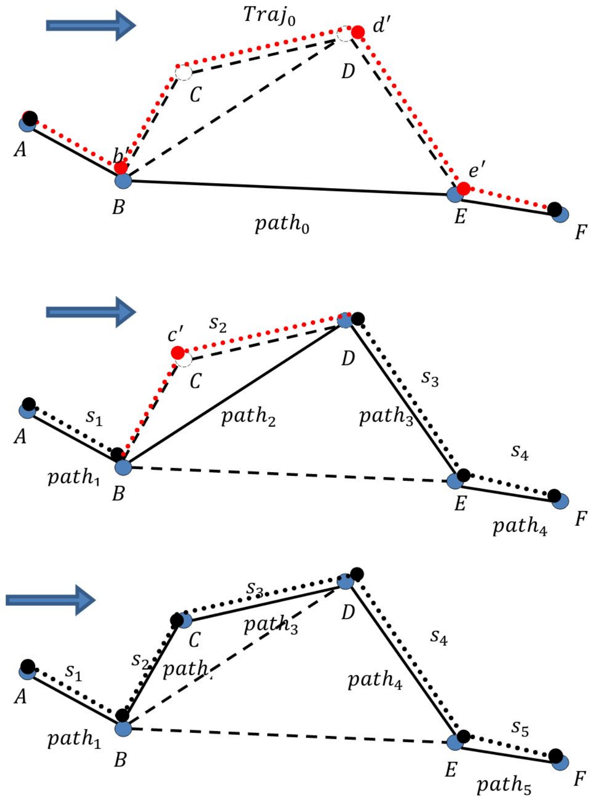
- Assigns points to road segments
- Reconstructs the movement between consecutive points
- Ensures coherence of the overall process

- Two sample approaches:
  - Based on shortest path
  - Based on probabilities

Shortest path-based Map Matching

Used by MappyMatch

- Similar ideas as trajectory simplification
  - Match first and last point
  - Compute shortest path on the network
  - Find farthest point from shortest path
  - If distance > threshold ⇒
    - split into two parts
    - run recursively the process on both

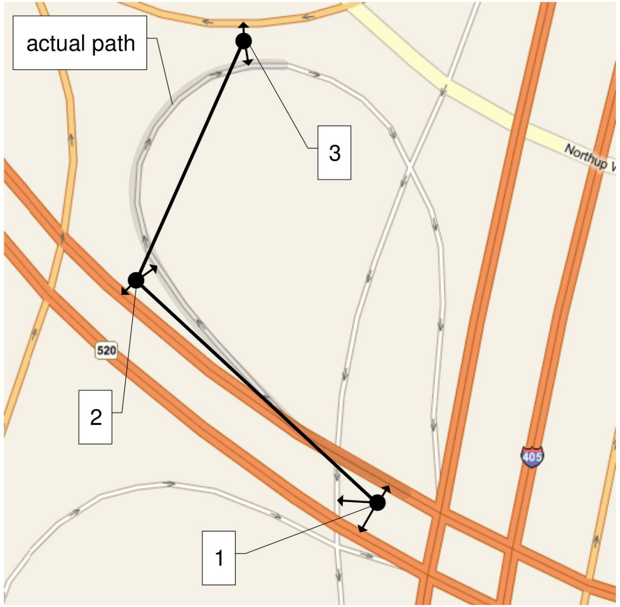


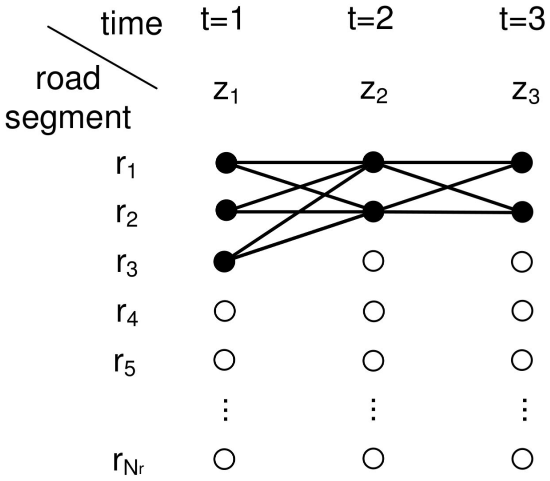
Reference: Zhu, honda & Gonder. A Trajectory Segmentation Map Matching Approach for Large-Scale, High-Resolution GPS Data. TRB 2017.

Probability-based Map Matching

Used by pyTrack

- Consider possible point-to-road assignments, with probabilities
- Compute most likely path that visits all points in the correct sequence

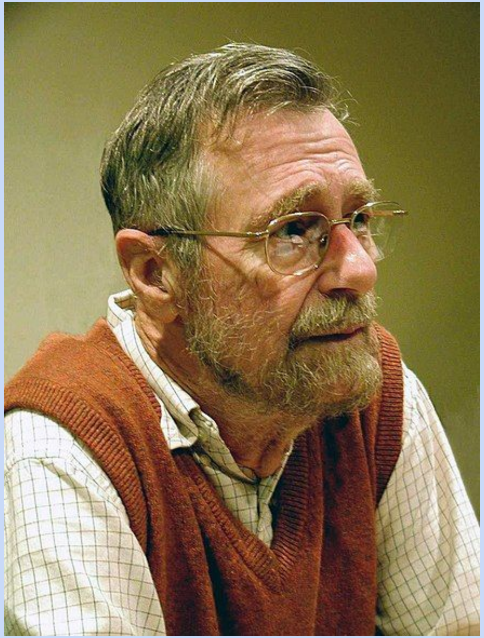




Reference: Newson & Krumm. Hidden Markov Map Matching Through Noise and Sparseness. ACM GIS'09.

INTERVALLO

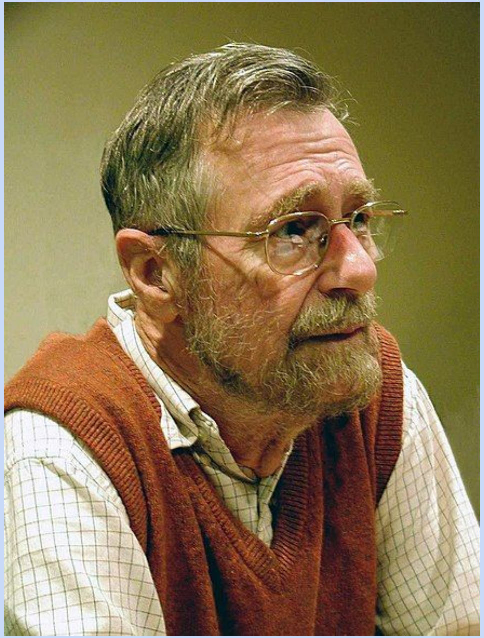
Who’s Dijkstra



- 1930 - 2002
- Dutch computer scientist, programmer, software engineer, systems scientist, and science essayist
- 1972 Turing Award for “fundamental contributions to developing programming languages”

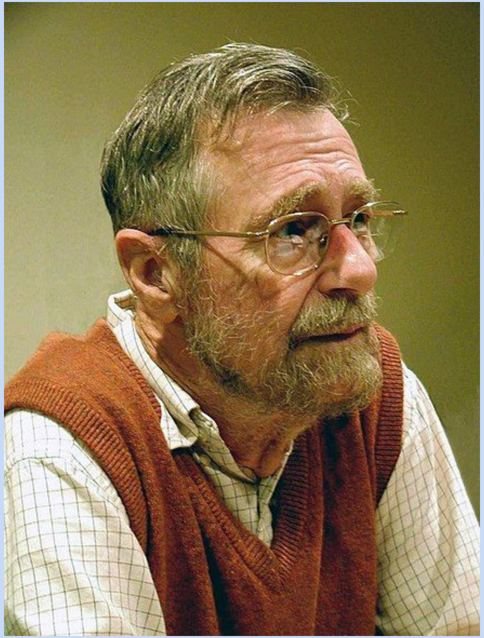
INTERVALLO

Dijkstra is famous for...



- Dijkstra’s algorithm, of course
- Contributions to “self-stabilization of program computation”
    - Won him the “ACM PODC Influential Paper Award”, later renamed “Dijkstra Prize”
- Hundreds of papers on computational and science philosophy issues

Dijkstra is famous for...

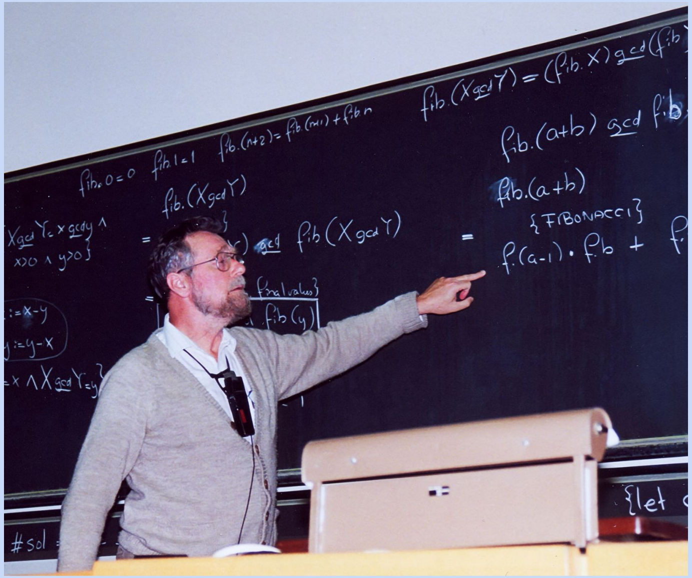


- His habit of writing everything with paper & fountain pen
- Hundreds of papers, many unpublished
    - E. W. Dijkstra Archive
- Counting should start from 0, not 1...

When dealing with a sequence of length N, the elements of which we wish to distinguish by subscript, the next vexing question is what subscript value to assign to its starting element. Adhering to convention a) yields, when starting with subscript 1, the subscript range $1 \leq i < N+1$; starting with 0, however, gives the nicer range $0 \leq i < N$. So let us let our ordinals start at zero: an element's ordinal (subscript) equals the number of elements preceding it in the sequence. And the moral of the story is that we had better regard —after all those centuries!— zero as a most natural number.

INTERVALLO

Dijkstra the teacher



- Chalk & blackboard, no projectors
- No textbooks
- Improvisation & long pauses
- No references in papers

  "For the absence of a bibliography I offer neither explanation nor apology."
- Long exams
  - Each student was examined in Dijkstra's office or home, and an exam lasted several hours

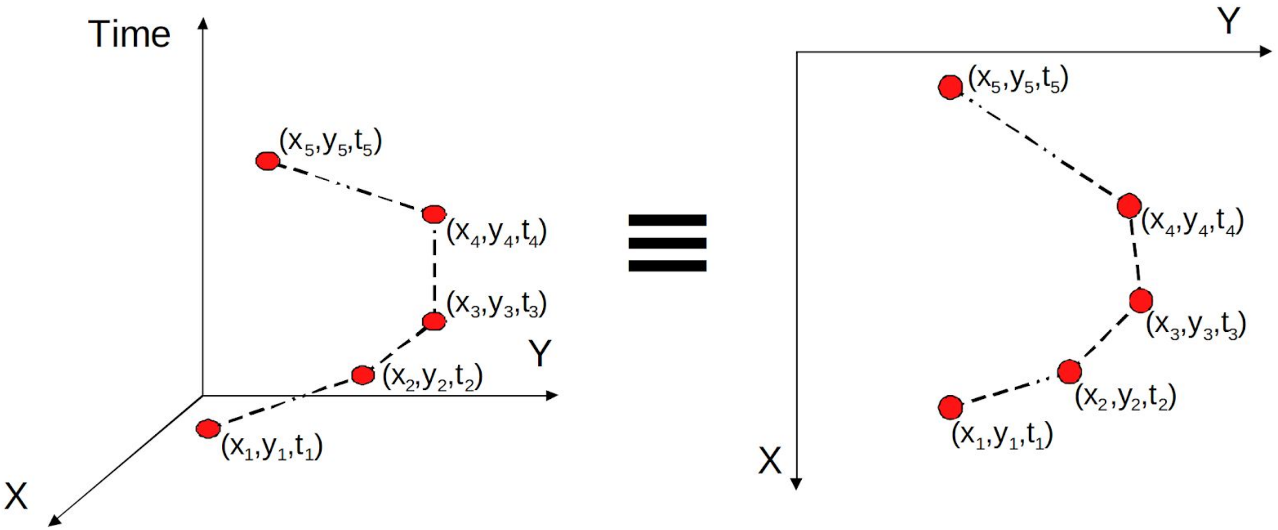
Trajectory compression / simplification

- Many algorithms for trajectories are expensive
  - Their complexity depends on the number of points
  - Sometimes trajectories have more points than needed

- Objective of compression / simplification
  - Reduce the number of points…
  - … without affecting the quality of results

Trajectory data

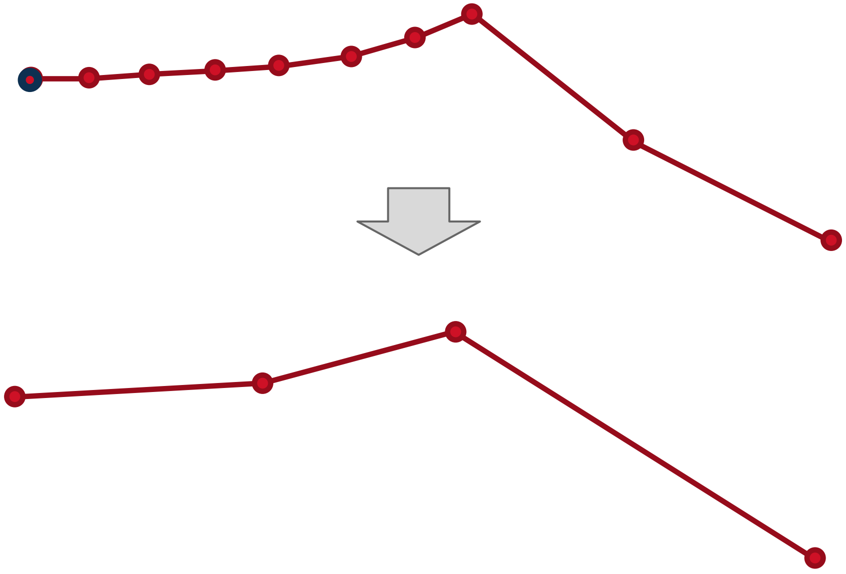
- A trajectory is a temporal sequence of time-stamped locations
- Most methods focus on the spatial component



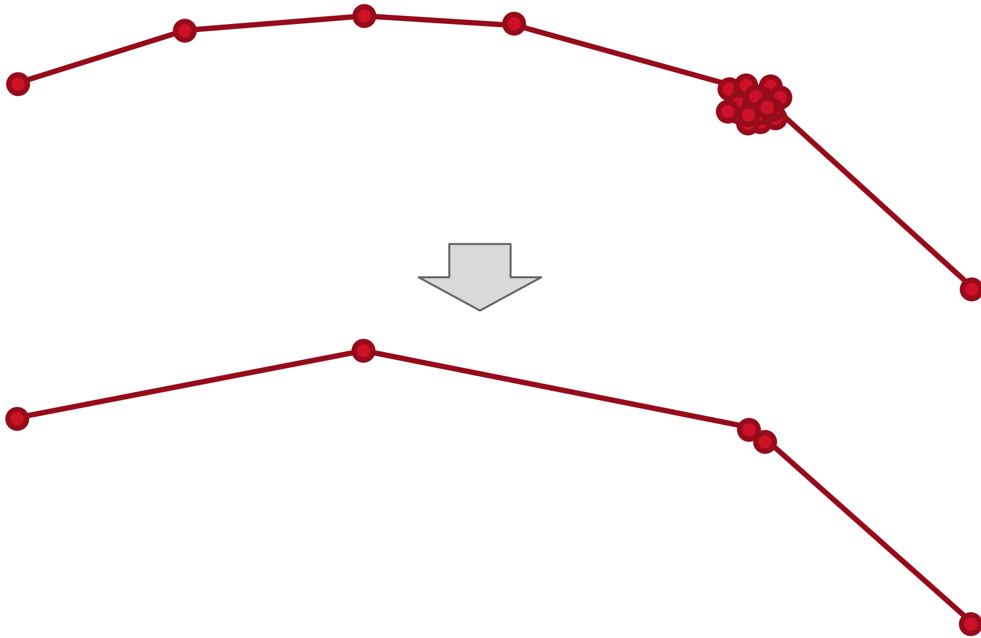
Trajectory compression / simplification

- Typical cases where points might be removed

Straight line movement



Negligible movement



Compression/simplification methods

Some standard methods for simplifying polygonal curves:

- Ramer–Douglas–Peucker, 1973
- Driemel–HarPeled–Wenk, 2010
- Imai–Iri, 1988

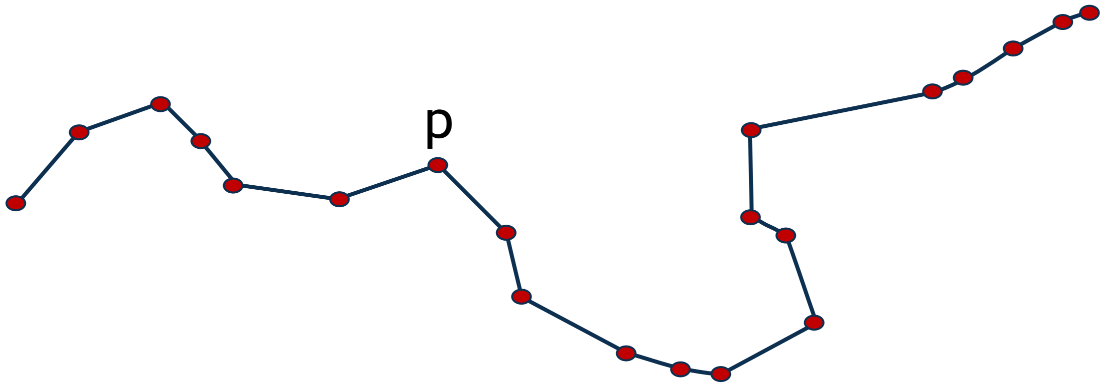
Ramer-Douglas-Peucker

1972 by Urs Ramer and 1973 by David Douglas and Thomas Peucker

- Mostly known as Douglas-Peucker (DP) algorithm

The most successful simplification algorithm. Used in GIS, geography, computer vision, pattern recognition…

Very easy to implement and works well in practice.



Ramer-Douglas-Peucker

Input polygonal path $P = \langle p_1,\ldots,p_n \rangle$ and threshold ε

Initially $i=1$ and $j=n$

Algorithm DP(P,i,j)

Find the vertex v_f between p_i and p_j farthest from p_ip_j.

dist := the distance between v_f and p_ip_j.

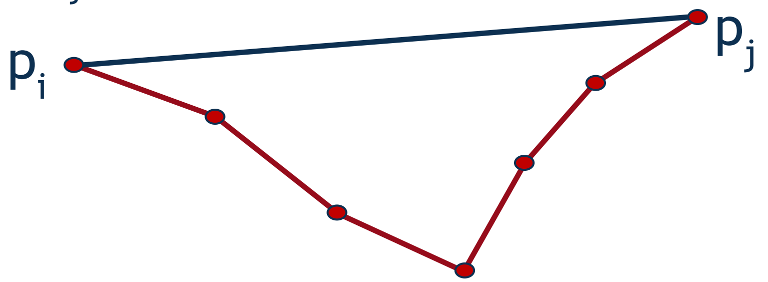
if dist > ε then

DP(P, v_i , v_f)

DP(P, v_f , v_j)

else

Output(v_iv_j)



Ramer-Douglas-Peucker

Input polygonal path $P = \langle p_1,\ldots,p_n \rangle$ and threshold ε

Initially $i=1$ and $j=n$

Algorithm DP(P,i,j)

Find the vertex p_f between p_i and p_j farthest from p_ip_j.

dist := the distance between p_f and p_ip_j.

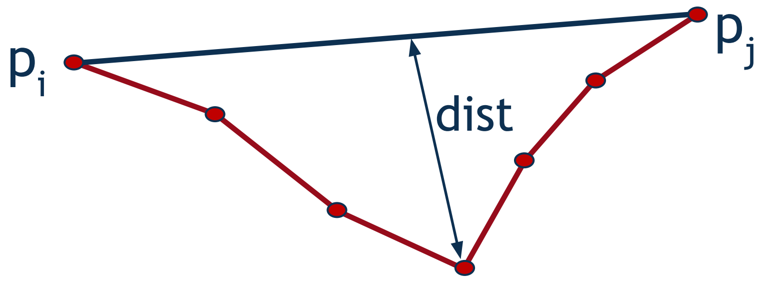
if dist > ε then

DP(P, p_i , p_f)

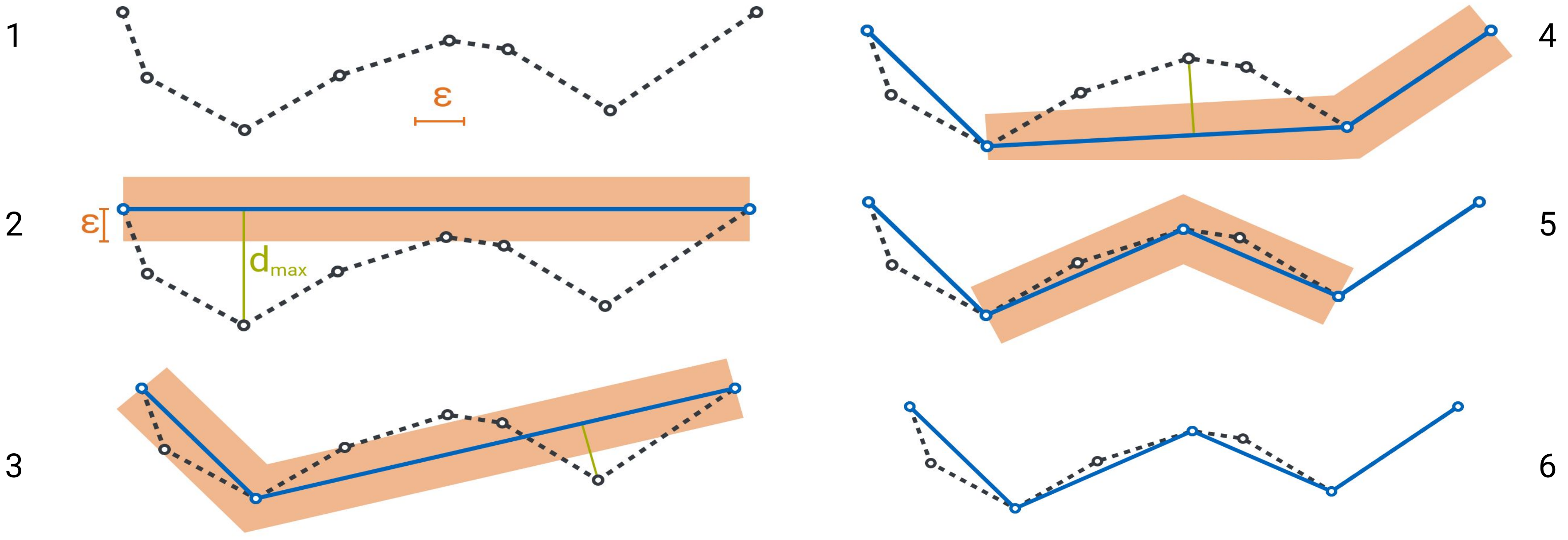
DP(P, p_f , p_j)

else

Output(p_ip_j)



Ramer-Douglas-Peucker



https://cartography-playground.gitlab.io/playgrounds/douglas-peucker-algorithm/

Ramer-Douglas-Peucker

Time complexity?

Testing a shortcut between p_i and p_j takes $O(j-i)$ time.

Worst-case recursion?

DP(P, v_i , v_{i+1})
DP(P, v_{i+1} , v_j)

Time complexity
$T(n) = O(n) + T(n-1) = O(n^2)$

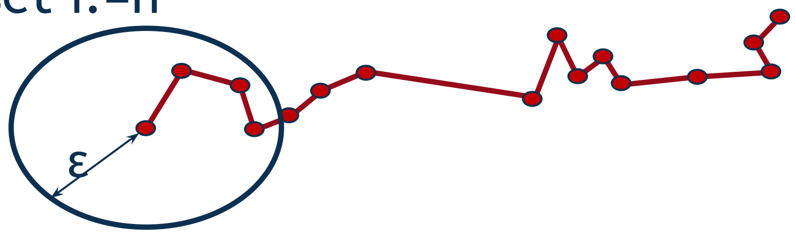
Algorithm DP(P,i,j)
  Find the vertex v_f farthest from p_ip_j.
  dist := the distance between v_f and p_ip_j.

  if dist > ε then
    DP(P, v_i , v_f)
    DP(P, v_f , v_j)
  else
    Output(v_iv_j)

Driemel et al.

Simple simplification($P = \langle p_1,\dots,p_n \rangle$, ε)

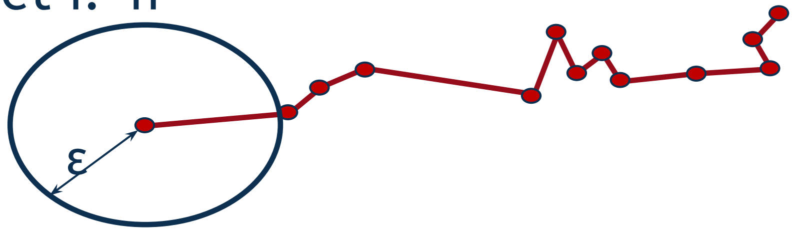
$P' := \langle p_1 \rangle$
$i:=1$
while $i<n$ do
    $q := p_i$
    $p_i :=$ first vertex p_i in $\langle q,\dots,p_n \rangle$ s.t. $|q-p_i| > \varepsilon$
    if no such vertex then set $i:=n$
    add p_i to P'
end
return P'



Driemel et al.

Simple simplification($P = \langle p_1,\ldots,p_n \rangle$, ε)

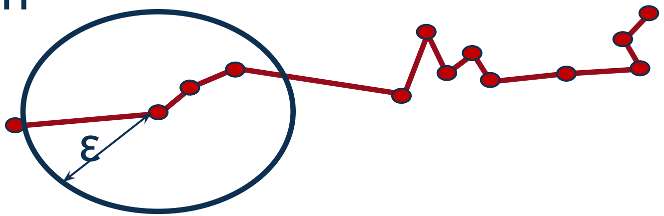
$P' := \langle p_1 \rangle$
$i := 1$
while $i<n$ do
    $q := p_i$
    $p_i :=$ first vertex p_i in $\langle q,\ldots,p_n \rangle$ s.t. $|q-p_i| > \varepsilon$
    if no such vertex then set $i:=n$
    add p_i to P'
end
return P'



Driemel et al.

Simple simplification($P = \langle p_1,\ldots,p_n \rangle$, ε)

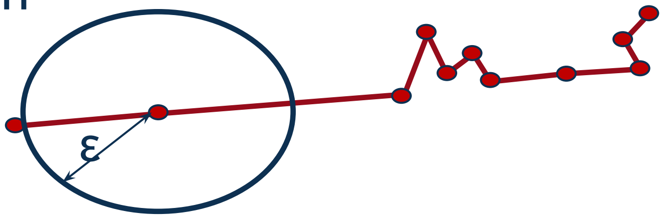
$P' := \langle p_1 \rangle$
$i:=1$
while $i<n$ do
    $q := p_i$
    $p_i :=$ first vertex p_i in $\langle q,\ldots,p_n \rangle$ s.t. $|q-p_i| > \varepsilon$
    if no such vertex then set $i:=n$
    add p_i to P'
end
return P'



Driemel et al.

Simple simplification$(P = \langle p_1,\ldots,p_n\rangle, \varepsilon)$

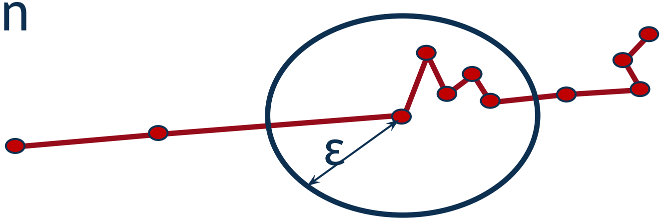
$P' := \langle p_1 \rangle$
$i:=1$
while $i<n$ do
  $q := p_i$
  $p_i :=$ first vertex p_i in $\langle q,\ldots,p_n\rangle$ s.t. $|q-p_i| > \varepsilon$
  if no such vertex then set $i:=n$
  add p_i to P'
end
return P'



Driemel et al.

Simple simplification$(P = \langle p_1,\ldots,p_n \rangle, \varepsilon)$

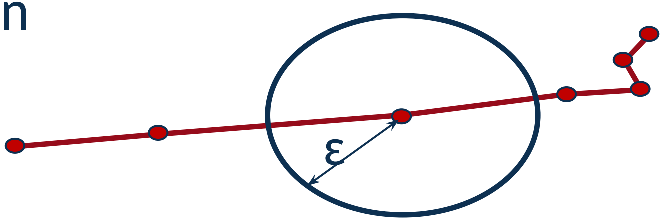
$P' := \langle p_1 \rangle$
$i := 1$
while $i<n$ do
    $q := p_i$
    $p_i :=$ first vertex p_i in $\langle q,\ldots,p_n \rangle$ s.t. $|q-p_i| > \varepsilon$
    if no such vertex then set $i:=n$
    add p_i to P'
end
return P'



Driemel et al.

Simple simplification($P = \langle p_1,\ldots,p_n \rangle$, ε)

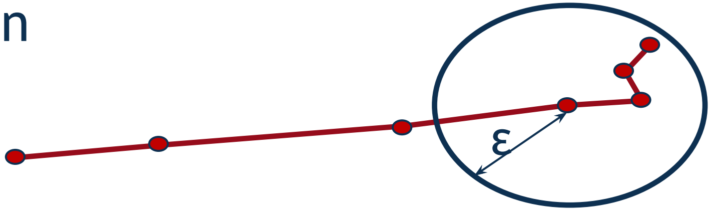
$P' := \langle p_1 \rangle$
$i:=1$
while $i<n$ do
    $q := p_i$
    p_i := first vertex p_i in $\langle q,\ldots,p_n \rangle$ s.t. $|q-p_i| > \varepsilon$
    if no such vertex then set $i:=n$
    add p_i to P'
end
return P'



Driemel et al.

Simple simplification$(P = \langle p_1,\ldots,p_n \rangle, \varepsilon)$

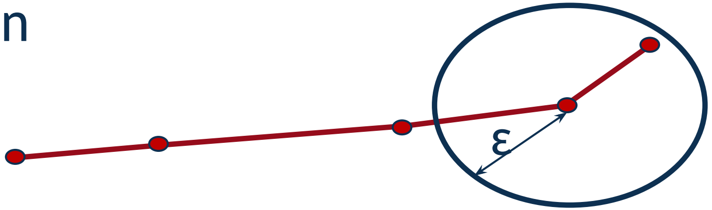
$P' := \langle p_1 \rangle$
$i := 1$
while $i < n$ do
    $q := p_i$
    $p_i :=$ first vertex p_i in $\langle q,\ldots,p_n \rangle$ s.t. $|q-p_i| > \varepsilon$
    if no such vertex then set $i := n$
    add p_i to P'
end
return P'



Driemel et al.

Simple simplification($P = \langle p_1,\ldots,p_n\rangle, \varepsilon$)

$P' := \langle p_1 \rangle$
$i := 1$
while $i<n$ do
    $q := p_i$
    $p_i :=$ first vertex p_i in $\langle q,\ldots,p_n\rangle$ s.t. $|q-p_i| > \varepsilon$
    if no such vertex then set $i:=n$
    add p_i to P'
end
return P'



Driemel et al.

Simple simplification$(P = \langle p_1,\ldots,p_n \rangle, \varepsilon)$

$P' := \langle p_1 \rangle$

$i := 1$

while $i<n$ do

  $q := p_i$

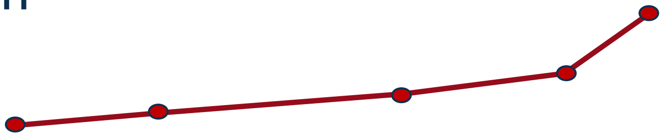
  $p_i :=$ first vertex p_i in $\langle q,\ldots,p_n \rangle$ s.t. $|q-p_i| > \varepsilon$

  if no such vertex then set $i:=n$

  add p_i to P'

end

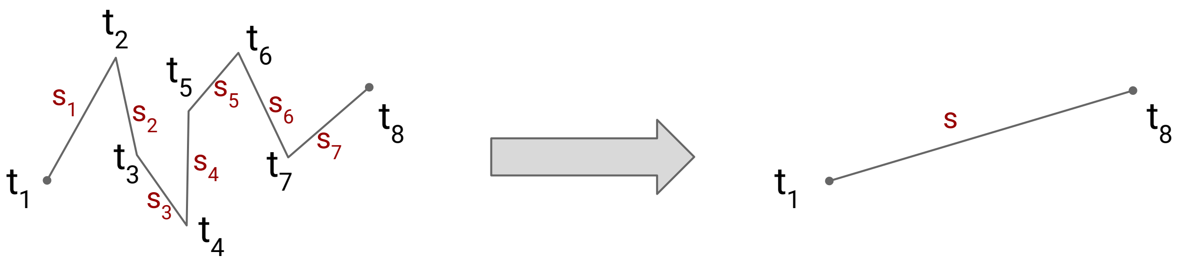
return P'



Impact on speed

- What about time and speeds?
  - Time-stamps were never considered in the algorithms
  - They considered on impact on space / geometry of trajectories
  - What impact on time-related aspects, e.g. speed?

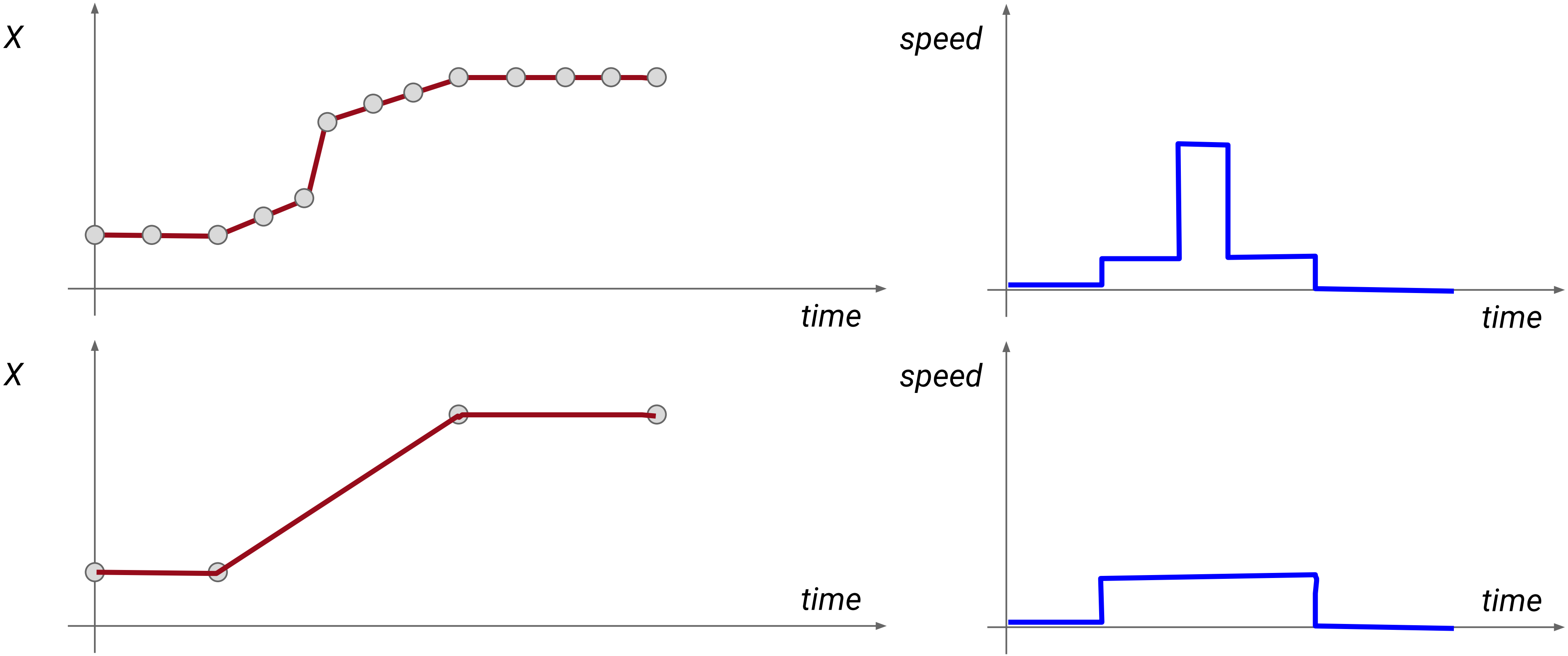
- Typically, simplification reduces average speed estimates:



Avg speed = $(s_1 + \ldots + s_7) / (t_8 - t_1)$ **>** Avg speed = $s / (t_8 - t_1)$

Impact on speed

- Flattening effect



INTERVALLO

How fast is a cow?

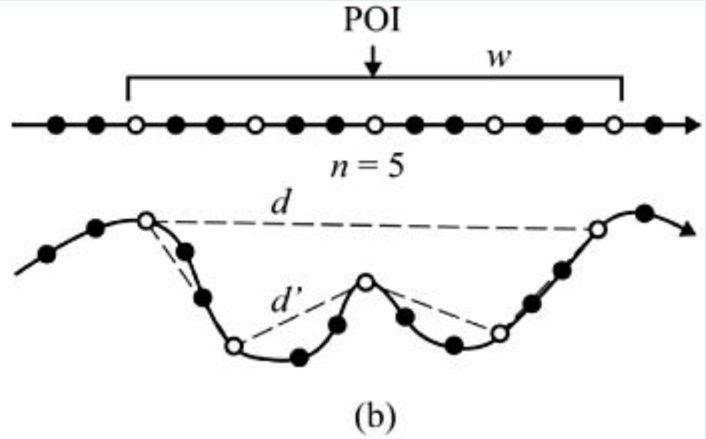
How fast is a cow?

- Trajectory compression / simplification changes the scale of the analysis
  - Simplified data → macroscopic analysis
  - Detailed data → microscopic analysis
- Several movement characteristics can be affected

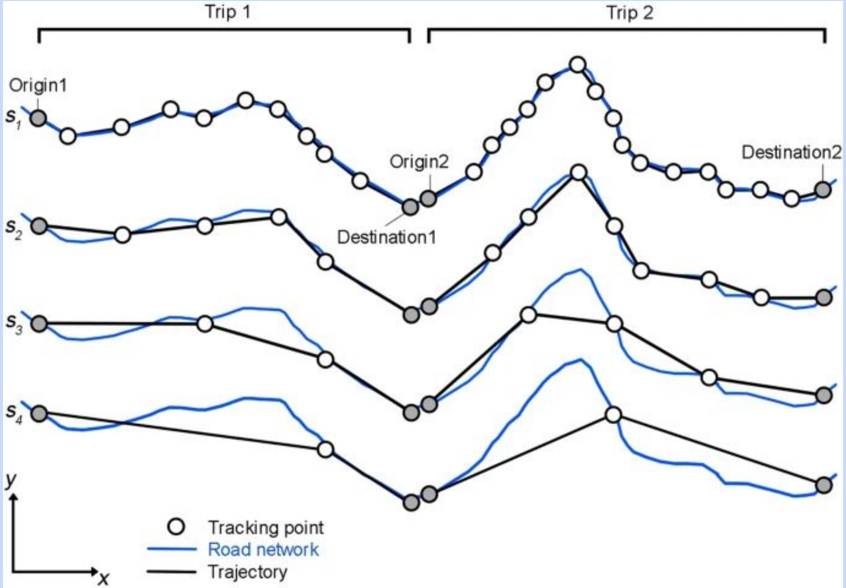
INTERVALLO

How fast is a cow?

How fast is a cow? Cross-Scale Analysis of Movement Data
Laube P, Purves RS (2011)



Understanding the impact of temporal scale on human movement analytics
Su, R., Dodge, S. & Goulias, K.G (2022)

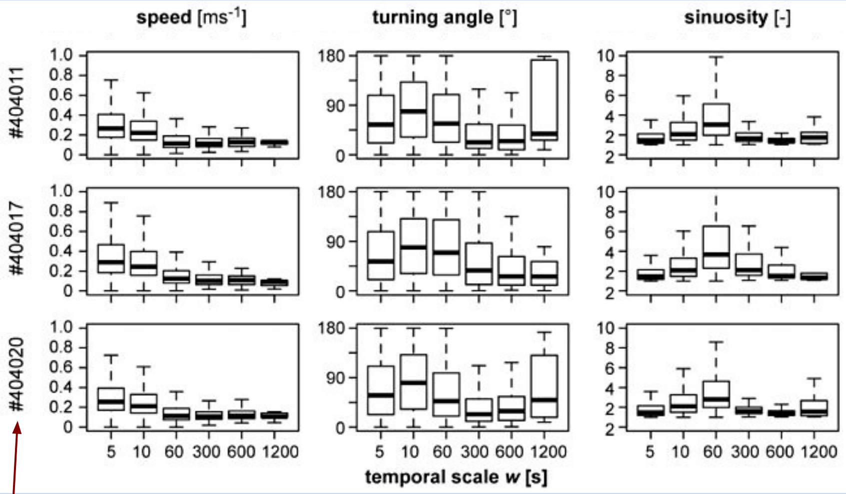


INTERVALLO

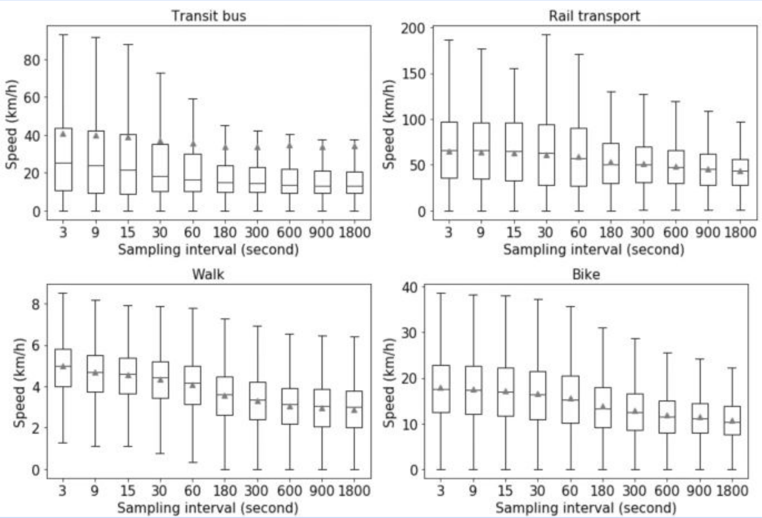
How fast is a cow?

How fast is a cow? Cross-Scale Analysis of Movement Data
Laube P, Purves RS (2011)

Understanding the impact of temporal scale on human movement analytics
Su, R., Dodge, S. & Goulias, K.G (2022)

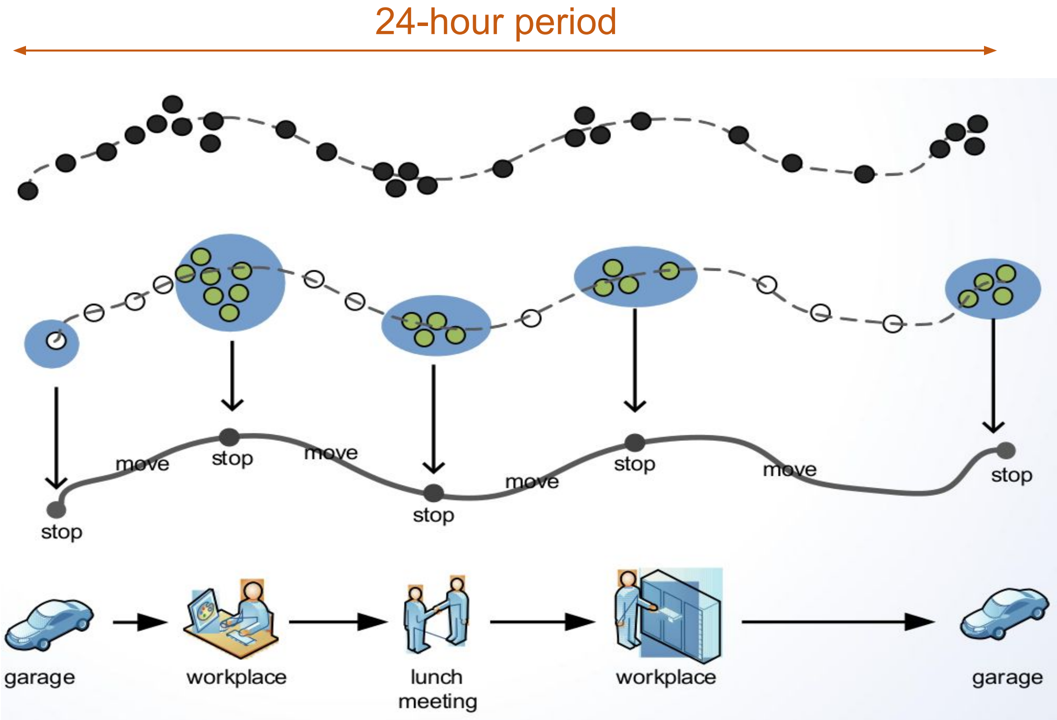


Cow ID



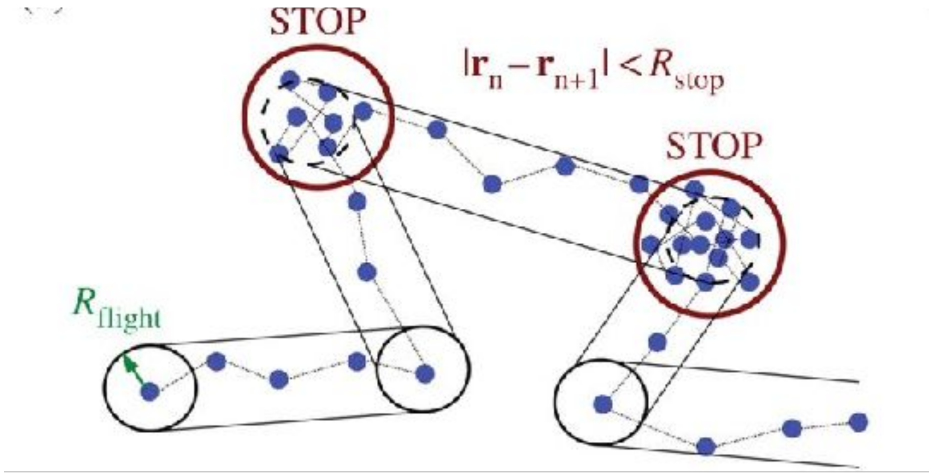
Stop detection & Trajectory segmentation

- Raw data forms a continuous stream of points
- Typical unit of analysis: the trip
- How to segment?
  - Basic idea: identify stops



Stop detection & Trajectory segmentation

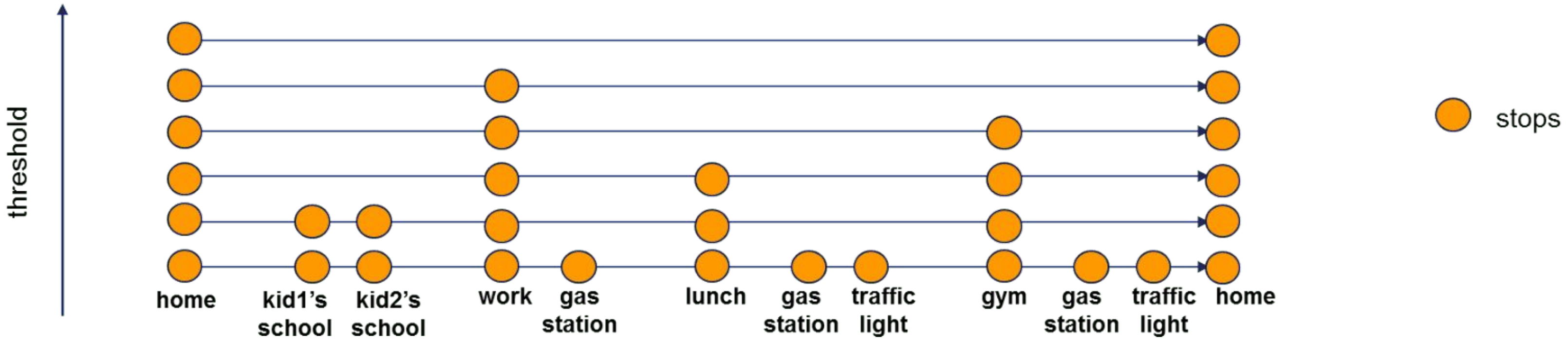
- General criteria based on speed
  - If it **moves very little** (threshold Th_S) over a significant **time interval** (threshold Th_T)
    => it is practically a stop
  - Trajectory (trip) = contiguous sequence of points between two stops



- Typical values:
  - Th_S within [50, 250] meters
  - Th_T within [1, 20] minutes

Stop detection & Trajectory segmentation

- Different time thresholds yield different semantics

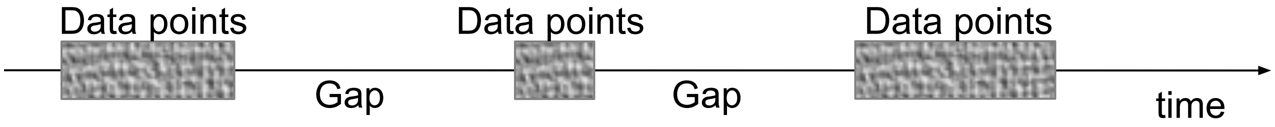


- Which one is the best for you?
  - Application dependent

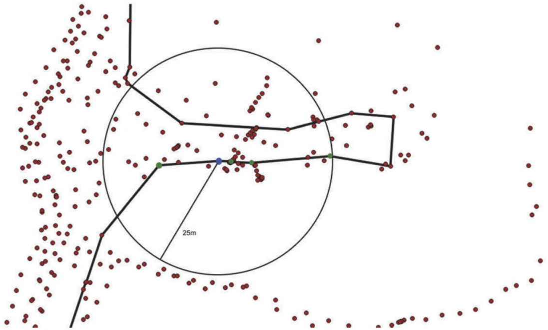
Stop detection & Trajectory segmentation

- Special cases, easier to treat
  - Stop explicitly in the data: e.g. engine status on/off
    - Simply "cut" trajectories on status transitions

  - Device is off during stops:
    - Typical of cars data
    - A stop results in a time gap in the data
    - Exceptions: short stops might remain undetected

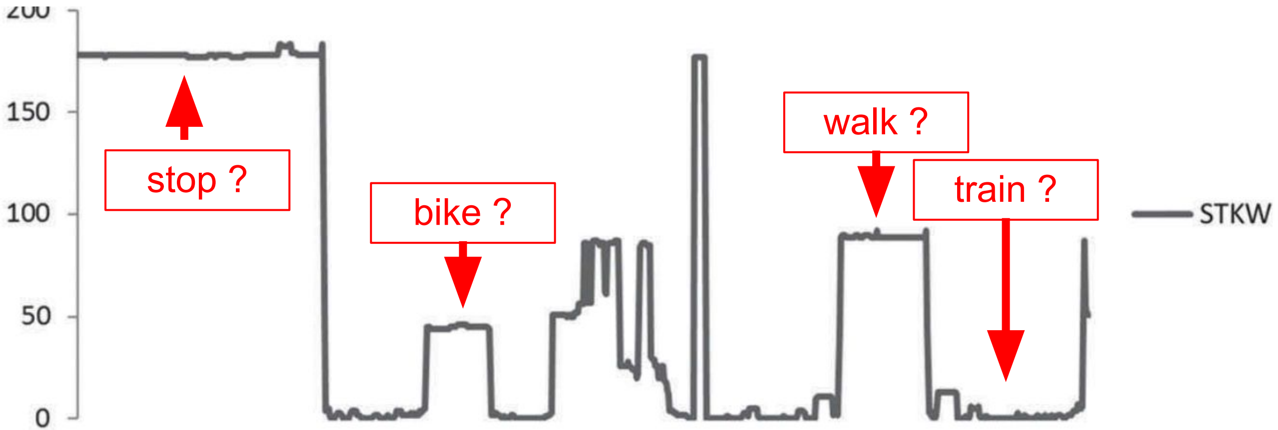


Generalization: transportation means segmentation



- Speed / density-based approach
- Idea: faster means less of my points around me

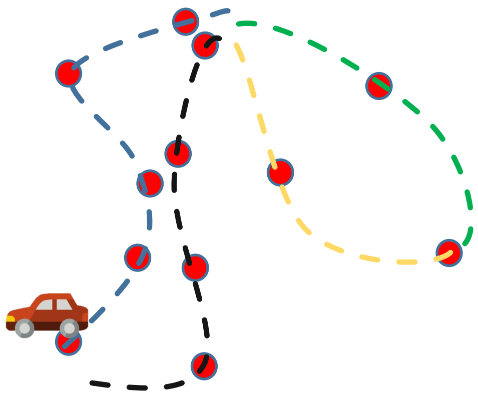
Number of points within radius R



Katarzyna Siła-Nowicka, Jan Vandrol, Taylor Oshan, Jed A. Long, Urška Demšar & A. Stewart Fotheringham (2016) Analysis of human mobility patterns from GPS trajectories and contextual information, International Journal of Geographical Information Science, 30:5, 881-906

User's Mobility History

- What do we get after segmentation?
- Several trajectories associated to the same subject
- Enables individual-level analyses
    - E.g. explore user's habits, find deviations from usual, etc.



Inferring Home / Work locations

- Take all trips of a vehicle / user
- Build a "Individual Mobility Network"
  - Graph abstraction of the overall mobility based on locations (nodes) and movements (edges).

Individual Mobility Network

- Focus on start and stop points
  - Dense areas represent important places

Individual Mobility Network

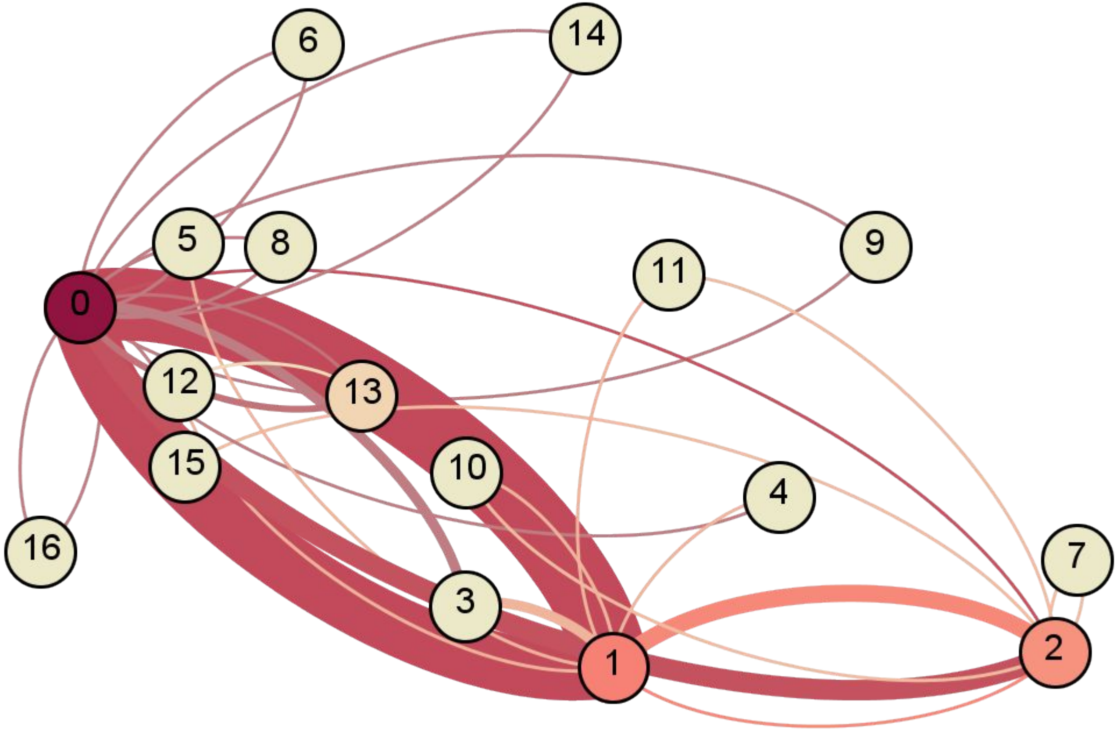
- Cluster points to identify locations

Individual Mobility Network

- Each location is characterized by its frequency

Individual Mobility Network

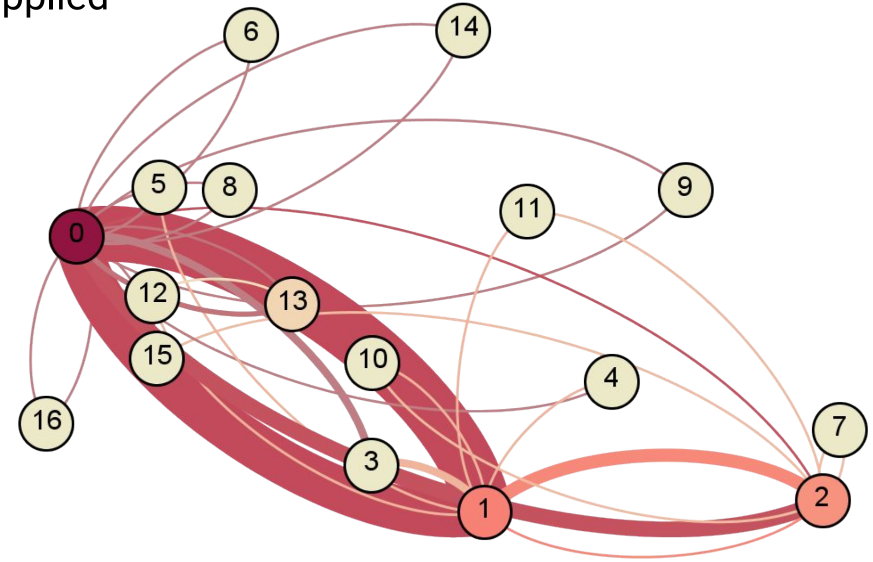
- Trips between points area aggregated as edges between nodes/locations



Inferring Home / Work locations

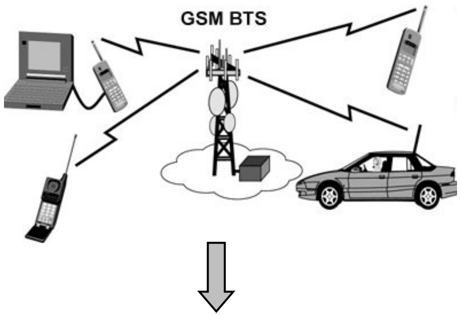
- Basic approach is based on frequency only
  - Most frequent location (L0) := Home
  - Second most frequent location (L1) := Work
    - A minimum frequency threshold is applied

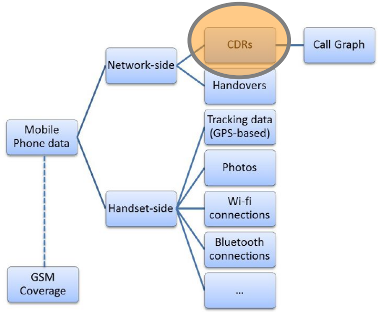
- Various alternatives & refinement are possible
  - Check time of stop & stay duration
    - Home: stop at 20-22, stay 8-10 hrs
    - Work: stop at 7-10, stay 6-9 hrs



Inferring Home / Work locations with Phone Data
Tha case of GSM traces

Data gathered from mobile phone operator for billing purpose





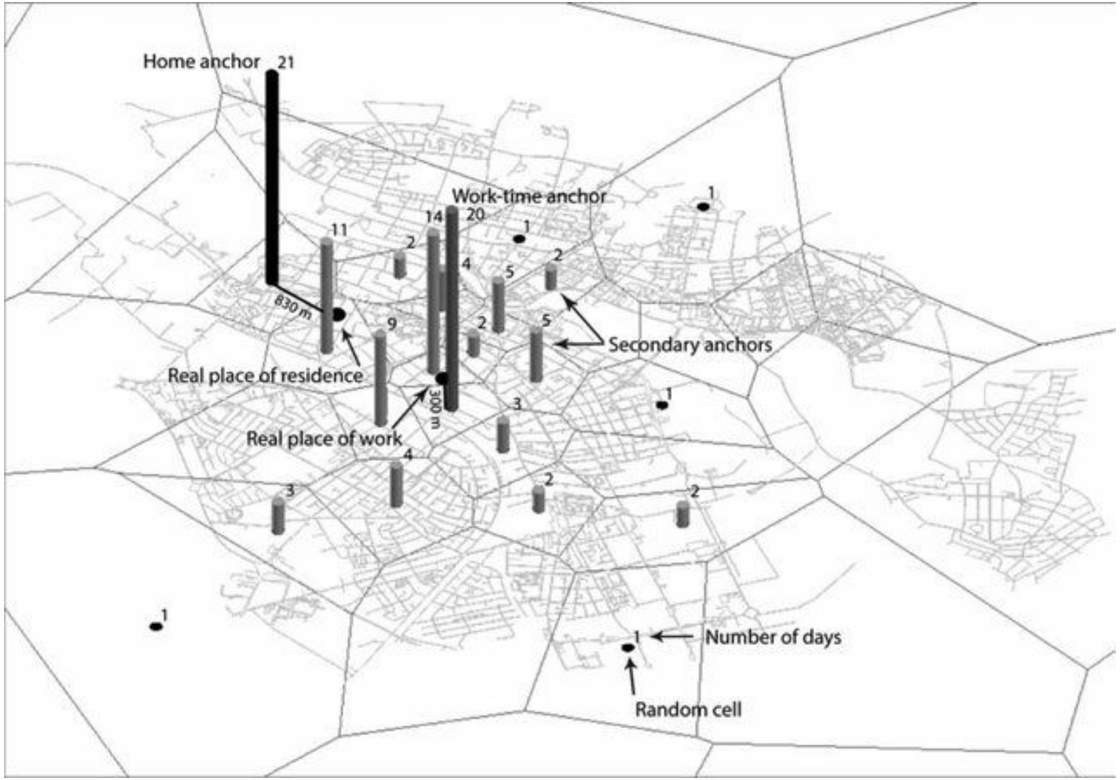
User id	Time start	Cell start	Cell end	Duration
10294595	"2014-02-20 14:24:58"	"PI010U2"	"PI010U1"	48
10294595	"2014-02-20 18:50:22"	"PI002G1"	"PI010U2"	78
10294595	"2014-02-21 09:19:51"	"PI080G1"	"PI016G1"	357

Inferring Home / Work locations with Phone Data
Tha case of GSM traces - 1

- "**Personal Anchor Points**": high-frequency visited places of a user
  - Select top 2 cells with max number of days with calls
  - Determine home and work through time constraints:
    - Based on average start time (AST) of calls and its deviation (std)
    - IF AST<17:00 & std<0.175 ⇒ WORK
    - ELSE HOME

Inferring Home / Work locations with Phone Data
Tha case of GSM traces - 1

- “Personal Anchor Points”



AHAS, R., SILM, S., JARV, O., SALUVEER, E., AND TIRU, M. 2010. *Using mobile positioning data to model locations meaningful to users of mobile phones*. Journal of Urban Technology 17, 1, 3–27.

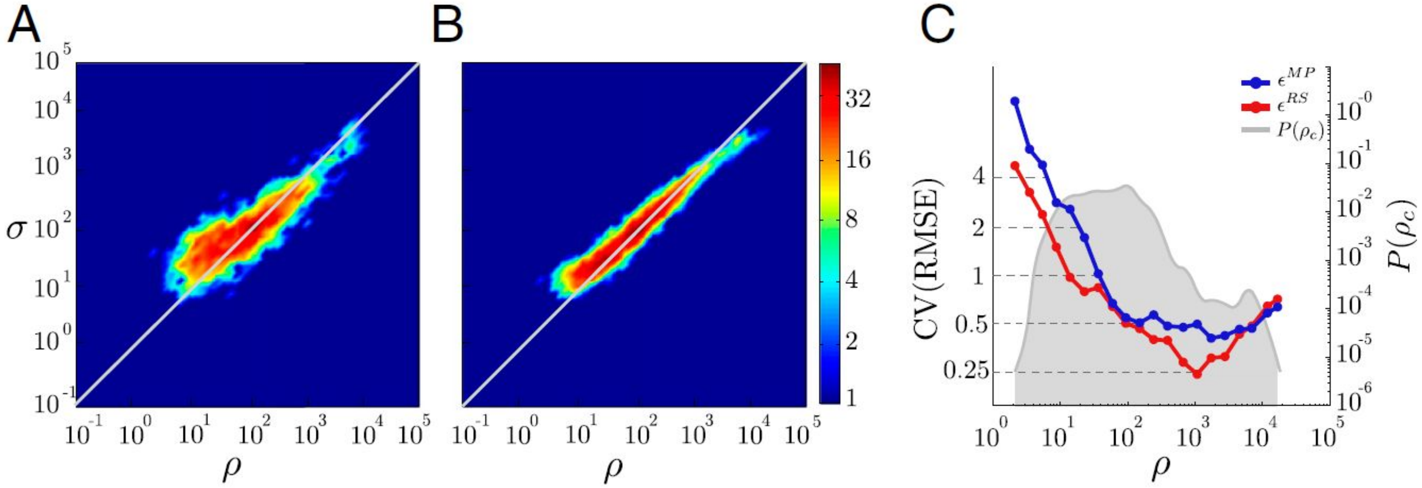
Inferring Home / Work locations with Phone Data
Tha case of GSM traces - 2

- Estimating users' **residence through night activity**
  - Home = region with highest frequency of calls during nighttime
  - More suitable for larger scales
    - E.g. region = municipality

Pierre Deville et al.
Dynamic population mapping using mobile phone data.
PNAS vol. 111 no. 45, pp. 15888–15893, doi: 10.1073/pnas.1408439111

Inferring Home / Work locations with Phone Data
Tha case of GSM traces - 2

- Sample results on national level (France)
  - estimate resident density (ρ) vs. real one (σ)



A = GSM data estimates B = Environment/Infrastructures-based estimates

Activity labelling / recognition

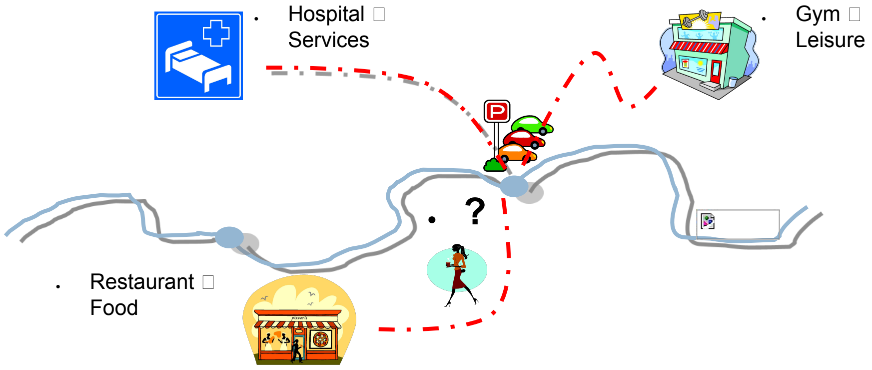
Objective: adding information to points / locations

Two main ways:

- Assign a single activity
- Assign a distribution of POIs / activity types

Activity labelling / recognition

Given a dataset of GPS tracks of private vehicles, annotate trajectories with the most probable activities performed by the user.

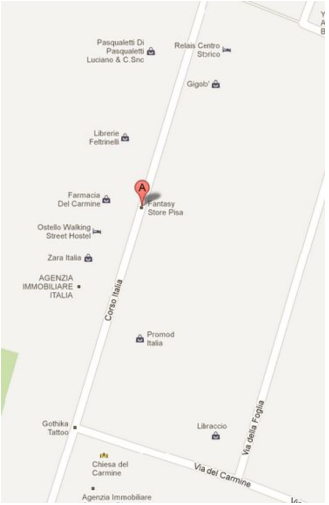


Associates the list of possible POIs (with corresponding probabilities) visited by a user moving by car when he stops.

A mapping between POIs categories and Transportation Engineering activities is necessary.

Activity labelling / recognition

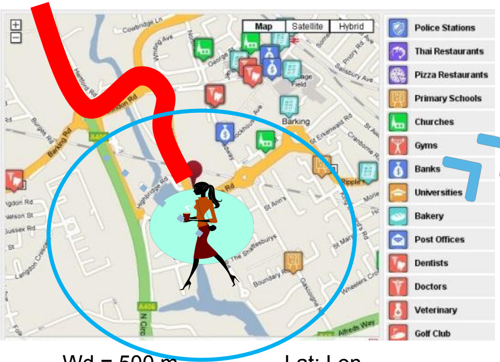
- **POI collection** from available sources, e.g. from Google Places.
- **Association POI – Activity**: Each POI is associated to a ``activity". For example Restaurant → Eating/Food, Library → Education, etc.
- **Basic elements/characteristics:**
  - C(POI) = {category, opening hour, location}
  - C(Trajectory) = {stop duration, stop location, time of the day}
  - C(User) = {max walking distance}
- **Computation of the probability to visit a POI/ to make an activity:** For each POI, the probability of ``being visited" is a function of the POI, the trajectory and the user features.
- **Annotated trajectory**: The list of possible activities is then associated to a Stop based on the corresponding probability of visiting POIs



Activity labelling / recognition

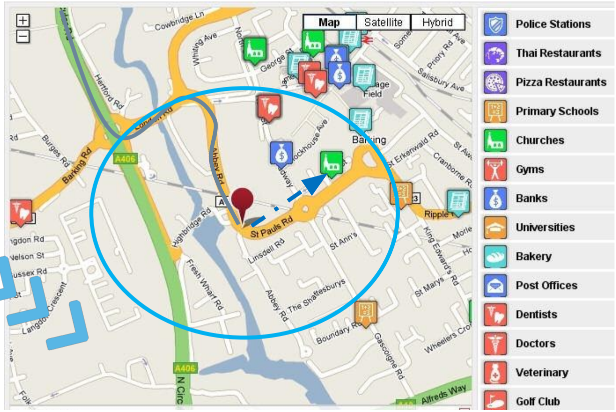
POI assignment

Filter POIs based on time constraints



Wd = 500 m

- Lat; Lon
- TimeStamp: Sun 10:55 – 12:05



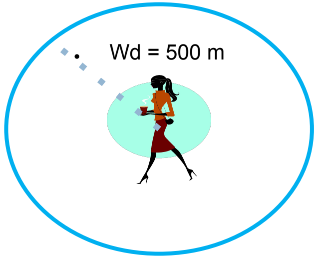
- Bank: Mon – Fri [8:00 – 15:30]
- Dentist: Mon – Sat [9:00 – 13:00] [15:30 – 18:00]
- Church: Mon – Sat [18:00 – 19:00]
- Sun [11:00 – 12:00]
- Primary School: Mon – Sat [8:00 – 13:00]

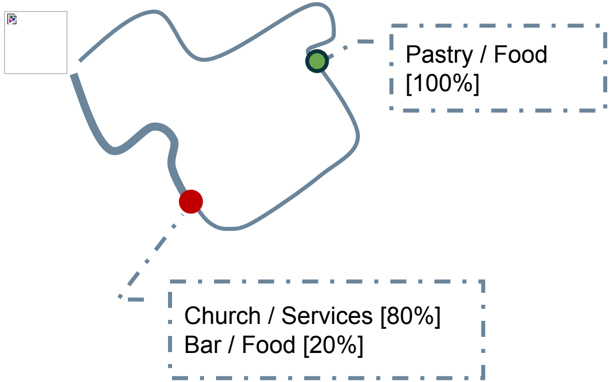
Activity labelling / recognition

What if multiple POIs match

- Select closest one

- Assign a distribution of probability:

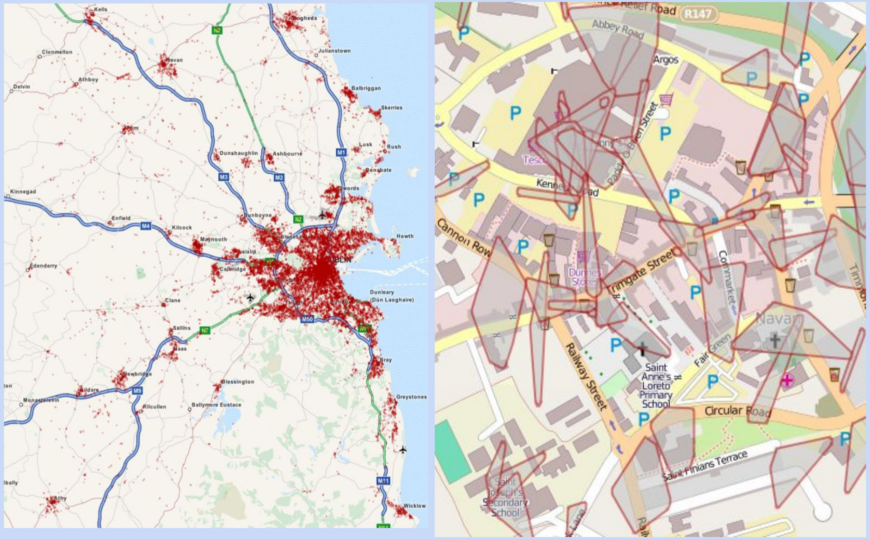




Reading social media to find POIs

An Irish experiment on Twitter

The points of each trajectory taken separately were grouped into spatial clusters of maximal radius 150m. For groups with at least 5 points, convex hulls have been built and spatial buffers of small width (5m) around them.
1,461,582 points belong to the clusters (89% of 1,637,346); 24,935 personal places have been extracted.



Examples of extracted places

Statistical distribution of the number of places per person

INTERVALLO

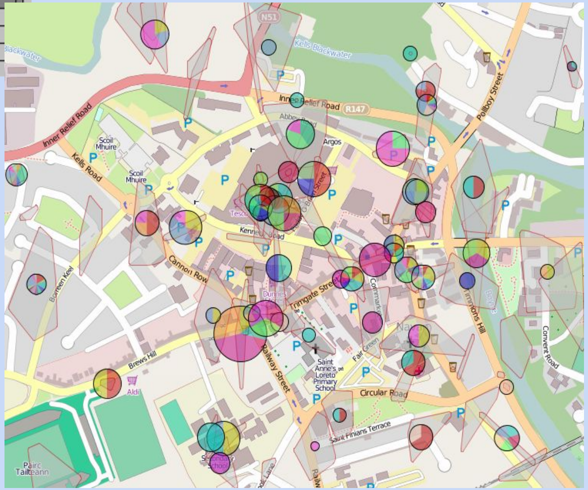
Reading social media to find POIs

Topics have been assigned to 208,391 messages (14.3% of the 1,461,582 points belonging to the personal places)

Message	Features	topic=family: Occurrences of topic	topic=home: Occurrences of topic	topic=education: Occurrences of topic	topic=work: Occurrences of topic
@joe_lennon I usually	education	0	0	1	0
@joe_lennon together	education	0	0	1	0
@jas_103 deadly; don	work	0	0	0	1
Just got home and see	home	0	1	0	0
So excited about my ne	sweets	0	0	0	0
@lamtcdizzy I haven't b	shopping	0	0	0	0
Get in from my night ou	family;home;work	1	1	0	1
Home again at 6pm! N	home	0	1	0	
Bussing it home for th			1	0	
Ah shite. It's been a p				0	
@ronanhutchinson be	education	0	0	1	

Get in from my night out; my dad gets home from work two minutes later. Great timing :)

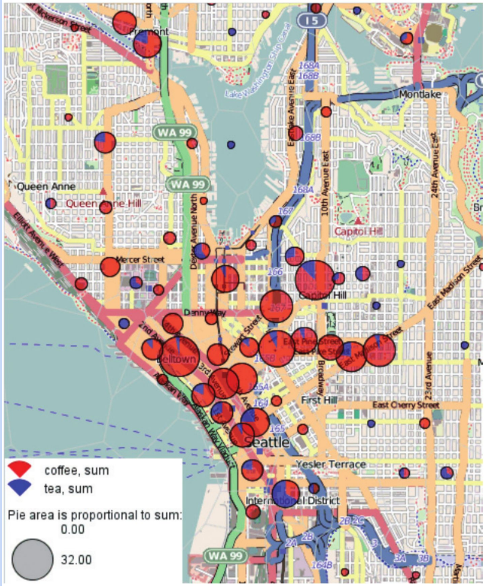
...

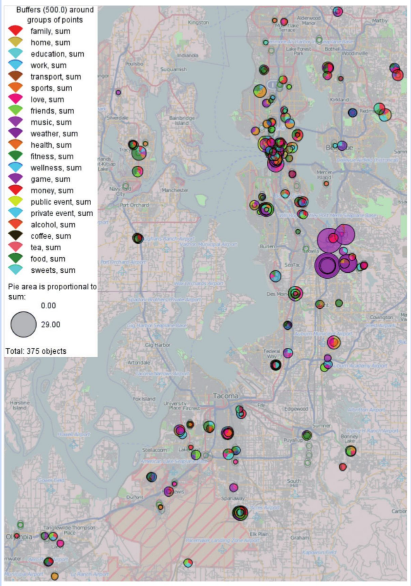


1) Some places did not get topic summaries (about 20% of the places)
2) In many places the topics are very much mixed
3) The topics are not necessarily representative of the place type (e.g., topics near a supermarket: family, education, work, cafe, shopping, services, health care, friends, game, private event, food, sweets, coffee)

INTERVALLO

In the meanwhile, in Seattle…





G. Andrienko et al. Thematic Patterns in Georeferenced Tweets through Space-Time Visual Analytics. Computing in Science & Engineering, 2013.

Homeworks

Homework 4.1

How fast are users?

Choose one of the datasets seen at lesson (taxis, Geolife, etc.), select at least 10 users/vehicles and compute distributions of lengths. Remove 10% of points in each trajectory and repeat the distribution. Do the same for 20%, 30%, … 90%. How does length distribution change?

- Prepare a (well commented) python notebook

Homework 4.2

Implement a “speed-aware” trajectory compression method, that preserves speed, and test it on a dataset of your choice, e.g. a subset of taxis or Geolife users.

- Show the effects of simplification on some sample trajectories
- Study how the lengths of trajectories are affected
- Prepare a (well commented) python notebook

Homework 4.3

Inferring Home locations is often used to estimate the resident population of geographical areas. What are the existing approaches to face the problem?

- Make a research on Internet on the methods, including big data-based ones (GPS, GSM data, maybe satellite data or others) but also any other approach – e.g. coming from statistics/demography, sociology, etc.
- Prepare a blog (basically a survey) summarizing your discoveries.

Homework 4.4

Estimating GPS errors. Choose a bounding rectangle covering SF city. Download the road network/graph of that area. Select the GPS points of taxis in the same area. Assign each point P to its closest road segment R. Define pseudo-error(P) as the distance dist(P,R).

- Analyze the overall distribution of the pseudo-errors. Is it coherent with GPS.gov estimates of errors?
- Are pseudo-errors the same downtown vs. out of city?
- Prepare a (well commented) python notebook

Homework 4.5

The typical filtering algorithm relies on the individual. Define this new algorithm: label the points which have less than X neighbohrs in the set of points of all trips within an y radius as outliers, and discard them.

- Test it on SF taxi data
- Prepare a (well commented) python notebook