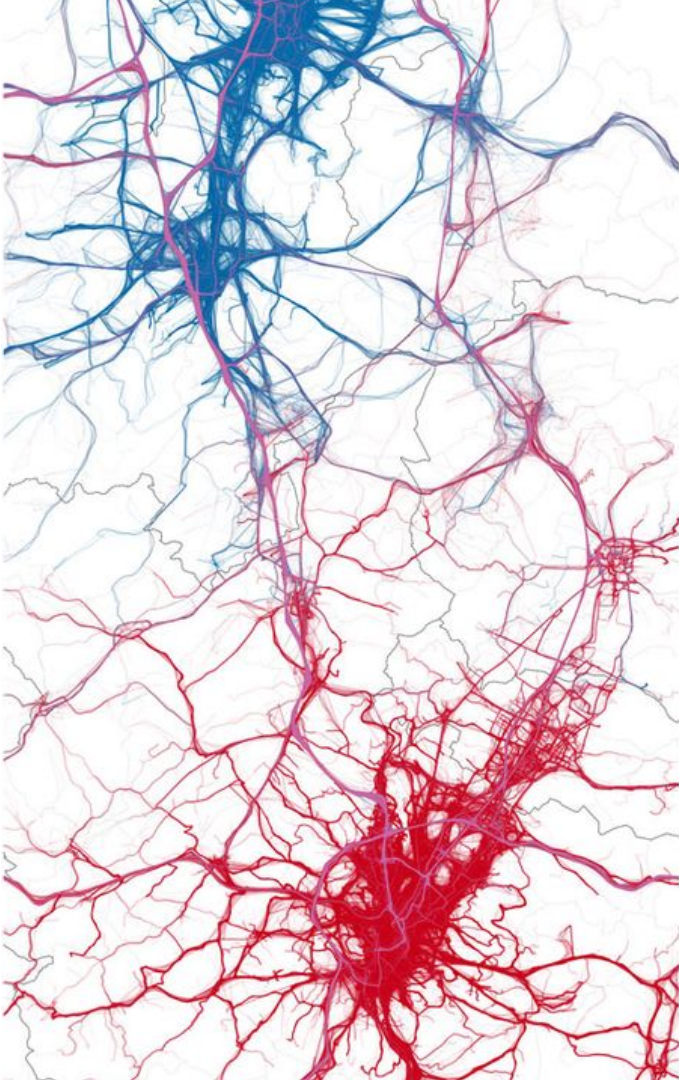




(Pre)processing Mobility Data



Consiglio Nazionale
delle Ricerche

Content of this lesson / 1

- **Processing high-resolution trajectories (GPS-like)**
 - trajectory filtering
 - point map matching
 - route reconstruction
 - trajectory compression
 - Semantic enrichment
 - stop detection / trajectory segmentation
 - home location detection & Individual Mobility Network
 - activity recognition (POI-based)

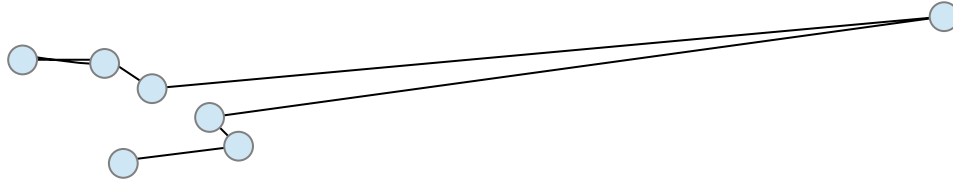
Content of this lesson / 2

- **Processing mobile phone data**
 - adapting high-res methods
 - home location detection (GPS & MobPhones)
 - ping-pong effect

Trajectory filtering

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

What is the real length of this trip?

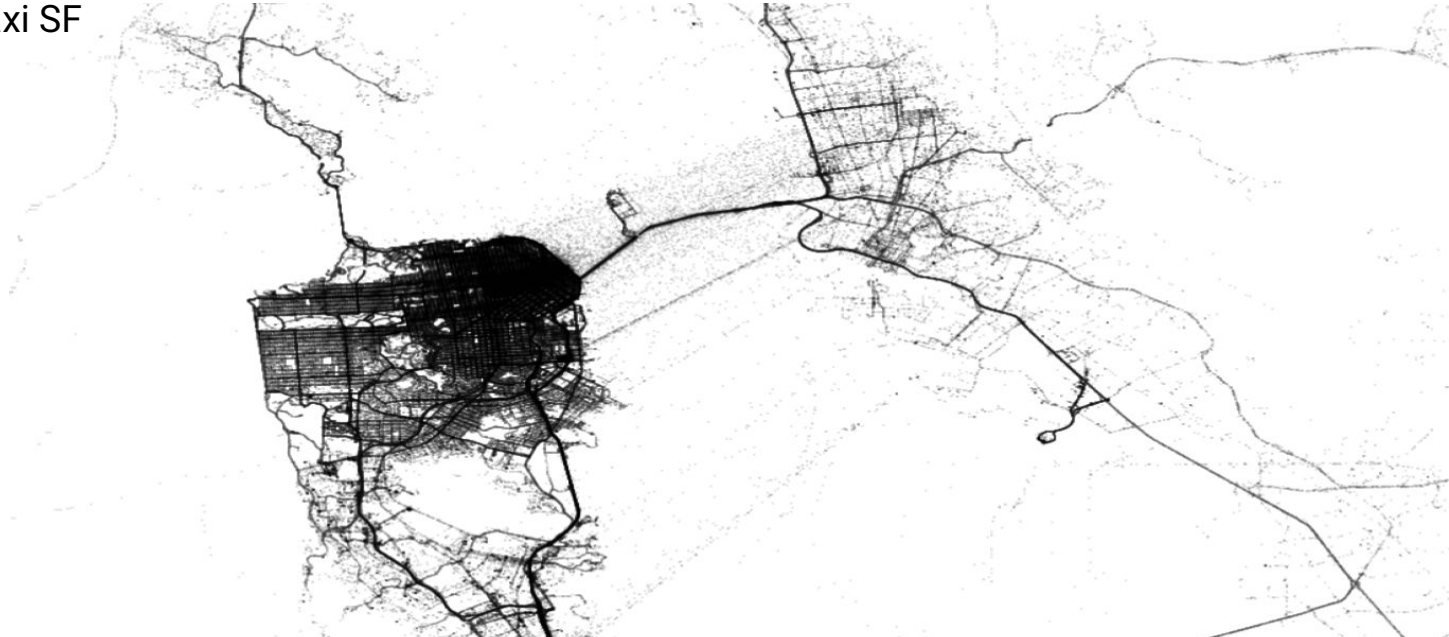


- Two families of approaches:
 - Context-based filtering
 - Movement-based filtering

Context-based filtering

- Single points might contain errors of various kinds

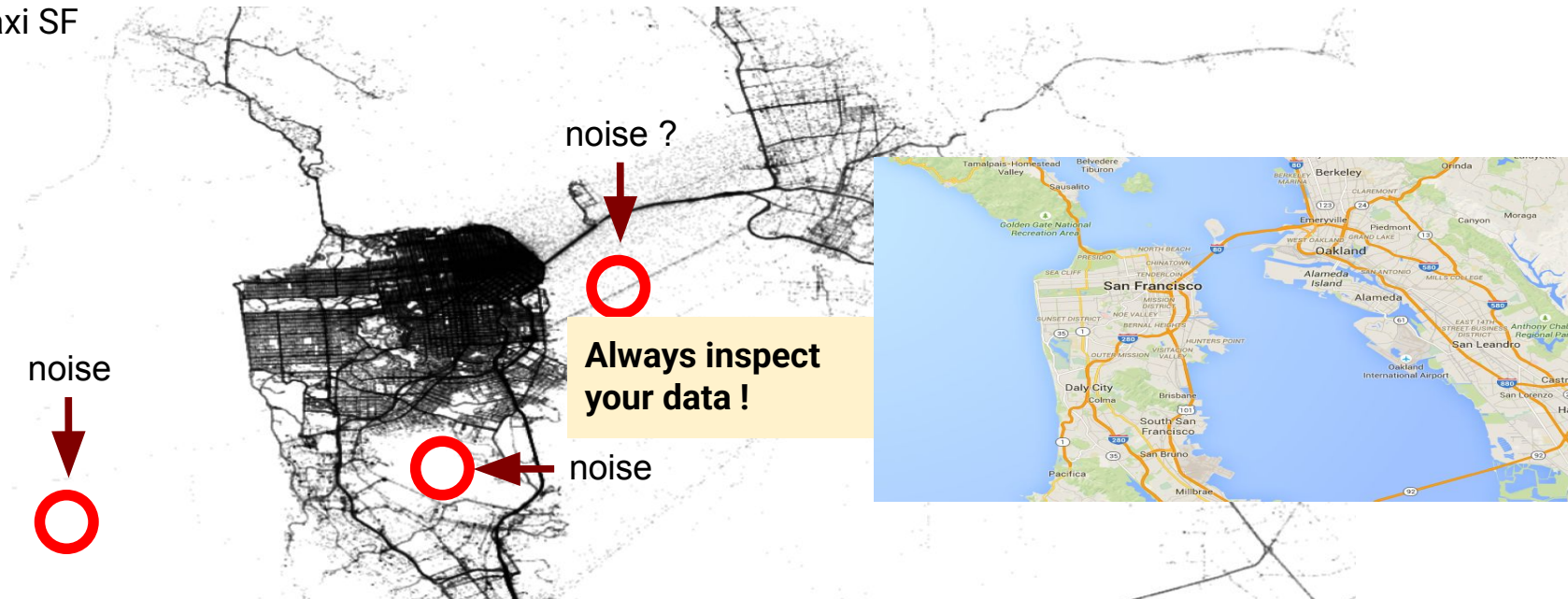
dataset: Taxi SF



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?

dataset: Taxi SF



Context-based filtering

Input: Trajectory $T = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n)$, Road network R , Distance threshold δ

Output: Filtered trajectory T'

```
1  $T' \leftarrow ( )$ ;  
2 foreach point  $\mathbf{p}$  in  $T$  do  
3    $d \leftarrow \text{min\_distance}(\mathbf{p}, R)$ ;    // Dist. from closest road  
4   if  $d \leq \delta$  then  
5     Append  $\mathbf{p}$  to  $T'$ ;  
6 return  $T'$ ;
```

Context-based filtering

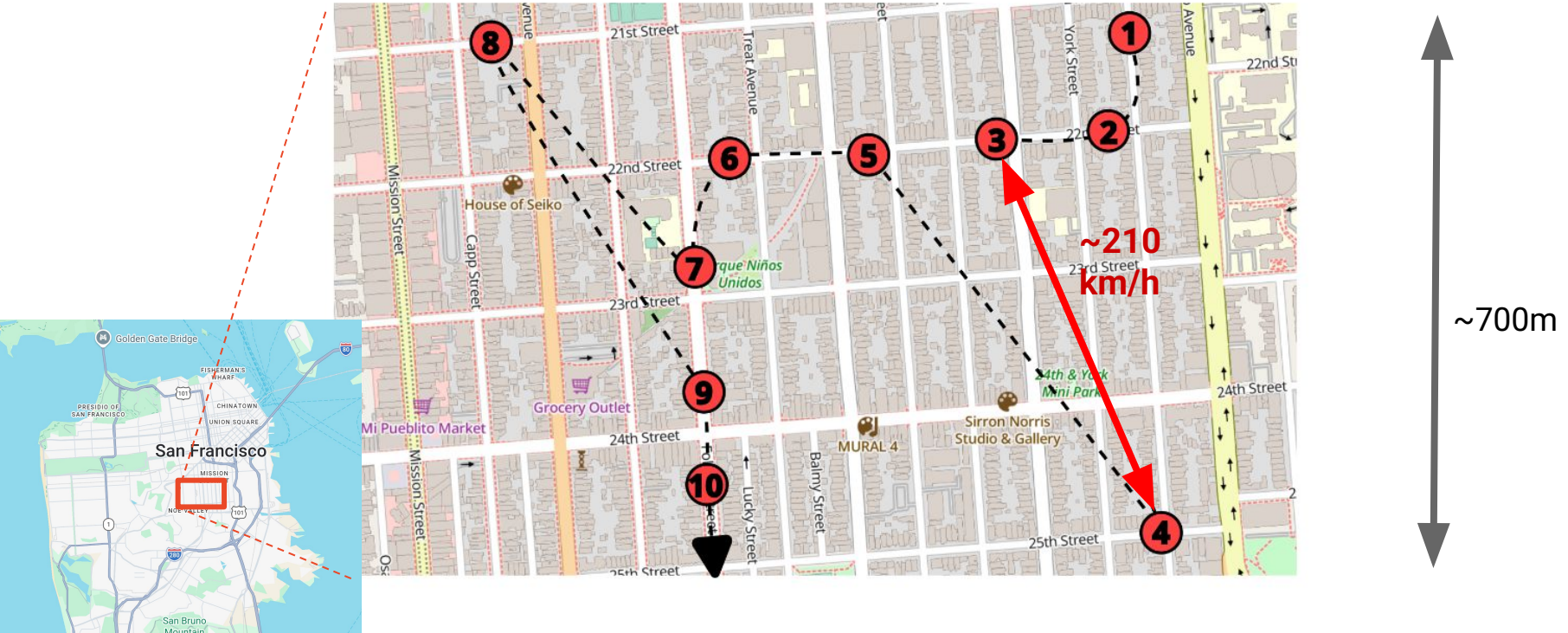
- Why looking at the context might be not enough



Movement-based filtering

- Why looking at the context might be not enough

Sampling period:
1pt every 10secs

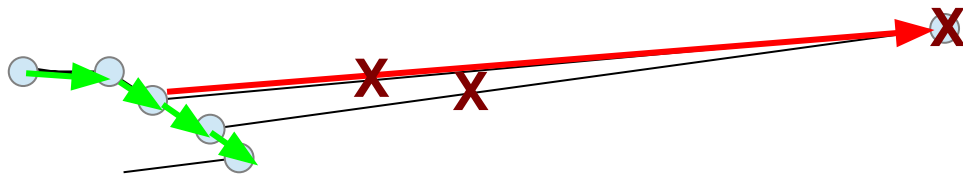


Movement-based filtering

Input: Trajectory $T = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n)$, Maximum speed σ

Output: Filtered trajectory T'

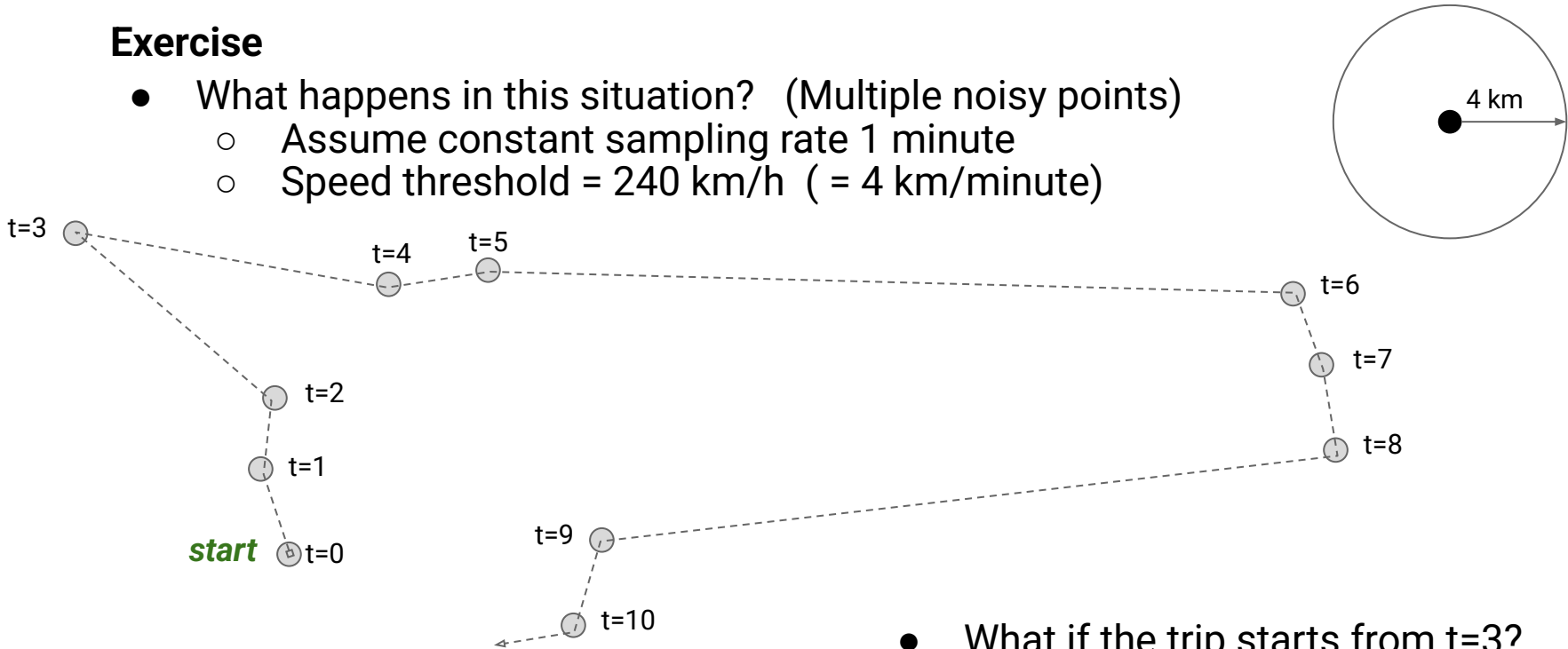
```
1  $T' \leftarrow (\mathbf{p}_1)$ ;  
2  $\mathbf{p}_{last} \leftarrow \mathbf{p}_1$ ;  
3 for  $i = 2, \dots, n$  do  
4    $d \leftarrow \text{distance}(\mathbf{p}_{last}, \mathbf{p}_i)$ ;           // Euclidean or Haversine  
5    $t \leftarrow \text{time\_diff}(\mathbf{p}_{last}, \mathbf{p}_i)$ ;  
6   if  $d/t \leq \sigma$  then                       // Check avg. speed  
7     Append  $\mathbf{p}_i$  to  $T'$ ;  
8      $\mathbf{p}_{last} \leftarrow \mathbf{p}_i$ ;  
9 return  $T'$ ;
```



Movement-based filtering

Exercise

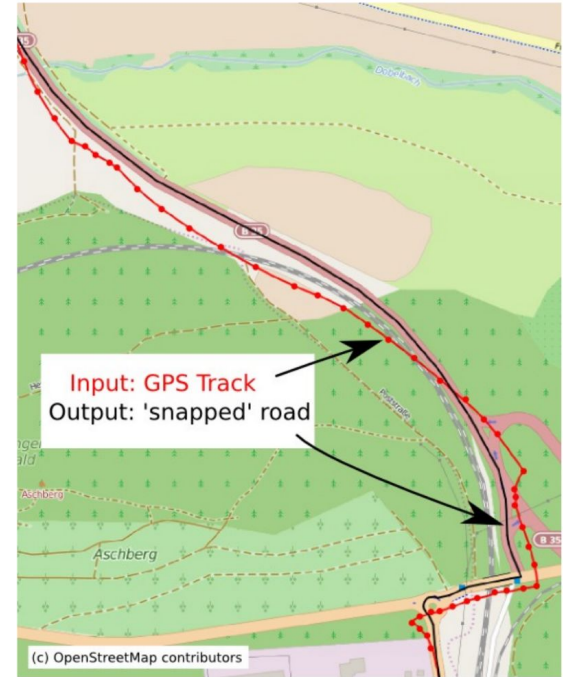
- What happens in this situation? (Multiple noisy points)
 - Assume constant sampling rate 1 minute
 - Speed threshold = 240 km/h (= 4 km/minute)



- What if the trip starts from t=3?
Which points of the trip survive?

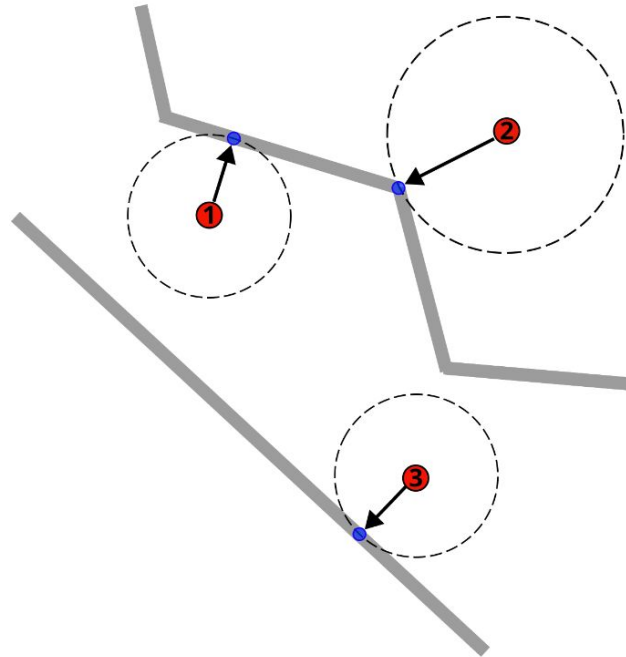
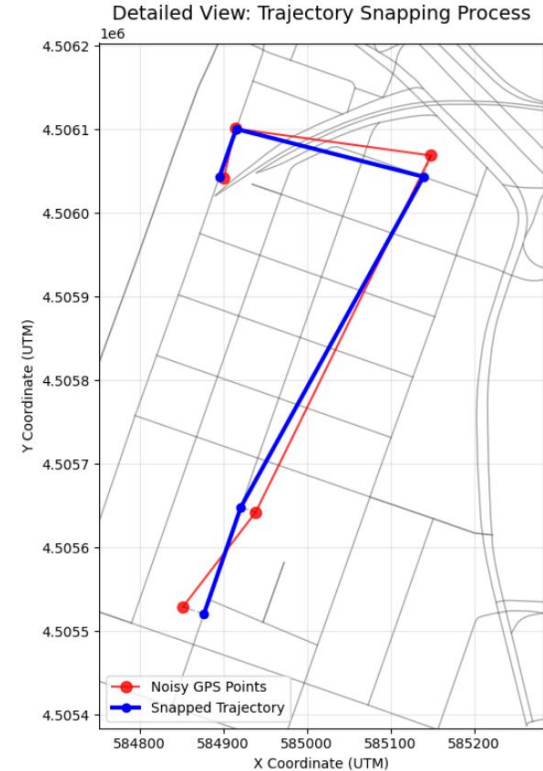
Point map matching

- 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



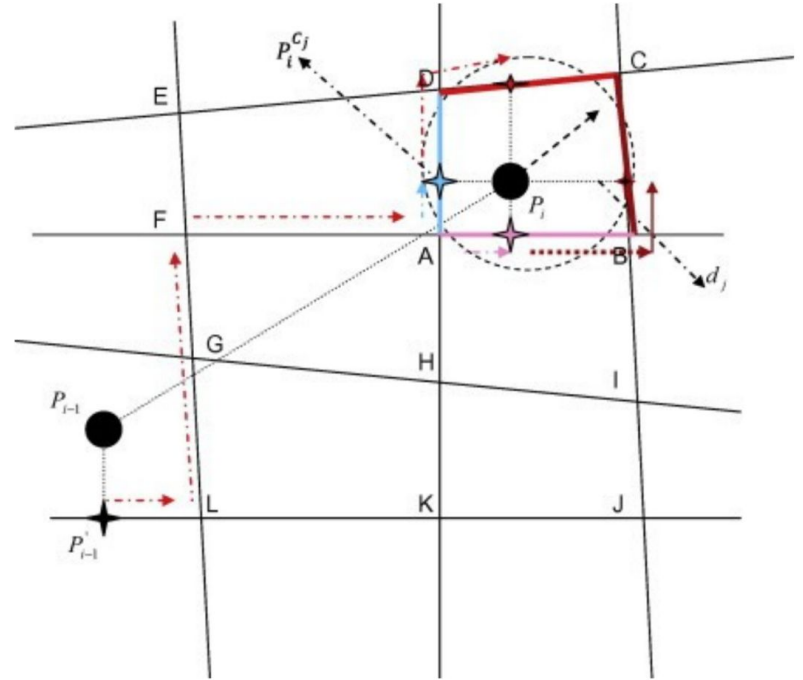
Point map matching

- Point projection
 - Requires to compare each point to each road segment
 - Assign closest point over closest segment
- Anything wrong with ③ ?



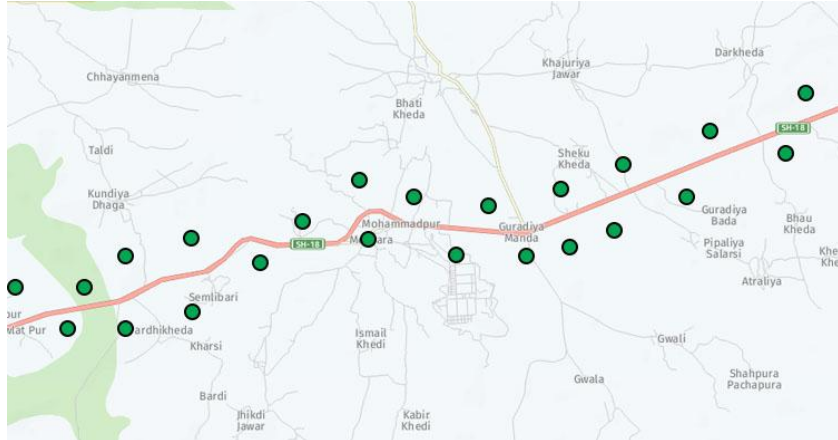
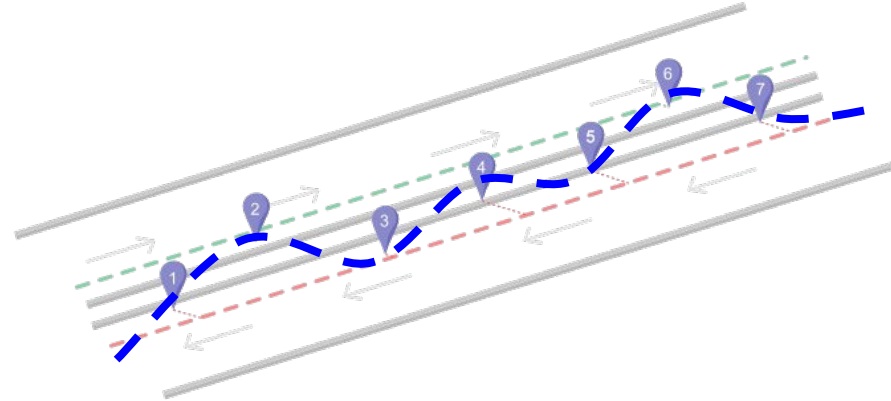
Point map matching

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



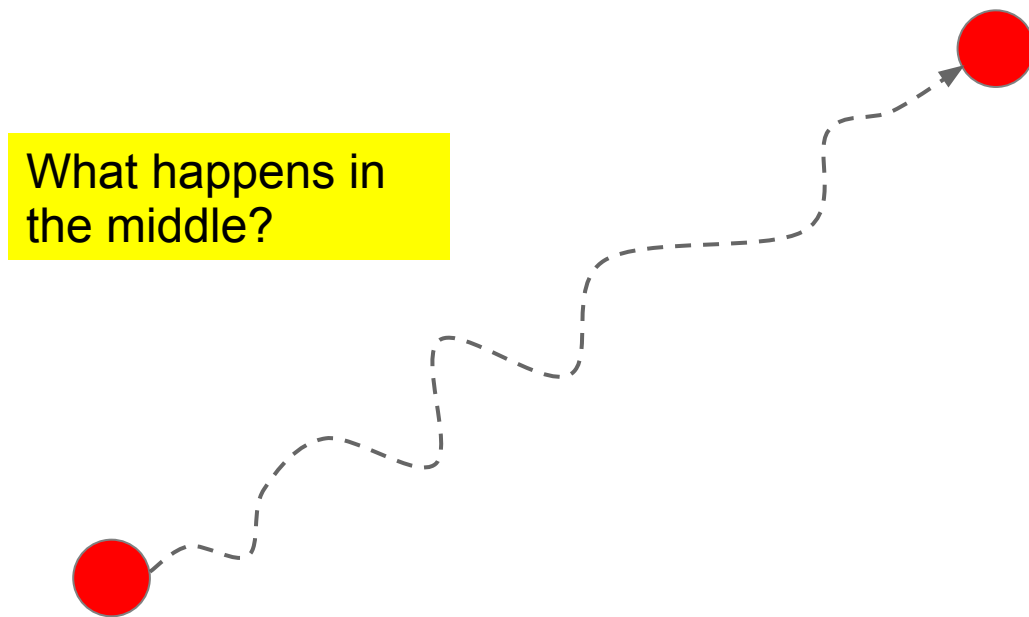
Point map matching

- 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



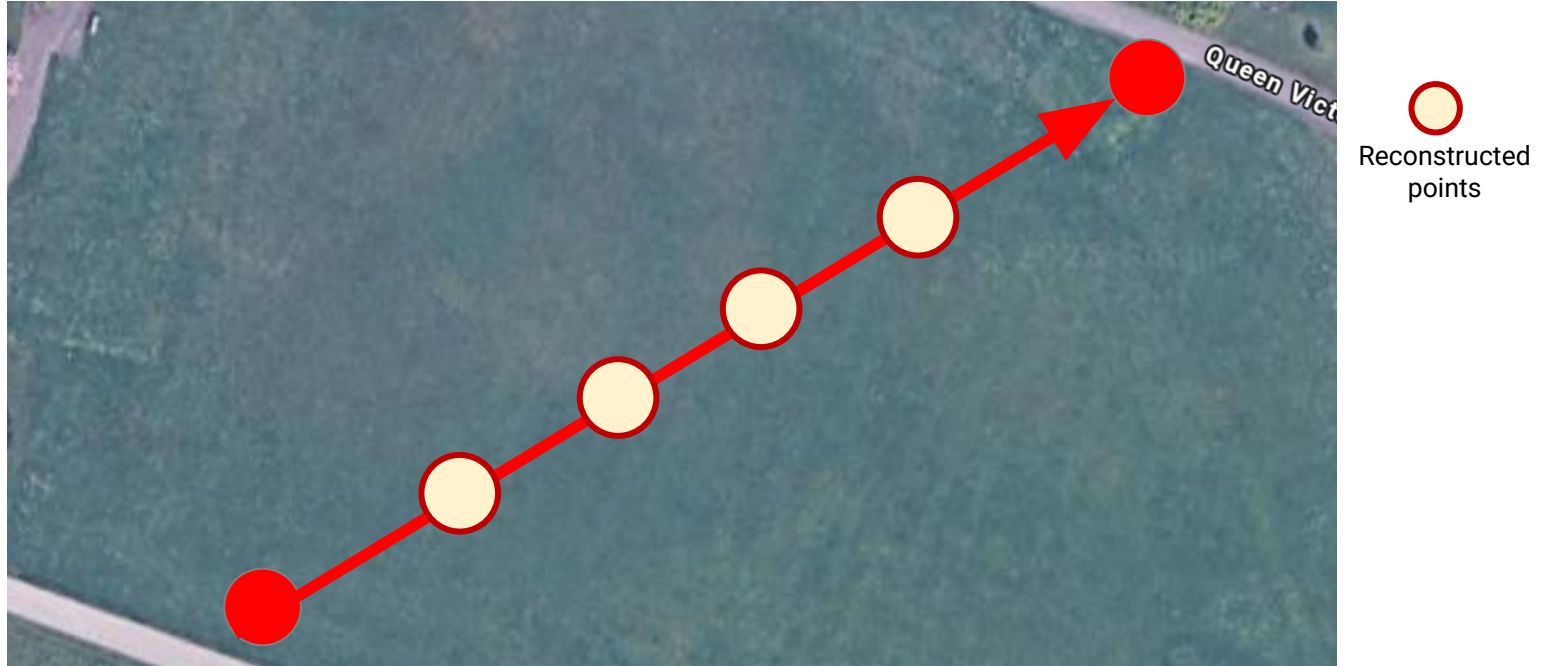
Route reconstruction

- 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 => optimal path



Reconstructed points



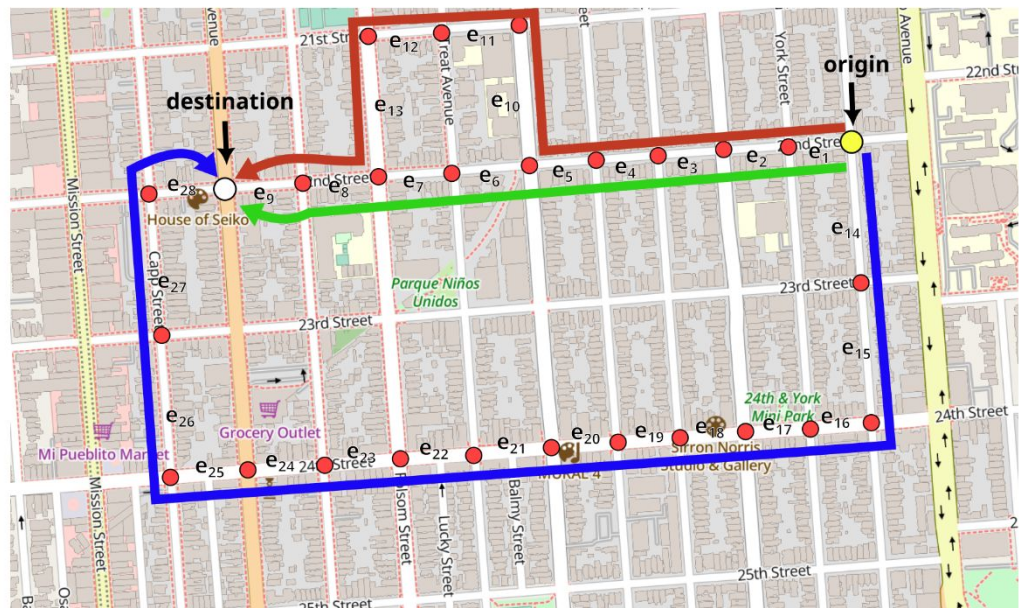
Route reconstruction

- Optimizing what?

$$SP(o, d) = \arg \min_{p \in \mathcal{P}_{o,d}} \sum_{e \in p} cost(e)$$

$$\mathcal{P}_{o,d} = \{ \underbrace{\langle (p_0, p_1), \dots, (p_{n-1}, p_n) \rangle}_{\forall_i (p_i, p_{i+1}) \in E} \mid \underbrace{o = p_0}_{\text{origin}} \wedge \underbrace{d = p_n}_{\text{destination}} \}$$

- Trip length
 - $cost(e) = \text{length}$
- Travel time
 - $cost(e) = \text{length}/\text{speed}$
- Num. intersections
 - $cost(e) = 1$



Route reconstruction

Generally referred as “Shortest paths”, yet alternative “optimal paths” can be used

- Usually based on a notion of single-segment costs
- Most 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

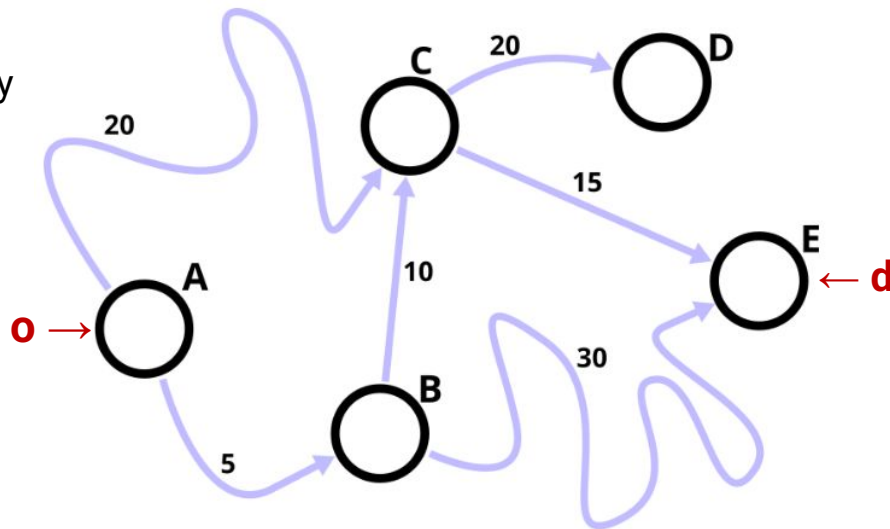
Refresher: Dijkstra's minimum cost algorithm

- Simple and efficient: $O(M + N \log N)$ time complexity ($M = |\text{edges}|$, $N = |\text{nodes}|$)

Input: Graph $G = (V, E)$, origin node o , destination node d

Output: Shortest distance $\text{dist}[d]$ from o to d , shortest path P

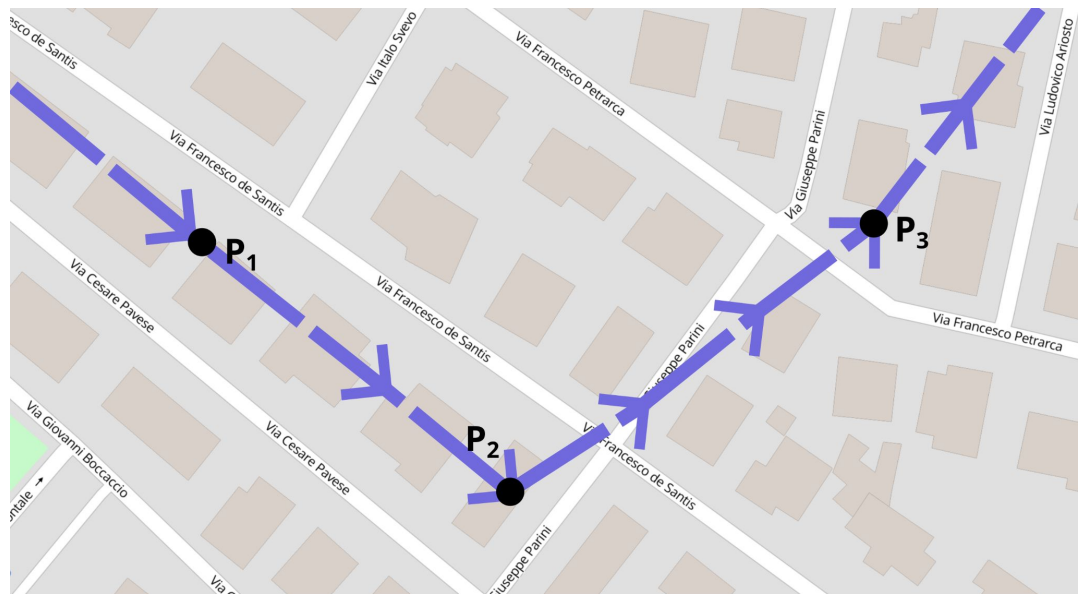
```
1 foreach  $v \in V$  do
2    $\text{dist}[v] \leftarrow \infty$ ;
3    $\text{prev}[v] \leftarrow \text{undefined}$ ;
4  $\text{dist}[o] \leftarrow 0$ ;
5  $Q \leftarrow V$ ;
6 while  $Q$  is not empty do
7    $u \leftarrow$  extract node in  $Q$  with minimum  $\text{dist}[u]$ ;
8   if  $u = d$  then break // Stop when  $d$  is reached
9   foreach neighbor  $v$  of  $u$  in  $Q$  do
10    if  $\text{dist}[u] + \text{cost}(u, v) < \text{dist}[v]$  then
11       $\text{dist}[v] \leftarrow \text{dist}[u] + \text{cost}(u, v)$ ;
12       $\text{prev}[v] \leftarrow u$ ;
```



	A	B	C	D	E
Init	0	∞	∞	∞	∞

Trajectory Map Matching

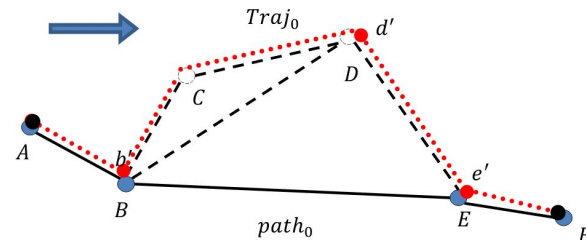
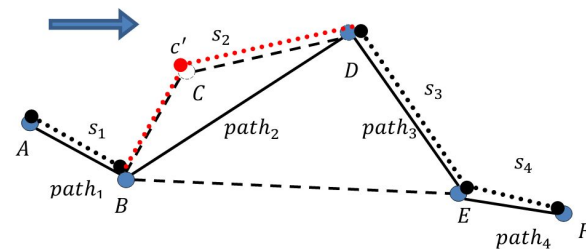
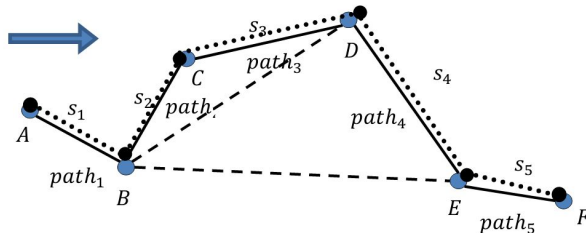
- Assigns points to road segments
 - Reconstructs the movement between consecutive points
 - Ensures coherence of the overall process
-
- Two sample approaches:
 - Based on shortest path
 - Based on probabilities



Shortest path-based Map Matching

Zhu-Honda-Gonder method, used by MappyMatch

- Similar ideas as trajectory simplification
 - Match first and last point
 - there can be multiple candidates to try
 - Compute shortest path on the network
 - Compute “fitting” of path w.r.t. Trajectory (based on distance of real points from path)
 - If fitting < threshold $\Rightarrow$
 - split into two or more parts
 - run recursively the process on both
 - If multiple O/D candidates, select the one with best fitting value



Probability-based Map Matching

Newson-Crumm method, used by pyTrack

- Point-to-road (p->C) assignments, with probabilities

$$p(p_i \mapsto C_k) = e^{-\frac{1}{2}d(p_i, C_k)^2/\sigma^2}/\sqrt{2\pi}\sigma$$

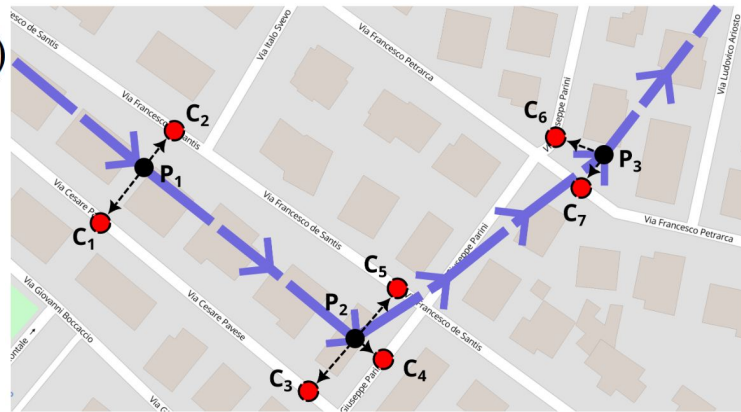
- Road-to-road (C->C) assignments

$$P(C_k \rightarrow C_l) = e^{-\Delta/\beta} / \beta$$

$$\Delta = shortest_path_dist(C_k, C_l) - d(C_k, C_l)$$

- Compute best path that visits all points in the correct sequence

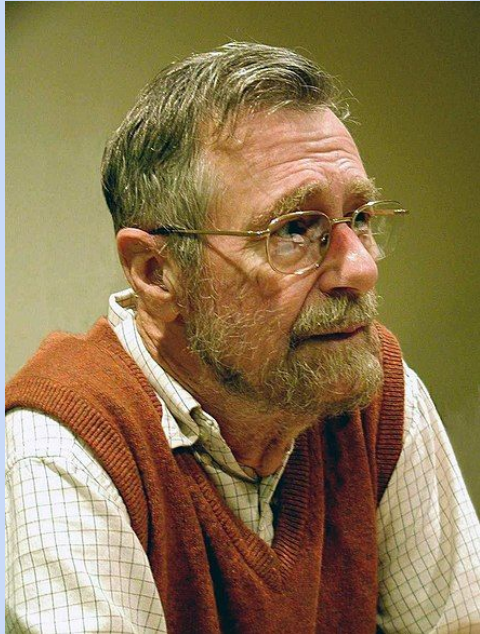
$$\text{maximize} \quad \prod_i p(p_i \mapsto C_i) p(C_i \rightarrow C_{i+1})$$



Reference: Newson & Krumm. ACM GIS'09.

INTERVALLO

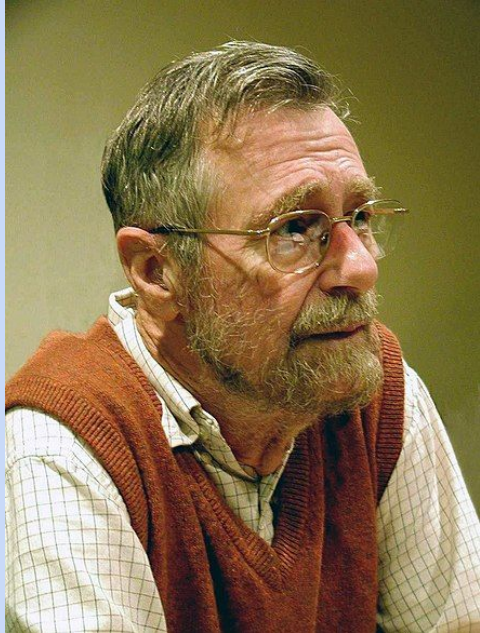
Edsger W. Dijkstra



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

INTERVALLO

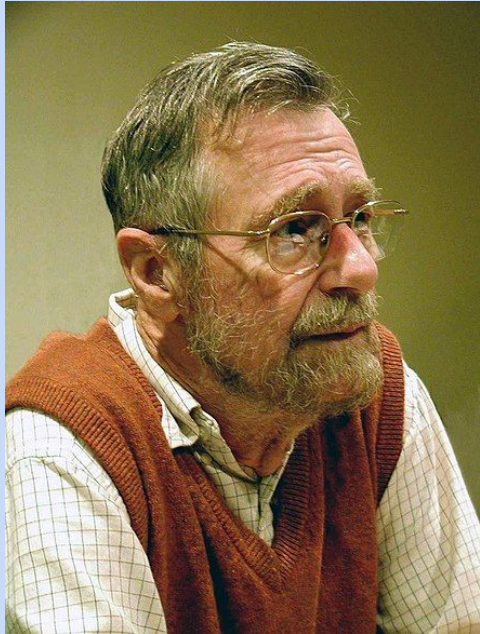
Dijkstra is famous for...



- 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

INTERVALLO

Dijkstra is famous for...

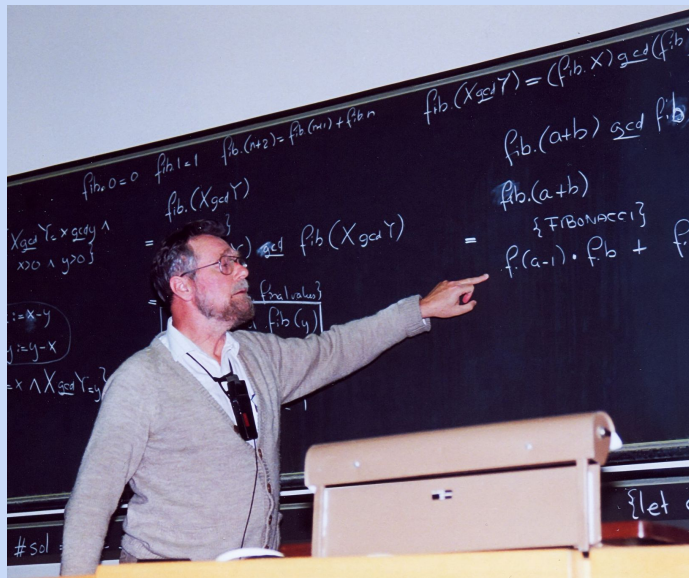


- His habit of writing everything with paper & fountain pen
- Hundreds of papers, many unpublished
 - E. W. Dijkstra Archive
- Counting should start from 0, not 1...

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 $1 \leq i \leq N+1$; starting with 0, however, gives the nicer range $0 \leq i \leq N$. 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."

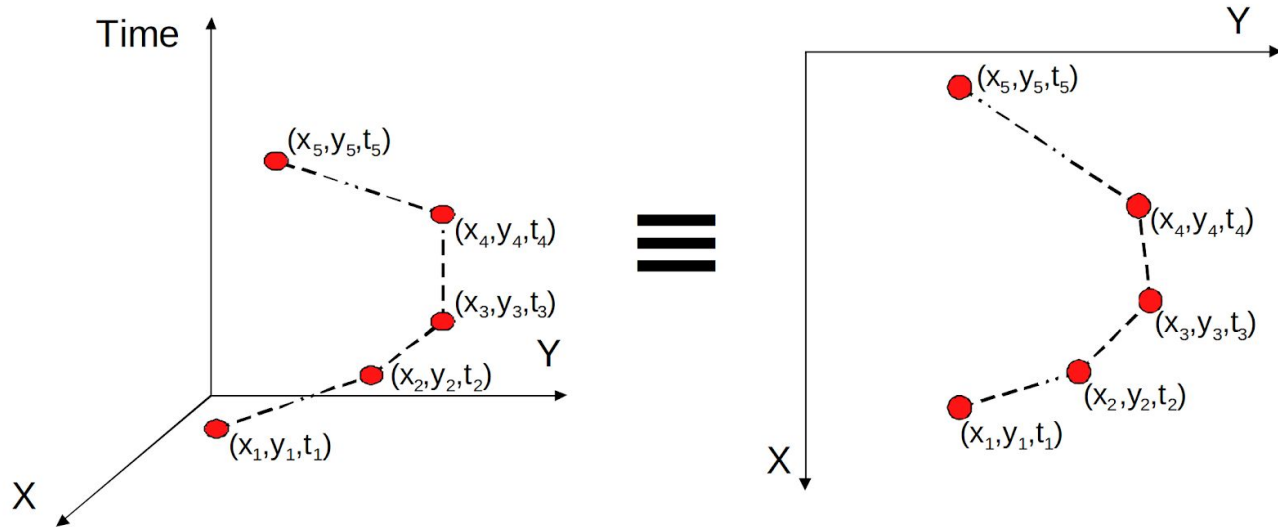
- Long exams
 - Each student was examined in Dijkstra's office or home, and an exam lasted several hours

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

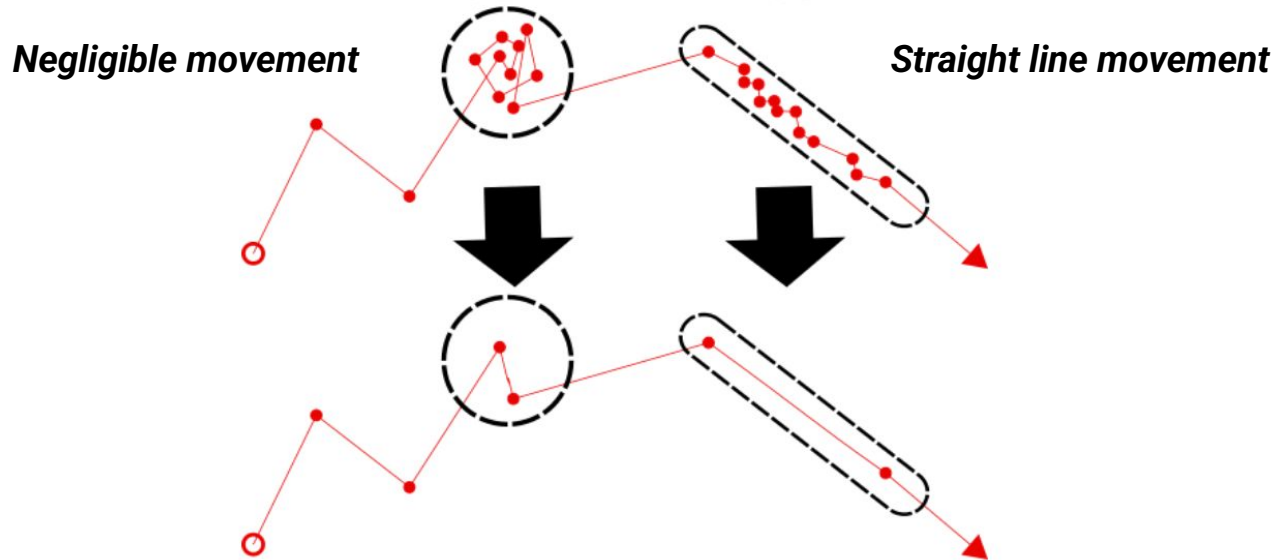
Trajectory data

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



Trajectory compression / simplification

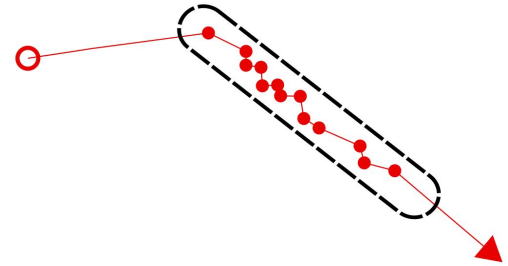
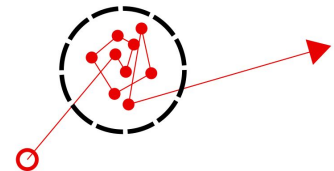
- Typical cases where points might be removed



Compression/simplification methods

Some standard methods for simplifying polygonal curves:

- Ramer-Douglas-Peucker, 1973
 - aimed to simplify straight-line movements
- Driemel-HarPeled-Wenk, 2010
 - aimed to simplify no-movements



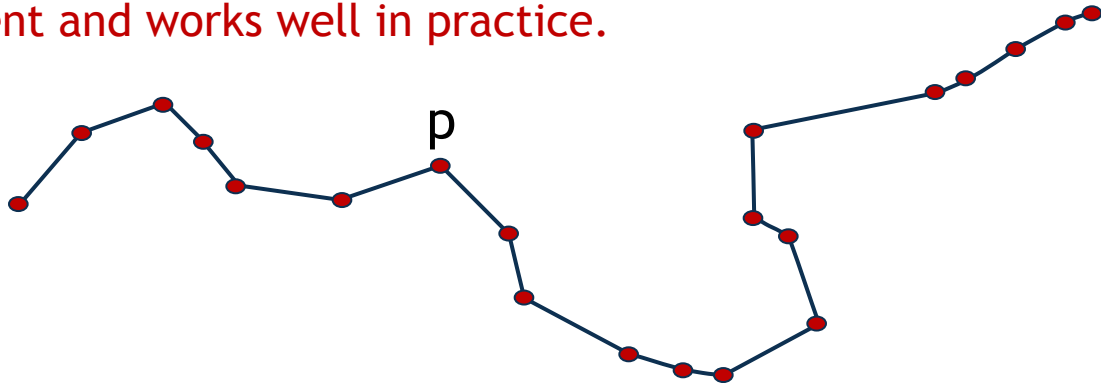
Ramer-Douglas-Peucker

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.

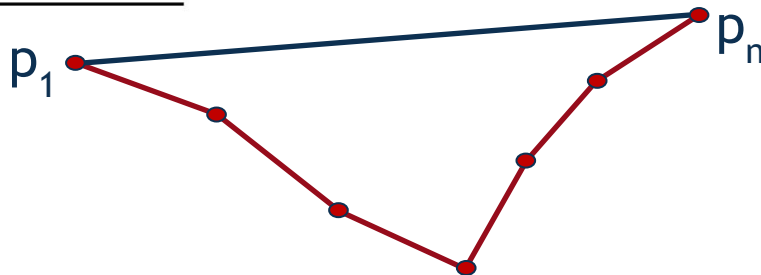


Ramer-Douglas-Peucker

Input: Trajectory $T = \langle p_1, \dots, p_n \rangle$, spatial threshold ϵ

Output: Simplified trajectory T'

```
1  $i \leftarrow \text{index } j \in [1, \dots, n] \text{ such that } d(p_j, \overline{p_1, p_n}) \text{ is maximized;}$   
2 if  $d(p_i, \overline{p_1, p_n}) > \epsilon$  then                                //  $p_i$  deviates too much, split!  
3    $T'_{left} \leftarrow DP(\langle p_1, \dots, p_i \rangle);$   
4    $T'_{right} \leftarrow DP(\langle p_i, \dots, p_n \rangle);$   
5   return joined  $T'_{left}$  and  $T'_{right};$   
6 else  
7   return  $T;$ 
```

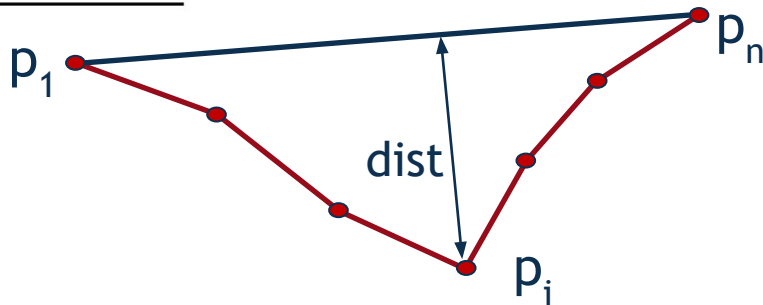


Ramer-Douglas-Peucker

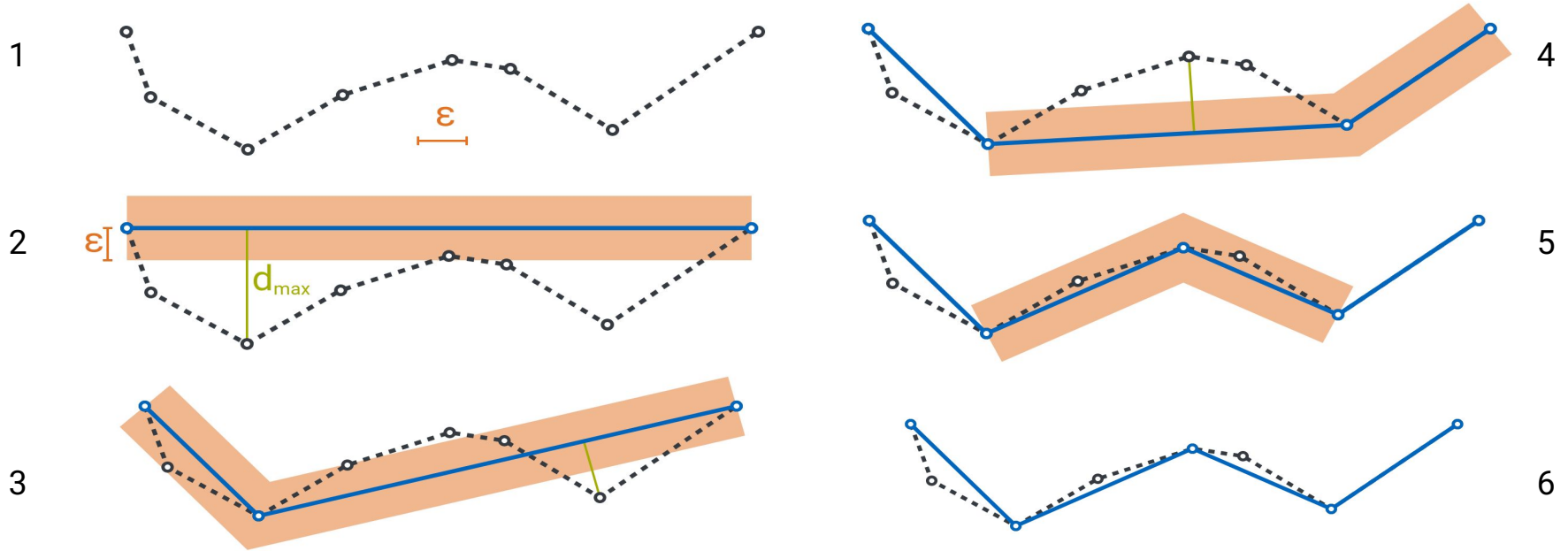
Input: Trajectory $T = \langle p_1, \dots, p_n \rangle$, spatial threshold ϵ

Output: Simplified trajectory T'

```
1  $i \leftarrow \text{index } j \in [1, \dots, n]$  such that  $d(p_j, \overline{p_1, p_n})$  is maximized;  
2 if  $d(p_i, \overline{p_1, p_n}) > \epsilon$  then                                //  $p_i$  deviates too much, split!  
3    $T'_{left} \leftarrow DP(\langle p_1, \dots, p_i \rangle)$ ;  
4    $T'_{right} \leftarrow DP(\langle p_i, \dots, p_n \rangle)$ ;  
5   return joined  $T'_{left}$  and  $T'_{right}$ ;  
6 else  
7   return  $T$ ;
```



Ramer-Douglas-Peucker



Driemel-HarPeled-Wenk

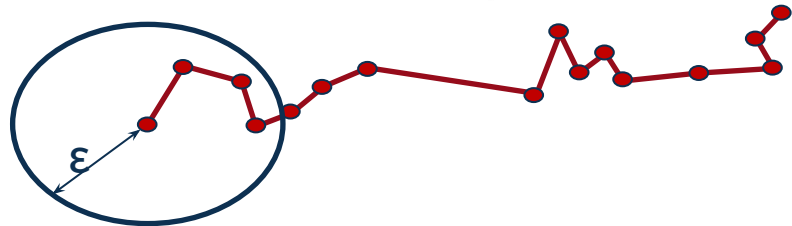
- Simplest solution
- Aims to produce well spaced points
 - Basically a spatial subsampling: takes a point every ε meters
 - More local vision than The DP algorithm
- Very fast and easy to implement

Driemel-HarPeled-Wenk

Input: Trajectory $T = \langle p_1, \dots, p_n \rangle$, spatial threshold ϵ

Output: Simplified trajectory T'

```
1  $T' \leftarrow \langle p_1 \rangle$ ;  
2  $i \leftarrow 1$ ;  
3 while  $i < n$  do  
4    $q \leftarrow p_i$ ;  
5    $i \leftarrow$  smallest index  $j \in [i + 1, \dots, n]$  such that  $d(q, p_j) > \epsilon$ ;  
6   if  $i$  is undefined then  $i \leftarrow n$  // We reached end of  $T$   
7   Append  $p_i$  to  $T'$ ;  
8 return  $T'$ ;
```

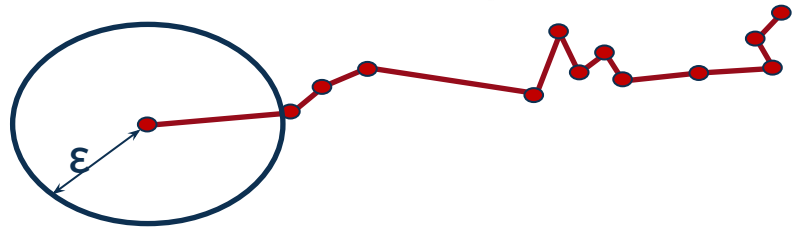


Driemel-HarPeled-Wenk

Input: Trajectory $T = \langle p_1, \dots, p_n \rangle$, spatial threshold ϵ

Output: Simplified trajectory T'

```
1  $T' \leftarrow \langle p_1 \rangle$ ;  
2  $i \leftarrow 1$ ;  
3 while  $i < n$  do  
4    $q \leftarrow p_i$ ;  
5    $i \leftarrow$  smallest index  $j \in [i + 1, \dots, n]$  such that  $d(q, p_j) > \epsilon$ ;  
6   if  $i$  is undefined then  $i \leftarrow n$  // We reached end of  $T$   
7   Append  $p_i$  to  $T'$ ;  
8 return  $T'$ ;
```

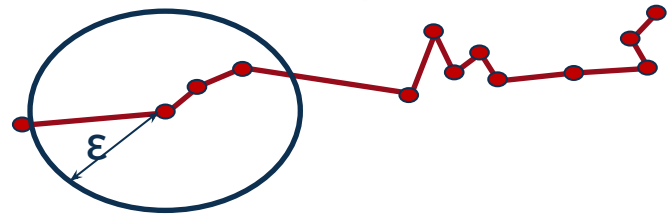


Driemel-HarPeled-Wenk

Input: Trajectory $T = \langle p_1, \dots, p_n \rangle$, spatial threshold ϵ

Output: Simplified trajectory T'

```
1  $T' \leftarrow \langle p_1 \rangle$ ;  
2  $i \leftarrow 1$ ;  
3 while  $i < n$  do  
4    $q \leftarrow p_i$ ;  
5    $i \leftarrow$  smallest index  $j \in [i + 1, \dots, n]$  such that  $d(q, p_j) > \epsilon$ ;  
6   if  $i$  is undefined then  $i \leftarrow n$  // We reached end of  $T$   
7   Append  $p_i$  to  $T'$ ;  
8 return  $T'$ ;
```

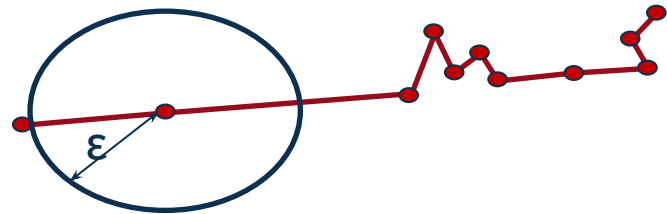


Driemel-HarPeled-Wenk

Input: Trajectory $T = \langle p_1, \dots, p_n \rangle$, spatial threshold ϵ

Output: Simplified trajectory T'

```
1  $T' \leftarrow \langle p_1 \rangle$ ;  
2  $i \leftarrow 1$ ;  
3 while  $i < n$  do  
4    $q \leftarrow p_i$ ;  
5    $i \leftarrow$  smallest index  $j \in [i + 1, \dots, n]$  such that  $d(q, p_j) > \epsilon$ ;  
6   if  $i$  is undefined then  $i \leftarrow n$  // We reached end of  $T$   
7   Append  $p_i$  to  $T'$ ;  
8 return  $T'$ ;
```

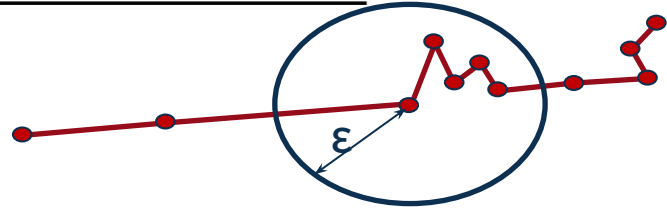


Driemel-HarPeled-Wenk

Input: Trajectory $T = \langle p_1, \dots, p_n \rangle$, spatial threshold ϵ

Output: Simplified trajectory T'

```
1  $T' \leftarrow \langle p_1 \rangle$ ;  
2  $i \leftarrow 1$ ;  
3 while  $i < n$  do  
4    $q \leftarrow p_i$ ;  
5    $i \leftarrow$  smallest index  $j \in [i + 1, \dots, n]$  such that  $d(q, p_j) > \epsilon$ ;  
6   if  $i$  is undefined then  $i \leftarrow n$  // We reached end of  $T$   
7   Append  $p_i$  to  $T'$ ;  
8 return  $T'$ ;
```

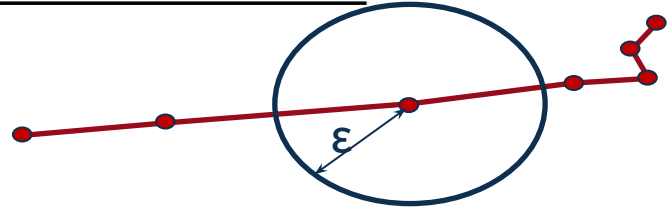


Driemel-HarPeled-Wenk

Input: Trajectory $T = \langle p_1, \dots, p_n \rangle$, spatial threshold ϵ

Output: Simplified trajectory T'

```
1  $T' \leftarrow \langle p_1 \rangle$ ;  
2  $i \leftarrow 1$ ;  
3 while  $i < n$  do  
4    $q \leftarrow p_i$ ;  
5    $i \leftarrow$  smallest index  $j \in [i + 1, \dots, n]$  such that  $d(q, p_j) > \epsilon$ ;  
6   if  $i$  is undefined then  $i \leftarrow n$  // We reached end of  $T$   
7   Append  $p_i$  to  $T'$ ;  
8 return  $T'$ ;
```

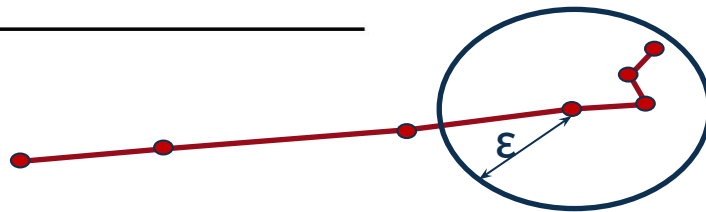


Driemel-HarPeled-Wenk

Input: Trajectory $T = \langle p_1, \dots, p_n \rangle$, spatial threshold ϵ

Output: Simplified trajectory T'

```
1  $T' \leftarrow \langle p_1 \rangle$ ;  
2  $i \leftarrow 1$ ;  
3 while  $i < n$  do  
4    $q \leftarrow p_i$ ;  
5    $i \leftarrow$  smallest index  $j \in [i + 1, \dots, n]$  such that  $d(q, p_j) > \epsilon$ ;  
6   if  $i$  is undefined then  $i \leftarrow n$  // We reached end of  $T$   
7   Append  $p_i$  to  $T'$ ;  
8 return  $T'$ ;
```

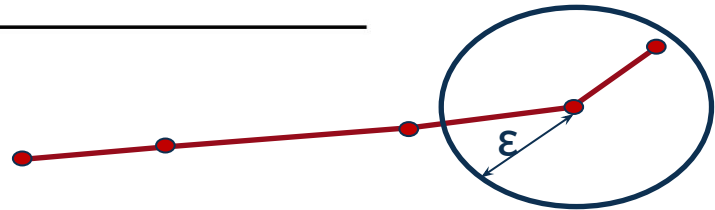


Driemel-HarPeled-Wenk

Input: Trajectory $T = \langle p_1, \dots, p_n \rangle$, spatial threshold ϵ

Output: Simplified trajectory T'

```
1  $T' \leftarrow \langle p_1 \rangle$ ;  
2  $i \leftarrow 1$ ;  
3 while  $i < n$  do  
4    $q \leftarrow p_i$ ;  
5    $i \leftarrow$  smallest index  $j \in [i + 1, \dots, n]$  such that  $d(q, p_j) > \epsilon$ ;  
6   if  $i$  is undefined then  $i \leftarrow n$  // We reached end of  $T$   
7   Append  $p_i$  to  $T'$ ;  
8 return  $T'$ ;
```



Driemel-HarPeled-Wenk

Input: Trajectory $T = \langle p_1, \dots, p_n \rangle$, spatial threshold ϵ

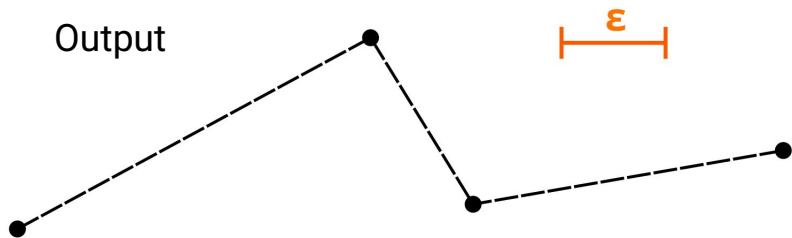
Output: Simplified trajectory T'

```
1  $T' \leftarrow \langle p_1 \rangle$ ;  
2  $i \leftarrow 1$ ;  
3 while  $i < n$  do  
4    $q \leftarrow p_i$ ;  
5    $i \leftarrow$  smallest index  $j \in [i + 1, \dots, n]$  such that  $d(q, p_j) > \epsilon$ ;  
6   if  $i$  is undefined then  $i \leftarrow n$  // We reached end of  $T$   
7   Append  $p_i$  to  $T'$ ;  
8 return  $T'$ ;
```

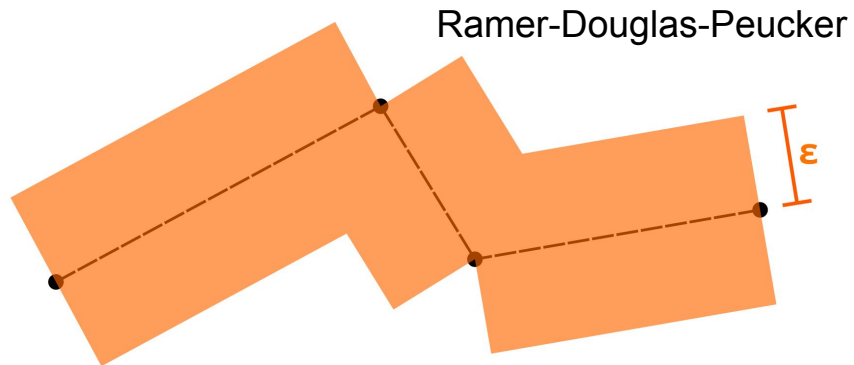
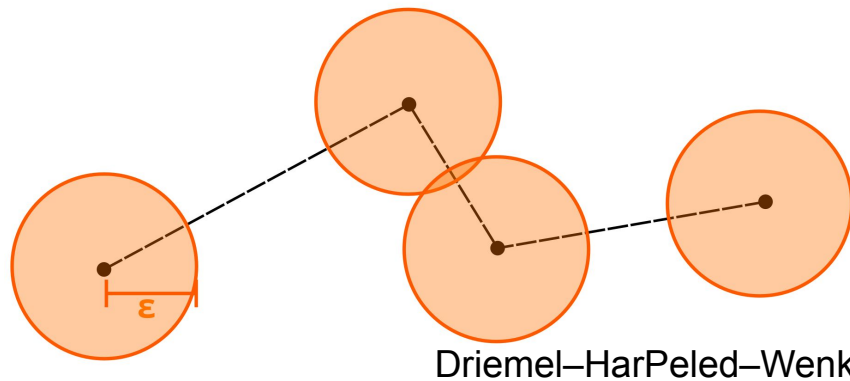


Ramer-Douglas-Peucker vs Driemel-HarPeled-Wenk

Output

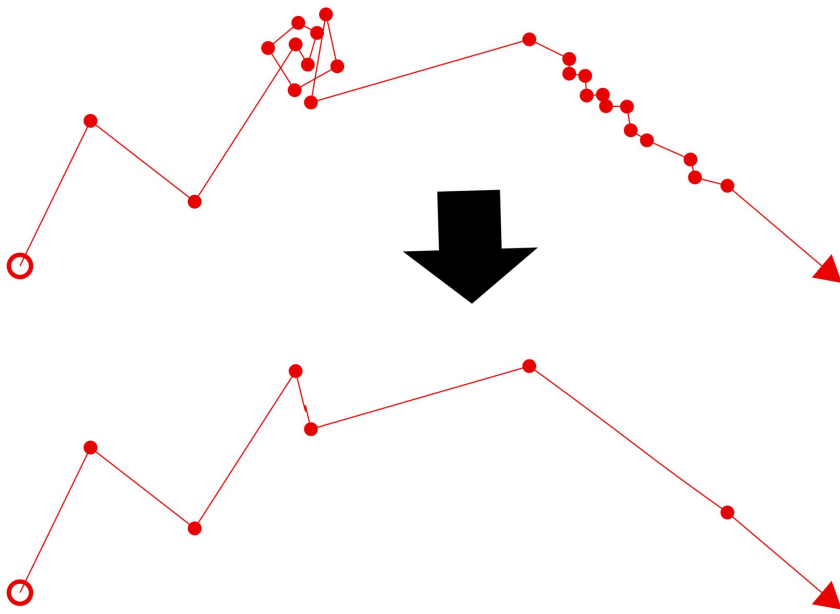


=



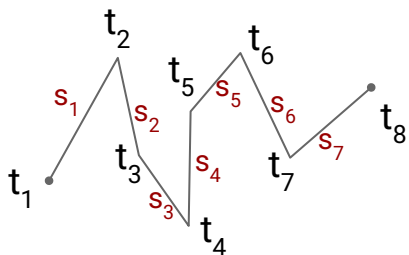
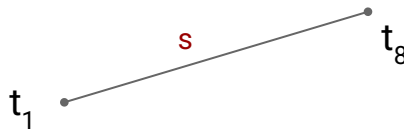
Impact on speed

- 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?



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:



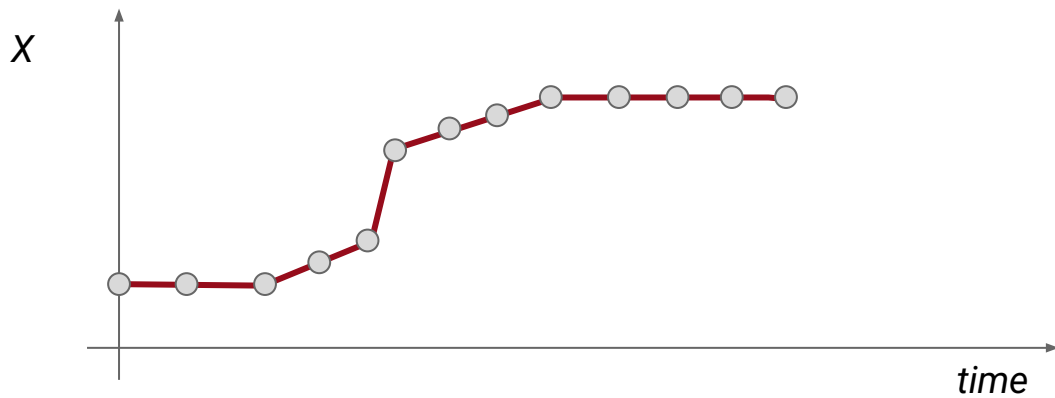
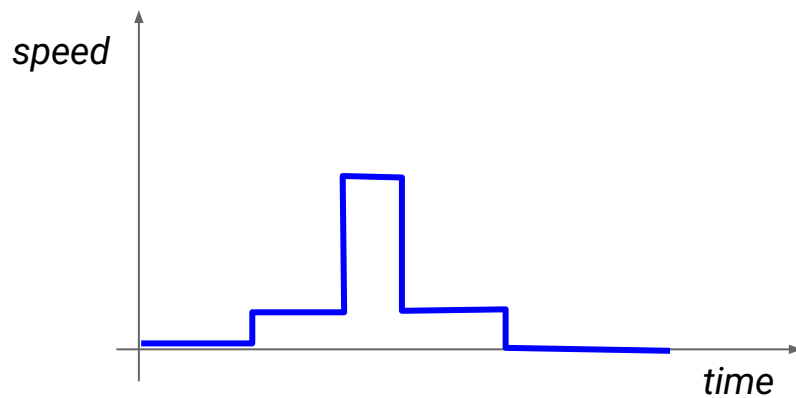
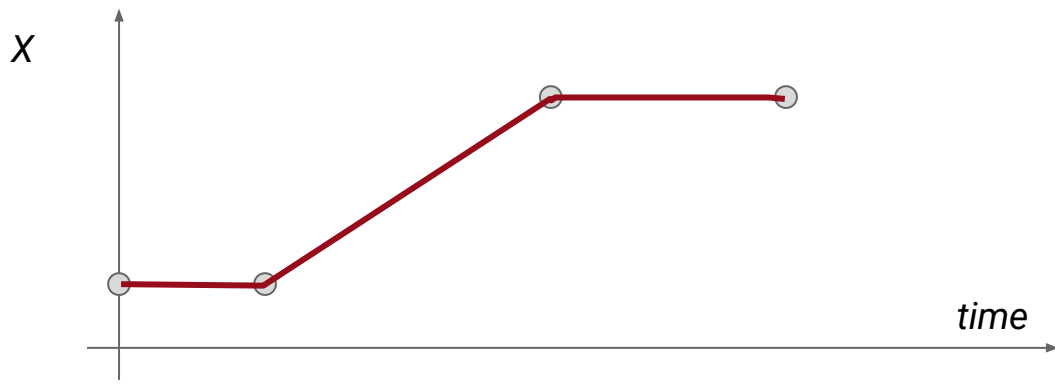
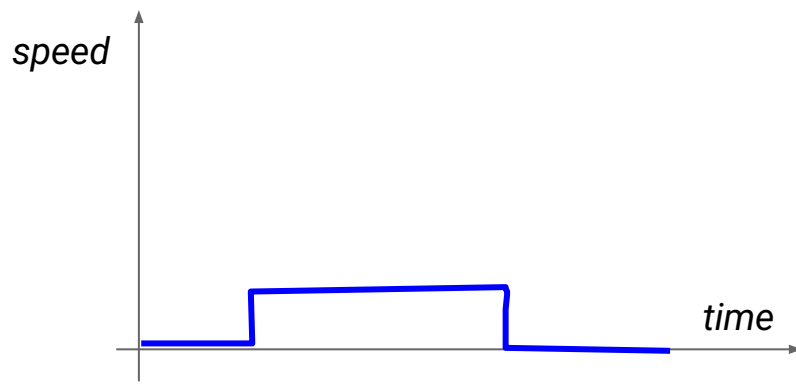
$$\text{Avg speed} = (s_1 + \dots + s_7) / (t_8 - t_1)$$

>

$$\text{Avg speed} = s / (t_8 - t_1)$$

Impact on speed

- Flattening effect



INTERVALLO

How fast is a cow?



INTERVALLO

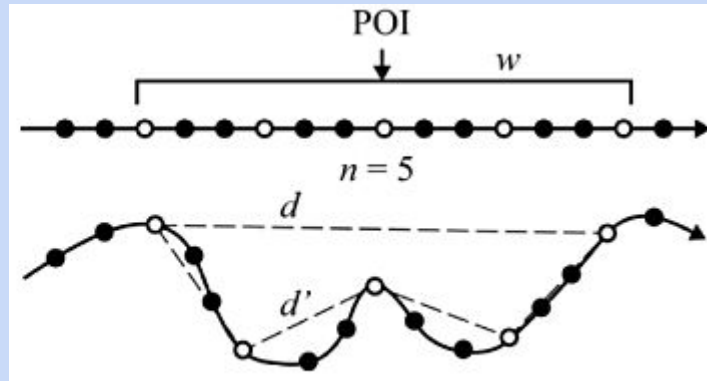
How fast is a cow?

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

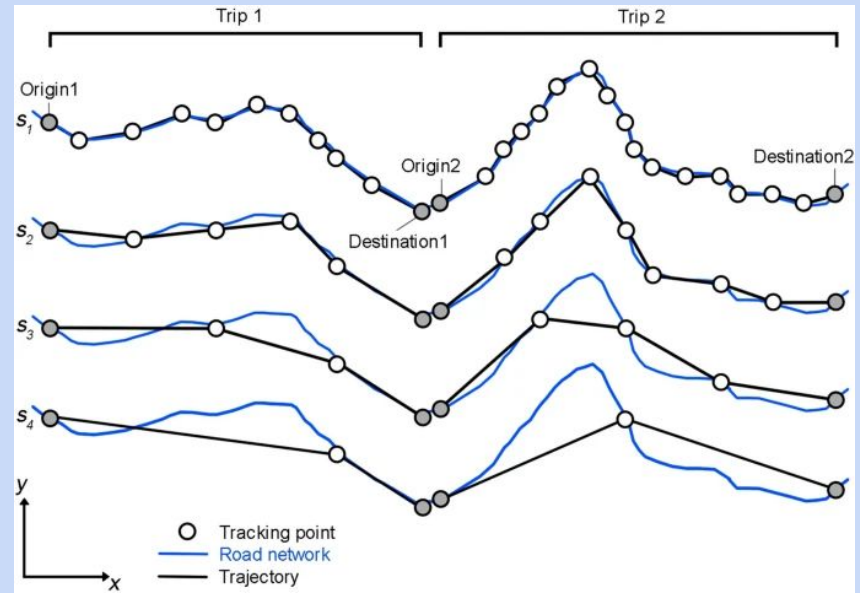
INTERVALLO

How fast is a cow?

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



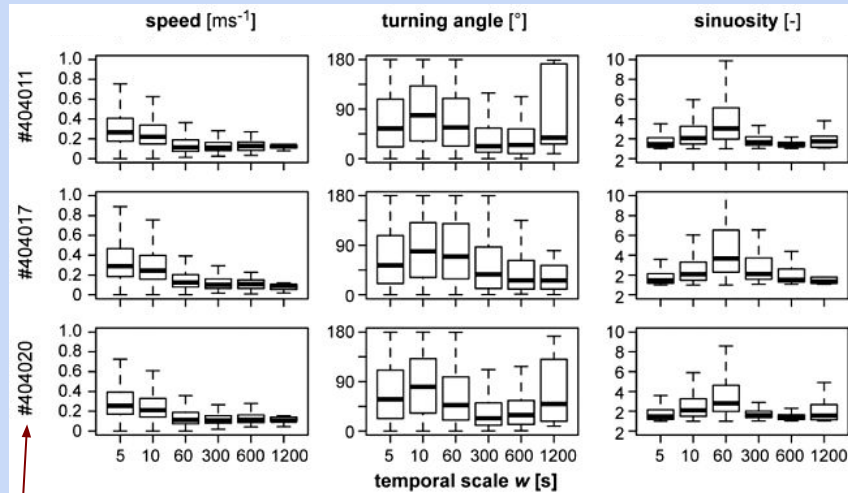
Understanding the impact of temporal scale on human movement analytics
Su, R., Dodge, S. & Goulias, K.G (2022)



INTERVALLO

How fast is a cow?

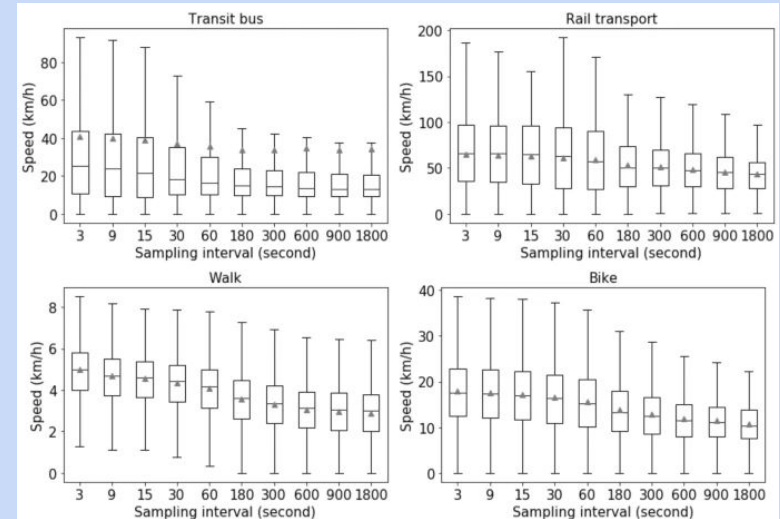
How fast is a cow? Cross-Scale Analysis of Movement Data
Laube P, Purves RS (2011)



Cow ID

Understanding the impact of temporal scale on human movement analytics

Su, R., Dodge, S. & Goulias, K.G (2022)

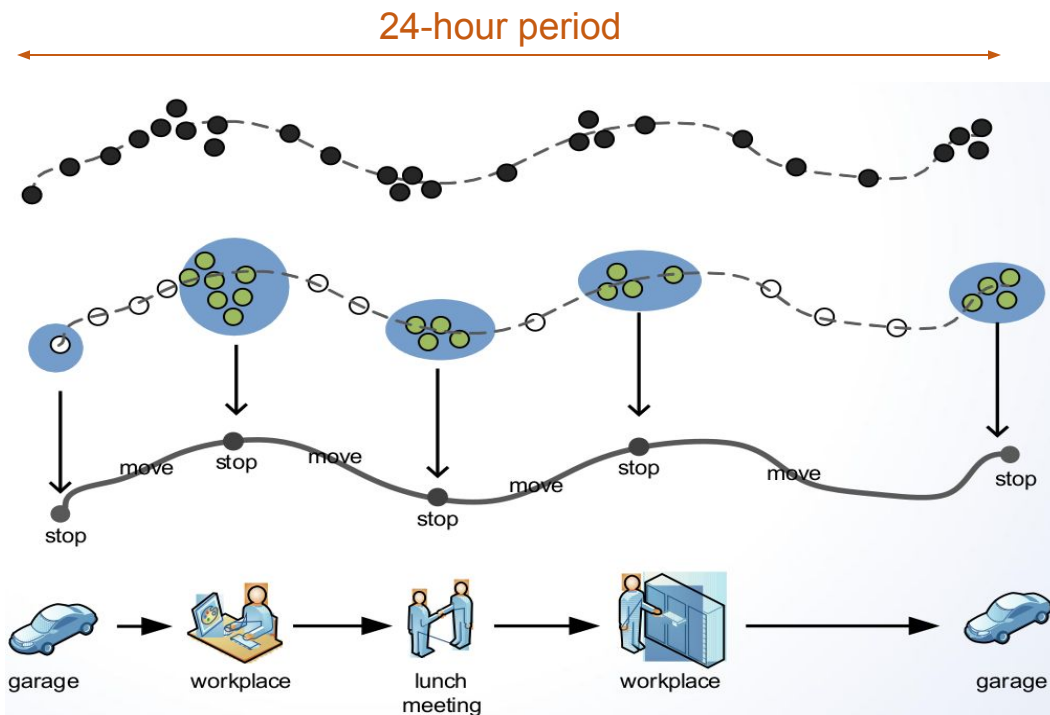


Semantic Enrichment

- Objective: label raw data with inferred knowledge
 - It is already a form of data analysis / classification / prediction
- We will see a few examples:
 - stop detection / trajectory segmentation
 - detect home and other locations & Individual Mobility Network
 - activity recognition (POI-based)

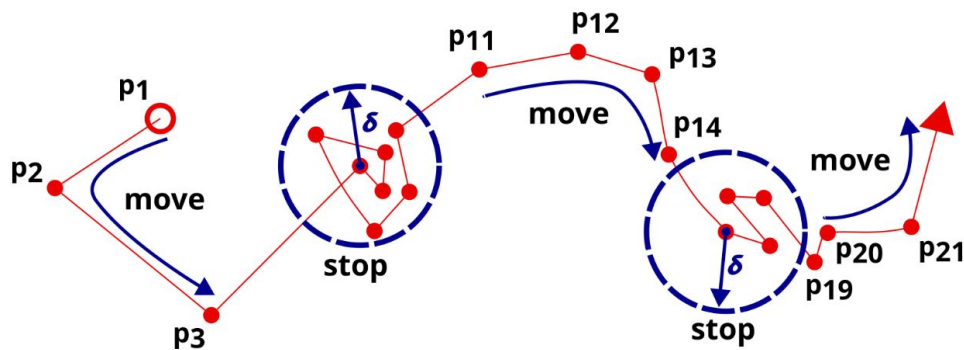
Stop detection & Trajectory segmentation

- 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



($Th_S = \delta$, $Th_T = 3'$, sampling rate = 1 pt/minute)

Stop detection & Trajectory segmentation

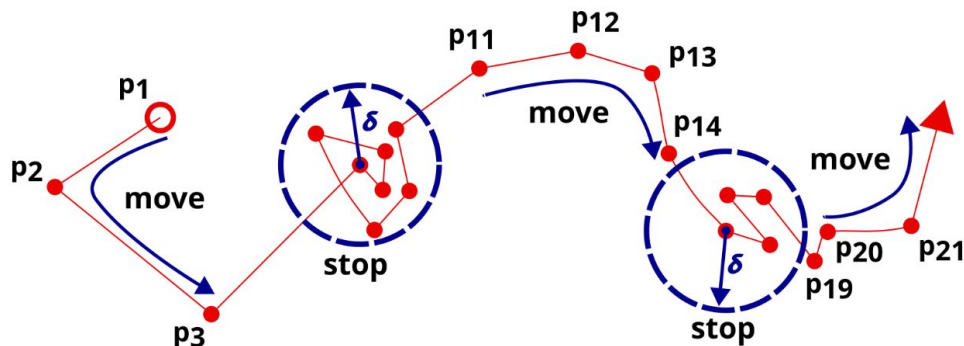
Input: Trajectory $T = (\mathbf{p}_1, \dots, \mathbf{p}_n)$, spatial threshold δ , temporal threshold τ

Output: Set of trajectory segments *trajs_list* and stop periods

stops_list

```

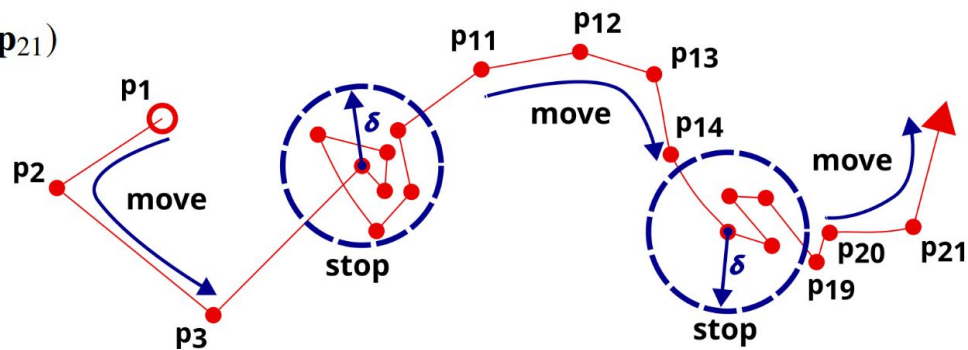
1 stops_list ← ∅;
2 trajs_list ← ∅;
3 a ← 1;
4 t ← ();
5 while a ≤ n do
6   a' ← max{i ∈ [a, n] | ∀ j ∈ [a, i] : dist(p_j, p_a) ≤ δ};
7   if time(p_{a'}) - time(p_a) ≥ τ then    // A stop: interrupt
8     stops_list ← stops_list ∪ {(p_a, p_{a+1}, ..., p_{a'})};
9     if t ≠ () then
10      trajs_list ← trajs_list ∪ {t};
11      t ← ();
12    a ← a';
13  else                                     // Normal point: continue trip
14    Append p_a to t;
15    a ← a + 1;
16  if t ≠ () then
17    trajs_list ← trajs_list ∪ {t};
18  return trajs_list, stops_list;
```



Stop detection & Trajectory segmentation

- Sample output:

$(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3), (\mathbf{p}_{11}, \mathbf{p}_{12}, \mathbf{p}_{13}, \mathbf{p}_{14}), (\mathbf{p}_{19}, \mathbf{p}_{20}, \mathbf{p}_{21})$

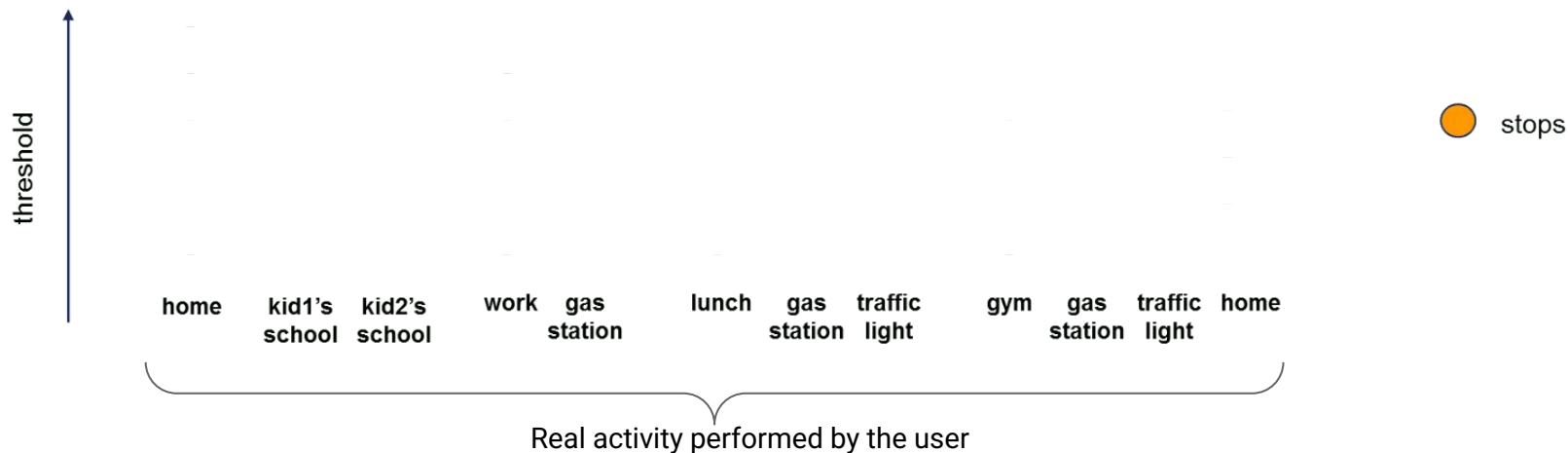


- Usually adjusted adding extreme points

$(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4), (\mathbf{p}_{10}, \mathbf{p}_{11}, \mathbf{p}_{12}, \mathbf{p}_{13}, \mathbf{p}_{14}, \mathbf{p}_{15}), (\mathbf{p}_{18}, \mathbf{p}_{19}, \mathbf{p}_{20}, \mathbf{p}_{21})$

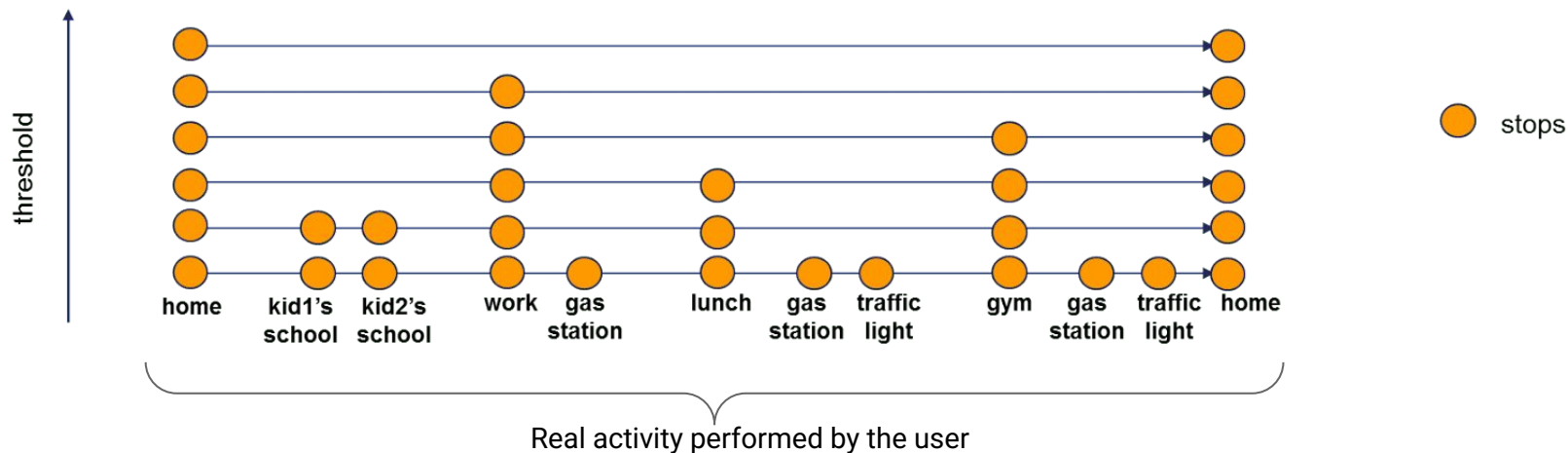
Stop detection & Trajectory segmentation

- Different time thresholds yield different semantics



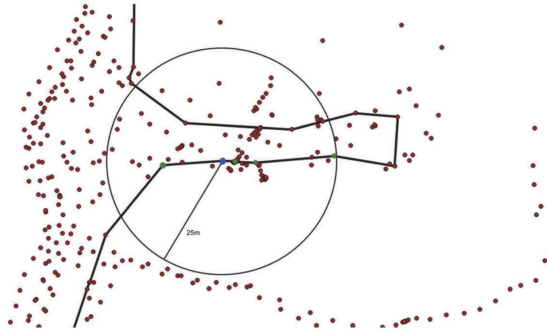
Stop detection & Trajectory segmentation

- Different time thresholds yield different semantics

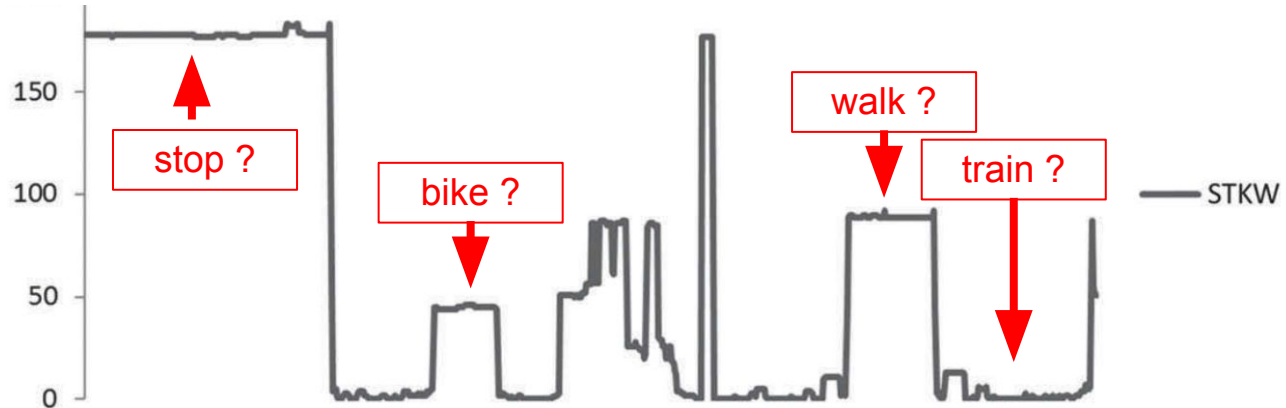


- Which one is the best for you?
 - Application dependent

Generalization: transportation means segmentation

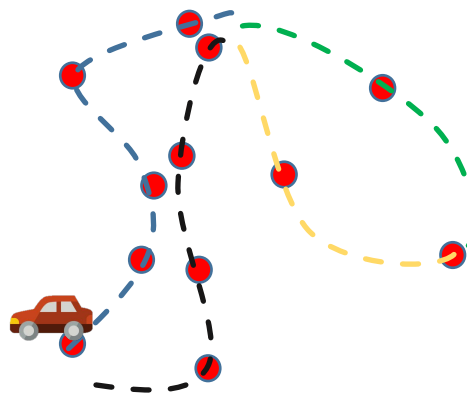


- Speed / density-based approach
 - $STKW(p_i) = \max | [a,b] |$ such that:
 $i \in [a,b]$ and $\forall j \in [a,b]: \text{dist}(p_i, p_j) < \delta$
- Idea: faster means less of my points around me



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 User 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).



Inferring User Locations

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



-

Inferring User Locations

- Cluster points to identify locations
 - Apply a simple radius-based clustering algorithm



Input: List of stop points $stops$, spatial threshold δ

Output: Set of locations $\mathcal{L}$ and stops-to-location mapping

$$f : stops \rightarrow \mathcal{L}$$

// Compute neighborhood size (freq) of stops

1 **foreach** $p_{pivot} \in stops$ **do**

2 $N \leftarrow \{p \in stops \mid dist(p, p_{pivot}) \leq \delta\};$

3 $freq(p_{pivot}) \leftarrow |N|;$

// Build locations in frequency order

4 **while** $stops \neq \emptyset$ **do**

5 $p_{pivot} \leftarrow \arg \max_{p \in stops} freq(p);$

6 $P \leftarrow \{p \in stops \mid dist(p, p_{pivot}) \leq \delta\};$

7 $stops \leftarrow stops - P;$

8 $c \leftarrow \text{mean position of } P;$

9 $\mathcal{L} \leftarrow \text{append } c;$

10 **foreach** $p \in P$ **do**

11 $f(p) \leftarrow c;$

12 **return** $\mathcal{L}, f;$

Inferring Home Location

- Frequency-based approach: home is the location with the most stop points

$$home(\mathcal{L}, S) = \arg \max_{\ell \in \mathcal{L}} |\{\mathbf{p} \in S | f(\mathbf{p}) = \ell, hod(\mathbf{p}) \in \alpha\}|$$

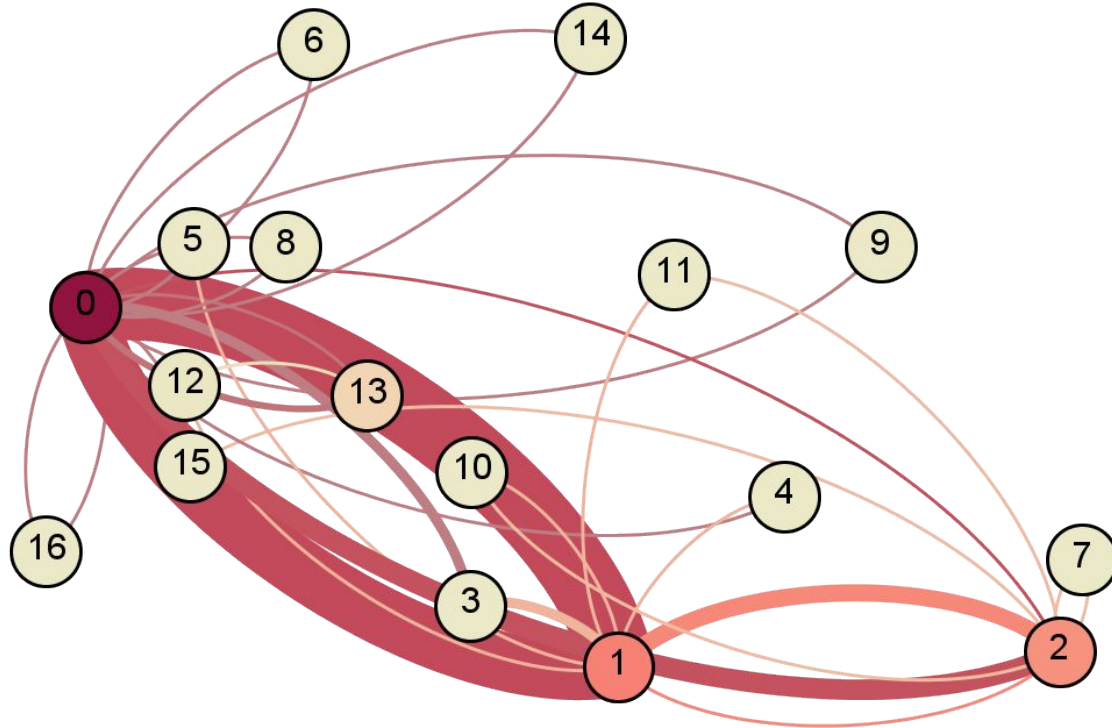
$hod(p)$ = hour of the day of point p

- Potentially constrained by hours
 - $\alpha = [0h - 24h]$ → pure frequency
 - $\alpha = [20h - 7h]$ → night stops
- Similar definition for **Work**
 - Second-most freq. Location
 - α might be set accordingly

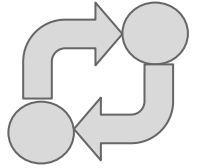


Individual Mobility Network

- Trips between points are aggregated as edges/flows between nodes/locations



Directions of flows:



Individual Mobility Network

Given an individual u and their location sequence L_u , the *Individual Mobility Network* (IMN) of u is a graph

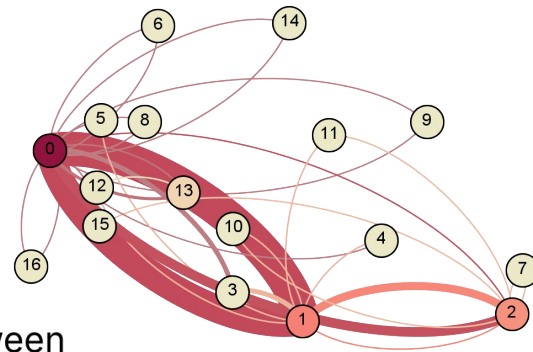
$$G_u = (\mathcal{S}_u, M_u),$$

where:

- $\mathcal{S}_u$ is the set of nodes corresponding to the locations visited by u ;
- $M_u \subseteq \mathcal{S}_u \times \mathcal{S}_u$ is the set of directed edges corresponding to trips between locations.

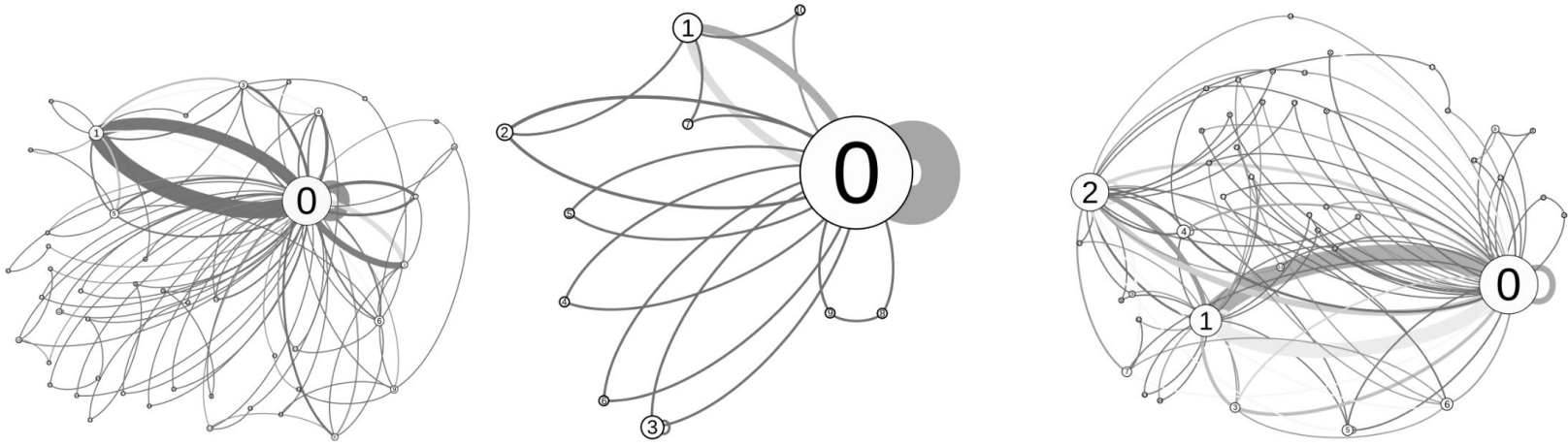
Nodes and edges are associated with weights:

- $\omega(\ell)$: number of trips in L_u reaching location ℓ ;
- $\omega(\ell_1, \ell_2)$: number of trips in L_u from location ℓ_1 to location ℓ_2 .



Individual Mobility Network

- Some real examples
 - $\omega(L)$ shown as size of node
 - $\omega(L_1, L_2)$ shown as thickness of edge



Activity labelling / recognition

Objective: adding information to points / locations

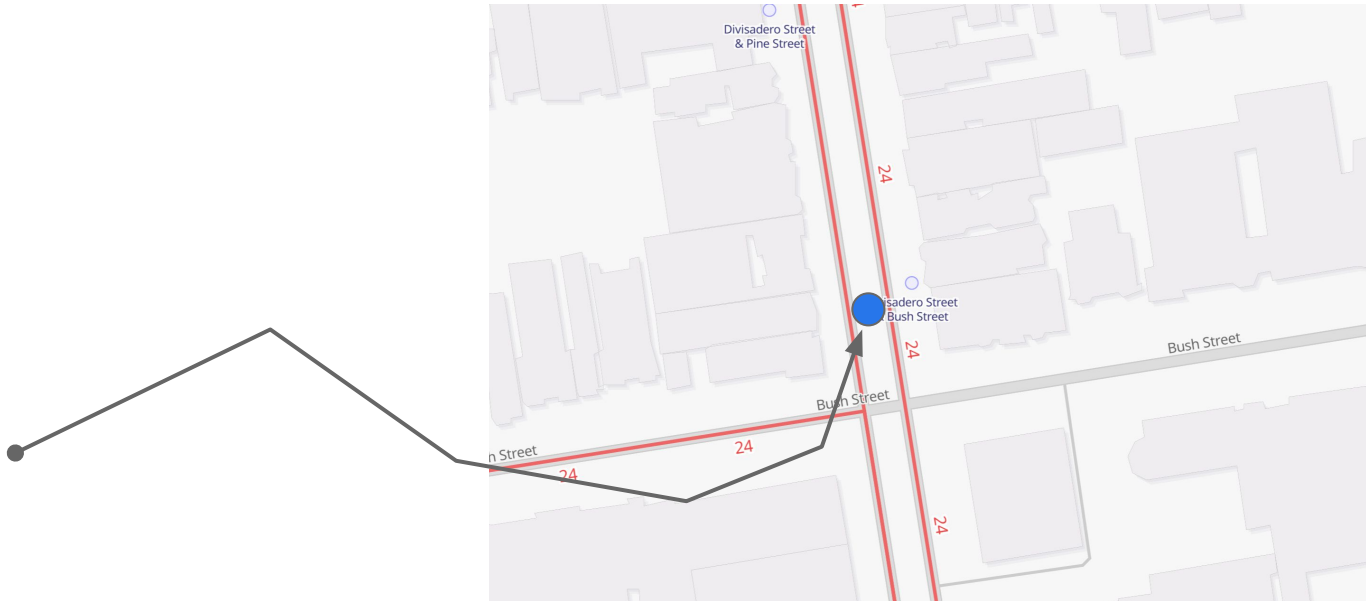
- Home & Work are two particular examples
 - we want something more general
- E.g: education activity; sport; eating; shopping; ...

Two main ways:

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

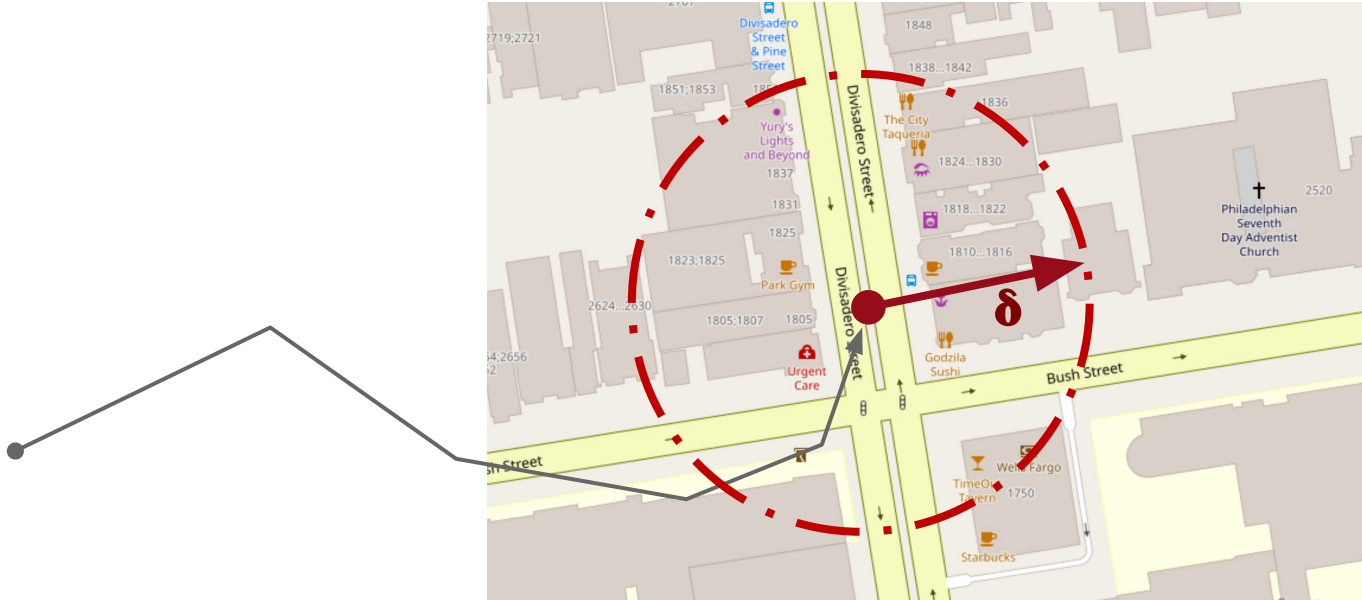
Activity labelling / recognition

- Given a stop point of a GPS track, annotate it with the most probable activities performed



Activity labelling / recognition

- Method: associate the list of possible POIs visited by a user moving by car when they stop
- Distance-based: collect all POIs within a radius δ



Activity labelling / recognition

- Method: associate the list of possible POIs visited by a user moving by car when they stop
- Distance-based: collect all POIs within a radius δ



3		Restaurant (except fast food) / food court
2		Cafe
-		Fast food restaurant or snack bar or sandwich bar or similar
1		Bar
-		Pub
-		Ice cream shop
-		Pharmacy
-		Hospital (for stationary treatment)
1		surgery or a small clinic (for ambulant treatment) / Doctor's practice
-		Car store
-		Kiosk / Newsstand
1		Beauty services except hairdressing
-		Greengrocer / Farm shop
1		Laundry shop / Clothes dry cleaning service
-		Optician

Activity labelling / recognition

- Method: associate the list of possible POIs visited by a user moving by car when they stop
- Distance-based: collect all POIs within a radius δ



Output: vector of POI frequencies

- $F = \langle 3, 2, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0 \rangle$

usually normalized

- $f = \langle 0.33, 0.22, 0, 0.11, 0, 0, 0, 0, 0.11, 0, 0, 0.11, 0, 0.11, 0 \rangle$

Alternative, simpler solution: most frequent

- $mf = \arg \max f =$ 

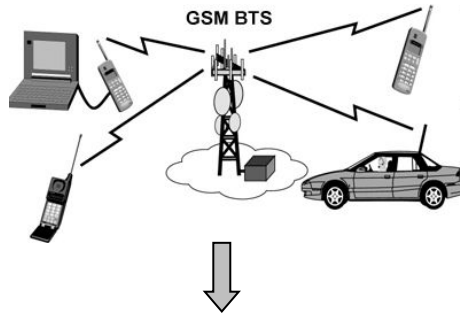
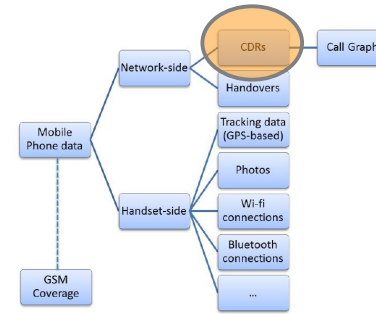
Lower-resolution data: Mobile Phone Data

- Do the methods seen so far apply also to mobile phone data?
 - trajectory filtering: **YES, but** only for really noisy points
 - point mapping & route reconstruction: **YES, but** only for large-scale movements
 - compression methods: **NO**, generally unnecessary
 - stop detection: **YES, but** just checking repeated observations within the same phone cell
 - activity recognition: **YES, but** simplified → see later
- Any ad hoc preprocessing for MP data?
 - **YES**: ping-pong effect → see later

Inferring Home / Work locations with Phone Data

The 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

The case of GSM traces - 1

- 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

The case of GSM traces - 2

- “**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:

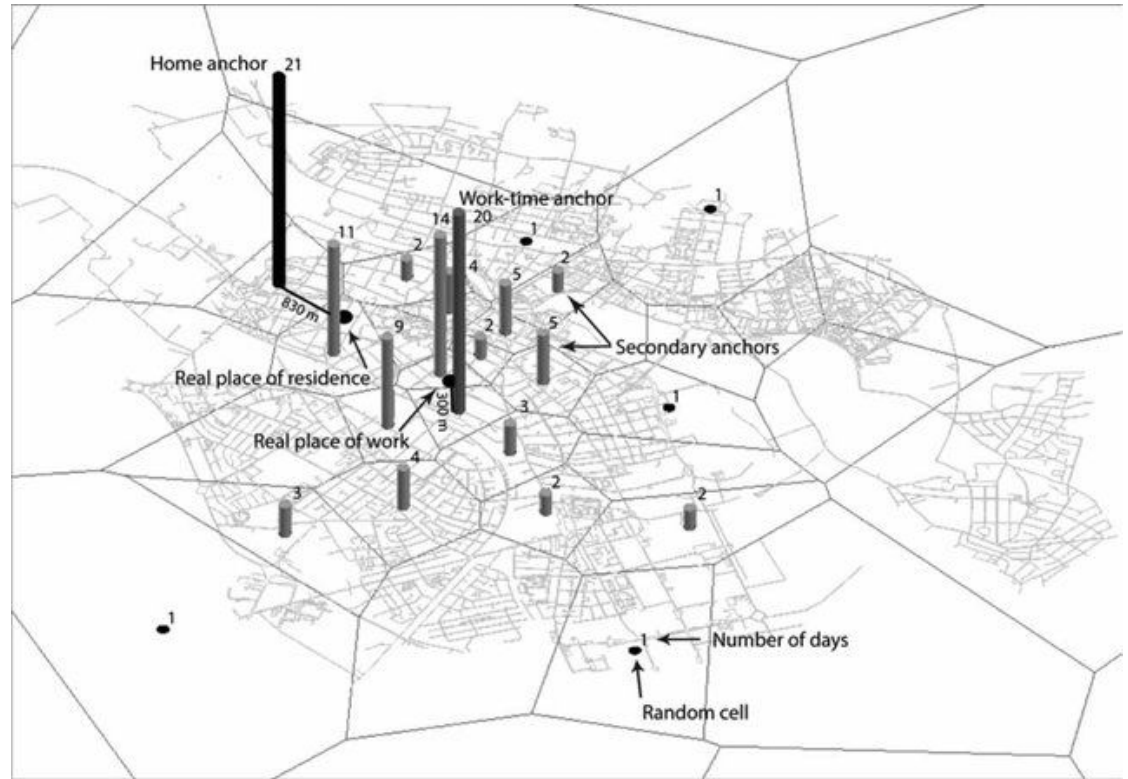
if $AST(loc) < \tau$ **and** $std(loc) < \sigma$
 then WORK
 else HOME

- Based on average start time (AST) of calls and its deviation (std)
- Example:
 - critical time $\tau = 5:00$ pm
 - max std $\sigma = 0.175$

Inferring Home / Work locations with Phone Data

The case of GSM traces - 2

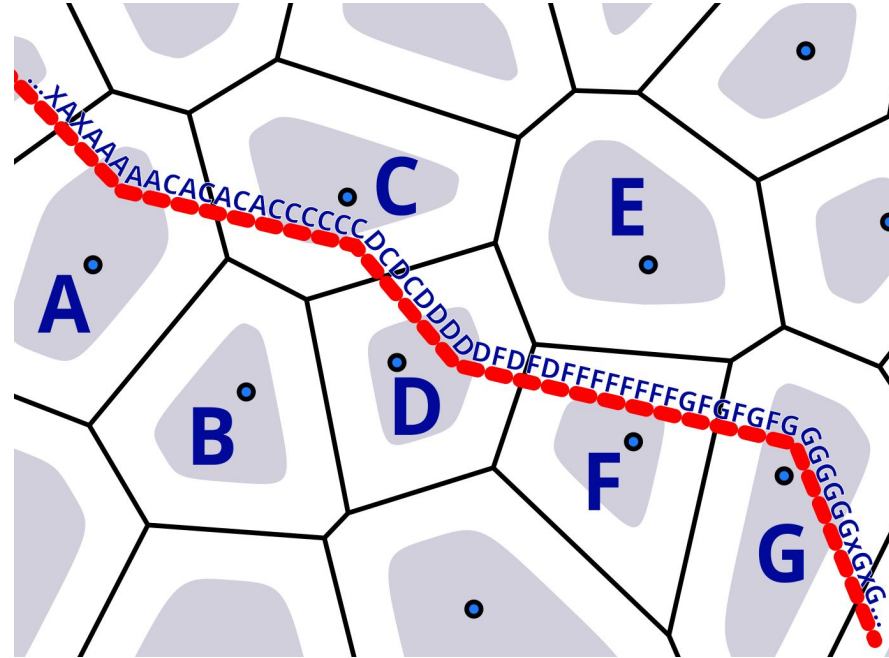
“Personal Anchor Points”



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.

Ping-pong effect

- When close to borderline between phone cells, the connection could wander
- Various reasons:
 - the phone enters and exits several times a cell
 - traffic load of antennas connect/disconnect users to balance
- Result: apparent bouncing between cells
...CCCCDCDCDDDDDD...



Ping-pong effect

Speed-heading-based detection

- average speed between consecutive points

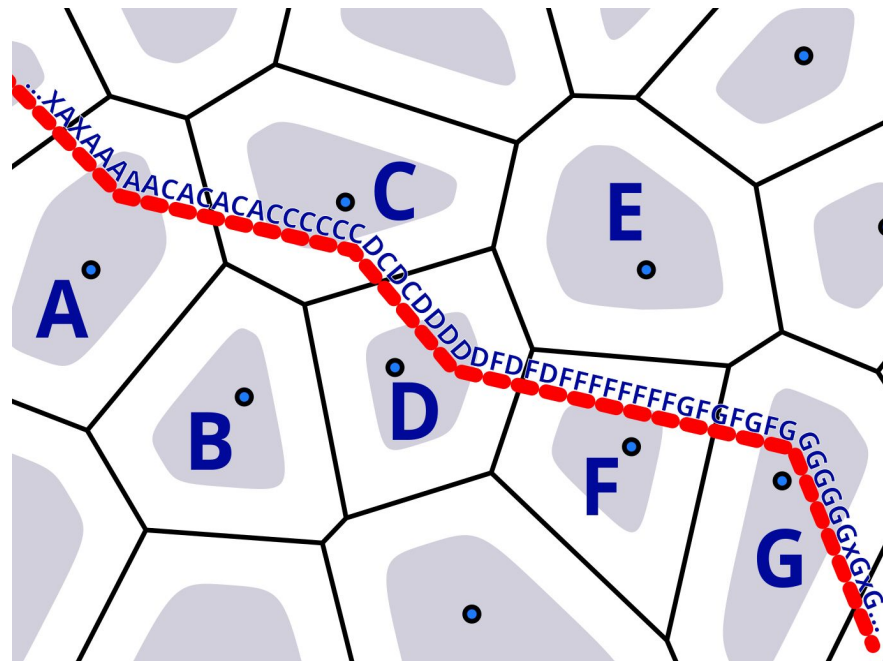
$$v_k = \frac{d(\mathbf{p}_k, \mathbf{p}_{k-1})}{time_k - time_{k-1}}$$

- heading change

$$h_k = |dir_k - dir_{k-1}|$$

- ping-pong handover detection

$$HO = \{\mathbf{p}_k \mid v_k > \sigma \wedge h_k = 180^\circ\}$$

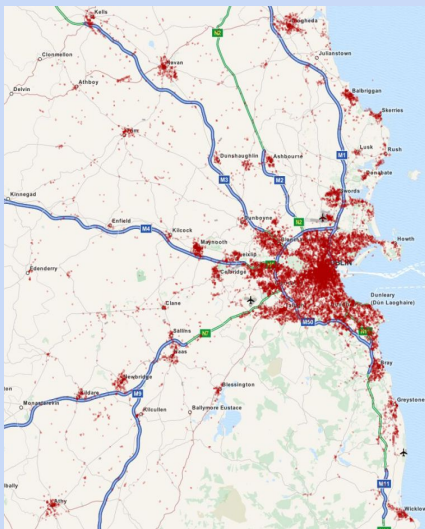
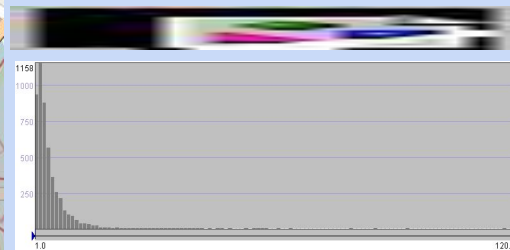


INTERVALLO

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.



Statistical distribution of the number of places per person

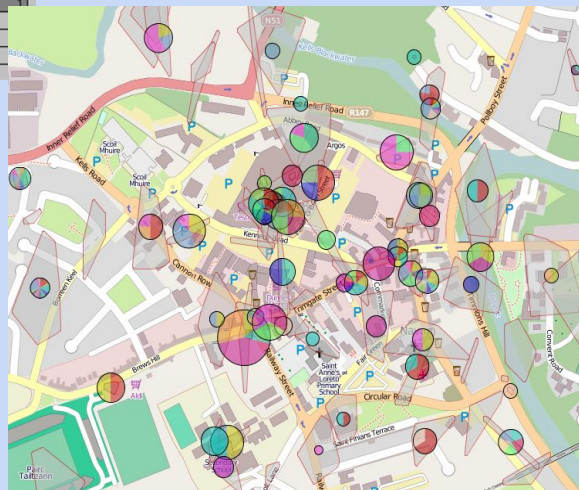
Examples of extracted places

Reading social media to find POIs

■ ■ ■

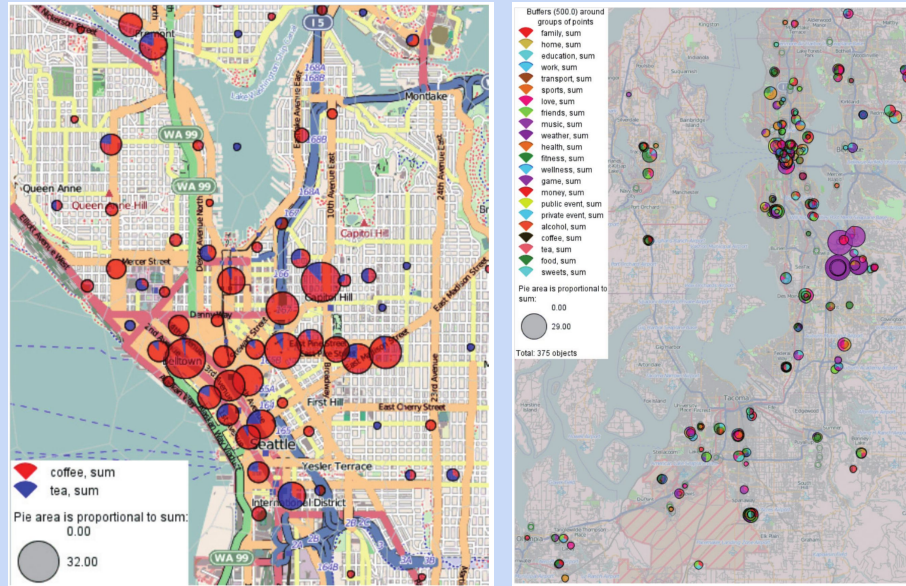
Message	Features	topic=family: Occurrences of topic	topic=home: Occurrences of topic	topic=education: Occurrences of topic	topic=work: Occurrences of topic
@joe_lennon I usually	education	0	0	1	0
@joe_lennon together	education	0	0	1	0
@jas_103 deadly, don't	work	0	0	0	1
Just got home and see	home	0	1	0	0
So excited about my new	sweets	0	0	0	0
@lamtcdizzy I haven't	shopping	0	0	0	0
Get in from my night of	family,home,work	1	1	0	1
Home again at 6pm!	home	0	1	0	0
Bussing it home for t	Get in from my night out, my dad gets home from work	0	0	0	0
Ah shite. It's been a p	two minutes later. Great timing :)	0	0	0	0
@ronanhutchinson be	education	0	0	1	0

- 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

(DIY)

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?

Homework 4.2

(DIY)

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 $\text{pseudo-error}(P)$ as the distance $\text{dist}(P,R)$.

- 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?

Homework 4.3

(Can we do better?)

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

Homework 4.4

(Can we do better?)

*The typical filtering algorithm relies on the individual. Define this new algorithm: label the points which have less than **N** neighbors in the set of points of all trips within a **δ** radius as outliers, and discard them.*

- Test it on SF taxi data

Main Material

- [chapter] Textbook [Human Mobility Science](#)
 - [chapter 4 \(Data Preprocessing\)](#).

Auxiliary material

- [paper] [Review and classification of trajectory summarisation algorithms: From compression to segmentation](#)
- [paper] [Algorithms for the reduction of the number of points required to represent a digitized line or its caricature \(Douglas-Peucker\)](#)
- [paper] [A Trajectory Segmentation Map-Matching Approach for Large-Scale, High-Res GPS Data](#)
- [paper] [Hidden Markov Map Matching Through Noise and Sparseness](#)