
# Mobility Data Management & Exploration

## Ch. 01. Introduction

**Nikos Pelekis & Yannis Theodoridis**

InfoLab | University of Piraeus | Greece
infolab.cs.unipi.gr

v.2014.05

*“Τά πάντα ρεί, μηδέποτε κατά τ’ αυτό μένειν –*
*Everything changes, nothing remains still.”*
*Heraclitus*

# Chapter outline

1.1. About Mobility Data

1.2. What Can We Learn from Mobility Data

1.3. Location- and Mobility-Aware Applications

1.4. Adding Mobility in Spatial Database Systems

1.5. Summary

# 1.1.

# About Mobility Data

# Mobile devices and services

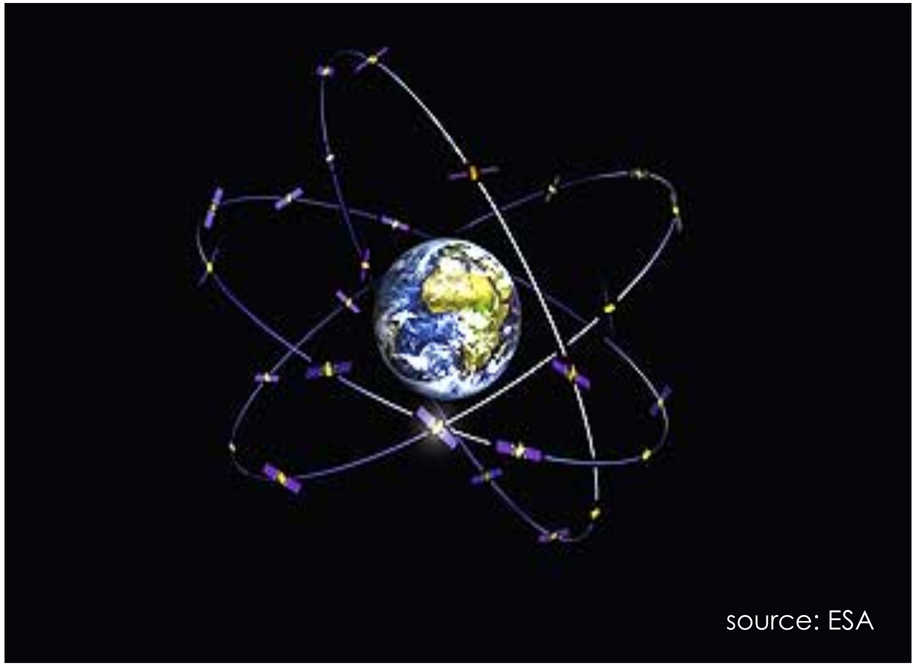
- Large diffusion of mobile devices, mobile services and location-based services ➔ **location- and mobility-aware data**

# Which data?

- Location data from mobile phones
  - i.e., cell positions in the GSM/UMTS network
- **Location (and trajectory) data from GPS devices**
  - Humans (pedestrians, drivers) carrying GPS-equipped smartphones
  - Vessels equipped with AIS transmitters (due to maritime regulations)
- Location data from indoor posistioning systems
  - RFIDs (radio-frequency ids)
  - Wi-Fi access points
  - Bluetooth sensors

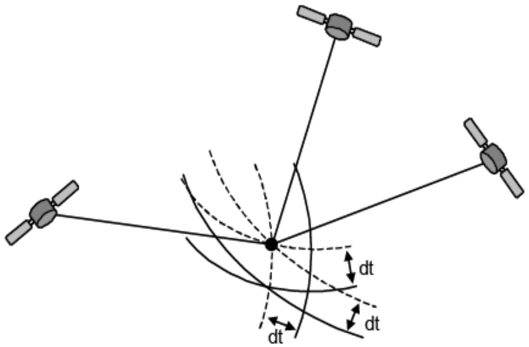
# Geo-positioning

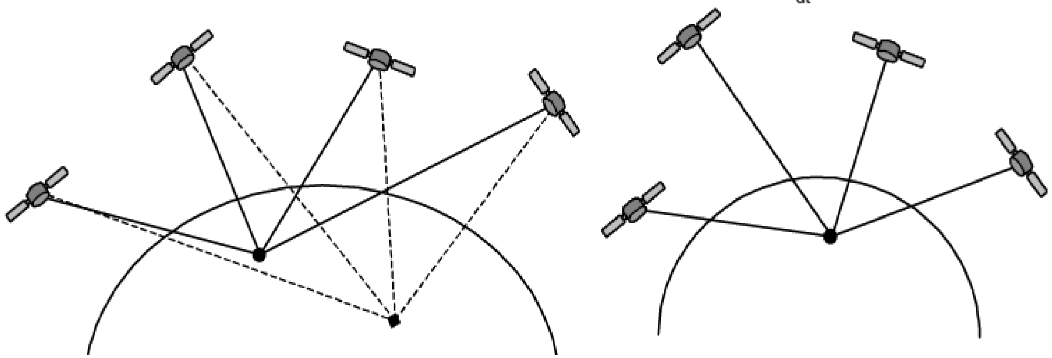
- Positioning technologies using information from satellites
  - **Global Positioning System** (GPS) and its variations …
  - **Assisted GPS** (A-GPS); **Differential GPS** (D-GPS)



# Geo-positioning (cont.)

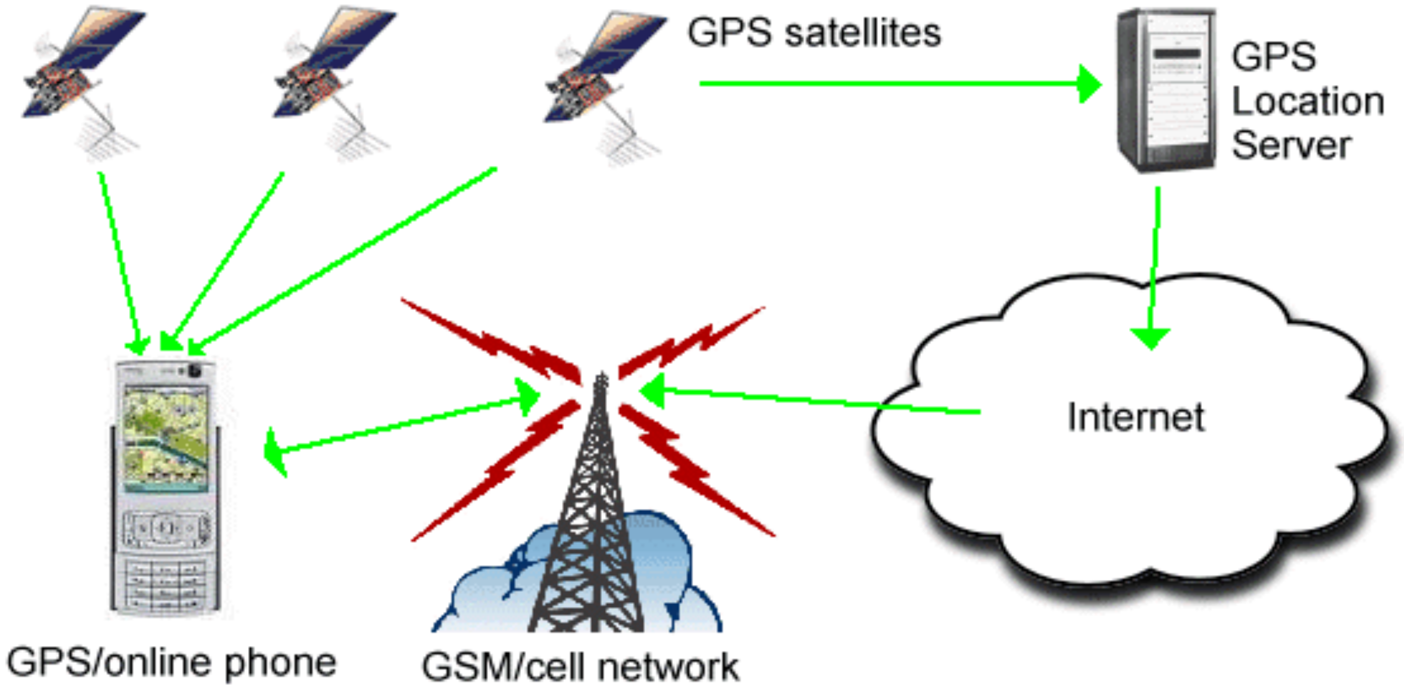
- **GPS (Global Positioning System)**
  - 24-satellite constellation
    - monitored by 5 monitoring stations and 4 ground antennas; handled with (extremely precise) atomic clocks
    - At least 5 satellites are in view from every point on the globe
  - GPS receiver gathers information from 4 (or 3, the minimum) satellites and
    - triangulates to position itself;
    - fixes its (non-atomic) clock
  - Position accuracy: ~20m



# Geo-positioning (cont.)

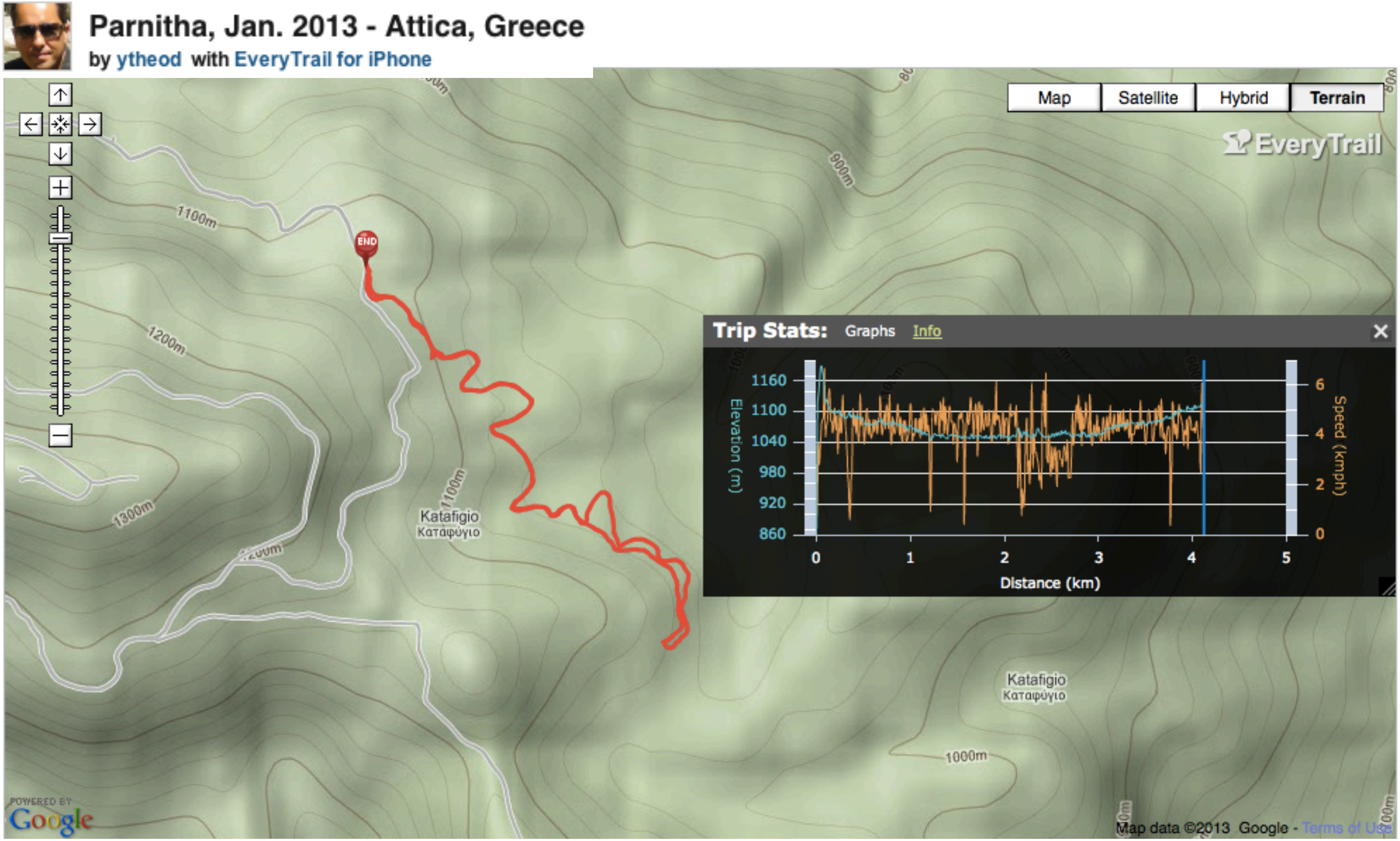
- GPS accuracy improvements
  - **Assisted GPS (A-GPS)**: provides pre-calculated satellite orbits to the receiver; accuracy ~10m
  - **Differential GPS (D-GPS)**: combines with land (antennas) information; accuracy down to 1m



# Geo-positioning (cont.)

- GPS competitors
  - **Glonass** (Russia) – currently, semi-operational
    - 24-satellite constellation; 1-10m accuracy
  - **Galileo** (EU) – to be fully operational by 2019
    - 30-satellite constellation; 1m accuracy
  - **Beidou** (China) – to be fully operational by 2020
    - 35-satellite constellation; 10m accuracy

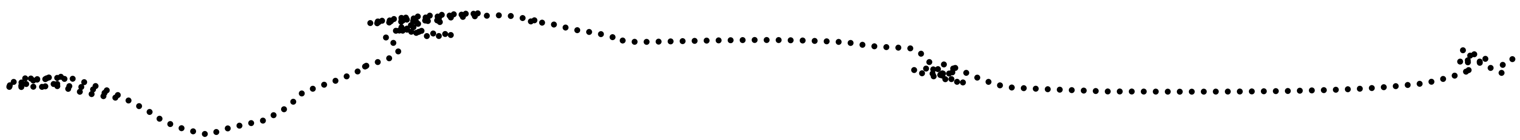
# GPS data – an example



# GPS data – format

- Raw data: GPS recordings (.gpx format)

```
<trk>
...

<trkpt lat="38.17733919" lon="23.74038222”> <ele>862.62</ele>
        <time>2013-01-19T08:54:57.608Z</time> </trkpt>

<trkpt lat="38.17725880" lon="23.74043843"> <ele>1117.98</ele>
        <time>2013-01-19T08:55:21.609Z</time> </trkpt>

<trkpt lat="38.17717291" lon="23.74039676"> <ele>1129.98</ele>
        <time>2013-01-19T08:55:31.608Z</time> </trkpt>

<trkpt lat="38.17707471" lon="23.74038878"> <ele>1155.93</ele>
        <time>2013-01-19T08:55:45.584Z</time> </trkpt>

...
</trk>
```



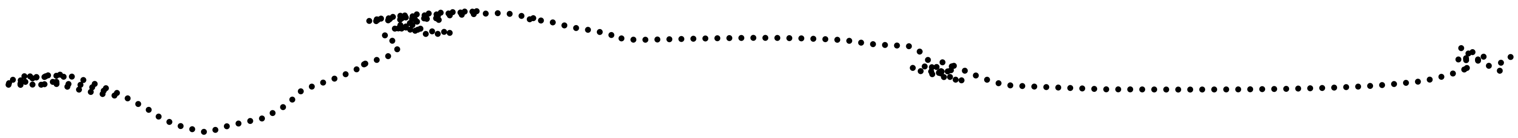
# GPS data – format (cont.)

- Where and when?

```
<trkpt lat="38.17733919" lon="23.74038222">
<time>2013-01-19T08:54:57.608Z</time>
```

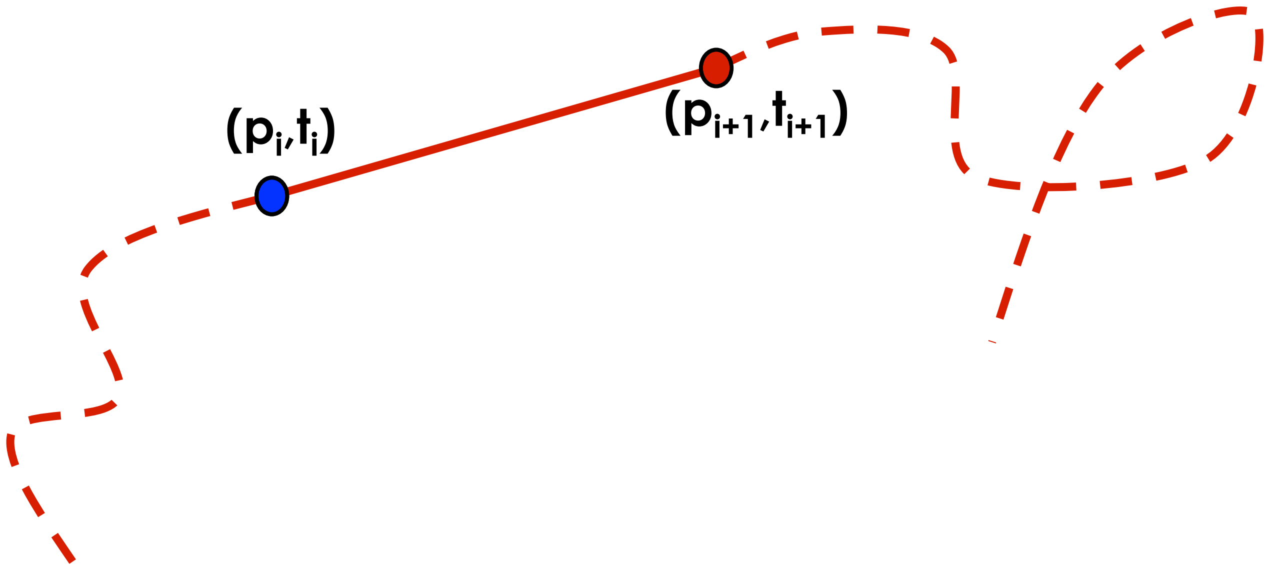
# From GPS data to trajectories

- A **trajectory** is a model for a motion path of a moving object (human, animal, robot, …)
  - (due to discretization) a sequence of **sampled time-stamped locations** $(p_i, t_i)$ where:
    - $p_i$ is a 2D point $(x_i, y_i)$ and
    - $t_i$ is the recording timestamp of $p_i$



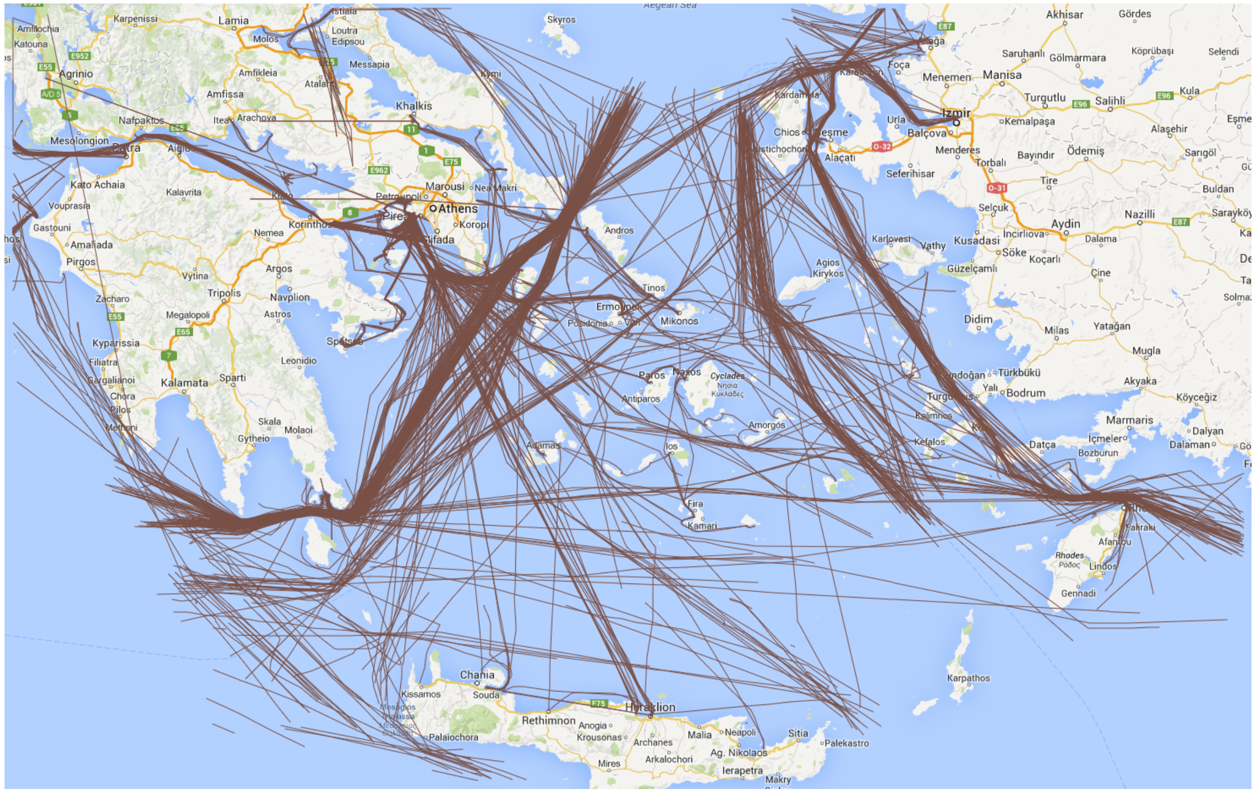
# From GPS data to trajectories (cont.)

- A common representation of a trajectory is a **3D polyline** whose vertices correspond to time-stamped locations $(p_i, t_i)$
  - Usually, **linear interpolation** is assumed between $(p_i, t_i)$ and $(p_{i+1}, t_{i+1})$



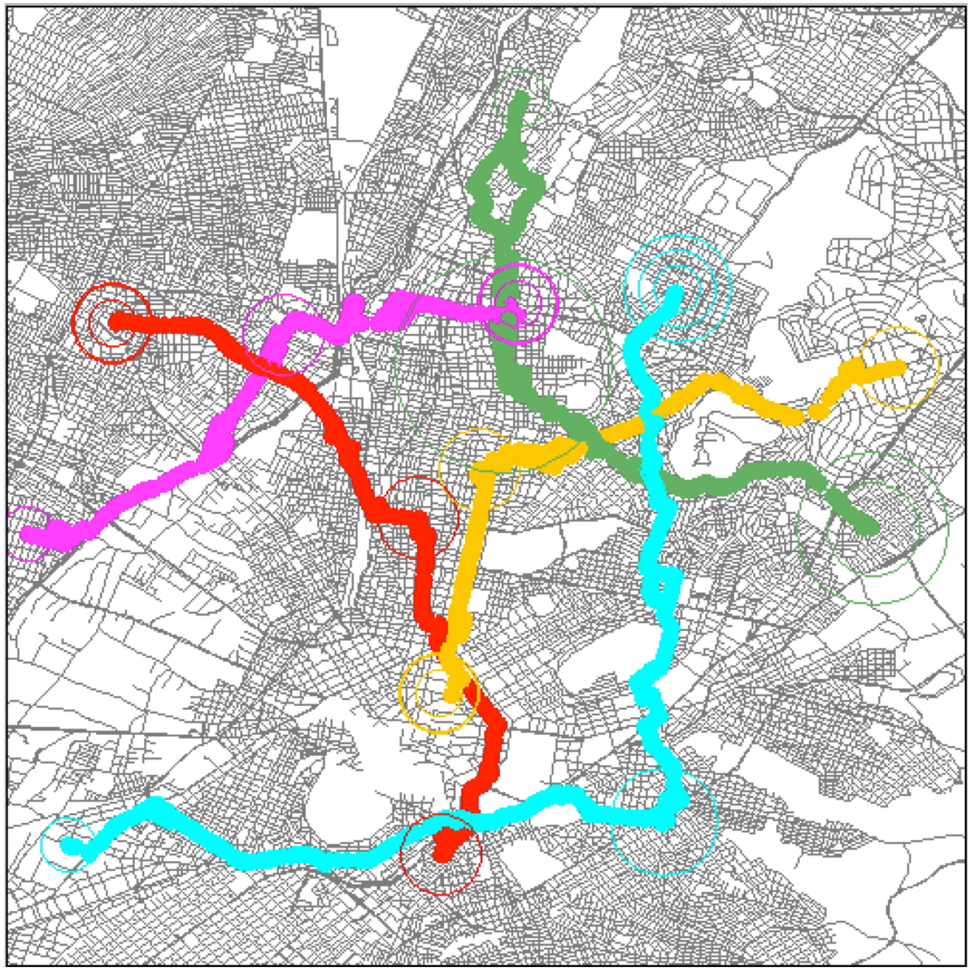
# Examples of trajectory datasets

- **Vessels sailing on the sea**
  - Example: “IMIS%” datasets, available at ChoroChronos.org
  - IMIS-3-days statistics:
    - ~3 million GPS recordings from 933 vessels sailing in Aegean sea during a 3 days period
    - 1500 trajectories



# Examples of trajectory datasets

- **Vehicles moving on a road network**
  - Example: "Attiki" dataset, available at http://infolab.cs.unipi.gr/hermes/
  - Produced by Hermoupolis data generator
  - Statistics:
    - ~1.5 million GPS recordings of vehicles moving in Athens metropolitan area road network
    - ~4,500 trajectories



# 1.2.

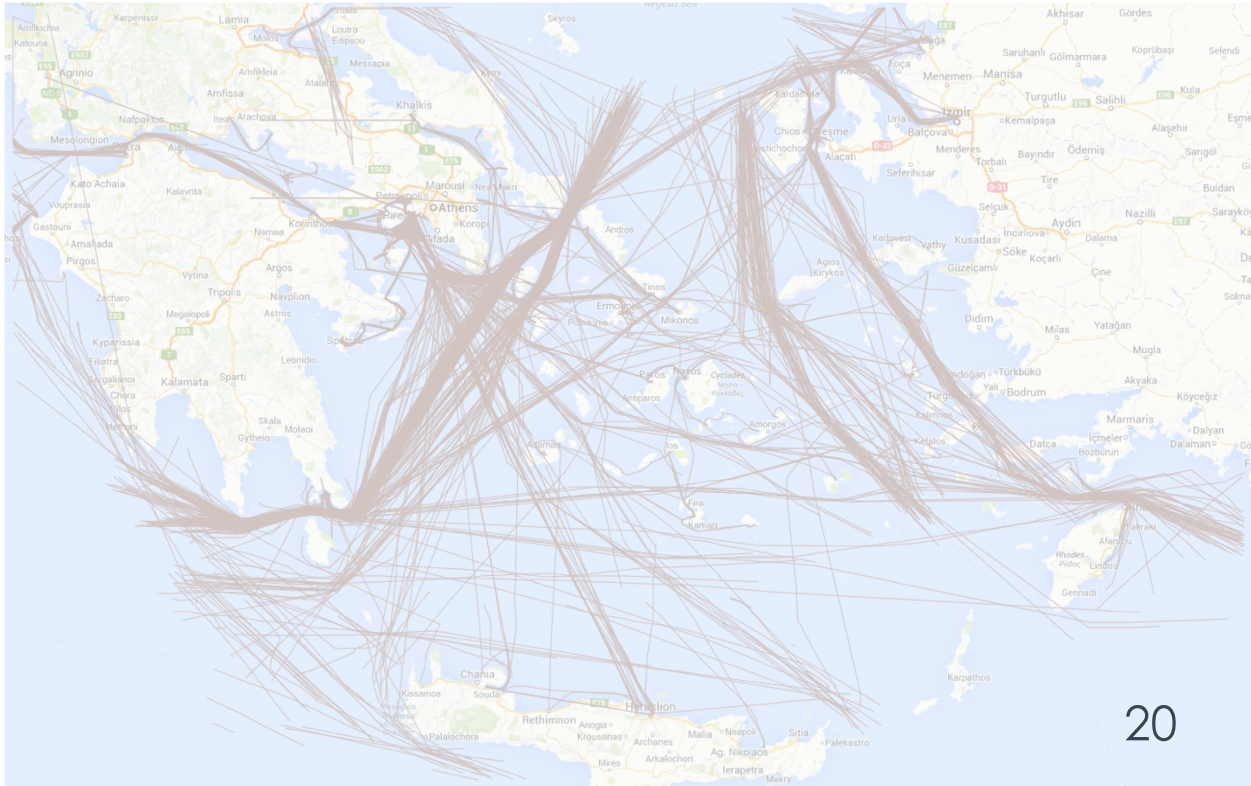
# What Can We Learn from Mobility Data

# Learning from vessels datasets

- Analysis at **individual level** (i.e. per vessel):
  - Draw a vessel track (detailed vs. simplified)
  - Calculate similarity between a vessel's actual and expected route
  - Calculate minimum distance between a vessel's track and the shore (where and when)
  - Calculate maximum number of vessels in a vessel's vicinity (e.g. 10 n.m. radius)
  - Find whether (and how many times) a vessel sailed through narrow passages or biodiversity boxes
  - Find whether (and how many times) a vessel has performed sharp changes in its direction
  - etc.

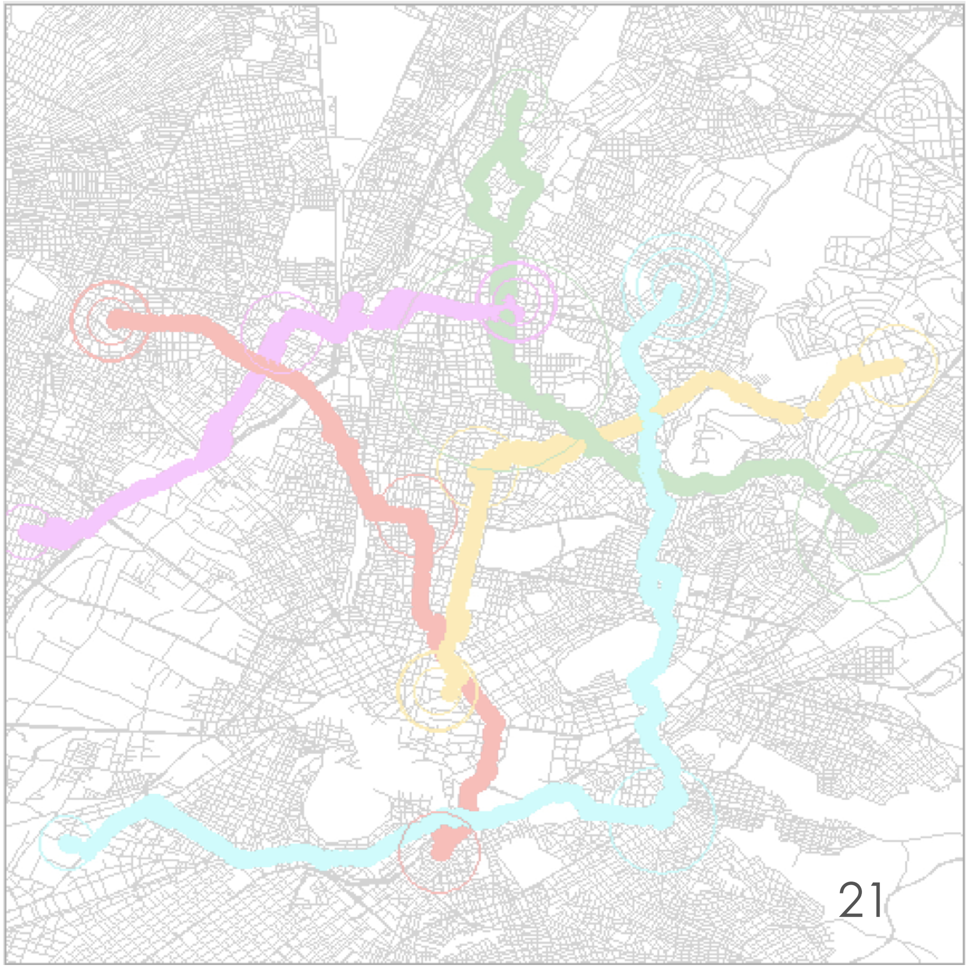
# Learning from vessels datasets (cont.)

- Analysis at **collective level** (i.e. per population of vessels)
  - Find the most popular among vessels' routes as well as the outliers
  - Calculate the Origin-Destination (OD) matrix
  - Calculate the population's environmental fingerprint
  - Search for correlation between flags of convenience and outliers
  - etc.



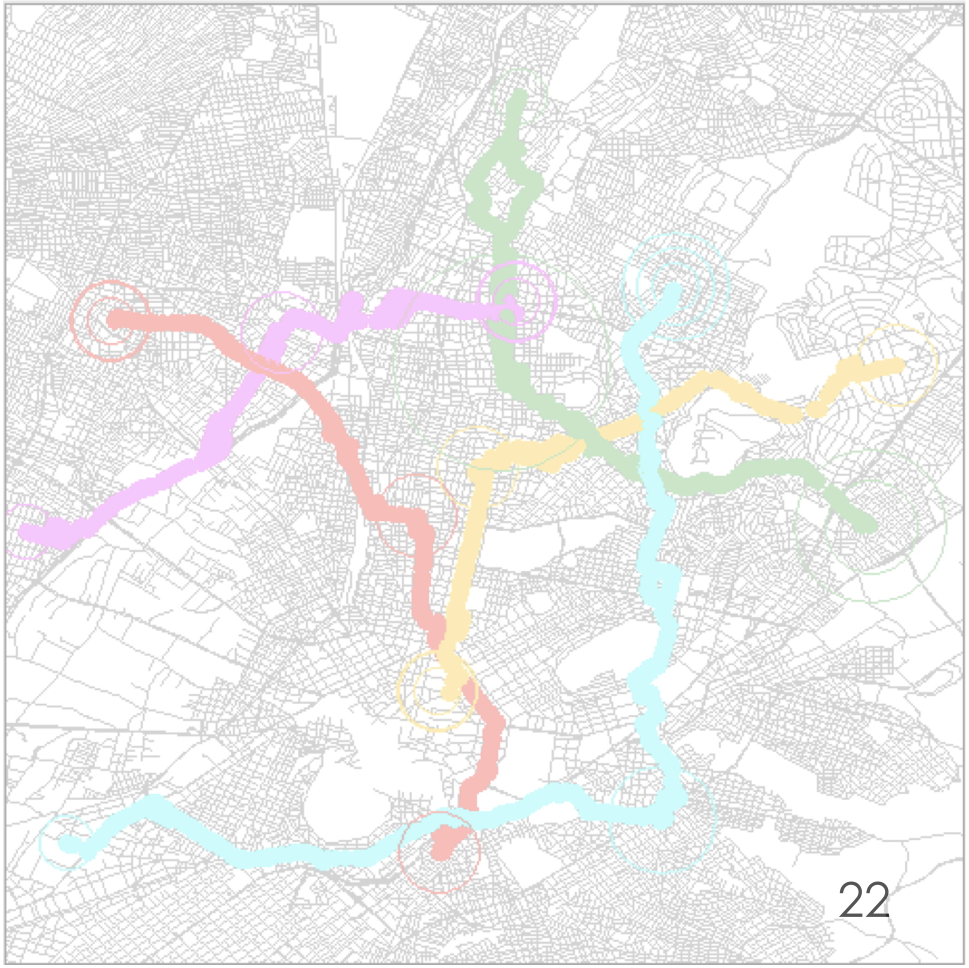
# Learning from vehicles datasets

- (holistic) **Traffic analysis**
  - How many cars are currently moving on the ring of the town?
  - What is the queue per traffic light?
  - Does the “green wave” between traffic lights appear as it has been designed?
  - Which cars follow eco-driving recommendations?



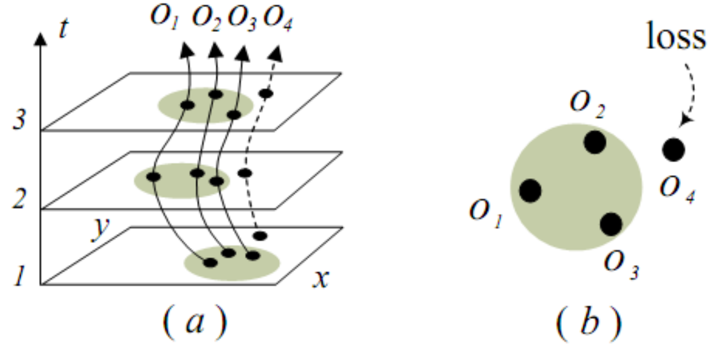
# Learning from vehicles datasets (cont.)

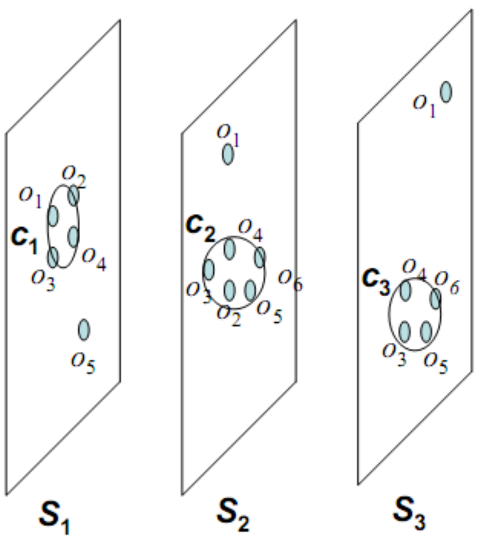
- (personalized) **Location-aware services**
  - Where is my nearest restaurant?
  - Which gas stations are at a maximum of 3 km deviation from my planned trip?
  - Are there any Facebook friends around?

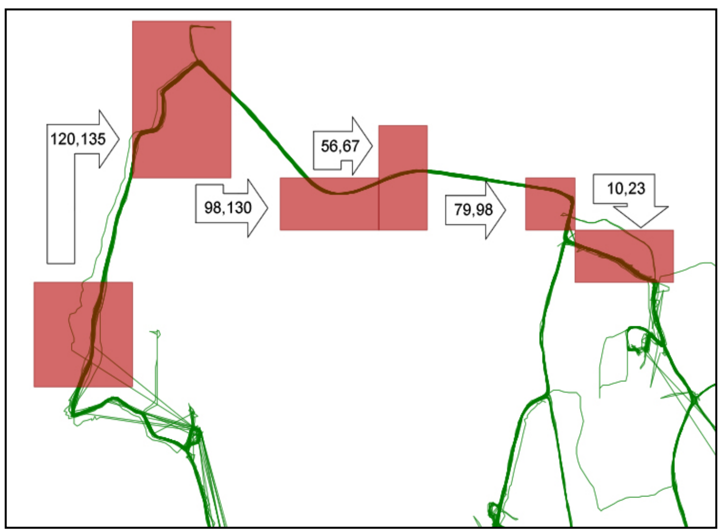


# Learning from mobility data

- More ambitious patterns, extracted by KDD techniques. Examples:
  - **Typical routes** (clusters vs. outliers): T-Patterns, hot motion paths, etc.
  - **Flows of movement**: moving clusters, flocks, convoys, etc.



# 1.3.

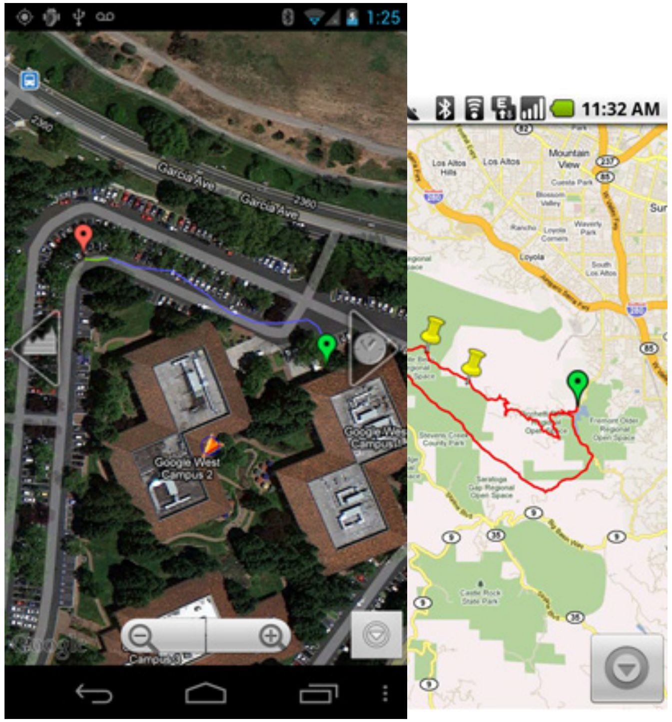
# Location- and Mobility-Aware Applications

# Location-based services (LBS)

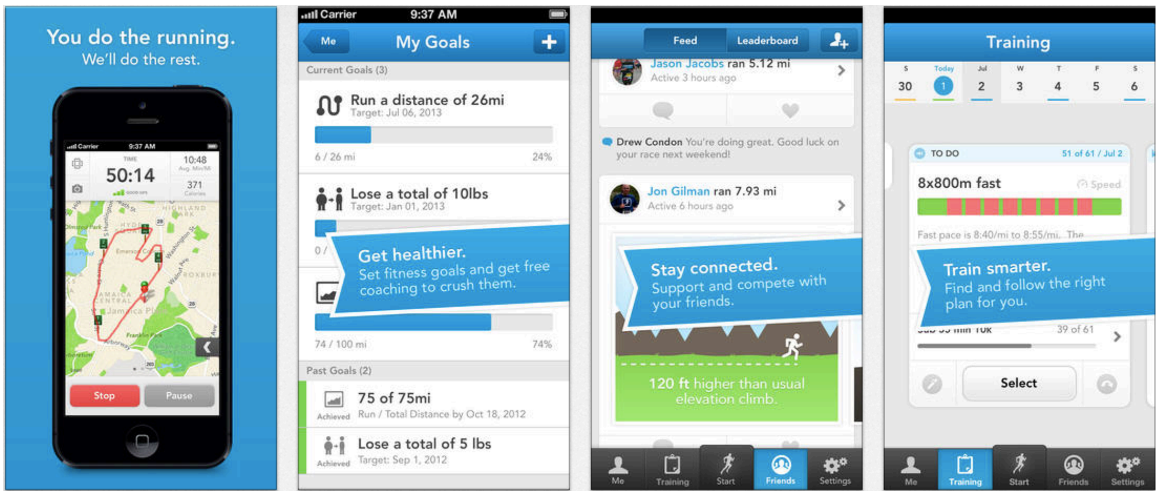
- **Navigation** (vehicle or pedestrian) & **Location-aware information**
  - Routing, finding the nearest point-of-interest (POI)
  - Location-based yellow pages ("what-is-around", …)
- **Resource management** & **Tracking**
  - (taxis, trucks, etc.) fleet management, administration of container goods, location-based charging
  - Tracing of a stolen car, locating persons in an emergency situation, …
- **Location-aware social networking**
  - Google Latitude, Facebook places, Foursquare, etc.

# Commercial examples


**EveryTrail**



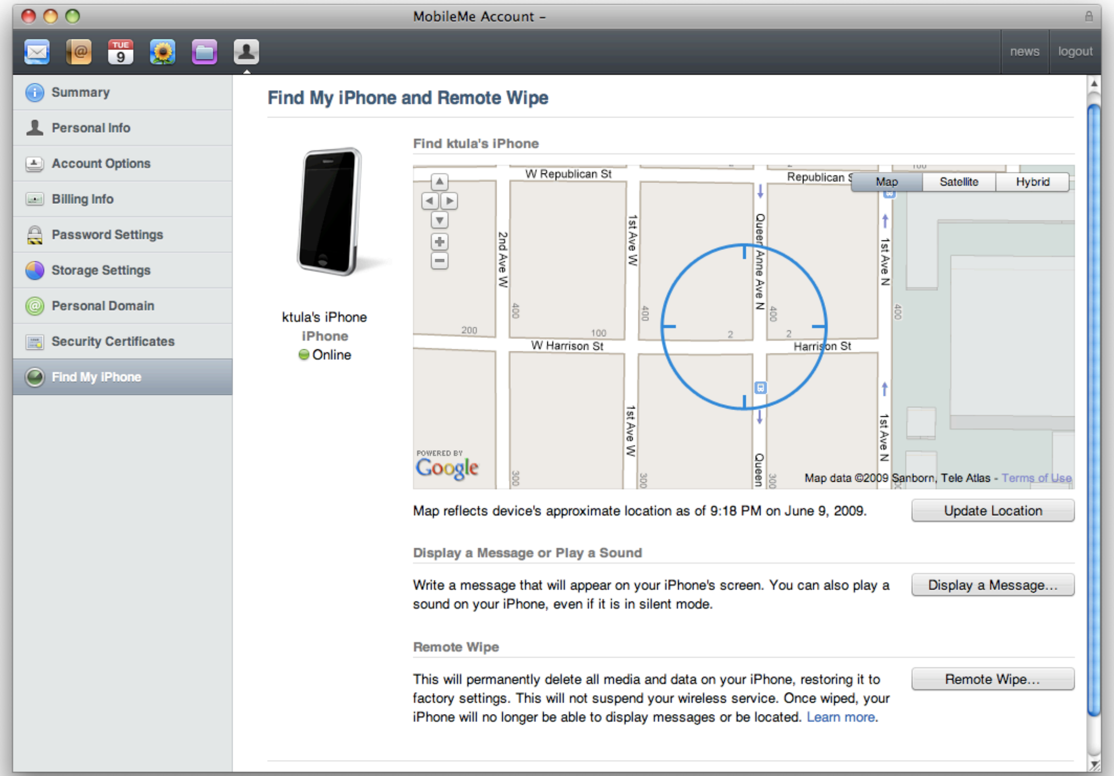

**Google My Tracks**




**RunKeeper**

# Commercial examples (cont.)

- **Tracing** lost or stolen products
  - Lost smartphones: “Find my iPhone”, “Where’s my Droid” etc.
  - Stolen cars: “Volvo connectivity”

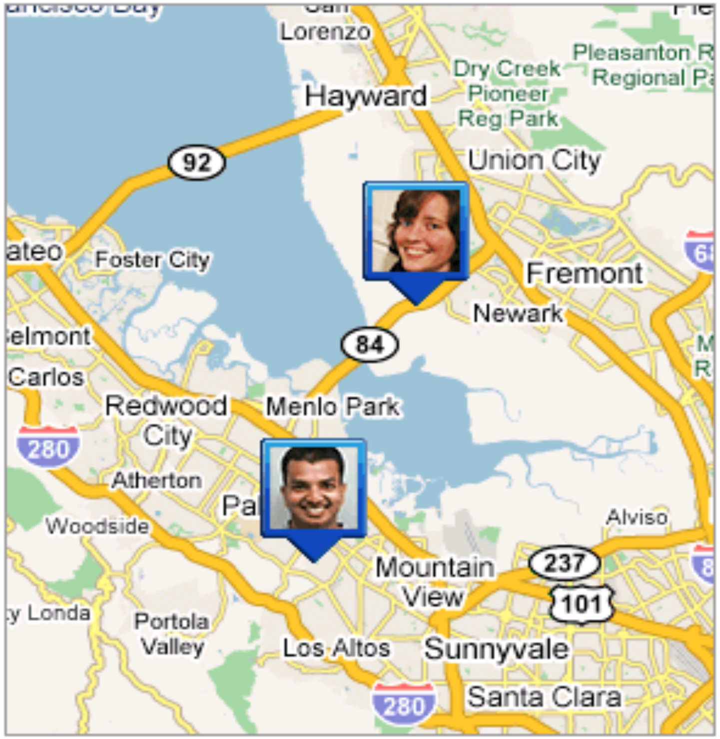




# Commercial examples (cont.)

- Google Latitude

See in real time **where** your friends are! (launched Feb.09 by Google)




# Commercial examples (cont.)

Google places

- **Google places**: Rate and share places on Google
  - **Google places for business**:
    - (business perspective) Get your business found on Google
    - (end user's perspective) rate products, search for similar, compare prices, etc.

# Commercial examples (cont.)

facebook

Facebook Places

Who. What. When. And now **where.**


- Tag yourselves and find tagged friends


- Tell your friends where you are, suggest places, etc.

# Taxonomy of location-aware apps

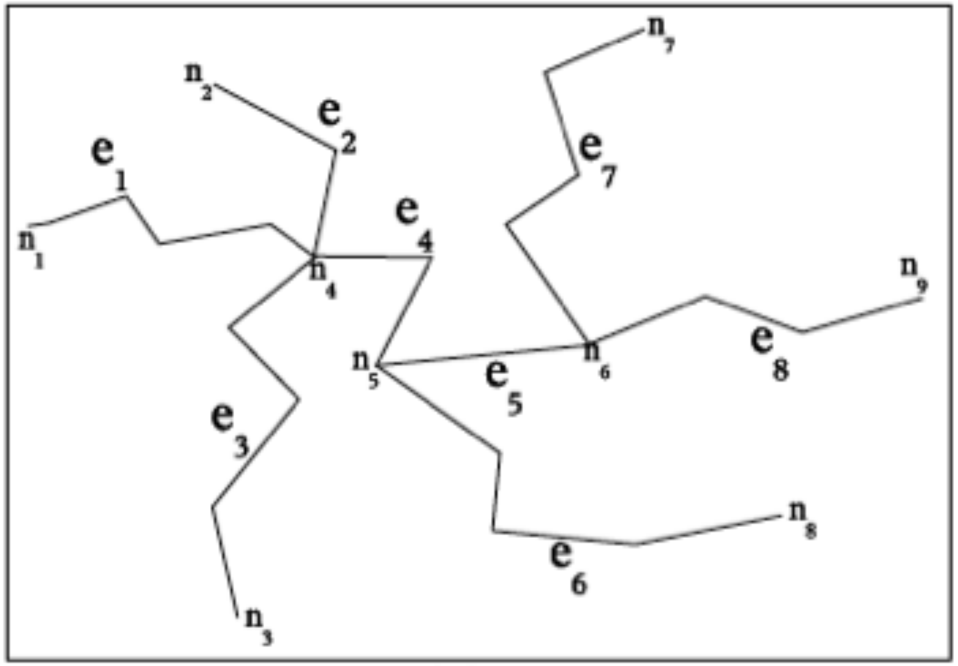
- With respect to mobility of involved objects

| Reference object (user) / Database objects | stationary | mobile |
|---|---|---|
| **stationary** | ⇒ Routing (& constrained routing)<br>⇒ What-is-around<br>⇒ Find-the-nearest | ⇒ Time-dependent routing<br>⇒ Guide-me |
| **mobile** | ⇒ Find-me | ⇒ Get-together |

**Interesting: the vast majority of current commercial apps are classified as stationary**

# (underlying) Transportation network

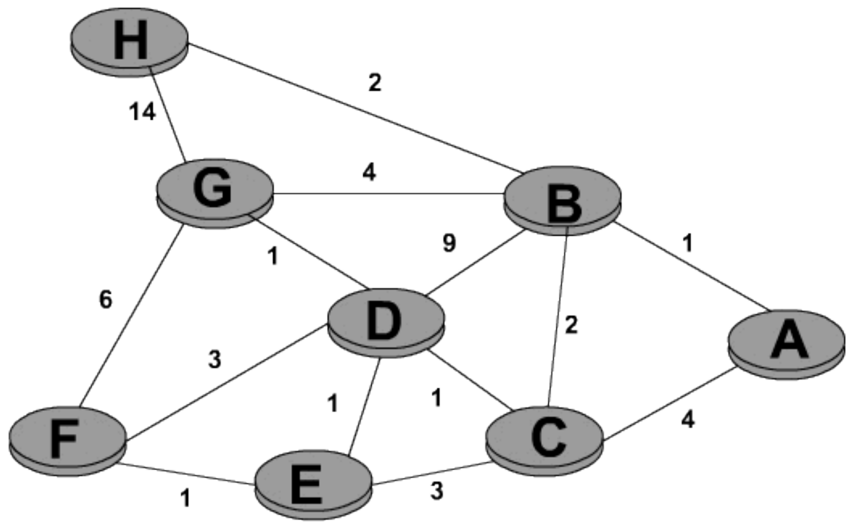
- Transportation networks are usually modeled as directed graphs G(N, E)
  - N: set of nodes
  - E: set of edges between nodes
- Alternative models*:
  - **Edge-oriented** model
    - edges <$e_1$>, <$e_2$, $e_3$>, etc.
  - **Route-oriented** model
    - Routes <$e_2$, $e_3$>, etc.




* See more details in Chap. 3

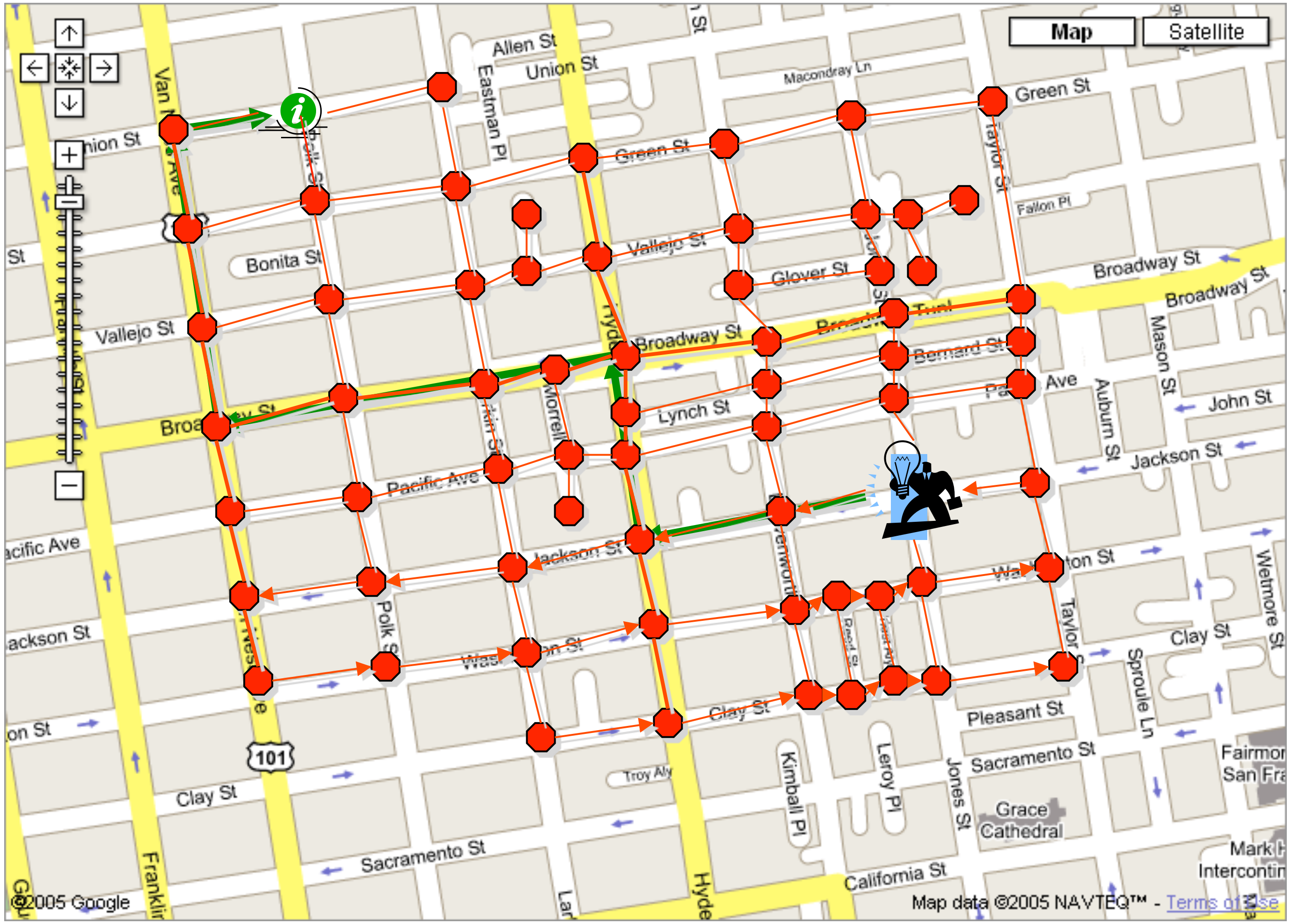
# “Routing”

- (aka “Get Directions” in Google maps)
- Find the optimal route from a departure to a destination point
  - Goal to be optimized: the minimal network distance traveled; the minimal traveled time; etc.
- Relates to the well-known **shortest path (SP)** problem from graph theory and network analysis
  - Several off-the-shelf solutions, e.g. Dijkstra’s single-start SP algorithm
  - Recall the SP problem: “find the SP from A to D”



# "Routing" (cont.)

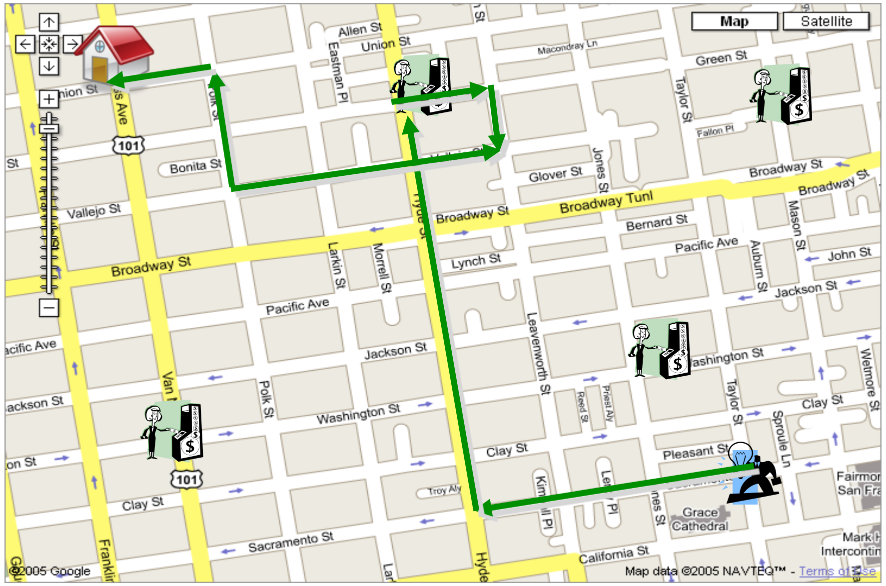
- Example:



# "Constrained routing"

(currently, not supported in Google maps!)

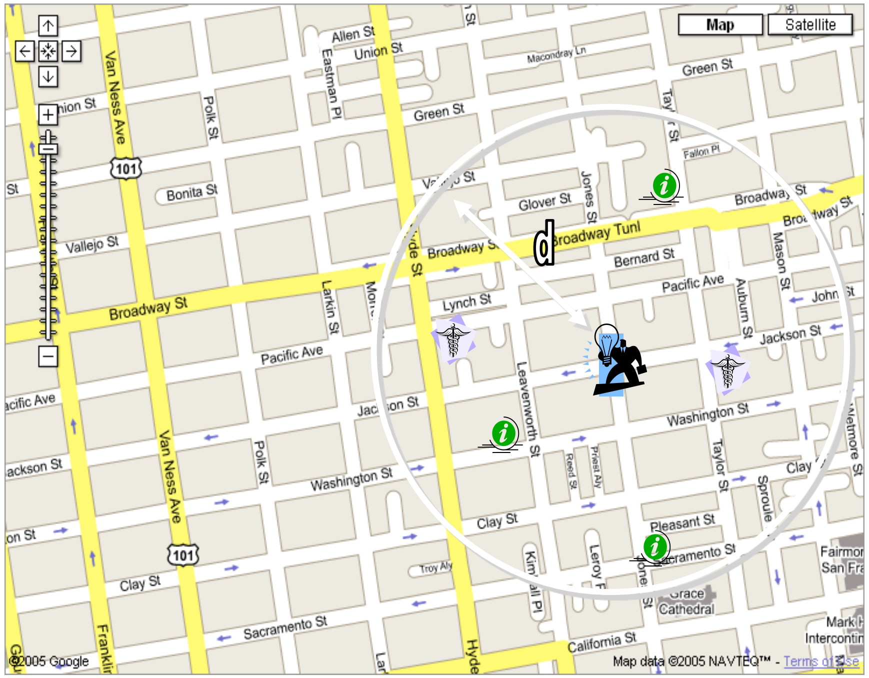
- Find the optimal route from a departure to a destination point …
  - … with the extra constraint to pass through one among a specified set of candidate points
  - Example: "Find the best route office-to-home constrained to pass from a bank ATM "
  - Issue: how to find (efficiently) the best deviation?
  - Technically, a variation of SP problem*



* See more details in Chap. 3

# "What-is-around"

- (aka "Search nearby" in Google maps)
- Retrieve and display all points of interest (POI) located in the surrounding area
  - Usually rectangular or circular area
  - Example: "Find all gas-stations within a distance of 1km from my current location"
  - As simple as a typical **range query*** in Spatial Databases

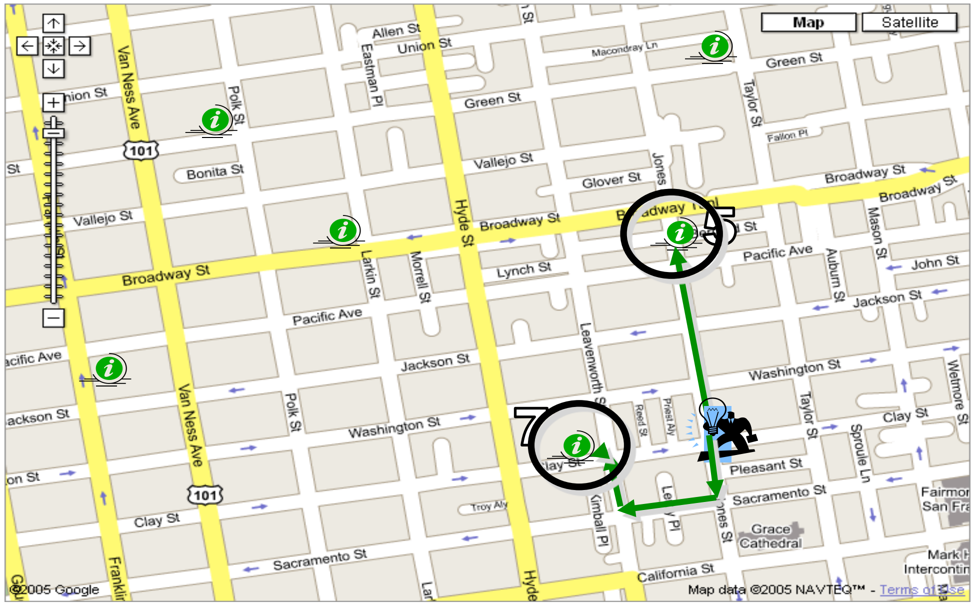


* See more details in Chap. 2

# "Find-the-nearest"

(currently, not supported in Google maps!)

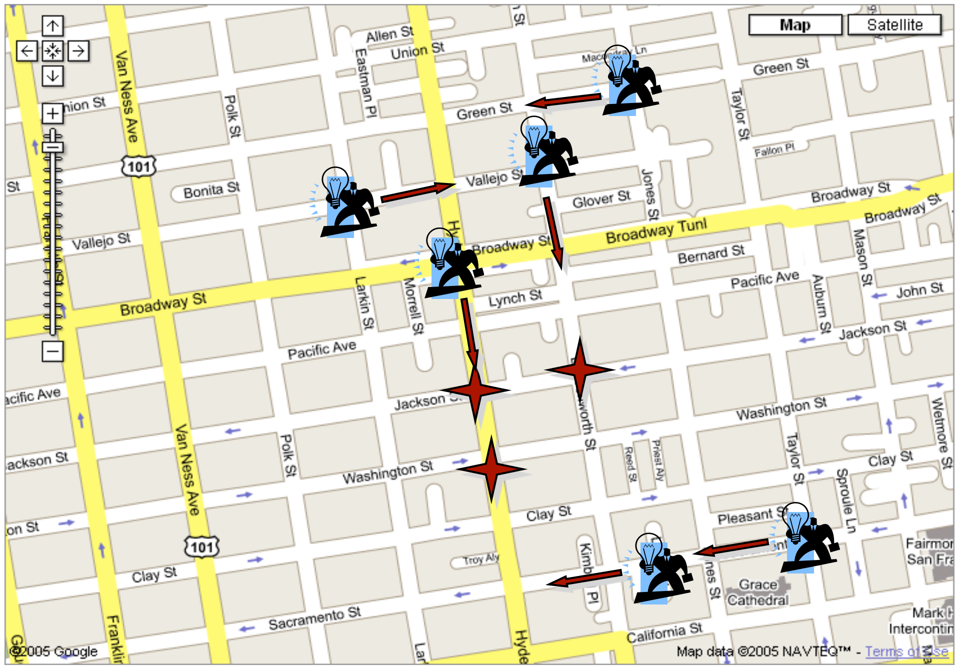
- Retrieve and display the nearest POI (restaurants, museums, etc.) with respect to a reference location
  - Example: "find the nearest bank ATM w.r.t. my current location"
  - Issue: distance is computed over the network rather than the free (Euclidean) space
  - How to solve it efficiently?*




* See more details in Chap. 3

# "Find me" / "Get-together"

- Several **moving** objects try to reach a **stationary** / **moving** object
  - Could be an advanced "Google Latitude" or "Facebook Places" app
- How-to:
  - other objects are routed to a point where the target object is estimated to arrive after a time interval
  - the "meeting point" is periodically refreshed
  - Challenge: future location prediction*




* See more details in Chap. 7

# 1.4.
# Adding Mobility in Spatial Database Systems

# From the DBMS perspective …

- So far, we've outlined a variety of mobile-aware applications
  - from 'smart' (updated in real-time) routing …
  - … to mobile social networking (Google Latitude, Facebook Places, etc.)

- Internally (either at server or client side), what kinds of **operations** should be **efficiently** supported?
  - Spatial database and graph-based operations
  - Trajectory management operations

# Operations to be supported

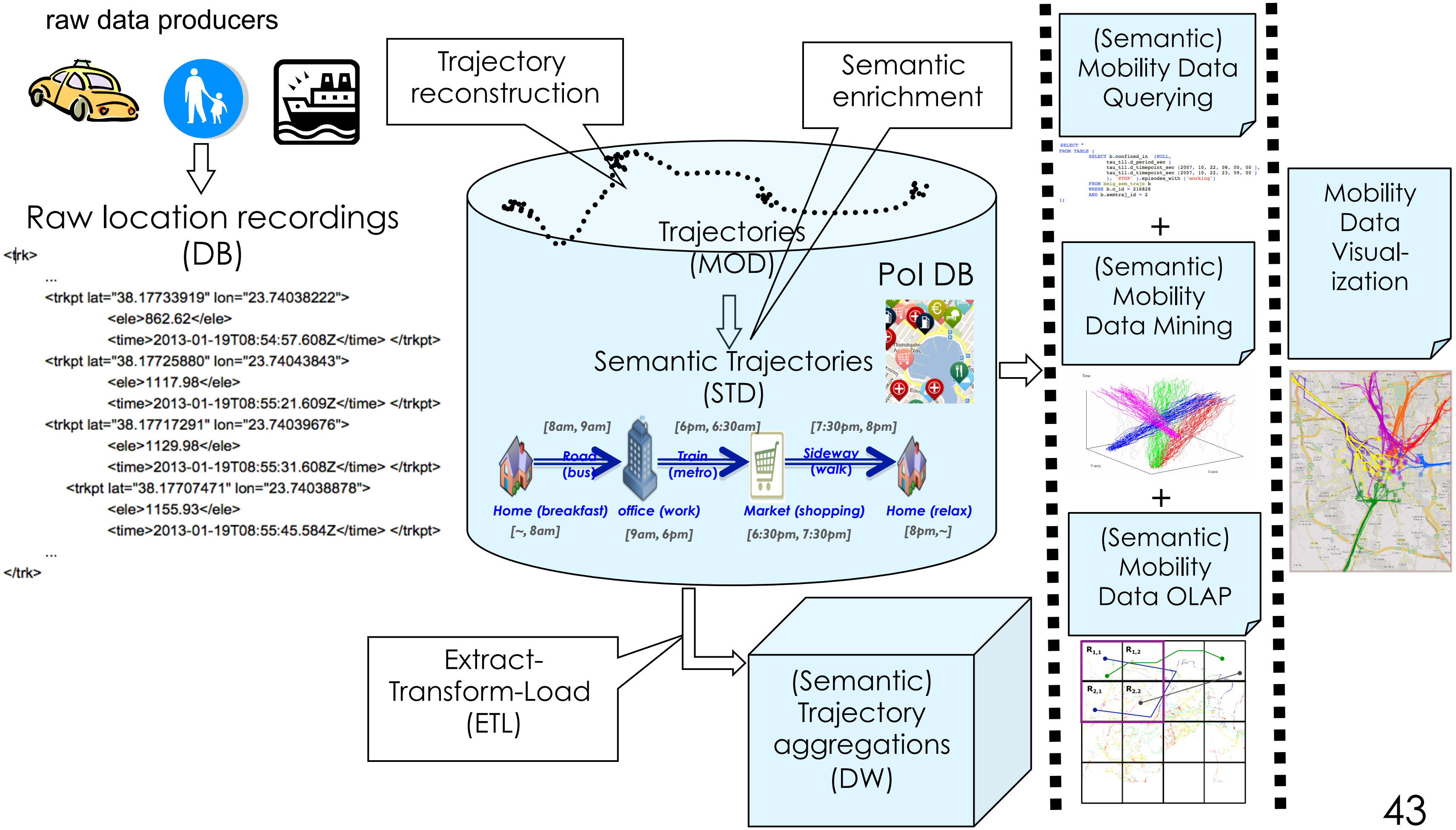
- Spatial database and graph-based operations
  - **Window search**: select points within a (circular or rectangular) window
  - **k-NN search**: select the k- nearest points to an object (point or region)
  - **Routing**: find the optimal route between two points (taking into consideration a number of constraints)
- Trajectory management operations
  - …

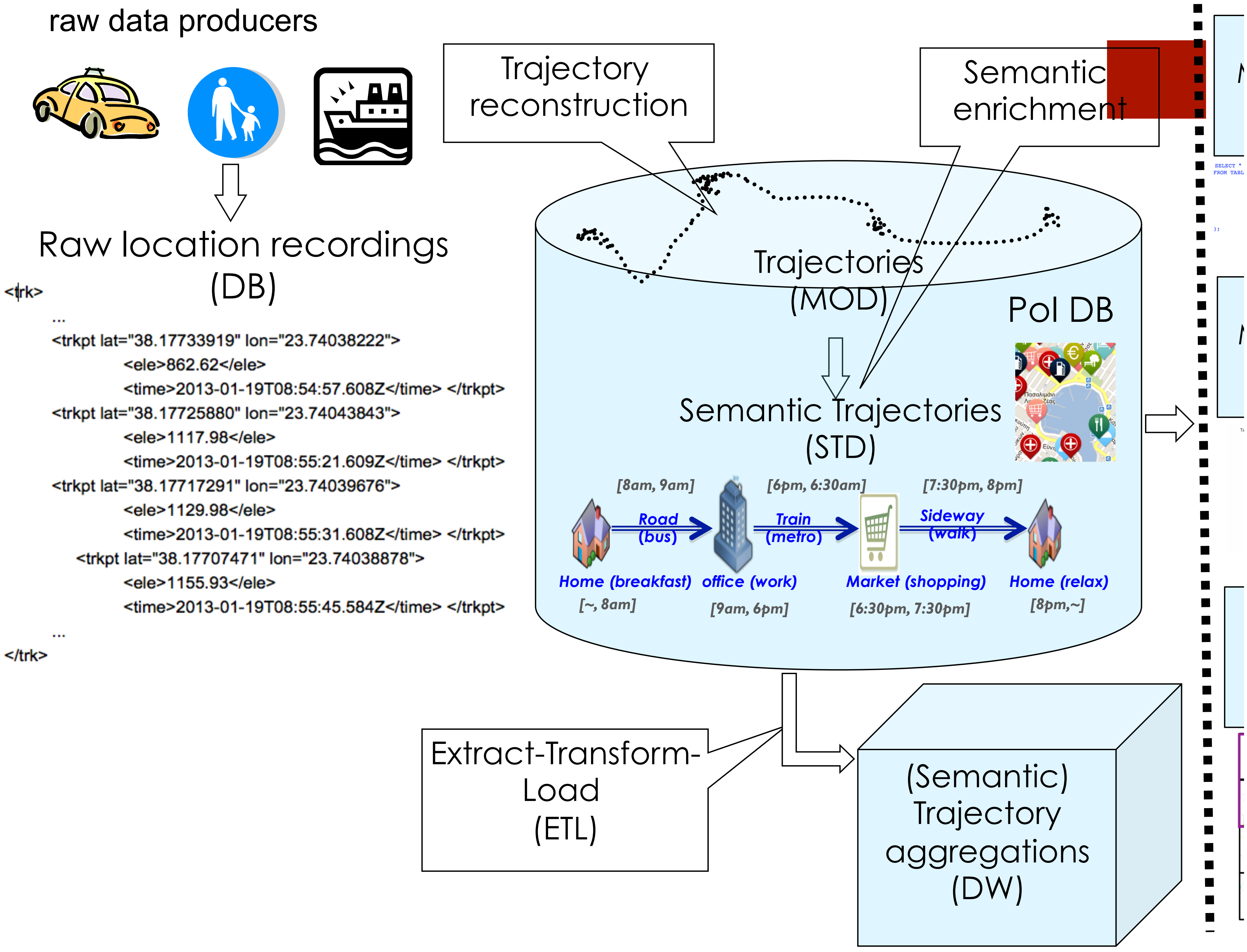
# Operations to be supported (cont.)

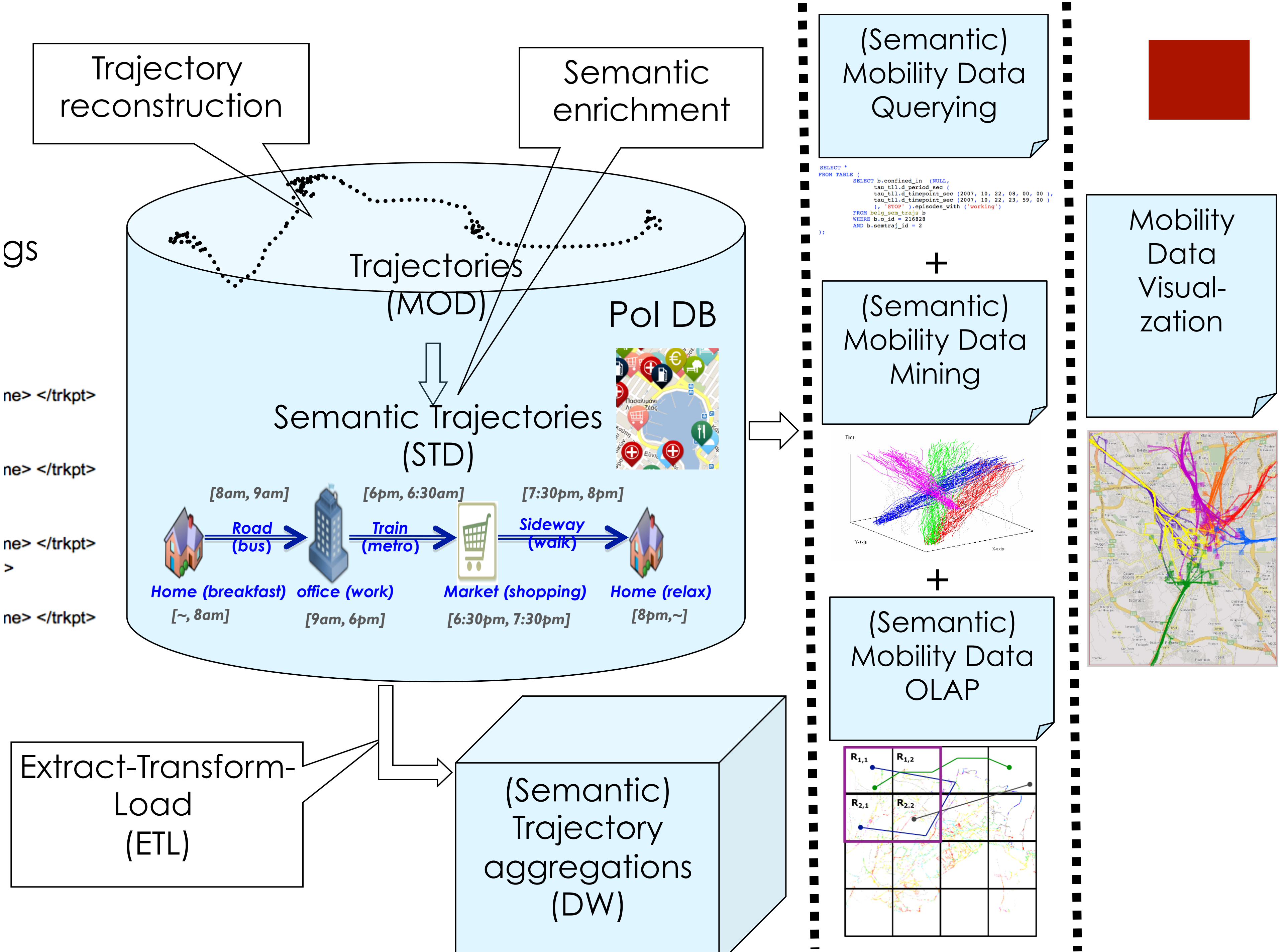
- Spatial database and graph-based operations
- Trajectory management operations
  - **Trajectory update (& map-matching)**: add a new coming position to an existing trajectory (mapped to the underlying network)
  - **Trajectory-based search**: perform (spatial and/or temporal) range, NN, trajectory similarity, etc. search
  - **Trajectory projection**: estimate the anticipated position of an object at a future time (perhaps, taking its history into consideration)

# The big picture

- raw GPS data → (raw & semantic) trajectories → mobility patterns




raw data producers
Raw location recordings (DB)
<trk>
...
<trkpt lat="38.17733919" lon="23.74038222">
<ele>862.62</ele>
<time>2013-01-19T08:54:57.608Z</time> </trkpt>
<trkpt lat="38.17725880" lon="23.74043843">
<ele>1117.98</ele>
<time>2013-01-19T08:55:21.609Z</time> </trkpt>
<trkpt lat="38.17717291" lon="23.74039676">
<ele>1129.98</ele>
<time>2013-01-19T08:55:31.608Z</time> </trkpt>
<trkpt lat="38.17707471" lon="23.74038878">
<ele>1155.93</ele>
<time>2013-01-19T08:55:45.584Z</time> </trkpt>
...
</trk>
Trajectory reconstruction
Semantic enrichment
Trajectories (MOD)
PoI DB
Semantic Trajectories (STD)
[8am, 9am]
[6pm, 6:30am]
[7:30pm, 8pm]
Road (bus)
Train (metro)
Sideway (walk)
Home (breakfast)
office (work)
Market (shopping)
Home (relax)
[~, 8am]
[9am, 6pm]
[6:30pm, 7:30pm]
[8pm,~]
Extract-Transform-Load (ETL)
(Semantic) Trajectory aggregations (DW)


gs
ne> </trkpt>
ne> </trkpt>
ne> </trkpt>
>
ne> </trkpt>
Trajectory reconstruction
Semantic enrichment
Trajectories (MOD)
PoI DB
Semantic Trajectories (STD)
[8am, 9am]
[6pm, 6:30am]
[7:30pm, 8pm]
Road (bus)
Train (metro)
Sideway (walk)
Home (breakfast)
office (work)
Market (shopping)
Home (relax)
[~, 8am]
[9am, 6pm]
[6:30pm, 7:30pm]
[8pm,~]
Extract-Transform-Load (ETL)
(Semantic) Trajectory aggregations (DW)
(Semantic) Mobility Data Querying
SELECT *
FROM TABLE (
SELECT b.confined_in (NULL,
tau_tll.d_period_sec (
tau_tll.d_timepoint_sec (2007, 10, 22, 08, 00, 00 ),
tau_tll.d_timepoint_sec (2007, 10, 22, 23, 59, 00 )
), 'STOP' ).episodes_with ('working')
FROM belg_sem_trajs b
WHERE b.o_id = 216828
AND b.semtraj_id = 2
);
+
(Semantic) Mobility Data Mining
+
(Semantic) Mobility Data OLAP
Mobility Data Visual-zation

# Key questions that arise

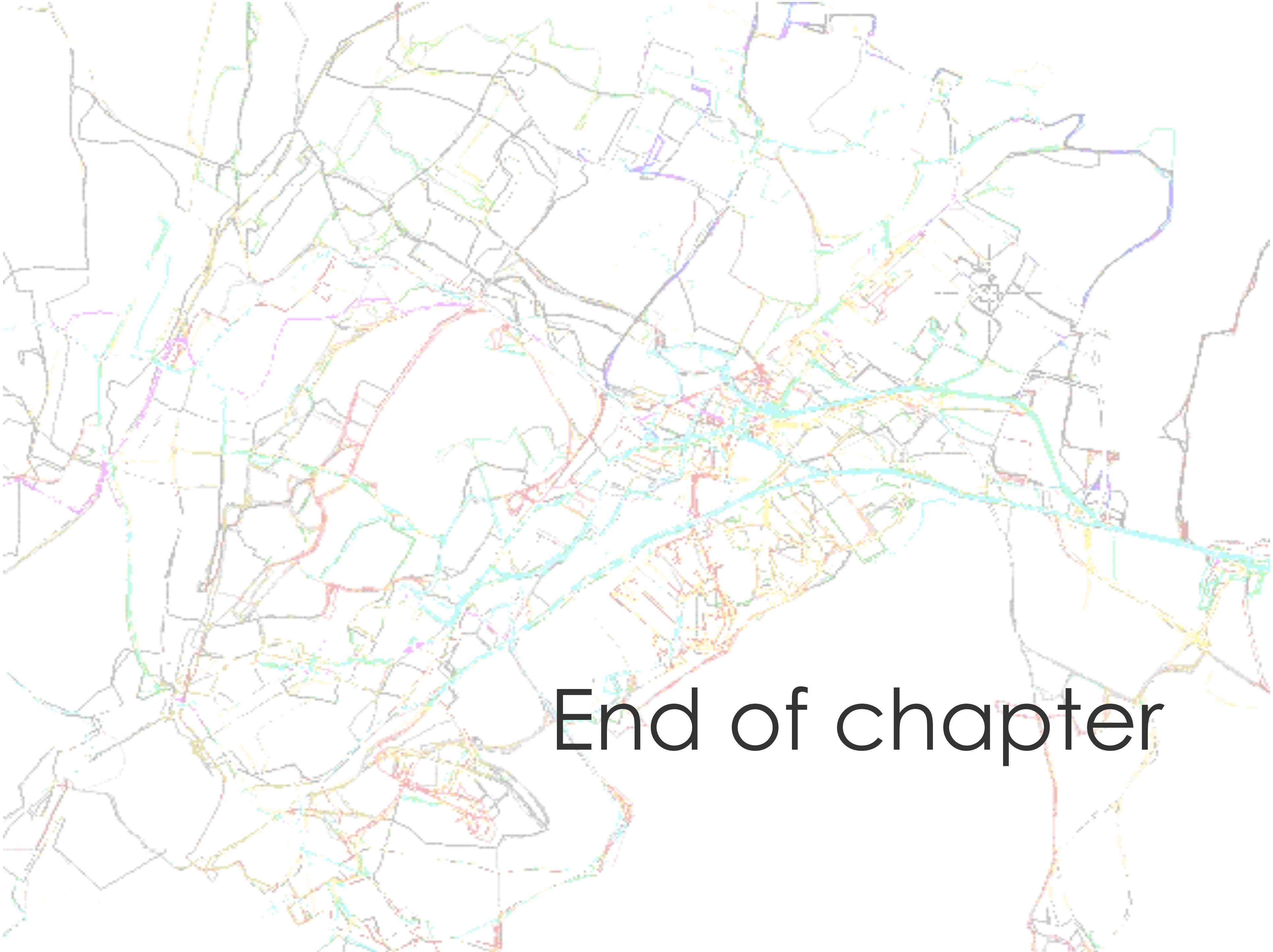
- How to **reconstruct trajectories** from raw logs?
- How to **store trajectories** in a DBMS?
- What kind of **analysis** is suitable for mobility data (in particular, trajectories of moving objects)?
  - How does infrastructure (e.g. road network) affect this analysis?
- Which **patterns / models** can be extracted from them?
  - Clusters, frequent patterns, anomalies / outliers, etc.
  - How to compute such patterns / models efficiently?
- How to **protect privacy – user anonymity**?
  - What is the trade-off between privacy protection and quality of analysis?
- Which are the **semantics** hidden into mobility data?
- What if **extremely large volumes** of data are collected?

# 1.5.

## Summary

# Summarizing …

- Mobility data management and exploration is a hot research topic
  - due to the wide spread of GPS devices
- In this chapter, we discussed:
  - the concept of mobility data
  - what can we learn from mobility data
  - the core applications for mobility data (traffic analysis, LBS/LBSN)
  - the technical challenges from the data management perspective



End of chapter