Tutorial: Urban Trajectory Data Visualization
- Data Model and Management

Ye Zhao
Outline

• Urban Data and Availability
• Urban Trajectory Data Types
• Data Preprocessing and Data Registration
• Urban Trajectory Data and Query Model
• Spatial Database and Indexing Schemes
• Urban Trajectory Data Management with Examples
Urban Data

Urban structures: Defining urban space

• Road Network
  • Roads may be categorized such as Primary, Secondary, Residential, …

• POIs
  POIs may be categorized into hierarchical levels.
Urban Data

Other Related Data

• News, public reports, statistics…

• Social Media
  • Blogs, tweets, …
Urban Mobility Data

Urban trajectories: recording human behavior traces

- Humans, vehicles, fleets, public transits, ...
  - GPS, Wi-Fi, Cellular, RFID, etc.

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>06/17/12 08:19:11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>1</td>
</tr>
<tr>
<td>Speed</td>
<td>50KM/H</td>
</tr>
<tr>
<td>Latitude</td>
<td>22.533</td>
</tr>
<tr>
<td>Longitude</td>
<td>114.044</td>
</tr>
</tbody>
</table>
Trajectory Data Sources

- Personal positioning devices
- Vehicle positioning devices
- Cellular stations
- Wifi access points
- RFID detectors
- Geo-tagged messages
- Others
Availability

• Private: data recorders, administrators, businesses, etc.
  • Privacy protection
  • Human subject sensitivity
  • Business properties and values
• Possibly acquire as collaborative partners
• Public: mostly anonymized data for research use
  • Examples:
    • Dataset of Trajectories of Taxi Cabs in Porto, Portugal.
    • Dataset of Trajectories of Taxi Cabs in Rome, Italy.
    • Dataset of OD of Taxi Cabs in NYC, USA.
Trajectory Data Types

- GPS Point Samples
  - E.g., Origin and destination of taxi trips
Trajectory Data Types

- POI Point Samples
  - E.g., Bus check-in and check-out data

Subway usage in Paris, a data visualization made by Data-Publica
Trajectory Data Types

- Polyline Trajectory Data
- Linked point samples
Trajectory Data Types

- Aggregated Data over Geospatial Spaces
- Group motion behavior among regions, streets, POIs, …
- Less sensitive, relatively easy to acquire and share

Taxi trips Berlin, by TU Berlin
Related Geospatial-Temporal Data

Inherently linked to trajectory data

- Demographics
- News, public reports, business data…
- Social Media
  - Blogs, tweets ..

Tutorial: Urban Trajectory Visualization
Related Geospatial-Temporal Data

- Law enforcement reports
- Street view pictures and videos

![Mapping Decline](http://blogoscoped.com/files/google-street-view-api-large.jpg)
Trajectory Data Issues

- Inaccuracy and error
  - Sampling errors
  - Transformation/transfer errors
  - Missing data
- Cleaning
  - Removing
  - Correcting and Interpolation
# Data Cleaning

## Removing Errors

<table>
<thead>
<tr>
<th>Trajectory_ID</th>
<th>latitude</th>
<th>longitude</th>
<th>date/time</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>41.1411</td>
<td>-8.61861</td>
<td>7/1/2013 00:00</td>
<td>2.813618</td>
</tr>
<tr>
<td>1</td>
<td>41.1414</td>
<td>-8.6135</td>
<td>7/1/2013 00:01</td>
<td>47.05799</td>
</tr>
<tr>
<td>1</td>
<td>41.1425</td>
<td>-8.62033</td>
<td>7/1/2013 00:01</td>
<td>20.41256</td>
</tr>
<tr>
<td>1</td>
<td>41.1438</td>
<td>-8.62215</td>
<td>7/1/2013 00:01</td>
<td>49.55936</td>
</tr>
<tr>
<td>1</td>
<td>41.1444</td>
<td>-8.62979</td>
<td>7/1/2013 00:01</td>
<td>14.12102</td>
</tr>
<tr>
<td>1</td>
<td>41.1448</td>
<td>-8.62968</td>
<td>7/1/2013 00:01</td>
<td>14.12102</td>
</tr>
<tr>
<td>1</td>
<td>41.1447</td>
<td>-8.62737</td>
<td>7/1/2013 00:02</td>
<td>59.06425</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trajectory_ID</th>
<th>latitude</th>
<th>longitude</th>
<th>date/time</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>41.1398</td>
<td>-8.64035</td>
<td>7/1/2013 00:08</td>
<td>37.55105</td>
</tr>
<tr>
<td>1</td>
<td>41.1391</td>
<td>-8.64472</td>
<td>7/1/2013 00:08</td>
<td>46.44904</td>
</tr>
<tr>
<td>1</td>
<td>41.1395</td>
<td>-8.64145</td>
<td>7/1/2013 00:08</td>
<td>20.76127</td>
</tr>
<tr>
<td>1</td>
<td>41.1408</td>
<td>-8.64592</td>
<td>7/1/2013 00:08</td>
<td>63.51959</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trajectory_ID</th>
<th>latitude</th>
<th>longitude</th>
<th>date/time</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>41.1362</td>
<td>-8.65</td>
<td>7/1/2013 00:08</td>
<td>55.02051</td>
</tr>
<tr>
<td>1</td>
<td>41.1362</td>
<td>-8.65317</td>
<td>7/1/2013 00:08</td>
<td>57.42575</td>
</tr>
<tr>
<td>1</td>
<td>41.1367</td>
<td>-8.65841</td>
<td>7/1/2013 00:08</td>
<td>50.46672</td>
</tr>
<tr>
<td>1</td>
<td>41.1362</td>
<td>-8.65571</td>
<td>7/1/2013 00:08</td>
<td>77.02357</td>
</tr>
<tr>
<td>1</td>
<td>41.1362</td>
<td>-8.65199</td>
<td>7/1/2013 00:08</td>
<td>84.41072</td>
</tr>
<tr>
<td>1</td>
<td>41.1362</td>
<td>-8.64734</td>
<td>7/1/2013 00:08</td>
<td>84.41072</td>
</tr>
<tr>
<td>1</td>
<td>41.1362</td>
<td>-8.64274</td>
<td>7/1/2013 00:08</td>
<td>84.41072</td>
</tr>
<tr>
<td>1</td>
<td>41.1362</td>
<td>-8.63907</td>
<td>7/1/2013 00:08</td>
<td>84.41072</td>
</tr>
<tr>
<td>1</td>
<td>41.1362</td>
<td>-8.63543</td>
<td>7/1/2013 00:08</td>
<td>84.41072</td>
</tr>
</tbody>
</table>

## Tutorial: Urban Trajectory Visualization
Trajectory Map Matching

• Map GPS sampling points to roads
• A research topic in GIS with many algorithms
Trajectory Data Registration

- Also mapped to POIs, geo-regions and other geospatial elements
- A critical process in most visualization tasks
- Link trajectory data to urban structures/objects and then urban attributes
Urban Trajectory Data Query Model

• Accessing trajectory data involves five components
  • Trajectory data
  • Geographic structures
  • Time constraints
  • Query modes
  • Query results
Trajectory Data Elements

- Linking GPS points to form a trajectory
- May only know Start/End points
- May not have accurate Latitude and Longitude
  - Approximate region (cellular tower region)
  - Street address
  - POI
Geographic Structures

- Provide meaningful conditions for various data access
- Highly related to GIS information of
  - Regions (zipcodes, demographics, …)
  - Street networks
- POI
- Geometric information
- Semantic information

![Geographic Structure G diagram]

- Region Name, Bounding polygon
- Street Name, Polyline
- POI Name, Radius

Tutorial: Urban Trajectory Visualization
Time Constraints

- Time Period
- In-week distribution
  - Monday, Tuesday, … Sunday
- In-Day distribution
  - Morning, Noon, Afternoon, Night
- Hourly
Query Modes

• A query retrieves trajectory data items $\Phi_i$ that
  • pass
  • start from
  • end at
  • are contained inside
given geographic structure
• Joint query
  • Start from A end at B
  • … …
Query Results

• A series of trajectory elements including points, traces and their geographical structures (e.g., streets)
• To be visualized
  • Visualize trajectories directly
  • Visualize specific points
  • Project and aggregate over street/region/POI
Supporting Data Retrieval

• Spatial database specifically designed for trajectory data
  • Data indexing structure and algorithm
  • Data aggregation
• Many challenges
  • Performance
  • Web transfer and communication
  • Big data issues
Spatial Database

- Database that is optimized for storing and querying a geometric spatial data.
- It allows the representation of Spatial Data Types SDT such as points, lines, polygons.
Spatial Database

• Supports SDT with:
  • Spatial indexing.
  • Efficient query language.
  • Relationships among geographic structures
  • Efficient algorithms for spatial joins.
Spatial Database Systems

- PostGIS extension with Postgresql database.
- MonetDB/GIS extension for MongoDB.
- Spatial extension with MySQL.
- Oracle Spatial.
- Geocouch extension with CouchDB.
- Microsoft SQL Server.
- Others.
Spatial Indexing

• **Spatial index** is a type of extended index that allows you to index a spatial data type to optimize spatial queries.
  • Speed up spatial queries
  • Efficient for modification
  • Redundancy but not too much space

• **Algorithms:**
  • Grid (Spatial Index)
  • R-tree/R+ tree/R* tree
  • Quadtree
  • kd-tree
  • …
Grid (Spatial Index)

- A **grid** is a tessellation of 2-D surface.
- It divides the surface into a series of contiguous cells.
- Cells is assigned unique identifiers.

Quadtrees

- A quadtree is a tree data structure in which each node has zero or four children.
- It is recursively dividing a flat 2-D space into four quadrants.
**R-tree**

- The **R-tree** indexing method organizes data in a tree-shaped structure.
- The index uses a bounding box.
- Bounding box is a rectilinear shape that completely contains the data objects or other bounding boxes.

GIST Indexing

- **GIST** stands for Generalized Search Tree.
- A data structure and API used in PostGIS.
- For spatial indexing, R-Tree is integrated into Gist
Spatial Queries

- A **spatial query** is a special type of database **query** supported by spatial databases and indexing, such as
  - Finding all geographic structures within a given region boundary
  - Finding all points inside a given geometric region
  - Finding all trajectories intersected by a query linestring
Urban Trajectory Data Management

- Urban trajectory data tables
- Spatial queries over trajectory data
- Examples with PostGIS and PostGreSQL database
Postgres Example

- Input **GPS** points of trips data set.
  ✓ Create table.

    ```sql
    CREATE TABLE Porto_GPS_Points
    (
        tripid integer,
        latitude double precision,
        longitude double precision,
        pdateDateTime timestamp without time zone,
        speed double precision
    );
    ```

  ✓ Import from csv file.

    ```sql
    COPY Porto_GPS_Points FROM 'File_Path\File_Name.csv'
    DELIMITERS ',' CSV HEADER;
    ```
Raw Trajectory Dataset

- GPS points
- Each has Trip ID indicating the trajectory it belongs to
- Trajectory (i.e., Trip) attributes vary by different applications
  - Time
  - Speed
  - Others
- A csv file from Porto datasets:
  - Taxi trajectories by all the 442 taxis running in the city of Porto, Portugal

Overview of Data Table Scheme

- Create tables from raw data
  - Table of Point
  - Table of Trips

![Diagram showing data table scheme](image-url)
GPS_Points Table

- Each record is one sampling point from a trajectory
- A GIST indexing is created together

```
CREATE TABLE GPS_Points1 AS
(
    SELECT tripid,
    ST_MakePoint(longitude, latitude) AS point,
    pdateTime timestamp,
    Speed double,
    FROM Porto_GPS_Points //csv file
);
CREATE INDEX gps_points1_Index ON GPS Points
USING GIST (point);
```
Trajectory Tables

- Each record is a complete trajectory as a connected line of its sampling points.
- Use LineString fields for trajectory geometry, start/end point
  - In PostGIS, a Linestring accommodates point, multipoint, or line geometries

```
CREATE TABLE Trips AS
    SELECT *
             , ST_StartPoint(trip) AS startpoint
             , ST_EndPoint(trip) AS endpoint,
    FROM
    (SELECT tripid
         , ST_MakeLine(point order by pdatetime) AS trip,
         min(pdatetime) AS starttime
         , max(pdatetime) AS endtime,
         array_agg(speed order by pdatetime) AS speeds,
         array_agg(pdatetime order by pdatetime) AS pointstime
    FROM GPS_Points1
    GROUP BY tripid
    )
```
Indexing Trajectory Tables

- Create spatial index Gist over them

  CREATE INDEX Trips_index1 ON Trips USING GIST (trip);
  CREATE INDEX Trips_index2 ON Trips USING GIST (startpoint);
  CREATE INDEX Trips_index3 ON Trips USING GIST (endPoint);
Queries over Data Tables

1. **Pass** Query

   **Input parameters:**
   1. Query Region.
   2. \([\text{Time1}, \text{Time2}]\).

```
SELECT Trajectory ID, Trajectory FROM Trajectories
WHERE Start Time BETWEEN (\text{Time1} AND \text{Time2})
AND (Query Region INTERSECT Trajectory).
```
Pass Query Example

- SELECT tripid, ST_AsText(trip) AS trajectory FROM Trips WHERE starttime BETWEEN '2013-07-05 00:00:00' AND '2013-07-01 09:00:00' AND ST_Intersects(ST_GeomFromText('POLYGON((-8.624954223632814 41.1437074383407, -8.624954223632814 41.1597351586533, -8.586158752441408 41.1597351586533, -8.586158752441408 41.1437074383407, -8.624954223632814 41.1437074383407))', 4326), trip);
2. **Start From** Query

   Input parameters:
   1. Query Region.
   2. [Time1,Time2].

   ```
   SELECT Trajectory ID, Trajectory FROM Trajectories
   WHERE Start Time BETWEEN (Time1 AND Time2)
   AND (Query Region CONTAINS Start Point).
   ```
Start From Query Example

- `SELECT tripid, ST_AsText(trip) AS trajectory FROM Trips WHERE starttime BETWEEN '2013-07-01 05:00:00' AND '2013-07-01 09:00:00' AND ST_Contains(ST_GeomFromText('POLYGON((-8.624954223632814 41.1437074383407,-8.624954223632814 41.1597351586533, -8.586158752441408 41.1597351586533, -8.586158752441408 41.1437074383407,-8.624954223632814 41.1437074383407))', 4326), startpoint);`
Queries over Data Tables

3. End At Query

Input parameters:
1. Query Region.
2. [Time1, Time2].

```
SELECT Trajectory ID, Trajectory FROM Trajectories
WHERE End Time BETWEEN (Time1 AND Time2)
AND (Query Region CONTAINS End Point).
```
End At Query Example

- `SELECT tripid, ST_AsText(trip) AS trajectory FROM Trips WHERE endtime BETWEEN '2013-07-01 05:00:00' AND '2013-07-01 09:00:00' AND ST_Contains(ST_GeomFromText('POLYGON((-8.624954223632814 41.1437074383407, -8.624954223632814 41.1597351586533, -8.586158752441408 41.1597351586533, -8.586158752441408 41.1437074383407, -8.624954223632814 41.1437074383407))', 4326), endpoint);`
Queries over Data Tables

4. Contain Query

Input parameters:
1. Query Region.
2. \([\text{Time1}, \text{Time2}]\).

```
SELECT Trajectory ID, Trajectory FROM Trajectories
WHERE Start Time BETWEEN (Time1 AND Time2)
AND (Query Region CONTAINS Trajectory).
```
Contain Query Example

- `SELECT tripid, ST_AsText(trip) AS trajectory FROM Trips WHERE starttime BETWEEN '2013-07-01 05:00:00' AND '2013-07-01 09:00:00' AND ST_Contains(ST_GeomFromText('POLYGON((-8.624954223632814 41.1437074383407, -8.624954223632814 41.1597351586533, -8.586158752441408 41.1597351586533, -8.586158752441408 41.1437074383407, -8.624954223632814 41.1437074383407))', 4326), trip);`
Queries over Data Tables

5. Joint Query
Combine query conditions
For example: Trajectories start at Region1 and end at Region2

Input parameters:
1. Query Region1 & Region2
2. [Time1, Time2].
Aggregation Queries

• **Temporal** aggregation: grouping trajectory data on
  • Week days.
  • Day hours.
• **Spatial** aggregation: grouping trajectory data on
  • Roads.
  • POIs.
  • Regions.
Temporal Aggregation Queries

- Week days aggregation with Pass query.
- Retrieve the number of trips in each week day.

```sql
SELECT startday, COUNT(*) FROM

(  
    SELECT tripid, EXTRACT(DOW FROM starttime) AS startday FROM Trips WHERE starttime BETWEEN '2013-06-16 00:00:00' AND '2013-06-18 23:59:59' AND
    ST_Intersects(ST_GeomFromText('POLYGON((-8.624954223632814 41.1437074383407,-8.6249542223632814 41.1597351586533,-8.586158752441408 41.1597351586533,-8.586158752441408 41.1437074383407,-8.624954223632814 41.1437074383407))',4326)),trip)

) GROUP BY startday ORDER BY startday";```
Temporal Aggregation Queries

- Day hours aggregation with Pass query.
- Retrieve the number of trips in each hour in a day.

```sql
SELECT starthour, COUNT(*) FROM
(
    SELECT tripid, EXTRACT(Hour FROM starttime) AS starthour FROM Trips WHERE starttime BETWEEN '2013-06-16 00:00:00' AND '2013-06-18 23:59:59' AND ST_Intersects(ST_GeomFromText('POLYGON((-8.624954223632814 41.1437074383407, -8.624954223632814 41.1597351586533, -8.586158752441408 41.1597351586533, -8.586158752441408 41.1437074383407, -8.624954223632814 41.1437074383407))', 4326), trip)
) GROUP BY starthour ORDER BY starthour ;
```
Temporal Aggregation Result
Create Tables with Road/Region Information

- Create data tables based on map-matching results over roads (and regions)
Retrieve aggregated attributes on roads

- Counts of trajectories
- Average speeds (max, min, etc.)

```sql
SELECT road_id, COUNT(*) AS count, AVG(speed) AS average_speed FROM
(
    SELECT unnest(road_ids_array) AS road_id, unnest(speeds_array) AS speed
    FROM Trips
    WHERE starttime BETWEEN '2013-07-01 05:00:00' AND '2013-07-01 09:00:00'
    AND ST_Contains(ST_GeomFromText('POLYGON((-8.624954223632814 41.1437074383407,-8.6249542223632814 41.1597351586533, -8.586158752441408 41.1437074383407,-8.624954223632814 41.1437074383407)),4326), trip);
) GROUP BY road_id
```

Spatial Aggregation Queries

Tutorial: Urban Trajectory Visualization
Traffic Speed on Roads
Counts of Taxi Trips on Roads
Data Performance and Issues

- Data aggregation on the fly
  - Easy implementation
  - Slow when the amount of trajectories is large
- Potential solution
  - Precomputing aggregations
  - Create caching structures (e.g. data cubes) in database
Thanks!