Data Warehousing & Mining Techniques for Moving Object Databases

Gerasimos Marketos

Information Systems Lab (InfoLab)
University of Piraeus (UniPi), Greece

http://infolab.cs.unipi.gr

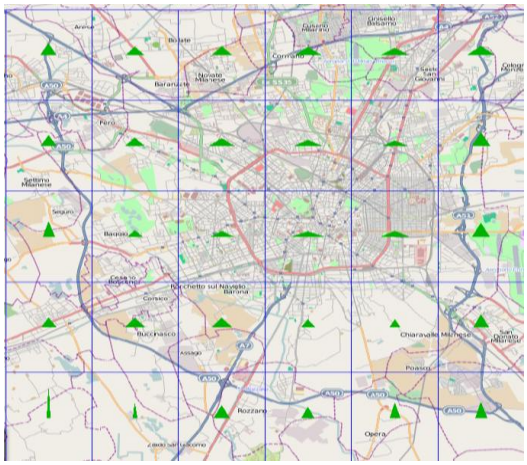
Outline

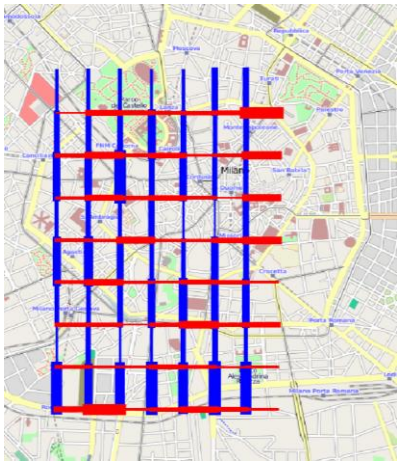
- Preliminaries
- Efficient Trajectory Data Warehousing
  - A framework for Trajectory Data Warehousing
  - Ad-hoc OLAP on trajectory data for supporting multiple trajectory definitions
- Trajectory-inspired Data Mining
  - Mining Interaction Patterns for spatiotemporal representation, synthesis and classification
  - Mining Traffic Patterns for monitoring the traffic flow
- Building real-world Trajectory Data Warehouses
  - Visual OLAP analysis
- Summary & open issues

Analyzing Mobility Data

- The usage of location aware devices is widely spread nowadays, allowing access to large **spatiotemporal** datasets
- The traditional database technology has been extended into **Moving Object Databases** (MODs) that handle modeling, indexing and query processing issues for trajectories
- We are interested in **analyzing mobility data** in order to discover behavioral patterns. For this purpose, we use:
  - **Trajectory Data Warehousing** (TDW) to analyze various measures such as the variable number of moving objects in different urban areas and the average speed of vehicles
  - **Trajectory-inspired Mining** algorithms that can help us to gain insights on the traffic problem and on interactions among objects

Analyzing Mobility Data

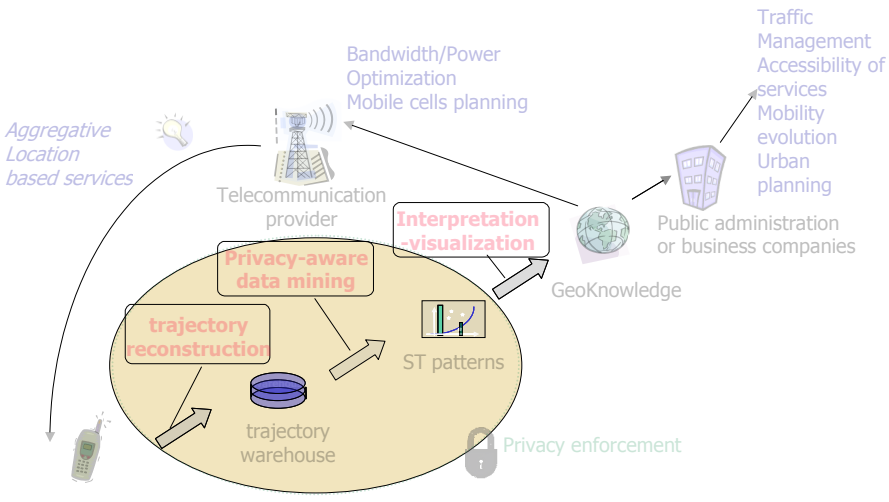




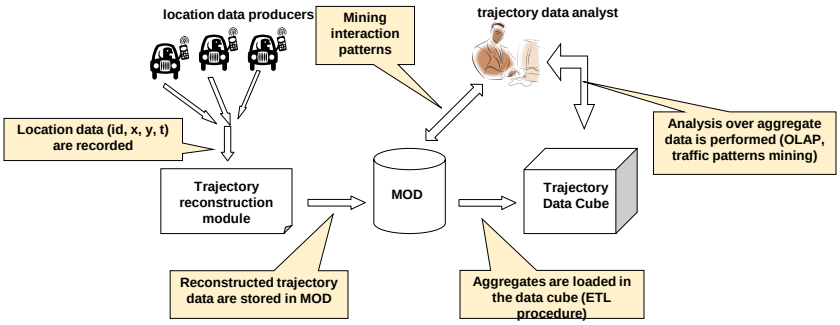
Motivations

- The research efforts that are presented in this thesis are motivated by two core scenarios from the **human movement activity** and the **transportation management**
  - e.g. an advertising company is interested in analyzing mobility data in different areas of a city so as to decide upon road advertisements (placed on panels on the roads)
  - e.g. city authorities need tools so as to study the traffic flow in order to be able to improve traffic conditions, react effectively in case of some traffic problems etc
- The above motivation scenarios can be realized using **novel analytical techniques** that will be able to exploit the available rich mobility semantics

The big picture (the GeoPKDD project)



Our contribution



Our contribution

- We provide a technique for transforming raw timestamped location points to trajectories
- We propose two frameworks for Trajectory Data Warehousing:
  - considering a static definition of the notion of trajectory
  - supporting multiple trajectory definitions
- We propose three techniques for extracting mobility patterns:
  - Extracting interaction patterns
  - Clustering the traffic edges of the road network
  - Discovering relationships between the edges of a road network
- We develop T-WAREHOUSE, a prototype for Visual TDW

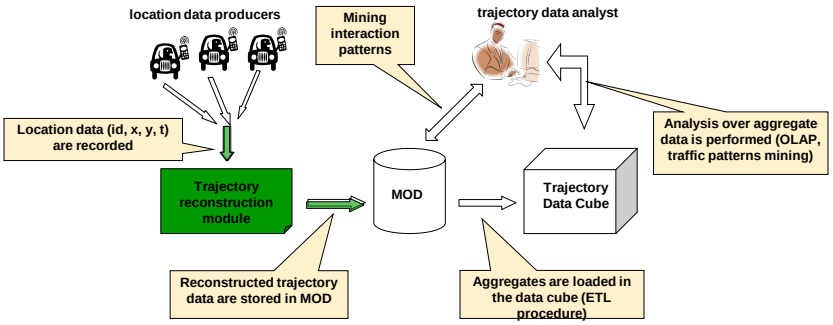
Outline

- Preliminaries
- Efficient Trajectory Data Warehousing
  - A framework for Trajectory Data Warehousing
  - Ad-hoc OLAP on trajectory data for supporting multiple trajectory definitions
- Trajectory-inspired Data Mining
  - Mining Interaction Patterns for spatiotemporal representation, synthesis and classification
  - Mining Traffic Patterns for monitoring the traffic flow
- Building real-world Trajectory Data Warehouses
  - Visual OLAP analysis

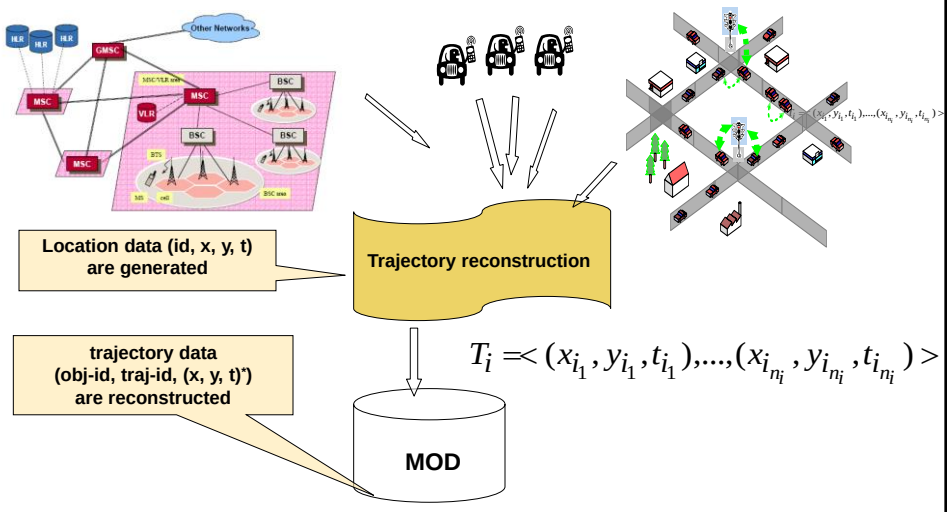
Related work

- Some research work has been done in the area of Spatial Data Warehousing (Han et al., 1998)
  - (Shekhar et al., 2001) extend the idea of cube dimensions so as to include spatial and non-spatial ones, and of cube measures so as to represent space regions and/or calculate numerical data
- One step beyond Spatial DW is modeling a TDW (Pelekis et al., 2007)
- A simple TDW model is presented in (Orlando et al., 2007)
- (Leonardi et. al, 2009) discusses storing and aggregation issues for frequent spatiotemporal patterns mined from trajectories

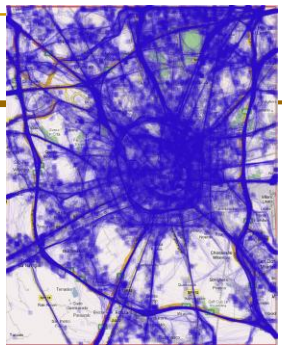
Our contribution



Location data producers: GSM, GPS, WiFi

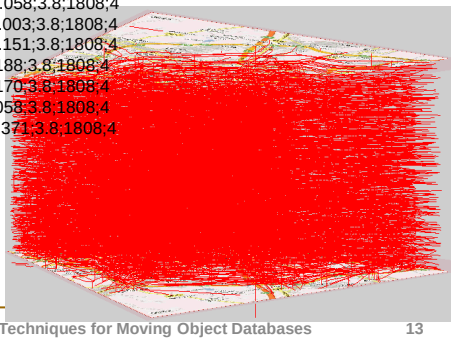


Mobility Data



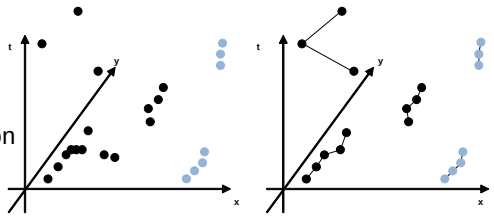
- Typical structure and size:

N;Time;Lat;Lon;Height;Course;Speed;PDOP;State;NSat
…
8;22/03/07 08:51:52;50.777132;7.205580; 67.6;345.4;21.817;3.8;1808;4
9;22/03/07 08:51:56;50.777352;7.205435; 68.4;35.6;14.223;3.8;1808;4
10;22/03/07 08:51:59;50.777415;7.205543; 68.3;112.7;25.298;3.8;1808;4
11;22/03/07 08:52:03;50.777317;7.205877; 68.8;119.8;32.447;3.8;1808;4
12;22/03/07 08:52:06;50.777185;7.206202; 68.1;124.1;30.058;3.8;1808;4
13;22/03/07 08:52:09;50.777057;7.206522; 67.9;117.7;34.003;3.8;1808;4
14;22/03/07 08:52:12;50.776925;7.206858; 66.9;117.5;37.151;3.8;1808;4
15;22/03/07 08:52:15;50.776813;7.207263; 67.0;99.2;39.188;3.8;1808;4
16;22/03/07 08:52:18;50.776780;7.207745; 68.8;90.6;41.170;3.8;1808;4
17;22/03/07 08:52:21;50.776803;7.208262; 71.1;82.0;35.058;3.8;1808;4
18;22/03/07 08:52:24;50.776832;7.208682; 68.6;117.1;11.371;3.8;1808;4
…



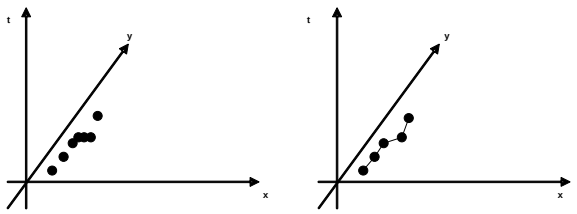
Reconstructing trajectories

- Collected raw data represent time-stamped geographical locations
- Apart from storing raw data in the MOD, we are also interested in reconstructing trajectories:
  - Raw points arrive in bulk sets
  - We need a filter that decides if the new series of data is to be appended to an existing trajectory or not:
    - Tolerance distance
    - Temporal gap
    - Spatial gap
    - Maximum speed
    - Maximum noise duration



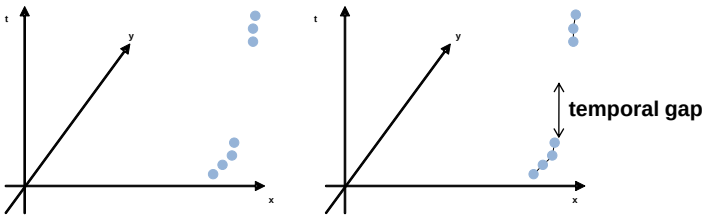
Reconstructing trajectories: parameters

- Tolerance distance
  - The tolerance of the transmitted time-stamped positions. In other words, it is the **maximum distance between two consecutive time-stamped positions** of the same object in order for the object to be **considered** as **stationary**



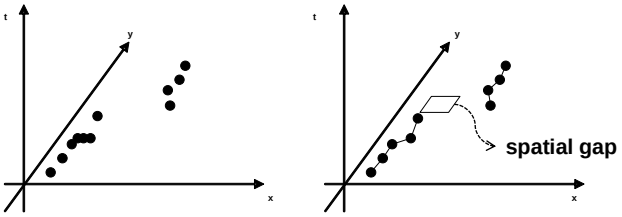
Reconstructing trajectories: parameters

- Tolerance distance
- Temporal gap between trajectories
  - The **maximum allowed time interval** between two consecutive time-stamped positions of the same trajectory for a single moving object



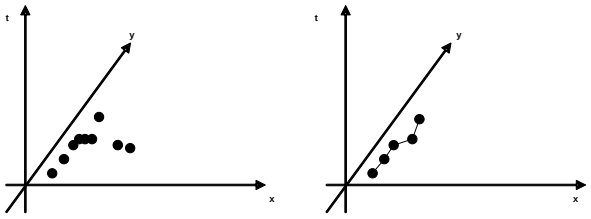
Reconstructing trajectories: parameters

- Tolerance distance
- Temporal gap between trajectories
- Spatial gap between trajectories
  - The **maximum allowed distance** in 2D plane between two consecutive time-stamped positions of the same trajectory



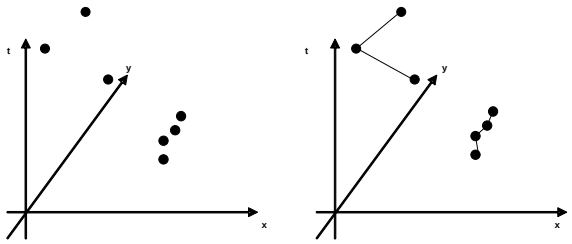
Reconstructing trajectories: parameters

- Tolerance distance
- Temporal gap between trajectories
- Spatial gap between trajectories
- Maximum speed
  - It is used in order to determine whether a reported time-stamped position must be considered as **noise** and consequently discarded from the output trajectory



Reconstructing trajectories: parameters

- Tolerance distance
- Temporal gap between trajectories
- Spatial gap between trajectories
- Maximum speed
- Maximum noise duration
  - The **maximum duration of a noisy part** of a trajectory. Any sequence of noisy time-stamped positions of the same object will result in a new trajectory given that its duration exceeds $noise_{max}$



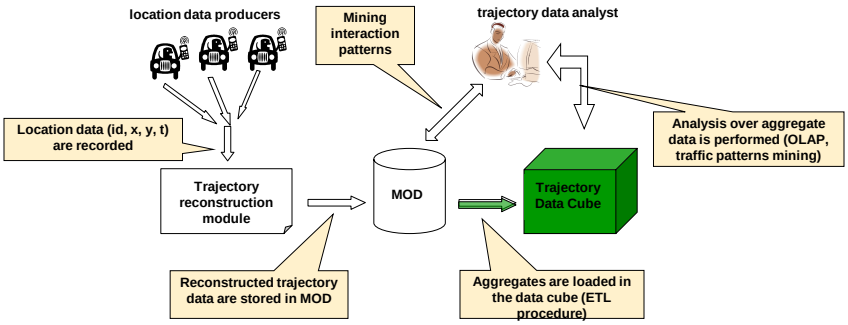
Reconstructing trajectories: algorithm

- As a first step (lines 1-6), the algorithm checks whether the object has been processed
  - if so, it retrieves its partial trajectory from the corresponding list
  - otherwise, it creates a new trajectory and adds it to list
- Then (lines 7-31), it compares the incoming point P with the tail of the partial trajectory (LastPoint) by applying the trajectory reconstruction parameters

```
Algorithm Trajectory-Reconstruction
          (PartialTrajectories List, P Point, OId ObjectId)

 1.     IF NOT PartialTrajectories.Contains(OId) THEN
 2.       CTrajectory=New Trajectory;
 3.       CTrajectory.AddPoint(P);
 4.       PartialTrajectories.Add(CTrajectory);
 5.     ELSE
 6.       CTrajectory=PartialTrajectories(OId);
 7.       IF Distance(CTrajectory.LastPoint,P)<= D_TOL THEN
 8.         IF P.T - CTrajectory.LastPoint.T > gap_Time THEN
 9.           Report Ctrajectory.LastPoint;
10.           CTrajectory.Id=CTrajectory.Id+1;
11.           CTrajectory.AddPoint(P);
12.         ENDIF
13.       ELSEIF Speed(CTrajectory.LastPoint,P)> V_max THEN
14.         IF P.T - CTrajectory.LastPoint.T > noise_max THEN
15.           Report CTrajectory.Noise;
16.         ELSE
17.           CTrajectory.AddNoise(P);
18.         ENDIF
19.       ELSEIF Distance(CTrajectory.LastPoint,P)> gap_space THEN
20.         Report Ctrajectory.LastPoint;
21.         CTrajectory.Id=CTrajectory.Id+1;
22.         CTrajectory.AddPoint(P);
23.       ELSE
24.         IF P.T - CTrajectory.LastPoint.T > gap_Time THEN
25.           Report Ctrajectory.LastPoint;
26.           CTrajectory.Id=CTrajectory.Id+1;
27.           CTrajectory.AddPoint(P);
28.         ELSE
29.           CTrajectory.AddPoint(P);
30.         ENDIF
31.       ENDIF
32.     ENDIF
```

Our contribution



Trajectory data warehousing

- The **analysis** of **trajectory data** raises opportunities for discovering behavioral patterns that can be exploited in various applications
- TDW should
  - extract aggregate information from MOD
  - support a variety of dimensions (temporal, spatial, thematic, …) and measures (about space, time and their derivatives)
  - Storing **measures** associated with facts, concerning the **set of trajectories** crossing the cell
    ⇒ **aggregate** information in base cells
- Challenges
  - design of a trajectory-oriented data cube
  - high volume and complex nature of data; special **ETL processing** requirements
  - extensions of traditional aggregation techniques to produce summary information for **OLAP analysis**

Sample schemas

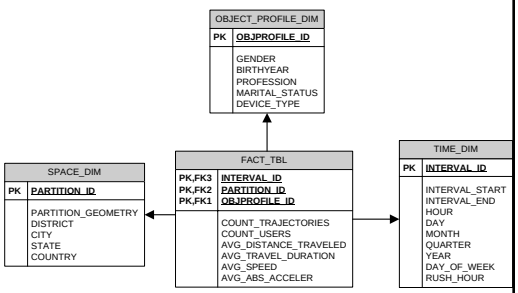
- Moving Object Database

OBJECTS (object-id: *identifier*, description: *text*, gender: {M | F}, birth-date: *date*, profession: *text*, device-type: *text*)

RAW_LOCATIONS (object-id: *identifier*, timestamp: *datetime*, eastings-x: *numeric*, northings-y: *numeric*, altitude-z: *numeric*)

MOD_TRAJECTORIES (trajectory-id: *identifier*, object-id: *identifier*, trajectory: *3D geometry*)

- Trajectory Data Warehouse
  - *Dimensions:* Spatial, Temporal, Object Profile
  - *Measures:* count (trajectories), count (users), avg (distance traveled), avg (travel duration), avg (speed), avg (abs (acceler))

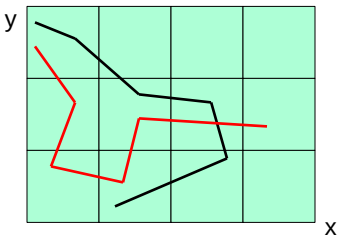


ETL processing: measures

Measure	Formula
COUNT_ TRAJECTORIES	count all distinct trajectory ids that pass through *base cell (bc)*
COUNT_USERS	count all the distinct object ids that pass through *bc*
AVG_DISTANCE_ TRAVELED	$AVG\_DISTANCE\_TRAVELED(bc) = \frac{SUM\_DISTANCE(bc)}{COUNT\_TRAJECTORIES(bc)}$ $SUM\_DISTANCE(bc) = \sum_{TP_i \in bc} len(TP_i)$
AVG_TRAVEL_ DURATION	$AVG\_TRAVEL\_DURATION(bc) = \frac{SUM\_DURATION(bc)}{COUNT\_TRAJECTORIES(bc)}$ $SUM\_DURATION(bc) = \sum_{TP_i \in bc} lifespan(TP_i)$
AVG_SPEED	$AVG\_SPEED(bc) = \frac{SUM\_SPEED(bc)}{COUNT\_TRAJECTORIES(bc)}$ $SUM\_SPEED(bc) = \sum_{TP_i \in bc} \frac{len(TP_i)}{lifespan(TP_i)}$
AVG_ABS_ ACCELER	$AVG\_ABS\_ACCELER(bc) = \frac{SUM\_ABS\_ACCELER(bc)}{COUNT\_TRAJECTORIES(bc)}$ $SUM\_ABS\_ACCELER(bc) = \sum_{TP_i \in bc} \frac{\lvert speed_{fin}(TP_i) - speed_{init}(TP_i) \rvert}{lifespan(TP_i)}$

ETL processing: loading

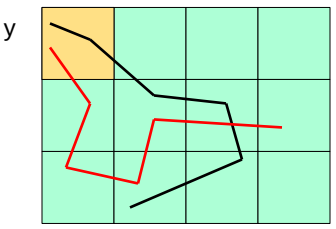
- Loading data into the dimension tables ➔ straightforward
- Loading data into the fact table ➔ complex
  - Fill in the measures with the appropriate numeric values
  - In order to calculate the measures, we have to extract the portions of the trajectories that fit into the base cells of the cube
    - We propose two alternative solutions to this problem:
      - cell-oriented
      - trajectory-oriented



ETL processing: algorithms

***Cell-oriented approach* (COA)**

- Search for the portions of trajectories that they reside inside a spatiotemporal cell
  - Perform a **spatiotemporal range query** that returns the portions of trajectories that satisfy the range constraints
  - This is efficiently supported by the **TB-tree**
- Decompose the trajectory portions with respect to the user profiles they belong to
- Compute measures for this cell
- Repeat for the next cells



COUNT_TRAJECTORIES = 2
COUNT_USERS = 2
…

ETL processing: algorithms

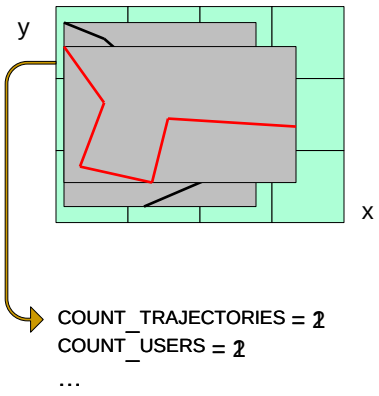
***Trajectory-oriented approach* (TOA)**

- Discover the spatiotemporal cells where each trajectory resides in
  - In order to avoid checking all cells, use the trajectory Minimum Bounding Rectangle (MBR)
- Identify the cells that overlap with the MBR and contain portions of the trajectory
- Compute measures for each cell

…

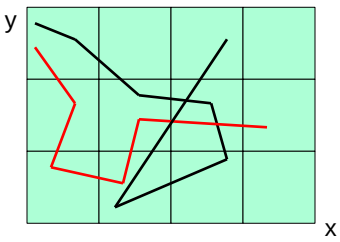
- Repeat for the next trajectories

…



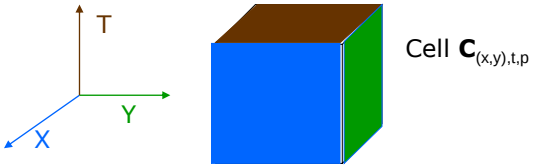
The distinct count problem: definition

- Once the fact table has been fed, aggregate-only information is stored inside the TDW (no trajectory / user ids)
- When rolling up, COUNT_USERS, COUNT_TRAJECTORIES and, hence, all other measures defined over COUNT_TRAJECTORIES are subject to the distinct count problem (Tao et al., 2004):
  - if an object remains in the query region for several timestamps during the query interval, instead of counting this object once, it is counted multiple times in the result
  - We can define:
    - an distributive aggregate function or
    - an algebraic aggregate function or
    - a holistic aggregate function



The distinct count problem: solution (1/2)

- We store in the base cells ($C_{(x,y),t,p}$) a tuple of auxiliary measures that help us correct the errors due to the duplicates when rolling-up:
  - $C_{(x,y),t,p}.Traj$: number of distinct trajectories of profile *p* intersecting the cell
  - $C_{(x,y),t,p}.cross\text{-}x$: number of distinct trajectories of profile *p* crossing the *spatial* border between $C_{(x-1,y),t,p}$ and $C_{(x,y),t,p}$
  - $C_{(x,y),t,p}.cross\text{-}y$: number of distinct trajectories of profile *p* crossing the *spatial* border between $C_{(x,y-1),t,p}$ and $C_{(x,y),t,p}$
  - $C_{(x,y),t,p}.cross\text{-}t$: number of distinct trajectories of profile p crossing the *temporal* border between $C_{(x,y),t-1,p}$ and $C_{(x,y),t,p}$



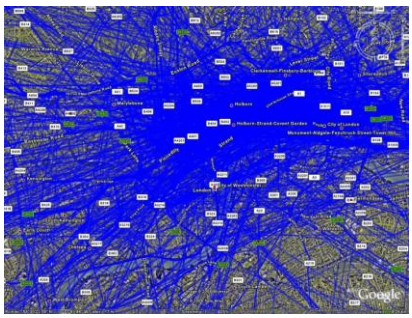
The distinct count problem: solution (2/2)

- Let $C_{(x',y'),t',p'}$ be a cell consisting of the union of two adjacent cells (i.e. $C_{(x,y),t,p} \cup C_{(x+1,y),t,p}$)
- In order to compute the **number of distinct trajectories**:

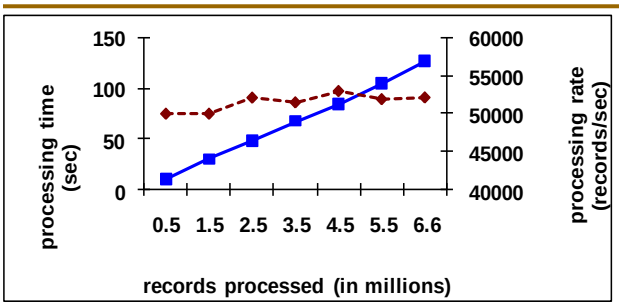
  $C_{(x',y'),t',p'}.Traj = C_{(x,y),t,p}.Traj + C_{(x+1,y),t,p}.Traj - C_{(x+1,y),t,p}.cross\text{-}x$
  - application of the well-known Inclusion/Exclusion principle for sets: $|A \cup B| = |A| + |B| - |A \cap B|$
  - **BUT** in some cases it holds that $C_{(x+1,y),t,p}.cross\text{-}x \neq |A \cap B|$
  - Example: fast and agile trajectories

Experimental Study

- For evaluation, we used **e-Courier** real-world dataset:
  - the movement of 84 couriers moving in greater London (covered area 66,800 km^2) during a one month period (July 2007) with a 10sec sample rate
  - a total of 6.67 millions of raw location records (file size 504 Mb)

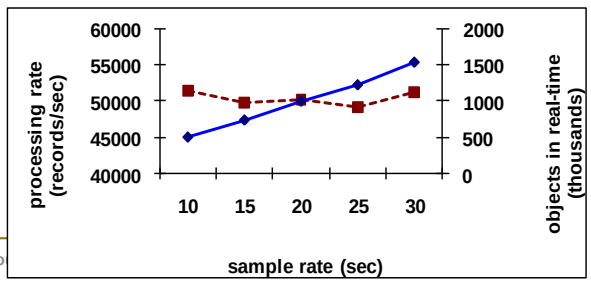


Experimental Study

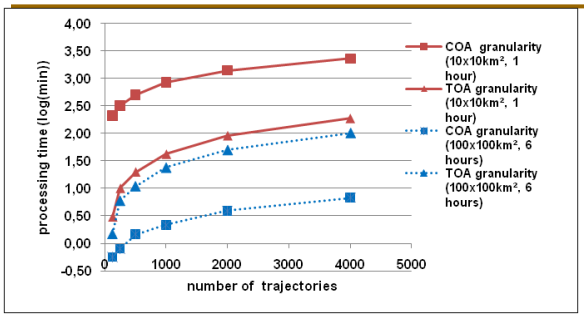


Performance of trajectory reconstruction (solid line: processing time; dotted line: processing rate)

The effect of sample rate in real-time processing (solid line: objects in real-time; dotted line: processing rate)



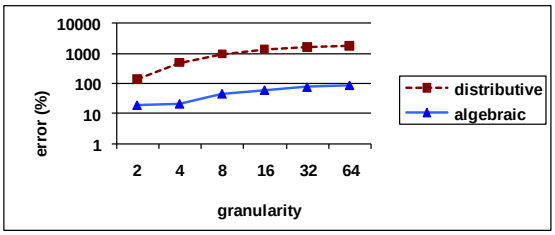
Experimental Study: ETL & distinct count



Comparison of alternative ETL processes

Distributive vs. algebraic aggregate functions

$$Error = \frac{\sum_{q \in Q} |\overline{Mq} - Mq|}{\sum_{q \in Q} Mq}$$



Synopsis and related publications

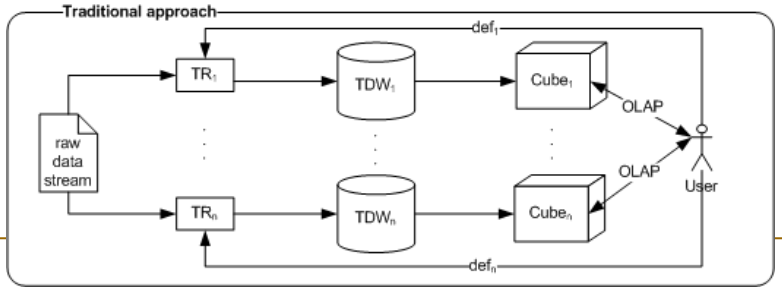
- We proposed an efficient technique for the solution of the **trajectory reconstruction** problem
- We explored solutions for the efficient and effective development of trajectory warehouses. In particular:
  - efficient **ETL support** for trajectory data
  - handling measure aggregation issues, with a special attention to the **distinct count problem**
- Publications
  - Marketos, G., Theodoridis, Y., Mobility Data Warehousing & Mining. *Proceedings of VLDB PhD Workshop*, 2009.
  - Marketos, G., Frentzos, E., Ntoutsi, I., Pelekis, N., Raffaeta, A., and Theodoridis, Y., Building Real-World Trajectory Warehouses. *Proceedings of MobiDE*, 2008.

Outline

- Preliminaries
- Efficient Trajectory Data Warehousing
  - A framework for Trajectory Data Warehousing
  - Ad-hoc OLAP on trajectory data for supporting multiple trajectory definitions
- Trajectory-inspired Data Mining
  - Mining Interaction Patterns for spatiotemporal representation, synthesis and classification
  - Mining Traffic Patterns for monitoring the traffic flow
- Building real-world Trajectory Data Warehouses
  - Visual OLAP analysis

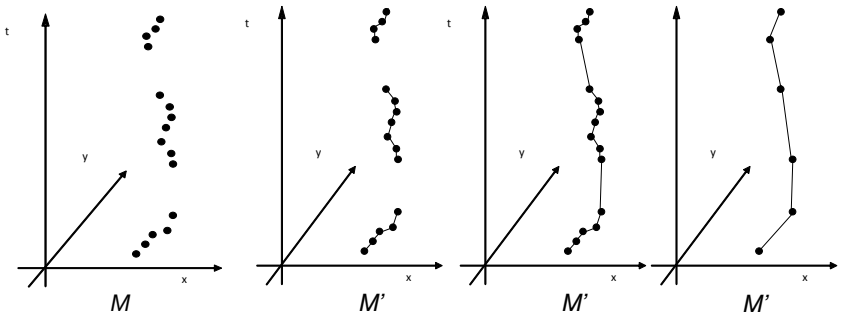
The need for flexibility in TDW

- Trajectory analysis is based on the specific requirements of each application
- There may be a considerable difference on the semantic definitions given to trajectories by different applications
  - E.g., trajectory is different for a logistics manager and a traffic analyst
  - this is unusual in other (conventional or not) data warehousing scenarios



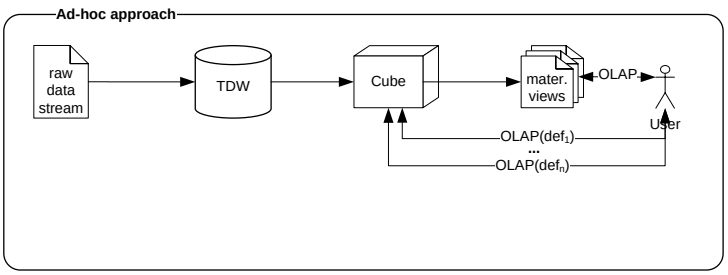
The need for flexibility in TDW

- the movement of an object can be defined as $M = ((x_1, y_1, t_1),...,(x_n, y_n, t_n))$
- after trajectory reconstruction, we get a set of trajectories $M' = (T_1,...,T_j)$ where $1 \le k, m \le n$ and $\bigcup_i T_i \subset M$
- M' satisfies specific requirements



Our proposal for ad-hoc OLAP

- We revisit basic structures (fact table, ETL) of TDW
- We extend traditional OLAP techniques to be ad-hoc in order to handle appropriately the dynamic nature of trajectories
- We discuss cube materialization issues that pre-calculate results for known semantic definitions of trajectories



How to model different trajectory defs?

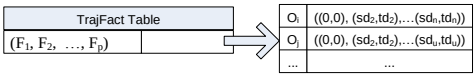
- (Spaccapietra et al., 2008) argue that different time granularities result in different semantic definitions of trajectories
- So, the answer on the question *"How many trajectories exist?"* depends on the time granularity level which we are interested in!
  - We adopt this model and we provide this functionality to the OLAP user!
  - We use sem_{time}: The *maximum allowed time interval* between two consecutive time-stamped positions of the same moving object

Our model

- We propose the transformation of movement as *a sequence of spatiotemporal distances between consecutive points*

$$M_i^{''} =< (0,0),(sd_2,td_2),...,(sd_n,td_n) >$$

  - where (sd_i, td_i) represent the (Euclidean) spatial (*sd*) and temporal distance (*td*) between the i^{th} and the $(i-1)^{th}$ point
  - the first pair is always set to (0, 0) as there is no previous point to compare it with
- We introduce **TrajFact table** as an n-ary relation over $K \times DT$, where:
  i. K is the set of attributes representing the primary key of the fact table formed by $F_1 \times F_2 \times \ldots \times F_p$, where each F_i, $1 \le i \le p$ is a foreign key to the dimension tables;
  ii. DT is a distance table which consists of:
    - Object identification O_j;
    - A sequence of pairs (sd_i, td_i)



  iii. $n = p+1$.

Ad-hoc OLAP

- We need a new OLAP mechanism that utilizes the notion of multiple semantic definitions of trajectories
- We define **Aggregation query** as a set of $\{(bc_1, bc_2, \ldots, bc_i), m, mtd\}$, where:
  - Each bc_j, $1 \le j \le i$ is a base cell that was filtered based on the selected members of dimension (which result in specific rows in the *TrajFact* table)
  - m is the measure
  - mtd is the selected value for the parameter sem_{time}

```
Algorithm General-Aggregation(SetOfCells SOC,
TargetMeasure TM, MaxTemporalDistance MTD)
1.  FOR EACH cell C of SOC DO
2.  //Process the distance table DT of the cell C
3.   FOR EACH object P of C DO
4.    FOR EACH TemporalDistance t of P in DT DO
5.     IF t < MTD THEN
6.      //the same trajectory evolves
7.      UpdateMeasure(TM);
8.     ELSE
9.      //a new trajectory is identified
10.      newTrajectory();
11.    END-IF
12.   END-FOR
13.  END-FOR
14. END-FOR
```


Ad-hoc OLAP: an example

Obj										
	(0,0)	(150,10)	(160,13)	(150,10)	(1,600)	(160,15)	(150,12)	(0,900)	(100,9)	(150,12)

Selecting "300" as maximum temporal distance

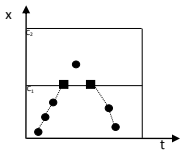
Obj										
	(0,0)	(150,10)	(160,13)	(150,10)	(1,600)	(160,15)	(150,12)	(0,900)	(100,9)	(150,12)

Obj										
	(0,0)	(150,10)	(160,13)	(150,10)	(0,0)	(160,15)	(150,12)	(0,0)	(100,9)	(150,12)
	Len() = 0+150+160+150 = 460				Len() = 0+160+150 = 310			Len() = 0+100+150 = 250		
	Lifespan() = 0+10+13+10 = 33				Lifespan() = 0+15+12 = 27			Lifespan() = 0+9+12 = 21		

Obj	
	COUNT_TRAJECTORIES = 3
	SUM_DISTANCE = 1020
	SUM_DURATION = 81

An appropriate ETL process

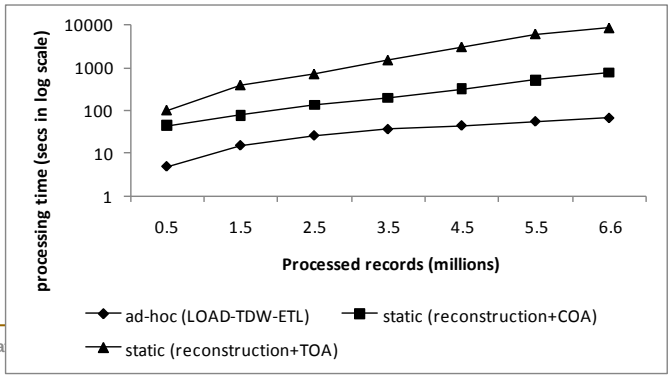
- Using traditional approaches, the computation of almost every measure assumes a specific semantic definition of trajectory
- We propose a more flexible ETL strategy
  - The aim of the ETL procedure is to find which points reside in each cell and compute the distance tables




```
Algorithm Load-TDW-ETL(ListOfPoints LoP)
1. //We assume LoP is sorted by Object-id, timestamp
2. FOR EACH base cell bc_j DO
3. //Find the set of points inside the cell
4.  S = contains(LoP, bc_j);
5.    //Consider a list of points LP for
6.    //each object of S
7.    FOR EACH LP of S DO
8.     Compute_Distances(LP);
9.    END-FOR
10. END-FOR
```

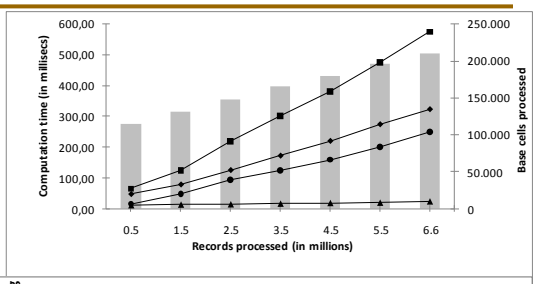

Experimental Study

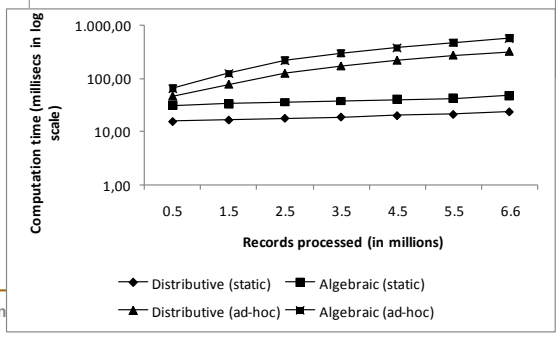
- For evaluation, we used the **e-Courier** real-world dataset
- We provide a comparison between the components of the ad-hoc and the static (traditional) TDW approaches
- Our ETL algorithm performs linear with the size of the input dataset and performs better than the static one



Experimental Study

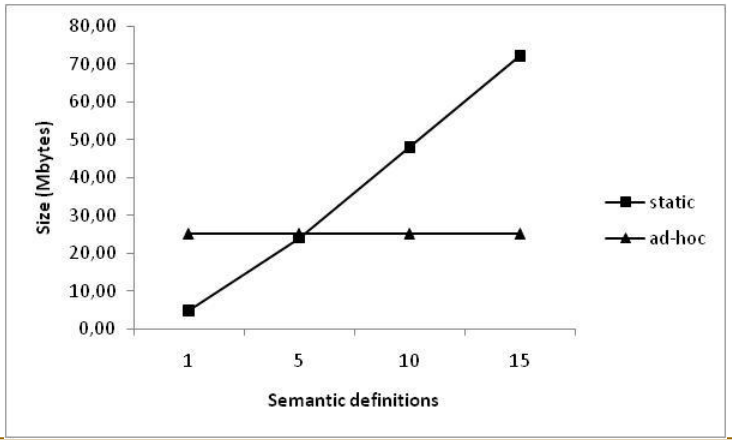
- We evaluated the time required for the computation of measures experimenting with different sizes of the input dataset
- We compared computation times between static and ad-hoc approach





Experimental Study

- We compared the sizes of data cubes built using our ad-hoc and the static approach



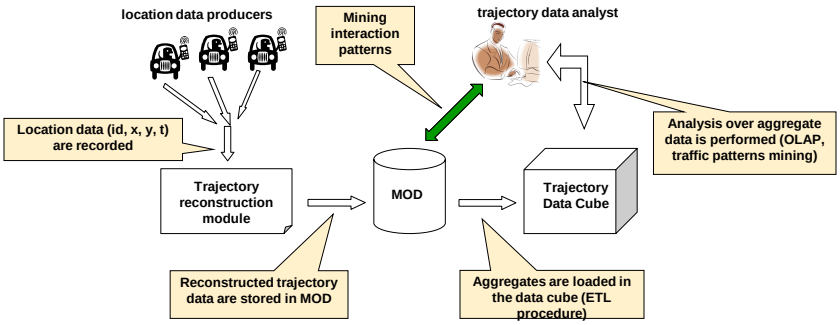
Synopsis and related publications

- We proposed an innovative organization of a trajectory data cube to consider different interpretations of the notion of trajectory
- We did not propose another model for identifying different semantic interpretations of trajectories but we followed an existing model (Spaccapietra et al., 2008)
  - It can be easily extended so as to encapsulate a more complex model: the organization of the ad-hoc TDW remains unchanged and only the definition of aggregation query has to be adapted accordingly
- Publications
  - Marketos, G., Theodoridis, Y., Ad-hoc OLAP on Trajectory Data. *Submitted*.

Outline

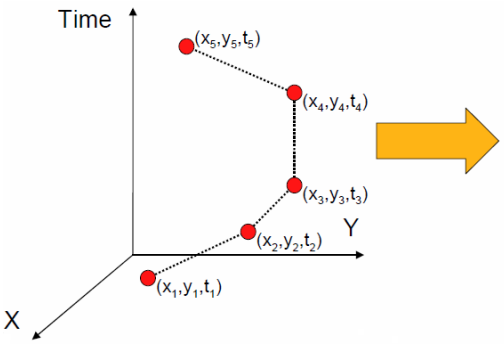
- Preliminaries
- Efficient Trajectory Data Warehousing
  - A framework for Trajectory Data Warehousing
  - Ad-hoc OLAP on trajectory data for supporting multiple trajectory definitions
- Trajectory-inspired Data Mining
  - Mining Interaction Patterns for spatiotemporal representation, synthesis and classification
  - Mining Traffic Patterns for monitoring the traffic flow
- Building real-world Trajectory Data Warehouses
  - Visual OLAP analysis

Our contribution



Problem Definition

- Given a set of labeled moving object trajectories
- Build a classification model *C: Trajectories → Labels*



- { authorized, unauthorized}
- { dangerous, non-dangerous}
- { truck, car, motorbike }
- { tourist, inhabitant }
- …

Related work

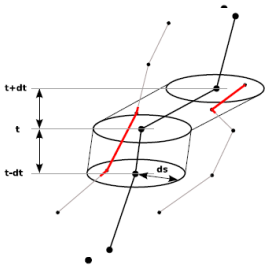
- (Lee et al., 2008) proposed *TraClass* for trajectory classification showing that it is necessary and important to mine interesting knowledge on partial trajectories
- Other approaches (e.g., Dockstader, 2006) apply variants of Markov models to describe the movement of trajectories
  - trajectories to classify are compared against each component of the model to assign it to one of the classes
- In (Geurts, 2001) time series are classified by the application of patterns as test criteria in decision trees
  - Each pattern thereby corresponds to a temporally confined, constant model of the signal which can, for example, represent the velocity of an object

Feature Extraction

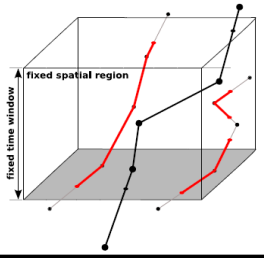
- Moving objects are described only by their trajectories
- Trajectories are not suitable feature vectors
  - Variable length
  - Spatial and time coordinates might be too specific
  - Small time shifts are not taken into consideration
- A set of higher level features need to be computed
  - Descriptors of the movement:
    - Global statistics over each trajectory (simple feature extraction) e.g. trajectory length, {average, max, min} speed, max acceleration, number of lanes changed
    - The behavior of the individual can be better understood if it is compared with its neighbors → **Interaction Descriptors**

Interaction Descriptors

- How to compute them?
- Dynamic neighborhood approach
  - For each moving object at each time period:
    - Find the nearby neighbors (*ds*)
    - Compute Interaction Attributes by comparing the object characteristics with neighbors



- Static neighborhood approach
  - For each moving object at each st-cell:
    - Find all neighbors
    - Compute Interaction Attributes by comparing the object characteristics with neighbors

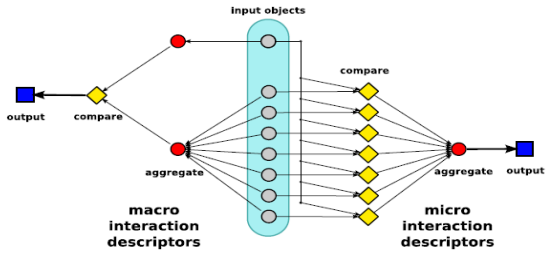


Sample Interaction Attributes

- Let us consider a trajectory *T* and a set of trajectories *ST* that reside in a specific region *R*
- AVG_PERC_DISTANCE: allows us to compare the length of *T* with the average length of *ST*
- AVG_PERC_DURATION: the lifespan of *T* is compared with the average lifespan of *T*
- AVG_PERC_SPEED: divide the average speed of *T* with the average of the speeds of each trajectory *T'* of set *ST*
- AVG_PERC_ABS_ACCELER: divide the average acceleration of *T* with the average of the accelerations of each trajectory *T'* of set *ST*
- VAR_PERC_ACCELER: the proportion of the variance of acceleration of *T* to the variance of acceleration of *ST*

Computation Issues

- The computation of Interaction Attributes can be done in two ways; depending on the computation process they require:
  - *macro*: **all other objects** are aggregated to a single value or complex object, then the reference object under analysis is compared to it
  - *micro*: the reference object is compared **against each** of the neighboring objects, then the set of results obtained are aggregate to a single value



Computation Issues

- COMP-DESCRIPTORS-DYNAMICNEIGHBORHOOD is used to compute descriptors using the dynamic neighborhood approach

```
Algorithm Comp-Descriptors-DynamicNeighborhood (ListOfPoints
lop, ListOfTemporalPeriods lotp, MaxSpatialDistance msd)
 1. FOR EACH period p of lotp DO
 2.  //choose the subset of points that are temporally
 3.  //restricted inside period p
 4.  lop' = GetPointsForPeriod(lop, p);
 5.   FOR EACH object o of lop' DO
 6.    lot = ReconstructTrajectory(o, lop');
 7.     FOR EACH trajectory T in lot DO
 8.      //lont is a list of trajectories that are considered
 9.      //as the neighborhood of T
10.      lont = ComputeNeighborhood(T, msd);
11.      //compare T with lont set of trajectories
12.      ComputeDescriptors(T, lont);
13.     END-FOR
14.   END-FOR
15. END-FOR
```

- COMP-DESCRIPTORS-FIXEDGRID algorithm is used to compute descriptors on a fixed spatiotemporal grid

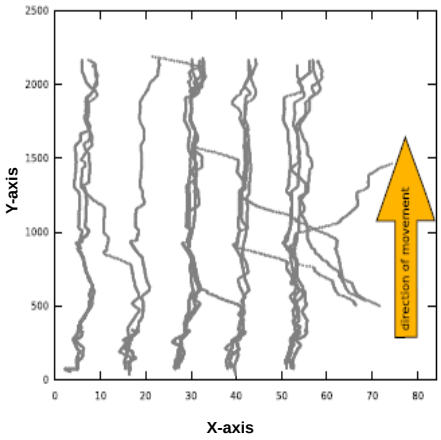
```
Algorithm Comp-Descriptors-FixedGrid
(ListOfPoints lop, ListOfSpatiotemporalCells lostc)
 1. FOR EACH spatiotemporal cell c of lostc DO
 2.  //choose the subset of points that are spatiotemporally
 3.  //restricted inside cell c
 4.  lop' = GetPointsForSTCell(lop, c);
 5.  lot = ReconstructTrajectories(lop');
 6.   FOR EACH trajectory T in lot DO
 7.    lont = lot.Exclude(T);
 8.     //compare T with lont set of trajectories
 9.     ComputeDescriptors(T,lont);
10.   END-FOR
11. END-FOR
```

Extracting Interaction Patterns

- **Step 1: Region extraction**
  - select some regions of space and time to be used later as reference locations (either the *dynamic neighborhood* and the *fixed grid* approach is chosen)
- **Step 2: Computing Interaction descriptors**
  - within each selected region *R*, we consider all the segments of trajectories that lay in *R*, then we compute various descriptors (either the *macro* or the *micro* approach is chosen)
- **Step 3: Computing Interaction patterns**
  - sequences of regions are analyzed, searching for evolutions of the descriptors that occur frequently
  - Interaction patterns can be extracted using a conventional sequential patterns mining algorithm.

Experimental Study

- For evaluation, we used the NGSIM U.S. 101 dataset
  - It describes the traffic of vehicles over a 2100-feet-long and 6-lanes-wide covering a time interval of 45 minutes.
  - The data contains 6101 objects corresponding to approx. 4 million points, each given as a quadruple (ID, t, x, y),

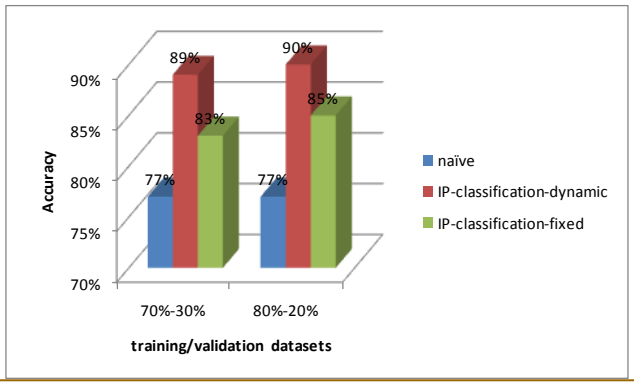


Experimental Study: settings

- We try to prove that using our framework we can accurately predict dangerous vs. non dangerous profiles. A dangerous vehicle (23% of the cases):
  - changes 2 or more lanes; or
  - moves 20% (or more) of the time periods above average speed; or
  - moves 10% (or more) of the time periods over the speed limit.
- We set the temporal period for **dynamic neighborhood** at 1.5 minute and the spatial tolerance at 5 meters
- As for the **fixed grid** approach, we consider spatiotemporal grids of 10 m (width) x 64 m (height) x 2 minutes
- Used Interaction Attributes: AVG_PERC_DISTANCE, AVG_PERC_DURATION, AVG_PERC_SPEED, AVG_PERC_ABS_ACCELER, VAR_PERC_ACCELER

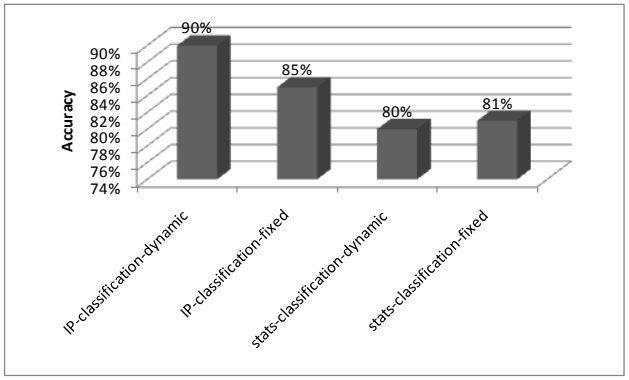
Experimental Study: results

- We compare three classifiers
  - Naïve (that always classifies drivers as “not dangerous”)
  - IP classification using dynamic approach
  - IP classification using fixed grid approach



Experimental Study: results

- We also compute simple statistics for each trajectory: average {speed, acceleration}, lanes changed etc:
  - for the dynamic neighborhoods and
  - for the fixed cells



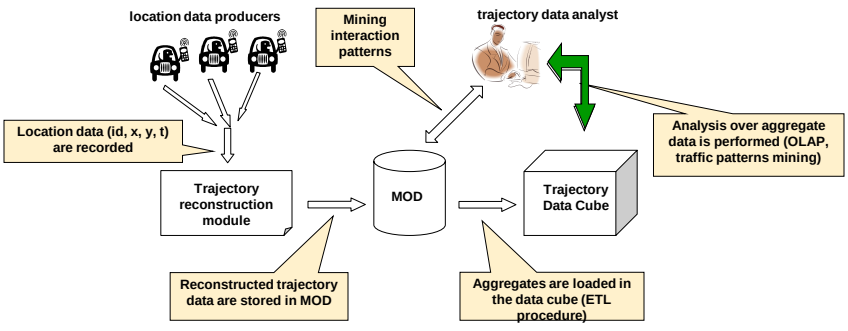
Synopsis and related publications

- We provided a framework for mining interaction patterns that enables spatiotemporal representation, synthesis and classification
- This framework allows an adequate understanding of what is happening to moving objects with regards to the context
  - i.e. the geographical space but also the interaction with other objects etc where they move.
- Publications
  - Nanni, M., Marketos, G., Quattrociocchi. W. Mining Interaction Patterns for Spatiotemporal Representation, Synthesis and Classification. *Manuscript prepared*.

Outline

- Preliminaries
- Efficient Trajectory Data Warehousing
  - A framework for Trajectory Data Warehousing
  - Ad-hoc OLAP on trajectory data for supporting multiple trajectory definitions
- Trajectory-inspired Data Mining
  - Mining Interaction Patterns for spatiotemporal representation, synthesis and classification
  - Mining Traffic Patterns for monitoring the traffic flow
- Building real-world Trajectory Data Warehouses
  - Visual OLAP analysis

Our contribution



The 'traffic' problem and challenges

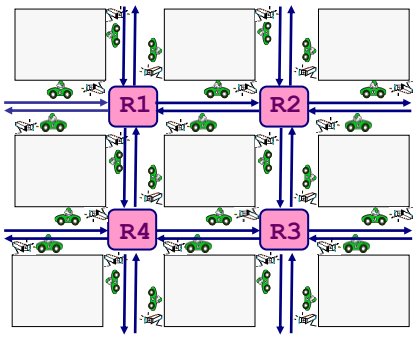
- Traffic in cities is a well-known problem:
  - the majority of people living in large cities and use cars for their transportation
  - the number of cars moving in a city increases from year to year
- Recently, the technology achievements allow us to collect huge amount of traffic related data:
  - mobile phones, traffic cameras, sensors, GPS devices …
- How can we exploit this huge volume of data so as to gain insights on the traffic problem?
  - The focus of our work is to detect how the different road segments of the network are related to each other.

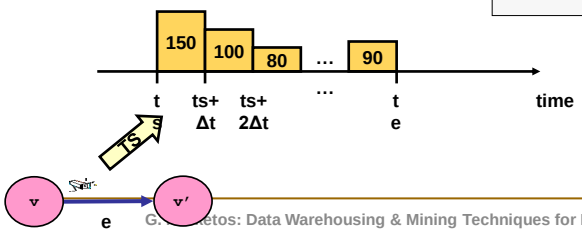
Related work

- (Li et al., 2007) proposed a technique for the discovery of hot routes in a road network
  - They introduced a density-based algorithm, called FlowScan, which requires the trajectories of the objects that move within the network
- (Lee et al., 2007) proposed a partition-and-group framework for trajectory clustering
  - This work concerns the trajectories of the objects, free movement and no some predefined network like, in our case, the road network
- (Kalnis et al., 2005) introduced the notion of moving clusters and (Spiliopoulou et al, 2006) focused on detecting transitions between clusters
  - These works require the IDs of the objects
- (Giannoti et al., 2007) proposed the notion of trajectory pattern as sequence of spatial areas that are temporally related

Traffic Series

- Road network: a fixed network consisting of a set of non-overlapping regions
  - regions = road intersections or POI's
- Each edge is associated with a (traffic) time series





Clustering traffic edges

- We employ a divisive hierarchical clustering algorithm
- The distance function is the global distance

$$dis(e_1, e_2) = a * dis_{shape}(e_1, e_2) + b * dis_{struct}(e_1, e_2) + c * dis_{value}(e_1, e_2)$$

- The notion of distance between two edges has to be defined:
  - Distance based on the corresponding traffic series "shape"
  - Distance according to the network graph topology
  - Distance based on the corresponding traffic series values

Value based distance between edges

- Let e_1, e_2 be two network edges and let

  $TS_1=\{(v_{1i},t_i)\}$, $TS_2=\{(v_{2i},t_i)\}$

  be their corresponding traffic series, $t_i \in [t_s, t_e]$
- The **value based distance** between e_1, e_2 is given by the Euclidean distance of their traffic series TS_1, TS_2:

$$dis_{value}(e_1,e_2) = dis_{value}(TS_1,TS_2) = \sqrt{\sum (v_1[t_i]-v_2[t_i])^2}, t_s \le t_i \le t_e$$

- The value based distance looks for traffic series with the same values

Shape based distance between edges

- Normalization of traffic series

$$\mu = \frac{1}{n}\sum_{i=1}^{n} v_i \qquad \sigma = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(v_i-\mu)^2}$$

$$v_i' = \frac{v_i - \mu}{\sigma}$$

  - $TS_1=\{(v_{1i},t_i)\}$
  - $TS_2=\{(v_{2i},t_i)\}$

  Normalization → $TS_1' = \{(v_{1i}',t_i)\}$, $TS_2' = \{(v_{2i}',t_i)\}$

- The **shape based distance** between e_1, e_2 is given by the Euclidean distance of their normalized traffic series TS_1', TS_2':

$$dis_{shape}(e_1,e_2) = dis_{shape}(TS_1,TS_2) = \sqrt{\sum (v_1'[t_i]-v_2'[t_i])^2}, t_s \le t_i \le t_e$$

- The shape based distance looks for traffic series of the *same shape*, i.e. edges that follow the same pattern of traffic

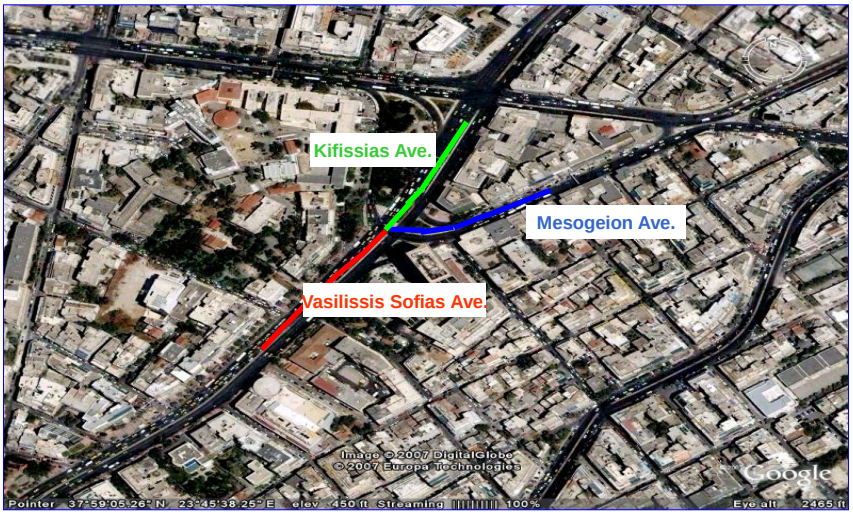
A 3-level clustering algorithm

- Initially all edges are placed into the same cluster
- Step 1 [Edges of similar shape] → L_1
  - The cluster is split into subclusters based on dis_{shape} and this continues until a split is caused by dis_{struct}
  - Output: clusters of edges with the same traffic shape
- Step 2 [Nearby edges] → L_2
  - The clusters generated by step 1 are further split according to dis_{struct} until a split occurs due to dis_{value}
  - Output: clusters of edges with the same traffic shape that are also nearby in the graph
- Step 3 [Edges of similar values] → L_3
  - The clusters generated by step 2 are further split based on dis_{value}
  - Output: clusters of edges with the same values

How can the road segments be related? - 1

How can the road segments be related? - 2

How can the road segments be related? - 3



Time-focused similarity measures

- We extend the similarity measures so as to be time-focused
- We define the *value-based distance* between edges e_1, e_2 during periods p and p' as the absolute distance of their corresponding traffic series TS_1, TS_2 at these periods:

$$dis_{value}\left(e_1^p, e_2^{p'}\right) = dis_{value}\left(TS_1^p, TS_2^{p'}\right)$$

- We define a shape window (p-w_b, p+w_a) in which we look for shape similarity We utilize three well-known techniques:
  - Euclidean distance: $dis_{shapeEu}\left(e_1^p, e_2^p\right) = dis_{value}\left(TS_1', TS_2'\right)$
  - Dynamic Time Warping (DTW):

$$dis_{shapeDTW}\left(e_1^p, e_2^p\right) = \sqrt{\sum\left(v_1[t_i] - v_2[t_j]\right)^2} \text{ and } \sum\left|v_1[t_i] - v_2[t_j]\right| = min_w\left[\sum_{k=1}^{K} d(w_k)\right]$$

  - Correlation-Coefficient (r): $dis_{shapeCC}\left(e_1^p, e_2^p\right) = r\left(TS_1^{(p-w_b, p+w_a)}, TS_2^{(p-w_b, p+w_a)}\right)$

Traffic relationships

- **Traffic propagation**
  - traffic from e_{12} propagates to e_{23}
  - This might indicate objects that continue moving in a highway
  - Conditions: $dis_{value}\left(TS_{e_1}^p, TS_{e_2}^{p'}\right) \geq t_v$

$$dis_{shape}\left(e_1^p, e_2^p\right) \geq t_s$$

- **Traffic split**
  - traffic from e_{12} splits into e_{23} and e_{26}
  - This might indicate objects that leave a highway and follow different directions to their destination
  - Conditions:

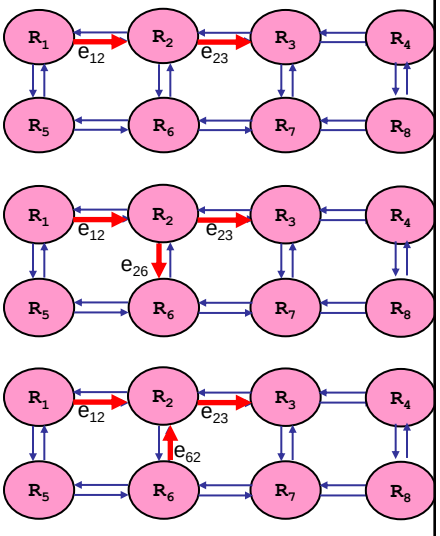
$$dis_{value}\left(TS_e^p, \sum_i (TS_{e_i}^{p'})\right) \geq t_v$$

$$\forall\, 1 \leq i \leq k : dis_{shape}\left(TS_e^p, TS_{e_i}^p\right) \geq t_s.$$

- **Traffic merge**
  - traffic to e_{23} merges traffic from e_{12} and e_{62}
  - This might indicate objects that enter a highway from different directions
  - Conditions:

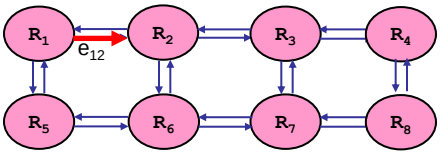
$$dis_{value}\left(\sum_i (TS_{e_i}^p), TS_e^{p'}\right) \geq t_v$$

$$\forall\, 1 \leq i \leq k : dis_{shape}\left(TS_{e_i}^p, TS_e^p\right) \geq t_s.$$

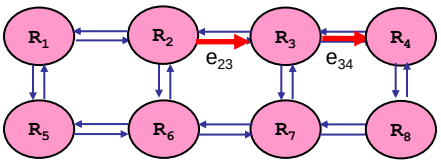


Traffic relationships

- **Traffic sink**
  - traffic in e_{12} is sank because there is no propagation or split relationship with its outgoing edges



- **Traffic source**
  - traffic starts (source) from e_{23} because there is no propagation or merge relationship with its incoming edges



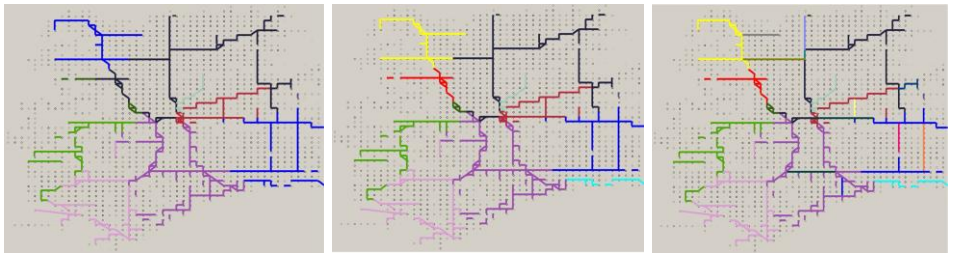
Detecting Traffic Relationships

- We propose an algorithm which creates possible pairs of incoming and outgoing edges and looks for all types of relationships
  - If more than one is found then it is possible to decide and characterizes this pair as complex
- Finally, the algorithm searches for sinks and sources

```
Algorithm Traffic-Relationship-Detector(ListOfVertices lov,
ValueSimThreshold t_v, ShapeSimThreshold t_s, TimeInterval w_b, TimeInterval
w_a)
1. FOR EACH Vertice v IN lov
2.  FOR EACH Period p
3.   //check incoming and outgoing edges for v
4.   IF incomingEdges(v) = 1 AND outgoingEdges(v) = 1 THEN
5.    checkForPropagate(v.e_in, v.e_out, p, t_v, w)
6.   ELSE IF incomingEdges(v) = 1 AND outgoingEdges(v) > 1 THEN
7.    //it is defined the set of outgoing edges to look for a
8.    //relationship between the set and the incoming edge
9.    le = defineOutEdges(v)
10.   checkForSplit(v.e_in, le, p, t_v, t_s, w)
11.  ELSE IF incomingEdges(v) > 1 AND outgoingEdges(v) = 1 THEN
12.   //it is defined the set of incoming edges to look for a
13.   //relationship between the set and the outgoing edge
14.   le = defineInSetEdges(v)
15.   checkForMerge(le, v.e_out, p, t_v, t_s, w)
16.  ELSE
17.   FOR EACH pair pr IN v
18.    rels = checkForPropagate(pr.e, pr.e', p, t_v, t_s, wb)
19.    rels += checkForSplit(pr.e, pr.e_i, p, t_v, t_s, w)
20.    rels += checkForMerge(pr.e_i, pr.e, p, t_v, t_s, w)
21.    //rels stores the number of total relationships found
22.    IF rels >1 THEN
23.    //a complex relationship is found
24.    complexRel(pr, p)
25.   END IF
26.  END FOR
27.  END IF
28. END FOR
29.  //consider as r the set of discovered relationships
30.  FOR EACH Edge e IN v
31.   IF v.e IN rightPart(r) AND NOT IN leftPart(r) THEN
32.    //the edge keeps its traffic at period p
33.    sinkEdge(e, p)
34.   ELSE IF v.e in leftPart(r) AND NOT IN rightPart(r) THEN
35.    //the edge keeps its traffic at period p
36.    sourceEdge(e, p)
37.   END IF
38.  END FOR
39. END FOR
```

Experimental Study: Clustering edges

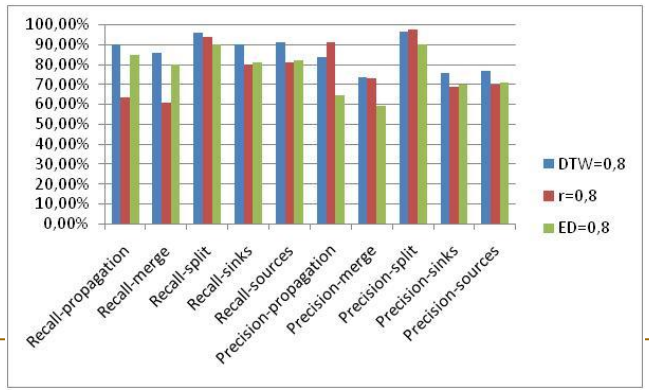
- For evaluation, we used synthetic data generated by the network-based data generator developed by Brinkhoff
  - 2000 moving objects with 400 sampled positions per each object
- Below the clusters discovered at the various execution levels



Edges of similar shape | Nearby edges of similar shape | Nearby edges of similar shape & value

Experimental Study: Discovering relationships

- For evaluation, we used a real data set consisting of 59263 location points, belonging to 990 objects moving in the greater Milano area, map matched on 9256 edges
- We experimented with various similarities measures and we compute the precision/recall:



Synopsis and related publications

- We consider the problem of mining traffic flow in a road network placed at regions of interest (crossroads, etc.).
- We employ edge similarity measures based on time series comparison
- Traffic relationships are introduced and formally defined
- Publications
  - Marketos, G., Theodoridis, Y., Mobility Data Warehousing & Mining. *Proceedings of VLDB PhD Workshop*, 2009.
  - Ntoutsi, I., Mitsou, N., Marketos, G., Traffic mining in a road-network: How does the traffic flow?. *IJBIDM* 3(1), 2008.

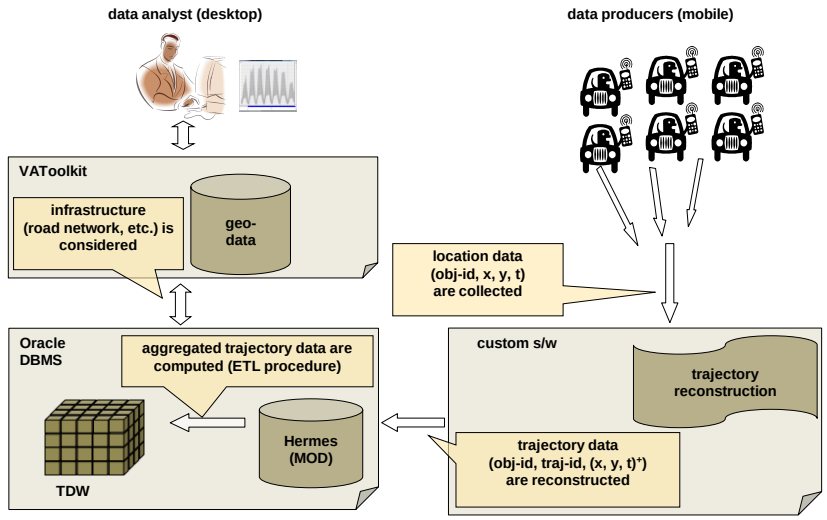
Outline

- Preliminaries
- Efficient Trajectory Data Warehousing
  - A framework for Trajectory Data Warehousing
  - Ad-hoc OLAP on trajectory data for supporting multiple trajectory definitions
- Trajectory-inspired Data Mining
  - Mining Interaction Patterns for spatiotemporal representation, synthesis and classification
  - Mining Traffic Patterns for monitoring the traffic flow
- Building real-world Trajectory Data Warehouses
  - Visual OLAP analysis

T-WAREHOUSE functionality

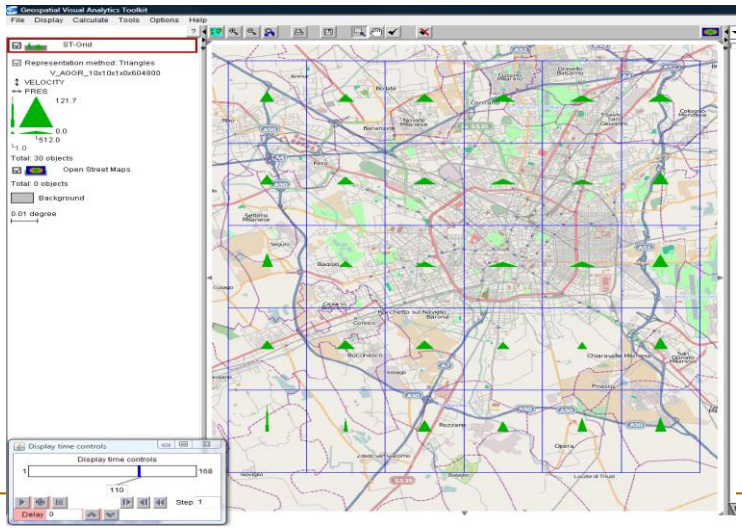
- Supports advanced kind of analysis:
  - Where does the highest traffic appear? At what hour?
  - What happens exactly at the road network level?
  - How does the movement propagate from place to place?
- Incorporates:
  - Trajectory reconstruction
    - convert sampled time stamped positions into trajectory data and store in MOD
  - TDW feeding
    - the ETL process performs spatiotemporal range queries so as to fill in the measures with the appropriate values for each base cell
  - Aggregation
    - an approximate solution is used to face the distinct count problem

T-WAREHOUSE architecture



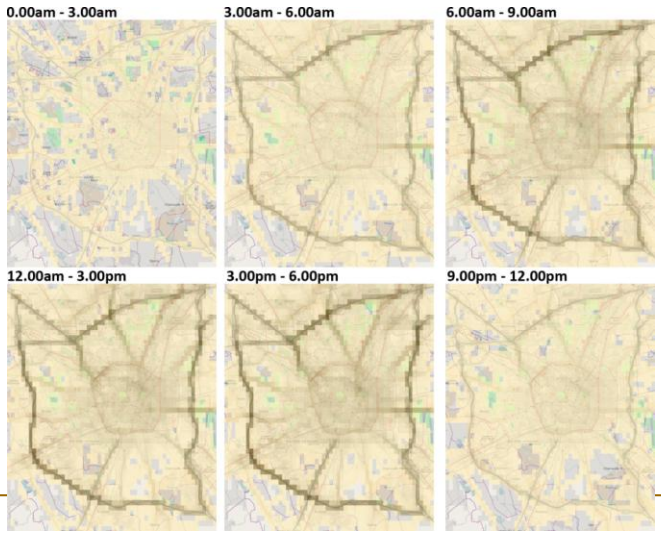
T-WAREHOUSE screenshots

- Relationship between **Presence** and **Velocity**



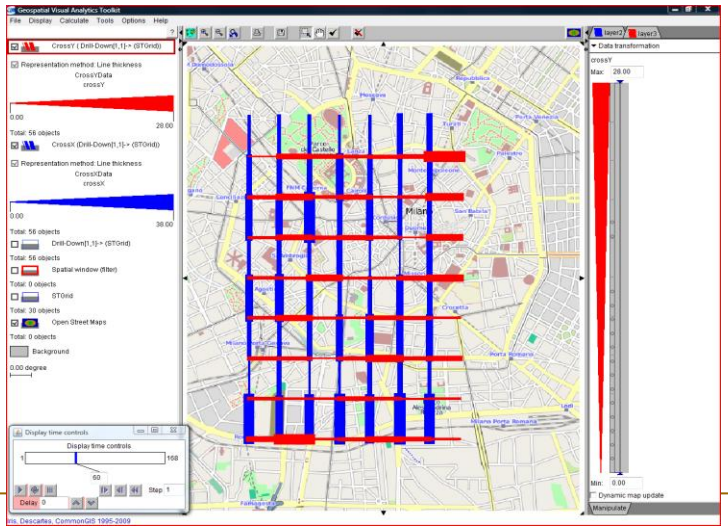
T-WAREHOUSE screenshots

- **Presence** on Tuesday at base granularity



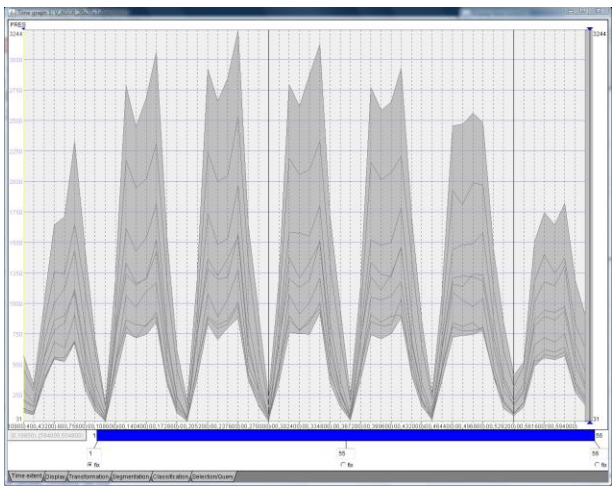
T-WAREHOUSE screenshots

- Visualization of **crossX** and **crossY**



T-WAREHOUSE screenshots

- Evolution of **Presence** during the week



Synopsis and related publications

- We presented T-WAREHOUSE, a system that integrates all the required steps for Visual TDW, from *trajectory reconstruction* and ETL processing to Visual OLAP analysis on mobility data.
- We illustrated the architectural aspects of our framework and investigates its power, flexibility and efficiency for applying OLAP analysis on real world mobility data

- Publications
  - Leonardi, L., Marketos, G., Frentzos, E., Giatrakos, N., Orlando, S., Pelekis, N., Raffaetà, A., Roncato, A., Silvestri, C., Theodoridis, Y. T-Warehouse: Visual OLAP Analysis on Trajectory Data. *Proceedings of ICDE*, 2010 (to appear).

Summary

- We proposed a complete framework for Trajectory Data Warehousing
- We extended the OLAP model so as to support multiple trajectory definitions
- We proposed a framework for extracting interaction patterns
- We provided an approach for extracting traffic patterns focusing on
  - the clustering of traffic edges; and
  - the discovery of relationships between the edges
- We presented a prototype for Visual TDW

Open issues

- **[trajectory reconstruction]** Explore intelligent ways to automatically extract proper values of trajectory reconstruction parameters
  - Identify different movement types so as to apply customized reconstruction
- **[trajectory OLAP]** Extend the TDW so as to include both numerical and movement based measures
  - Examples of such a measure are the centroids and medoids of the set of trajectories that describe the trend of movement within a cell
- **[ad-hoc OLAP]** Explore cross-cell measures and sophisticated methods for materialization
- **[interaction mining]** Utilize interaction descriptors in OLAP operations
- **[traffic mining]** Discover complex relationships between edges

Support

- EU FP6-14915 IST/FET Project GeoPKDD (Geographic Privacy-aware Knowledge Discovery and Delivery)
- PENED'2003 grant funded by the General Secretariat for Research and Technology of the Greek Ministry of Development.

Thank you!

Questions?

References

- Dockstader, S.L. Motion Trajectory Classification for Visual Surveillance and Tracking. *Proceedings of AVSS*, 2006.
- Geurts, P. Pattern Extraction for Time Series Classification. *Proceedings of PKDD*, 2001.
- Giannotti, F., Nanni, M., Pinelli, F., and Pedreschi, D. Trajectory pattern mining. Proc. KDD, 2007.
- Han, J., Stefanovic, N., and Koperski, K. Selective Materialization: An Efficient Method for Spatial Data Cube Construction. Proc. PAKDD, 1998.
- Kalnis, P., Mamoulis, N., and Bakiras, S. On discovering moving clusters in spatio-temporal data. *Proc. SSTD*, 2005.
- Leonardi, L., Orlando, R., Raffaetà, A., Roncato, A., and Silvestri, C Frequent spatio-temporal patterns in trajectory data warehouses. *Proceedings of SAC*, 2009.
- Lee, J., Han, J., Li, X., and Gonzalez, H. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. VLDB Endowment 1(1), pp. 1081-1094, 2008.
- Lee, J., Han, J., and Whang, K. Trajectory Clustering: A Partition-and-Group Framework. Proceedings of ACM SIGMOD, 2007.
- Li, X., Han, J., Lee, J.-G. and Gonzalez, H.: "Traffic density-based discovery of hot routes in road networks." in Proc. 10th International Symposium on Spatial and Temporal Databases., 2007.
- Orlando, S., Orsini, R., Raffaetà, A., Roncato, A., and Silvestri, C. Spatio-Temporal Aggregations in Trajectory Data Warehouses. Proc. DaWaK, 2007.
- Pfoser, D., Jensen, C.S., and Theodoridis, Y. Novel Approaches to the Indexing of Moving Object Trajectories, Proc. VLDB, 2000.
- Shekhar, S.., Lu, C., Tan, X., Chawla, S., Vatsavai, R. Map Cube: a Visualization Tool for Spatial Data Warehouses, Chapter in Geographic Data Mining and Knowledge Discovery. Taylor and Francis, 2001.
- Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y., and Schult, R. Monic: modeling and monitoring cluster transitions. Proc. KDD, 2006.
- Tao, T., and Papadias, D. Historical Spatio-Temporal Aggregation. ACM TODS, 23(1):61-102, 2005.
- Tao, Y., Kollios, G., Considine, J., Li, F., and Papadias, D. Spatio-Temporal Aggregation Using Sketches. Proc. ICDE, 2004.