Uncertainty for Privacy and 2-Dimensional Range Query Distortion

Spyros Sioutas, Emmanouil Magkos, Ioannis Karydis
Department of Informatics, Ionian University, Corfu, Greece
e-mail: {sioutas,emagos,karydis}@ionio.gr

V.S. Verykios
Department of Computer and Communication Engineering,
University of Thessaly, Greece
e-mail: verykios@inf.uth.gr

In this work, we study the problem of privacy-preserving data publishing in moving objects databases. In particular, the trajectory of a mobile user on the plane is no longer a polyline in a two-dimensional space, instead it is a two-dimensional surface of fixed width $2A_{min}$, where A_{min} defines the semidiameter of the minimum spatial circular extent that must replace the real location of the mobile user on the XY-plane, in the anonymized (kNN) request. Since a malicious attacker can observe that during the time, many of the neighbours ids change except for a small number of users, the desired anonimity is not achieved and the whole system becomes vulnerable to attackers. Thus, we reinforce the privacy model by clustering the mobile users according to their motion patterns in (u, θ) plane, where u and θ define the velocity measure and the motion direction (angle) respectively. In this case the anonymized (kNN) request lookups neighbours, who belong to the same cluster with the mobile requester in (u, θ) space: So, we know that the trajectory of the k-anonymous mobile user is within this surface, but we do not know exactly where. We transform the surface's boundary poly-lines to dual points and we focus on the information distortion introduced by this space translation. We develop a set of efficient spatio-temporal access methods and we experimentally measure the impact of information distortion by comparing the performance results of the same spatio-temporal range queries executed on the original database and on the anonymized one.

Categories and Subject Descriptors: Moving Objects Databases [**Smart and intelligent computing**]:

General Terms:
Additional Key Words and Phrases: Uncertainty, Privacy, Anonymity, Moving Objects Databases, Voronoi Clustering

---

Preliminary version of this work was presented in the International Conference on Privacy in Statistical Databases (PSD2010), and appeared in Vol.6344, pp.85-96 of Springer LNCS series.

1. INTRODUCTION

The technological advances in sensors and wireless communications have made possible the offering of high location accuracy in location tracking at a low cost [Bajaj et al. ; Clarke 2001; Hoh et al. 2007]. Such accuracy gave rise to a series of location-based applications that exploit positional data to offer high-end location-based services (LBS) to their subscribers [D'Roza and Bilchev 2003]. On the other hand mobile users require the protection of their context information (*e.g.*, location and/or identity information) against privacy adversaries (*e.g.*, Big-brother type threats, users profiling, unsolicited advertising) [Hauser and Kabatnik 2001; Gruteser and Grunwald 2003; Duckham and Kulik 2006; Ardagna et al. 2009]. The privacy issue is amplified by the requirement in modern telematics and location-aware applications for real-time, continuous location updates and accurate location information (*e.g.*, traffic monitoring, asset tracking, location-based advertising, location-based payments, routing directions) [Gruteser and Liu ; Kulik 2009; Ghinita 2009].

We consider a population of mobile users who are supported by some telecommunication infrastructure, owned by a telecom operator. Every user periodically transmits through his/her mobile device a location update to some traffic monitoring system residing in a untrusted Internet-connected LBS provider. A trusted server in the telecom operator plays the role of a proxy that mediates between the user and the LBS provider. Our setting involves an LBS provider that is requested to efficiently answer Range Queries (RQs) of mobile users moving on the plane. For example historical queries of the form:

Select Count (Mobile-User-Ids)
From Moving_Object_Database (MOD)
where spatial-area=MBR('Acropolis') and $t\_past < time < t\_now$

or prediction queries of the form:

Select Count (Mobile-User-Ids)
From Moving_Object_Database (MOD)
where spatial-area=MBR('Acropolis') and $t\_now < time < t\_future$

that provide traffic monitoring services to mobile users. This type of queries focuses on the problem of indexing mobile users in two dimensions and efficiently answering range queries over the users locations. This problem is motivated by a set of real-life applications such as intelligent transportation systems, cellular communications, and meteorology monitoring.

Our privacy requirements, include:

—*Location privacy*: The protocol does not reveal the (exact) user's location information to the LBS provider.

—*Tracking protection (Unlinkability)*: The LBS provider is not able to link two or more successive user positions.

In our privacy threat model we assume an attacker who has access to the k-anonymized (cloaked) location queries and to the algorithms used by the trusted

server to support user privacy. We assume that the proxy server of the telecom operator is trusted on not to collude with the LBS provider or other internal/external non-authorized entities in order to expose the atomic location updates of system users.

Traditionally, the transformation of the exact user location to a spatiotemporal area is achieved through the use of the k-anonymity principle for relational data [Samarati 2001; Sweeney 2002], which requires that each record in a given dataset is indistinguishable from at least $k-1$ other records with respect to a certain set of identifying variables, collectively known as the "quasi-identifier". The k-anonymity principle requires that the spatiotemporal area that is generated by the trusted server from the exact location of the mobile user is such that the identity of the user cannot be disclosed with a probability that is larger than $1/k$, among $k-1$ other users. The k-anonymity primitive is essential to protect the privacy of the users, starting from the point of request for a RQ service and continuing for as long as the requested service withstands completion.

1.1 A high level description of our framework

As part of our framework, we deliver the type of k-trajectory privacy [Gkoulalas-Divanis et al. 2009] that identifies $k-1$ users that are close to the requester at the time of request and thus could have issued the request. This includes a minimum circular spatial area A_{min} around the requester, where the participants of the anonymity set should be searched for so that the user is adequately covered up. The proposed privacy framework relies on a user privacy profile that stores the necessary information related to his/her privacy requirements. This includes (i) the preferred value of k (in k-anonymity) for each requested RQ service, (ii) the minimum circular spatial area A_{min}, around the requester, where the participants of the anonymity set should be searched for so that the user is adequately covered up. This threshold defines the minimum extent of the spatial area that must replace the real location of the user, in the anonymized request.

The proposed framework deals with the event of failure in the provision of k-anonymity, in the case where the number of participants inside this minimum spatial area is less than k-1. In this case, the trusted server postpones the servicing of the user request for a small period of time. After that, if the anonymization process fails again, the requester is protected by blocking the servicing of the request.

The proposed privacy framework can be reinforced by clustering the mobile users according to their motion patterns on the (u, θ) plane, where u and θ define the velocity measure and the motion direction (angle) respectively. In this case the anonymized (kNN) request lookups neighbours, who belong to the same cluster with the mobile requester in (u, θ) space.

There are two basic approaches used when trying to handle RQ services; those that deal with discrete and those that deal with continuous movements. In a discrete environment the problem of dealing with a set of moving objects can be considered to be equivalent to a sequence of database snapshots of the object positions/extents taken at time instants $t_1 < t_2 < \ldots$, with each time instant denoting the moment where a change took place. From this point of view, the indexing problems in such environments can be dealt with by suitably extending indexing techniques from the area of temporal [Salzberg and Tsotras 1999] or/and spatial databases [Gaede

and Günther 1998]; in [Manolopoulos et al. 2000] it is elegantly exposed how these indexing techniques can be generalized to handle efficiently queries in a discrete spatiotemporal environment. When considering continuous movements there exists a plethora of efficient data structures [Jensen et al. 2004; Kollios et al. 1999; Papadopoulos et al. 2002; Patel et al. 2004; Tao et al. 2003]. The common thrust behind these indexing structures lies in the idea of abstracting each object's position as a continuous function $f(t)$ of time and updating the database whenever the function parameters change; accordingly an object is modeled as a pair consisting of its extent at a reference time (design parameter) and of its motion vector. One categorization of the aforementioned structures is according to the family of the underlying access method used. In particular, there are approaches based either on R-trees or on Quad-trees. On the other hand, these structures can be also partitioned into (a) those that are based on geometric duality and represent the stored objects in the dual space [Kollios et al. 1999; Patel et al. 2004] and (b) those that leave the original representation intact by indexing data in their native n-d space [Beckmann et al. 1990; Papadopoulos et al. 2002; Tao et al. 2003]. The *geometric duality transformation* is a tool heavily used in the Computational Geometry literature, which maps hyper-planes in R^n to points and vice-versa. Based on the proposed privacy model we implement a framework that uses the Spatial extensions of MySql 5.x to offer privacy in RQ services.

In this work, we study the problem of privacy-preserving data publishing in moving objects databases. In particular, the trajectory of a mobile user on the plane is no longer a polyline in a two-dimensional space, instead it is a two-dimensional surface: we know that the trajectory of the mobile user is within this surface, but we do not know exactly where. We transform the surface's boundary poly-lines to dual points [Kollios et al. 1999; Papadopoulos et al. 2002] and we focus on the information distortion introduced by this space translation. We develop a set of efficient spatio-temporal access methods and, we experimentally measure the impact of information distortion by comparing the performance results of the same spatio-temporal range queries executed on the original database and on the anonymized one.

The remainder of the paper is organized as follows. Section 2 lays out the system model for our proposed methodology and gives a formal description of the problem, while Section 3 discusses preliminary notions and results that are used throughout the paper. In Section 4 we present the method of transforming the trajectory poly-lines to two-dimensional surfaces. In Section 5 we present:(1) the duality transformation of surface's boundary poly-lines, (2) a simple observation based on which we enforce the privacy model of our method and (3) the information distortion introduced by this space translation. Section 6 presents an extended experimental evaluation and section 7 concludes the paper.

2. SYSTEM MODEL AND PROBLEM DESCRIPTION

This section lays out the system model in which our proposed methodology is applicable. The considered system framework interconnects a set of registered LBS providers to a set of subscribed users, to enable the offering of RQSs in a privacy-aware manner. Figure 1 depicts our system framework that has two main

components: the trusted server, which is assigned the task of trajectory privacy preservation, and the service providers, which offer the actual RQSs. As part of its functionality, the trusted server maintains a database that stores the privacy profiles of the users, their history of movement, as well as knowledge of motion patterns.

A user of the system can be any individual who is equipped with a mobile device and is registered to the trusted server. Upon his/her registration, the user is assigned a unique profile in the system and his/her movement is subsequently monitored by the trusted server. The tracking of user movement is made possible through the transmission of periodic location updates from the mobile device of the user to the trusted server. A privacy profile for a user consists of a set of tuples (sid, K, A_{min}), which define his/her privacy requirements that involve (i) the value of k for each provided service sid, where k indicates that the user wants to remain K-anonymous when using this service, and (ii) the minimum area of generalization A_{min}, where A_{min} indicates the minimum spatial area to which the anonymized request should point to.

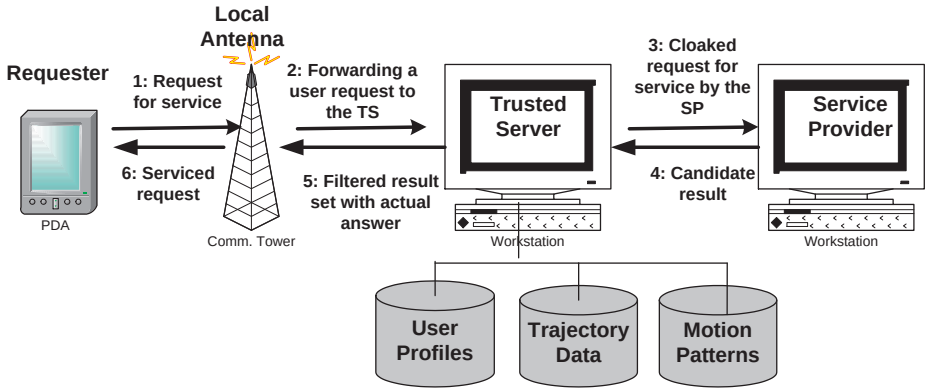

Fig. 1. The System Architecture

Figure 1 depicts the regular scenario for the privacy-aware provision of RQSs. Whenever a user requests an RQS (step 1), the trusted server receives the original request (step 2), examines the point of request and determines the appropriate anonymity strategy that needs to be enforced. Following that, the original request is transformed to a 'secure' equivalent that preserves the K-anonymity of the requester and is forwarded to the appropriate service provider to be serviced (step 3). After the request has been serviced, the result set, computed by the service provider, is returned to the trusted server (step 4) where it is filtered to contain only the actual answer that is subsequently forwarded to the requester (steps 5 and 6). This concludes the privacy-aware provision of the service.

We consider a database that records the position of moving objects in two dimensions on a finite terrain. We assume that objects move with a constant velocity vector starting from a specific location at a specific time instant. Thus, we can calculate the future object position, provided that its motion characteristics remain

the same. Velocities are bounded by $[u_{min}, u_{max}]$. Objects update their motion information, when their speed or direction changes. The system is dynamic, i.e. objects may be deleted or new objects may be inserted. Let $P_z(t_0) = [x_0, y_0]$ be the initial position at time t_0 of object z. If object z starts moving at time $t > t_0$, its position will be $P_z(t) = [x(t), y(t)] = [x_0 + u_x(t - t_0), y_0 + u_y(t - t_0)]$, where $U = (u_x, u_y)$ is its velocity vector. For example, in Figure 2 the lines depict the objects trajectories on the (t, y) plane. We would like to answer queries of the form:

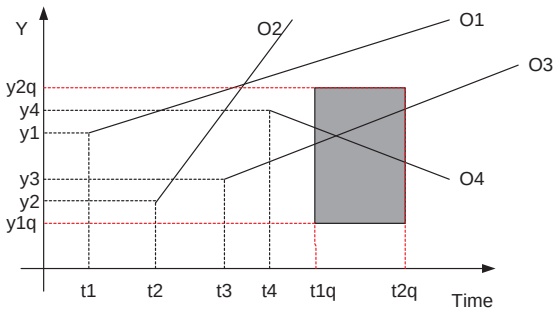

Fig. 2. Trajectories and query in (t, y) plane

"Report the objects located inside the rectangle $[x_{1_q}, x_{2_q}] \times [y_{1_q}, y_{2_q}]$ at the time instants between t_{1_q} and t_{2_q} (where $t_{now} \leq t_{1_q} \leq t_{2_q}$), given the current motion information of all objects".

3. PRELIMINARIES

In the following we briefly describe the most basic methods for indexing mobile objects, as well as the best current clustering algorithms that will be used throughout the paper and specifically for clustering mobile objects with similar motion-pattern.

3.1 Literature Survey - The Most Basic Methods for Indexing Mobile Objects

In the sequel, let N denote the input size (number of stored objects), B the block size, K the output size and thus $n = N/B$ and $k = K/B$ are the size of the input and output in blocks, respectively.

In [Kollios et al. 1999] a set of indexing techniques was presented, which used the geometric duality transformation at the cost of increasing by one the dimensionality. Hence, for the one-dimensional case they reduced the indexing problem to the two-dimensional simplex range searching problem and they employed external memory partition trees to solve their indexing problem in $O(n)$ space, $O(n^{1/2} + k)$ I/Os query time and $O(\log n)$ I/Os update time.

Partition trees, though having a guaranteed worst case performance, are generally considered non-practical since they entail large hidden factors. Thus, in [Kollios et al. 1999] two more structures were presented, one based on k-d-trees and a more complex one based on B$^+$-trees; both structures used linear space and work well on the average. Moreover, they extended their results in the two-dimensional case for two distinct versions of the problem; first, the objects were allowed to move

on a network of one-dimensional routes, and, second, the objects were allowed to move arbitrary. The first version reduced to a number of one-dimensional subproblems that use the previously described structures, whereas the second is equivalent (through geometric duality) to simplex range queries in $\Re^3$, which can be solved in $O(n^{2/3} + k)$ I/Os with the use of external memory partition trees.

In [Agarwal et al. 1997] the above results were further refined. A new version of partition tree was introduced to handle the indexing problem in the plane in $O(n)$ space, $O(n^{1/2} + k)$ query time, and $O(\log_B n)$ expected amortized update time; the results could apply in higher-dimensional spaces as well, degrading only the update time (it became $O(log_B^2 n)$ I/Os). If it is assumed that the queries arrive in chronological order, then the query time can be further reduced to $O(\log_B^2 n / \log_B \log_B n)$ I/Os; this is achieved by employing the kinetic data structures framework at external range trees. Moreover, by combining multiversion kinetic data structures with partition trees, they developed an indexing scheme with small query time for near-future queries and increased time for distant in the future queries under the bound of $O(n^{1/2+\epsilon} + k)$ I/Os. Finally, an indexing technique was described for handling $O(\delta)$-approximate queries; the query time was $(n^{1/2+\epsilon}/\delta)$, the expected update time $O(log_B^2 n/\delta)$ and the space $O(n/\delta)$ disk blocks.

The TPR tree [Saltenis et al. 2000] in essence is an R*-tree generalization to store and access linearly moving objects. The leaves of the structure store pairs with the position of the moving point and the moving point ID, whereas internal nodes store pointers to subtrees with associated rectangles that minimally bound all moving points or other rectangles in the subtree. The difference to the classical R*-tree lies in the fact that the bounding rectangles are time-parameterized (their coordinates are functions of time). It is considered that a time-parameterized rectangle bounds all enclosed points or rectangles at all times not earlier than current time. The algorithms for search and update operations in the TPR tree are straightforward generalizations of the respective algorithms in the R*-tree. Moreover the various kinds of spatiotemporal queries can be handled uniformly in 1-, 2-, and 3-dimensional spaces.

The TPR-tree constituted the base structure for further developments in the area [Saltenis and Jensen 2002]. An extension to the TPR-tree was proposed in [Tao et al. 2003], the so-called *TPR*-tree*. The main improvement lies in the update operations, where it is shown that local optimization criteria (at each tree node) may degrade seriously the performance of the structure and more particularly in the use of update rules that are based on global optimization criteria. They proposed a novel probabilistic cost model to validate the performance of a spatiotemporal index and analyze with this model the optimal performance for any data-partition index.

The STRIPES index [Patel et al. 2004] is based on the application of the duality transformation and employs disjoint regular space partitions (disk based quadtrees [Gaede and Günther 1998]); the authors claim, through the use of a series of implementations, that STRIPES outperforms TPR*-trees for both update and query operations.

Finally, in [Sioutas et al. 2008] a new access method (LBTs) was presented for indexing mobile objects that move on the plane to efficiently answer range queries

about their future location. It has been proved that its update performance is the most efficient in all cases. Regarding the query performance, the superiority of LBTs method has been shown as far as the query rectangle length remains in realistic levels (greatly outperforming other methods). If the query rectangle length becomes extremely huge in relation to the whole terrain, then STRIPES is better than any other solution, however, only to a small margin in comparison to LBTs method.

3.2 Literature Survey - The Most Basic Clustering Algorithms

Clustering is the method of grouping the data into sets so that data points within the set should have high similarity while those within different sets are dissimilar [Han and Kamber 2007], [Dunham 2007].

3.2.1 *K-Means Algorithm.* A simple and widely used clustering algorithm is k-means clustering [MacQueen 1965]. The K-Means algorithm is based on squared error minimization method. We discuss the K-Means algorithm as follows:

Algorithm 1 K-Means

1: Randomly choose k data points from the whole data set as initial cluster centers;
2: Calculate Euclidean distance of each data point from each cluster center and assign the data points to its nearest cluster center;
3: Calculate new cluster center so that the squared error distance of each cluster should be minimized;
4: Repeat step 2 and 3 until the cluster centers do not change;
5: Stop the process;

The time complexity of K-Means algorithm is $O(nkt)$, where n is the number of data objects, k the number of clusters and t the number of iterations.

The obvious shortcomings of the basic K-Means clustering are that the number of clusters needs to be determined in advance and its computational cost with respect to the number of observations, clusters, and iterations. Though, K-Means is efficient in the sense that only the distance between a point and the k cluster centers –not all other points– needs to be (re)computed in each iteration. Typically the number of observations n is much larger than k.

There have been many approaches trying to alleviate both of these problems. Variations of K-Means clustering that are supposed to cope without prior knowledge of the number of cluster centers have been presented [Pelleg and Moore 2000] [Hamerly 2004]. While the proposed solutions to K-Means clustering certainly improve its practical behavior, they do not overcome the fundamental problems associated with the algorithm.

In [Koivistoinen et al. 2006] density-based algorithms have been proposed as a better alternative to identifying the correct number of clusters in the data. These algorithms make use of Voronoi diagrams (as in [Kao et al. 2010] and [P.S. and V. 2009]) and are statistically significantly more accurate than K-Means clustering.

3.2.2 *Voronoi Diagram.* In the field of computational geometry Voronoi diagram is one of the most important and useful techniques. Given a set D of n data points o_1, o_2, o_3,... o_n in a plane, the Voronoi diagram ($Vor(D)$) is a subdivision of the plane into Voronoi cells ($V(o_i)$ for o_i) (see Figure 3). The Voronoi cells are the set of points u that are closer to o_i than any other point in D. i.e. $V(o_i) = \{u|d(o_i, u) \leq d(o_j, u) \forall j \neq i\}$, where d is the Euclidian distance. The Voronoi diagram divides the plane into n convex polygon regions (for each o_i), the vertices (P of Figure 3) of the diagram are the Voronoi vertices, and the borders between two adjacent Voronoi cells are called as Voronoi edges (E of Figure 3). Note that each Voronoi vertex is the center of a circle touching three or more data points lying in its adjacent Voronoi cells (see Figure 3).

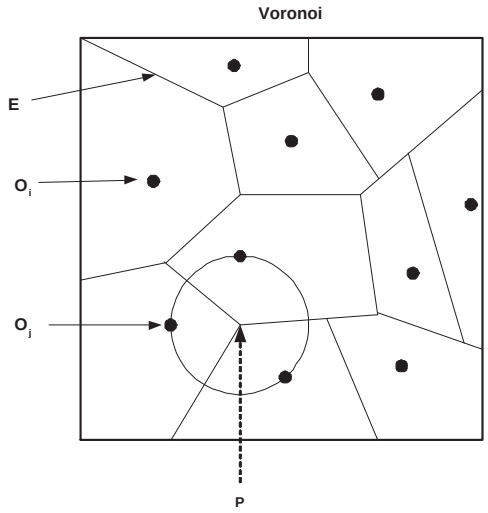

Fig. 3. Voronoi Diagram

3.2.3 *Voronoi Clustering.* The algorithm presented in [Koivistoinen et al. 2006], begins by constructing the Voronoi diagram for S, where $S = s_1, \ldots, s_n \subseteq \Re^d$ is a given data set of n points which is going to be clustered. The computational complexity of Voronoi diagram construction in the general case is $O(n \log n)$ [Aurenhammer et al.], but requires only linear time in some restricted cases [Aggarwal et al. 1989]. The algorithm is given as input parameter a threshold value max indicating the maximum volume allowed to a cell that still can be combined into an evolving cluster.

In the Voronoi diagram for the data each point makes up its own cell. The volumes (areas in case of the plane) of the cells are computed (or approximated) next. This approximation is not very accurate in small dimensions, but it improves as the number of dimensions grows. Each cell is associated with a class label. The algorithm assigns increasing integer values as class labels. The point having the smallest Voronoi cell is handled first.

Algorithm 2 Voronoi Clustering(set S, real max)

1: Construct the Voronoi diagram for the sample $S = s_1, \ldots, s_n$;
2: Approximate the Voronoi cell volumes and order the points accordingly. Without loss of generality, let the obtained order be $s_1, \ldots, s_n$;
3: For i = 1 to n do
4: If the volume of the cell R_i associated with s_i is at most max then
5: (a) For j = 1 to i - 1 do
6: If cluster C_j and cell R_i are adjacent then Merge them together;
7: If R_i has no class number yet then it assumes that of C_j else both assume the minimum class label among those of C_j and R_i;
8: (b) If cell R_i has no class number then assign a new one to it;
9: else assign R_i to the closest neighboring cluster;

In the Voronoi diagram of a point set the cell boundaries are by definition equally distant from the relevant points. Thus, the Voronoi diagram naturally gives raise to a kind of maximum margin separation of the data. The algorithm combines existing cells instead of creating artificial center points like, e.g., K-Means does. Hence, the maximal margins are maintained also in the clustered data. Since there are no reasons for placing the cluster border closer to one or other cluster, maximal margin clustering seems a natural choice.

One can view Voronoi clustering as a change of paradigm from the K-Means, whose Euclidean distance minimization can be made autonomous with respect to the number of clusters only by optimizing a parameter that is too hard to learn.

A similar pruning technique that is based on Voronoi diagrams in order to reduce the number of expected distance calculation, was proposed in [Kao et al. 2010], in case the mobile objects draw uncertain locations.

4. TRAJECTORY POLY-LINES AND TWO-DIMENSIONAL SURFACES

For every mobile user, we calculate a circular range query with center its current 2-D location and radious a given value R defined by a minimum circular spatial area A_{min}. If this circular spatial area includes at least $k-1$ other neighbours, then the mobile user is adequately covered up. Otherwise, if the number of participants inside this minimum spatial area is less than k-1, the trusted server postpones the servicing of the user request for a small period of time. After that, if the anonymization process fails again, the requester is protected by blocking the servicing of the request. As a result, consecutive circular spatial areas construct a 2-D buffer defined by its upper and lower boundary poly-lines y' and y'' respectively, which anonymize the original trajectory y of mobile user A (see figure 4). By using the Spatial extensions of MySql 5.x we can create each mobile user as 2-dimensional point as follows:

```
CREATE TABLE Points (
name VARCHAR(20) PRIMARY KEY,
location Point NOT NULL,
description VARCHAR(200),
SPATIAL INDEX(location)
);
```

In order to obtain points in a circular area as a counted result ordered by distance from the center of the selection area, we write:

SELECT COUNT(name), AsText(location), SQRT(POW(ABS(X(location) - X(@center)), 2) + POW(ABS(Y(location) - Y(@center)), 2)) AS distance
FROM Points
WHERE Intersects(location, GeomFromText(@bbox))
AND SQRT(POW(ABS(X(location) - X(@center)), 2) + POW(ABS(Y(location) - Y(@center)), 2)) ≤ @R
ORDER BY distance;

If the result returned is less than k-1, the trusted server postpones the servicing of the user request for a small period of time.

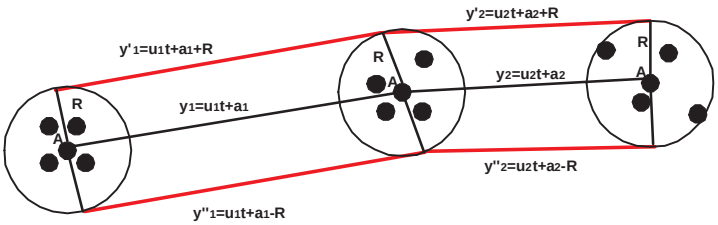

Fig. 4. 2-D Buffer and Boundary Trajectories y' and y'' of mobile user A

So, the RQ service depicted in figure 2, was transformed to the Privacy-Aware RQ service depicted in the figure 5:

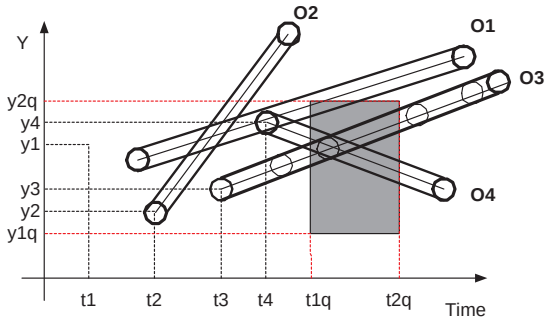

Fig. 5. Boundary Trajectories and query in (t, y) plane

If the whole buffer lies inside the query area (see the buffers of mobile users O_3 or O_4 in figure 5), meaning that both upper and lower boundary poly-lines lie in the query rectangle then the same holds for the original trajectory. If the whole buffer lies outside the query area (see the buffer of mobile user O_2 in figure 5), meaning

that both upper and lower boundary poly-lines lie outside the query rectangle then the same holds for the original trajectory. In the worst-case, we face the distortion effect, where one of the two boundary poly-lines lie in the query rectangle (see the buffer of mobile user O_1 in figure 5). In the later case, TS (Trusted Server) has to check out what happens with the original trajectory.

5. DUALITY TRANSFORMATION OF BOUNDARY-TRAJECTORIES AND INFORMATION DISTORTION

The duality transform, in general, maps a hyper-plane h from R^n to a point in R^n and vice-versa. In this subsection we briefly describe how we can address the problem at hand in a more intuitive way, by using the duality transform on the 1-d case.

5.1 Hough-X transform

One duality transform for mapping the line with equation $y(t) = ut + a$ to a point in R^2 is by using the dual plane, where one axis represents the slope u of an objects trajectory (i.e. velocity), whereas the other axis represents its intercept a. Thus we get the dual point (u, a) (this is the so called *Hough-X transform* [Kollios et al. 1999; Papadopoulos et al. 2002]). Accordingly, the 1-d query $[(t_{1_q}, t_{2_q}), (y_{1_q}, y_{2_q})]$ becomes a polygon in the dual space.

By using a linear constraint query [Goldstein et al. 1997], the query in the dual Hough-X plane (Figure 6) is expressed as:

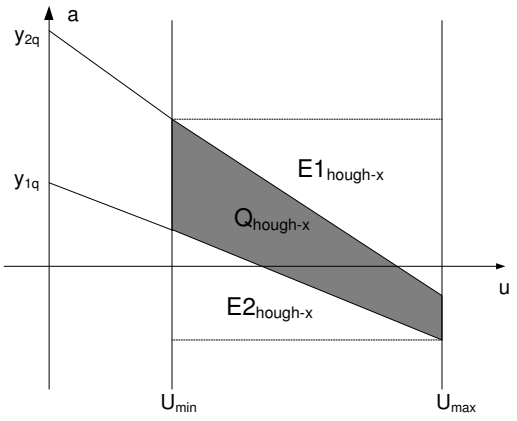

Fig. 6. Query in the Hough-X dual plane

If $u > 0$, then $Q_{Hough\text{-}X} = A_1 \cap A_2 \cap A_3 \cap A_4$, where A_i is defined as follows:

$A_1 = u \geq u_{min}$,
$A_2 = u \leq u_{max}$,
$A_3 = a \geq y_{1_q} - t_{2_q} u$,
$A_4 = a \leq y_{2_q} - t_{1_q} u$.

Inequalities of the A_1 and A_2 areas are obvious. The inequalities for A_3 and A_4 can be derived as follows: $\forall t \in [t_{1_q}, t_{2_q}] \Rightarrow y_{1_q} \leq y \leq y_{2_q} \Rightarrow y_{1_q} - t_{2_q}u \leq y_{1_q} - tu \leq a \leq y_{2_q} - tu \leq y_{2_q} - t_{1_q}u$, since $t_{1_q} \leq t \leq t_{2_q}$.

If $u < 0$, then $Q_{Hough\text{-}X} = B_1 \cap B_2 \cap B_3 \cap B_4$, where B_i is defined as follows:

$B_1 = u \leq -u_{min}$,
$B_2 = u \geq -u_{max}$,
$B_3 = a \geq y_{1_q} - t_{1_q}u$,
$B_4 = a \leq y_{2_q} - t_{2_q}u$.

Inequalities of the B_1 and B_2 areas are obvious. For B_3 and B_4 we are working in the same way as in the case of A_3 and A_4.

$\forall t \in [t_{1_q}, t_{2_q}] \Rightarrow y_{1_q} \leq y \leq y_{2_q} \Rightarrow y_{1_q} - t_{1_q}u \leq y_{1_q} - tu \leq a \leq y_{2_q} - tu \leq y_{2_q} - t_{2_q}u$, since $0 \leq t_{1_q} \leq t \leq t_{2_q}$ and $u < 0$.

In Figure 6 the line $a = y_{1_q} - t_{1_q}u$ for $u = u_{max}$ becomes $a = y_{1_q} - t_{1_q}u_{max}$ and the line $a = y_{2_q} - t_{2_q}u$ for $u = u_{min}$ becomes $a = y_{2_q} - t_{2_q}u_{min}$. Thus, for the Service Provider (SP), the initial query $[(t_{1_q}, t_{2_q}), (y_{1_q}, y_{2_q})]$ in the (t, y) plane is transformed to the rectangular query $[(u_{min}, u_{max}), (y_{1_q} - t_{1_q}u_{max}, y_{2_q} - t_{2_q}u_{min})]$ in the (u, a) plane.

In a similar way, for the upper ($y'(t) = ut + a + R$) and lower ($y''(t) = ut + a - R$) boundary trajectories, Service Provider (SP) gets the dual points $(u, a + R)$ and $(u, a - R)$ as well as the final (transformed) rectangular queries become $[(u_{min}, u_{max}), (y_{1_q} - R - t_{1_q}u_{max}, y_{2_q} - R - t_{2_q}u_{min})]$ and $[(u_{min}, u_{max}), (y_{1_q} + R - t_{1_q}u_{max}, y_{2_q} + R - t_{2_q}u_{min})]$ respectively in the (u, a) plane.

5.2 Hough-Y transform

By rewriting the equation $y = ut + a$ as $t = \frac{1}{u}y - \frac{a}{u}$, we can arrive to a different dual representation (the so called *Hough-Y transform* in [Kollios et al. 1999; Papadopoulos et al. 2002]). The point in the dual plane has coordinates (b, n), where $b = -\frac{a}{u}$ and $n = \frac{1}{u}$. Coordinate b is the point where the line intersects the line $y = 0$ in the original space. By using this transform horizontal lines cannot be represented. Similarly, the Hough-X transform cannot represent vertical lines. Nevertheless, since in our setting lines have a minimum and maximum slope (velocity is bounded by $[u_{min}, u_{max}]$), both transforms are valid.

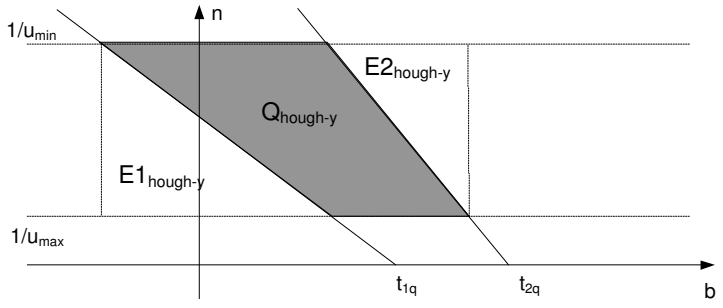

Fig. 7. Query on the Hough-Y dual plane

The query in the dual Hough-Y plane (Figure 7) is expressed as follows. If $u > 0$, then $Q_{Hough\text{-}Y} = C_1 \cap C_2 \cap C_3 \cap C_4$, where

$C_1 = n = \frac{1}{u} \geq \frac{1}{u_{max}}$,
$C_2 = n = \frac{1}{u} \leq \frac{1}{u_{min}}$,
$C_3 = n \geq -\frac{1}{y}b + \frac{t_{1_q}}{y}$,
$C_4 = n \leq \frac{1}{y}b + \frac{t_{2_q}}{y}$.

Inequalities of the C_1 and C_2 areas are obvious. The inequalities for C_3 and C_4 can be derived as follows: $\forall t \in [t_{1_q}, t_{2_q}] \Rightarrow n = -\frac{1}{y}b + \frac{t}{y} \leq -\frac{1}{y}b + \frac{t_{1_q}}{y}$ and $n = -\frac{1}{y}b + \frac{t}{y} \leq -\frac{1}{y}b + \frac{t_{2_q}}{y}$. Hence, the two lines in Figure 7 have negative slope and for $b = 0$ intersect the axis b in t_{1_q} and t_{2_q}, respectively. The intersection of the four regions C_1, C_2, C_3 and C_4 form the shaded polygon query of Figure 7.

If $u < 0$, then $Q_{Hough\text{-}Y} = D_1 \cap D_2 \cap D_3 \cap D_4$, where

$D_1 = n = \frac{1}{u} \leq -\frac{1}{u_{max}}$,
$D_2 = n = \frac{1}{u} \geq -\frac{1}{u_{min}}$,
$D_3 = n \geq -\frac{1}{y}b + \frac{t_{1_q}}{y}$,
$D_4 = n \leq -\frac{1}{y}b + \frac{t_{2_q}}{y}$.

The line $n = -\frac{1}{y}b + \frac{t_{1_q}}{y}$ for $n = \frac{1}{u_{min}}$ implies that

$$b = t_{1_q} - \frac{y}{u_{min}}. \tag{1}$$

In the same way the line $n = -\frac{1}{y}b + \frac{t_{2_q}}{y}$ for $n = \frac{1}{u_{max}}$ implies that

$$b = t_{2_q} - \frac{y}{u_{max}}. \tag{2}$$

However, according to the initial query and equation (1), we have

$$y_{1_q} \leq y \leq y_{2_q} \Leftrightarrow t_{1_q} - \frac{y_{2_q}}{u_{min}} \leq b \leq t_{1_q} - \frac{y_{1_q}}{u_{min}}. \tag{3}$$

Analogously, according to the initial query and equation (2), we have

$$y_{1_q} \leq y \leq y_{2_q} \Leftrightarrow t_{2_q} - \frac{y_{2_q}}{u_{max}} \leq b \leq t_{2_q} - \frac{y_{1_q}}{u_{max}}. \tag{4}$$

According to (3) and (4) the initial query $\left[\left(t_{1_q}, t_{2_q}\right), \left(y_{1_q}, y_{2_q}\right)\right]$ in (t, y) plane can be transformed to the following rectangular query in the (b, n) plane: $\left[\left(t_{1_q} - \frac{y_{2_q}}{u_{min}}, t_{2_q} - \frac{y_{1_q}}{u_{max}}\right), \left(\frac{1}{u_{max}}, \frac{1}{u_{min}}\right)\right]$. In a similar way for the upper ($y'(t) = ut + a + R$) and lower ($y''(t) = ut + a - R$) boundary trajectories, Service Provider (SP) gets the transformed rectangular queries $[(t_{1_q} - \frac{y_{2_q} - R}{u_{min}}, t_{2_q} - \frac{y_{1_q} - R}{u_{max}}), (\frac{1}{u_{max}}, \frac{1}{u_{min}})]$ and $[(t_{1_q} - \frac{y_{2_q} + R}{u_{min}}, t_{2_q} - \frac{y_{1_q} + R}{u_{max}}), (\frac{1}{u_{max}}, \frac{1}{u_{min}})]$ respectively in the (b, n) plane.

Algorithm 3 SP-Index-Building

1: BEGIN_PSEUDOCODE
2: Decompose the 2-d motion into two 1-d motions on the (t, x) and (t, y) planes;
3: For each projection, build the corresponding index structure;
4: END_PSEUDOCODE

Algorithm 4 SP-Mobile-User-Partitioning

1: BEGIN_PSEUDOCODE
2: Users with small velocity are stored using the Hough-X dual transform;
3: The rest are stored using the Hough-Y dual transform;
4: Motion information about the other projection is also included;
5: END_PSEUDOCODE

Algorithm 5 SP-Privacy-Aware-RQ Query

1: BEGIN_PSEUDOCODE
2: SP decomposes the query into two 1-d queries, for the (t, x) and (t, y) projection;
3: For each projection, SP gets the dual-simplex query;
4: For each projection, SP calculates a specific criterion c (for details see [Kollios et al. 1999; Papadopoulos et al. 2002]) and chooses the one (say p) that minimizes it;
5: For all dual-points of H_xS or H_yS sets, SP searches in projection p the simplex query of the Hough-X or the MBR of the simplex query of the Hough-Y partition. In the latter case, it performs a refinement or filtering step "on the fly", by using the whole motion information;
6: if the dual-points of both upper and lower boundary trajectories $((u_i, a_i + R), (u_i, a_i - R)$ or $b'_i, b''_i)$ lie inside the dual-simplex spatial area then SP detects that the same holds for the dual-point $((u_i, a_i)$ or $b_i)$ of the original trajectory;
7: else if the dual-points of both upper and lower boundary trajectories lie outside the dual-simplex spatial area then SP detects that the same holds for the dual-point of the original trajectory;
8: else host=send_RQ(Trusted-Server, Algorithm6);
9: END_PSEUDOCODE

5.3 The proposed algorithm for privacy-aware indexing

Let $S = \{y_1, y_2, \ldots, y_n\}$ be the initial set of original trajectory equations, and $S' = \{y'_1, y''_1, y'_2, y''_2 \ldots, y'_n, y''_n\}$ the set of boundary trajectory equations defined by the buffer.
Then, let $H_xS = \{(u_1, a_1 + R), (u_1, a_1 - R), \ldots, (u_n, a_i + R), (u_n, a_i - R)\}$ and $H_yS = \{b'_1, b''_1, \ldots, b'_n, b''_n\}$ be the set of dual Hough-X and Hough-Y transforms respectively.

Algorithm 3 runs in Service Provider (SP) and depicts the procedure for building the index. Algorithm 4 runs also in Service Provider (SP) and presents the procedure for Partitioning the Mobile Users according to their velocity. Algorithm 5 is running in Service Provider (SP) and at the end calls Algorithm 6, which is running in Trusted Server (TS) in order to perform the final filtering step. In particular, Algorithms 5 and 6 outline the privacy-aware algorithm for answering the exact 2-d

Algorithm 6 TS-Privacy-Aware-RQ Query

1: BEGIN_PSEUDOCODE
2: TS having knowledge of value R, it searches the simplex query of the Hough-X or Hough-Y partition for the dual-point ((u_i, a_i) or b_i) of the original trajectory;
3: END_PSEUDOCODE

RQ query:

Algorithm 7 SP-Privacy-Aware Indexing of $Q_{Hough-Y}$ partition with Lazy B-tree

1: BEGIN_PSEUDOCODE
2: Decompose the query into two 1-d queries, for the (t, x) and (t, y) projection;
3: For each projection get the dual-simplex query;
4: Search the MBR of the simplex query of the Hough-Y partition and perform a filtering step "on the fly", by using the whole motion information;
5: Let $B \subset H_y S$ be the answer set of dual parameters, which satisfy the query above;
6: $Result = 0$;
7: For all elements of B do
8: Begin_for
9: if ($b_i' \in B$ AND $b_i'' \in B$) then
$Result = Result \cup (idofuser_i, neighbours(idofuser_i, k-1)$;
10: return Result;
11: else if ($b_i' \notin B$ AND $b_i'' \notin B$) then return Result;
12: else host=send_RQ(Trusted-Server, Algorithm8);
13: End_for
14: END_PSEUDOCODE

Algorithm 8 TS-Privacy-Aware Indexing of $Q_{Hough-Y}$ partition with Lazy B-tree

1: BEGIN_PSEUDOCODE
2: if $b_i \in MBR$ of Hough-Y simplex partition then
3: begin
4: $Result = Result \cup (idofuser_i, neighbours(idofuser_i, k-1)$;return Result;
5: else return Result;
6: end
7: END_PSEUDOCODE

In [Kollios et al. 1999; Papadopoulos et al. 2002], $Q_{Hough-X}$ is computed by querying a 2-d partition tree, whereas $Q_{Hough-Y}$ is computed by querying a B^+-tree that indexes the b parameters. Here, we consider the case, where the users are moving with non small velocities u, meaning that the velocity value distribution is skewed (Zipf) towards u_{min} in some range $[u_{min}, u_{max}]$ and as a consequence the $Q_{Hough-Y}$ transformation is used (denote that u_{min} is a positive lower threshold).

Moreover, our method will incorporate the Lazy B-tree [Kaporis et al. 2005] indexing scheme[1], since the latter can handle update queries in optimal (constant) number of block-transfers (I/Os). As a result, we get Algorithm 7 and 8. Algorithm 7 is running in SP machine and it calls Algorithm 8, which running in TS machine, because the latter has the appropriate knowledge of the original trajectory parameter b_i in order to execute the final filtering step.

Algorithm 9 SP-Robust-Privacy-Aware Indexing

1: BEGIN_PSEUDOCODE
2: Decompose the query into two 1-d queries, for the (t, x) and (t, y) projection;
3: For each projection get the dual-simplex query;
4: Search the MBR of the simplex query of the Hough-Y partition and perform a filtering step "on the fly", by using the whole motion information;
5: Let $B \subset H_y S$ be the answer set of dual parameters, which satisfy the query above;
6: $Result = 0$;
7: For all elements of B do
8: Begin_for
9: if ($b_i' \in B$ AND $b_i'' \in B$) then
10: Voronoi-Clustering of (u_i, θ_i) points, where $1 \leq i \leq n$;
11: $Result = Result \cup (idofuser_i, neighbours(idofuser_i, k-1)$;
12: return Result;
13: else if ($b_i' \notin B$ AND $b_i'' \notin B$) then return Result;
14: else host=send_RQ(Trusted-Server, Algorithm10);
15: End_for
16: END_PSEUDOCODE

Algorithm 10 TS-Robust-Privacy-Aware Indexing

1: BEGIN_PSEUDOCODE
2: if $b_i \in MBR$ of Hough-Y simplex partition then
3: begin
4: Voronoi-Clustering of (u_i, θ_i) points, where $1 \leq i \leq n$;
5: $Result = Result \cup (idofuser_i, neighbours(idofuser_i, k-1)$;return Result;
6: else return Result;
7: end
8: END_PSEUDOCODE

[1]Source code of Lazy B-tree index is free available at the following URL: http://www.ionio.gr/∼sioutas/New-Software.htm

5.4 Reinforcing the privacy model of our method using voronoi-clustering of mobile users with similar motion-pattern

In statements 9 and 4 of algorithm 7 and 8 respectively, we call the routine neighbours($idofuser_i, k-1$) routine, in order to find the $k-1$ nearest neighbours that will anonymize the original mobile user i. Let a malicious attacker who observes that during the time, many of the neighbours ids change except for a small number of users. In this case, the desired anonimity is not achieved and the whole system becomes vulnerable to attackers. For this reason and in order to reinforce the whole privacy model, we initially call the $Voronoi-Clustering$ routine in order to cluster the mobile users according to their motion pattern (see Figure 8) and then we call the $neighbours(idofuser_i, k-1)$ routine that chooses neigbours who belong to the same cluster with the user i. In this case the neighbours' ids do not change with high frequency and the whole privacy model becomes robust.

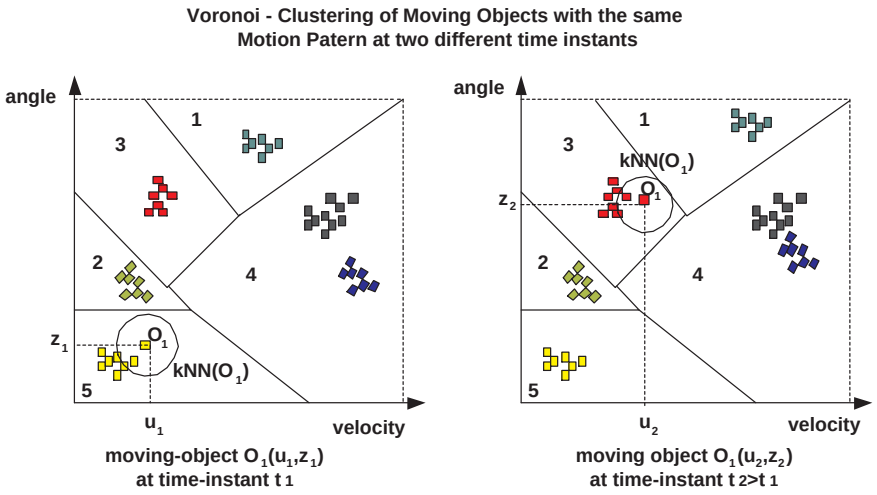

Fig. 8. Voronoi Clustering and neighbours of the same cluster

We define as motion-pattern of mobile user i at time instant t_i the point (u_i, θ_i) that represents both velocity value u_i and motion direction θ_i. Note here that we cannot avoid the 2-D clustering method mentioned before, by using clustering of 1-D elements like the parameters b_i or the projections of velocity vector ($u_{ix} = u_i cos\theta$) at X- or ($u_{ix} = u_i sin\theta$) Y-axis. Despite all these parameters include information about motion direction, these are not sufficient at all. Just think the case where mobile objects are moving at opposite directions, but also belong to the same cluster, since f.e. the cosine of opposite angles is exactly the same. As a result, we get Algorithm 9 and 10. In a similar way, Algorithm 9 is running in SP machine as well as Algorithm 10 executes the final filtering step in TS machine.

5.4.1 *Distortion and Competitive Ratio.* Let K be the number of b_i parameters associated to boundary trajectories of buffers that intersect with the query rectangle. Then, algorithms 9 and 10 require totally $T(n) = O(Cost(LazyB\_tree) + K)$

I/Os or block transfers. Moreover, and according to notations presented in [Abul et al. 2008], let say D be the initial Database that stores the N original trajectories and D' the Privacy-Aware Database that stores the $2N$ boundary-trajectories. Let also say, $Q(D)$ and $Q(D')$ be the Query Results obtained consuming $T(D)$ and $T(D')$ block-transfers (I/Os) in D and D' database schemes respectively. We define *Distortion ratio* to be $Distortion\_Ratio = \frac{|Q(D)-Q(D')|}{max(Q(D),Q(D'))}$ and *Competitive ratio* to be $Competitive\_Ratio = \frac{|T(D)-T(D')|}{max(T(D),T(D'))}$. In all cases, $T(D') > \ldots > T(D)$, thus it is very important to find out, how competitive to the optimal one ($T(D)$) is the privacy-aware method that answers the query in D'. Since, the distortion effect in D' absolutely depends on parameter K, an experimental evaluation of ***Competitive_Ratio vs*** K is also presented in the following section.

6. EXPERIMENTAL EVALUATION

This section compares the query performance of our privacy-aware method, when incorporates STRIPES [Patel et al. 2004] (the best known solution), Lazy B-trees (LBTs) and TPR*-tree, respectively. We deploy spatio-temporal data that contain insertions at a single timestamp 0. In particular, objects' MBRs are taken from the LA spatial dataset (128971 MBRs) [2]. We want to simulate a situation where all objects move in a space with dimensions 100x100 kilometers. For this purpose each axis of the space is normalized to [0,100000]. For the TPR*-tree, each object is associated with a VBR (Velocity Bounded Rectangle) such that (a) the object does not change spatial extents during its movement, (b) the velocity value distribution is skewed (Zipf) towards 30 in range [30,50], and (c) the velocity can be either positive or negative with equal probability. As in [Beckmann et al. 1990], we will use a small page size so that the number of index nodes simulates realistic situations.

Thus, for all experiments, the page size is 1 Kbyte, the key length is 8 bytes, whereas the pointer length is 4 bytes. Thus, the maximum number of entries ($< x >$ or $< y >$, respectively) in Lazy B-tree is 1024/(8+4)=85. In the same way, the maximum number of entries (2-d rectangles or $< x1, y1, x2, y2 >$ tuples) in TPR*-tree is 1024/(4*8+4)=27. On the other hand, the STRIPES index maps predicted positions to points in a dual transformed space and indexes this space using a disjoint regular partitioning of space. Each of the two dual planes, are equally partitioned into four quads. This partitioning results in a total of $4^2 = 16$ partitions, which we call *grids*. The fanout of each non-leaf node is thus 16. For each dataset, all indexes except for STRIPES have similar sizes. Specifically, for LA, each tree has 4 levels and around 6700 leaves apart from STRIPES index which has a maximum height of seven and consumes about 2.4 times larger disk space. Each query q has three parameters: $q_R len$, $q_V len$, and $q_T len$, such that (a) its MBR q_R is a square, with length $q_R len$, uniformly generated in the data space, (b) its VBR is $q_V = -q_V len/2, q_V len/2, -q_V len/2, q_V len/2$, and (c) its query interval is $q_T = [0, q_T len]$. The query cost is measured as the average number of node accesses in executing a workload of 200 queries with the same parameters. Implementations were carried out in C++ including particular libraries from SECONDARY LEDA

[2]Downloaded from the Tiger website `http://www.census.gov/geo/www/tiger/`.

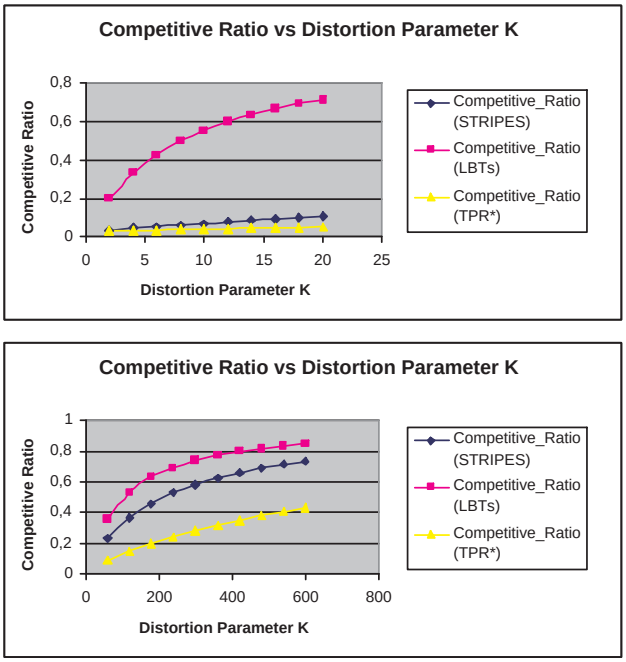

Fig. 9. $q_V len = 5$, $q_T len = 50$, $q_R len = 100$ (top), $q_R len = 1000$ (bottom), $R_{max} = 50$ (top), $R_{max} = 200$ (bottom)

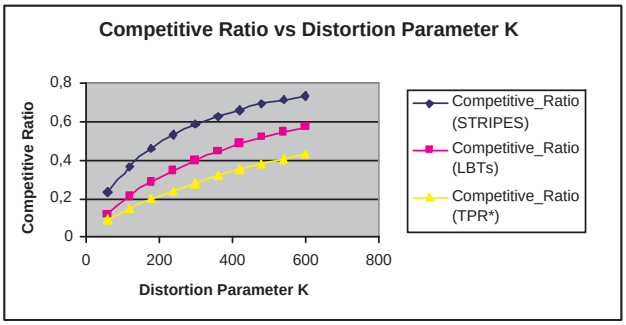

Fig. 10. $q_R len = 2000$, $q_V len = 5$, $q_T len = 50$, $R_{max} = 500$

v4.1.

6.1 Query cost comparison

We measure the Competitive Ratio of LBTs method (this method incorporates two Lazy B-trees that index the appropriate b parameters in each projection respectively, and finally combines the two answers by detecting and filtering all the pair permutations), the TPR*-tree and STRIPES presented in [Tao et al. 2003] and [Patel et al. 2004] respectively, using the same query workload, after every 10000

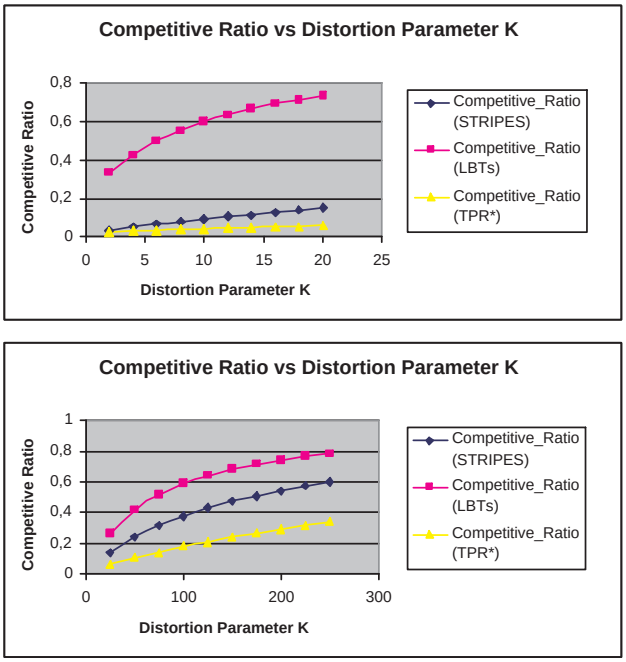

Fig. 11. $q_V len = 10$, $q_T len = 50$, $q_R len = 400$ (top), $q_R len = 1000$ (bottom), $R_{max} = 100$ (top), $R_{max} = 200$ (bottom)

updates. Figures 9 up to 13 show the Competitive Ratio as a function of K (for datasets generated from LA as described above), using workloads with different parameters. Parameter K represents boundary trajectories of buffers that intersect with the query rectangle, and obviously require an additional filtering on the fly process. Obviously, the required number of block transfers depends on the answer's size as well as the size of K.

Figure 9 depicts how competitive to the optimal solution the LBTs method is, in comparison to TPR*-tree and STRIPES. The Ratio degrades as the query rectangle length grows from 100 to 1000. When the query rectangle length or equivalently the query surface becomes extremely large (e.g. 2000), then the STRIPES index becomes more competitive (see Figure 10).

Figure 11 depicts how competitive to the optimal solution the LBTs method is, towards to TPR*-tree and STRIPES, in case the velocity vector grows. The Ratio degrades as the query rectangle length grows from 400 to 1000.

Figure 12 depicts the performance of all methods in case the time interval length degrades to value 1. Even in this case, the LBTs method is more competitive than STRIPES and TPR*-tree. As the query rectangle length grows from 400 to 1000, the LBTs method advantage decreases; we remark that STRIPES becomes faster, whereas LBTs method has exactly the same performance with the TPR*-trees.

Figure 13 depicts the efficiency of LBTs solution in comparison to that of TPR*-trees and STRIPES respectively in case the time interval length enlarges to 100.

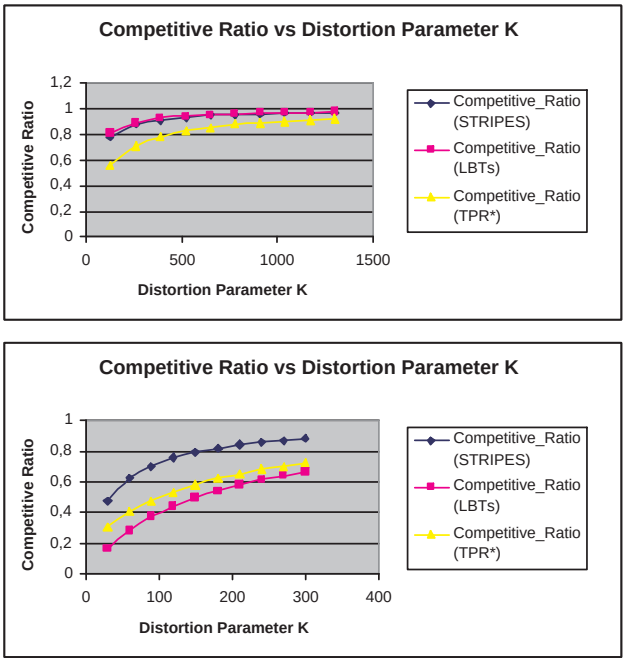

Fig. 12. $q_V len = 5$, $q_T len = 1$, $q_R len = 400$ (top), $q_R len = 1000$ (bottom), $R_{max} = 100$ (top), $R_{max} = 200$ (bottom)

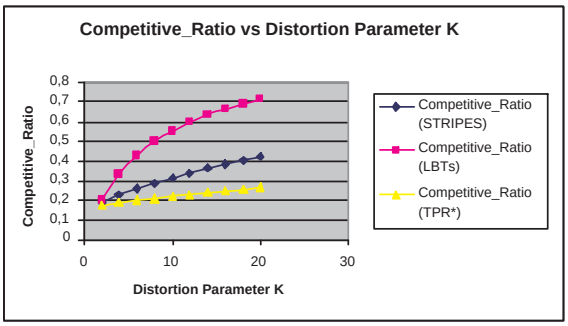

Fig. 13. $q_R len = 400$, $q_V len = 5$, $q_T len = 100$, $R_{max} = 100$

7. CONCLUSIONS

We presented the problem of anonymity preserving data publishing in moving objects databases. In particular, we studied the case where the trajectory of a mobile user on the plane is no longer a polyline in a two-dimensional space, instead it is a two-dimensional surface, which covers the real location of the mobile user on the XY-plane, in the anonymized (kNN) request. We reinforced the privacy model by clustering the mobile users according to their motion patterns in (u, θ) plane, where u and θ define the velocity measure and the motion direction (angle) respectively.

In this case the anonymized (kNN) request lookups neighbours, who belong to the same cluster with the mobile requester in (u, θ) space. By transforming the surface's boundary poly-lines to dual points we experimentally focused on the impact of information distortion introduced by this space translation.

REFERENCES

Abul, O., Bonchi, F., and Nanni, M. 2008. Never walk alone: Uncertainty for anonymity in moving objects databases. In *In ICDE*. 376–385.

Agarwal, P. K., Arge, L., Erickson, J., Franciosa, P. G., and Vitter, J. S. 1997. Efficient searching with linear constraints. In *PROC. 17TH ANNU. ACM SYMPOS. PRINCIPLES DATABASE SYST*. 169–178.

Aggarwal, A., Guibas, L., Saxe, J., and Shor, P. 1989. A linear-time algorithm for computing the voronoi diagram of a convex polygon. *Discrete & Computational Geometry 4*, 591–604.

Ardagna, C., Cremonini, M., De Capitani di Vimercati, S., and Samarati, P. 2009. Access control in location-based services. In *Privacy in Location-Based Applications*, C. Bettini, S. Jajodia, P. Samarati, and X. Wang, Eds. Lecture Notes in Computer Science, vol. 5599. Springer Berlin / Heidelberg, 106–126.

Aurenhammer, F., Graz, T. U., Klein, R., Hagen, F., and Vi, P. I. Voronoi diagrams. In *Handbook of Computational Geometry*. Elsevier Science Publishers B.V. North-Holland, 201–290.

Bajaj, R., Ranaweera, S., and Agrawal, D. P. Gps: Location-tracking technology. *IEEE Computer*.

Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. 1990. The r*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*. SIGMOD '90. 322–331.

Clarke, R. 2001. Person location person tracking - technologies risks and policy implications. *Information Technology and People 14,* 2, 206–231.

D'Roza, T. and Bilchev, G. 2003. An overview of location-based services. *BT Technology Journal 21*, 20–27.

Duckham, M. and Kulik, L. 2006. *Location privacy and location-aware computing*. CRC Press, Chapter Investigating Change in Space and Time.

Dunham, M. H. 2007. *Data Mining: Introductory and Advanced Topics*. Pearson Education.

Gaede, V. and Günther, O. 1998. Multidimensional access methods. *ACM Comput. Surv. 30*, 170–231.

Ghinita, G. 2009. Private queries and trajectory anonymization: a dual perspective on location privacy. *Trans. Data Privacy 2*, 3–19.

Gkoulalas-Divanis, A., Verykios, V. S., and Bozanis, P. 2009. A network aware privacy model for online requests in trajectory data. *Data Knowl. Eng. 68*, 431–452.

Goldstein, J., Ramakrishnan, R., Shaft, U., and Yu, J.-B. 1997. Processing queries by linear constraints. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. PODS '97. 257–267.

Gruteser, M. and Grunwald, D. 2003. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*. MobiSys '03. 31–42.

Gruteser, M. and Liu, X. Protecting privacy in continuous location-tracking applications. *IEEE Security and Privacy 2*, 28–34.

Hamerly, G., E. C. 2004. *Learning the k in k-means*. Vol. 16. MIT Press, Chapter Advances in Neural Information Processing Systems, 281288.

Han, J. and Kamber, M. 2007. *Data mining concepts and techniques*. Morgan Kaufmann Publishers.

Hauser, C. and Kabatnik, M. 2001. Towards privacy support in a global location service. In *In Proc. of the IFIP Workshop on IP and ATM Traffic Management (WATM/EUNICE 2001*. 81–89.

Hoh, B., Gruteser, M., Xiong, H., and Alrabady, A. 2007. Preserving privacy in gps traces via uncertainty-aware path cloaking. In *Proceedings of the 14th ACM conference on Computer and communications security*. CCS '07. 161–171.

Jensen, C. S., Lin, D., and Ooi, B. C. 2004. Query and update efficient b+-tree based indexing of moving objects. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*. VLDB '04. 768–779.

Kao, B., Lee, S. D., Lee, F. K., lok Cheung, D. W., and Ho, W.-S. 2010. Clustering uncertain data using voronoi diagrams and r-tree index. *IEEE Transactions on Knowledge and Data Engineering 22*, 1219–1233.

Kaporis, A., Makris, C., Mavritsakis, G., Sioutas, S., Tsakalidis, A., Tsichlas, K., and Zaroliagis, C. 2005. Isb-tree: A new indexing scheme with efficient expected behaviour. In *Algorithms and Computation*, X. Deng and D.-Z. Du, Eds. Lecture Notes in Computer Science, vol. 3827. Springer Berlin / Heidelberg, 318–327.

Koivistoinen, H., Ruuska, M., and Elomaa, T. 2006. A voronoi diagram approach to autonomous clustering. In *Discovery Science*, L. Todorovski, N. Lavrac, and K. Jantke, Eds. Lecture Notes in Computer Science, vol. 4265. Springer Berlin / Heidelberg, 149–160.

Kollios, G., Gunopulos, D., and Tsotras, V. J. 1999. On indexing mobile objects. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. PODS '99. 261–272.

Kulik, L. 2009. Privacy for real-time location-based services. *SIGSPATIAL Special 1*, 9–14.

MacQueen, J. 1965. On convergence of k-means and partitions with minimum average variance (abstract). *Annals of Mathematical Statistics 36*.

Manolopoulos, Y., Theodoridis, Y., and Tsotras, V. J. 2000. *Advanced database indexing*. Kluwer Academic Publishers, Norwell, MA, USA.

Papadopoulos, D., Kollios, G., Gunopulos, D., and Tsotras, V. J. 2002. Indexing mobile objects on the plane. In *Proceedings of the 13th International Workshop on Database and Expert Systems Applications*. 693–697.

Patel, J. M., Chen, Y., and Chakka, V. P. 2004. Stripes: an efficient index for predicted trajectories. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. SIGMOD '04. 635–646.

Pelleg, D. and Moore, A. 2000. X-means: Extending k-means with efficient estimation of the number of clusters. In *In Proceedings of the 17th International Conf. on Machine Learning*. Morgan Kaufmann, 727–734.

P.S., B. and V., B. 2009. Ctvn: Clustering technique using voronoi diagram. *International Journal of Recent Trends in Engineering 2,* 3, 13–15.

Saltenis, S. and Jensen, C. S. 2002. Indexing of moving objects for location-based services. In *Proceedings of the 18th International Conference on Data Engineering*. ICDE '02. 463–472.

Saltenis, S., Jensen, C. S., Leutenegger, S. T., and Lopez, M. A. 2000. Indexing the positions of continuously moving objects. In *ACM International Conference on Management of Data*. 331–342.

Salzberg, B. and Tsotras, V. J. 1999. Comparison of access methods for time-evolving data. *ACM Comput. Surv. 31*, 158–221.

Samarati, P. 2001. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng. 13*, 1010–1027.

Sioutas, S., Tsakalidis, K., Tsichlas, K., Makris, C., and Manolopoulos, Y. 2008. A new approach on indexing mobile objects on the plane. *Data Knowl. Eng. 67*, 362–380.

Sweeney, L. 2002. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10*, 557–570.

Tao, Y., Papadias, D., and Sun, J. 2003. The tpr*-tree: an optimized spatio-temporal access method for predictive queries. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*. VLDB '2003. 790–801.