Embedding advanced geometric techniques into SQL for efficient indexing of mobile objects

S. SIOUTAS
Computer Engineering and Informatics department
University of Patras
Building B, University Campus, 26500, Rion, Patras
GREECE

L. DROSSOS
Technological Institute of Messolongi,
Department of Applied Informatics in Administration and Economics
Technological Institute Campus, 30200, Messolongi
GREECE

D.TSOLIS
Computer Engineering and Informatics department
University of Patras
Building B, University Campus, 26500, Rion, Patras
GREECE

T. S. PAPATHEODOROU
Computer Engineering and Informatics department
University of Patras
Building B, University Campus, 26500, Rion, Patras
GREECE

Abstract - It is of great importance a trial to embed new geometric techniques into SQL in order to achieve more efficient indexing of objects moving on the plane and answer range queries about their future positions. This problem is motivated by real-life applications, such as allocating more bandwidth for areas where high concentration of mobile phones is imminent, or predicting future congestion areas in a highway Geographic Information System (GIS). We consider the problem in the external memory model of computation and present a variety of dynamic techniques.

Key – Words: -Spatio-Temporal Databases, Mobile Objects, Indexing, Computational Geometry, SQL

1 Introduction

Location-aware applications, such as traffic monitoring in GIS, intelligent navigation, and mobile communications management, cannot be efficiently supported by traditional database management systems. The assumption that data stored in the database remain constant, unless explicitly updated, supports a model where updates are issued in discrete steps. The aforementioned applications deal with continuously changing data, i.e. objects' location. Inevitably, a DBMS receiving requests of that kind at every unit of time would exhibit tremendous update overhead.

An attractive solution to tackle the problem is to use a function of time **g(t)** to abstract each object's location. Then the current location of a moving object at any time instant can be calculated. An update has to be issued only when the parameters of **g** change (e.g. speed or direction). Clearly, this approach minimizes the update overhead. However, it introduces new problems, such as the need for appropriate data structures, data models, query languages, query processing and optimization techniques.

Any data model in a large database requires efficient external storage support for its language features. Range searching and its variants are problems that often need to be solved efficiently. In RDBMS and SQL, one-dimensional dimensional range searching is a commonly used operation. Special cases of two-dimensional range searching are important for the support of new language features, such as constrain query languages. In spatial databases such as Geographic Information Systems (GIS), a large number of external data structures for answering such queries have been proposed. While most attention has been focused on *isothetic or orthogonal range searching,* in which a query is a d-dimensional axis-parallel hyperrectangle, the importance of nonisothetic queries has also been recognized. In the computational geometry community a nonisothetic range searching is called *half plane range searching,* where a query is a *linear constraint* of the form $x_d \leq a_0 + \sum_{i=1}^{d-1} a_i x_i$ and the target is to retrieve all points that satisfy the constraint.

The focus of this paper is on the problem of indexing mobile objects in two dimensions. We are interested in efficiently answer range queries over the objects location in the future. Here we present both approximate methods based on geometric duality transformation and worst-case methods based on the geometric techniques of [10]. In section 2 we give a formal problem description of moving object's indexing. In section 3 we present work related with the problem at hand. Next, section 4 describes the dual transform, which is the core of our approximate approach. The worst-case method is addressed in section 5. Section 6 closes the paper.

2 Problem descriptions

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 position of the object, provided that the characteristics of its motion 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 $P(t_0) = [x_0, y_0]$ be the initial position at time t_0. Then, the object starts moving and at time $t > t_0$ its position will be $P(t) = [x(t), y(t)] = [x_0 + u_x(t - t_0), y_0 + u_y(t - t_0)]$, where $U = [u_x, u_y]$ is its velocity vector. For example, in Figure 1, the lines depict the objects' trajectories on the (t,y) plane.

We would like to answer queries of the form: "Report the objects located inside the rectangle $[x_{1q}, x_{2q}] \times [y_{1q}, y_{2q}]$ at the time instants between $\mathbf{t_{1q}}$ and $\mathbf{t_{2q}}$ (where $t_{now} \leq t_{1q} \leq t_{2q}$), given the current motion information of all objects".

In general, the straightforward approach of representing an object moving on an 1-dimensional line is by plotting the trajectories as lines in the time-location (t,y) plane (same for **(t,x)** plane). The equation describing each line is y(t)=ut+a where **u** is the slope (velocity vector in this case) and a is the intercept (initial position vector in this case), which is computed using the motion information (Figure 1). Based on this setting, the query is expressed as the 2-dimensional interval $[(y_{1q},y_{2q}),(t_{1q},t_{2q})]$, and it reports the objects that correspond to the lines intersecting the query rectangle.

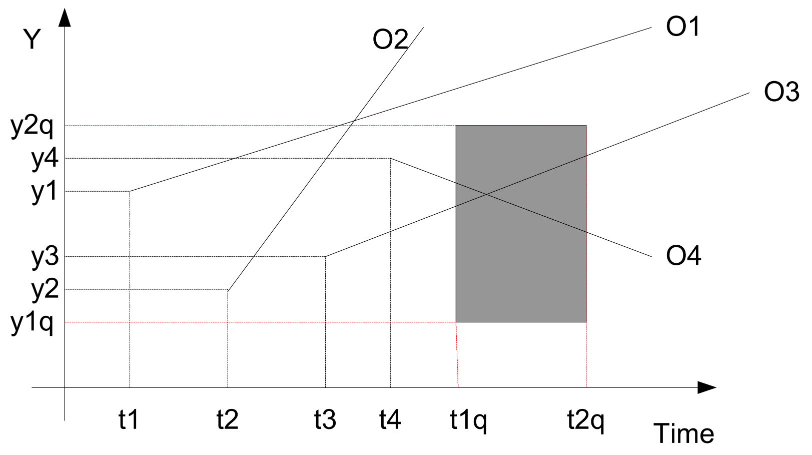

Fig. 1: Trajectories and query in (t,y) plane

Let the following spatio-temporal relation:

Mobile_objects (name: string, from :point, to :point, route: y(t)=ut+a)

Let a specific SQL language for moving objects, like that presented in [16], where new predicates like **trajectory** and **intersects** have been defined. Then the query can be expressed as follows:
Select name
From Mobile_Objects
Where ***trajectory*** *(route)* ***intersects***
rectangle$[t_{1q},t_{2q},y_{1q},y_{2q}]$.

3. Related work

The space-time approach provides an intuitive representation, but is also problematic, since the trajectories correspond to long lines. Traditional indexing techniques in this setting tend to show many drawbacks.

A common approach is to use a Spatial Access Method, such an R-tree [4] or an R*- tree [2]. In this setting each line is approximated by a minimum-bounding rectangle (MBR). Obviously, the MBR approximation has much larger area than the line itself. Furthermore, since the trajectory of an object is valid until an update is issued, it has a starting point but no end. Thus all trajectories expand till "infinity". Also, the Moving Objects Spatio-Temporal (MOST) model and a language (FTL) for querying the current and future locations of mobile objects are presented in [20, 24, 25]. In order to index line segments a method based upon the dual transform was proposed in [5]. The use of dual transformation to index mobile objects is also proposed in [11].

Saltenis et al. [7] presented a technique to efficiently index moving objects. They proposed the time-parameterized R-tree (TPR-tree), which extends the R*-tree. The coordinates of the bounding rectangles in the TPR-tree are functions of time and, intuitively, are capable of following the objects as they move. The position of a moving object is represented by its location at a particular time instant (reference position) and its velocity vector

In [6] a technique to index moving objects was introduced, based upon the dual transform, which we refer here.

4. Embedding duality geometric techniques into SQL for better approximate (average) performance.

The dual space-time representation: The dual transform, in general, maps a hyper-plane **h** from $\mathbf{R}^d$ to a point in $\mathbf{R}^d$ and vice-versa. In this section we briefly describe how we can address the problem at hand in a more intuitive way, by using the dual transform on the one-dimensional case.

Specifically, a line from the primal plane **(t,y)** is mapped to a point in the dual plane. A class of transforms with similar properties may be used for the mapping. The problem setting parameters determine which one is more useful.

4.1 Hough-X transform [5] and approximate rectangle query

One dual 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 of an object's trajectory (i.e. velocity) and the other axis its intercept. Thus we have the dual point **(u,a)** (this is called Hough-X transform in [5]). Accordingly, the 1-d query $[(y_{1q},y_{2q}),(t_{1q},t_{2q})]$ becomes a polygon in the dual space. By using a linear constraint query [3], the query in the dual Hough-X plane (Figure 2) is expressed in the following way [6]:

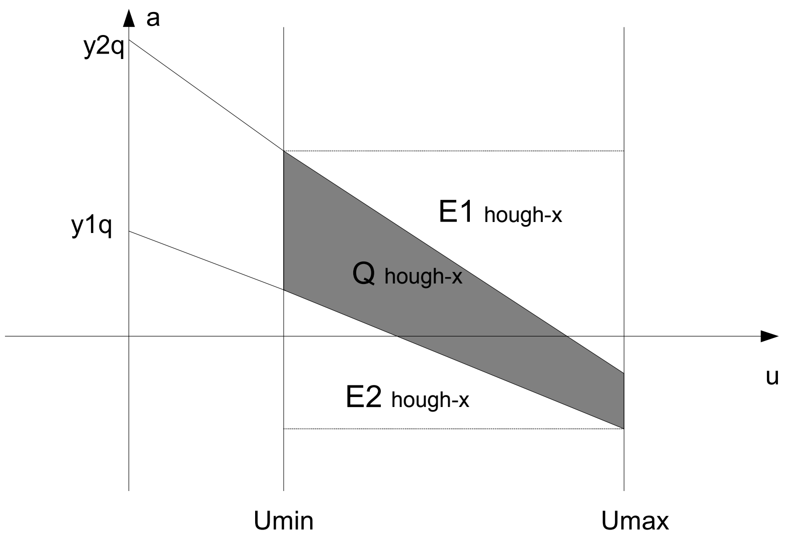

Fig. 2: Query on the Hough-X dual plane.

- If $u > 0$, then
  $Q_{Hough-X} = A_1 \cap A_2 \cap A_3 \cap A_4$, where:
  - $A_1 = u \geq u_{\min}, A_2 = u \leq u_{\max}$
  - $A_3 = a \geq y_{1q} - t_{2q}u, A_4 = a \leq y_{2q} - t_{1q}u$

The inequalities for A_1 and A_2 areas are obvious. The inequalities for A_3 and A_4 can be derived as follows:

$$\forall t \in [t_{1q},t_{2q}] \Rightarrow y_{1q} \leq y \leq y_{2q} \Rightarrow y_{1q} \leq a+tu \leq y_{2q} \Rightarrow y_{1q} - tu \leq a \leq y_{2q} - tu \Rightarrow$$

$$\Rightarrow y_{1q} - t_{2q}u \leq y_{1q} - tu \leq a \leq y_{2q} - tu \leq y_{2q} - t_{1q}u,$$

since $t_{1q} \leq t \leq t_{2q}$.

- If $u < 0$, then
  $Q_{Hough-X} = B_1 \cap B_2 \cap B_3 \cap B_4$, where:
  - $B_1 = u \leq -u_{\min}, B_2 = u \geq -u_{\max}$
  - $B_3 = a \geq y_{1q} - t_{1q}u, B_4 = a \leq y_{2q} - t_{2q}u$

The inequalities for B_1 and B_2 are obvious. For B_3 and B_4 we are working in the same way:

$$\forall t \in [t_{1q},t_{2q}] \Rightarrow y_{1q} \leq y \leq y_{2q} \Rightarrow y_{1q} \leq a+tu \leq y_{2q} \Rightarrow y_{1q} - tu \leq a \leq y_{2q} - tu \Rightarrow$$

$$\Rightarrow y_{1q} - t_{1q}u \leq y_{1q} - tu \leq a \leq y_{2q} - tu \leq y_{2q} - t_{2q}u,$$

since $0 \leq t_{1q} \leq t \leq t_{2q}$ but $u < 0$.

In figure 2 the line $a = y_{1q} - t_{1q}u$ for $u=u_{\max}$ becomes $a = y_{1q} - t_{1q}u_{\max}$ and the line $a = y_{2q} - t_{2q}u$ for $u=u_{\min}$ becomes $a = y_{2q} - t_{2q}u_{\min}$.

Thus the initial query $[(t_{1q},t_{2q}), [(y_{1q},y_{2q})]$ in **(t,y)** plane can be transformed to the following one approximate query in **(u.a)** plane: $[(u_{\min}, u_{\max}), (y_{1q} - t_{1q}u_{\max}, y_{2q} - t_{2q}u_{\min})]$.

So, instead of

> *Select name*
> *From Mobile_Objects*
> *Where* ***trajectory****(route)* ***intersects***
> *rectangle[t_{1q},t_{2q},y_{1q},y_{2q}]*

We write

> *Select name*
> *From Mobile_Objects*
> *Where* ***u_{min} <= u <=u_{max} and y_{1q}-$t_{1q}u_{max}$ <= y <= y_{2q}-$t_{2q}u_{min}$***

The second alternative approximate query avoids operations of huge overhead like **trajectory** or **intersects**.

4.2 Hough-Y transform [5] and approximate rectangle query

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 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 primal 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.

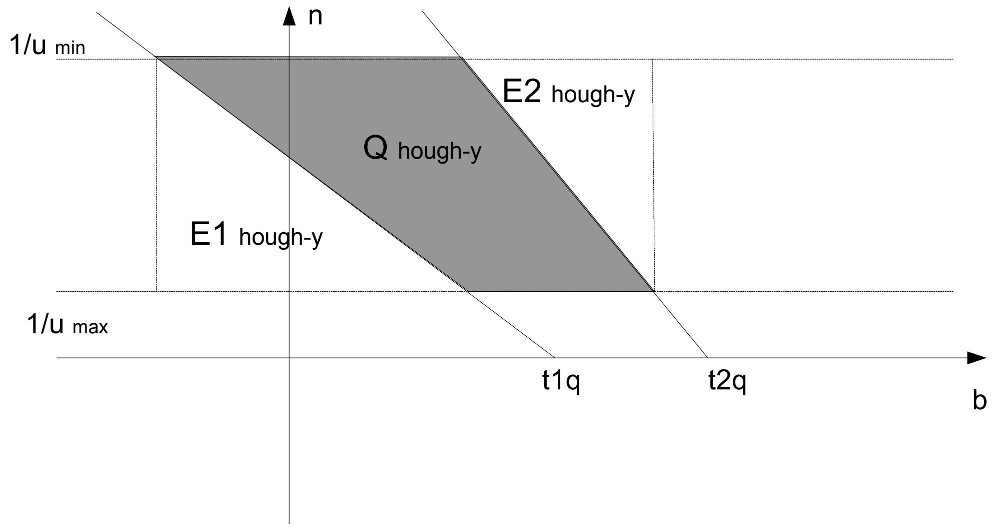

Fig. 3: Query on the Hough-Y dual plane.

The query in the dual Hough-Y plane (Figure 3) is expressed in the following way [6]:

- If $u > 0$, then
  $Q_{Hough-Y} = C_1 \cap C_2 \cap C_3 \cap C_4$, where:
  - $C_1 = n = \frac{1}{u} \geq \frac{1}{u_{\max}}, C_2 = n = \frac{1}{u} \leq \frac{1}{u_{\min}}$
  - $C_3 = n \geq -\frac{1}{y} b + \frac{t_{1q}}{y}, C_4 = n \leq -\frac{1}{y} b + \frac{t_{2q}}{y}$

The inequalities for C_1 and C_2 areas are obvious. The inequalities for C_3 and C_4 can be derived as follows:

$\forall t \in [t_{1q}, t_{2q}] \Rightarrow n = -\frac{1}{y} b + \frac{t}{y} \geq -\frac{1}{y} b + \frac{t_{1q}}{y}$ and $n = -\frac{1}{y} b + \frac{t}{y} \leq -\frac{1}{y} b + \frac{t_{2q}}{y}$. So, the two lines in figure 3 they have negative slope and for b=0 intersect the axis b in t_{1q} and t_{2q} respectively. The intersection of four regions C_1, C2, C3 and C_4 derives the polygon (shaded) query of figure 3.

For $u < 0$ the case is symmetric, that means: $Q_{Hough-Y} = D_1 \cap D_2 \cap D_3 \cap D_4$, where:

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

The line $n = -\frac{1}{y} b + \frac{t_{1q}}{y}$ for $n=1/u_{min}$ implies that: $\frac{1}{y} b = \frac{t_{1q}}{y} - n \Leftrightarrow b = t_{1q} - ny \Leftrightarrow$

$$\Leftrightarrow b = t_{1q} - \frac{y}{u_{\min}} \quad (1)$$

In the same way the line $n = -\frac{1}{y}b + \frac{t_{2q}}{y}$ for $n=1/u_{max}$ implies that: $b = t_{2q} - \frac{y}{u_{max}}$ (2). But according to initial query and equation (1) we have that: $y_{1q} \le y \le y_{2q} \Leftrightarrow -y_{1q} \ge -y \ge -y_{2q} \Leftrightarrow$

$$\Leftrightarrow -\frac{y_{1q}}{u_{min}} \ge -\frac{y}{u_{min}} \ge -\frac{y_{2q}}{u_{min}} \Leftrightarrow$$

$$t_{1q} - \frac{y_{1q}}{u_{min}} \ge t_{1q} - \frac{y}{u_{min}} \ge t_{1q} - \frac{y_{2q}}{u_{min}} \Leftrightarrow$$

$$\Leftrightarrow t_{1q} - \frac{y_{2q}}{u_{min}} \le b \le t_{1q} - \frac{y_{1q}}{u_{min}} \text{ (3).}$$

Proportionally according to initial query and equation (2) we have that: $y_{1q} \le y \le y_{2q} \Leftrightarrow -y_{1q} \ge -y \ge -y_{2q} \Leftrightarrow$

$$-\frac{y_{1q}}{u_{max}} \ge -\frac{y}{u_{max}} \ge -\frac{y_{2q}}{u_{max}} \Leftrightarrow$$

$$t_{2q} - \frac{y_{1q}}{u_{max}} \ge t_{2q} - \frac{y}{u_{max}} \ge t_{2q} - \frac{y_{2q}}{u_{max}} \Leftrightarrow$$

$$\Leftrightarrow t_{2q} - \frac{y_{2q}}{u_{max}} \le b \le t_{2q} - \frac{y_{1q}}{u_{max}} \text{ (4).}$$

According to (3) and (4) the initial query $[(t_{1q},t_{2q}), [(y_{1q},y_{2q})]$ in **(t,y)** plane can be transformed to the following one approximate query in **(b.n)** plane:

$$[(t_{1q} - \frac{y_{2q}}{u_{min}}, t_{2q} - \frac{y_{1q}}{u_{max}}), (\frac{1}{u_{max}}, \frac{1}{u_{min}})].$$

So, now instead of

Select name
From Mobile_Objects
*Where **trajectory**(route) **intersects***
rectangle[t_{1q},t_{2q},y_{1q},y_{2q}]

We write

Select name
From Mobile_Objects
*Where **1/u_{max} <= u <= 1/u_{min} and***
t_{1q}-y_{2q}/u_{min} <= t <= t_{2q}-y_{1q}/u_{max}

4.3 A new (u,t)-transform and approximate rectangle query

In this case we assume that objects move with constant velocity vector starting from a specific location at a specific time instant but the velocities are unbounded.

As you can see in figure 4, we represent each moving object by a dual point (u,t). For example the moving object O_k is associated to (u_k,t_k) point which means that the object O_k is moving with constant velocity vector u_k starting from a specific location at time instant t_k.

$$y_{1q} \le y \le y_{2q} \Leftrightarrow y_{1q} \le ut + a \le y_{2q} \Leftrightarrow \frac{y_{1q}-a}{t} \le u \le \frac{y_{2q}-a}{t} \Leftrightarrow$$

$$\frac{y_{1q}-a}{t_{2q}} \le \frac{y_{1q}-a}{t} \le u \le \frac{y_{2q}-a}{t} \le \frac{y_{2q}-a}{t_{1q}}, \text{ since}$$

$$0 \le t_{1q} \le t \le t_{2q}.$$

So, the initial query $[(t_{1q},t_{2q}), [(y_{1q},y_{2q})]$ in **(t,y)** plane can be transformed to the following one approximate query in **(u,t)** plane:

$$\left[(\frac{y_{1q}-a}{t_{2q}}, \frac{y_{2q}-a}{t_{1q}}), (t_{1q}, t_{2q}) \right].$$

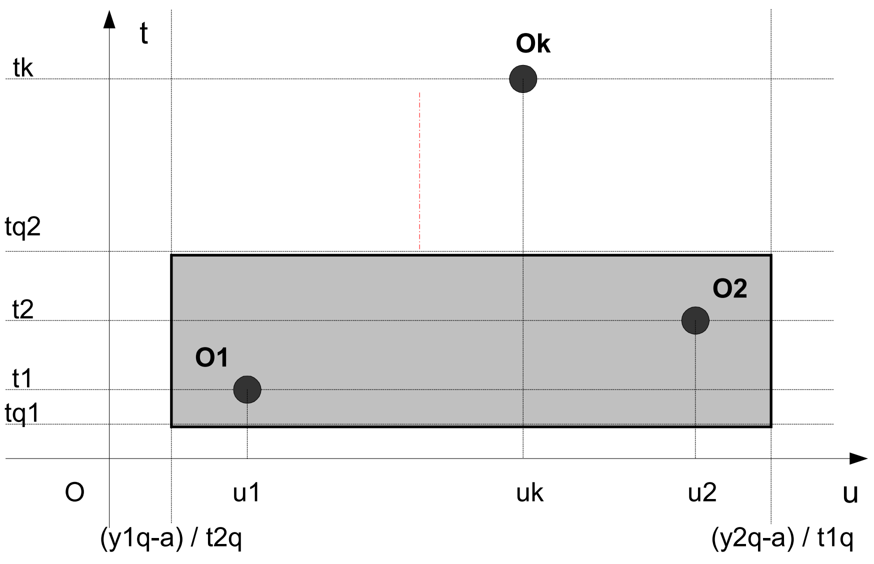

Fig. 4: (u,t) dual transform and approximate rectangle query

So, now instead of

Select name
From Mobile_Objects
Where ***trajectory****(route)* ***intersects***
rectangle[t_{1q},t_{2q},y_{1q},y_{2q}]

We write

Select name
From Mobile_Objects
Where t_{1q} **<= *t* <= t_{2q} *and***
$(y_{1q}-a)/t_{2q} <= u <= (y_{2q}-a)/t_{1q}$

4.4 Observations about Indexing in one dimension

Observation 1: Motions with small velocities in the Hough-Y approach are mapped into dual points (b,n) having large n coordinates (n=1/u). Thus, since few objects can have small velocities, by storing the Hough-Y dual points in an index structure such an R^*-tree, MBR's with large extents are introduced, and the index performance is severaly affected. On the other hand, by using a Hough-X for the small velocities' partition, we eliminate this effect, since the Hough-X dual transform maps an object's motion to the (u,a) dual point.

The query area in Hough-X plane is enlarged by the area E, which is easily computed as $E^{Hough-X} = (E_1$ hough-X + E_2 hough-X). Also, let the actual area of the simplex query be $Q^{Hough-X}$. Similarly, on the dual Hough-Y plane, let $Q^{Hough-Y}$ be the actual area of the query, and $E^{Hough-Y}$ be the enlargement. In order to answer the initial query we must choose the transform which minimizes the following criterio

$$c = \frac{E^{Hough-X}}{Q^{Hough-X}} + \frac{E^{Hough-Y}}{Q^{Hough-Y}}.$$

Observation 2: Since all 2-dimensional approximate queries in Hough-X plane have the same rectangle side (u_{min},u_{max}), the rectangle range search is equivalent to a **simple 1-dimensional range search** on the a coordinate axis. Thus each of the “a” indices can simply be a **simple B^+- tree.** When the objects' velocities are unbounded then it's obvious that the solution of (u,t) dual transform is needed. Similarly, in Hough-Y plane due to the fact that all 2-dimensional approximate queries have the same rectangle side ($1/u_{max}$, $1/u_{min}$), the rectangle range search is again equivalent to a simple 1- dimensional range search on the b coordinate axis. Thus each of the “b” indices can simply be again a **simple B^+- tree again.**

Observation3: When the objects' velocities are unbounded then it's obvious that the solution of (u,t) dual transform is needed. The 2-dimensional rectangle approximate query of figure 4 can be managed optimally by the external priority search tree [1].

4.5 Indexing mobile objects in two dimensions

The procedure for building the index follows:

1. Decompose the 2-d motion into two 1-d motions on the (t,x) and (t,y) planes.
2. For each projection, build the corresponding index structure.
   - Partition the objects according to their velocity:
     (a) Objects with small velocity are stored using the Hough-X dual transform, while the rest are stored using the Hough-Y dual transform.
     (b) Motion information about the other projection is also included

The outline of the algorithm for answering the exact 2-d query is presented next:

1. Decompose the query into two 1-d queries, for the (t,x) and (t,y) projection.
2. For each projection get the dual – simplex query
3. For each projection calculate the criterion c, according to the observation 1, and choose the one (say p) that minimizes it.
4. Search in projection p the Hough-X or Hough-Y partition.
5. Perform a refinement or filtering step "on the fly", by using the whole motion information. Thus, the result set contains only the objects that satisfy the query

5. Embedding half plane geometric techniques into SQL for better worst-case indexing performance.

Since the approximate range queries previously described require an overhead of I/O's in refinement step, it's worth a try to develop efficient indexing mechanisms that can answer directly the specified form of simplex (polygon) queries. As a result we avoid the expensive overhead of refinement step and now the performance of our index can be evaluated in worst-case.

Example 1: *Operations on single data types*

Let the relation *Companies (Name, Price, Earnings)* and the Query:*"Retrieve the names of all companies whose price/earnings ratio is less than 5"*. In SQL the query can be expressed as follows:
Select Name
From Companies
*Where (Price-5*Earnings<0).*

If we interpret each ordered pair *(Earning,Price)* as a point in the plane, the result of the query derives from all such points that satisfy the following linear constraint line: $y-5*x \le 0$.

Example 2: *Operations on spatio-temporal data types*

In one-dimensional space, operation **rangevalues** returns values assumed over time as a set of intervals. For the two-dimensional types, operations are offered to return the parts of the projections corresponding to our data types. For example, the projection of a moving point into the plane may consist of points and of lines; these can be obtained separately by operations **locations** and **trajectory**, respectively. Operation **length** gets the trajectory as parameter and returns the full length of the lines that constitute the trajectory of the mobile object.
Let the following spatio-temporal relation:

flight(airline:string, no:int, from:string, to:string, route:mpoint)

Attributes *airline* and *no* of the relation *flight* identify a flight. In addition, the relation records the names of the departure and destination cities and the route taken for each flight. The last attribute is of type *moving(point). We assume that a flight's route is defined only for the times the plane is in flight and not when it is on the ground.*

Query: "Give me the number (no) -pair of airlines LH and OL respectively that satisfy the following constraint: The plane of LH airlines routed double total distance from the respective one of OL airlines".

SQL Query:

```
flight: fl,f2
Select f1.no,f2.no
From f1,f2
where
        Length(trajectory(Select f1.route
        From f1
        Where f1.airline="LH" ))>2*
        Length(trajectory(Select f2.route
        From f2
        Where f2.airline="OL" ))
```

If we interpret each ordered pair *(length of OL,length of LH)* as a point in the plane, the result of the query derives from all such points that satisfy the linear constraint line *y-2*x>0.*
Several complex queries can be expressed as reporting all points lying within a given convex query region. Such queries can in turn be viewed as the intersection of a number of halfplane range queries.

The index in [12] was the first optimal data structure for answering two-dimensional halfpspace range queries in the worst case, based on the geometric technique called *filtering search* [13,14,15]. It uses O(n) blocks of space and answers a query using O($\log_B n$+t) I/Os. It is also simple enough to be efficient in practice.

6. Conclusions

We presented external memory approximate and worst-case mechanisms for indexing mobile objects that move on the plane, in order to efficiently answer range queries about their location in the future.

Acknowledgements

The authors would like to thank the Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the Program PYTHAGORAS, for funding the above work.

References

[1] L. Arge, V. Samoladas, and J.S. Vitter. On Two-Dimensinal Indexability and Optimal Range Search Indexing. In *Proc. of the 18th ACM Symp. on Principles of Database Systems (PODS)*, pages 346–357, June 1999.

[2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proc. of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 322–331, Atlantic City, May 1998.

[3] J. Goldstein, R. Ramakrishnan, U. Shaft, and J.B. Yu. Processing Queries By Linear Constraints. In *Proc. 16th ACM PODS Symposium on Principles of Database Systems*, pages 257–267, Tuscon, Arizona, 1997.

[4] A. Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In *Proc. ACM SIGMOD*, pages 47–57, Boston, Mass, June 1984.

[5] H. V. Jagadish. On Indexing Line Segments. In *Proc. 16th. International Conference on Very Large Data Ba`ses*, pages 614–625, Brisbane, Queensland, Australia, August 1990.

[6] G. Kollios, D. Gunopulos, and V. Tsotras. On Indexing Mobile Objects. *In Proc. of the 18th ACM Symp. on Principles of Database Systems (PODS)*, pages 261–272, June 1999.

[7] S. Saltenis, C. Jensen, S. Leutenegger, and Mario A. Lopez. Indexing the Positions of Continuously Moving Objects. *In Proceedings of the ACM SIGMOD*, pages 331–342, May 2000.

[8] S. Saltenis and C. S. Jensen. Indexing of Moving Objects for Location-Based Services. In *Proc. 18th. Inter. Conference on Data Engineering*, San Jose, CA, Feb 2002. pages 507–518, Brighton, England, September 1987.

[9] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and Querying Moving Objects. *In Proceedings of the 13th ICDE, Birmingham, U.K*, pages 422–432, April 1997. Indexing Method. *The Computer Journal*, 41(3):185–200, 1998.

[10] O. Wolfson, S. Chamberlain, S.Dao, L. Jiang, and G. Mendez. Cost and Imprecision in Modeling the Position of Moving Objects. In *Proc. 14th IEEE Inter. Conf. on Data Engineering*, pages 588–596, Orlando, Florida, February 1998.

[11] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving Objects Databases: Issues and Solutions. In *Proc. of 11th Int. Conf. on Scientfic and Statistical Database Management*, pages 111–122, Capri, Italy, Jul 1998.

[12] P.K.Agarwal, L.Arge,J.Erickson, P.G.Franciosa, J.S.Vitter: *Efficient Searching with Linear Constraints,* Journal of Computer and System Sciences 61, 194-216 (2000).

[13] B. Chazelle, Filtering Search:a new approach to *query-answering, SIAM J. Comput. 15 (1986), 703-724.*

[14] B. Chazelle, R. Cole, F.P. Preparata, C.K. Yap, *New upper bounds for neighbor searching,* Inform. Control 68(1986), 105-124.

[15] B. Chazelle, F.P.Preparata, *Halfspace range search:* An algorithmic application of k-sets, *Descrete Comput. Geom. 1 (1986), 83-93.*

[16] M. Vazirgiannis, *A foundation for Representing and Querying Moving Objects,* ACM Transactions on Database Systems, Vol. 25, No.1, March 2000, Pages 1-42.