Selectivity Estimation for Predictive Spatio-Temporal Queries

Yufei Tao[1], Jimeng Sun[2], Dimitris Papadias[2]

[1]*Department of Computer Science*
Carnegie Mellon University
Pittsburgh PA, USA, 15213-3891
taoyf@cs.cmu.edu

[2]*Department of Computer Science*
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{ jimeng, dimitris}@cs.ust.hk

Abstract

This paper proposes a cost model for selectivity estimation of predictive spatio-temporal window queries. Initially, we focus on uniform data proposing formulae that capture both points and rectangles, and any type of object/query mobility combination (i.e., dynamic objects, dynamic queries or both). Then, we apply the model to non-uniform datasets by introducing spatio-temporal histograms, which in addition to the spatial, also consider the velocity distributions during partitioning. The advantages of our techniques are (i) high accuracy (1-2 orders of magnitude lower error than previous techniques), (ii) ability to handle all query types, and (iii) efficient handling of updates.

1. Introduction

Spatio-temporal database management systems (STDBMS) have received considerable attention [AAE00, KGT99, PJT99, SJLL00, SJ02, TP01, TP02] in recent years due to the emergence of numerous applications (e.g., traffic supervision, flight control, weather forecast, etc) that require management of continuously moving objects. An important operation in these systems is to predict objects' future location based on information at the current time. For this purpose object movement is usually represented as a function of time, and the database stores the function parameters. For example, given the location $o(0)$ of object o at the current time 0 and its velocity o_V, its position at some future time t can be estimated as $o(t)=o(0)+o_V \cdot t$. An update to the database is necessary only when the function parameters (i.e., o_V) change.

The most common query type in STDBMS is the *window query*, which, given a query region q_R and a future time interval q_T, retrieves all objects whose extents will intersect q_R during q_T. For instance, consider "retrieve all residential areas that will be covered by typhoon *Mike* tomorrow based on its current spreading speed". In this example, data (residential areas) are static and the query (typhoon) is dynamic, while in some cases (e.g., "return all vehicles that will be in the city center within the next 10 minutes") the reverse is true. Furthermore, both the data and the query can be moving (e.g., "report all airplanes that will be within 10 miles from flight UA100 in 20 minutes").

The selectivity of a window query is defined as the number of retrieved objects divided by the cardinality of the dataset, and its accurate estimation is important for query optimization. Although various techniques [APR99, AN00, BF95, KF93, SAE02] have been proposed to estimate selectivity in traditional spatial databases (of static objects), their application to moving objects results in significant errors. Choi and Chung [CC02] conducted the first analysis for STDBMS, focusing on moving points and static queries. They derive formulae for one-dimensional space, which are then extended for the multi-dimensional case. As discussed shortly, however, this method may lead to large errors even in two dimensions. Furthermore, it does not address rectangular data and moving queries (which are common in practice).

This paper presents a comprehensive study for window query selectivity estimation that supports all types of objects (static/dynamic, points/rectangles) and moving queries. In particular, we prove several interesting properties which reduce complex problems (e.g., dynamic rectangle data) to simpler ones (i.e., static point data), and thus simplify the derivation and the resulting equations considerably. Unlike the previous methods, our analysis solves the problem directly in the multi-dimensional space, avoiding the inaccuracy caused by approximations. Furthermore, we present a spatio-temporal histogram, which (i) leads to accurate estimation for arbitrary data distributions, and (ii) can be incrementally maintained (while the traditional methods require very frequent re-building). Extensive experimentation confirms that the proposed techniques yield error less than 10% in all cases.

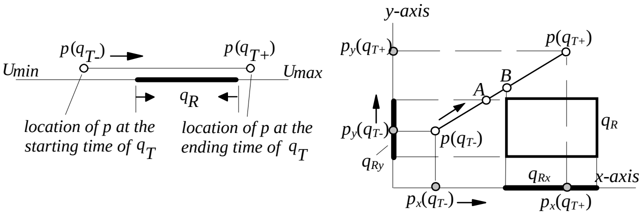
The rest of the paper is organized as follows. Section 2 introduces related work on selectivity estimation, histograms and spatio-temporal access methods. Section 3 analyzes the problem for uniform data, while Section 4 extends the results to non-uniform datasets using histograms. Section 5 experimentally evaluates the proposed methods, and Section 6 concludes the paper with directions for future work.

2. Related Work

Section 2.1 reviews the only existing approach ([CC02]) for window selectivity estimation in STDBMS. Then, Section 2.2 introduces *MinSkew*, a popular histogram for spatial databases, and discusses how it can be adapted for moving data.

2.1 Existing Estimation Technique

Choi and Chung [CC02] focus on dynamic point data and static queries (i.e., the query region remains fixed) starting from the one-dimensional case, where the spatial universe is a line segment [U_{min}, U_{max}] (U_{min} and U_{max} are the coordinates of the boundaries). Their goal is to predict the percentage of points that will intersect the query extent q_R during the query interval q_T=[q_{T-}, q_{T+}] ($0 \leq q_{T-} \leq q_{T+}$, the current time is 0). Figure 2.1a shows q_R and the positions $p(q_{T-})$ and $p(q_{T+})$ of a data point p at the starting q_{T-} and ending q_{T+} timestamps of q_T, respectively. The distance between $p(q_{T-})$ and $p(q_{T+})$ depends on the velocity p_V of p, which distributes uniformly in the range [V_{min}, V_{max}]. Clearly, point p satisfies the query if and only if the segment connecting $p(q_{T-})$ and $p(q_{T+})$ intersects q_R. Assuming that the location of p at the current time 0 follows uniform distribution in [U_{min}, U_{max}], [CC02] derives the probability (i.e., also the selectivity of q) that a data point qualifies q, as a function of U_{min}, U_{max}, V_{min}, V_{max}, and the query parameters.



(a) 1D case (b) 2D case

Figure 2.1: Window queries in 1D and 2D spaces

The multi-dimensional version of the problem is converted to the 1D case by projecting objects and queries onto individual dimensions. In particular, the probability that p satisfies q is computed as $\prod_{i=1 \sim m} Sel_i$, where m is the dimensionality and Sel_i is the 1D selectivity (i.e., the probability that the projection p_i of point p on the i-th dimension intersects the projection q_i of the query during interval q_T). This, however, is inaccurate due to the fact that a data point may still violate a query q, even if its projection intersects that of q on every dimension. For instance, in Figure 2.1b p is not a qualifying point because it never appears in the region q_R. However, the projections of its trajectory (during q_T) on both dimensions intersect the corresponding projections of q_R (i.e., segments q_{Rx} and q_{Ry}). Therefore $\prod_{i=1 \sim m} Sel_i$ over-estimates the actual probability.

In general, an object o satisfies a spatio-temporal window query q if (i) the trajectory projection of o intersects that of q on each dimension (i.e., the *spatial condition*), and (ii) the intersection time intervals on all dimensions must overlap (i.e., the *temporal condition*). Let T_A and T_B be the timestamps when p reaches location A and B in Figure 2.1b; then the x-intersection interval (i.e., the period when the x-projections of p and q intersect) is [T_B, q_{T+}], while that on the y-dimension is [q_{T-}, T_A]. Point p does not satisfy the query because the two intersection intervals are disjoint, thus violating condition (ii). The estimation in [CC02] ignores the temporal condition (hence in the sequel we refer to the method as the *time-oblivious approach*), which as shown in the experiments may lead to significant estimation error. In Section 3, we will mathematically quantify this error and elaborate the influential factors. Finally, as mentioned earlier, [CC02] does not address rectangle objects and moving queries.

2.2 MinSkew

MinSkew [APR99] is a spatial histogram originally proposed for selectivity estimation of window queries in non-uniform datasets. It partitions the space into a set of buckets such that the minimum bounding rectangles (MBRs) of all buckets are disjoint, and their union covers the entire universe. Each bucket b_i contains the number $b_i.num$ of objects whose centroids fall inside b_i.MBR, and the average extent $b_i.len$ of these objects. Figure 2.2 illustrates a query q and an intersecting bucket b in the 2D space. The gray area corresponds to the intersection between b.MBR and the *extended query region*, obtained by enlarging each edge of q with distance $b.len/2$. The expected number of rectangles in b intersecting q is estimated as $b.num \times areaG$/area(b.MBR), where $areaG$ and area(b.MBR) are the areas of the gray region and b.MBR, respectively. The estimated selectivity is obtained by summing the results for all such intersecting buckets.

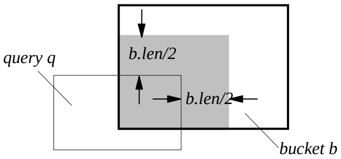


Figure 2.2: Estimating the selectivity inside a bucket

To ensure satisfactory accuracy, the above estimation requires that (i) objects in each bucket b have similar sizes and (ii) their centroids distribute uniformly in b.MBR. To quantify the degree of uniformity, [APR99] defines the *spatial-skew* (denoted as $b.skew$) for a bucket b as the statistical variance[1] of the spatial densities[2] of all points inside it. Since a small spatial-skew indicates better uniformity, MinSkew aims at minimizing $\sum_{i=1 \sim m}(b_i.num \cdot b_i.skew)$, i.e., the weighted sum of the spatial-skews of the buckets. Computing the optimal buckets, however, is NP-hard [MPS99]. To reduce the computation cost, [APR99] partitions the original space into a grid with $H \times H$ regular cells (where H is the *resolution*), and associate each cell c with (i) the number $c.num$ of objects whose centroids fall in c.MBR, (ii) the average extent length $c.len$ of objects satisfying (i), and (iii) the density $c.den$ of the

[1] Given n numbers a_1, a_2, …, a_n, the statistical variance equals $\frac{1}{n}\sum_{i=1}^{n}\left(a_i - \bar{a}\right)^2$, where $\bar{a}$ is the average of a_1, a_2, …, a_n.

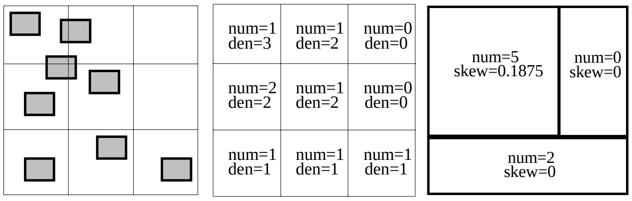
[2] The density of a point is defined as the number of objects that cover the point.

cell (i.e., the number of objects intersecting *c*.MBR). Figure 2.3a shows an example (*H*=3) for a dataset with 8 objects, and Figure 2.3b illustrates the information associated with the cells (*len* is not shown because it is not needed for partitioning). A greedy algorithm builds the histogram that minimizes the total-skew, under the constraint that each bucket must cover an integer number of cells. The final buckets are shown in Figure 2.3c, together with their associated information computed as follows:

$$b.num = \sum_{each\ cell\ c\ in\ b} c.num \, , \quad b.len = \frac{\sum_{each\ cell\ c\ in\ b} c.num \cdot c.len}{\sum_{each\ cell\ c\ in\ b} c.num}$$

$$b.skew = \frac{1}{|C|} \sum_{each\ cell\ c\ in\ b} \left(c.den - \overline{den}\right)^2$$

where *b* denotes a bucket, |*C*| is the number of cells covered by *b*, and $\overline{den}$ denotes their average density. MinSkew can be applied in arbitrary dimensionality with straightforward modifications.



num=1
den=3
	num=1
	den=2
		num=0
		den=0

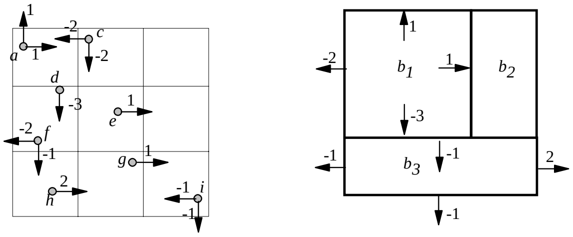
num=2
den=2
	num=1
	den=2
		num=0
		den=0

num=1
den=1
	num=1
	den=1
		num=1
		den=1

(a) 3×3 grid (b) Cell information (c) Final buckets

Figure 2.3: Building the histogram

[CC02] extends MinSkew with velocities for spatio-temporal window queries. Figure 2.4a shows 8 moving points, where the arrows (numbers) indicate the directions (values) of their velocities (a minus value indicates movement towards the negative direction of the axis). Figure 2.4b shows the corresponding spatio-temporal histogram built in two steps.



(a) Location and velocities (b) The final buckets

Figure 2.4: A spatio-temporal histogram

First, the spatial extents of the buckets are determined in the same way as the traditional MinSkew algorithm (by ignoring the velocities). Then, each bucket *b* is associated with a velocity bounding rectangle (VBR) (b_{Vx-}, b_{Vx+}, b_{Vy-}, b_{Vy+}), such that (i) b_{Vx-} (b_{Vx+}) equals the minimum (maximum) velocity along the x-dimension of the objects inside, and (ii) b_{Vy-} (b_{Vy+}) is defined similarly with respect to the y-dimension. In Figure 2.4b, the VBRs of buckets b_1, b_2, and b_3 are (-2, 1, -3, 1), (0, 0, 0, 0), and (-1, 2, -1, -1), respectively.

Accurate spatio-temporal selectivity estimation requires that the location (velocities) of the objects inside a bucket uniformly distribute in the bucket's MBR (VBR). The data partition in Figure 2.4, however, is decided according to spatial information; thus, the uniformity of velocity cannot be guaranteed, which may lead to significant estimation error. Furthermore, the histogram is not incrementally updatable and must be re-built very frequently to maintain satisfactory accuracy ([CC02] suggests re-building at every single timestamp). To see this, assume that in Figure 2.4 the y-velocity of object *d* (which determines b_{1Vy-}) changes to -1, after which b_{1Vy-} should be adjusted to the y-velocity of *c* (i.e., -2), because it is now the minimum y-velocity of all objects in b_1. This, however, is not possible because the histogram does not contain detailed information about the velocities of individual objects. In Section 4, we discuss alternative solutions to overcome these problems.

3. Spatio-Temporal Window Query Selectivity

Let *r* be a moving rectangle in *m*-dimensional space. The extent of *r* at the current time 0 is a 2*m*-dimensional vector $r_R = \{r_{R1-}, r_{R1+}, r_{R2-}, r_{R2+}, \ldots, r_{Rm-}, r_{Rm+}\}$, where $[r_{Ri-}, r_{Ri+}]$ is the extent along the *i*-th dimension ($1 \le i \le m$). Vector $r_V = \{r_{V1-}, r_{V1+}, r_{V2-}, r_{V2+}, \ldots, r_{Vm-}, r_{Vm+}\}$ represents the velocities of *r*, such that r_{Vi-} (r_{Vi+}) is the velocity of the lower (upper) boundary of *r* on the *i*-th dimension ($1 \le i \le m$). The extent $r_R(t)$ (also a 2*m*-dimensional vector) of *r* at some future time *t* can be computed from r_R and r_V as: $r_R(t)=r_R+t \cdot r_V$. A moving point *p* is represented in a similar way: $p_R=\{p_{R1}, p_{R2}, \ldots, p_{Rm}\}$ and $p_V=\{p_{V1}, p_{V2}, \ldots, p_{Vm}\}$ are the coordinates and velocities on the *m* dimensions. Note that this representation captures static objects by setting all velocities to 0. Without loss of generality, we consider that for the *i*-th dimension, the space has extent $[U_{min-i}, U_{max-i}]$, and possible velocity values fall in the range $[V_{min-i}, V_{max-i}]$ (i.e., the velocity space). Following the common methodology, we assume independent dimensions.

Given a set *S* of moving objects, a spatio-temporal query *q* specifies (i) a moving rectangle with current extent q_R and velocity vector q_V , (ii) a future time interval $q_T=[q_{T-}, q_{T+}]$ ($0 \le q_{T-} \le q_{T+}$), and retrieves all objects *o* that intersect *q* during q_T, or more formally, there exists some timestamp $t \in [q_{T-}, q_{T+}]$ such that $o_R(t)$ intersects $q_R(t)$. If $q_{T-}=0$ (i.e., the current time), the query is called a *current query*. In this section we deal with uniform data, i.e., the extent $[r_{Ri-}, r_{Ri+}]$ of a rectangle *r* (or p_{Ri} of a point *p*) uniformly distributes in $[U_{min-i}, U_{max-i}]$, and similarly, the velocity range $[r_{Vi-}, r_{Vi+}]$ of *r* (or p_{Vi} of *p*) follows uniform distribution in $[V_{min-i}, V_{max-i}]$, for all $1 \le i \le m$. The goal is to predict the selectivity using $[U_{min-i}, U_{max-i}]$, $[V_{min-i}, V_{max-i}]$ and *q*.

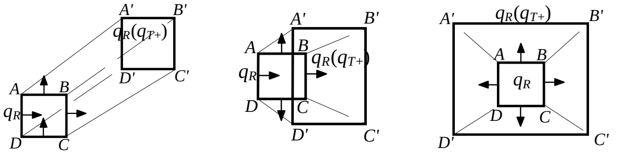
Our analysis is based on the observation that any case of spatio-temporal selectivity estimation can be reduced to predicting the selectivity of a moving rectangle query on a set of static points. In section 3.1 we study this basic problem, and then, illustrate how to reduce other problem instances to the basic case. Table 3.1 lists the symbols that will be used frequently in the derivation.

Symbol	Description
DS	the data space
m	dimensionality of DS
$[U_{min\text{-}i}, U_{max\text{-}i}]$	the extent of DS on the i-th axis
$[V_{min\text{-}i}, V_{max\text{-}i}]$	the velocity range on the i-th axis
$r_R=\{r_{R1\text{-}}, r_{R1+}, r_{R2\text{-}}, r_{R2+}, \ldots, r_{Rm\text{-}}, r_{Rm+}\}$	extent of moving rectangle r at the current time
$r_V=\{r_{V1\text{-}}, r_{V1+}, r_{V2\text{-}}, r_{V2+}, \ldots, r_{Vm\text{-}}, r_{Vm+}\}$	velocity vector of moving rectangle r
$p_R=\{p_{R1}, p_{R2}, \ldots, p_{Rm}\}$	coordinates of moving point p
$p_V=\{p_{V1}, p_{V2}, \ldots, p_{Vm}\}$	velocities of moving point p
$q_T=[q_{T\text{-}}, q_{T+}]$	query time interval
$CX(q)$	the convex hull of corner points of $q_R(q_{T-})$ and $q_R(q_{T+})$
$Sel_{static\text{-}pt}$	selectivity for static points
Sel_{pt}	selectivity for dynamic points
Sel_{rec}	selectivity for dynamic rectangles

Table 3.1: Frequent symbols in the analysis

3.1 Static Point Data

A static point p satisfies a moving query q if p lies inside $q_R(t)$ at some timestamp $t\in q_T$. For the sake of simplicity, we first focus on current queries (i.e., $q_{T\text{-}}=0$). Figure 3.1a shows a 2D moving query q, where q_R and $q_R(q_{T+})$ (i.e., rectangles $ABCD$ and $A'B'C'D'$) indicate the positions of q at the starting (0) and ending time (q_{T+}) of q_T, respectively. The velocity directions of q_V are indicated with arrows. Let $CX(q)$ be the convex hull of all the corner points of q_R and $q_R(q_{T+})$ (i.e., polygon $ADCC'B'A'$ in Figure 3.1a).



(a) 2 trapezoids (b) 3 trapezoids (c) 4 trapezoids

Figure 3.1: Calculating the area of $CX(q)$

$CX(q)$ corresponds to the region that is "swept" by q during q_T, and as a result, a data point p will be retrieved if and only if it lies in $CX(q)$. Since the data distribution is uniform, the probability for a point to fall inside $CX(q)$ is the ratio between the area of $CX(q)$ and that of the spatial universe, which is also the selectivity $Sel_{static\text{-}pt}$ of q:

$$Sel_{static\_pt}\left(q_{Ri-}, q_{Ri+}, q_{Vi-}, q_{Vi+}, q_T\right) = \frac{area\left(CX\left(q\right)\right)}{\prod_{i=1}^{m}\left(U_{\max-i} - U_{\min-i}\right)} \qquad (3\text{-}1)$$

The area of $CX(q)$ depends on the velocity directions of q_V. In Figure 3.1a, for example, $q_{Vi\text{-}}$ and q_{Vi+} have the same directions along all dimensions ($q_{Vx\text{-}}$ and q_{Vx+} are to the right, while $q_{Vy\text{-}}$ and q_{Vy+} are upwards). In this case, the area of $CX(q)$ is the sum of rectangle $ABCD$ (i.e., query's extent at the current time), and two trapezoids $ABB'A'$ and $BCC'B'$. In particular, trapezoid $ABB'A'$ ($BCC'D'$) is the region swept by segment AB (BC) during q_T. Figure 3.1b shows another case where $q_{Vx\text{-}}$ and q_{Vx+} still have the same direction, while $q_{Vy\text{-}}$ and q_{Vy+} are opposite. Then, the area of $CX(q)$ is the sum of rectangle $ABCD$, and three trapezoids $ABB'A'$, $BCC'B'$, and $DD'C'C$ (swept by segments AB, BC, CD respectively). Figure 3.1c illustrates the third case where velocities on all dimensions have opposite directions; the area of $CX(q)$ is the sum of rectangle $ABCD$ and four trapezoids $ABB'A'$, $BCC'B'$, $DD'C'C$, $AA'D'D$ (swept by segments AB, BC, CD, DA).

Computing the area of a single trapezoid is relatively easy. Consider, for example, trapezoid $ABB'A'$, where the lengths of AB and $A'B'$ are (q_{Rx+}–$q_{Rx\text{-}}$) and (q_{Rx+}–$q_{Rx\text{-}}$)+(q_{Vx+}–$q_{Vx\text{-}}$) ·(q_{T+}–$q_{T\text{-}}$), respectively. Furthermore, note that the vertical distance between AB and $A'B'$ is q_{Vy+}·(q_{T+}–$q_{T\text{-}}$); thus, the area of trapezoid $ABB'A'$ is given by:

$$area\left(ABB'A'\right) =$$
$$\tfrac{1}{2}\left[2\left(q_{Rx+} - q_{Rx-}\right) + \left(q_{Vx+} - q_{Vx-}\right)\left(q_{T+} - q_{T-}\right)\right] \cdot q_{Vy+}\left(q_{T+} - q_{T-}\right)$$

In general m-dimensional spaces, each trapezoid is the region swept by a boundary of q_R, which is a (m–1)-dimensional rectangle. Specifically, the trapezoid volumes decided by the lower and upper boundaries on the i-th dimension (1≤i≤m) can be calculated using equations (3-2) and (3-3), respectively:

$$area\left(Trapezoid_{Lower-i}\right) = \tfrac{1}{2} q_{Vi-}\left(q_{T+} - q_{T-}\right) \cdot \left\{\begin{array}{l} \prod_{j\neq i}\left(q_{Rj+} - q_{Rj-}\right) + \\ \prod_{j\neq i}\left[\left(q_{Rj+} - q_{Rj-}\right) + \left(q_{Vj+} - q_{Vj-}\right)\left(q_{T+} - q_{T-}\right)\right] \end{array}\right\} \qquad (3\text{-}2)$$

$$area\left(Trapezoid_{Upper-i}\right) = \tfrac{1}{2} q_{Vi+}\left(q_{T+} - q_{T-}\right) \cdot \left\{\begin{array}{l} \prod_{j\neq i}\left(q_{Rj+} - q_{Rj-}\right) + \\ \prod_{j\neq i}\left[\left(q_{Rj+} - q_{Rj-}\right) + \left(q_{Vj+} - q_{Vj-}\right)\left(q_{T+} - q_{T-}\right)\right] \end{array}\right\} \qquad (3\text{-}3)$$

Figure 3.2 shows the algorithm for computing the volume of $CX(q)$ in m-dimensional spaces, after which the selectivity of the query can be obtained using equation (3-1). The handling of non-current queries (i.e., $q_{T\text{-}}$>0) is straightforward. The only difference is that $CX(q)$ should be the convex hull of the corner points of rectangles $q_R(q_{T\text{-}})$ and $q_R(q_{T+})$. The volume of $CX(q)$ can still be calculated using the algorithm of Figure 3.2.

So far we have assumed that $CX(q)$ lies entirely in the spatial universe DS, while in some cases part of $CX(q)$ may fall outside DS (i.e., the query moves out of DS during q_T) as shown in Figure 3.3. Note that the probability that a data point satisfies q now corresponds to the area of the intersection between $CX(q)$ and DS. In Figure 3.3, for example, the intersection region is hexagon $AEFGCD$, whose area is the sum of *ABCD* (i.e., the extent of the query at the current time) and two trapezoids *ABFE* and *BCGF*.

```
Algorithm compute_CX_vol (q)
$\prod_{i=1\sim m}[q_{Ri+}(q_{T-})-q_{Ri-}(q_{T-})]$
for each dimension $1\le i\le m$
    if $q_{Vi-}<0$ and $q_{Vi+}<0$ then
        $sum=sum+area(Trapezoid_{Lower-i})$ (equation 3-2)
    if $q_{Vi-}>0$ and $q_{Vi+}>0$ then
        $sum=sum+area(Trapezoid_{Upper-i})$ (equation 3-3)
    if $q_{Vi-}<0$ and $q_{Vi+}>0$ then
        $sum=sum+area(Trapezoid_{Lower-i})$
            $+area(Trapezoid_{Upper-i})$
return sum
End compute_CX_vol
```

Figure 3.2: Algorithm for computing volume of *CX*(*q*)

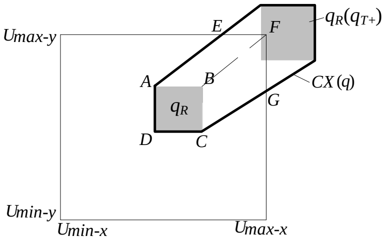


Figure 3.3: *CX*(*q*) is not completely in *DS*

3.2 Dynamic Point Data

In this section we discuss selectivity estimation for dynamic points, where the location p_{Ri} and velocity p_{Vi} of each point *p* along the *i*-th ($1\le i\le m$) dimension distributes uniformly in [$U_{min\text{-}i}$, $U_{max\text{-}i}$] and [$V_{min\text{-}i}$, $V_{max\text{-}i}$], respectively. Given a moving query *q*, we aim at deriving the probability $P(u_1, u_2, \ldots, u_m)$ that a point *p* satisfies *q* when its velocity p_{Vi} takes a specific value u_i ($1\le i\le m$). Once $P(u_1, u_2, \ldots, u_m)$ has been derived, the query selectivity Sel_{pt} can be obtained by integrating all possible values of p_{Vi}:

$$Sel_{pt}\left(q_{Ri-},q_{Ri+},q_{Vi-},q_{Vi+},q_T\right)= \int_{V_{\min-1}}^{V_{\max-1}}\int_{V_{\min-2}}^{V_{\max-2}}\cdots\int_{V_{\min-m}}^{V_{\max-m}} P\left(u_1,u_2,\ldots,u_m\right) f\left(u_1,u_2,\ldots,u_m\right) du_m \ldots du_2 du_1 \quad (3\text{-}4)$$

where $f(u_1, u_2, \ldots, u_m)$ is the joint probability density function of $u_1, u_2, \ldots, u_m$. Since all dimensions are independent and u_i uniformly distributes in [$V_{min\text{-}i}$, $V_{max\text{-}i}$], we have:

$$f\left(u_1,u_2,\ldots,u_m\right)=f\left(u_1\right)\cdot f\left(u_2\right)\cdot\ldots\cdot f\left(u_m\right)= \prod_{i=1}^{m}\left(\frac{1}{V_{\max-i}-V_{\min-i}}\right)$$

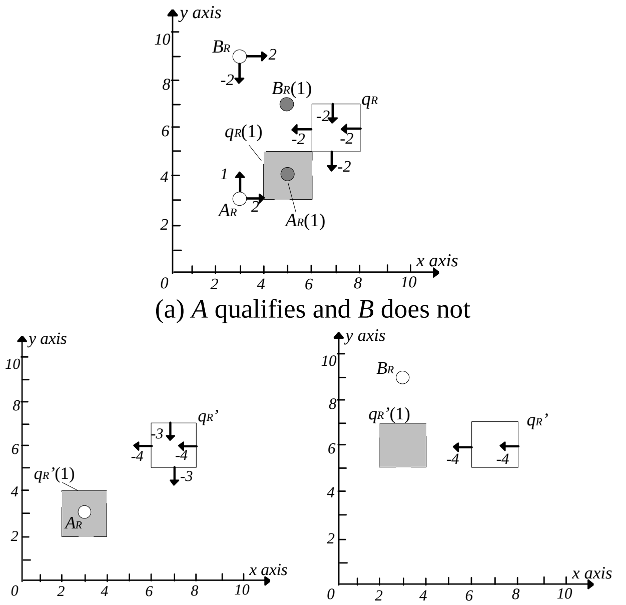
Hence equation (3-4) can be written as:

$$Sel_{pt}\left(q_{Ri-},q_{Ri+},q_{Vi-},q_{Vi+},q_T\right)=\prod_{i=1}^{m}\left(\frac{1}{V_{\max-i}-V_{\min-i}}\right)\cdot \int_{V_{\min-1}}^{V_{\max-1}}\int_{V_{\min-2}}^{V_{\max-2}}\cdots\int_{V_{\min-m}}^{V_{\max-m}} P\left(u_1,u_2,\ldots,u_m\right) du_m \ldots du_2 du_1 \quad (3\text{-}5)$$

The derivation of $P(u_1, u_2, \ldots, u_m)$ can be reduced to the case of static points based on the following lemma:

Lemma 3.1: Let *p* be a *m*-D point with current location p_R and velocity vector $p_V=\{p_{V1}, p_{V2}, \ldots, p_{Vm}\}$. Given a moving query *q*, we formulate another query *q'* such that (i) its current extent q_R' and time interval q_T' are the same as q_R and q_T, and (ii) $q_{Vi-}'=q_{Vi-}-p_{Vi}$, $q_{Vi+}'=q_{Vi+}-p_{Vi}$. Then, *p* satisfies *q*, if and only if query *q'* covers the static point p_R during q_T. ■

Lemma 3.1 indicates that deciding whether a moving point *p* intersects a moving rectangle *q* can be achieved by examining the intersection between a static point p_R and a moving rectangle *q'*, where p_R is the current location of *p*, and *q'* is formulated as described above. To illustrate this, consider Figure 3.4a which shows two moving 2D points *A*, *B* and query *q* with time interval q_T=[0, 1]. $A_R(1)$, $B_R(1)$, $q_R(1)$ correspond to the positions of points *A*, *B*, and query *q* at time 1, respectively. It is clear that *A* satisfies *q* while *B* does not. Figure 3.4b shows the formulated query *q'* in order to decide the intersection of *A* (observe how the velocities of *q'* change from those of *q*).



(a) *A* qualifies and *B* does not

(b) Query formulated for *A*   (c) Query formulated for *B*

Figure 3.4: Illustration of Lemma 3.1

According to Lemma 3.1 the fact that *A* is a qualifying object guarantees that *q'* must cover static point A_R during q_T, which is indeed the case as shown in Figure 3.4b. In particular, notice that the relative positions of A_R and $q_R'(1)$ in Figure 3.4b are the same as those of $A_R(1)$ and q_R (1) in Figure 3.4a. In general, given a data point *p* and a query *q*, the relative positions between $p_R(t)$ and $q_R(t)$ are always the same as those between static point p_R and the extent $q_R'(t)$ of the transformed query *q'* at any future time *t*. Figure 3.4c demonstrates the formulated query *q'* with respect to point *B* (notice that the y-velocities of *q'* are 0). Since *B* does not intersect q, by Lemma 3.1 we can infer that q' does not cover B_R.

Therefore, the probability $P(u_1, u_2, \ldots, u_m)$ for a moving point p with velocities $u_1, u_2, \ldots, u_m$ to intersect a query q equals the probability that the corresponding formulated query q' covers the static point p_R. Specifically, $P(u_1, u_2, \ldots, u_m)$ can be represented as:

$$\begin{aligned} P(u_1,u_2,\ldots,u_m) &= Sel_{static-pt}(q_{Ri-}', q_{Ri+}', q_{Vi-}', q_{Vi+}', q_T') \\ &= Sel_{static-pt}(q_{Ri-}, q_{Ri+}, q_{Vi-}-u_i, q_{Vi+}-u_i, q_T) \end{aligned} \qquad (3\text{-}6)$$

where $Sel_{static\text{-}pt}$ is the selectivity for static points in equation (3-1). As discussed earlier, after solving $P(u_1, u_2, \ldots, u_m)$, equation (3-5) estimates the selectivity of spatio-temporal window queries on moving points. Static queries over dynamic points (i.e., the case discussed in [CC02]) constitute just a special instance of the general problem and can be solved by the above method.

3.3 Dynamic Rectangles

This section analyzes the problem for a set S of moving rectangles. For each rectangle $r \in S$ and each dimension i ($1 \le i \le m$), (i) the extent of r equals L_i (a dataset constant), and r_{Ri-} (the left boundary) uniformly distributes in $[U_{min-i}, U_{max-i}–L_i]$, (ii) the velocity range $r_{Vi+}–r_{Vi-}$ equals LV_i (also a constant), and r_{Vi-} (the velocity of the left boundary) follows uniform distribution in $[V_{min-i}, V_{max-i}–LV_i]$. Datasets with rectangles that have different extents and velocity ranges will be handled using histograms in the next section. Similar to the analysis for dynamic points, we aim at deriving the probability $P(u_1, u_2, \ldots, u_m)$ that a rectangle r, whose r_{Vi-} takes specific a value u_i ($1 \le i \le m$), satisfies the query. Once $P(u_1, u_2, \ldots, u_m)$ is available, Sel_{rec} can be obtained by equation (3-7) (notice the changes in the upper limits of the integrals compared with equation 3-4):

$$Sel_{rec}(q_{Ri-}, q_{Ri+}, q_{Vi-}, q_{Vi+}, q_T) = \qquad (3\text{-}7)$$
$$\int_{V_{\min-1}}^{V_{\max-1}-LV_1} \int_{V_{\min-2}}^{V_{\max-2}-LV_2} \cdots \int_{V_{\min-m}}^{V_{\max-m}-LV_m} P(u_1,u_2,\ldots,u_m) f(u_1,u_2,\ldots,u_m)\, du_m \ldots du_2 du_1$$

Since u_i distributes uniformly in $[V_{min-i}, V_{max-i}–LV_i]$, we have:

$$f(u_1,u_2,\ldots,u_m) =$$
$$f(u_1)\cdot f(u_2)\cdot\ldots\cdot f(u_m) = \prod_{i=1}^{m}\left(\frac{1}{V_{\max-i}-LV_i-V_{\min-i}}\right)$$

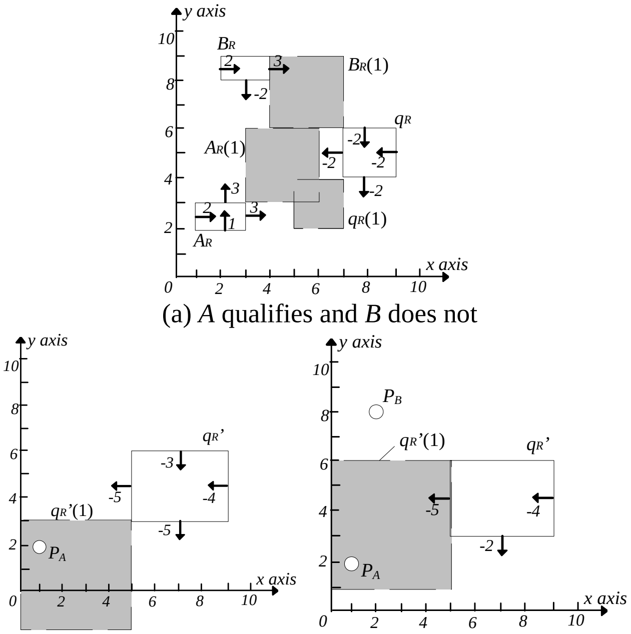
Thus, equation (3-7) becomes:

$$Sel_{rec}(q_{Ri-}, q_{Ri+}, q_{Vi-}, q_{Vi+}, q_T) = \prod_{i=1}^{m}\left(\frac{1}{V_{\max-i}-LV_i-V_{\min-i}}\right) \qquad (3\text{-}8)$$
$$\int_{V_{\min-1}}^{V_{\max-1}-LV_1} \int_{V_{\min-2}}^{V_{\max-2}-LV_2} \cdots \int_{V_{\min-m}}^{V_{\max-m}-LV_m} P(u_1,u_2,\ldots,u_m) du_m \ldots du_2 du_1$$

The following lemma reduces the intersection examination between two moving rectangles r and q to that between a static point (a corner point of r_R) and a formulated moving rectangle q'.

Lemma 3.2: Let r be a m-dimensional rectangle whose current extent is $r_R=\{r_{R1-}, r_{R1+}, r_{R2-}, r_{R2+}, \ldots, r_{Rm-}, r_{Rm+}\}$ and velocity vector is $r_V=\{r_{V1-}, r_{V1+}, r_{V2-}, r_{V2+}, \ldots, r_{Vm-}, r_{Vm+}\}$. Given a moving query q with $q_R=\{q_{R1-}, q_{R1+}, q_{R2-}, q_{R2+}, \ldots, q_{Rm-}, q_{Rm+}\}$, and $q_V=\{q_{V1-}, q_{V1+}, q_{V2-}, q_{V2+}, \ldots, q_{Vm-}, q_{Vm+}\}$, we formulate another query q' such that (i) $q_T'=q_T$, (ii) $q_{Ri-}'=q_{R1-}–(r_{R1+}–r_{R1-})$, $q_{Ri+}'=q_{Ri+}$, and (iii) $q_{Vi-}'=q_{Vi-}–r_{Vi+}$, $q_{Vi+}'=q_{Vi+}–r_{Vi-}$. Then, r satisfies q, if and only if query q' covers the static point p $\{r_{R1-}, r_{R2-}, \ldots, r_{Rm-}\}$ (i.e., a lower-left corner point of r_R). ■

Consider Figure 3.5a which shows data rectangles A, B, query q (with interval q_T=[0, 1]), and their extents at time 1. Notice that A intersects q during q_T, while B does not. Figure 3.5b shows the transformed query q' with respect to A, as well as the lower-left corner point P_A of A_R. Notice that the current extent q_R' of q' is obtained by enlarging q_R by the size of A_R. The value (-5) of q_{Vx-}', for example, is computed by subtracting A_{Vx+} (3) from q_{Vx-} (-2). Since q' covers static point P_A during q_T, by Lemma 3.2 we can assert that the original object A satisfies q. Similarly, Figure 3.5c demonstrates the formulated query q' for B, which does not cover point P_B (lower-left corner point of B_R) during q_T, indicating that object B does not qualify q.



(a) A qualifies and B does not

(b) Query formulated for A (c) Query formulated for B

Figure 3.5: Illustration for Lemma 3.2

Hence the probability $P(u_1, u_2, \ldots, u_m)$ that a moving rectangle r with $r_{Vi-}=u_i$ ($1 \le i \le m$) satisfies q can be represented as:

$$\begin{aligned} P(u_1,u_2,\ldots,u_m) &= Sel_{static-pt}(q_{Ri-}', q_{Ri+}', q_{Vi-}', q_{Vi+}', q_T') \\ &= Sel_{static-pt}(q_{Ri-}-L_i, q_{Ri+}, q_{Vi-}-u_i-LV_i, q_{Vi+}-u_i, q_T) \end{aligned} \qquad (3\text{-}6)$$

where L_i and LV_i are the spatial and velocity ranges of r along the i-th dimension respectively, and $Sel_{static\text{-}pt}$ is shown in equation (3-1), except that the volume of the universe should be modified to $\prod_{i=1\sim m}(U_{max\text{-}i}-L_i-U_{min\text{-}i})$ (i.e., the left boundary of a data rectangle ranges in $[U_{min\text{-}i}, U_{max\text{-}i}-L_i]$). Replacing $P(u_1, u_2, \ldots, u_m)$ with equation (3-6), we obtain the model for estimating the selectivity for moving rectangles. It is worth pointing out that the general reduction methodology is independent of the model, e.g., it can be applied in conjunction with the formulae of [CC02] to capture dynamic queries and rectangle objects.

3.4 Error of the Time-Oblivious Approach

As discussed in Section 2.1, the time-oblivious approach estimates the selectivity *Sel* by simply taking the product of the qualifying probability Sel_i on each dimension ($1 \le i \le m$). Note that Sel_i can also be obtained from our derivation (i.e., the dimensionality equals 1); hence by comparing the difference between *Sel* and $\prod_{i=1\sim m} Sel_i$ we can quantify the error of the time-oblivious approach. To illustrate the factors that affect the error, in the sequel we consider the case (moving points and static queries) targeted in [CC02], for which the resulting equations are simplest and can be solved into closed form (similar conclusions can be drawn for general settings as shown in the experiments). Specifically, given (i) a set *S* of 2D points such that, for each point $p \in S$, p_{Ri} and p_{Vi} uniformly distribute in $[0, U]$ and $[0, V]$ respectively, and (ii) a static query *q* whose extent is q_R and interval is [0, *T*] (i.e., a current query), the actual selectivity *Sel* is:

$$Sel = \frac{VT}{2U^2}\left[\left(q_{Rx+}-q_{Rx-}\right)+\left(q_{Ry+}-q_{Ry-}\right)\right] + \frac{\left(q_{Rx+}-q_{Rx-}\right)\left(q_{Ry+}-q_{Ry-}\right)}{U^2} \qquad (3\text{-}7)$$

The qualifying probability Sel_i on each dimension ($1 \le i \le m$) can be obtained with similar analysis:

$$Sel_i = \frac{q_{Ri+}-q_{Ri-}}{U} + \frac{VT}{2U} \qquad (3\text{-}8)$$

Thus, the estimation *Sel'* obtained by the time-oblivious approach is:

$$Sel' = Sel_x \cdot Sel_y = \frac{VT}{2U^2}\left[\left(q_{Rx+}-q_{Rx-}\right)+\left(q_{Ry+}-q_{Ry-}\right)\right] + \frac{\left(q_{Rx+}-q_{Rx-}\right)\left(q_{Ry+}-q_{Ry-}\right)}{U^2} + \frac{V^2T^2}{4U^2} \qquad (3\text{-}9)$$

Comparing equations (3-9) and (3-7), the relative error *Err* of *Sel'* is:

$$Err = \frac{Sel'-Sel}{Sel} = \frac{V^2T^2}{2VT\left[\left(q_{Rx+}-q_{Rx-}\right)+\left(q_{Ry+}-q_{Ry-}\right)\right]+4\left(q_{Rx+}-q_{Rx-}\right)\left(q_{Ry+}-q_{Ry-}\right)} \qquad (3\text{-}10)$$

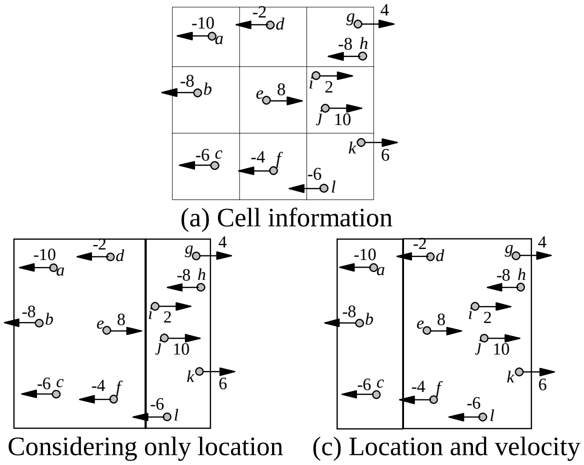
Note that $(q_{Rx+}-q_{Rx-})+(q_{Ry+}-q_{Ry-})$ and $(q_{Rx+}-q_{Rx-})\cdot(q_{Ry+}-q_{Ry-})$ correspond to the perimeter and area of q_R, respectively. The error grows with the interval *T* and the length of the velocity range, decreases with q_R, and is not affected by the length of the spatial universe *U*.

4. Spatio-Temporal Histograms

This section deals with non-uniform data using histograms that partition objects into buckets, such that the distribution within a bucket is almost uniform. Then, the uniform models are applied locally (in each bucket), and the overall prediction is calculated by summing up the individual estimations. In Section 4.1, we discuss the defects of existing histograms, and then present an alternative solution to avoid their problems. Section 4.2 elaborates the algorithm for estimation.

4.1 Histogram Construction and Maintenance

The spatio-temporal histogram of [CC02] partitions the objects based on their spatial location using the conventional MinSkew algorithm, and then decides the VBRs of the buckets. Since the velocity information is not considered during data partition, the resulting histogram cannot ensure the uniformity of velocity distribution in the buckets. Assume, for example, that we want to build a histogram with 2 buckets for the dataset in Figure 4.1a. In Figure 4.1b the buckets are decided based on the objects' location. In particular, the first two columns of cells are grouped into the same bucket because all of them contain exactly one point (i.e., no variance), while cells in the last column (with 2 points each) constitute the second bucket. Notice that, although the location distribution is fairly uniform, the velocity distribution is rather skewed. Consider the left bucket in Figure 4.1b, whose (x-) velocity range is [-10, 8] (i.e., decided by the velocities of points *b* and *e*). Notice that, there are 5 points with velocities in the range [-10, -2], while only one (i.e., *e*) has positive velocity (8). Similarly, the velocity range of the right bucket is [-8, 10], but ranges [-8, 0] and [2, 10] contain 2 and 4 points respectively.



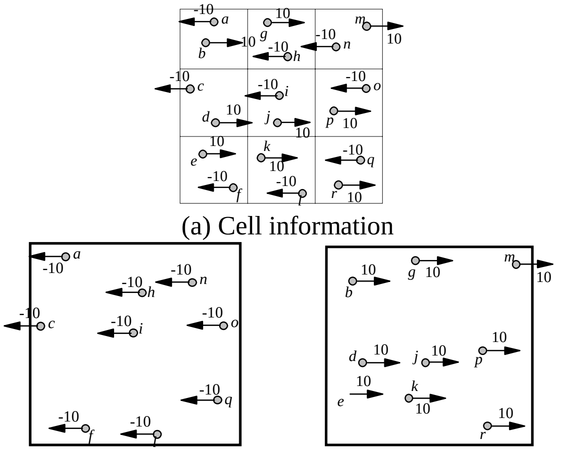
(a) Cell information

(b) Considering only location (c) Location and velocity

Figure 4.1: Uniform velocity distribution

An effective spatio-temporal histogram should partition the data based on both location and velocity information. Continuing the previous example, Figure 4.1c shows such a histogram, where the left and right buckets contain the first and the last two columns of cells respectively. Compared with Figure 4.1b, the spatial uniformity is slightly worse (only in the right bucket), while the velocity uniformity is significantly better. Specifically, the velocities uniformly distribute in ranges [-10, -6] and [-8, 10] for the two buckets respectively. As a result, the new histogram is expected to produce better prediction.

The overall velocity distribution for the dataset of Figure 4.1 is uniform. If the distribution is skewed, ignoring the velocities during partitioning is even more problematic. Consider, for example, Figure 4.2a where object velocities have only two values -10 and 10. Observe that, partitioning the spatial universe is useless because (i) the overall location distribution is already fairly uniform (i.e., 2 points in each cell), and (ii) for all possible partitions, the resulting buckets still have extremely skewed velocity distribution. In fact, in this case the best partition should be based entirely on the velocity dimension. Specifically, the first bucket (Figure 4.2b) contains all the points with negative velocities, whereas the second one (Figure 4.2c) involves those with positive ones. Notice that the resulting buckets have uniform location (one point from each cell) and velocity (all points have the same velocity) distributions.



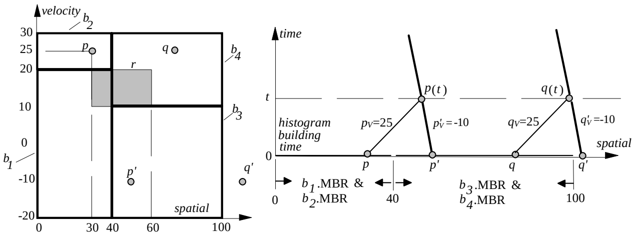
(a) Cell information

(b) Bucket 1 (c) Bucket 2

Figure 4.2: Skewed velocity distribution

Motivated by the above observations we propose spatio-temporal histograms (STHs) that partition on both velocity and location. A bucket b_j ($1 \le j \le h$) has spatial extents b_j.MBR, and velocity ranges b_j.VBR (where VBR stands for velocity bounding rectangle). In general, a m-dimensional dataset requires a 2m-dimensional STH. Figure 4.3a illustrates a STH with 4 buckets, assuming that the data space contains only one dimension (i.e., m=1). The MBR of b_1, for example, is [0, 40], while its VBR covers velocities [-20, 20] (i.e., the minimum and maximum velocity among all points in the bucket). Point p belongs to b_2, because its coordinate p_R=30 and velocity p_V=25 fall in b_2.MBR and b_2.VBR, respectively.

Moving intervals (hyper-rectangles in higher dimensions), on the other hand, are assigned according to the coordinates and velocities of their centroids. For instance, interval r (with spatial extent [30, 60] and velocity extent [10, 20]) is allocated to bucket b_4, which contains the coordinate 45 and the velocity 15 of its centroid. In addition to MBR and VBR, each bucket b_j also stores (i) the number $b_j.num$ of assigned objects, and (ii) the sum of velocity $b_j.LV_i$ and spatial (for hyper-rectangles) $b_j.L_i$ length of these objects along each dimension ($1 \le i \le m$). Similar to moving objects, the MBR of b_j also grows according to its VBR, and in the sequel we denote its MBR at future timestamp t as b_j.MBR(t). Such a STH can be constructed using any existing algorithm for conventional multi-dimensional histograms, by treating a m-dimensional moving object as a 2m-dimensional box.



(a) Histogram buckets (b) Update of point p

Figure 4.3: Updating the histogram

Assume that the histogram of Figure 4.3a is constructed at time 0, and point p updates its velocity (from 25 to -10) at some future time t (when its position is $p(t)$). After the change p does not belong to bucket b_2 any more, because its new velocity falls out of b_2.VBR [20, 30]. Furthermore, p cannot be inserted to the bucket that contains its current position $p(t)$ and velocity (-10), since the histogram is based on information at time 0 (meaning that future object positions are calculated based on the time elapsed with respect to time 0). To decide the new bucket for p, we must find its *surrogate point p'* at the histogram construction time (0), such that p' will reach the same position $p(t)$ with the updated velocity.

To illustrate this, consider Figure 4.2b, where the velocity of a point is represented as the slope of its trajectory. The projection point p' is the intersection of the spatial axis and the line with slope 10 that crosses $p(t)$, which spatially belongs to buckets b_3 and b_4, but only b_3.VBR covers the new velocity value[3]. To reflect the change, we should update $b_2.num$ (=$b_2.num$-1) and $b_2.LV$ (=$b_2.LV$-25), and modify b_3 accordingly ($b_3.num$+=1, $b_3.LV$–= 10). In some cases, the surrogate point may fall outside the universe, in which case the boundary bucket needs to be enlarged. As an

[3] Here we assume the bucket extents are disjoint, which holds for many histograms (e.g., the Minskew [APR99] deployed in our experiments), so that the bucket containing the projection point is unique. For histograms without this property, there may be multiple candidate buckets, in which case the final bucket can be selected randomly.

example, the MBR of bucket b_3 must be expanded to cover the surrogate point q' (of q) in Figure 4.3. It is worth mentioning that, the VBR of the selected bucket for expansion includes the updated velocity of q' (i.e., hence b_4 is not expanded).

Incrementally updating the histogram reduces the maintenance cost significantly. Whenever the system receives an object update, the new information is intercepted to modify the histogram accordingly. However, the uniformity (in buckets) may gradually deteriorate as the data (location and velocity) distributions vary. When the distribution changes significantly, the histogram needs to be re-built in order to ensure satisfactory estimation accuracy. A simple heuristic to ensure satisfactory estimation accuracy is to re-construct the histogram when the number of modifications reaches a certain threshold, as evaluated in the experiments.

4.2 Performing Estimation with Histograms

Given a query q, we estimate its selectivity by applying the uniform model in each bucket. Specifically, for a bucket b, the probability $b.Sel$ that an object (in b) satisfies q is estimated using the uniform model, treating b.MBR and b.VBR as the spatial and velocity spaces respectively. Thus, the expected number of objects in b satisfying q is $b.num \cdot b.Sel$, where $b.num$ is the total number of objects in b. As a result, the overall selectivity can be estimated by summing up the qualifying object number from every bucket and then dividing the sum by the dataset cardinality N, or more formally: $Sel=(\sum_{i=1\sim B} b_i.num \cdot b_i.Sel)/N$, where B is the total number of buckets in the histogram.

To reduce the estimation time, we aim at minimizing the number of buckets for which selectivity estimation is necessary (in our implementation we use a numerical approach, or specifically the trapezoid rule [PFTV02], to evaluate the integrals in the uniform models). Figure 4.4 shows the extents b_{1R}, b_{2R} of two buckets b_1, b_2 for point objects (arrows indicate the velocity directions of their VBRs). Consider the query (with q_{T-}=0), whose current extent q_R is the gray region, and the dashed rectangles represent the extents $b_{1R}(q_{T+})$, $b_{2R}(q_{T+})$, $q_R(q_{T+})$ of b_1, b_2, and q at time q_{T+}, respectively. Notice that selectivity estimation can be avoided for b_1, because its MBR does not intersect that of q during query interval q_T, indicating that none of the objects inside can possibly intersect q. Bucket b_2, on the other hand, must be considered (i.e., it is a qualifying bucket).

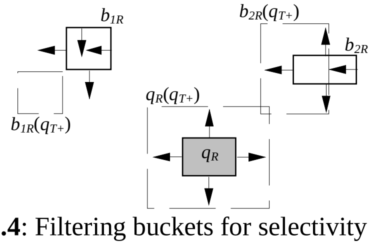


Figure 4.4: Filtering buckets for selectivity estimation

5. Experiments

This section experimentally evaluates the proposed methods. All the experiments were performed on a Pentium III 1Ghz CPU with 256 Mbytes memory. The first set of experiments demonstrates the correctness of the proposed formulae for uniform datasets. For this purpose we generated a dataset with 1 million points such that for each point (i) its location distributes uniformly in the 2D spatial universe $[0, 10000]^2$ (i.e., each axis has extent [0, 10000]) , and (ii) its velocity (on each dimension) is uniformly generated in [-50, 50]. A query q is a moving rectangle such that its extent q_R at the current time is a square with side length q_Rlen (e.g., if q_Rlen=1000, q_R covers 1% of the space) and its velocity extent $q_{Vi+}-q_{Vi-}$ (i.e., the difference of the velocities of the lower and upper boundaries) on each dimension i equals a constant q_Vlen (if q_Vlen=0, the extent of the query does not change with time). Query workloads consist of 200 queries with the same parameters q_Rlen, q_Vlen, and q_T (i.e., the query interval length): (i) the left boundary q_{Ri-} of each query q distributes uniformly in [0, 10000$-q_Rlen$] ($q_{Ri+}=q_{Ri-}+q_Rlen$), (ii) the velocity q_{Vi-} is generated uniformly in [-50, 50$-q_Vlen$], and (iii) q_{T-} follows a uniform distribution in [0, 100$-q_T$].

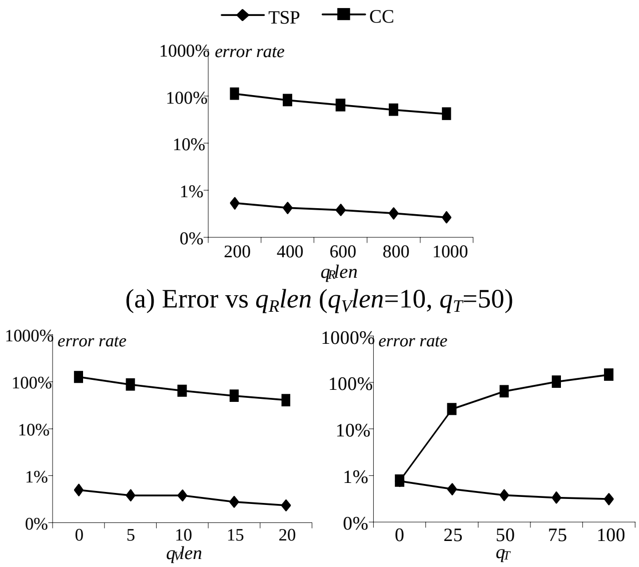
Let act_i and est_i be the actual and estimated numbers of objects retrieved from the i-th query ($1\le i\le 200$); then the workload error rate is computed as [APR99]:

$Err_{workload} = (\sum_{i=1\sim 200}|est_i-act_i|)/(\sum_{i=1\sim 200}act_i)$.

As mentioned earlier, in order to obtain est_i in our models we evaluate the integrals using the trapezoid rule approach [PFTV02], which partitions the integral range into 10 equal lengths and approximates the integral result with the area sum of a set of trapezoids. We compare the error rates of our model (denoted as TSP) with that of [CC02] (denoted as CC). Since the original CC only captures static queries over dynamic objects, we apply our reduction techniques to obtain the corresponding formulae for dynamic queries and rectangle objects.

Figure 5.1a shows the error rates of TSP and CC as a function of q_Rlen, fixing q_Vlen and q_T to 10 and 50, respectively. TSP yields extremely accurate prediction (with maximum error less than 1%), confirming the probabilistic correctness of our derivation. On the other hand, it is clear that CC leads to substantial errors (greater than 100%), indicating that the temporal intersection condition (introduced in Section 2.1) cannot be ignored. Observe that the error rates of both methods decrease when the query becomes larger, which is consistent with previous studies on spatial window selectivity [APR99, AN00].

Figure 5.1b shows the results with respect to various q_Vlen (from 0 to 20), fixing q_Rlen=600, q_T=50. Again our model is precise whereas CC produces around 100% error. In Figure 5.1c, we fix q_Rlen and q_Vlen, and increase q_T from 0 to 100. CC is accurate only when q_T=0 because, for timestamp queries, ignoring the temporal condition does not cause any error: if an object satisfies a query q, then the intersection intervals on all dimensions consist of a single timestamp q_{T-} (=q_{T+}). On the other hand, as predicted by equation (3-10), the error rate of CC increases fast with q_T. Experiments with uniform rectangles give similar results.



(a) Error vs q_Rlen (q_Vlen=10, q_T=50)

(b) Error vs q_Vlen (q_Rlen=600, q_T=50)

(c) Error rate vs q_T (q_Rlen=600, q_Vlen=10)

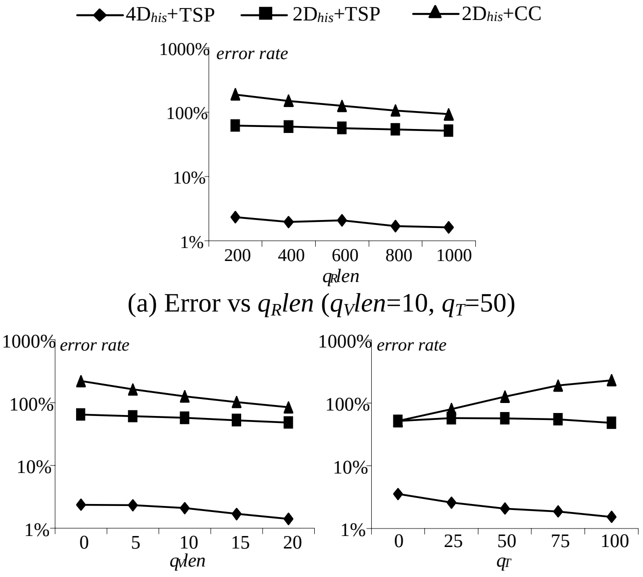
Figure 5.1: Accuracy for uniform data

Next we evaluate the proposed techniques for non-uniform datasets. Due to the lack of data for real moving objects, we generated synthetic datasets as follows. First, the location distribution is taken from real spatial datasets [Web] *CA* (with 2.2 million rectangles representing streets in California) and *LA* (containing 1.3 million rectangles corresponding to places in Los Angeles). Then, each rectangle r (in the static dataset) is associated with velocities such that on the i-th dimension ($1 \le i \le 2$), (i) the absolute value of r_{Vi-} (i.e., velocity of the lower boundary) follows a Zipf distribution (skew coefficient 0.8) in [0, $50-r_Vlen$], where r_Vlen is generated randomly in [0, 5] (i.e., objects can have different velocity extents), and (ii) r_{Vi-} has equal probability to be positive or negative. The creation of moving points is similar, except that (i) the current position of a point is the centroid of a rectangle in *CA* or *LA*, and (ii) r_Vlen is set to 0. In the sequel, we refer to the resulting datasets as CA_{rec} (CA_{pt}) and LA_{rec} (LA_{pt}) where the subscripts indicate rectangle (point) data.

We compare the error rates of three approaches. The first one, called $4D_{his}$+TSP, uses the proposed 4D STH (considering both location and velocities) and applies our uniform model in each bucket. The second one, $2D_{his}$+TSP, combines TSP with the histogram of [CC02] (i.e., where partitioning is based solely on location). The last method, referred to as $2D_{his}$+CC, corresponds to the solution proposed in [CC02], i.e., 2D histogram and the CC model. The resolutions (i.e., for the initial grid before applying MinSkew) are set to 15 and 50 for $4D_{his}$ and $2D_{his}$ respectively, so that we need only 4 (6) bits to represent a spatial boundary for $4D_{his}$ ($2D_{his}$). As a result, each bucket in $4D_{his}$ takes 8 bytes to store the associated information, while the size is 19 bytes for $2D_{his}$ (note that the velocities of $2D_{his}$ cannot be compressed as in $4D_{his}$). We allow 25k bytes memory for each histogram, and hence the number of buckets in $4D_{his}$ ($2D_{his}$) is set to 3000 (1200), respectively. After the cell initialization (the cost of which is the time of scanning the database), the construction time for $2D_{his}$ ($4D_{his}$) is 0.2 and 0.9 seconds respectively.

Figure 5.2a plots the error rates as a function of q_Rlen, fixing q_Vlen and q_T to their median values 10 and 50 respectively (CA_{pt} dataset). $4D_{his}$+TSP yields error below 5%, while the other methods are inaccurate. The fact that $2D_{his}$+TSP is considerably worse than $4D_{his}$+TSP indicates that the 4D histogram achieves much better uniformity in the buckets. In particular, since the velocities of each object follow skewed distribution, the velocity distribution (of each bucket) in $2D_{his}$ is also skewed. $2D_{his}$+CC is even less accurate than $2D_{his}$+TSP, due to the deficiency of CC.

Figure 5.2b illustrates the error rates with respect to various q_Vlen (q_Rlen=600, q_T=50). In Figure 5.2c, we fix q_Rlen and q_Vlen, and increase the query interval q_T up to 100. Note that the accuracy of both $4D_{his}$+TSP and $2D_{his}$+TSP improves with q_T (because the number of qualifying objects increases), while that of $2D_{his}$+CC actually deteriorates. This is not surprising because when q_T equals 0, $2D_{his}$+CC has the same performance (60% error) as $2D_{his}$+TSP, but as q_T increases so does the effect of the temporal intersection condition.



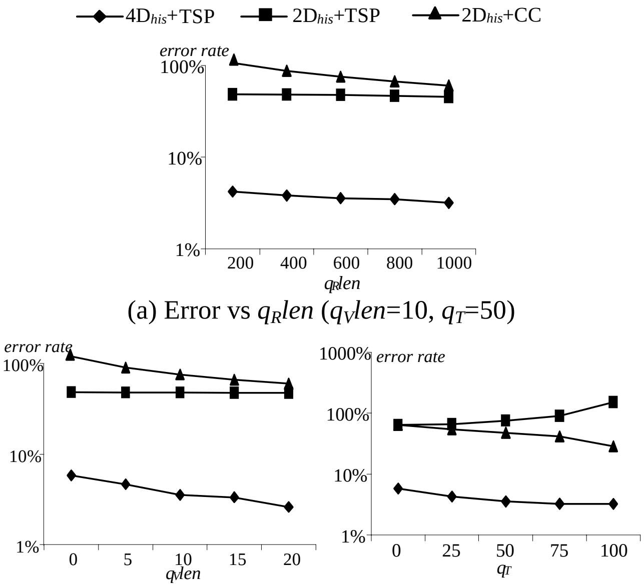
(a) Error vs q_Rlen (q_Vlen=10, q_T=50)

(b) Error vs q_Vlen (q_Rlen=600, q_T=50)

(c) Error rate vs q_T (q_Rlen=600, q_Vlen=10)

Figure 5.2: Accuracy for CA_{pt}

Figure 5.3 confirms the generality of the above observations by repeating the experiments using the LA_{pt} dataset. Figures 5.4 and 5.5 demonstrate the results of the same experiments with rectangles. The behavior of alternative approaches is very similar to those for point data. Specifically, 4D$_{his}$+TSP is accurate in all cases, while the other methods have significant errors. Notice that the error rates are slightly higher than those of points because rectangles have variable spatial and velocity extents, while each bucket records only average values.

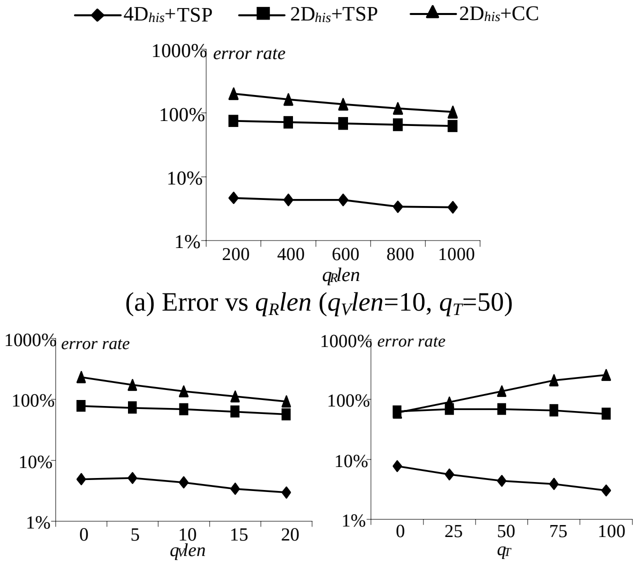


(a) Error vs q_Rlen (q_Vlen=10, q_T=50)

(b) Error vs q_Vlen (q_Rlen=600, q_T=50)

(c) Error rate vs q_T (q_Rlen=600, q_Vlen=10)

Figure 5.3: Accuracy for LA_{pt}

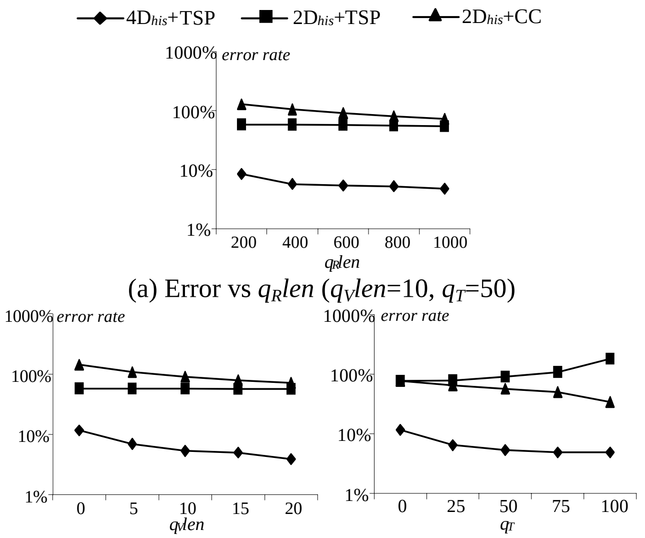


(a) Error vs q_Rlen (q_Vlen=10, q_T=50)

(b) Error vs q_Vlen (q_Rlen=600, q_T=50)

(c) Error rate vs q_T (q_Rlen=600, q_Vlen=10)

Figure 5.4: Accuracy for CA_{rec}



(a) Error vs q_Rlen (q_Vlen=10, q_T=50)

(b) Error vs q_Vlen (q_Rlen=600, q_T=50)

(c) Error rate vs q_T (q_Rlen=600, q_Vlen=10)

Figure 5.5: Accuracy for LA_{rec}

STHs can be incrementally maintained to capture object updates. To study the accuracy degradation with time, we created dynamic data as follows. The initial histogram is constructed at time 0, and at each of the subsequent 1000 timestamps, 10% of the objects update their velocities, such that the velocity changes are uniformly distributed in [-5, 5]. In this way, the data distribution will gradually become uniform. For each update the histogram is modified (if necessary) as described in Section 4.1.

Next, we perform window queries with standard parameters (i.e., q_Rlen=600, q_Vlen=10, q_T=50) every 100 timestamps and measure the average estimation error (using the histogram information at the query time). Figure 5.6 shows the results CA_{pt} and CA_{rec}. Notice that, the error rates increase very slowly (due to the distribution change) along with time for both cases. Even at the 1000-th timestamp, the error rates of our approach (i.e., 25% and 35% for point and rectangle data, respectively) are still significantly lower than those of the other approaches reported in Figures 5.2 and 5.4. As a result, the histogram needs re-building very infrequently (e.g., every 600 timestamps if maximum error 20% is allowed). The same observations hold for the *LA* dataset. Recall that 2D$_{his}$ requires re-construction at every timestamp.

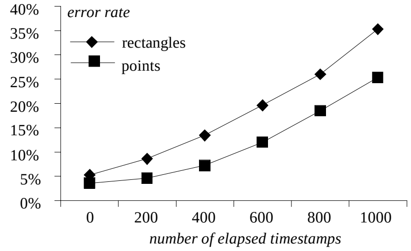
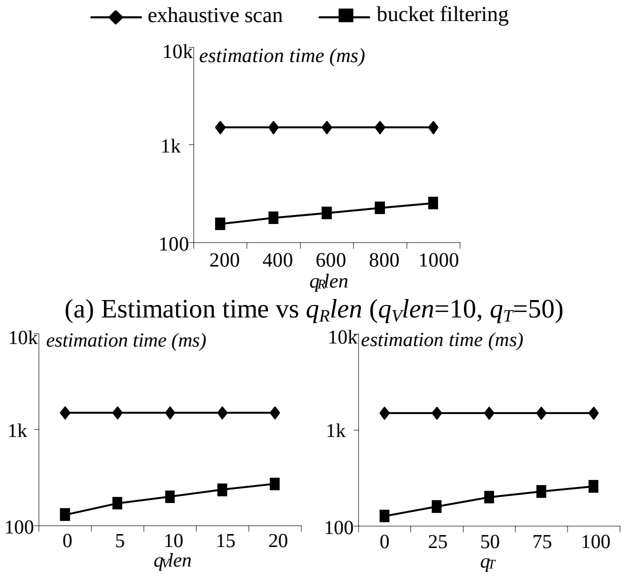


Figure 5.6: Accuracy degradation with time for 4D$_{his}$+TSP (*CA*, q_Rlen=600, q_Vlen=10, q_T=50)

The last set of experiments evaluates the time of obtaining estimate values. Figures 5.7a, b, c demonstrate the costs as functions of $q_R len$, $q_V len$, and q_T, respectively, comparing the bucket filtering technique with the brute force method (i.e., evaluating the uniform model for every bucket) on dataset CA_{pt}. It is clear that the cost of bucket filtering depends on the query parameters (the estimation time is less than 0.3 seconds in all cases), while that of the brute force is constant and significantly higher (around 1.5 seconds).



(a) Estimation time vs $q_R len$ ($q_V len$=10, q_T=50)

(b) Estimation time vs $q_V len$ ($q_R len$=600, q_T=50)

(c) Estimation time vs q_T ($q_R len$=600, $q_V len$=10)

Figure 5.7: Estimation costs for CA_{pt}

6. Conclusion

In spite of the importance of selectivity estimation in STDBMS, the existing approaches are not able to provide satisfactory prediction. This paper addresses the problem with a comprehensive study that covers all types of objects and query-object mobility combination. Particularly, we prove several important lemmas that reduce complex estimation problems into simple cases, and derive a model that (i) is able to capture the selectivity accurately, and (ii) is simpler and more flexible than the previous one. Furthermore, we present a new spatio-temporal histogram, which considers both locations and velocities for partitioning. Extensive experimentation confirms that the proposed techniques predict spatio-temporal query selectivity very accurately.

We believe this work provides a solid foundation for further analysis of spatio-temporal queries. For example, it will be interesting to investigate the selectivity of spatio-temporal join, which, given two sets of moving objects, retrieves all object pairs that satisfy some spatio-temporal predicate (e.g., intersection, distance). The selectivity is the number of retrieved pairs divided by the product of the input cardinalities. An even more challenging topic is to study the selectivity of complex queries involving several datasets. Furthermore, the proposed models may be extended to estimate the number of page accesses for answering queries using spatio-temporal access methods (e.g., TPR-trees).

Acknowledgements

This work was supported by grants HKUST 6081/01E and HKUST 6197/02E from Hong Kong RGC.

References

[AAE00] Agarwal, P.K., Arge, L., Erickson, J. Indexing Moving Points. *PODS*, 2000.

[AN00] Aboulnaga, A., Naughton, J. Accurate Estimation of the Cost of Spatial Queries. *ICDE*, 2000.

[APR99] Acharya, S., Poosala, V., Ramaswamy, S. Selectivity Estimation in Spatial Databases. *SIGMOD*, 1999.

[BF95] Belussi, A., Faloutsos, C. Estimating the Selectivity of Spatial Queries Using the Correlation's Fractal Dimension. *VLDB*, 1995.

[BKSS90] Beckmann, N., Kriegel, H., Schneider, R., Seeger, B. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. *SIGMOD*, 1990.

[CC02] Choi, Y., Chung, C. Selectivity Estimation for Spatio-Temporal Queries to Moving Objects. *SIGMOD*, 2002.

[KF93] Kamel, I., Faloutsos, C. On Packing R-Trees. *CIKM*, 1993.

[KGT99] Kollios, G., Gunopulos, D., Tsotras, V. On Indexing Mobile Objects. *PODS*, 1999.

[MPS99] Muthukrishnan, S., Poosala, V., Suel, T. On Rectangular Partitionings in Two Dimensions: Algorithms, Complexity, and Applications. *ICDT*, 1999.

[PFTV02] Press, W., Flannery, B., Teukolsky, S., Vetterling, W. Numerical Recipes in C++ (second edition). *Cambridge University Press*, 2002.

[PJT99] Pfoser, D., Jensen, C, Theodoridis, Y. Novel Approaches to the Indexing of Moving Object Trajectories. *VLDB*, 2000.

[SAE02] Sun, C., Agrawal, D., El Abbadi, A. Exploring Spatial Datasets with Histograms. *ICDE*, 2002.

[SJ02] Saltenis, S., Jensen, C.S. Indexing of Moving Objects for Location-Based Services. *ICDE*, 2002.

[SJLL00] Saltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A. Indexing the Positions of Continuously Moving Objects. *SIGMOD*, 2000.

[TP01] Tao, Y., Papadias, D. The MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries. *VLDB*, 2001.

[TP02] Tao, Y., Papadias, D. Time-Parameterized Queries in Spatio-Temporal Databases. *SIGMOD*, 2002.

[Web] Http://dias.cti.gr/~ytheod/research/datasets/spatial.html