

Selectivity Estimation for Spatio-Temporal Queries to Moving Objects

Yong-Jin Choi

omni@islab.kaist.ac.kr

Chin-Wan Chung

chungcw@cs.kaist.ac.kr

Division of Computer Science
Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology
Taejeon, Korea

ABSTRACT

A query optimizer requires selectivity estimation of a query to choose the most efficient access plan. An effective method of selectivity estimation for the future locations of moving objects has not yet been proposed. Existing methods for spatial selectivity estimation do not accurately estimate the selectivity of a query to moving objects, because they do not consider the future locations of moving objects, which change continuously as time passes.

In this paper, we propose an effective method for spatio-temporal selectivity estimation to solve this problem. We present analytical formulas which accurately calculate the selectivity of a spatio-temporal query as a function of spatio-temporal information. Extensive experimental results show that our proposed method accurately estimates the selectivity over various queries to spatio-temporal data combining real-life spatial data and synthetic temporal data. When Tiger/lines is used as real-life spatial data, the application of an existing method for spatial selectivity estimation to the estimation of the selectivity of a query to moving objects has the average error ratio from 14% to 85%, whereas our method for spatio-temporal selectivity estimation has the average error ratio from 9% to 23%.

1. INTRODUCTION

The development of technologies such as wireless communication systems and global positioning systems (GPS) shows that there can be many applications for spatio-temporal databases. In recent years, spatio-temporal databases have been studied intensively. Most research has progressed in modeling [3, 9] and in indexing [2, 5, 7, 8, 11]. In general, access methods to moving objects are considered for two kinds of selection queries: one for historical positions of moving objects and the other for future positions of moving objects. This paper is related to the latter, which is referred

to as a future query [9].

An example of a future query is as follows: “retrieve all airplanes which will be inside a query rectangle 10 minutes from now.” Airplanes can be represented as moving objects that move as time passes. Sistla et al. [9] proposed a data model that can manage the future positions of an object with the last update information: the spatial position, the velocity, and the last update time. The model reduces the number of updates, because the object is represented by a time function whose value continuously changes as time passes. Recently, various studies [2, 5, 8, 12] are based on this model. Our work is also based on this model.

In order to process the selection query efficiently, an accurate estimation of the selectivity is required. The selectivity is defined as the ratio of the number of data in the query result to the total number of data in the database. The query optimizer chooses an efficient execution plan among all the possible plans by estimating the cost of each plan. The accuracy of the selectivity estimation significantly affects the selection of an efficient plan. Existing methods for spatial selectivity estimation do not accurately estimate the selectivity of the query for the future positions of moving objects, because they handle the spatial positions of objects only with the current time, as shown in Figure 1.

Figure 1(a) shows that moving objects will move from positions of the current time to positions of the future time. Figure 1(b) shows the positions of moving objects and the query rectangle of the current time. The moving objects *O3* and *O4* exist in the query rectangle. Figure 1(c) shows the positions of moving objects and the query rectangle of the future time. The moving objects *O1*, *O2*, and *O4* exist in the query rectangle. Although the query rectangles are the same, the query results are different depending on the given time. Traditional methods of spatial selectivity estimation accurately estimate the selectivity of the query in the case of Figure 1(b); however, in the case of Figure 1(c), they cannot, because they do not consider the future positions of the objects.

In this paper, with the motivation from the above problem, we propose a spatio-temporal histogram method for selectivity estimation that can accurately estimate the selectivity of a future query to moving objects. We present analytical formulas which calculate the selectivity of a spatio-temporal query as a function of spatio-temporal information. The analytical formulas consider the future positions

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACM SIGMOD '2002 June 4-6, Madison, Wisconsin, USA
Copyright 2002 ACM 1-58113-497-5/02/06 ...\$5.00.

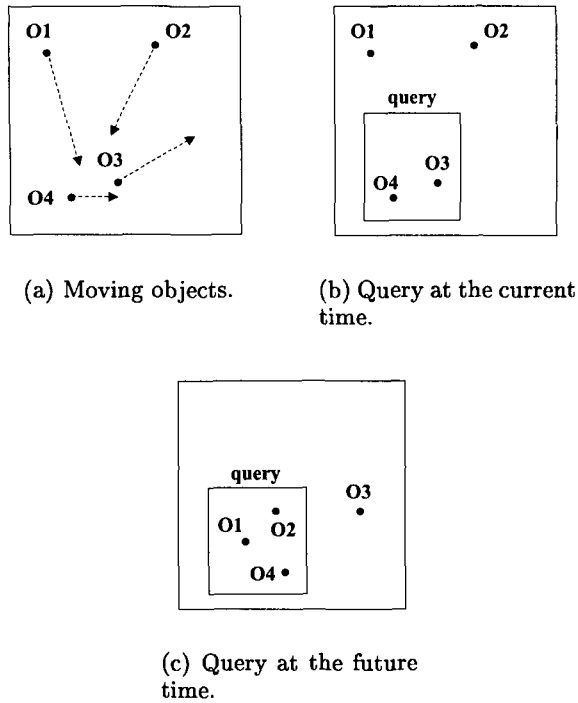


Figure 1: Moving objects and queries.

of moving objects to accurately estimate the selectivity for the query. Also, our analytical formulas provide an advantage that these formulas can be basically used by various cost models for moving objects.

Experimental results show that our selectivity estimation provides accurate estimation over various queries for synthetic moving objects. However, for supporting a more realistic experimental environment, we used real-life spatial data, Tiger/lines [6] and Sequoia [10] popularly used in spatial database research, to generate moving objects. Since there has been no previous work that specifically addresses the selectivity estimation method to moving objects, we compared our proposed method with an existing spatial selectivity estimation method which uses the spatial histogram. While the use of the existing method for the selectivity estimation of a spatio-temporal query has the average error ratio from 14% to 85%, our method for spatio-temporal selectivity estimation has the average error ratio from 9% to 23%. In addition, we studied the impact of the size of the histogram. As the histogram size increases, the accuracy of the selectivity estimation using the spatio-temporal histogram increases. However, the accuracy of the selectivity estimation using the spatial histogram that does not consider the future locations of moving objects is not significantly affected by the histogram size.

Our contributions are summarized as follows:

- We propose an effective method of the selectivity estimation for spatio-temporal queries. To our knowledge, the proposed method is the first work specifically addressing the selectivity estimation method for moving objects.
- We present a practical method to maintain the spatio-

temporal histogram. The method applies the property of moving objects to the previous research work [4]. While our spatio-temporal histogram is constructed using the previous method for spatial selectivity estimation [1], we consider a velocity bounding rectangle to each bucket of the spatio-temporal histogram to deal with the movements of moving objects.

- We provide extensive experimental results using various queries. For a more realistic experimental environment, we use real-life spatial data to generate moving objects. The experimental results show that our proposed method achieves a considerable accuracy.

The paper is organized as follows. In Section 2, we describe the related work, which consists of spatio-temporal databases, the selectivity estimation in spatial databases, and the histogram update. In Section 3, we explain how to estimate the selectivity of a query to moving objects. In Section 4, we show our experimental results and discuss them in detail. Finally, conclusions are made in Section 5.

2. RELATED WORK

First, we briefly describe spatio-temporal databases and explain the selectivity estimation in spatial databases, and then explain the histogram update.

Spatio-temporal databases manage objects that move as time passes. With new developments in technology, many applications have been proposed to deal with the problem of movement. These applications require spatio-temporal database management systems with modeling [3, 9] and query processing to moving objects [2, 5, 7, 8, 11, 12]. Sistla et al. [9] proposed a data model to represent moving objects. This model introduces a dynamic object that changes continuously its position as time passes. Many studies for query processing are based on this model [9].

Our work is aimed at an accurate estimation of the selectivity of a query to moving objects. The query optimizer chooses the most efficient plan by using the estimation for the selectivity of a query. The accuracy of the selectivity estimation significantly affects the selection of an efficient plan. However, existing selectivity estimations for spatial objects do not accurately estimate the selectivity to moving objects. Besides, to our knowledge, there has not been a proposed method designed with consideration of motion. Our selectivity estimation is related to the studies described below.

We use the histogram method for the selectivity estimation. The histogram is one of the most popular selectivity estimation methods, because it approximates any data distribution and requires reasonably small storage with low error rates [1]. The histogram is a set of buckets. A bucket consists of bounding values to cover objects and the number of the objects that fall in the bounding values. For example, a spatial bucket consists of a spatial bounding rectangle and the number of objects in the spatial bounding rectangle. However, a spatio-temporal bucket additionally requires a velocity bounding rectangle to cover the objects. Therefore, our spatio-temporal bucket consists of a spatial bounding rectangle, a velocity bounding rectangle, and the number of objects in those bounding rectangles. This concept of a spatio-temporal bucket is basically the same as the concept of the time-parameterized bounding rectangle of the TPR-tree [8].

We use the MinSkew algorithm [1] to construct a spatio-temporal histogram. The MinSkew algorithm constructs a spatial histogram to minimize the spatial-skew of spatial objects. The spatial histogram estimates the accurate selectivity for spatial objects with small storage requirement, compared to other work [1].

The positions of moving objects are updated very frequently. If new update information is not reflected to the histogram, the query optimizer may choose an inefficient plan because an inaccurate estimation of selectivity is used. However, it is clearly impractical to construct a new histogram whenever moving objects are updated. The histogram is not required to be updated from the raw objects, but it can be updated from sample data [4]. In spatio-temporal databases, the histogram can be maintained practically by using sample data. The sample data include only information related to the histogram construction. Considering the moving object, the information consists of the identification, the last update time, the spatial position of the time, and the velocity. The construction of a new histogram from the sample data reduces I/Os overhead remarkably, compared to the overhead of the construction of a new histogram from all of the data. Therefore, we construct the histogram only from the sample data of moving objects.

3. SELECTIVITY ESTIMATION IN SPATIO-TEMPORAL DATABASES

This section explains selectivity estimation in spatio-temporal databases. First, we describe our spatio-temporal histogram structure and the histogram construction algorithm. We explain how to estimate the selectivity of a spatio-temporal query to 1D moving objects. Then, we explain how to estimate the selectivity of a spatio-temporal query to 2D moving objects using the 1D selectivity estimation.

3.1 Histogram Construction

This section describes our spatio-temporal histogram structure to 2D moving objects. Table 1 presents the symbols used throughout the paper to describe the proposed algorithms.

Our spatio-temporal histogram $H_{B_{st}}$ is a set of buckets like other existing histogram structures. However, our spatio-temporal bucket additionally requires a velocity bounding rectangle. Therefore, the spatio-temporal bucket B_{st} consists of a spatial bounding rectangle SB , a velocity bounding rectangle VB , and the number, n , of objects in those bounding rectangles. $H_{B_{st}}$ is newly created at the histogram update time t^u . A query Q consists of a spatial rectangle QB and a time interval t , where the low value t^l of t is greater than or equal to the current time. The sample factor sf is defined as the ratio of the total number of moving objects to the number of sample moving objects. Subscripts 1 and 2 of SB , VB , and QB represent each dimension in 2D space.

Figure 2 shows the construction algorithm of the histogram in the spatio-temporal databases.

The constructSTH algorithm creates the spatio-temporal histogram $H_{B_{st}}$ using the MinSkew algorithm [1]. $H_{B_{st}}$ is constructed only by the sample moving objects. The MinSkew algorithm uses a spatial grid, SG of cells. Each cell relates to its spatial density which indicates the number of objects in the cell. SG is similar to the grid structure of

Algorithm constructSTH($t^u, SM, N_{B_{st}}$)
Input: t^u , histogram update time;
 SM , a set of sample moving objects;
 $N_{B_{st}}$, total number of B_{st} in $H_{B_{st}}$;
Output: $H_{B_{st}}$, spatio-temporal histogram

for each moving object m in SM ,
 calculate the spatial position x , located at t^u , of m .
 find a cell of SG such that the cell contains x .
 increase the spatial density of the cell.
 adjust the velocity bounding rectangle of the cell
 to contain the velocity of m .

execute the bucket creation module of the MinSkew algorithm using SG
to construct $N_{B_{st}}$ spatial bounding rectangles of the $H_{B_{st}}$.

for each cell of SG ,
 find B_{st} whose SB contains the cell.
 increase $B_{st}.n$ with the spatial density of the cell.
 adjust $B_{st}.VB$ to contain the velocity bounding rectangle of the cell.

Figure 2: Algorithm constructSTH for moving objects.

[1]; however our SG also requires a velocity bounding rectangle for each cell. The velocity bounding rectangle in the cell bounds the velocities of objects in the cell. The construction process of the spatio-temporal histogram is similar to that of the original MinSkew algorithm for spatial data. There are a few differences, however, as follows: To assign all the sample moving objects to SG , the spatial locations of the sample moving objects should be calculated at the histogram update time and the velocities of the objects should be adjusted at the velocity bounding rectangle of the corresponding cell. After executing the MinSkew algorithm with SG , the velocity bounding rectangle of each cell of SG should be adjusted at each $B_{st}.VB$ of $H_{B_{st}}$.

We construct the whole spatio-temporal histogram periodically. When an object is modified, previous techniques for the histogram update can update the part of the histogram related to the modified object without constructing the whole histogram. However, our spatio-temporal histogram structure to moving objects cannot only update the part of the histogram related to the modified object, because the spatial positions of the non-modified objects implicitly change as time passes. We construct the whole spatio-temporal histogram per time unit. In Section 4, we demonstrate how this update strategy for the spatio-temporal histogram can be possible.

The next section describes how to estimate the selectivity of a query to 1D moving objects. We then explain how the 1D selectivity estimation can be easily extended to the 2D selectivity estimation to 2D moving objects.

3.2 One-Dimensional Selectivity Estimation

We describe the selectivity estimation between a spatio-temporal bucket and a query to 1D moving objects. There are three methods to estimate the selectivity of the query. We explain each in detail. Since this section deals with only 1D moving objects, we illustrate the 1D selectivity estimation corresponding to the first dimension of 2D moving objects. We use x , v , and a rather than x_1 , v_1 and a_1 , respectively, for notational convenience.

3.2.1 Overview of One-Dimensional Selectivity Estimation

Table 1: Symbol description.

Symbol	Description
$I(I^l, I^h)$	interval: $I^l \leq I^h$; I^l , low value of I ; I^h , high value of I
t^u	histogram update time
$H_{B_{st}}$	spatio-temporal histogram: a set of B_{st}
$B_{st}(n, SB, VB)$	spatio-temporal bucket: n , number of moving objects in SB and VB
$SB(x_1, x_2)$	spatial bounding rectangle: x_1, x_2 are intervals
$VB(v_1, v_2)$	velocity bounding rectangle: v_1, v_2 are intervals
$N_{B_{st}}$	total number of B_{st} in $H_{B_{st}}$
$Q(QB, t)$	query: time interval t
$QB(a_1, a_2)$	spatial rectangle of query: a_1, a_2 are intervals
sf	sample factor = number of objects / number of sample objects

The 1D selectivity estimation is classified into three kinds of estimation methods. Figure 3 shows relationships between five objects bounded by a spatio-temporal bucket of time t^u and a query $Q(a, t)$. As shown in Figure 3, the objects move within the range of thick lines as time passes. The number of objects passing the query is equal to the number of objects that pass the sides of the query. When a moving object passes two sides of the query, the moving object should be checked from one side only. In this case, we only consider one side of the query that the moving object passes first. Specifically, the number of moving objects passing the query is equal to the number of moving objects that passes three sides of the query: the base side, the left side, and the right side. If we check three sides of the query, we can consider all the movements of moving objects that intersect with the query. The moving object $O3$ passes the left side of the query. The moving object $O4$ passes both the base side and the right side. However, it is estimated by using the base side of the query, because the moving object $O4$ passes the base side of the query first. The moving object $O2$ passes the right side of the query.

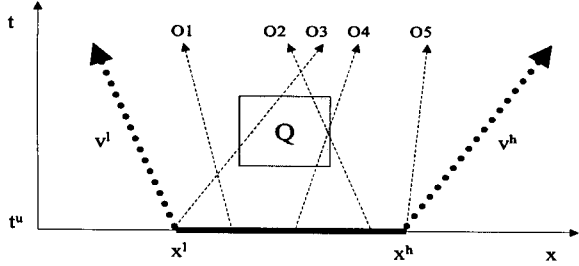


Figure 3: Spatio-temporal bucket and query to 1D moving objects.

Like the uniform assumption in each bucket of the histogram proposed by [1, 4], we assume a simple and general uniform distribution in each spatio-temporal bucket of the spatio-temporal histogram. That is, the spatial positions of moving objects are distributed uniformly in the spatial bounding interval x and the velocities of moving objects are distributed uniformly in the velocity bounding interval v .

We describe several definitions for explaining the proposed algorithms. The following definitions describe four relationships between moving objects bounded by the velocity bounding interval and the query.

Definition 1. Let t^u be the histogram update time, v the velocity bounding interval of the spatio-temporal bucket, a

the spatial interval of the query, and t the time interval of the query. CQ is the maximum spatial interval at t^u of moving objects that can pass the query. CQB , CQL , and CQR are the maximum spatial intervals at t^u of moving objects that can pass the base side of the query, the left side of the query, and the right side of the query first, respectively. \square

For example, let us consider a query and moving objects limited by a velocity bounding interval $v[v^l < 0, v^h > 0]$. As shown in Figure 4, for CQ , the thick line I depicts the maximum spatial interval of moving objects that can pass the query.

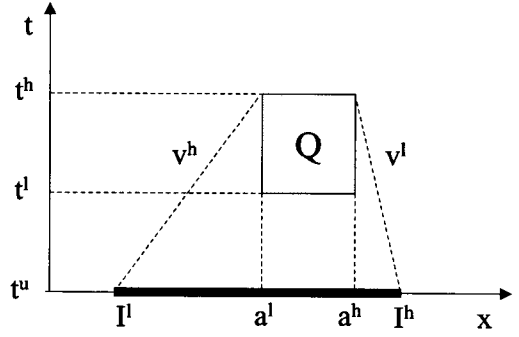


Figure 4: Maximum spatial interval I derived by CQ .

The following are formulas derived for the above definition.

- $CQ(t^u, v, a, t)$ has an interval I as follows:

$$\begin{aligned}
 I^l &= \begin{cases} a^l - (t^l - t^u)v^h & \text{if } v^h < 0 \\ a^l - (t^h - t^u)v^h & \text{otherwise} \end{cases} \\
 I^h &= \begin{cases} a^h - (t^l - t^u)v^l & \text{if } 0 < v^l \\ a^h - (t^h - t^u)v^l & \text{otherwise} \end{cases}
 \end{aligned}$$

- $CQB(t^u, v, a, t)$ has an interval I as follows:

$$\begin{aligned}
 I^l &= a^l - (t^l - t^u)v^h \\
 I^h &= a^h - (t^l - t^u)v^l
 \end{aligned}$$

- $CQL(t^u, v, a, t)$ has an interval I as follows:

$$\begin{aligned}
 I &= \begin{cases} \emptyset & \text{if } v^h < 0 \\ [I^l, I^h] & \text{otherwise} \end{cases} \\
 I^l &= a^l - (t^h - t^u)v^h
 \end{aligned}$$

$$I^h = \begin{cases} a^l & \text{if } v^l < 0 \\ a^l - (t^l - t^u)v^l & \text{otherwise} \end{cases}$$

- $CQR(t^u, v, a, t)$ has an interval I as follows:

$$I = \begin{cases} \emptyset & \text{if } 0 < v^l \\ [I^l, I^h] & \text{otherwise} \end{cases}$$

$$I^l = \begin{cases} a^h - (t^l - t^u)v^h & \text{if } v^h < 0 \\ a^h & \text{otherwise} \end{cases}$$

$$I^h = a^h - (t^h - t^u)v^l$$

Let us briefly consider the estimation of 1D selectivity of a query on a single spatio-temporal bucket with n moving objects bounded by x and v . A formula $(t^l - t^u)(v^h - v^l)$ indicates the length of all possible positions at t^l of a moving object that starts from a point in x . This is illustrated in Figure 5. The length corresponds to all the movements of the moving objects. Then $(x^h - x^l)(t^l - t^u)(v^h - v^l)$ corresponds to all the movements of all the moving objects that start from x . To consider all the movements of moving objects that pass the query, we present a function $L(s)$ as illustrated in Figure 6. $L(s)$ is the length of a sub-interval of x . Moving objects that start from the sub-interval corresponding to $L(s)$ should pass both the point (s, t^l) and the query. $[s_1, s_2]$ is the interval intersecting with the movements that can pass the query. Then, $\int_{s_1}^{s_2} L(s)ds$ corresponds to all the movements of moving objects that start from a subset of x and pass the query. Therefore, the selectivity is estimated as follows (for, $v^l \neq v^h \wedge t^u \neq t^l$):

$$n \frac{\int_{s_1}^{s_2} L(s)ds}{(x^h - x^l)(t^l - t^u)(v^h - v^l)} \quad (1)$$

In the case of $v^l = v^h$ or $t^u = t^l$, we should consider a different formula for (1), because the formula $(x^h - x^l)(t^l - t^u)(v^h - v^l)$ becomes 0. More details are given in Section 3.2.4.

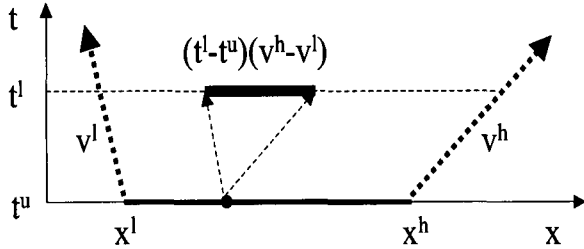


Figure 5: All possible positions at t^l of a moving object.

Figure 7 shows the calculation of $\int_{s_1}^{s_2} L(s)ds$ to 1D moving objects. We deal with only the formula $\int_{s_1}^{s_2} L(s)ds$ until Section 3.2.4, because n and $(x^h - x^l)(t^l - t^u)(v^h - v^l)$ of formula (1) are considered when we make an extension for the 2D selectivity estimation. As shown in Figure 7, the E algorithm summates the estimated result of each case after checking whether moving objects that can pass each of the three sides of the query exist. The EQB, the EQL, and the EQR are the estimation algorithms using the base side of the query, the left side of the query, and the right side of the query, respectively. Because the EQL and the EQR are symmetrically calculated, we only describe the EQB and the EQL in order.

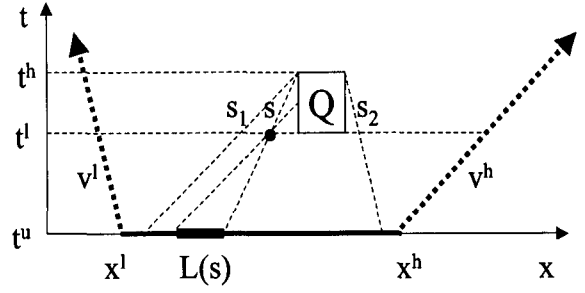


Figure 6: $L(s)$.

Procedure $E(t^u, x, v, a, t)$

Input: t^u , histogram update time; x , spatial bounding interval; v , velocity bounding interval; a , spatial interval of query; t , time interval of query

Output: sum , the value of the integration of $L(s)$

$sum \leftarrow 0$.

// computation using the base side

$I \leftarrow CQB(t^u, v, a, t)$.

$I \leftarrow x \cap I$.

if $I \neq \emptyset$, $sum \leftarrow sum + EQB(t^u, x, v, a, t)$.

// computation using the left side

$I \leftarrow CQL(t^u, v, a, t)$.

$I \leftarrow x \cap I$.

if $I \neq \emptyset \wedge t^l \neq t^h$, $sum \leftarrow sum + EQL(t^u, x, v, a, t, I)$.

// computation using the right side

$I \leftarrow CQR(t^u, v, a, t)$.

$I \leftarrow x \cap I$.

if $I \neq \emptyset \wedge t^l \neq t^h$, $sum \leftarrow sum + EQR(t^u, x, v, a, t, I)$.

Figure 7: Algorithm E.

3.2.2 Estimation Using Query Base Side

We consider only moving objects that first pass the base side of the query. As shown in Figure 8, there exists three possible cases between moving objects and the base side of the query. The function $L(s)$ has different formulas for three sub-intervals of the base side (here, we consider the base side of the query can be the same as I).

To find three different formulas of the $L(s)$, we introduce a function PESB to partition the base side. PESB partitions the interval of x that is extended at time t^l into three sub-intervals, as shown in Figure 8. Let t^u be the histogram update time, x the spatial bounding interval of a spatio-temporal bucket, v the velocity bounding interval of the spatio-temporal bucket, and t the time interval of the query. From Figure 8, an interval list $LI = \langle I_1, I_2, I_3 \rangle$ produced by $PESB(t^u, x, v, t)$ can be determined as follows:

$$I_1^l = x^l + (t^l - t^u)v^l$$

$$I_1^h = \begin{cases} x^h + (t^l - t^u)v^l & \text{if } (x^h - x^l) < (t^l - t^u)(v^h - v^l) \\ x^l + (t^l - t^u)v^h & \text{otherwise} \end{cases}$$

$$I_2 = I_1^h$$

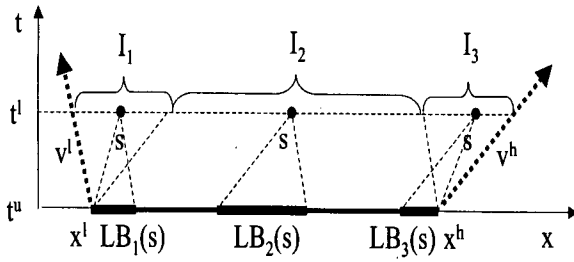


Figure 8: Three sub-intervals for $L(s)$.

$$I_2^h = \begin{cases} x^l + (t^l - t^u)v^h & \text{if } (x^h - x^l) < (t^l - t^u)(v^h - v^l) \\ x^h + (t^l - t^u)v^l & \text{otherwise} \end{cases}$$

$$I_3^l = I_2^h$$

$$I_3^h = x^h + (t^l - t^u)v^h$$

Now, we consider the calculation of $\int_{a^l}^{a^h} L(s)ds$. For three sub-intervals of the base side, $L(s)$ has different formulas: $LB_1(s)$, $LB_2(s)$, and $LB_3(s)$. As depicted in Figure 9, $LB_1(a^l)$ is $(a^l - (t^l - t^u)v^l) - x^l$. And, $LB_1(a^l + \lambda) = (a^l - (t^l - t^u)v^l) - x^l + \lambda$ for $0 \leq \lambda \leq a^h - a^l$. Therefore, the integration of $LB_1(s)$ using the base side of the query is $\int_0^{a^h - a^l} ((a^l - (t^l - t^u)v^l) - x^l + s)ds$. $LB_2(s)$ and $LB_3(s)$ are similar to $LB_1(s)$.

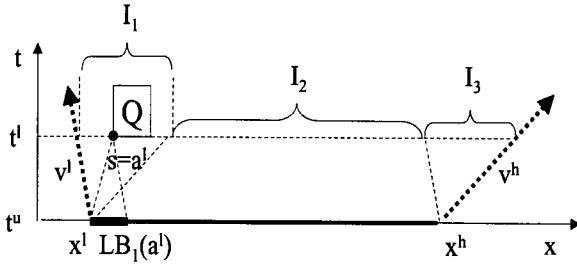


Figure 9: Calculation of $\int_{a^l}^{a^h} LB_1(s)ds$.

The procedure in Figure 10 shows a simple algorithm to calculate the estimation using the base side of the query to moving objects. This algorithm is called the EQB algorithm. After producing three intervals I_1 , I_2 , and I_3 by PESB, we can obtain common intervals QI_1 , QI_2 , and QI_3 , between a and I_1 , I_2 , and I_3 , respectively. EQB summates the result of each case. The following equations provide the integration of $L(s)$ for QI_1 , QI_2 , and QI_3 , respectively.

- $EQB_1(t^u, x, v, t, QI_1)$:

$$\int_0^{QI_1^h - QI_1^l} ((QI_1^l - (t^l - t^u)v^l) - x^l + s)ds$$

- $EQB_2(t^u, x, v, t, QI_2)$:

$$\begin{aligned} & \int_0^{QI_2^h - QI_2^l} (x^h - x^l)ds && \text{if } (x^h - x^l) < (t^l - t^u)(v^h - v^l) \\ & \int_0^{QI_2^h - QI_2^l} (t^l - t^u)(v^h - v^l)ds && \text{otherwise} \end{aligned}$$

- $EQB_3(t^u, x, v, t, QI_3)$:

$$\int_0^{QI_3^h - QI_3^l} (x^h - (QI_3^l - (t^l - t^u)v^h) - s)ds$$

Procedure EQB(t^u, x, v, a, t)

Input: t^u , histogram update time; x , spatial bounding interval; v , velocity bounding interval; a , spatial interval of query; t , time interval of query

Output: sum , the value of the integration of $L(s)$

$sum \leftarrow 0$.

$LI \leftarrow PESB(t^u, x, v, t)$.

for $i \leftarrow 1$ **to** 3 **do**,

$QI \leftarrow a \cap LI.I_i$.

if $QI \neq \emptyset$, $sum \leftarrow sum + EQB_i(t^u, x, v, t, QI)$.

Figure 10: Algorithm EQB.

3.2.3 Estimation Using Query Left Side

We consider only moving objects that first pass the left side of the query. Since the velocities of moving objects are limited between 0 and v^h , we consider only those moving objects.

As shown in Figure 11, let us consider a spatial bounding interval I at t^u , where $I^l = a^l - (t^h - t^u)v^h$ and $I^h \leq a^l$. We present a function $LL(s)$ that represents the length of a sub-interval of I satisfying the following two conditions. First, moving objects that start from the sub-interval should pass both the point (s, t^l) and the left side of the query. Second, the velocities of moving objects should be limited between 0 and v^h , where $0 < v^h$. As seen in Figure 11, the function $LL(s)$ has different formulas for two sub-intervals between s^l and s^h . s^l is the s value of a point (s, t^l) on a line that connects two points: (I^l, t^u) and (a^l, t^h) . s^m is the s value of a point (s, t^l) on a line that connects two points: (I^h, t^u) and (a^l, t^h) . s^h is the minimum value between $I^h + (t^l - t^u)v^h$ and a^l .

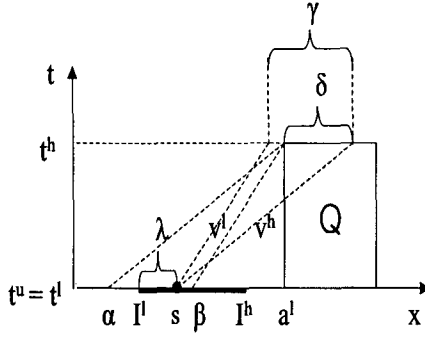
The integration of $LL(s)$ between s^l and s^h is the sum of the integration of $LL_1(s)$ between s^l and s^m and the integration of $LL_2(s)$ between s^m and s^h . For $s^l \leq s < s^m$, consider $s = s^l + \lambda$. As seen in Figure 11, since $LL_1(s^l + \lambda) = (\frac{t^l - t^u}{t^h - t^l})\lambda$, $\int_{s^l}^{s^m} LL_1(s)ds = \int_{s^l}^{s^m} (\frac{t^l - t^u}{t^h - t^l})(s - s^l)ds = \int_0^{s^m - s^l} (\frac{t^l - t^u}{t^h - t^l})sds$. Similarly, consider $s = s^l + \lambda$ for $s^m \leq s \leq s^h$. Since $LL_2(s^l + \lambda) = I^h - (a^l - (t^h - t^u)v^h) - \lambda$, $\int_{s^m}^{s^h} LL_2(s)ds = \int_{s^m}^{s^h} (I^h - (a^l - (t^h - t^u)v^h) - s)ds$. Since $LL_1(s^l + \lambda)$ and $LL_2(s^l + \lambda)$, $LL_1(s) = (\frac{t^l - t^u}{t^h - t^l})s$ and $LL_2(s) = I^h - (a^l - (t^h - t^u)v^h) - s$. Therefore, the integration of $LL(s)$ between s^l and s^h , called EQLS(t^u, I^h, v^h, a, t), is given by:

- $EQLS(t^u, I^h, v^h, a, t)$:

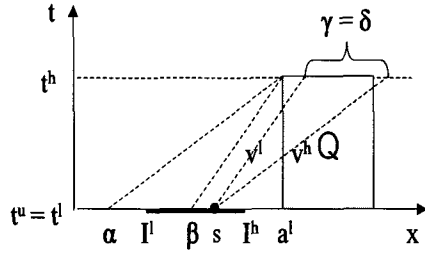
$$\int_0^{s^m - s^l} (\frac{t^l - t^u}{t^h - t^l})sds + \int_{s^m}^{s^h} (I^h - (a^l - (t^h - t^u)v^h) - s)ds$$

where $s^l = a^l - (t^h - t^l)v^h$, $s^m - s^l = (t^h - t^l)(v^h - \frac{a^l - I^h}{t^h - t^u})$, $s^h - s^l = \min(I^h - (a^l - (t^h - t^u)v^h), (t^h - t^l)v^h)$.

We develop a general algorithm called EQL based on the EQLS function as shown in Figure 12. We provide a geometrical explanation of the computation for the EQL algorithm(see Figure 13). If J^l , corresponding to the input J of the EQL algorithm, is not equal to $a^l - (t^h - t^u)v^h$, we subtract the result of the EQLS applied to a spatial interval



(a) $PL_1(s)$.



(b) $PL_2(s)$.

Figure 14: $P(s)$ using the left side of the query ($t^u = t^l$).

Figure 7. Like the E algorithm, the ETu algorithm summarizes three estimation results using the base side of the query, the left side of the query, and the right side of the query. The function $IL(I)$ has the length of an interval I . The EQRTu is the estimation algorithm using the right side the query in the condition of $t^u = t^l$.

3.3 Two-Dimensional Selectivity Estimation

The 1D selectivity estimation is easily extended to the 2D selectivity estimation. The estimation of one dimension is independent of that of the other. Therefore, the selectivity estimation to 2D moving objects becomes the product of two individual 1D estimations. Figure 16 shows the selectivity estimation algorithm to 2D moving objects. The estimation is calculated for each B_{st} . If SB intersects with the result of CQ for QB , the selectivity is estimated for the query and the corresponding B_{st} . In Figure 16, 1(a) shows the estimation method for the first dimension. In the condition of $v^l = v^h$, the estimation is only for moving objects with a constant velocity and corresponds to the ratio of the subspatial interval length of x_1 intersecting with the result of CQ for a_1 to the length of x_1 . In the condition of $t^u = t^l$, the estimation is calculated according to the method of Section 3.2.4. In the third condition, the estimation is calculated according to the method of Section 3.2. In Figure 16, 1(b) shows the estimation method for the second dimension. 1(c) describes the selectivity estimation for a B_{st} . In 2 of Figure 16, the final result of selectivity estimation is obtained

Procedure ETu(t^u, x, v, a, t)

Input: t^u , histogram update time; x , spatial bounding interval; v , velocity bounding interval; a , spatial interval of query; t , time interval of query

Output: sum , the value of the integration of $P(s)$

$sum \leftarrow 0$.

$I \leftarrow x \cap a$.

if $I \neq \emptyset$, $sum \leftarrow sum + IL(I)$.

$I \leftarrow CQL(t^u, v, a, t)$.

$I \leftarrow x \cap I$.

if $I \neq \emptyset \wedge t^l \neq t^h$, $sum \leftarrow sum + EQLTu(t^u, v, a, t, I)$.

$I \leftarrow CQR(t^u, v, a, t)$.

$I \leftarrow x \cap I$.

if $I \neq \emptyset \wedge t^l \neq t^h$, $sum \leftarrow sum + EQRTu(t^u, v, a, t, I)$.

Figure 15: Algorithm ETu.

by multiplying the sample factor sf .

4. EXPERIMENTS

In this section, we present the experimental environment and the experimental results of our proposed method. Moving objects are synthetically generated by using real-life spatial data to make a realistic experimental environment. We evaluate the accuracy of approximate results by using queries with various spatial rectangle sizes and various time interval lengths. In addition, we show that the existing spatial estimation method does not accurately estimate the selectivity of a query to moving objects.

4.1 Experimental Environment

Our experimental environment is similar to the previous works proposed by [5, 8]. Objects with the maximum speed 3.0 km/min (180 km/h) were set to move in 1000 km \times 1000 km 2D space. However, we generated moving objects to make a more realistic experimental environment than a simple synthetic experimental environment. We used a real-life spatial data, Tiger/lines [6] popularly used in spatial database research, to generate moving objects. As shown in Figure 17(a), we represented the initial spatial positions of moving objects using road data of California area Tiger/lines. The number of moving objects is 374,231. Figure 17(b) describes the velocity distribution to update the velocities of moving objects. To update the velocity of a moving object, we randomly chose a point in a circle with a radius 3 corresponding to the maximum speed. The position of the chosen point with respect to the center point of the circle determines the movement of the object. The distance between two points indicates the new updated speed of the object. The direction from the center point to the chosen point indicates the new updated direction of the object. Also, we used vertices of road data of Tiger/lines to generate realistic movements of moving objects with skewed speed distribution and skewed direction distribution. In the same way, we made another realistic experiment environment by using Sequoia [10] for the initial spatial positions and the movements of moving objects, as in Figure 18. The

Procedure selectivityST($t^u, H_{B_{st}}, Q, sf$)

Input: t^u , histogram update time;

$H_{B_{st}}$, spatio-temporal histogram; Q , query;

sf , sample factor

Output: sum , the result of selectivity estimation

$sum \leftarrow 0$.

1. **for each** $B_{st}(n, SB, VB)$ in the $H_{B_{st}}$,

$I_1 \leftarrow CQ(t^u, v_1, a_1, t)$.

$I_1 \leftarrow x_1 \cap I_1$.

$I_2 \leftarrow CQ(t^u, v_2, a_2, t)$.

$I_2 \leftarrow x_2 \cap I_2$.

if $I_1 \neq \emptyset \wedge I_2 \neq \emptyset$,

// computation for the first dimension

(a) **if** $v_1^l = v_1^h$, $sum_1 \leftarrow \frac{IL(I_1)}{IL(x_1)}$.
else if $t^u = t^l$, $sum_1 \leftarrow \frac{ETu(t^u, x_1, v_1, a_1, t)}{IL(x_1)}$.
else $sum_1 \leftarrow \frac{E(t^u, x_1, v_1, a_1, t)}{(x_1^h - x_1^l)(t^l - t^u)(v_1^h - v_1^l)}$.

// computation for the second dimension

(b) **if** $v_2^l = v_2^h$, $sum_2 \leftarrow \frac{IL(I_2)}{IL(x_2)}$.
else if $t^u = t^l$, $sum_2 \leftarrow \frac{ETu(t^u, x_2, v_2, a_2, t)}{IL(x_2)}$.
else $sum_2 \leftarrow \frac{E(t^u, x_2, v_2, a_2, t)}{(x_2^h - x_2^l)(t^l - t^u)(v_2^h - v_2^l)}$.

// product of two individual 1D estimations

(c) $sum \leftarrow sum + n(sum_1)(sum_2)$.

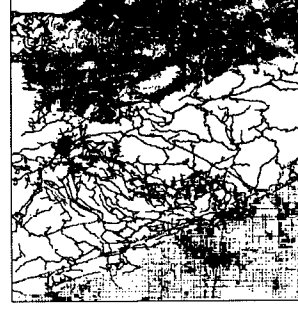
2. $sum \leftarrow (sum)(sf)$.

Figure 16: Algorithm selectivityST for 2D moving objects.

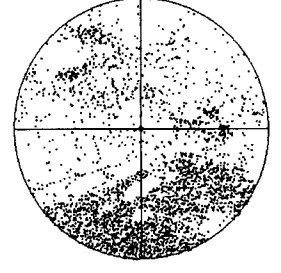
number of moving objects using Sequoia data is 670,201.

In general, most research presents the concept of time units to easily represent the passage of time [5, 7, 8, 11]. We used one minute as the time unit and updated approximately 1% of the whole moving objects every time unit. We randomly selected the updating data from the whole moving objects in general. The number of the sample objects for the spatio-temporal histogram is 1% of the whole moving objects. Therefore, the sf for the selectivityST algorithm of Figure 16 was set to 100. We selected the sample data from the whole moving objects using a simple round robin scheme. Also, an efficient random sampling scheme can be used. The sample data includes only information related to the histogram construction. The information for a sample object needs only the identification, the last update time, the spatial position of the time, and the velocity. This requirement decreases the size of the sample data remarkably.

We updated the spatio-temporal histogram from the sample data every time unit. This histogram update strategy can be possible because the I/O overhead for the spatio-temporal histogram update from the sample data is remarkably low, compared to the update overhead of approximately 1% of the whole moving objects per time unit. Experiments conducted on a Pentium IV 1.7 Ghz PC with 256 Mbytes of main memory. It took about 0.15 second to update the histogram from the sample data. The size of the spatio-temporal histogram was set to 4 Kbytes, which corresponds



(a) Initial spatial location.

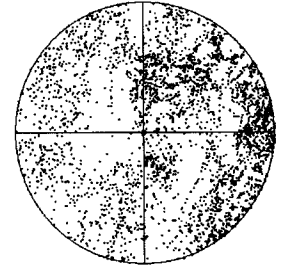


(b) Speed and direction for update.

Figure 17: Synthetic moving objects generated by Tiger/lines data.



(a) Initial spatial location.



(b) Speed and direction for update.

Figure 18: Synthetic moving objects generated by Sequoia data.

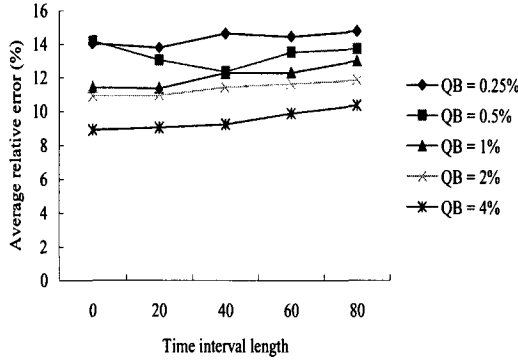
to a page size. We can use only the array values of SG for representing SB because the histogram is constructed only for the sample data. Like SB , we can use the approximated values for representing VB . So, the size of a spatio-temporal bucket was set to 12 bytes for $n = 4$ bytes, for $SB = 4$ bytes, and for $VB = 4$ bytes. The number of the spatio-temporal buckets $N_{B_{st}}$ is $\frac{4096}{12} \approx 341$. A spatial grid (SG) for constructing the spatio-temporal histogram was set to 50×50 .

Experiments were executed for 500 time units. The size of the query spatial rectangle (QB) varied as 0.25%, 0.5%, 1%, 2%, 4% of the size of the data space. The spatial position of the query was randomly chosen in the data space. The length of the query time interval (t) varied as 0, 20, 40, 60, 80. The t^l of the time interval was randomly chosen between t^u and $t^u + 80$. To generate various queries, we dealt with 25 queries: All combinations for five types of QB and five types of t as mentioned above. For each time unit, 25 queries were generated. The actual results of queries were compared to the results estimated using the proposed method in this paper. % error was used to assess the accuracy of estimation results:

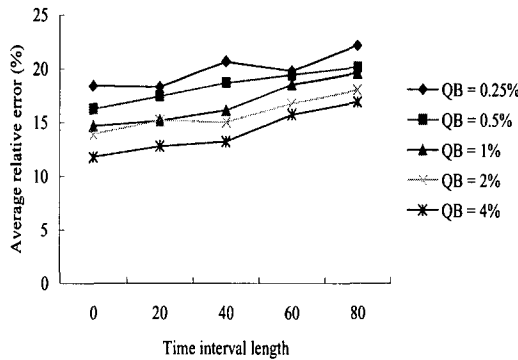
$$\text{error (\%)} = \frac{|\text{actual result} - \text{estimated result}|}{\text{actual result}} \times 100.$$

4.2 Experimental Results

Figure 19 shows the experimental results evaluated from 12500 queries ($\frac{25 \text{ queries}}{\text{time unit}} \times 500 \text{ time units}$). The results represent the average relative error with respect to the time interval length of the query and the spatial rectangle of the query. Each query type is evaluated by the average relative error of 500 queries ($\frac{1 \text{ query}}{\text{time unit}} \times 500 \text{ time units}$). Figure 19(a) shows the experimental results to moving objects using Tiger/lines data. The average relative error is from 9% to 15%. As the spatial rectangle size of the query increases, the average relative error decreases, as in the general experimental results [1]. However, as the time interval length of the query increases, the average relative error gradually increases. This is because our method considers the movements of moving objects that pass the left/right sides of the query as time passes. That is, the more the time interval length increases, the more the error in calculation using the left/right sides of the query increases. Figure 19(b) shows the similar experimental results to moving objects using Sequoia data.



(a) Moving objects generated by Tiger/lines data.



(b) Moving objects generated by Sequoia data.

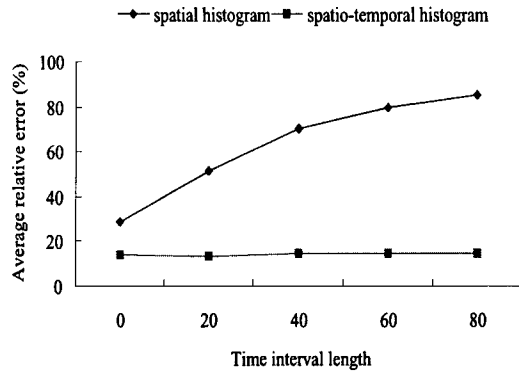
Figure 19: Average relative error for different QB sizes as the time interval length of the query is varied.

Since there has been no previous work that specifically addresses a selectivity estimation method to moving objects, we compared our proposed method with existing spatial selectivity estimation method using the MinSkew algorithm [1]. The comparative experiments between our spatio-temporal histogram and the previous spatial histogram are based on moving objects using Tiger/lines data. The use of Sequoia data shows similar results. Like the spatio-temporal histogram, the spatial histogram was reconstructed every time unit. The number of the spatial histogram buckets increased to 512 because only the number of objects and the spatial bounding rectangle are required. Figure 20(a) shows the selectivity estimation results with the spatial histogram and the spatio-temporal histogram with respect to the time interval length of the query applied to the QB size 0.25%. As expected, the spatial histogram that does not consider the property of moving objects produces high error rates. As the time interval length of the query increases, the accuracy of the estimation using the spatial histogram decreases remarkably. The average error rate of a query with a time interval length of 80 is about 85%. However, the spatio-temporal histogram considering the property of moving objects has accurate estimation results with an average error rate from 13% to 15%. Figure 20(b) shows the selectivity estimation results with the spatial histogram and the spatio-temporal histogram with respect to the time interval length of the query applied to the QB size 4%. Like Figure 20(a), our spatio-temporal histogram has better estimation results, compared to the spatial histogram.

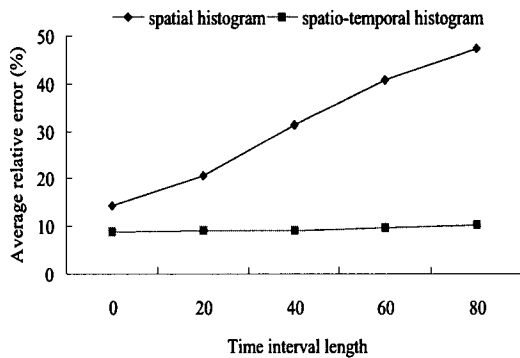
We also studied the impact of the size of the histogram. Figure 21 shows the average relative error with respect to the size of the spatio-temporal histogram for different QB sizes. Each query type is evaluated by the average relative error of 500 queries. The time interval length of queries is set to 40. As the size of the histogram increases, the average relative error decreases as in the general experimental results. Since more space can more accurately represent the distribution of data, the results are quite natural. Also, as QB size of the query increases, the average relative error decreases.

In addition, we assessed the average relative error with the spatial histogram and the spatio-temporal histogram with respect to histogram size for two QB sizes. The time interval length is set to 40. For Figure 22(a), QB size is set to 0.25%. As expected, our spatio-temporal histogram has accurate estimation results, compared to the spatial histogram. As the histogram size increases, the accuracy of the estimation using the spatio-temporal histogram increases. However, the accuracy of the estimation using the spatial histogram that does not consider the future locations of moving objects is not significantly affected by the histogram size. Figure 22(b) shows the impact of the histogram size applied to the QB size 4%. Like Figure 22(a), the spatial histogram technique does not accurately estimate the selectivity, compared to our spatio-temporal histogram.

Figure 23 shows the average relative error with respect to the size of the spatio-temporal histogram for different time interval lengths. The QB size of the query is set to 1%. As the size of the histogram increases, the average relative error decreases as in the general experimental results. However, as the time interval length of the query increases, the average relative error increases. The more the time interval length of the query increases, the more the error in calculation using the left/right sides of the query increases.



(a) QB size : 0.25%.

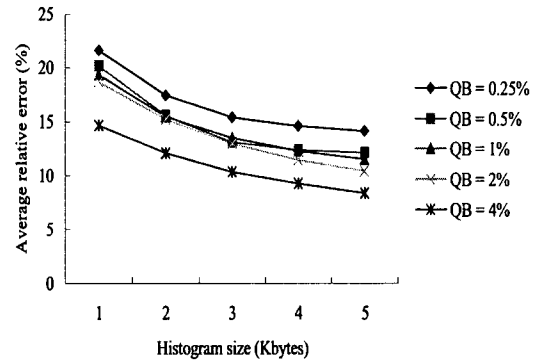


(b) QB size : 4%.

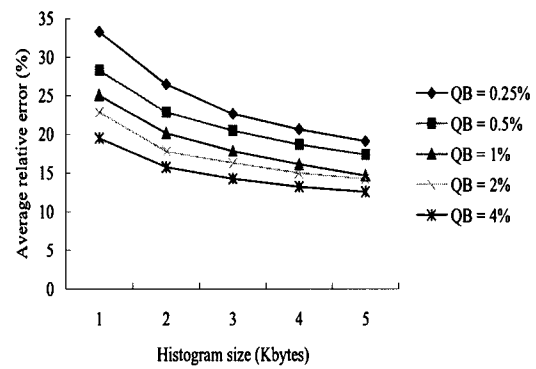
Figure 20: Average relative error for the spatial histogram and the spatio-temporal histogram as the time interval length of the query is varied when Tiger/lines is used.

5. CONCLUSIONS

Spatio-temporal databases have been studied intensively in recent years in light of technological developments taking into account the movements of objects. In this paper, we proposed a method to estimate the selectivity of a query for the future locations of moving objects. We used analytical formulas that accurately estimate the movements of moving objects. Our analytical formulas provide an advantage that these formulas can be basically used by various cost models for moving objects. To experiment in a realistic environment, we generated synthetic moving objects by using real-life spatial data with a reasonable skew distribution. In the experiments, the proposed method provided accurate estimation results over various queries with different spatial area sizes and time interval lengths. To our knowledge, the proposed method is the first work specifically addressing the selectivity estimation method for moving objects. So, we compared our proposed method with an existing spatial selectivity estimation method. It was observed that our proposed method accurately estimated the selectivity of a query to moving objects, compared with the existing spatial



(a) Moving objects generated by Tiger/lines data.



(b) Moving objects generated by Sequoia data.

Figure 21: Average relative error for different QB sizes as the histogram size is varied ($t = 40$).

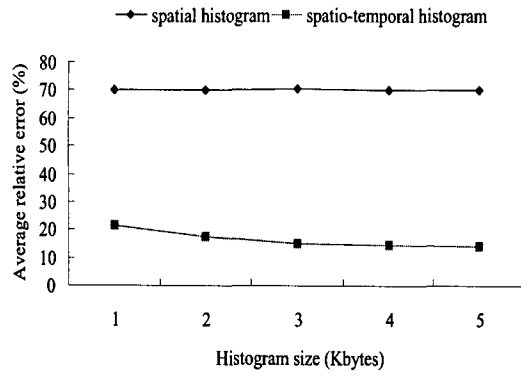
selectivity estimation method.

6. ACKNOWLEDGMENTS

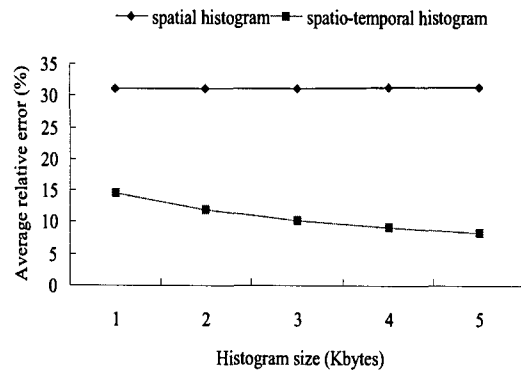
This work was supported by grant No. R01-1999-00324 from the Basic Research Program of the Korea Science & Engineering Foundation.

7. REFERENCES

- [1] S. Acharya, V. Poosala, and S. Ramaswamy. Selectivity estimation in spatial databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 13–24, 1999.
- [2] P. K. Agarwal, L. Arge, and J. Erickson. Indexing moving points. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 175–186, 2000.
- [3] L. Forlizzi, R. H. Guting, E. Nardelli, and M. Schneider. A data model and data structures for moving objects databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 319–330, 2000.

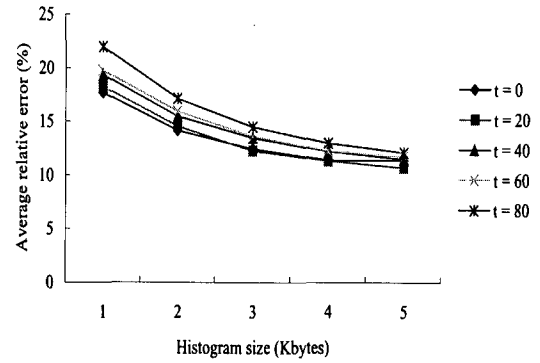


(a) QB size : 0.25%.

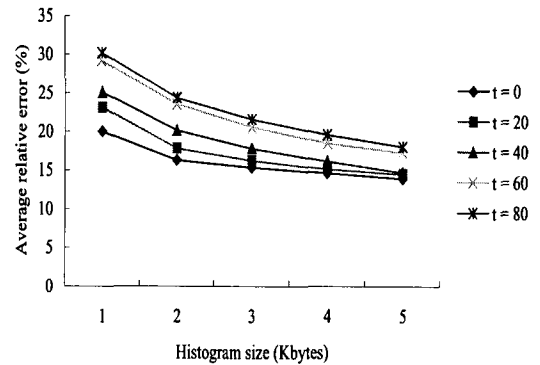


(b) QB size : 4%.

Figure 22: Average relative error for the spatial histogram and the spatio-temporal histogram as the histogram size is varied ($t = 40$) when Tiger/lines is used.



(a) Moving objects generated by Tiger/lines data.



(b) Moving objects generated by Sequoia data.

Figure 23: Average relative error for different time interval lengths as the histogram size is varied ($QB = 1\%$).

- [4] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. In *Proceedings of the International Conference on Very Large Data Bases*, pages 466–475, 1997.
- [5] G. Kollios, D. Gunopulos, and V. J. Tsotras. On indexing mobile objects. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 261–272, 1999.
- [6] U. S. B. of Census. Tiger/lines precensus files: 1994 technical documentation, technical report, 1994.
- [7] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel approaches in query processing for moving object trajectories. In *Proceedings of the International Conference on Very Large Data Bases*, pages 395–406, 2000.
- [8] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 331–342, 2000.
- [9] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao.

Modeling and querying moving objects. In *Proceedings of the IEEE International Conference on Data Engineering*, pages 422–432, 1997.

- [10] M. Stonebraker, J. Frew, K. Gardels, and J. Meredith. The sequoia 2000 storage benchmark. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 2–11, 1993.
- [11] Y. Tao and D. Papadias. Mv3r-tree: A spatio-temporal access method for timestamp and interval queries. In *Proceedings of the International Conference on Very Large Data Bases*, pages 431–440, 2001.
- [12] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez. Cost and imprecision in modeling the position of moving objects. In *Proceedings of the IEEE International Conference on Data Engineering*, pages 588–596, 1998.