Available online at www.sciencedirect.com

ScienceDirect

Information Sciences 177 (2007) 832–843

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

An efficient location encoding method for moving objects using hierarchical administrative district and road network ☆

SangYoon Lee [a], Sanghyun Park [a,*], Woo-Cheol Kim [a], Dongwon Lee [b]

[a] *Department of Computer Science, Yonsei University, South Korea*
[b] *College of Information Sciences and Technology, The Pennsylvania State University, USA*

Received 14 July 2005; received in revised form 1 July 2006; accepted 3 July 2006

Abstract

Due to the rapid development in mobile communication technologies, the usage of mobile devices such as cell phone or PDA has increased significantly. As different devices require different applications, various new services are being developed to satisfy the needs. One of the popular services under heavy demand is the location-based service (LBS) that exploits the spatial information of moving objects per temporal changes. In order to support LBS well, in this paper, we investigate how spatio-temporal information of moving objects can be efficiently stored and indexed. In particular, we propose a novel location encoding method based on hierarchical administrative district information. Our proposal is different from conventional approaches where moving objects are often expressed as geometric points in two-dimensional space, (x, y). Instead, in ours, moving objects are encoded as one-dimensional points by both administrative district as well as road information. Our method becomes especially useful for monitoring traffic situation or tracing location of moving objects through approximate spatial queries.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Location-based service; Road network; Moving object; Location encoding

1. Introduction

Due to the recent development in mobile communication technologies, the usage of mobile devices such as cell phone or PDA becomes increasingly popular, and novel services are being developed to serve various needs. One of the popular services for mobile devices is the location-based service (LBS) that exploits the location information of moving objects (i.e., mobile devices). For instance, the following queries are utilizing the "location" of moving objects: "Find the location of a person with a phone number X.", "What is the nearest Thai restaurant to a hotel Y?", or "Where is the delivery truck, shipping the TV that I purchased over the Internet?", etc.

☆ This article is the extended version of the paper [9].
* Corresponding author. Tel.: +82 2 2123 5714; fax: +82 2 365 2579.
E-mail addresses: sylee@cs.yonsei.ac.kr (S. Lee), sanghyun@cs.yonsei.ac.kr (S. Park), twelvepp@cs.yonsei.ac.kr (W.-C. Kim), dongwon@psu.edu (D. Lee).

0020-0255/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2006.07.016

The LBS is the service that keeps track of the location information of moving objects per time unit, stores them into databases, and handles users' queries based on the stored location information. The queries used in the LBS can be categorized as spatial (i.e., finding moving objects within some spatial constraints), trajectory (i.e., finding the trajectory of a specific moving object for a given time interval), and hybrid (i.e., both spatial and trajectory) queries [15].

In particular, moving objects in the context of LBS have the following challenges: (1) update cost is high since databases have to update location information as time passes; (2) storage cost is high since location information is typically multi-dimensional (i.e., object, time, location, etc.); (3) data to handle are large-scale since databases need to maintain temporal data (i.e., past and present) [19] and (4) retrieval cost is high due to large amount of data. Therefore, it is important to devise an indexing and query processing technique that can handle such large-scale multi-dimensional spatio-temporal data efficiently.

In this paper, we investigate a data encoding method to enable effective indexing and query processing for such a setting. In conventional approaches, the location information of moving objects were expressed as a geometric coordinate (x, y) in two-dimensional space. However, instead, we propose to express location information using both *hierarchical administrative district* and *road network* in one-dimensional space that, we believe, fits real world better. For instance, if a moving object is in a building at a coordinate of latitude $= 125.58$ and longitude $= -37.34$, then it can be expressed as a set of fields according to an administrative district such as city, road-name, road-block (e.g., Seoul, Main road, 165th block). Furthermore, by converting the fields into a binary string that has efficient ways to process queries, we overcome the aforementioned challenges of the LBS.

Our proposed scheme has at least three advantages: (1) it reduces the storage cost and dimensions of index by expressing location information in one-dimensional space, instead of two-dimensional space. This results in the improved query processing. (2) In real world, moving objects can only follow along the "roads". However, if one expresses location information as geometric coordinates, then one may include spaces where moving objects can never move into, so-called *dead space*, incurring storage waste. (3) Since location information is based on the information of the administrative district, the results can be easily converted into address formats that are easier, as answers, for human users to interpret.

This paper is organized as follows. Section 2 surveys related work on how to express location information using coordinates and road network. In Section 3, a novel way to express location information using hierarchical administrative district is described in detail. The architecture of the LBS system using the proposed method is described in Section 4, and the corresponding query processing method using the proposal is discussed in Section 5. In Section 6, our claims are carefully validated through experimentations using real-world data. Finally, concluding remarks follow in Section 7.

2. Related work

Existing indices such as 3DR-tree [21], HR-tree [10], STR-tree [14], TB-tree [15], or MV3R-tree [20] represent location information of moving objects as (x, y) coordinates in two-dimensional space, and index them using R-tree [8] structures with at least three dimensions. It is known that 3DR-tree and HR-tree are effective for spatial queries while STR-tree, TB-tree, and MV3R-tree are good for trajectory queries. However, the size of these indices rapidly increases as time passes, resulting in non-negligible search cost. In addition, since these schemes use geometric coordinates for capturing location information, they may include locations into which moving objects cannot move, incurring unnecessary waste in storage. For instance, consider the four locations, $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$, where only $(0,0)$ and $(1,1)$ are valid locations. Then, aforementioned schemes require 2 bits to represent the four locations when only 1 bit is sufficient to represent the two valid locations, $(0,0)$ and $(1,1)$. In general, dead spaces like $(0,1)$ and $(1,0)$ waste a lot of storage spaces.

Therefore, to remedy these shortcomings, recent developments try to exploit location information along with the road network in real world, or to reduce dimensions of indices by separately storing the dimensions of moving objects.

Research using road network [7,12] limits the movable boundary of objects to roads so that it represents location information as a pair of (the identifier of the nearest road, the distance from the nearest road). In [12], Papadias et al. propose an index suitable for such road coordinates and discuss an algorithm that uses the network distance, instead of the Euclidean distance, to measure the distance between two coordinates. For instance, for a query "find the nearest hotel from the current location", the algorithm of [12] finds the closest hotel in terms of actual distance aligned with real roads, instead of the shortest geometric distance. Although such road coordinates do not cause the dead space problem, the locations of roads and moving objects are still represented in two-dimensional space.

In [7], Gupta et al. propose a scheme where nodes on the road network are systematically converted into binary strings, and thus various queries can be efficiently handled by utilizing simple operations on the binary strings (e.g., Hamming distance). In this scheme, however, the length of binary strings is proportional to the number of nodes in the road network, and the relative locations of roads and moving objects are not easily obtainable from the binary strings of road network nodes.

Refs. [3,6,1,2] are some of the representative works that aim at reducing the dimensions of indices to improve query processing performance. Their basic idea is to utilize two types of indices, one for storing location information and the other for storing time information. In [3], Chakka et al. divide geographic space into lattice of cells and create an index for each cell. Then, the time information of all objects within a cell is stored in the corresponding index. In [6], Frentzos constructs a two-dimensional R-tree for the locations of roads, and makes each leaf node point to another R-tree to keep the time information of objects on the road. In [1,2], Almeida and Guting maintain two types of two-dimensional R-trees, one for the location information of roads, and the other for the time and relative position information of objects on the road. These three methods express the locations of moving objects effectively by exploiting the observation that moving objects seldom change their locations much per time. However, they still have to use every information of moving objects, which are at least three-dimensional, for query processing.

The method proposed in [16,17] uses the Hilbert curve [5] to transform two-dimensional locations into one-dimensional representation. Since this method does not consider hierarchical administrative district during the transformation, it does not fit well the real world situations. The binary string representation proposed in this paper, however, is based on the information of hierarchical administrative district and thus can be easily converted into address formats that are easier for human users to interpret.

3. Proposed encoding method of location information

In many countries, addresses are often represented as a set of fields such as district name, road name, and location on the road. For example, the address of the City Hall of Seoul, Korea is represented as a triplet of (Seoul, Eulji road, 31). Similarly, the address of the Natural History Museum of England is (London, Cornwell Road, –), where the third field is null. Exploiting this addressing scheme, one can easily encode the location of a moving object as a one-dimensional binary string. By adding more fields, it is trivial to extend the scheme to be able to support more general addresses. From here forward, to keep the presentation simple, we only focus on the triplet scheme, (district, road, location on road), to represent addresses within a specific country.

The procedure to encode the location of a moving object consists of three steps: (1) obtain the address of the place at which the moving object is located and express it as a triplet, (2) transform each field of the triplet into a binary string, and (3) concatenate the three binary strings into a single binary string. The first step will be elaborated in Algorithm 2 of Section 4, and the second step will be discussed in the current section. We omit the discussion of the third step since it is trivial.

The second step in turn consists of three binary string mappings for administrative districts, roads, and relative locations on roads, respectively.

3.1. Mapping administrative districts into binary strings

We first discuss how to encode districts. For easier illustration, let us consider an imaginary country with 4 counties (A, B, C, D) as a whole and 8 cities ($a,b,\ldots,g$) in each county – a total of 32 districts to encode. The simplest encoding method is to use their lexicographical orders. That is, by using 2 bits for county names and 3 bits for city names, one can encode a district as a 5 bit string whose first two bits represent the lexicographical order of its county name and the remaining three bits represent the lexicographical order of its city name.

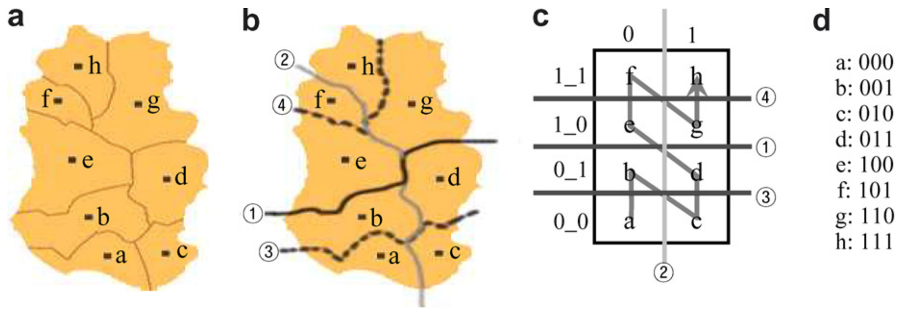


Fig. 1. An example which illustrates how Algorithm 1 works on county *A*. (a) Compute the centroid of each district in country *A*, (b) divide the region into two sub-regions repeatedly, (c) map the centroids onto a two-dimensional space, and (d) assign a binary string to each district.

For example, one can express the district "county *A* city *a*" as "00 000", the district "county *A* city *b*" as "00 001" and "county *B* city *a*" as "01 000".

Although this encoding scheme is simple to implement, it does not provide the relative position of districts. For example, let us consider two moving objects, one located at the district "00 000" and the other at the district "00 001". Comparing these two binary strings, one can deduce that two objects be in the same county but in different cities. These two binary strings, however, do not provide any clue as to the relative positions of the two objects.

To overcome these limitation, we propose to use a mapping technique based on space-filling curves such as Z-ordering [11], R-ordering [4], and H-ordering [5]. A space-filling curve is a one-dimensional curve which visits every point within a multi-dimensional space.

The detailed algorithm, which follows the rule of Z-ordering to encode the districts contained in a region, is given in Algorithm 1, and an illustrative example is shown in Fig. 1.

Compared to the encoding method based on lexicographical order, the proposed encoding method produces more informative binary strings. Let us consider two moving objects again, one located at the district "00 000" and the other at the district "00 001". In addition to the fact that two objects are in the same county but in different cities, we can infer that: (1) since the first two bits for cities are all "00", the cities are located at southwest area of the county, and (2) since the last bits for cities are different, the city where the first object is located is south of the city where the second object is located.

Algorithm 1. Mapping administrative districts into binary strings

```
1. Compute the centroid of each district.
2. Divide the region into two sub-regions, south and north, so that the numbers of centroids in both south
   and north are similar.
3. If region south has more than one centroid, divide it into two sub-regions, south_east and south_west, so
   that the numbers of centroids in both south_east and south_west are similar.
4. Do the same for region north symmetrically.
5. For each sub-region obtained from Steps 3 and 4, if it contains more than one centroid, repeat Steps 2–4.
6. Considering the division process undergone, map the of each district onto a two-dimensional space.
7. Using a Z-ordering, assign a binary string to each district.
```

3.2. Mapping roads and relative locations on roads into binary strings

The algorithm to encode the roads within a district is not much different from Algorithm 1. The changes needed to be made to Algorithm 1 are as follows: (1) every instance of word "district" is to be replaced with word "road", and (2) every instance of word "region" is to be replaced with word "district".

With these changes to Algorithm 1, road information can be mapped to binary strings as well. However, unlike administrative district information that is represented as a "region", road information is represented as a "line". Therefore, when one uses the center points of roads to represent the location of moving objects, it can

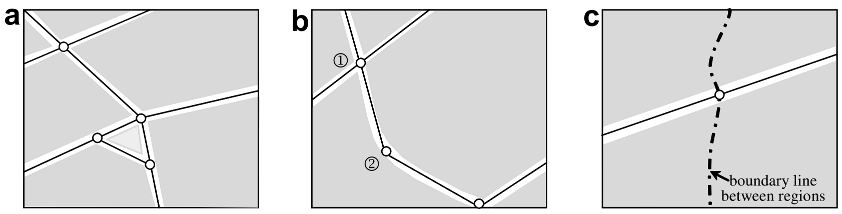


Fig. 2. Decomposition of roads into a set of manageable units. (a) Divide the roads at their intersections, (b) express the curvy roads as a list of straight roads, (c) divide the roads at the boundary line.

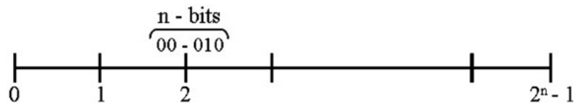


Fig. 3. A road which is partitioned into $2^n - 1$ units of the same size.

be erroneous for curvy roads whose center points may correspond to multiple points (to be elaborated in Section 3.3). To overcome this limitation, one needs to partition roads as follows: (1) Partition roads into sub-roads using crossroads. For instance, the roads of Fig. 2(a) can be partitioned into ten sub-roads using four crossroads (circles in the figure). (2) Partition curvy or crooked roads into a combination of straight roads such that each straight road is stretched furthest. For instance, in Fig. 2(b), ① and ② represent the partition points upon the crossroad and curvy road, respectively. (3) Finally, if roads are across multiple administrative regions, partition them at the boundary line of regions (refer to Fig. 2(c)).

Once all roads are partitioned into sub-roads, their mappings to binary strings are performed using Algorithm 1, and combined with binary strings mapped from administrative districts (created in Section 3.1).

Now let us consider how to encode the location on road. We first partition a road into $2^n - 1$ units of the same size, and then represent each boundary as an n-bit binary string as shown in Fig. 3. Then, we choose the boundary nearest from an object and use its binary string as the location of the object on the road.

The proposed encoding scheme has the following characteristics: (1) one can find out the *lowest* common administrative district by extracting the longest common prefix of a given set of binary strings, and (2) a district containing a set of lower districts can be represented by the range of binary strings; for example, county "A" in Fig. 1 is represented by the range [00000, 00111].

3.3. Consideration of geo-problems

Let us consider various cases where errors may occur during the mapping of location information, and possible solutions to fix the errors.

The first source of errors that degrades the accuracy of our proposal is that the location of moving objects on roads may not be mapped to points on the roads. This is mainly caused by two reasons: (1) GPS typically allows error margin of 10 m for an object. (2) When moving objects are not located in the middle of roads, our proposal may incur error margin up to the width of the road. This is because roads are only represented as straight lines in our scheme. For instance, Fig. 4(a) illustrates the locations of moving objects collected by GPS. If one simply relies on the collected location information, the locations of the moving objects will be represented as in Fig. 4(b). However, by considering error margins, Fig. 4(b) must be corrected to Fig. 4(c).

The second source of errors is the granularity of partition. The same issue is important in the traditional two-dimensional approach of (x, y), but becomes more important in our scheme. For instance, consider the partition of a road of 320 m long into 16 sub-roads (20 m each). Although two objects are located in the same sub-road, it is possible that they can be as far as 20 m apart each other – non-negligible distance in some LBS applications. One can increase the granularity of partition further to avoid such a problem at the cost of longer binary strings and thus increased storage.

The last source of errors is objects intersected among various regions. Our binary string scheme uses the center points of objects in representing their locations. Therefore, even if an object is intersected by many

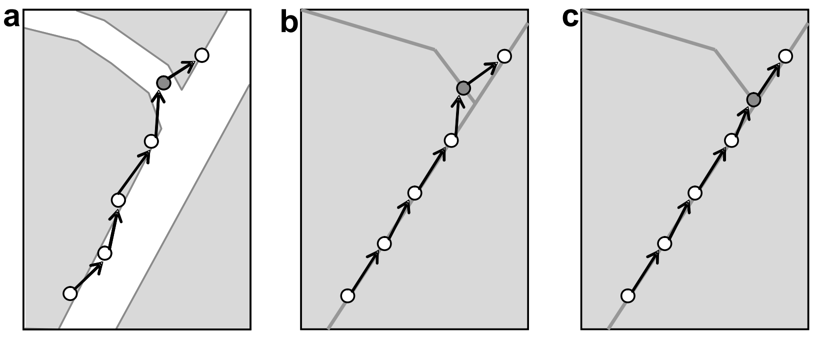


Fig. 4. GPS errors and their corrections. (a) Locations of moving objects collected by GPS, (b) mapping onto road networks without error correction, (c) mapping onto road networks with error correction.

regions, only one region to which the center point of the object belongs represents the object, causing errors in the search. For instance, consider the city hall intersected by many regions. Now, we want to find out which rooms Sam has walked around in the city hall. Even though Sam has walked into several rooms of the city hall, since the address of the city hall is only represented as its center point, one cannot track down which rooms he has visited. One way to remedy this limitation is to use a range of binary strings such as "000000–000011" for objects intersected by many regions (especially static objects like buildings).

4. System organization

As shown in Fig. 5, our LBS implementation consists of two sub-systems for *population* and *query processing*. The population sub-system is responsible for collecting the information of moving objects and storing it into databases, and the query processing sub-system is in charge of answering queries for moving objects. To support the proposed encoding scheme, in addition, the LBS system needs three conversion modules, XY2BS, AD2BS, and BS2AD.

4.1. Module XY2BS

In this paper, we map administrative district information and relative locations of objects on the roads into binary strings. However, not all locations of objects are correctly mapped onto roads. Therefore, we need to fix the mapping of geometric coordinates into the points on the roads. Module XY2BS converts a two-dimensional coordinate for the location of a moving object into a binary string. To expedite the conversion process, XY2BS maintains an R-tree [8] built from the roads in administrative districts. For a given road R, let *bitstring*(R) and *rectangle*(R) denote the binary string of R and the rectangle for the two end points of R, respectively. For each road R in districts, then, the R-tree stores *rectangle*(R) and *bitstring*(R) in one of its leaf nodes. Algorithm 2 describes how XY2BS makes use of the R-tree to quickly convert two-dimensional points to corresponding binary strings.

Algorithm 2. Utilizing an R-tree to quickly convert a two-dimensional point, (x,y), into the equivalent binary string

```
1. Generate the rectangle $uMBR$ by expanding $x$ to its left and right by $uR$, and expanding $y$ up and down by
   $uR$. $uMBR$ is then expressed as $([x - uR, x + uR], [y - uR, y + uR])$. Here, $uR$ is a system parameter used
   for determining the nearness of roads from a two-dimensional point.
2. Search the R-tree for the roads whose MBRs overlap $uMBR$.
3. From the roads obtained in Step 2, select the road $R$ whose Euclidean distance to $(x,y)$ is the smallest.
4. Project $(x,y)$ onto the road $R$. Let $(x',y')$ denote the coordinate of $(x,y)$ after the projection.
5. Using the relative position of $(x',y')$ on the road $R$, calculate the binary string for $(x',y')$.
6. Concatenate bitstring($R$) and the binary string for $(x',y')$.
```

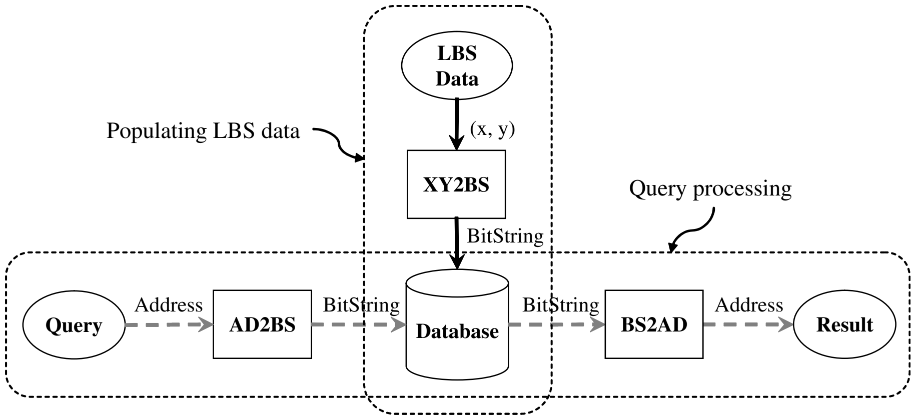


Fig. 5. LBS system which uses the proposed location encoding scheme.

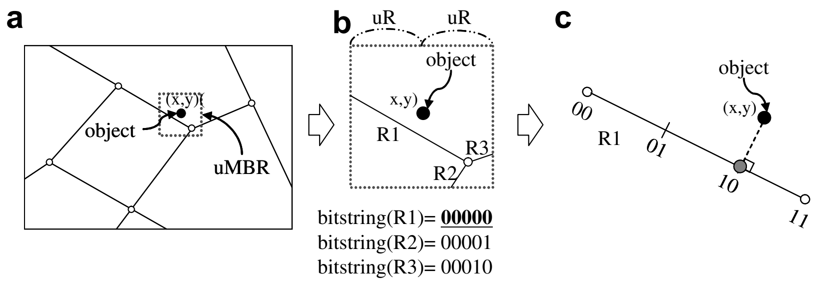


Fig. 6. An example which illustrates how Algorithm 2 works. (a) Generate the *uMBR* by expanding x and y by *uR*, (b) retrieve the roads whose MBRs overlap *uMBR*, (c) select the closest road and project (x,y) onto it.

The example of Fig. 6 further illustrates Algorithm 2. Fig. 6(a) depicts the *uMBR* for a given two-dimensional point, (x,y), and Fig. 6(b) shows the three roads, $R1$, $R2$, and $R3$, obtained from the R-tree. Since the Euclidean distance from $R1$ to (x,y) is the smallest, (x,y) is projected onto $R1$, as shown in Fig. 6(c). Lastly, both binary string of $R1$ and that of the relative position of (x',y') on $R1$ are concatenated to produce the final output, "00000 10". Here, (x',y') denotes the coordinate of (x,y) after the projection.

4.2. Modules AD2BS and BS2AD

It is more intuitive for users to ask queries using real-life address such as "Seoul, Main road, 100" than using coordinates such as "longitude = −65, latitude = +45". Similarly, it is also preferable to use such real-life address in the query results. Therefore, in our prototype, we assume that both users' queries and query results are in the real address format. Module AD2BS converts this real-life addresses into equivalent binary string representations, and module BS2AD converts binary strings back to their equivalent real-life addresses. For a rapid conversion to binary strings, AD2BS maintains a B-tree where district and road names are used as a key and binary strings are stored at leaf nodes. For a fast conversion to real-life addresses, BS2AD also maintains a B-tree where binary strings are used as a key, and district and road names are stored at leaf nodes.

5. Query processing

This section describes how our LBS implementation processes typical LBS range and trajectory queries.

5.1. Range query processing

Range queries are to find the moving objects within a specific region during a given time interval or to find a set of time intervals during which a specific moving object is within a given region. Let us consider an example query: "Find all cell phone users who have been in city *b* of county *A* during the time interval [8 pm, 9 pm]". To answer this query, system first calls module AD2BS to convert the district name (i.e., "county *A*, city *b*") to the corresponding binary string. When there are more than a single road in the given district, the district name is expressed as a range of binary strings. The system then searches the database using the range of binary strings and the time interval (i.e., [8 pm, 9 pm]) as a query predicate.

In this paper, we focus on range queries that use administrative district names as part of search condition. Conventional range queries with two-dimensional (x, y) rectangle can be processed as follows: (1) locate the smallest intersection of administrative regions that fully contain the (x, y) rectangle, (2) execute a range query using all administrative district names contained in the (x, y) rectangle, and (3) among the results, prune away those objects not contained in the (x, y) rectangle.

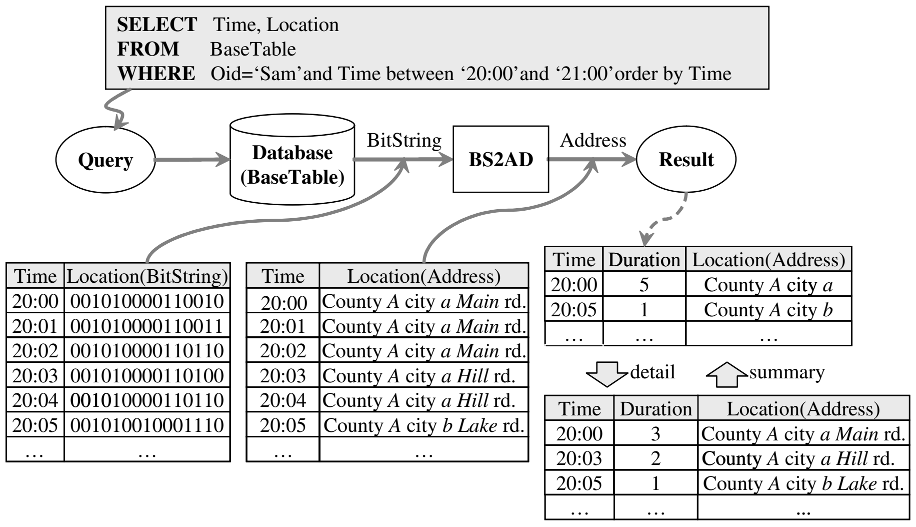
5.2. Trajectory query processing

Trajectory queries are to retrieve the path along which a moving object has traversed during a given time interval. Let us consider an example query: "Between the time interval [8 pm, 9 pm], where has Sam been moving around?". If the system represents the locations of moving objects as two-dimensional geometric points, the answer to such a query consists of a set of line segments and thus can be meaningfully displayed only on electronic maps. However the answers from the proposed LBS system can be easily converted to real-life addresses and thus can be delivered to users in text or voice format (in addition to being useful on electronic maps as well).

To process trajectory queries, the system first searches the database using the object and time interval information, and then sort the result in ascending order of time as shown in Fig. 7. The system then calls module BS2AD to convert the binary strings in the result into the corresponding administrative district addresses, and finally sends out the result in text or voice format to users.

When showing the result to users, the system may represent a set of adjacent rows as a single row by extracting their common prefixes.

For instance, as to the query of Fig. 7, "find the trajectory of Sam from 20:00 to 21:00", if answers are to be shown in the unit of "county", they can be compressed to ⟨entrance timestamp to "county", duration of stay



```
SELECT  Time, Location
FROM    BaseTable
WHERE   Oid='Sam'and Time between '20:00'and '21:00'order by Time
```

Time	Location(BitString)
20:00	001010000110010
20:01	001010000110011
20:02	001010000110110
20:03	001010000110100
20:04	001010000110110
20:05	001010010001110
…	…

Time	Location(Address)
20:00	County *A* city *a Main* rd.
20:01	County *A* city *a Main* rd.
20:02	County *A* city *a Main* rd.
20:03	County *A* city *a Hill* rd.
20:04	County *A* city *a Hill* rd.
20:05	County *A* city *b Lake* rd.
…	…

Time	Duration	Location(Address)
20:00	5	County *A* city *a*
20:05	1	County *A* city *b*
…	…	…

Time	Duration	Location(Address)
20:00	3	County *A* city *a Main* rd.
20:03	2	County *A* city *a Hill* rd.
20:05	1	County *A* city *b Lake* rd.
…	…	...

Fig. 7. An example of trajectory query processing.

in "county", name of "county"⟩. That is, the answer tuple ⟨20:00, 5, "county *A* city *a*"⟩ indicates that "Sam entered into the city *a* at 20:00 and stayed for 5 min". Similarly, for the unit of "city", answers can be compressed to the tuple ⟨entrance timestamp to "city", duration of stay in "city", name of "city"⟩. That is, it is feasible that the result is displayed in the unit of "county" first and, whenever necessary, in the unit of "city" (similar to the drill-down of OLAP applications).

6. Experimental validation

To validate the effectiveness of the proposed location encoding scheme, we performed experiments with real district and road data from a specific region in Seoul, Korea. The region consists of 2 counties (called "gu" in Korea), 46 cities (called "dong" in Korea), and 387 roads. For testing, we generated up to 2000 moving objects within this region and randomly move them around along the roads for 500 min. We then traced their changing locations every minute, and stored them in (1) 3DR-tree as a triplet of (timestamp, *x*-coordinate, *y*-coordinate) and (2) in 2DR-tree as a pair of (timestamp, binary string representation of location). Identifiers of moving objects were used as a key and thus stored in leaf nodes of both 3DR-tree and 2DR-tree. As evaluation metrics, we use index size and query processing time. The experiments were done in a PC with a Pentium-IV 2.6 GHz CPU, the main memory of 512 MB, and Linux Fedora core 3.

6.1. Performance of XY2BS

The proposed binary string representation will not be very useful if the process of converting two-dimensional coordinates into equivalent binary strings requires too much overhead. Therefore, before evaluating the effectiveness of the proposed binary string representation, we measured the performance of module XY2BS. More specifically, while increasing the number of two-dimensional locations from 400 K to 2000 K, we measured how long it would take for XY2BS to convert all the locations into the equivalent binary string representation. As shown in Fig. 8, the total time spent for the conversion increases almost linearly with the number of two-dimensional locations and more than seven million locations can be processed per second. From this experimental result, we conclude that the overhead of XY2BS be negligible for a typical LBS environment and tolerable even for an environment with a huge number of locations.

6.2. Index size

While increasing the number of moving objects from 400 to 2000 for 500 min, we measured the sizes of both 2DR-tree and 3DR-tree. As shown in Table 1, the 2DR-tree that uses the proposed binary string representation consumed about 58% of the storage space spent by the 3DR-tree on average.

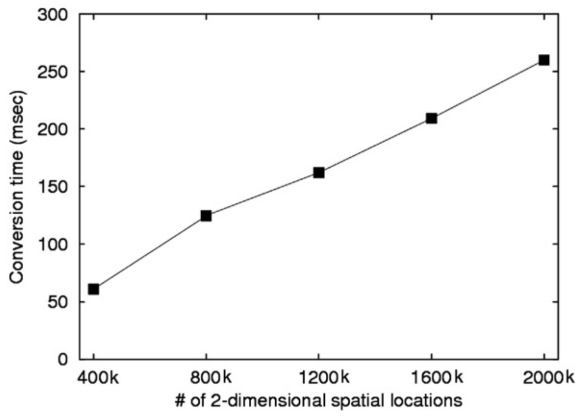


Fig. 8. Elapsed time to convert two-dimensional locations into equivalent binary strings.

Table 1
Sizes of 2DR-tree and 3DR-tree

Number of moving objects	Size of 3DR-tree (KB)	Size of 2DR-tree (KB)	Reduction ratio (%)
400	10,973	6393	41.7
800	22,467	12,779	43.1
1200	34,342	19,329	43.7
1600	46,221	25,906	44.0
2000	58,218	32,565	44.1

6.3. Query processing time

While increasing the number of moving objects from 400 to 2000, we measured the elapsed time to process 1000 of range and 1000 of trajectory queries for both 2DR-tree and 3DR-tree.

We first performed two types of range queries: "Find a set of time intervals during which a specific object was in a given *city*" (type 1) and "Find a set of time intervals during which a specific object was in a given *county*" (type 2). As shown in Fig. 9, the query processing times of both the 2DR-tree and the 3DR-tree increase linearly as the number of moving objects grows. However the increase rate of 2DR-tree is slower than that of 3DR-tree. The 2DR-tree outperforms the 3DR-tree by 98% for type 1 and by 67–69% for type

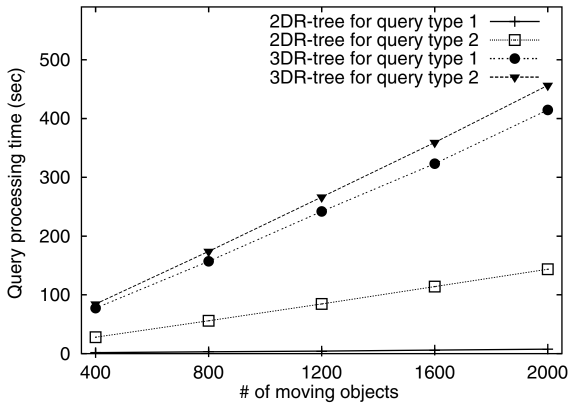


Fig. 9. Elapsed time to process type 1 and type 2 queries.

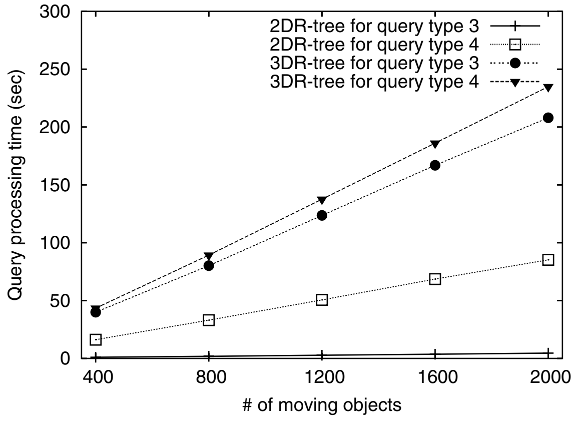


Fig. 10. Elapsed time to process type 3 and type 4 queries.

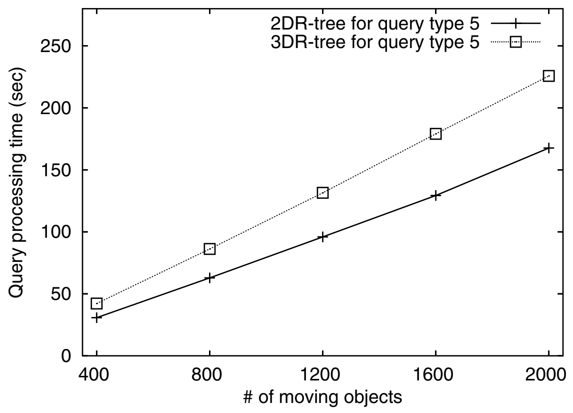


Fig. 11. Elapsed time to process type 5 queries.

2 queries. The improvement of our proposal is mainly due to the compact and efficient representation of search regions as ranges of one-dimensional binary strings.

We then performed another two types of range queries: "Find the moving objects which were within a given *city* at any time in the first 250 min" (type 3) and "Find the moving objects which were within a given *county* at any time in the first 250 min" (type 4). To process these types of range queries, we have to search an index using rectangles representing the time and location constraints. Remember that the 2DR-tree and the 3DR-tree express locations as one-dimensional binary strings and two-dimensional geometric points, respectively. Therefore, the search regions for the 2DR-tree become two-dimensional while those for the 3DR-tree become three-dimensional. As shown in Fig. 10, the performance improvement of the 2DR-tree has increased as the number of moving objects increases. As a result, compared to the 3DR-tree, the 2DR-tree achieved improvement up to 98% for type 3, and up to 64% type 4 queries.

We lastly performed a trajectory query: "Where has a specific object been moving around for the first 250 min?" (type 5). To process such a query, we have to traverse down an index using the time constraint. Since traversing the 2DR-tree is more effective than traversing the 3DR-tree in terms of CPU and I/O cost, as shown in Fig. 11, the 2DR-tree achieved about 44% performance improvement when there were 400 objects and about 41% when there were 2000 objects (Fig. 11).

7. Conclusion

In this paper, we have proposed an effective location encoding method that uses the information in the hierarchical administrative district and the road network of real world. Our method captures moving objects as binary strings in one-dimensional space instead of conventional (x, y) coordinates in two-dimensional space, and thus can reduce storage cost by up to 44% while improving query processing by 64–98%. The benefits of our proposal include: (1) it improves upon previous indexing and query processing algorithms by exploiting binary strings; (2) it is easy to drill-down or roll-up query results in a hierarchical administrative district; (3) since it uses the ontologies of administrative district that are intuitive to human users, it is suitable to display query results as text or voice even without electronic maps.

We are currently extending the proposed binary encoding method in the following two directions: One is to make use of the proposed method to support future queries, which are to predict the positions at which some target objects are located at a specific time in future. There have been several approaches (e.g., TPR-tree [18] and dynamic histogram [13]) for processing future queries but our aim is to utilize binary-encoded trajectory data to answer such queries. The other is to reduce the rapidly increasing spatio-temporal data without sacrificing query performance by exploiting the property such that when two binary strings of location information share the same prefix, two corresponding moving objects on the road network must be located in the same administrative district. That is, the location information of moving objects can be further compressed per administrative district by using common prefixes.

Acknowledgement

This work was supported by the Seoul R&BD Program(10561) in 2006.

References

[1] V. Almeida, R.H. Guting, Indexing the trajectories of moving objects in networks, in: Proc. 16th International Conference on Scientific and Statistical Database Management, Santorini Island, 2004, pp. 115–118.
[2] V. Almeida, R.H. Guting, Indexing the trajectories of moving objects in networks, in: V. Springer Science and Business Media B.V., GeoInformatica, 2005, pp. 33–60.
[3] V.P. Chakka, A. Everspaugh, J.M. Patel, Indexing large trajectory data sets with SETI, in: Proc. 1st Conference on Innovative Data Systems Research, Asilomar, CA, 2003, pp. 164–175.
[4] C. Faloutsos, Gray codes for partial match and range queries, in: V. Friedrich, H. Vogt (Eds.), IEEE Transactions on Software Engineering, vol. 14, 1988, pp. 1381–1393.
[5] C. Faloutsos, S. Roseman, Fractals for secondary key retrieval, in: Proc. 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Philadelphia, Pennsylvania, 1989, pp. 247–252.
[6] E. Frentzos, Indexing objects moving on fixed networks, in: Proc. 8th International Symposium on Spatial and Temporal Databases, Santorini Island, 2003, pp. 289–305.
[7] S. Gupta, S. Kopparty, C. Ravishankar, Roads, codes, and spatiotemporal queries, in: Proc. 22th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Diego, California, 2004, pp. 115–124.
[8] A. Guttman, R-trees: a dynamic index structure for spatial searching, in: Proc. the 1984 ACM SIGMOD Conference on Management of Data, Boston, Massachusetts, 1984, pp. 47–54.
[9] S. Lee, S. Park, W. Kim, D. Lee, An efficient location encoding method based on hierarchical administrative district, in: Proc. 16th International Conference on Database and Expert Systems Applications, Copenhagen, Denmark, 2005, pp. 890–899.
[10] M.A. Nascimento, J.R.O. Silva, Towards historical R-trees, in: Proc. 13th ACM Symposium on Applied Computing, Atlanta, Georgia, 1998, pp. 235–240.
[11] J.A. Orenstein, T.H. Merrett, A class of data structures for associative searching, in: Proc. 3rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Waterloo, Ontario, 1984, pp.181–190.
[12] D. Papadias, J. Zhang, N. Mamoulis, Y. Tao, Query processing in spatial network databases, in: Proc. 29th International Conference on Very Large Databases, Berlin, 2003, pp. 802–813.
[13] H. Park, J. Son, M. Kim, Dynamic histograms for future spatiotemporal range predicates, Information Sciences 172 (1–2) (2005) 195–214.
[14] D. Pfoser, Y. Theodoridis, C.S. Jensen, Indexing trajectories in query processing for moving objects, Chorochronos Technical Report, 1999, CH-99-3.
[15] D. Pfoser, C.S. Jensen, Y. Theodoridis, Novel approaches in query processing for moving objects, in: Proc. 26th International Conference on Very Large Databases, Cairo, 2000, pp. 395–406.
[16] D. Pfoser, C.S. Jensen, Indexing of network constrained moving objects, in: Proc. 11th ACM Symposium on Advances in Geographic Information Systems, New Orleans, Louisiana, 2003, pp. 25–32.
[17] Dieter Pfoser, Christian S. Jensen, Trajectory indexing using movement constraints, in: V. Springer Science and Business Media B.V., GeoInformatica, 2005, pp. 25–32.
[18] S. Saltenis, C.S. Jensen, S.T. Leutenegger, M.A. Lopez, Indexing the positions of continuously moving objects, in: Proc. 19th ACM SIGMOD on Management of data, Dallas, Texas, 2000, pp. 331–342.
[19] Z. Shen, J. Luo, C. Zhou, S. Cai, J. Zheng, Q. Chen, D. Ming, Q. Sun, Architecture design of grid GIS and its applications on image processing based on LAN, Information Sciences 166 (1–4) (2004) 1–17.
[20] Y. Tao, D. Papadias, MV3R-Tree: a spatio-temporal access method for timestamp and interval queries, in: Proc. 27th International Conference on Very Large Databases, Roma, 2001, pp. 431–440.
[21] Y. Theodoridis, M. Vazirgiannis, T.K. Sellis, Spatio-temporal indexing for large multimedia applications, in: Proc. 3rd IEEE International Conference on Multimedia Computing and Systems, Hiroshima, 1996, pp. 441–448.