TR-IIS-10-007

Compressing Trajectories Using Inter-Frame Coding

Cheng-Yu Lin, Hung-Chia Chen, Ying-Yu Chen, Wang-Chien Lee, and Ling-Jyh Chen

Sep. 8, 2010 || Technical Report No. TR-IIS-10-007
http://www.iis.sinica.edu.tw/page/library/TechReport/tr2010/tr10.html

Compressing Trajectories Using Inter-Frame Coding

Cheng-Yu Lin[†], Hung-Chia Chen[†], Ying-Yu Chen[†], Wang-Chien Lee[‡], and Ling-Jyh Chen[†]

[†]Institute of Information Science, Academia Sinica, Taiwan

{ntuaha, hcchen, ciy, cclljj}@iis.sinica.edu.tw

[‡]Department of Computer Science and Engineering, The Pennsylvania State University

wlee@cse.psu.edu

Abstract

With the advances in wireless communications and GPS technology, there is increasing interest in the field of location-aware services. Because of the proliferation of GPS-enabled devices and applications, in this study, we address the scalability issue in trajectory data management. Specifically, we propose a scheme called Inter-Frame Coding (IFC) for lossless compression of trajectory data, and implement two classical database queries based on the scheme. Evaluations of the IFC scheme using real trajectory datasets show that it can achieve a compression ratio of 58%. Moreover, it can reduce the computational complexity of range queries by a factor of 0.45, while maintaining an acceptable execution time in k-nearest neighbor searches. The IFC scheme is simple, efficient, and lossless; thus, it has the potential to facilitate trajectory-based data storage, compression, and computation.

I. Introduction

With the advances in wireless communications and GPS technology, location-aware services are rapidly permeating every part of our living environments. An enormous number of innovative applications have been developed and deployed in recent years, e.g., location-based social networks, driving navigation aids, and mobile object tracking devices. The difference between the new genre of applications and conventional ones is that they are driven by moving objects with the location information as a function of time [32].

Moving object databases (MODs) have shown promise in supporting applications of this genre [36], and a substantial amount of research effort has been invested in the areas of indexing [7–9, 19, 24, 30], querying [11, 27, 29, 33, 34], searching [13–16, 18, 20, 28, 35], and security [22, 26]. Many of these approaches are based on trajectory data provided by moving objects; however, the amount of data increases dramatically over time, leading to storage, transmission, and computation problems. Consequently, an effective trajectory data compression solution that can improve the scalability of moving object databases is highly desirable.

In this study, we propose a novel algorithm, called Inter-Frame Coding (IFC), for trajectory compression. The IFC scheme exploits the spatial and temporal localities between contiguous data points on a trajectory, and compresses data by reducing the amount of redundant information in the raw spatial-temporal data points. Unlike existing approaches [17, 25, 31], the proposed scheme is simple and *lossless*; and it can support data calculation and database queries directly using the compressed database. Thus, IFC has the potential to facilitate large-scale storage, compression, and computation of trajectory data.

Using trajectory datasets collected by the two real-world systems, namely the Taipei eBus system [6] and the Microsoft GeoLife Project [1], we evaluate the proposed scheme in terms of the data compression ratio and the execution time for range queries and k-nearest neighbor searches. Based on the results, we draw the following conclusions.

1) The IFC scheme achieves a high compression rate in large-scale databases. In our evaluations, the IFC scheme achieved compression ratios of 49% and 58% on the TPE eBus and Microsoft GeoLife datasets respectively.
2) The scheme can reduce the computational complexity of range queries by a factor of 0.28 in the TPE scenario and 0.45 in the GeoLife scenario. Moreover, the performance gain is consistent regardless of the query range, as long as the database is sufficiently large.
3) Although the computational complexity of the IFC scheme is greater than that of the non-IFC approach for k-nearest neighbor searches, its execution time is still moderate and affordable. Moreover, the execution time is consistent regardless of the k value.
4) The IFC scheme is simple and ready for immediate real-world deployment. In addition, it can be implemented easily in conjunction with unequal erasure protection and data prioritization schemes for operations in lossy or resource-constrained environments.

The remainder of this paper is organized as follows. In Section II, we present the IFC algorithm and an analysis of the proposed scheme. In Section III, we describe the IFC-based implementation of two popular database queries, namely range queries and k-nearest neighbor searches. We provide a comprehensive set of evaluation results based on realistic trajectory datasets in Section IV, and discuss several issues related to

TABLE I
EXAMPLES OF MOBILE OBJECTS

Mobility	Speed (km/hr)	Movement (5 sec)	Movement (1 min)
Airplane	800	1,111.11 m	13,333.33 m
Car	100	138.89 m	1,666.67 m
Bus	60	83.33 m	1,000.00 m
Pedestrian	5	6.94 m	83.33 m

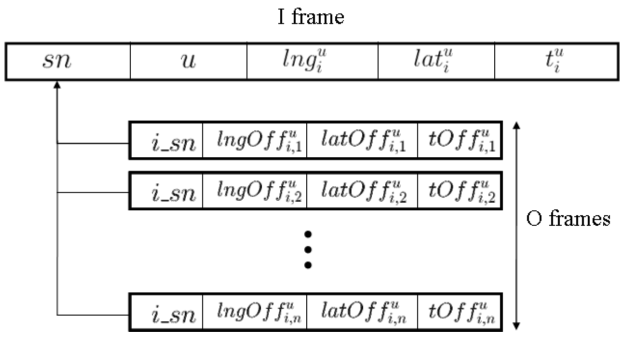


Fig. 1. An illustration of I frames and O frames in the IFC scheme.

the proposed scheme in Section V. In Section VI, we review related works on trajectory data management. We then summarize our conclusions in Section VII.

II. INTER-FRAME CODING

A. The IFC Algorithm

The rationale for this study is based on the observation that *spatial and temporal localities are common in a trajectory*, and *the spatial and temporal offsets between any two contiguous data points in a trajectory are limited to the object's mobility and the trajectory's sampling rate*, as shown in Table I. Thus, we propose a novel trajectory compression algorithm, called *Inter-Frame Coding* (IFC), to exploit the spatial and temporal localities of contiguous spatial-temporal data points, thereby reducing the amount of redundant information in the raw trajectory data.

As shown in Figure 1, there are two types of data points in the IFC scheme: I frames, which contain the *index* data points of a trajectory; and O frames, which contain the *offsets* of the subsequent data points that correspond to the I frames. Let $\mathcal{I}_i^u$ denote the i-th I frame that represents the v-th spatial-temporal data point of the u-th trajectory (i.e., $\mathcal{T}_v^u$); and let $\mathcal{O}_{i,j}^u$ denote the j-th O frame associated with $\mathcal{I}_i^u$, i.e., the offset of $\mathcal{T}_{v+j}^u$ to $\mathcal{T}_v^u$.

Specifically, $\mathcal{I}_i^u = (sn, u, lng_i^u, lat_i^u, t_i^u)$, where sn is the sequence number of $\mathcal{I}_i^u$; u is the trajectory

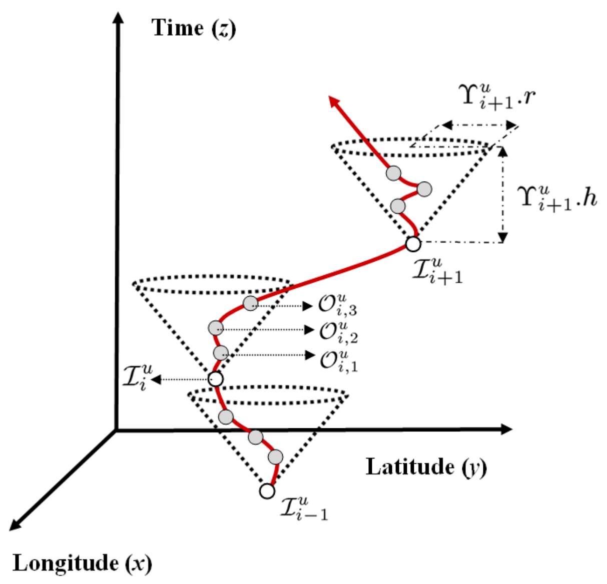


Fig. 2. The inverted right circular cones of $\mathcal{I}^u_{i-1}$, $\mathcal{I}^u_i$, and $\mathcal{I}^u_{i+1}$ in the projected 3D spatial-temporal space. ($n = 3$)

identifier; and lng_i, lat_i, and t_i are the longitude, latitude, and timestamp of $\mathcal{T}^u_v$ respectively. Meanwhile, $\mathcal{O}^u_{i,j} = (i\_sn, lngOff^u_{i,j}, latOff^u_{i,j}, tOff^u_{i,j})$, where i_sn is the sequence number of the I frame that $\mathcal{O}^u_{i,j}$ is associated with (i.e., $i\_sn = \mathcal{I}^u_i.sn$); and $lngOff^u_{i,j}$, $latOff^u_{i,j}$, and $tOff^u_{i,j}$ represent, respectively, the longitude, latitude, and time offsets of $\mathcal{T}^u_{v+j}$ to $\mathcal{T}^u_v$.

Generally, an I frame is associated with n O frames. The value of n is a system parameter tunable based on several factors, such as the sampling rate of the trajectory data, the speed of the moving object, and the data compression ratio required for the application (which we discuss in detail in the next subsection). However, when the offset values exceed the range allowed in an O frame[1], a new I frame must be created, even though the number of O frames associated with the former I frame is less than n.

We project trajectory data points in a 3D spatial-temporal space, where the x, y, and z axes represent the longitude, latitude, and time respectively. The maximum possible space of O-frame data points associated with $\mathcal{I}^u_i$ form an *inverted right circular cone* Υ^u_i, which is rooted at $\mathcal{I}^u_i$ with a height $\Upsilon^u_i.h$ equal to the maximum temporal offset and a radius $\Upsilon^u_i.r$ equal to the maximum spatial offset to $\mathcal{I}^u_i$. Moreover, we know that, $\Upsilon^u_i.r = V_{max} \times \Upsilon^u_i.h$, where V_{max} is the maximum speed of a moving object in the database. Figure 2 shows three contiguous I frames and their associated O frames ($n = 3$) in the projected 3D spatial-temporal space.

[1]For instance, the offset values may exceed the range allowed in an O frame if, in the data stream, a long period containing data points is missed due to transmission errors or GPS errors.

TABLE II
AN EXAMPLE OF A TRAJECTORY'S RAW SPATIAL-TEMPORAL DATA ($u = 1$)

sn	u	lng	lat	t
1	1	121.493710	25.048517	2010/4/26 20:55
2	1	121.493463	25.048624	2010/4/26 20:56
3	1	121.493334	25.048689	2010/4/26 20:57
4	1	121.493222	25.048785	2010/4/26 20:58
5	1	121.493098	25.048715	2010/4/26 20:59
6	1	121.492926	25.048898	2010/4/26 21:00
7	1	121.153431	23.042658	2010/4/27 13:23
8	1	121.153476	25.042723	2010/4/27 13:24
9	1	121.153546	25.042754	2010/4/27 13:25
10	1	121.153721	25.042818	2010/4/27 13:27

TABLE III
THE I FRAMES OF THE TRAJECTORY SHOWN IN TABLE II

sn	u	lng	lat	t
1	1	121.493710	25.048517	2010/4/26 20:55
2	1	121.493098	25.048715	2010/4/26 20:59
3	1	121.153431	23.042658	2010/4/27 13:23

TABLE IV
THE O FRAMES OF THE TRAJECTORY SHOWN IN TABLE II

i_sn	$lngOff$	$latOff$	$tOff$
1	-25	11	1
1	-38	17	2
1	-49	27	3
2	-17	18	1
3	4	7	1
3	11	10	2
3	29	16	4

Table II shows an example of ten contiguous spatial-temporal data points contributed by a random user's GPS logger. Given the IFC configuration $n = 3$, the ten data points are IFC encoded into three I frames and seven O frames, as shown in Tables III and IV respectively. More precisely, in this example, the second I frame is created because n O frames have been created for the first I frame; and the third I frame is created because the longitude and latitude offsets between the 7th data point and the second I frame (i.e., the 5th data point) exceed the maximum offset value allowed for an O frame.

B. Performance Analysis

In this subsection, we use analysis to examine the intrinsic properties of the proposed IFC scheme. We assume that 1) the maximum possible speed of a moving object along lines of latitude and lines of longitude is V_{max} meters per second; 2) the trajectory data is collected at a rate of s data points per second; 3) the maximum value of the latitude and the longitude offsets is Max_dist_offset; and 4) the maximum value of the time offset is Max_time_offset. We know that

$$V_{max} \times \frac{n}{s} \leq Max\_dist\_offset \text{ , and} \tag{1}$$

$$\frac{n}{s} \leq Max\_time\_offset. \tag{2}$$

Therefore, we can obtain the upper bound of n by Equation 3. We also know that the maximum possible value of n is 29 for airplane trajectories (i.e., $V_{max} = 800$ km/hr) and 393 for bus trajectories (i.e., $V_{max} = 60$ km/hr), when $s = 0.2$ (i.e., one data point every 5 seconds). However, when $s = 1/60$ (i.e., one data point every minute), the maximum possible value of n is 2 for airplane trajectories and 33 for bus trajectories.

$$n \leq Min\left(\frac{Max\_dist\_offset \times s}{V_{max}}, Max\_time\_offset \times s\right). \tag{3}$$

In addition, we define the *compression ratio* Ψ as the ratio of the data size using the IFC scheme over the data size without using the scheme, i.e., in the raw data format. Since the size of an I frame is $Size\_I$, the size of an O frame is $Size\_O$, and each I frame is associated with n O frames at most, we can obtain the value of Ψ by

$$\begin{aligned} \Psi &= \frac{Size\_I + n \times Size\_O}{Size\_I \times (n+1)} \\ &= \frac{Size\_O}{Size\_I} + \frac{Size\_I - Size\_O}{Size\_I \times (n+1)}. \end{aligned} \tag{4}$$

Thus, the higher the value of n, the smaller will be the compression ratio Ψ achieved. The 'best' compression ratio is $\frac{Size\_O}{Size\_I}$, which can be achieved when n approaches infinity. However, even though a large n can yield a high compression ratio, which is desirable, we find that a very large n value is infeasible in the IFC scheme. There are two reasons for this phenomenon. First, when n is very large, O frames are used to store most spatial-temporal data points. This is computationally expensive for spatial-temporal applications because (i) each data query in the application layer involves two separate database queries; and (ii), restoring the O frames to the raw data format requires a tremendous amount of computation.

Moreover, if n is very large, the loss of a single I frame may result in contiguous data losses of the original data points, as I frames are crucial for restoring the original data points of O frames.

The second reason is that, the larger the value of n, the greater the likelihood that an I frame will have less than n O frames in the database. In other words, when n is large, the subsequent n points of an I frame are more likely to have oversized offset values that can not be represented by O frames. As a result, a new I frame will be created, even though the number of O frames associated with the former I frame is less than n. Hence, the IFC scheme cannot achieve the theoretical compression ratio when n is large.

In addition to satisfying the constraint in Equation 3, we suggest that, to be efficient, the value of n in the IFC scheme should not be larger than 10. At this value, the scheme can achieve a good compression ratio for storage efficiency, and it is computationally efficient. We discuss these aspects in detail in the evaluation section.

III. IFC-based Database Queries

In this section, we describe the processing of two classical spatial queries, namely range queries (RQ) and k-nearest neighbor (kNN) searches over the proposed IFC scheme. Let R_s be the reference location; R_t be the reference time; and the longitude and the latitude of R_s be $R_s.lng$ and $R_s.lat$ respectively.

A. Range Queries

We define a range query **RQ(**B_s**,**B_t**,**R_t**,**R_s**)** in DEFINITION 1, where B_s and B_t determine, respectively, the spatial and temporal area specified in the query.

Definition 1: The query **RQ(**B_s**,**B_t**,**R_t**,**R_s**)** seeks to find all distinct trajectories crossing the square area that is within B_s distance (along the lines of latitude or longitude) of the reference location R_s in the B_t time units prior to the reference time R_t.

Specifically, when projecting trajectory data points in a 3D spatial-temporal space, the query range space forms a $2B_s \times 2B_s \times B_t$ *square cuboid* centered at the reference location R_s and the time $R_t - \frac{B_t}{2}$, as shown in Figure 3. There are two cases where the u-th trajectory meets the criteria of the range query:

Case 1: There exists an I frame within B_s distance (along the lines of latitude or longitude) of R_s no more than B_t time units before R_t.

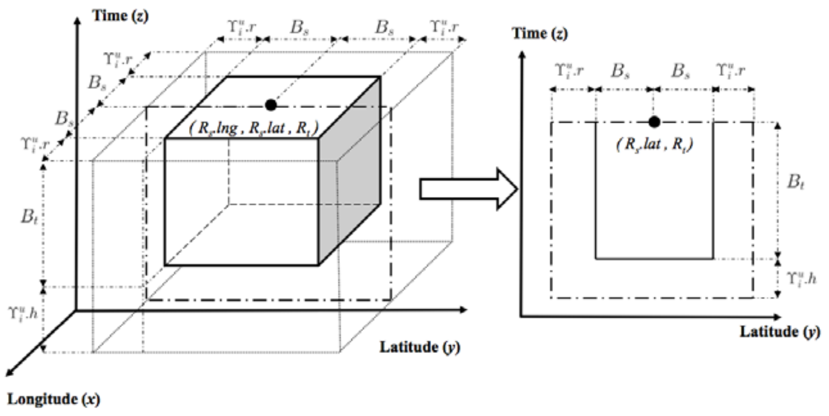


Fig. 3. The projected square cuboid of the range query RQ(B_s,B_t,R_t,R_s) in the 3D spatial-temporal space, and its cross-section in the 2D latitude-time space.

Case 2:There exists an I frame that is not within the query range, but it has at least one O-frame data point within the range.

The first case can be processed in the same way as the conventional method (i.e., without the IFC scheme), and the u-th trajectory meets the criteria of the range query if there exists an i such that

$$\begin{cases} R_s.lng - B_s \le lng_i^u \le R_s.lng + B_s; \\ R_s.lat - B_s \le lat_i^u \le R_s.lat + B_s; \text{ and} \\ R_t - B_t \le t_i^u \le R_t. \end{cases} \tag{5}$$

However, in the second case, it may be necessary to examine all I frames that are not within the query range, as well as their associated O frames. This is computationally expensive and unaffordable when the database is very large. To resolve this problem, we propose a simple algorithm that can identify the set of candidate I frames that may have O frames within the query range, thereby avoiding extensive lookups of all I frames in the databases. Using the projected 3D spatial-temporal space, we regard the i-th I frame of the u-th trajectory $\mathcal{I}_i^u$ as a candidate I frame if and only if there is an intersection between Υ_i^u and the query range space; that is, if $\mathcal{I}_i^u$ satisfies the criteria of either Equation 6 or Equation 7.

$$\begin{cases} R_s.lng - B_s - \Upsilon_i^u.r \le lng_i^u \le R_s.lng + B_s + \Upsilon_i^u.r; \\ R_s.lat - B_s - \Upsilon_i^u.r \le lat_i^u \le R_s.lat + B_s + \Upsilon_i^u.r; \text{ and} \\ R_t - B_t - \Upsilon_i^u.h \le t_i^u < R_t - B_t. \end{cases} \tag{6}$$

$$\begin{cases} B_s < |lng_i^u - R_s.lng| \leq B_s + \Upsilon_i^u.r; \\ B_s < |lat_i^u - R_s.lat| \leq B_s + \Upsilon_i^u.r; \text{ and} \\ R_t - B_t \leq t_i^u < R_t. \end{cases} \tag{7}$$

B. k-*Nearest Neighbor Searches*

We formally define the k-nearest neighbor search k**NN(**k**,**R_s**,**R_t**)** in DEFINITION 2.

Definition 2: **kNN(**k**,**R_s**,**R_t**)** is a database lookup that finds the top k spatial nearest objects to the reference location R_s at the reference time R_t.

Answering the query k**NN(**k**,**R_s**,**R_t**)** involves three steps: 1) find the location of each moving object (i.e., the trajectory) at time R_t; 2) calculate the distance between R_s and the object's location at time R_t; and 3) sort the distances in ascending order and take the corresponding objects of the first k distance as the result. Let $L(u, R_t)$ denote the location of the object's u-th trajectory at time R_t. There are two cases related to obtaining $L(u, R_t)$.

Case 1: There exists a j such that the timestamp of the i-th I frame of the u-th trajectory is exactly equal to R_t (i.e., $t_i^u = R_t$). In this case, $L(u, R_t) = (lng_i^u, lat_i^u)$.

Case 2: If the u-th trajectory does not have any I frames with timestamp values equal to R_t, we find the two adjacent I frames that have the largest timestamp values prior to R_t. Suppose the two I frames are $\mathcal{I}_{i-1}^u$ and $\mathcal{I}_i^u$; then we can calculate $L(u, R_t)$ in the following three subcases[2].

Case 2a: If two or more O frames are associated with $\mathcal{I}_i^u$ (say the last two O frames are $\mathcal{O}_{i,j-1}^u$ and $\mathcal{O}_{i,j}^u$), we obtain the location $L(u, R_t)$ by

$$\begin{aligned} L(u, R_t) &= (lng_i^u + (R_t - t_i^u - tOff_{i,j}^u)\times \\ &\tfrac{lngOff_{i,j}^u - lngOff_{i,j-1}^u}{tOff_{i,j}^u - tOff_{i,j-1}^u}, \\ &lat_i^u + (R_t - t_i^u - tOff_{i,j}^u)\times \\ &\tfrac{latOff_{i,j}^u - latOff_{i,j-1}^u}{tOff_{i,j}^u - tOff_{i,j-1}^u}). \end{aligned} \tag{8}$$

Case 2b: If only one O frame is associated with $\mathcal{I}_j$, we obtain the location $L(u, R_t)$ by

$$\begin{aligned} L(u, R_t) &= (lng_i^u + \tfrac{lngOff_{i,1}^u(R_t - t_i^u - tOff_{i,1}^u)}{tOff_{i,1}^u}, \\ &lat_i^u + \tfrac{latOff_{i,1}^u(R_t - t_i^u - tOff_{i,1}^u)}{tOff_{i,1}^u}). \end{aligned} \tag{9}$$

[2]The location $L(u, R_t)$ can be obtained by either interpolation or extrapolation. However, for simplicity, we only consider the interpolation approach in this study.

Case 2: If no O frames are associated with $\mathcal{I}_i^u$, we let $\mathcal{O}_{i-1,j}^u$ denote the last O frame associated with $\mathcal{I}_{i-1}^u$; then, we can calculate $L(u, R_t)$ by

$$\begin{aligned} L(u,R_t) &= (lng_i^u + (R_t - t_i^u)\times \\ &\quad \tfrac{lng_i^u - (lng_{i-1}^u + lngOff_{i-1,j}^u)}{t_i^u - (t_{i-1}^u + tOff_{i-1,j}^u)}, \\ &\quad lat_i^u + (R_t - t_i^u)\times \\ &\quad \tfrac{lat_i^u - (lat_{i-1}^u + latOff_{i-1,j}^u)}{t_i^u - (t_{i-1}^u + tOff_{i-1,j}^u)}). \end{aligned} \tag{10}$$

If no O frames are associated with $\mathcal{I}_{i-1}^u$, we obtain $L(u, R_t)$ by

$$\begin{aligned} L(u,R_t) &= (lng_i^u + \tfrac{(lng_i^u - lng_{i-1}^u)(R_t - t_i^u)}{t_i^u - t_{i-1}^u}, \\ &\quad lat_i^u + \tfrac{(lat_i^u - lat_{i-1}^u)(R_t - t_i^u)}{t_i^u - t_{i-1}^u}). \end{aligned} \tag{11}$$

After obtaining the location of each moving object at time R_t, we calculate the Euclidean distance between $L(u, R_t)$ and R_s for all trajectories in the database. Then, k**NN(**k**,**R_s**,**R_t**)** reports the k trajectories that have the smallest distance to R_s at time R_t.

IV. Evaluation

Next, we evaluate the proposed scheme in terms of the data compression ratio and the scheme's ability to support database queries (i.e., range queries and k-nearest neighbor searches). Using the open-source PostgreSQL database (version 8.4.4) [4] and the PostGIS spatial database extension (version 1.5.1) [3], we implement the two GIS applications in the form of stored procedures. We then measure the execution time of the database queries using the EXPLAIN command [5] on a BSD machine, which has two Intel Xeon 2.0 GHz CPUs, a 16GB RAM, and the FreeBSD 8.0 OS.

For I frame data in the IFC scheme, we store the location information (lng and lat) using the *Point* data type (16 bytes), the time information (t) using the *Timestamp* data type (8 bytes), and the sequence number (sn) and the trajectory identifier (u) using the *Integer* data type (4 bytes). For O frames, we store the I frame sequence number (i_sn) using the *Integer* data type; and the longitude, latitude, and time offsets $lngOff$, $latOff$, and $tOff$) all use the *Short Integer* data type (2 bytes). Therefore, the size of an I frame ($Size\_I$) is 32 bytes, and the size of an O frame ($Size\_O$) is 10 bytes. Note that the distance offsets (i.e., $lngOff$ and $latOff$) are measured in meters[3], and the time offset (i.e., $tOff$) is measured

[3]For simplicity, we approximate the distance between two data points along lines of latitude and lines of longitude by one meter every 10^{-5} degree, which is valid at sea level on the Equator.

TABLE V
THE BASIC PROPERTIES OF THE TWO DATASETS USED IN THIS STUDY

	TPE	GeoLife
Data source	Taipei e-bus System	GeoLife Project, Microsoft Research Asia
Duration	22 days (2010/04/01 - 2010/04/23)	1,129 days (2007/04/13 - 2010/05/15)
Coverage	greater Taipei area, Taiwan	mostly in Beijing, China
# of distinct moving objects	4,028	104
trajectory types	bus	bus, car, bike, and walk
avg # of trajectories per day	3,865	12
avg # of data points per day	3,235,460	24,020
avg # of data points per trajectory per day	836	1.517
data resolution	about 1 minute	about 5 seconds

in seconds. In addition, we use B-tree and Generalized Search Tree (GiST) [2] to index the temporal and spatial data in this study. We present the evaluation results in the following subsections.

A. Dataset

As mentioned earlier, we evaluate the proposed scheme using two real-world trajectory datasets, namely the Taipei eBus Dataset (TPE) [6] and the GeoLife Dataset (GeoLife) [1]. Table V summarizes the basic properties of the two datasets.

The TPE dataset was provided by the *Taipei e-bus system*, which was deployed by the Taipei City Government in 2004. In the system, each participating bus has an on-board unit (OBU), which is a thin-client with a GPS receiver and a GPRS interface. The OBU transmits the bus's information (the bus identifier, GPS location, and status codes) to the network control center (NCC) via the GPRS connection periodically (every 15 $\sim$ 25 seconds). In 2010, the e-bus deployment involved 4,028 buses covering 287 routes and 15 operating agencies. The system covers nearly the entire greater Taipei area (i.e., Taipei city, Taipei county, and Keelung City), and there are more than 180 million passenger-trips every day. Starting on 04/01/2010, we used the system's API to download real-time bus data every minute for a period of 22 days. On average, there were 3,865 trajectories and 3,235,460 data points each day.

The GeoLife dataset is provided by the GeoLife project of Microsoft Research Asia [37, 38]. It was compiled from data collected by volunteers using different GPS loggers and GPS-phones. As a result, it is very diverse in terms of the sampling rate (ranging from 2 to 5 seconds per point), mobility type (including driving, walking, hiking, and cycling), and demography. Although the dataset spans over 30 cities in China, the USA, and Europe, most of the data relates to the Beijing area. The collection period was approximately three years (from April 13, 2007 to May 15, 2010), and involved 104 distinct moving

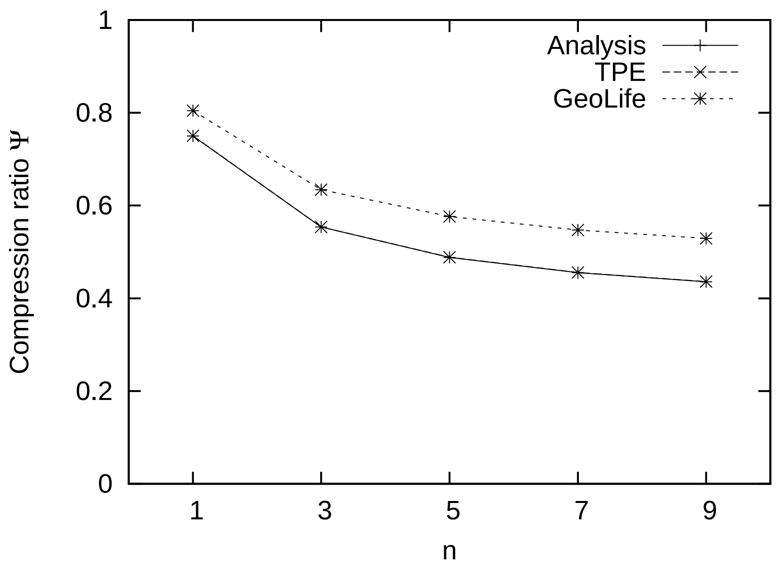
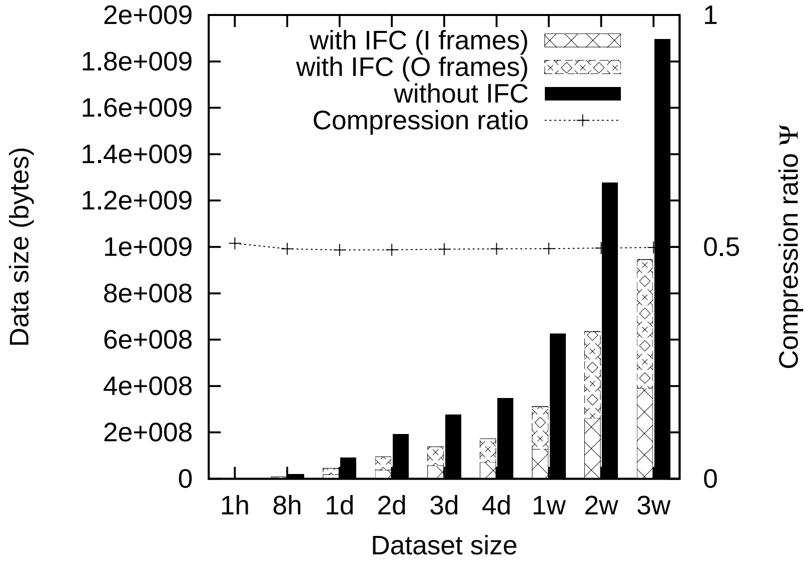


Fig. 4. Comparison of the theoretical compression ratio and the compression ratio achieved using the full-length TPE and GeoLife datasets under the IFC scheme with different n values.

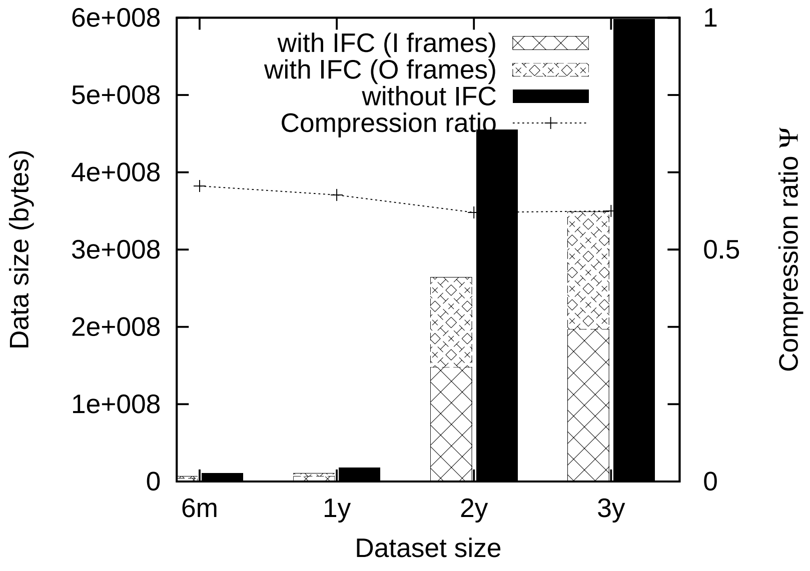
objects.

In Figure 4, we compare the theoretical compression ratio (cf. Equation 4) and the ratio achieved in the TPE and GeoLife scenarios with different n values under the IFC scheme. We observe that the GeoLife curve is consistently higher than the other two curves, and the TPE curve and the theoretical compression ratio curve are overlapped completely. The reason is that the GeoLife dataset is comprised of a large number of short trajectories contributed by volunteers on a daily basis, while the TPE dataset contains a set of uninterrupted trajectories contributed by the Taipei e-buses on a day-and-night basis. Consequently, the GeoLife dataset has a higher percentage of I frames that have less than n O frames due to the truncation of the trajectory data collection, resulting in a higher compression ratio than the theoretical value. By contrast, in the TPE dataset, most of the I frames have complete n frames, and only a limited number of I frames have less than n O frames due to the truncation of the data collection. Therefore, the TPE dataset can achieve a comparable compression ratio to the theoretical ratio.

Figure 5 compares the storage required, in terms of the byte size and the compression ratio, before and after applying the IFC scheme ($n = 5$) with different lengths of the TPE and GeoLife datasets. We observe that the proposed scheme can achieve a compression ratio of approximately 49.5% in all test cases, which represents a 50.5% storage saving. The compression ratio is slightly higher than the theoretical value (i.e., 48.81%) because it is inevitable that some I frames have fewer than n O frames in the compressed dataset.



(a) TPE dataset



(a) GeoLife dataset

Fig. 5. Comparison of the size of the two datasets (for different time periods) with and without the IFC scheme ($n = 5$).

For instance, the number of data points in a trajectory may not be a multiple of $(n+1)$, so there will be fewer O frames for the last I frame. Moreover, most trajectories are related to urban areas, which have poor GPS reception due to buildings and other obstacles; therefore, an I frame may not have exactly n O frames if one data point floats away from the maximum offset value allowed for an O frame. Since the I frame is larger than the O frame, the compression ratio becomes larger if some I frames do not have n O frames in the dataset.

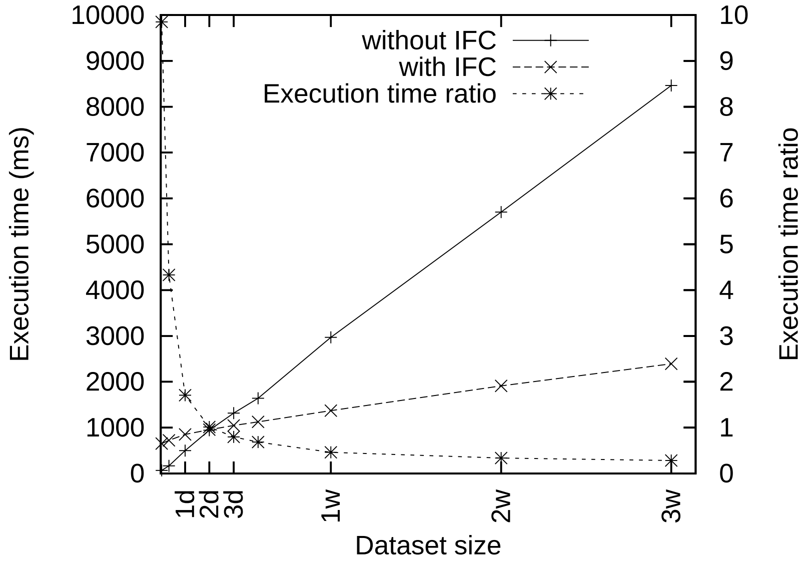
B. Range Queries

We evaluate the proposed IFC scheme to support range queries. For simplicity, we set R_t as the timestamp of the latest data point of the dataset. In addition, we set R_s as the center of one of the 100 predefined cells (which we explain below) for each dataset used in the evaluation. All the results are based on the average performance using the 100 different R_s.

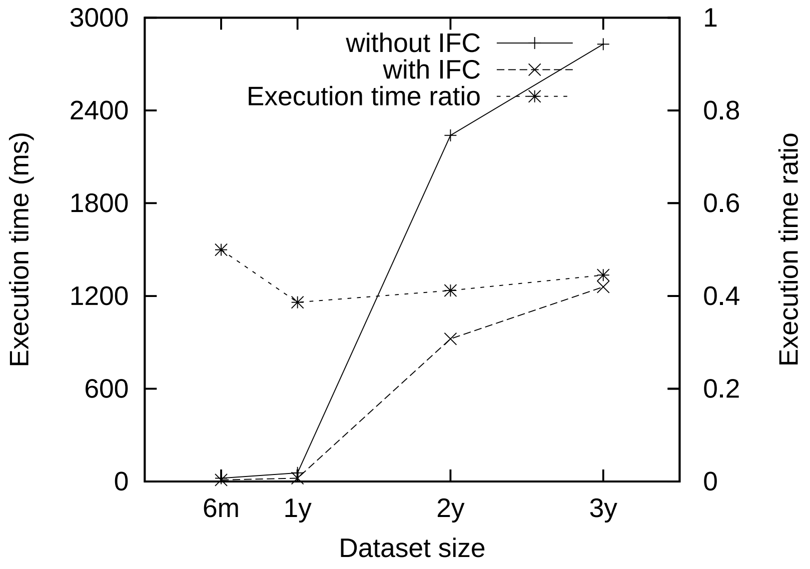
For the TPE dataset, we use the map of central Taipei (north to Taipei SongShan Airport, east to Taipei City Hall, south to National Taiwan University of Technology, and west to Taipei Main Station). The area covers approximately 30 sq. km (6 km from east to west, and 5 km from south to north), and we divide it into a 10 $\times$ 10 equal-sized cells. Similarly, for the GeoLife dataset, we use the map of Beijing (north to Shang Qing Qiao Qu, east to Yuan Tong Qiao, south to Nan Zhong Zhou Lu, and west to Ya Men Kou Qiao). The area covers approximately 812 sq. km (28 km from east to west, and 29 km from south to north), and we divide it into a 10 $\times$ 10 equal-sized cells.

First, we evaluate the impact of the dataset's size on the evaluation time of range queries when the values of B_s and B_t are fixed at *1 km* and *5 minutes* respectively. From the results shown in Figure 6, we observe that the *execution time ratio*, i.e., the execution time when the scalable data scheme is applied over that when the original data format is used, decreases as the time period covered by the dataset increases. Specifically, the ratio is less than 1 when the time period of the TPE dataset is more than 2 days, and it is consistently less than 1 in the GeoLife case. Moreover, the ratio tends to converge to the values of 0.28 and 0.45 in the TPE and GeoLife datasets respectively, i.e. *the execution time of range query operations is reduced by more than 50%*. The results demonstrate that the proposed data scheme can support range queries more efficiently than the conventional approach, as long as the dataset is large enough. Fortunately, this is not a problem for durable applications in reality.

Next, using the TPE and GeoLife datasets, we apply the scalable data format with the B_t value fixed at *5 minutes*, and evaluate its execution time ratio using different-sized datasets and various B_s values. The results shown in Figure 7 confirm our previous findings that the larger the dataset, the greater will be the reduction in the execution time (i.e., a lower execution time ratio is achieved). In addition, the results show that, overall, the execution time ratio tends to increase with the value of B_s; however, the curves drop after B_s becomes larger than a certain value when the dataset size is less than 1 day in the TPE scenario. The reason is that, as B_s increases, the query must examine a larger number of data



(a) TPE dataset

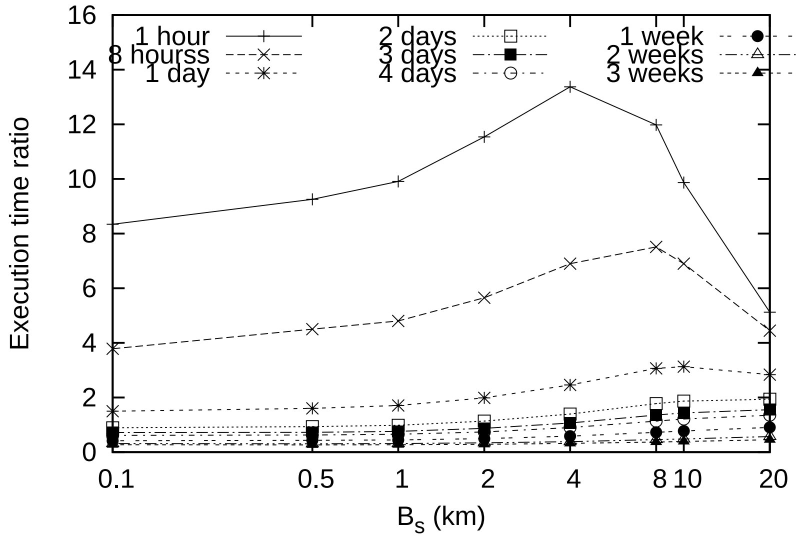


(b) GeoLife dataset

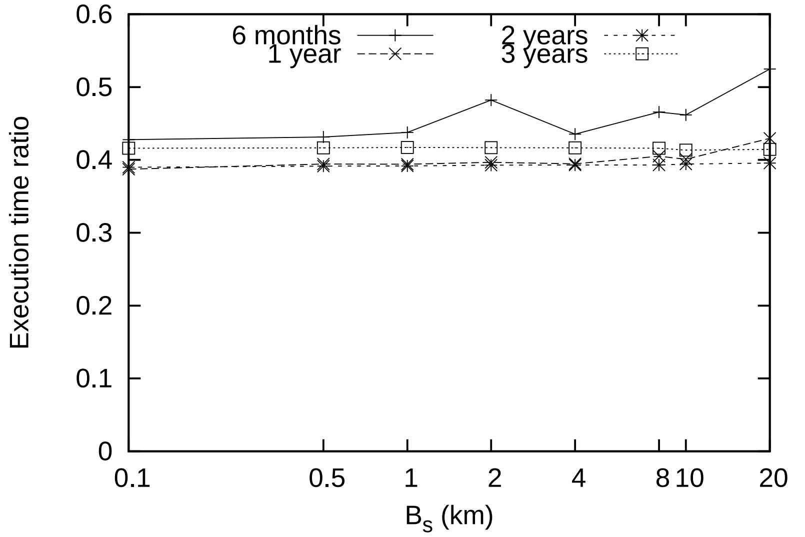
Fig. 6. Comparison of the execution times of range query operations ($B_s = 1$ km and $B_t = 5$ minutes) using different-sized datasets with and without the IFC scheme.

points, which involves an increased number of I frame and O frame lookups, resulting in a higher time complexity. However, after B_s becomes larger than a certain threshold, the number of data points involved in the query is considerable, given the size of the dataset. As a consequence, the time complexity of the range query is reduced because of the B-tree index used for the data points. Note that the threshold value depends on the dataset's size; in general, the larger the dataset, the greater will be the threshold value.

In addition, we fix the B_s value at 1 km, and evaluate the proposed scheme with different B_t values using the TPE and GeoLife datasets. Again, the results in Figure 7 confirm our previous findings that,



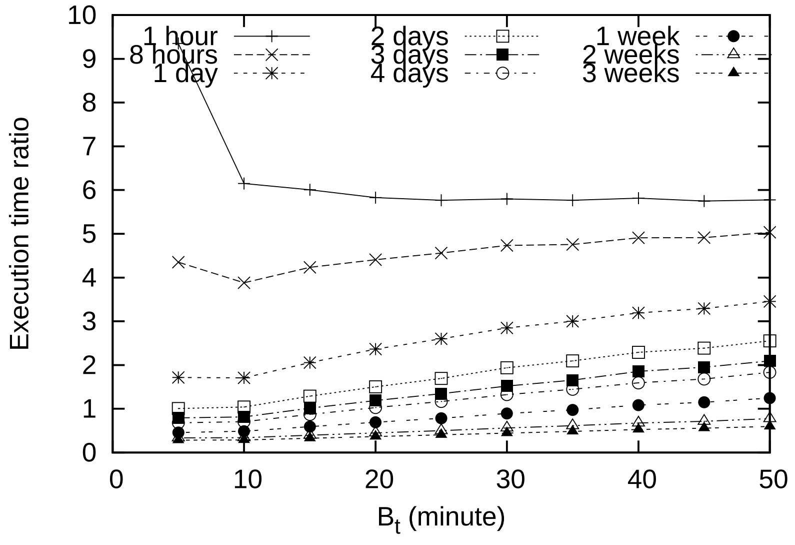
(a) TPE dataset



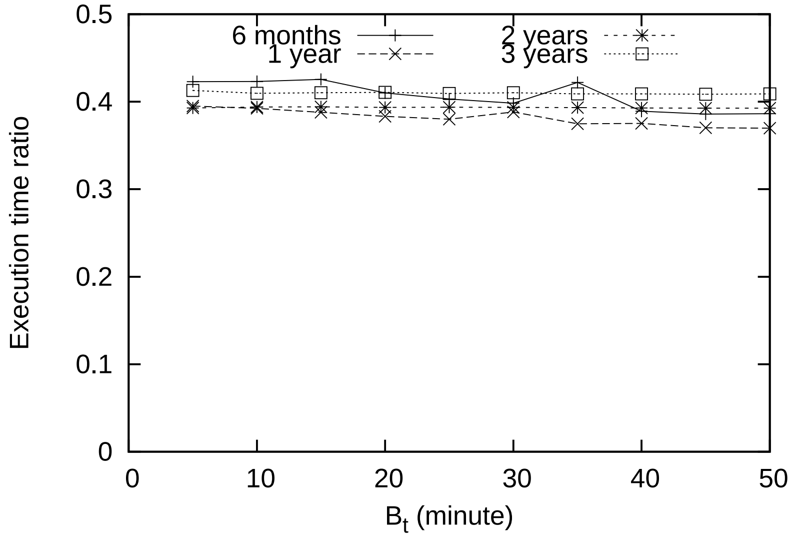
(b) GeoLife dataset

Fig. 7. Comparison of the execution times of range query operations ($B_t = 5$ minutes) with different-sized datasets and B_s values in the logarithmic scale under the IFC scheme.

for both datasets, the execution time ratio improves with the dataset size, regardless of the value of B_t. Moreover, the execution time ratio is stable under different B_t values in the GeoLife case. However, in the TPE dataset, the ratio increases with the B_t values, except that the curve of the 1-hour TPE dataset drops initially and than remains stable when the value of B_t increases. The reason is that the query must examine a larger number of data points as the value of B_t increases, which involves a greater number of data lookups for both I frames and O frames, resulting in higher time complexity. Meanwhile, when the 1-hour TPE dataset is used, the number of data points involved in the query is considerable, given the



(a) TPE dataset



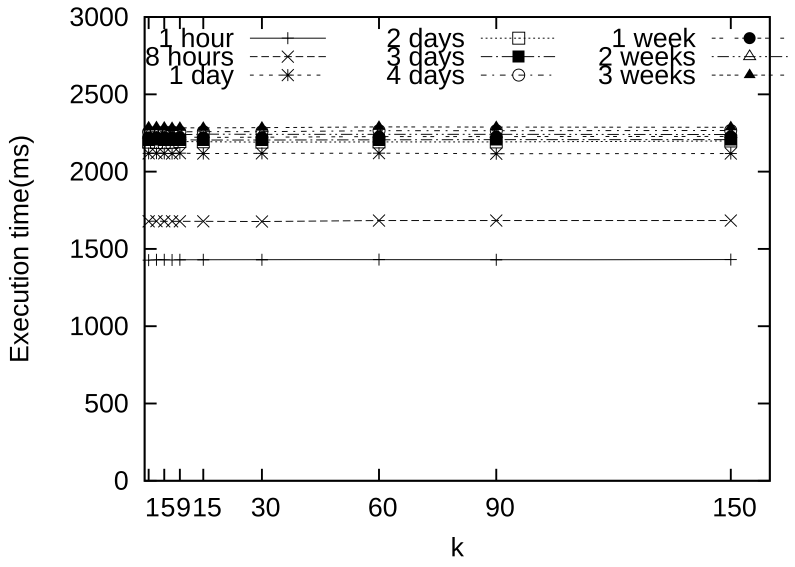
(b) GeoLife dataset

Fig. 8. Comparison of the execution times of range query operations ($B_s = 1$ km) with different-sized datasets and B_t values under the IFC scheme.

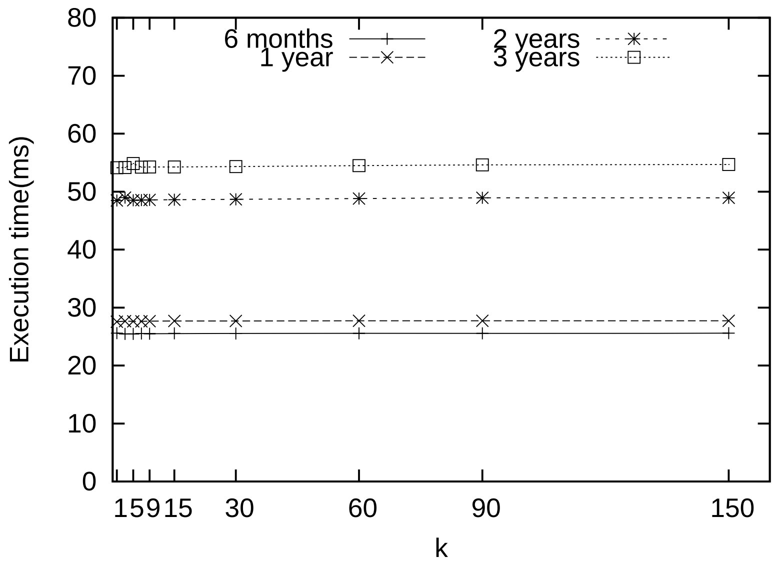
size of the dataset; thus, the time complexity of the range query declines because of the GiST index used for the data points.

C. k-Nearest Neighbor Searches

We also evaluate the proposed scheme's ability to support k-nearest neighbor (kNN) searches. Similar to the previous evaluation, we set R_t as the timestamp of the latest data point in the dataset, and set R_s as the center of one of the 100 predefined cells for both TPE and GeoLife datasets. All the results are



(a) TPE dataset



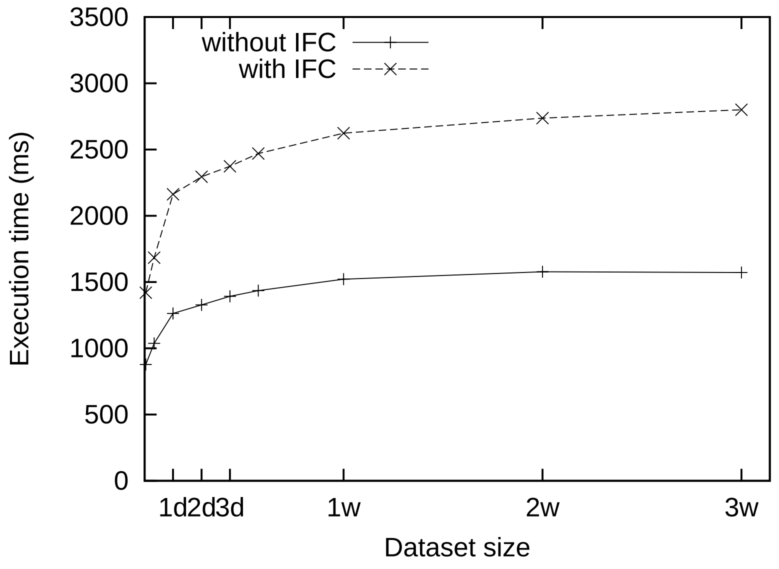
(b) GeoLife dataset

Fig. 9. Comparison of the execution times of k-NN searches with different k values under the IFC scheme.

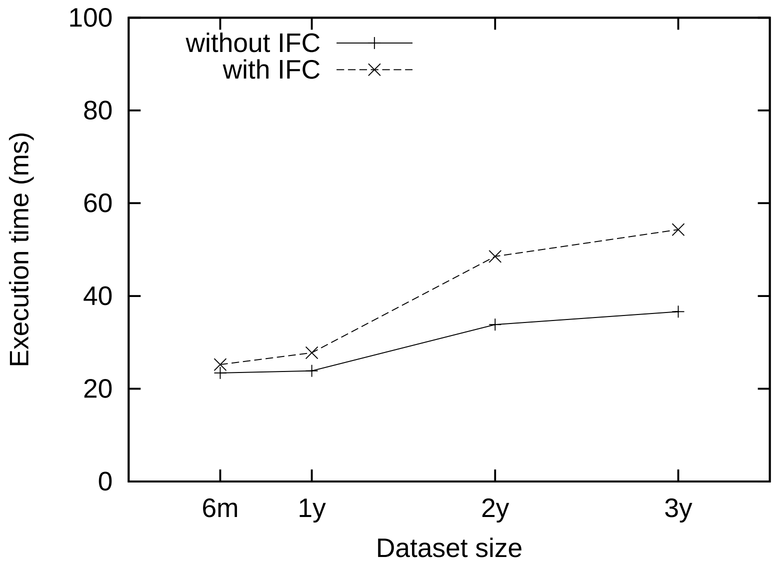
based on the average performance using the 100 different R_s.

From the results shown in Figure 9, we observe that the execution time of kNN searches is consistent regardless of the value of k. The results also show that the larger dataset used, the longer will be the execution time. The reason is quite straightforward: as the search space increases with the size of the dataset, it takes longer to perform kNN searches.

Moreover, the results in Figure 10 demonstrate that the IFC scheme requires a consistently longer execution time than the original scheme (i.e., using raw data format). The execution time ratio is about



(a) TPE dataset



(b) GeoLife dataset

Fig. 10. Comparison of the execution times of k-NN searches with and without the proposed IFC scheme when $k = 5$ under different-sized TPE and GeoLife datasets.

1.5 in all test cases. The reason is that, when performing kNN searches, the IFC scheme has to restore the latest data point by doing two database lookups (i.e., one for the latest I frame, and the other for the last O frame associated with the latest I frame). In contrast, the non-IFC scheme can retrieve the latest data point by only doing lookups on the raw dataset. Even so, we note that the execution time of kNN searches is still affordable under the IFC scheme. For instance, the results in Figure 10 show that the execution time is less than 3 seconds under the TPE dataset and less than 60 milliseconds under the GeoLife dataset.

V. Discussion

In this section, we consider a number of issues related to the proposed IFC scheme that require further investigation.

First, the performance of an IFC-based moving object database depends to a great extent on the maximum number of O frames that can be associated with an I frame (i.e., the n value). However, rather than adopt a 'one-size-fits-all' solution (i.e., apply a fixed n value to all trajectories), it is desirable to adapt the value of n in accordance with the mobility (i.e., V_{max}) and the sampling rate (i.e., s) of a trajectory, especially when there is heterogeneity among the trajectories in the database. Similarly, to improve the computational efficiency of range queries, it is desirable to have tighter settings for the height and the radius of the inverted right circular cone (i.e., Υ_i^h and Υ_i^r for Υ_i, which represents the maximum possible space for O-frame data points associated with the i-th I frame in the projected 3D spatial-temporal space). We defer a detailed discussion of this issue to a future work.

Second, the proposed IFC scheme only has two layers (i.e., the I frame layer and the O frame layer); however, the concept can be extended to support multiple layered coding, which would be beneficial in scalable GIS-based services, i.e., GIS applications that require different resolutions. For instance, when plotting trajectories on a map, the IFC-enabled scheme only requires lookups of I frames when the scale of the map is several kilometers, but it requires lookups of both I frames and O frames when the scale of the map is several meters. Due to the space limitation of the paper, we do not discuss this aspect in detail.

Finally, the IFC scheme can be implemented in conjunction with *unequal erasure protection* (UEP) [10], which provides different levels of erasure protection to the layered trajectory data, depending on its essentiality, by adding various amounts of redundancy. In the IFC scheme, since O frames cannot be interpreted without I frames, I frames are more important and should be given more protection/redundancy. The IFC scheme can also be implemented in conjunction with *data prioritization* [21, 23] to give priority to I frames when sending trajectory data in lossy or low-bandwidth networks. Again, we defer a detailed discussion of this issue to a future work.

VI. Related Work

A Trajectory Database (TD) is a moving object database in which a number of moving objects change their locations over time [32]. The field of trajectory data management has been studied extensively in the last ten years. For instance, a substantial amount of research effort has been invested in the indexing of moving object databases [8, 19, 24], and several data structures, such as r*-tree [7], TPR-tree [30], and SETI [9], have been proposed to facilitate the indexing of spatial-temporal data.

Using moving object databases, a number of approaches have been proposed to improve the handling of basic database queries of historical data (a.k.a. *instantaneous queries*) [27, 29] and trajectory prediction (a.k.a. *continuous queries*) [11, 33, 34]. In addition, several advanced functions have been developed to support moving object databases in emerging location-based service (LBS) and geographic information system (GIS) applications, such as similarity search [16, 20, 28, 35] and nearest neighbor search [13–15, 18]. Nergiz [26] and Jin [22] also studied the security and privacy issues related to trajectory-based moving object databases.

Meanwhile, several trajectory compression algorithms have been proposed to improve the scalability of moving object databases by exploiting the spatial and temporal localities in trajectory data [17, 25, 31]. For instance, Meratnia et al. proposed the *time ratio algorithm* [25], which discards a data point on a trajectory, as long as that data point can be interpolated/extrapolated by any two adjacent data points on the trajectory within an error threshold. Gudmundsson et al. [17] proposed a fast implementation of the well-known *Douglas-Peucker* line simplification algorithm [12] that can reduce the computational complexity from $O(n^2)$ to $O(nlog^k n)$. Finally, Schmid et al. [31] proposed *semantic trajectory compression* (STC), which achieves its compression rate by replacing raw spatial-temporal data points with a semantic representation of the trajectory comprised of a sequence of events. However, the main drawback of these approaches is that they achieve trajectory compression by means of a lossy compression approach. Thus, information loss is inevitable, and applications built on top of the approaches are dependent on the accuracy of the compressed trajectory data.

VII. Conclusion

In this paper, we propose the Inter-Frame Coding algorithm (IFC) for lossless compression of trajectory data. We have also implemented two database queries, i.e., the range query and the k-nearest neighbor search, based on the proposed scheme. Using realistic datasets compiled by two real-world systems, we evaluated the proposed IFC scheme and verified that it can achieve a compression ratio of 58%. The results also demonstrate that the IFC scheme can effectively reduce the execution time of range queries; and it only requires a moderate execution time for k-nearest neighbor searches. The scheme is simple, lossless, efficient, and extensible with advance features (e.g., unequal erasure protection and data prioritization). Thus, we believe that it could facilitate the development of trajectory databases and future location-aware services.

References

[1] GeoLife: Building social networks using human location history. http://research.microsoft.com/en-us/projects/geolife/.

[2] GIST: Generalized Search Tree. http://gist.cs.berkeley.edu/.

[3] PostGIS. http://postgis.refractions.net/.

[4] PostgreSQL Database. http://www.postgresql.org/.

[5] PostgreSQL EXPLAIN Command. http://www.postgresql.org/docs/current/static/sql-explain.html.

[6] Taipei e-bus system. http://www.e-bus.taipei.gov.tw/index.htm.

[7] N. Beckmann, H. peter Begel, R. Schneider, and B. Seeger. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *International Conference on Mangament of Data*, 1990.

[8] C.-I. Cha, S.-W. Kim, J.-I. Won, J. Lee, and D.-H. Ba. Efficient Indexing in Trajectory Databases. *International Journal of Database Theory and Application*, 1(1):21–28, 2008.

[9] V. P. Chakka, V. Prasad, C. Adam, A. C. Everspaugh, and J. M. Patel. Indexing Large Trajectory Data Sets With SETI. In *International Conference on Biennial Conference on Innovative Data Systems Research*, 2003.

[10] P. A. Chou, H. J. Wang, and V. N. Padmanabhan. Layered multiple description coding. In *IEEE Packet Video Workshop*, 2003.

[11] H. DivyakantAgrawal and AmrElAbbadi. Storage and Retrieval of Moving Objects. In *MDM*, 2001.

[12] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[13] X. feng Liu, Y. sheng Liu, and Y. yuan Xiao. K Nearest Neighbors Search for the Trajectory of

Moving Object. In *International Conference on Wireless Communications, Networking and Mobile Computing*, 2005.

[14] E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis. Nearest Neighbor Search on Moving Object Trajectories. In *International Symposium on Spatial and Temporal Databases*, 2005.

[15] E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis. Algorithms for Nearest Neighbor Search on Moving Object Trajectories. *GeoInformatica*, 11:159–193, 2007.

[16] E. Frentzos, K. Gratsias, and Y. Theodoridis. Index-based Most Similar Trajectory Search. In *IEEE International Conference on Data Engineering*, 2007.

[17] J. Gudmundsson, J. Katajainen, D. Merrick, C. Ong, and T. Wolle. Compressing spatio-temporal trajectories. In *International Conference on Algorithms and Computation*, 2007.

[18] R. H. Guting, T. Behr, and J. Xu. Efficient k-nearest neighbor search on moving object trajectories. *The VLDB Journal*, In press.

[19] M. Hadjieleftheriou, G. Kollios, D. Gunopulos, and V. J. Tsotras. Efficient Indexing of Spatiotemporal Objects. In *International Conference on Extending Database Technology: Advances in Database Technology*, 2002.

[20] H. Jeung, M. Lung, Y. Xiaofang, Z. Christian, S. Jensen, and H. T. Shen. Discovery of Convoys in Trajectory Databases. In *VLDB*, 2008.

[21] Z. Jiang and L. Kleinrock. A Packet Selection Algorithm for Adaptive Transmission of Smoothed Video Over a Wireless Channel. *Journal of Parallel and Distributed Computing*, 60(4):494–509, April 2000.

[22] X. Jin, Z. Zhang, J. Wang, and D. Li. Watermarking Spatial Trajectory Database. In *International Conference on Database Systems for Advanced Applications*, 2005.

[23] R. Kapoor, M. Cesana, and M. Gerla. Link layer support for streaming MPEG video over wireless links. In *IEEE ICCCN*, 2003.

[24] G. Kollios, V. J. Tsotras, D. Gunopulos, A. Delis, and M. Hadjieleftheriou. Indexing Animated Objects Using Spatiotemporal Access Methods. *IEEE Transactions on Knowledge and Data Engineering*, 13(5):758–777, September 2001.

[25] N. Meratnia and R. A. de By. Spatiotemporal Compression Techniques for Moving Point Objects. In *EDBT*, 2004.

[26] M. E. Nergiz, M. Atzori, Y. Saygin, and B. Guc. Towards Trajectory Anonymization: a Generalization-Based Approach. *Transactions on Data Privacy*, 2(1):47–75, 2009.

[27] D. Papadias, Y. Tao, J. Zhang, N. Mamoulis, Q. Shen, and J. Sun. Indexing and Retrieval of Historical Aggregate Information about Moving Objects. *IEEE Data Engineering Bulletin*, 25(2):10–17, March 2002.

[28] N. Pelekis, I. Kopanakis, G. Marketos, I. Ntoutsi, G. Andrienko, and Y. Theodoridis. Similarity Search in Trajectory Databases. In *IEEE International Symposium on Temporal Representation and Reasoning*, 2007.

[29] K. Porkaew, I. Lazaridis, and S. Mehrotra. Querying Mobile Objects in Spatio-Temporal Databases. In *International Symposium on Advances in Spatial and Temporal Databases*, 2001.

[30] S. Saltenisy, C. S. Jenseny, S. T. Leutenegger, and M. A. Lopez. Indexing the Positions of Continuously Moving Objects. *ACM SIGMOD Record*, 29(2):331–342, June 2000.

[31] F. Schmid, K.-F. Richter, and P. Laube. Semantic Trajectory Compression. In *International Symposium on Advances in Spatial and Temporal Databases*, 2009.

[32] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and Querying Moving Objects. In *IEEE ICDE*, 1997.

[33] Y. Tao, C. Faloutsos, D. Papadias, and B. Liu. Prediction and Indexing of Moving Objects with Unknown Motion Patterns. In *ACM SIGMOD*, 2004.

[34] Y. Tao, D. Papadias, and J. Sun. The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. In *VLDB*, 2003.

[35] E. Tiakas, A. N. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, D. Stojanovic, and S. Djordjevic-Kajan. Trajectory Similarity Search in Spatial Networks. In *IEEE International Database Engineering and Applications Symposium*, 2006.

[36] O. Wolfson. Moving Objects Information Management: The Database Challenge. In *NGITS*, 2002.

[37] Y. Zheng, L. Liu, L. Wang, and X. Xie. Learning Transportation Modes from Raw GPS Data for Geographic Application on the Web. In *WWW*, 2008.

[38] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from GPS trajectories. In *WWW*, 2009.