Curve-Based Representation of Moving Object Trajectories

Byunggu Yu
Department of Computer Science University of Wyoming
yu@uwyo.edu

Seon Ho Kim
Department of Computer Science University of Denver
seonkim@cs.du.edu

Thomas Bailey
Department of Computer Science University of Wyoming
tbailey@uwyo.edu

Ruben Gamboa
Department of Computer Science University of Wyoming
ruben@uwyo.edu

Abstract

In recent years, many emerging database applications deal with continuously moving data objects – each data object moves continuously and frequently reports its current location, moving direction, and speed to the database server. A database server for these applications keeps track of the trajectories of individual moving objects and processes queries referring to the past or future trajectories. Related techniques view a moving object trajectory as a sequence of connected line segments. However, most natural moving objects, such as airplanes, vessels, and vehicles, draw a smooth trajectory with no angles. This paper presents our curve-based trajectory representation models. The presented results show that the curve-based models provide much more accurate trajectories than the line-based models when we have the same amount of data (same number of reported points). In other words, the curve-based models require a smaller amount of data while providing the same accuracy in trajectory representation.

1. Introduction

In recent years, many emerging database applications deal with continuously moving data objects. These applications, which are known as *Moving-objects database* (*MOD*) applications, include mobile communication systems, location-based services (LBS), digital battlefields, transportations, and air- or ground-traffic control systems, to name a few. In MOD applications, each data object moves continuously and frequently reports its current spatiotemporal attribute values (*spatiotemporal records*) representing its current location, moving direction, and speed to the database server. Moving objects may even change their reporting time intervals. The database server must constantly commit a large number of incoming spatiotemporal records to the database. This aspect of MOD applications poses a major challenge for developing a specialized database server that supports real-time processing.

To support large-scale MOD applications, one requires an on-line database server that can store, update, and retrieve large sets of moving objects. Each moving object has both spatiotemporal properties representing the trajectory and non-spatiotemporal properties such as identification, phone number, and address. Conventional database technology can efficiently manage the non-spatiotemporal properties of moving objects and efficiently process queries referring to only non-spatiotemporal properties of moving objects. Therefore, the most important and interesting research issues in designing and implementing a MOD server are storing trajectories of moving objects, updating trajectories of moving objects, and processing queries referring to these trajectories.

A MOD server must be able to keep track of the trajectories of individual moving objects and process queries referring to the past or future trajectories. Most existing techniques view a trajectory as a sequence of connected line segments (*line-based trajectory representation*) in a 3-dimensional or 4-dimensional space-time [1, 4, 5, 6, 8, 9, 10, 11, 12, 13]. However, most natural moving objects, such as airplanes, vessels, vehicles, humans, trains, and animals, draw a smooth trajectory with no angles. This is because they continuously move with momentum. Therefore, representing the trajectory of a moving object as a sequence of connected curve segments (*curve-based trajectory representation*) is more intuitive.

This paper presents our study of trajectory representation models, specifically curve-based trajectory representation of moving objects. Representing the trajectory of a moving object more accurately with a fewer number of reported points is a crucial issue in designing MOD servers because the frequency of trajectory updates is a critical factor in determining the performance of a real-time MOD server. Conventional line-based models using a linear function create trajectories that have angles at reported (factual) locations. Thus it does not represent well the smooth trajectories of moving objects. Our proposed trajectory representation models harness a higher degree polynomial to draw more accurate trajectories in which not only locations but also some important derivatives (e.g., velocity) change smoothly. Our experimental results demonstrate the superiority of the proposed curve-based models over the conventional line-based ones in 1) tracking the actual trajectories, 2) reducing the number of reporting points (trajectory updates).

In recent years, we are witnessing that MOD applications are approaching to the mainstream as GPS (Global Positioning System) devices proliferate [8]. Consequently, a high-performance, scalable MOD server is an important requirement. Our research presented in this paper is well-positioned to address this requirement. Other related applications include national security (e.g., monitoring numerous moving objects near the border lines), transportation security and safety (e.g., per-airport early warning system for airplane-terrain collision), and collision avoidance for orbital space objects.

The rest of this paper is organized as follows. Section 2 presents a classification of existing spatiotemporal update policies, describes conventional line-based trajectory models, and proposes our curve-based trajectory representation models. In Section 3, we quantify our discussion by comparing real trajectory gathered from a GPS device with analytical results from our trajectory models and from conventional line-based models. Conclusions and future research directions are discussed in Section 4.

2. Trajectory representation

While data objects continuously move, their trajectories in the database cannot be continuously updated due to the limited network bandwidth and the server's database update performance. In a conventional database system, if too many transactions updating the database are constantly given (e.g., 1000 transactions every second), data values accumulate in the buffer because the database system cannot commit all the incoming data to the database on the disk (i.e., disk I/O bottleneck). This results in a buffer overflow, after which the system begins losing (or rejecting) the incoming data and is no longer able to process transactions. Thus, each moving object discretely reports the spatiotemporal attribute values (spatiotemporal records) representing its location, direction, and speed. This discrete update can make the trajectory of a moving object stored in a database (*database trajectory*) different from the actual trajectory of the object. A database trajectory is represented by a sequence of connected segments, each of which joins two consecutive reported locations using interpolation (estimation). Each segment is associated with a certain degree of uncertainty representing the deviations (e.g., Euclidian distance) between the points of the segment and the corresponding points of the real trajectory.

2.1. Update policies

Existing discrete update policies can be classified as follows [12]:

(1) *Fixed time-interval* (*FTI*) *update policy*: A *reporting interval* is defined as the time interval between a pair of consecutive updates. Each moving object has a fixed reporting interval x selected in an ad hoc fashion, and sends a spatiotemporal record to the server every x time units. Except for the first spatiotemporal record, each spatiotemporal record contains a valid *uncertainty* value: each moving object estimates its current location using the same technique (the same mathematical equations) as the database server and measures the deviation (Euclidian distance) between the real location and the estimated location every time unit. The maximum deviation found is written in the next spatiotemporal record as the uncertainty and cleared. Thus, the uncertainty value associated with a reported location P_i represents the maximum deviation of the curve segment joining P_{i-1} and P_i.

(2) *Plain dead-reckoning* (*PDR*) *update policy*: Each moving object has the last reported spatiotemporal record and a fixed threshold *th* that is selected in an ad hoc fashion. Each moving object estimates its current database location as the database server does and measures the deviation between the real location and the database location every time unit. When the actual deviation *th'* between the current location and the corresponding database location exceeds *th*, a spatiotemporal record with the uncertainty value *th'* is sent to the database server. Thus, the uncertainty value associated with a reported location P_i represents the maximum deviation of the curve segment joining P_{i-1} and P_i.

(3) *Adoptive dead-reckoning* (*ADR*) *update policy*: Basically, this is the same as PDR update policy except that moving objects can change their update thresholds. Each moving object has the last reported spatiotemporal record and *th*. When a moving object reports a new spatiotemporal record, it can change the threshold value for future updates (the details of this can be found in [12]).

Although in [12, 13], these update policies are used in a database server that keeps track of only the current locations, speeds, and directions of moving objects, the update policies can also be used to keep track of the trajectories of moving objects and to associate a proper uncertainty with each past trajectory segment. Then, processing techniques for queries referring to the past trajectories can produce a result set in which each result item is associated with its likelihood (i.e., the probability that the result item satisfies the given query predicate).

2.2. Linear interpolation and extrapolation

A database trajectory is represented by a sequence of connected segments each of which joins two consecutive reported points. To produce these segments, interpolation schemes can be used. Conventionally, linear interpolation between reported (factual) location-times has been widely used in estimating past trajectories of a moving object. Similarly, future locations are extrapolated by linearly extending the most recently reported velocity. These line-based models create trajectories with angles at joints (i.e., factual location-times), which is unusual in real trajectories of continuously moving objects.

In recent years, several MOD access methods have been designed to support database queries referring to the trajectories of continuously moving data objects. These access methods can be classified into *Past Trajectory Access Methods* (*PTAMs*) and *Future Trajectory Access Methods* (*FTAMs*): while FTAMs [1, 5, 10, 11] are designed for spatiotemporal MOD queries referring to the current or future trajectories of moving objects, PTAMs, such as [4, 8, 9], index the past trajectories of moving objects. All these methods are based on the linear interpolation or extrapolation method, and the uncertainty issue is not considered well.

2.3. Curve-based interpolation and extrapolation

This section presents our first curve-based interpolation and extrapolation approaches that can improve query performance by reducing the number of trajectory update points as well as the number of trajectory segments that are indexed without compromising the accuracy of database trajectory. A trajectory is represented by a sequence of curve segments, rather than line segments, each of which connects two consecutive reported *location-times*. A location-time of a 2-dimensional moving object is a point <*X, Y, TIME*> in a 3-dimensional space-time; a location-time of a 3-dimensional moving object is a point <*X, Y, Z, TIME*> in a 4-dimensional space-time.

We view a trajectory as a spline composed of a sequence of low degree curves (e.g., parametric cubic curves)[1]. There are several families of splines. *B-splines* [7] are used often in computational geometry and computer graphics. However, B-splines do not go through a given set of control points. Since we use reported location-times as control points, trajectory splines must pass through all control points. The *Catmull-Rom spline* [2, 7] has the following desirable characteristics: (1) the spline passes through all the control points; (2) the spline is continuous: the curve segments are joined with C^1 continuity (i.e., the first derivatives of two adjacent curves are equal at the joint). However, the Catmull-Rom spline does not reach the first and the last control points. Given a sequence of control points P_0, P_1, …, P_n, since the slope of the tangent at each control point P_i is the slope of the linear line connecting P_{i-1} and P_{i+1}, the slopes of the tangents at P_0 and P_n cannot be derived.

In our applications, each spatiotemporal record contains not only a location-time (joint) P but also a velocity P' (a vector whose direction represents the moving direction and whose magnitude represents the speed). Because each reported location-time is used as the joint of two adjacent curve segments, each pair of adjacent curve segments has the same velocity (i.e., the first derivative) at the joint. We propose to use a parametric cubic function $P(t) = a_0+a_1t+a_2t^2+a_3t^3$ to obtain a spline that passes through any given sequence of joint-velocity pairs $<<P_0\ P_0'> <P_1\ P_1'> <P_2\ P_2'> \ldots <P_n\ P_n'>>$ where P_i is a location-time in a 3-dimensional or 4-dimensional space-time and P_i' is the velocity at P_i, for all $i = \overline{1,n}$.

Given a pair of two consecutive joint-velocities $<P_i\ P_i'>$ and $<P_j\ P_j'>$, one can derive the coefficients of $P(t) = a_0+a_1t+a_2t^2+a_3t^3$ by solving the following constraints for a_0, a_1, a_2, and a_3: $P(t{=}0) = P_i$; $P(t{=}P_j.TIME\text{-}P_i.TIME) = P_j$; $P'(t{=}0) = P_i'$; $P'(t{=}P_j.TIME\text{-}P_i.TIME) = P_j'$. Substituting these coefficients into the polynomial equation, we have the following function:

$$P(t) = [1 \quad t \quad t^2 \quad t^3]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3/t_x^2 & 3/t_x^2 & -2/t_x & -1/t_x \\ 2/t_x^3 & -2/t_x^3 & 1/t_x^2 & 1/t_x^2 \end{bmatrix}\begin{bmatrix} P_i \\ P_j \\ P_i' \\ P_j' \end{bmatrix},$$

where $P' = <X' = \Delta X/\Delta t,\ Y' = \Delta Y/\Delta t,\ Z' = \Delta Z/\Delta t,\ TIME' = \Delta TIME/\Delta t = 1>$, $t_x = P_j.TIME\text{-}P_i.TIME$, and $0 \le t \le t_x$. (1)

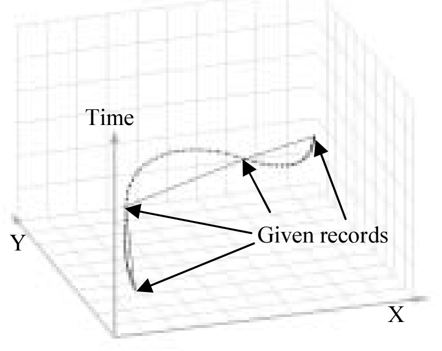
Alternatively, this can be written as follows:

$$P(u) = [1 \quad u \quad u^2 \quad u^3]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}\begin{bmatrix} P_i \\ P_j \\ P_i' \\ P_j' \end{bmatrix},$$

where $P' = <X' = \Delta X/\Delta u,\ Y' = \Delta Y/\Delta u,\ Z' = \Delta Z/\Delta u,\ TIME' = \Delta TIME/\Delta u = P_j.TIME\text{-}P_i.TIME>$, and $0 \le u \le 1$. (2)

[1] A single high-degree curve is not desirable because of the following reasons: (1) the polynomial may take very large values between the points, and the size of these excursions can grow exponentially with the degree N of the polynomial; (2) the polynomial can be very sensitive to small changes in the points [3].

Figure 1 shows the difference between this parametric cubic function and the conventional linear interpolation.



Record#	X	Y	Time	ΔX/Δt	ΔY/Δt	ΔTime/Δt
1	10	10	10	5	50	1
2	20	50	11	100/3	100	1
3	60	50	12	100/3	-100	1
4	70	10	14	50/502	-500/502	1

Figure 1. Interpolating four given spatiotemporal records

To predict the future trajectory of a moving object, conventional linear extrapolation schemes [1, 5, 10, 11] extend the last known velocity or line segment to estimate the future trajectory. This represents a linear movement with a fixed velocity. In our curve-based approach, given a future (or current) point in time T_f, we can extrapolate the location of a moving object by extending the object's trajectory segment connecting the last two reported spatiotemporal records P_{n-1} and P_n to T_f (i.e., $P(t{=}T_f)$). However, this extrapolation must be appropriately modified because curves can quickly deviate from the real trajectory. For example, if the 3rd degree polynomial Equation (1) is used, acceleration changes linearly. This implies that moving objects will keep increasing or decreasing their accelerations at the same rate as time progresses, which is very unlikely for most moving objects.

Without any correction in acceleration, an extrapolation for future trajectory using the 3rd degree polynomial may be much worse than that of the linear extrapolation. Nevertheless, extrapolating the last curve segment is still helpful in estimating the future trajectory of a moving object, since the recent trajectory of the object may represent the momentum that the object has gained. In Physics, a well known, simplest form of momentum (i.e., linear momentum) is $m \times v$, where m and v are mass and velocity, respectively. In addition, the effect of momentum is indirectly proportional to time (i.e., the current momentum gradually disappears as time progresses). As a simple model, we tested the following formula to maintain future trajectories in control by weighting both linear and non-linear factors. The weight W given to $P(t)$ decreases as time progresses and the function $L(t)$ represents the linear extrapolation. Note that, the following function is not yet mature. We designed this simple preliminary formula in order to show that our curve-based trajectory representation model can be used not only for past trajectory interpolation but also for future trajectory extrapolation.

$$F(t{=}T_f) = W*P(t{=}T_f) + (1\text{-}W)*L(t{=}T_f),\ \text{where } W = C/(T_f\text{-}P_{n\text{-}1}.T) \text{ and } L(T_f){=}\ (P_n{+}(T_f\text{-}P_n.T)*P_n') \tag{3}$$

3. Experimental results

3.1. Background

To verify the effectiveness of our proposed curve-based trajectory approach, we have conducted the following experiment. Using a portable GPS device (Trimgle Navigation's ProXRS Receiver with GPS logger), which can record a joint-velocity pair every second, we collected real GPS data. We placed the GPS device in a car and drove from a location near the north boundary of Denver, Colorado, to Loveland, Colorado along the interstate highway 25. Every second, we logged a spatiotemporal data from the GPS device. We believe this trajectory includes both relatively straight road and some winding road, which is useful for a better comparison. Then, we divided the recorded trajectory points into two subsets as shown in Figure 2. Note that Set1 represents driving on a straight road – its trajectory has almost constant velocity. Unlike Set1, the trajectory of Set2 has more noticeable changes in velocity (i.e., direction and speed).

For the comparison between the conventional line-based (linear) model and our curve-based model, we created trajectories based on a subset of logged update points. For each of Set1 and Set2, we randomly selected logged spatiotemporal records with various sampling ratios. Both of the proposed cubic function and the conventional line-based interpolation scheme were used to connect the selected samples (4-dimensional spatiotemporal trajectories were produced). Figure 3 gives magnified views of the circled parts in Figure 2 (the sampling ratio was about 5%; for illustration sake, we projected the 4-dimensional spatiotemporal trajectories onto the XY-plane). Finally, we quantified 1) the actual deviations between the non-sampled real location-times and the corresponding computed points in the curve-based trajectory and 2) the actual deviations between the non-sampled real location-times and the corresponding computed points in the line-based trajectory for the comparison.

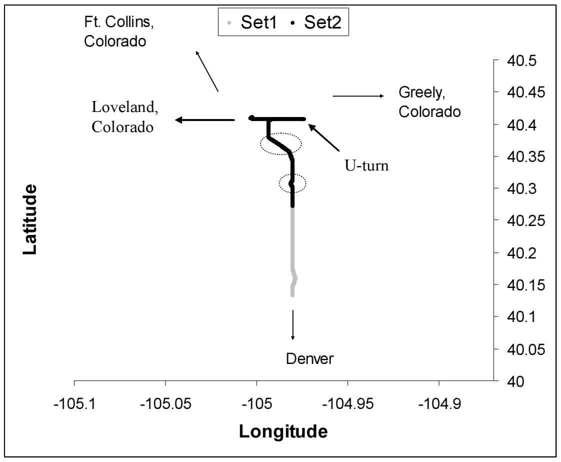


Figure 2. A real trajectory points collected: Set1 consists of 482 spatiotemporal records logged every second; Set2 consists of 742 spatiotemporal records logged every second

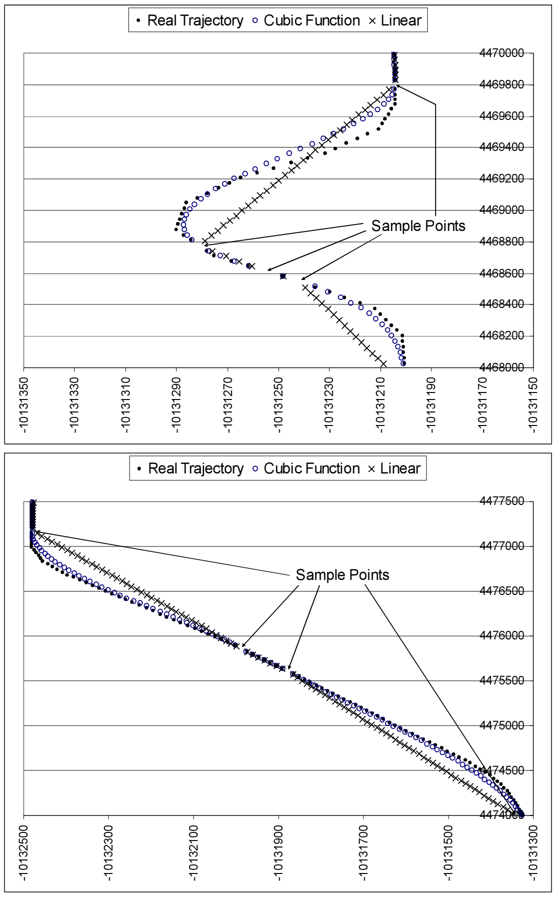
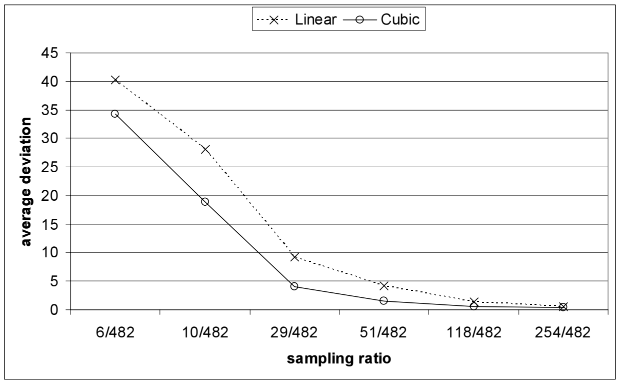


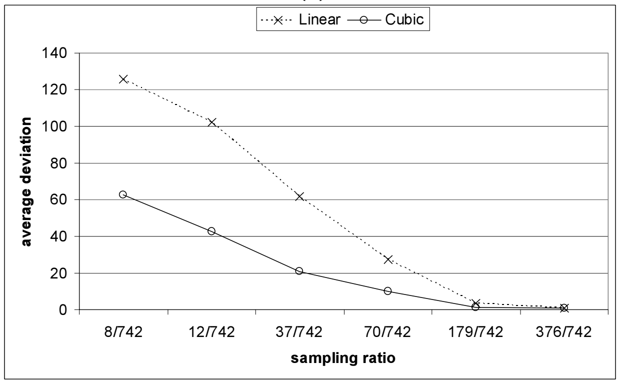
Figure 3. Trajectories projected onto XY-plane: the X-axis is longitude in meters; the Y-axis is latitude in meters; the sampling ratio was 37/742 (≈ 5%)

3.2. Comparison

In all cases (i.e., Sets 1 and 2), the proposed cubic function (i.e., Equations (1)) produced significantly smaller average deviations (up to 3 times smaller, Figure 4) and standard error deviations (Figure 5) than those of the line-based (linear) interpolation scheme. Figure 4 shows the average deviation between a real location and a computed location for various subsets of update points. For example, with 37 out of 742 points in Set2, the average distance deviation of the curve-based trajectory was 21 meters while that of the line-based trajectory was 62 meters, which is approximately 200% greater. The maximum deviation in this section was 231 meters for the curve-based trajectory and 683 meters for the line-based trajectory (again, approximately 200% greater). As shown in Figure 5, the standard deviation in this section was 41 meters for the curve-based trajectory and 134 meters for the line-based trajectory (approximately 230% greater).



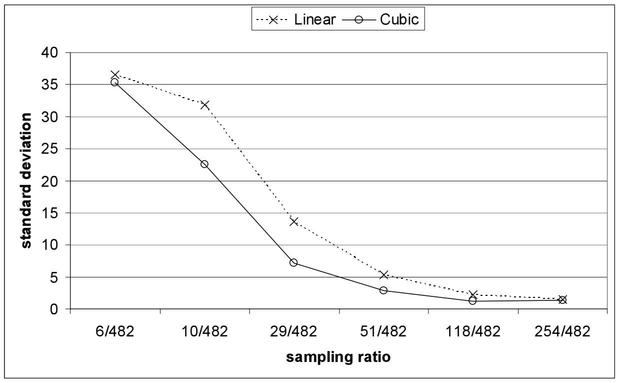
(a)



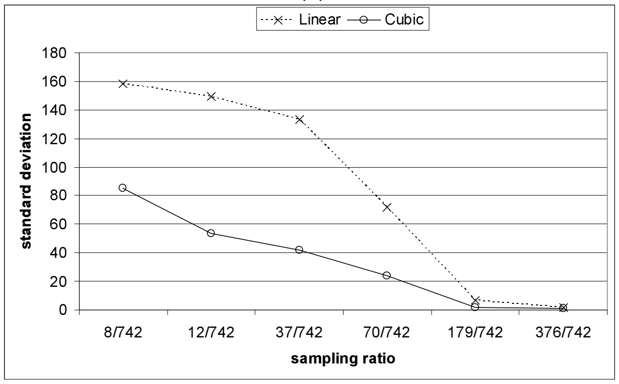
(b)

Figure 4. Average spatial deviations (in meters) with various sampling ratios on (a) Set1 and (b) Set2

Observe that the difference between the line-based trajectory and the curve-based trajectory was smaller when the road is straight, while it was widen where the road is winding (see Figures 4 and 5): Figure 4(a) shows that, given an average-deviation threshold of 2 meters, the linear model requires about 118 updates, and the cubic model requires about 51 updates (about 2.3 times smaller number); Figure 4(b) shows that, given an average-deviation threshold of 62 meters, the linear model requires about 37 updates, and the cubic model requires about 8 updates (a 4.6 times smaller number of updates). In all cases, the curve-based approach excelled the linear approach.



(a)



(b)

Figure 5. Standard error deviations (in meters)

In the next experiment, we used the 3rd degree trajectory extrapolation formula (i.e., Equation (3)) and reused the data in Set2. The constant *C* was set to 0.5 and we assumed Plain Dead-Reckoning (PDR) update policy explained in Section 2.1. Given a fixed threshold *th*, our simulation program counted the updates required by the linear extrapolation scheme $L(t)=(P_n+(t-P_n.T)*P_n')$ and the updates required by the function $F(t)$ (i.e., Equation (3)). Recall that, in PDR update policy, an update is required whenever the estimated location deviates from the corresponding factual location in the GPS device's log by more than *th* (meters). We ran the simulation program with various threshold values. The results summarized in Figure 6 show that the tested function $F(t)$ requires noticeably smaller number of updates than the conventional linear extrapolation $L(t)$. That is, given a maximum deviation allowed, the curve-based trajectory representation can significantly reduce the number of required spatiotemporal records (updates). In other words, the curve-based trajectory representation can more accurately extrapolate future trajectories. Considering that Equation (3) is still preliminary, this result shows the potential of curve-based model in future trajectory extrapolation.

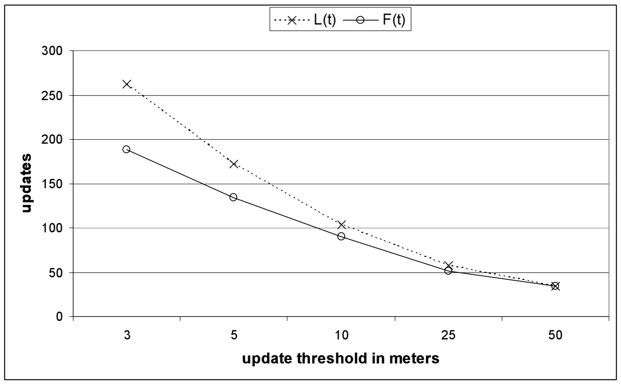


Figure 6. The impacts of the preliminary curve-based approach on future trajectory estimation

4. Discussions and future directions

Given a maximum allowed deviation between a point of a database trajectory and the corresponding point of the real trajectory, our proposed curve-based trajectory representation, which has no angles (i.e., C^1 or higher degree continuity at every joint), may require a smaller number of spatiotemporal records than the line-based trajectory representation. Thus, one can expect the following positive effects: 1) a smaller number of trajectory update transactions per time unit, 2) a reduced amount of secondary storage space occupied by trajectories, 3) a reduced size of trajectory index structures, and 4) a smaller number of disk I/Os in processing trajectory update transactions and trajectory queries. These all result in enhancing the performance of a MOD server. In another aspect, our approach can present more accurate trajectories than the conventional approach with the same given resources.

In the experiments presented in this paper, we designed, implemented, and tested curve-based trajectory models based on a 3rd degree polynomial (a specialized parametric cubic function). When each update gives location and the first derivative (velocity) of the real trajectory, our past trajectory model based on a 3rd degree polynomial can generate visually smooth curve segments (i.e., the location changes smoothly). However, in each segment, the acceleration changes linearly. If each update gives both the first and the second derivatives (i.e., velocity and acceleration), a 5th degree polynomial can be used to generate a curve segment in which location, velocity, and acceleration change smoothly. We can generalize our model by considering higher degree derivatives. We will continue to investigate this generalized trajectory representation. We will consider various types of moving objects (e.g., vehicles, airplanes, vessels, humans, animals) and investigate optimization solutions that, given a proper description of a set of moving objects, can choose the most efficient equation for the objects in the set. We reserve these as our future work.

At this point, a question regarding the trade-offs between CPU overhead and I/O overhead may naturally arise. As we can see in Section 2, our solutions (i.e., Equations (1) and (3)) incur more computation overhead, which requires additional CPU time. According to our experiments (For the experiments, we used Dell Precision 420 Linux workstation equipped with Intel Pentium III 800MHz and 256MB main-memory space), it took the linear model 0.7 – 0.8 microsecond of CPU time to interpolate (compute) a point in-between two consecutive joints. For the same job, the cubic function required 4.3 – 4.6 microseconds of CPU time. Considering large scale applications, incoming spatiotemporal records will form a long, seamless update pipeline, which requires the server to constantly perform disk I/O operations. The curve-based model reduces the I/O overhead without compromising the accuracy of the trajectories. Our experimental results show that the tested cubic function can allow us to reduce the number required updates by a factor of up to 4 or 5. This has much more significant impacts on the system performance and scalability than the CPU overhead: Typically, a single disk access requires several milliseconds. For example, IBM Deskstar 14GPX has an average disk access time of approximately 10 milliseconds. Typically, a main-memory access takes less than 60 nanoseconds. More importantly, it is well-known that CPU and memory performances have been improved at a much faster pace than that of secondary storage devices. Nevertheless, we believe that investigating the relevant issues in developing an adaptive system that can automatically balance the CPU-I/O trade-offs using various trajectory models is very interesting and practically viable, especially for small-size moving object databases backed by large database buffer space and for peculiar MOD systems that have limited CPU power or relatively faster secondary storages. Our future work will cover this issue.

References

[1] P. K. Agarwal, L. Arge, and J. Erickson. Indexing Moving Points. *ACM PODS SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 175-186, 2000.

[2] E. Catmull and R. Rom. A Class of Local Interpolation Splines. *R.E. Barnhill and R.F. Riesenfled, editors, Computer Aided Geometric Design*, Academic Press, New York, 1974.

[3] W. R. Franklin. Applications of Analytical Cartography. *Cartography and Geographic Information Systems*, 2000: *www.ecse.rpi.edu/Homepages/wrf/research/gisapps/gisapps.pdf.*

[4] B. Jun, B. Hong, and B. Yu. Dynamic Splitting Policies of the Adaptive 3DR-tree for Indexing Continuously Moving objects. *Proc. DEXA International Conference on Database and Expert Systems Applications, LNCS Lecture Notes in Computer Science,* Vol. 2736, pp. 308-317, Springer-Verlag, Berlin Hidelberg, 2003.

[5] G. Kollios, D. Gunopulos, V. J. Tsotras. On Indexing Mobile Objects. *ACM PODS SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 261-272, 1999.

[6] D. Kwon, S. Lee, S. Lee. Indexing the Current Positions of Moving objects Using the Lazy Update R-tree. *Third International Conference on Mobile Data Management, January, Singapore*, p. 113, 2002.

[7] On-Line Geomteric Modeling Notes. *Computer Science Department, University of California, Davis,* http://graphics.cs.ucdavis.edu/CAGDNotes/.

[8] D. Pfoser and C. S. Jensen. Querying the Trajectories of On-Line Mobile Objects. *Proc. ACM MobiDE International Workshop on Data Engineering for Wireless and Mobile Access*, pp. 66-73, 2001.

[9] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel Approaches to the Indexing of Moving object Trajectories. *Proc. VLDB Very Large Data Base Conference*, pp. 395-406, 2000.

[10] S. Saltenis, C.S. Jensen, S.T. Leutenegger, and M.A. Lopez. Indexing the Positions of Continuously Moving objects. *ACM SIGMOD International Conference on Management of Data*, pp. 331-342, 2003.

[11] Y. Tao, D. Papadias, and J. Sun. The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. *Proc. VLDB Very Large Data Base*, pp. 790-801, 2003.

[12] O. Wolfson, L. Jiang, A. P. Sistla, S. Chamberlain, N. Rishe, and M. Deng. Databases for Tracking Mobile Units in Real Time. *C. Beeri and P. Buneman, editors, ICDT International Conference on Database Theory, LNCS Lecture Notes in Computer Science*, pp. 169-186, Springer-Verlag, Berlin Hidelberg, 1998.

[13] O. Wolfson, P. Sistla, B. Xu, J. Zhou, and S. Chamberlain. DOMINO: Databases fOr Moving objects traking. *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 547-549, 1999.