The Tornado Model: Uncertainty Model for Continuously Changing Data

Byunggu Yu[1], Seon Ho Kim[2], Shayma Alkobaisi[2],
Wan D. Bae[2] and Thomas Bailey[3]

[1]Computer Science Department, National University
La Jolla, CA 92037, USA
byu@nu.edu
[2]Computer Science Department, University of Denver
Denver, CO 80208, USA
{seonkim, salkobai, wbae}@cs.du.edu
[3]Computer Science Department, University of Wyoming
Laramie, WY 82071, USA
tbailey@uwyo.edu

Abstract. To support emerging database applications that deal with continuously changing (or moving) data objects (CCDOs), such as vehicles, RFIDs, and multi-stimuli sensors, one requires an efficient data management system that can store, update, and retrieve large sets of CCDOs. Although actual CCDOs can continuously change over time, computer systems cannot deal with continuously occurring infinitesimal changes. Thus, in the data management system, each object's spatiotemporal values are associated with a certain degree of uncertainty at virtually every point in time, and the queries are mostly processed over estimates characterizing the uncertainty. The smaller the uncertainty is, the better the query performance becomes. The paper proposes a sophisticated asymmetric uncertainty model, called the Tornado Model, which can effectively represent, process, and minimize the data uncertainty for a wide variety of CCDO database applications.

Keywords: spatiotemporal database, trajectory, uncertainty

1 Introduction

An increasing number of emerging applications deal with a large number of *continuously changing* (or moving) *data objects (CCDO*s), such as vehicles, sensors, and mobile computers. For example, in earth science applications, temperature, wind speed and direction, radio or microwave image, and various other measures (e.g., CO2) associated with a certain geographic region can change continuously. Accordingly, new services and applications dealing with large sets of CCDOs are appearing. In the future, more complex and larger applications that deal with higher dimensional CCDOs (e.g., a moving sensor platform capturing multiple stimuli) will become commonplace – increasingly complex sensor devices will continue to proliferate alongside their potential applications. Efficient support for these CCDO applications will offer significant benefit in many broader challenging areas including mobile databases, satellite image analysis, sensor networks, homeland security, internet security, environmental control, and disease surveillance.

To support large-scale CCDO applications, one requires a data management system that can store, update, and retrieve CCDOs. Each CCDO has both multidimensional-temporal (i.e., 2 or 3D geographic space or other information dimensions that vary with time) properties representing its continuous trajectory in an information space-time continuum as well as non-temporal properties such as identification, associated phone number, and address. Importantly, although CCDOs can continuously move or change (thus drawing continuous trajectories in a space-time), computer systems cannot deal with continuously occurring infinitesimal changes – this would effectively require infinite computational speed and sensor resolution. Thus, each object's continuously changing attribute values (e.g., location, velocity, and acceleration) can only be discretely updated. Hence, they are always associated with a degree of uncertainty, especially when there is a considerable time gap between two updated points.

Over the past years, database research communities have mainly focused on representing the uncertainty of spatiotemporal CCDO locations (i.e., the position property of estimated CCDO states). In contrast, sensor technology has evolved to detect or approximate higher order derivatives (e.g., velocity and acceleration). For example, most GPS loggers in the market are now capable of recording highly accurate velocity (and even acceleration) values. Virtually any intelligent sensory device that can rapidly detect its stimuli can be used to produce the higher order derivatives at the sensor level. By properly utilizing these additional sensory inputs, we can possibly support CCDO queries referring to the higher order derivatives of the trajectories (e.g., report every CCDO that possibly had an acceleration within the given acceleration range at the given point in time). Importantly, these additional inputs can be utilized to minimize the uncertainty of a database trajectory.

The paper presents our research initiated on the following goals: first, provide a basis for developing an uncertainty model that can represent not only the position uncertainty but also the derivative uncertainties; second, investigate the problem of minimizing the uncertainty using the sensor-level derivatives.

The rest of this paper is organized as follows: Section 2 summarizes related techniques and models. Section 3 characterizes CCDOs. Sections 4 and 5 propose and verify our novel trajectory uncertainty model that fulfils the above goals. Section 6 concludes the paper and proposes some future work.

2 Related Work

Several application-specific models of uncertainty have been proposed. One popular uncertainty model is that, at any point in time, the location of an object is within a certain distance *d*, of its last reported location. If the object moves further than *d*, it reports its new location and possibly changes the distance threshold *d* for future updates [12]. Given a point in time, the uncertainty is a circle with a radius *d*, bounding all possible locations of the object.

Another model assumes that an object always moves along straight lines (linear routes). The location of the object at any point in time is within a certain interval, centered at its last reported location, along the line of movement [12]. Different CCDO trajectory models that have no uncertainty consideration are found in the literature [7]. These models make sure that the exact velocity is always known by requiring reports whenever the object's speed or direction changes. Other models assume that an object travels with a known velocity along a straight line, but can deviate from this path by a certain distance [10, 11].

An important study on the issues of uncertainty in the recorded past trajectories (history) of CCDOs is found in [6]. Assuming that the maximum velocity of an object is known, they prove that all possible locations of the object during the time interval between two consecutive observations (reported states) lie on an error ellipse. A complete trajectory of any object is obtained by using linear interpolation between two adjacent states. That is, a trajectory is approximated by a sequence of connected straight lines, each of which connects two consecutively reported CCDO observations. By using the error ellipse, the authors demonstrate how to process uncertainty range queries for trajectories. The error ellipse defined and proved in [6] is the projection of a three-dimensional spatiotemporal uncertainty region onto the two-dimensional data space. Similarly, [4] represents the set of all possible locations based on the intersection of two half cones that constraint the maximum deviation from two known locations. It also introduces multiple granularities to provide multiple views of a moving object.

Our approach, a mechanism that explicitly leverages an understanding and characterization of uncertainty for a generalized case of the CCDO, offers an alterative construct that is suitable for higher dimensional data and that can produce minimally bounding spatiotemporal uncertainty regions for both past and future trajectories of CCDOs by taking into account the temporally-varying higher order derivatives, such as velocity and acceleration. We call this uncertainty model the *Tornado Uncertainty Model* (*TUM*) because, for each reported CCDO state, the model produces a tornado-shaped uncertainty region in the space-time.

3 Explication of CCDO

Before presenting our uncertainty model, we define "Continuously Changing Data Object" (*CCDO*) through a series of ontological abstractions. Table 1 represents our explication of CCDO.

Table 1. Multi-level abstraction of CCDO

Abstraction	Definition
CCDO	A CCDO is a data object consisting of one or more *trajectories* and zero, one, or more non-temporal properties.
trajectory	A trajectory consists of *dynamics* and *f:time* → *snapshot*, where *time* is a past, current, or future point in time.
snapshot	A snapshot is a probability distribution that represents the probability of every possible *state* at a specific point in time. Depending on the *dynamics* and update policies, the probability distribution may or may not be bounded.
state	A state is a point in a multidimensional information space-time of which time is one dimension. Each state associated with zero or more of the following optional properties: velocity (i.e., direction and speed of changes, the 1st derivative), acceleration (the 2nd derivative), and higher order derivatives.
dynamics	The dynamics of a state is a set of domains each of which represents all possible values corresponding to a certain property of the state.

Considering an observer who reports the state of a CCDO as often as possible, the trajectory drawn by the object is viewed as a sequence of connected segments in space-time, and each segment connects two consecutively reported states of the object. Examining of Table 1, one may observe the following: only a subset of *states* can be stored in the database, due to the fact that no database can be continuously updated. We call these stored states *reported states*. Each pair of consecutive reported states of a CCDO represent a single trajectory segment. The reported states are the factual known states of the CCDOs, and only these known states can be committed to the database. All possible in-between states and future states of the CCDOs are then interpolated and extrapolated on the fly when it is necessary (e.g., query processing, data visualization, index maintenance, and data management). Given the theoretical possibility of an infinite number of states between two factual states, a mathematical model and computational approach is required to efficiently manage the 'in-between' and 'future' states.

4 The Tornado Model

As discussed in Section 3, a CCDO consists of a set of conventional non-temporal attributes and one or more temporal attributes (i.e., a location anchor or boundary points) each of which can draw a trajectory over time in a multidimensional attribute space (e.g., geographic space and sensor stimuli space). Among the trajectory components defined in Table 1, only a subset of states, called reported states (*RS*), and a subset of dynamics, which we call known dynamics (*KD*) in this paper, can be stored in a database. Then the trajectory snapshots, which represent the uncertainty of the discretely recorded trajectory, are calculated when necessary. Therefore, in order to formally present our trajectory model, we first explicate a *database trajectory* (a discretely recorded CCDO trajectory stored in a database). As given in Definition 1, a database trajectory consists of a sequence of reported states and some known dynamics.

Definition 1. ***Database Trajectory*:**
A database trajectory ***DBTRAJ*** in a (d+1)-dimensional space-time with d data (or spatial) dimensions and one time dimension consists of a sorted set ***RS*** of n reported states and a sorted set ***KD*** of m known dynamics, where

- For all i=0,..,n-1, RS_i is a tuple <$P^{(0)}$, $P^{(1)}$, … $P^{(k\text{-}1)}$, T, $IME^{(0)}$, $IME^{(1)}$, … $IME^{(k\text{-}1)}$>, where
  - For all l=0,..,k-1, $P^{(l)}$ is a d-dimensional vector representing the lth derivative of the trajectory at T (e.g., $P^{(0)}$, $P^{(1)}$, and $P^{(2)}$ are, respectively, a d-dimensional location, velocity, and acceleration at T);
  - T is a specific time point at which the above state was observed (or sensed);
  - For all l=0,..,k-1, $IME^{(l)}$ is the domain of possible instrument-and-measurement errors (deviations from the real) associated with $P^{(l)}$.
- For all j=0,..,m-1, KD_j is a tuple <$D^{(0)}$, $D^{(1)}$, … $D^{(k)}$, T>, where
  - For all l=0,..,k, $D^{(l)}$ is the domain of the lth derivative of the trajectory at T;
  - T is a specific time point at which the above domains are valid.
- For all i=0,..,n-2, $RS_i.T \leq RS_{i+1}.T$.
- For all j=0,..,m-2, $KD_j.T \leq KD_{j+1}.T$.

Considering the location $P^{(0)}$ to be the 0th derivative, the value of k represents the number of derivatives reported to the database system. For example, for an application wherein sensors can detect and report only the 3-dimensional geographic location of the object each time, d is set to 3 and k is set to 1. If the sensors can report not only locations but also velocities (i.e., $P^{(1)}$), k is set to 2.

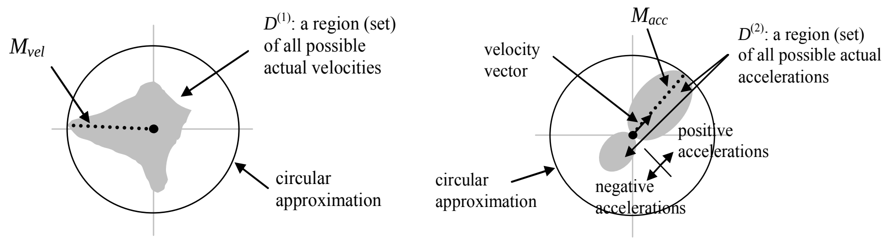
Based on this database trajectory model (i.e., Definition 1), we define our uncertainty model (i.e., *snapshot* defined in Table 1) as given in Definition 2 to calculate the snapshots (uncertainty) of the trajectory given a time point t.

Definition 2. ***Snapshot*** (***Uncertainty Region***)**:**
$SNAPSHOT^{(i)}(DBTRAJ, t)$ can be defined as follows:

$$\begin{cases} E^{(i)}(RS_{l-1}, t) \cap E^{(i)}(RS_l, t) & \text{if } \exists_l (RS_{l-1}.T \le t \le RS_l.T) \\ E^{(i)}(RS_{n-1}, t) & \text{if } RS_{n-1}.T < t \\ E^{(i)}(RS_0, t) & \text{if } RS_0.T > t \\ \phi & \text{otherwise (i.e., no reported state)} \end{cases} \tag{1}$$

, where RS_0 and $RS_{n\text{-}1}$ are, respectively, the first and last reported states of trajectory *DBTRAJ*; $E^{(i)}()$ is a function that takes a reported state *rs* and a time point *t* as input and produces a set of all possible *i*th derivatives of the trajectory at *t*. The calculation of the snapshot falls in one of four cases: (1) *t* is between the times of two consecutive reported states (i.e., $\exists_l (RS_{l-1}.T \le t \le RS_l.T)$); (2) *t* is greater (later) than the last reported state (i.e., $RS_{n-1}.T < t$); (3) *t* is smaller (earlier) than the first reported state (i.e., $RS_0.T > t$); (4) *DBTRAJ* has no reported state.

As shown in Definition 2, one must define the estimation function series *E* in order to fully define this trajectory uncertainty (snapshots) model.


M_{vel}
$D^{(1)}$: a region (set) of all possible actual velocities
circular approximation
M_{acc}
velocity vector
$D^{(2)}$: a region (set) of all possible actual accelerations
positive accelerations
negative accelerations
circular approximation

a. Circular approximation of velocity b. Circular approximation of acceleration

Fig. 1. Examples of the circular approximation

4.1 *E* of Degree 1: Revised Ellipse Model

Definitions 1 and 2 are generalized to accommodate any arbitrary set of instrument-and-measurement errors (*IME*) and dynamics (*D*) each of which can possibly be a highly complicated multidimensional shape. In addition, the model supports any level *k* of derivatives. In practical applications, the parameter *k* should be set to a specific value, and, for efficient calculation of trajectory snapshots, some form of approximation is necessary for both *IME* and *D* components. Let us first consider the following approximation: For every *DBTRAJ*,
(1) k=1;
(2) $\forall_{i=0,..,n-1}$, $RS_i.IME^{(0)}$ is a *d*-dimensional hyper-circle with a constant radius M_{err};
(3) $\forall_{i=0,..,m-1}$, $KD_i.D^{(0)}$ is a *d*-dimensional constant hyper-square *SPACE*;
(4) $\forall_{i=0,..,m-1}$, $KD_i.D^{(1)}$ is a *d*-dimensional hyper-circle with a constant radius M_{vel}.

Note that approximations (2) and (4) represent a circular approximation. Fig. 1a shows an example of the circular approximation of velocity. M_{vel} is the norm of the maximum possible actual velocity. Because of this approximation, possible velocities are independent of the location. Then, $E^{(0)}$ and $E^{(1)}$ can be defined as follows:

$$E^{(1)}(rs,t) = \{p \mid \|p\| \le M_{vel}\} \tag{2}$$

$$E^{(0)}(rs,t) = \{p \mid p \in SPACE \wedge \exists_{iloc,ivel,t'} \left(\begin{array}{l} \|iloc - rs.P^{(0)}\| \le M_{err} \wedge iloc \in SPACE \wedge \\ \|ivel\| \le M_{vel} \wedge \\ 0 \le t' \le |t - rs.T| \wedge \\ p = iloc + ivel \cdot t' \end{array} \right) \} \tag{3}$$

$E^{(1)}$ represents the hyper-circle of all possible velocities and $E^{(0)}$ represents all possible locations that the object, which starts with an initial location *iloc* and any valid constant velocity *ivel*, can reach within the given time interval $|t\text{-}rs.T|$. As an example of this model, consider an object moving through one dimension of space over time. Fig. 2a shows an example of a trajectory segment connecting two reported states of the object. Let RS_0 = $<A, t_i, M_{err}>$, RS_1=$<B, t_j, M_{err}>$ and let M_{vel} be the maximum change rate (i.e., the norm of the maximum possible velocity) of the CCDO. Then all possible states of the CCDO between t_i and t_j are bounded by the lines where $|\cot\theta| = M_{vel}$. The shaded region covers all possible locations of the object between t_i and t_j. The snapshot of the CCDO at any time point *t* that is between t_i and t_j is the cross section of this uncertainty region, produced by the cutting line at *time* = *t*. In this *d*=1 example, $SNAPSHOT^{(1)}$ is $[-M_{vel}, +M_{vel}]$ and $SNAPSHOT^{(0)}$ is
$[A - M_{err} - M_{vel} \cdot |t - t_i|, A + M_{err} + M_{vel} \cdot |t - t_i|] \cap$
$[B - M_{err} - M_{vel} \cdot |t - t_j|, B + M_{err} + M_{vel} \cdot |t - t_j|]$.

Similarly, when a CCDO continuously changes in a two-dimensional space, the snapshots between two consecutive reported states collectively represent the overlapping region of the two funnels (see Fig. 2b). It is important to note that, as shown in Fig. 2b, the projection of the snapshots onto the 2-dimensional data space is, in fact, the uncertainty ellipse that can be defined by the ellipse model [6] with a modification taking into account the instrument and measurement errors.

4.2 *E* of Degree 2: Tornado Uncertainty Model

As discussed in Section 3, in many applications, an accurate sensor-level approximation of trajectory derivatives is possible. For example, most GPS loggers can record not only geographic positions but also corresponding velocity vectors. Hence, *k*=2 holds in the related CCDO applications. This section presents a specialization of the proposed uncertainty model with the following approximations in order to better support *k*=2 applications: For every *DBTRAJ*,

(1) *k*=2;

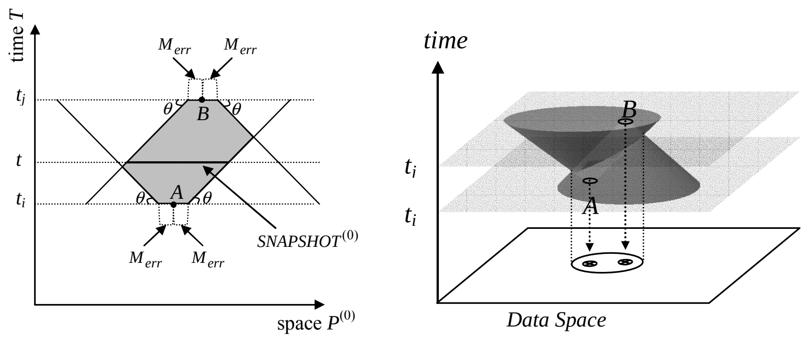
(2) $\forall_{i=0,..,n-1}$, $RS_i.IME^{(0)}$ is a *d*-dimensional hyper-circle with a constant radius $M_{err}^{(0)}$;

(3) $\forall_{i=0,..,n-1}$, $RS_i.IME^{(1)}$ is a *d*-dimensional hyper-circle with a constant radius $M_{err}^{(1)}$;

(4) $\forall_{i=0,..,m-1}$, $KD_i.D^{(0)}$ is a *d*-dimensional constant hyper-square *SPACE*;

(5) $\forall_{i=0,..,m-1}$, $KD_i.D^{(1)}$ is a d-dimensional hyper-circle with a constant radius M_{vel};

(6) $\forall_{i=0,..,m-1}$, $KD_i.D^{(2)}$ is a d-dimensional hyper-circle with a constant radius M_{acc}.



a. In a 2-dimensional space-time b. In a 3-dimensional space-time

Fig. 2. Spatiotemporal uncertainty representing a trajectory segment

Note that this approximation includes a circular approximation of accelerations as shown in Fig. 1b. M_{acc} represents the norm of the maximum possible actual acceleration. Because we use this circular approximation, possible accelerations are independent of the corresponding velocity vector. That is, $E^{(2)}$ is defined as Eq. 4. Then, the estimation functions $E^{(1)}$ and $E^{(0)}$ can be defined as Eqs. 5 and 6:
$E^{(2)}$ represents the constant hyper-circle of all possible accelerations and $E^{(1)}$ represents all possible velocities that the object, which starts with the initial velocity *ivel* and a valid acceleration *iacc*, can reach within the given time interval |*t-rs.T*|. Then, $E^{(0)}$ defines all possible locations the object can reach.

In the last condition of Eq. 6, *iloc*, *ivel*, and *iacc* represent a possible initial location, velocity, and acceleration at *rs.T*. The integral term represents a cumulative location displacement while the object is constantly accelerating at the rate of *iacc*, and the last term represents the location displacement that can be produced by a fixed velocity *ivel*+*iacc*·*t'* during |*t-rs.T*|- *t'*. Note that, unlike the symmetric estimation $E^{(1)}$, $E^{(0)}$ needs to compare *t* and *rs.T*. When $E^{(0)}$ estimates a future snapshot (i.e., $t \geq rs.T$), possible displacements are added to the initial location *iloc* (i.e., adding future possible displacements); for past estimations, possible displacements are subtracted from *iloc* (i.e., canceling out past possible displacements). Note that Eqs. 3 and 5 do not need this differentiation because, given a single reported state *rs*, their snapshots are symmetric to the *d*-dimensional hyper-plane that is perpendicular to the time axis at *rs.T* (i.e., the future snapshots are the past snapshots). The maximum directional displacement in location (i.e., the maximum distance the object can travel in a certain direction during |*t-rs.T*|) is obtained when *iacc* is a boundary point of the acceleration circle $E^{(2)}(rs,t)$ and *ivel*+*iacc*·*t'* is a boundary of the velocity circle $E^{(1)}(rs,t')$.

Because the symmetric circular approximation of acceleration (resp., velocity) fully encloses all possible actual accelerations (resp., velocities), no real object can go beyond the boundary of $E^{(2)}$, $E^{(1)}$, or $E^{(0)}$.

$$E^{(2)}(rs,t) = \{p \mid \|p\| \leq M_{acc}\} \qquad (4)$$

$$E^{(1)}(rs,t) = \{p \mid \|p\| \le M_{vel} \wedge \exists_{ivel,iacc,t'} \begin{pmatrix} \|ivel - rs.P^{(1)}\| \le M_{err}{}^{(1)} \wedge \|ivel\| \le M_{vel} \\ \|iacc\| \le M_{acc} \wedge \\ 0 \le t' \le |t - rs.T| \wedge \\ p = ivel + iacc \cdot t' \end{pmatrix} \} \quad (5)$$

$$E^{(0)}(rs,t) =$$

$$\{p \mid p \in SPACE \wedge \exists_{iloc,ivel,iacc,t'} \begin{pmatrix} \|iloc - rs.P^{(0)}\| \le M_{err}{}^{(0)} \wedge iloc \in SPACE \wedge \\ \|ivel - rs.P^{(1)}\| \le M_{err}{}^{(1)} \wedge \|ivel\| \le M_{vel} \wedge \\ \|iacc\| \le M_{acc} \wedge \\ 0 \le t' \le |t - rs.T| \wedge \\ ivel + iacc \cdot t' \in E^{(1)}(rs,t') \wedge \\ p = \begin{cases} iloc + \int_0^{t'} (ivel + iacc \cdot T)dT + (ivel + iacc \cdot t') \cdot (|t - rs.T| - t') \\ \text{if } t \ge rs.T \\ iloc - \int_0^{t'} (ivel + iacc \cdot T)dT - (ivel + iacc \cdot t') \cdot (|t - rs.T| - t') \\ \text{otherwise} \end{cases} \end{pmatrix} \} \quad (6)$$

For an example, let us assume that a CCDO (a car) moves in two dimensional space from RS_0 (located at x=0.0 and y=0.0 at time 0) to RS_1 (located at x=-1.5 and y=655.80 at time 20) with an initial velocity *ivel* (0.083 meters per second along x axis and 32.34 m/s along y axis). The maximum velocity and acceleration of the car are M_{vel} (50 meters per second) and M_{acc} (2.78 m/s per second), respectively.

Step 1: From RS_0, calculate the maximum possible displacements $E^{(0)}(RS_0\ ,\ t)$ of the CCDO in all directions over time $0 \le t \le 20$. First, we calculate all possible accelerations $E^{(2)}$ using the circular approximation. To discretely represent the boundary of the hyper-circle of $E^{(2)}$, one can choose a certain number of discrete points[1] along the boundary of the hyper-circle with a fixed interval. Then, these points represent the set of all possible maximum accelerations. Then $E^{(1)}$ and $E^{(2)}$ can be calculated by Eq. 5 and Eq. 6, respectively. Fig. 3 example shows the calculated results for $E^{(i)}(RS_0, 6)$. The same process is applied to calculate $E^{(0)}(RS_1, t)$ from RS_1.

Step 2: Two polygons can be created by connecting adjacent points in $E^{(0)}(RS_0, t)$ and $E^{(0)}(RS_1, t)$, respectively. We use the Graham's algorithm [5], which finds the convex hull of a given set of points. The two polygons in Fig. 3b represent the maximum displacements (boundaries) from the two locations, one from RS_0 and the other from RS_1 at any time t between 0 and 20.

Step 3: Quantify the overlapping area of the two intersecting polygons at time t. First, we used the *ray drawing and crossing number algorithm* [5] for each boundary point of one polygon against the other polygon to see if the point is common. Second, the set of points that are common (i.e., overlapping points) were used to create another convex polygon using the Graham's algorithm, which represents the overlapping area of $E^{(0)}(RS_0, t)$ and $E^{(0)}(RS_1, t)$ (shaded area in Fig.3b). Finally, we use the following formula to calculate the

[1] The more points we use to create the polygons the more accurate the estimation of the uncertainty region is. However, it is not practical to use too many points at each time step since it takes a lot of computing time. For our experiments in Sec. 5, we used 100 points.

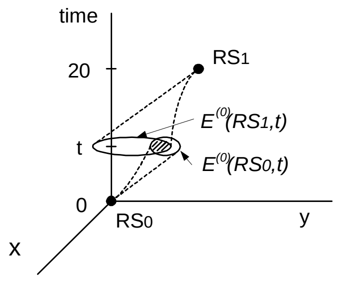
area of the uncertainty region: $A(pol) = \sum_{i=0}^{n-1}(x_i y_{i+1} - y_i x_{i+1})/2$, where $A(pol)$ is the area of the polygon *pol*, and x_i, y_i are the coordinates of a point in *pol*.

Step 4: We repeated Steps 1-3 as a function of time, for example every second, to quantify the overlapping area of the two intersecting polygons at a certain time. The summation of all the areas over the whole interval, from time 0 to 20, is the uncertainty volume.

Similarly, the uncertainty volume of the revised ellipse model can be quantified using Eqs. 2 and 3.

$E^{(2)}$		$E^{(1)}$		$E^{(0)}$	
x	*y*	*x*	*y*	*x*	*y*
1.8	1.3	12.7	41.5	32.0	217.3
0.7	2.1	4.90	44.4	12.5	232.4
-0.7	2.1	-4.7	44.4	-11.5	232.4
-1.8	1.3	-12.5	41.5	-31.0	217.3
-2.2	-0.0	-15.5	32.3	-38.5	194.4
-1.8	-1.3	-12.5	23.2	-31.0	171.5
-0.7	-2.1	-4.7	17.6	-11.5	157.3
0.7	-2.1	4.9	17.6	12.5	157.3
1.8	-1.3	12.7	23.2	32.0	171.5
2.2	0.0	15.6	32.3	39.5	194.4

a. Calculation of $E^{(i)}(RS_0, 6)$



b. Illustration

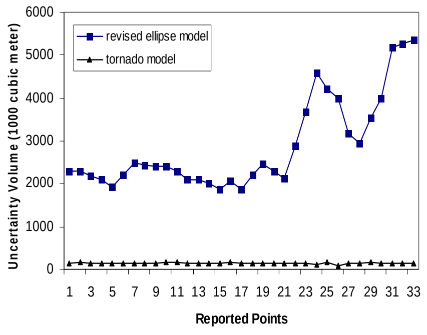
Fig. 3. Overlapping region at time *t*

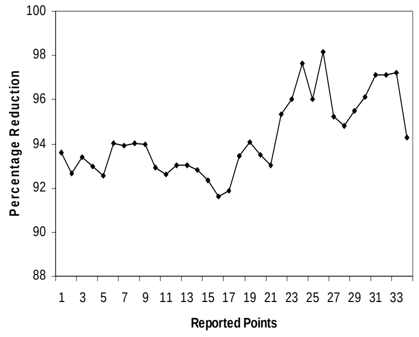
5 Experiments

Using a portable GPS device (Trimgle Navigation's ProXRS Receiver with GPS logger), which can record a pair <location-time, velocity> every second, we collected real GPS data. Every report was 3-dimensional (i.e., longitude, latitude, and time). We placed the GPS device in a car and drove from a location in the north of Denver, Colorado, to Loveland, Colorado along the interstate highway 25. Every second, we logged spatiotemporal data from the GPS device. Our collected data include both relatively straight movement on a highway and some winding movement in a city area, which is useful for a better comparison. For the comparison between the two models, we created trajectories based on the logged records. A time interval T_{int} defines the elapsed real time between two selected adjacent records. First we selected the logged records with a fixed 20-second time interval (i.e., T_{int}=20) and we also randomly selected the records with a sampling ratio of about 5%.

In all experiments, the maximum velocity, M_{vel}, was set to 180km/hour (50meters/sec). The maximum acceleration, M_{acc}, was set to 10km/hour per second (2.78 meters/sec per second). The maximum report (instrument and measurement) errors $M_{err}^{(1)}$ and $M_{err}^{(0)}$ were set to 0 and 2 meters, respectively. For the circular approximation discussed in Sec. 4.2, we selected 100 points along the boundary of the acceleration hyper-circle with a fixed interval.

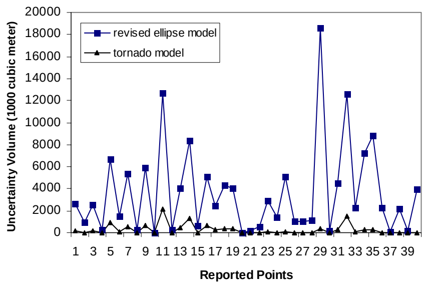
First, using the real GPS data selected with a fixed time interval (T_{int} = 20 seconds), we constructed 33 reported states (*RS*) and quantified the uncertainty region volume between each two adjacent states following the steps in Sec. 4.2. Fig. 4a shows the quantified uncertainty volumes of the two models. Fig. 4b shows the percentage reduction in uncertainty volume between the tornado uncertainty model (TUM) and the revised ellipse model (REM). On the average, TUM produced 94% of reduction compared to REM. The first 20 reported states were collected while driving on a straight highway with a high velocity and the last 13 reported states were collected while driving in a city area with a low velocity. The results show that the volume difference between TUM and REM becomes greater when the object moves slowly.

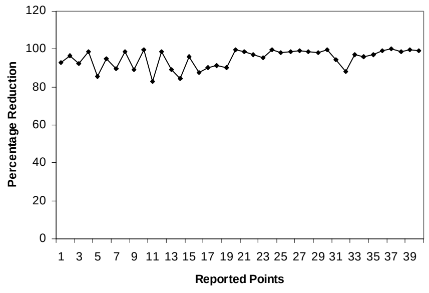




a. Quantified uncertainty volumes   b. Percentage reduction

Fig. 4. Comparison of two models, with 20 sec fixed interval





a. Quantified uncertainty volumes   b. Percentage reduction

Fig. 5. Comparison of two models, with random interval

Next, we repeated the same experiment with randomly selected records by generating arbitrary intervals between 5 and 35 seconds ($5 \leq T_{int} \leq 35$). Figs. 5a and 5b show the quantified uncertainty volumes of TUM and REM and the percentage difference in uncertainty volume between the two models, respectively. TUM produced a 95% reduction compared to REM on the average.

All results show that the uncertainty volumes produced by TUM are significantly smaller than their counterparts produced by REM. To investigate how efficient TUM is over REM, we performed the following experiments by varying a couple of factors. First, assuming identical initial velocity *ivel*, maximum velocity M_{vel}, and maximum acceleration M_{acc}, we quantified the uncertainty volumes with varying T_{int} between reported states. Table 2a shows that TUM becomes more efficient compared to REM as T_{int} gets smaller. Second, assuming identical *ivel*, M_{vel}, and T_{int}, we quantified the uncertainty volumes with varying M_{acc}. Table 2b shows that TUM becomes more efficient compared to REM as M_{acc} gets smaller.

As shown in Eqs. 3 and 6, $E^{(0)}$ is a function of *ivel*, M_{vel}, M_{acc}, and the elapsed time t from a reported state. $E^{(0)}$ gets more dispersed as the velocity of the object approaches M_{vel}. Eq. 3 (i.e., REM) produces more dispersion of $E^{(0)}$ than Eq. 6 (i.e., TUM) because REM assumes that M_{vel} is possible during the whole interval $|t\text{-}rs.T|$ (i.e., $t'=|t\text{-}rs.T|$). However, by considering possible accelerations and the reported velocity, TUM gradually increases the velocity from *ivel* to M_{vel} over time (i.e., the integral term of Eq. 6), which also gradually increases the dispersion of $E^{(0)}$. Thus, the slower the velocity reaches the maximum (i.e, as the maximum possible t' increases), the smaller the dispersion of $E^{(0)}$ becomes. The difference of uncertainty volumes between REM and TUM becomes accordingly wider. This is the reason why TUM becomes more efficient as M_{acc} and/or *ivel* decreases. Similarly, TUM becomes more efficient when the time interval is shorter. The velocity may not even reach to the maximum when the time interval is short.

Table 2. The average percentage reduction of uncertainty volume

T_{int} Range in Seconds	Average % Reduction
5-10	99.25
11-15	98.41
16-20	96.40
21-25	93.94
26-30	88.22
30-35	87.71

a. Varying time interval (M_{acc}=10)

Max. Acceleration (M_{acc})	Average % Reduction
10	94.30
20	81.79
30	69.26
40	58.90
50	51.66
60	46.30

b. Varying M_{acc} (T_{int}= 20 seconds)

6 Conclusion

In this paper, we proposed a novel and practical framework for managing multidimensional CCDOs (i.e., spatiotemporal trajectory representation and processing). The *Tornado* model can more efficiently support both conventional CCDOs that move in a 2- or 3-dimensional geographic space and emerging high-dimensional CCDOs, such as combined sensor streams and satellite data. Based on the framework, one might be able to devise an uncertainty model that employs a more precise approximation of the actual derivatives.

Processing a query with uncertainty means that each result data item is associated with the probability (or likelihood) that the item really satisfies the query predicates [3]. To support probabilistic query processing, one needs to calculate the probability density of each snapshot: An appropriate application-specific distribution (e.g., skewed-normal random distribution) can be used to estimate the probability density [1,2,9]. [13] provides some non-linear methods that can significantly reduce the errors associated with the peak point of the probability density. If the snapshots can be further minimized, the spatiotemporal regions requiring indexing can also be commensurately limited and the query results will be associated with more probable likelihoods. By taking into account how the environment may be variably constraining movement and thus variably affecting the set of possible states of the CCDO, one can further reduce the snapshots through a separate processing steps of contextualizing (modifying) the probability distributions [8].

In the two-phase (filtering-refinement) query processing, the smaller the uncertainty regions are, the lower the rate of false-drops (i.e., the objects that are selected in the filtering-phase but discarded in the refinement-phase) becomes. On the other hand, the computation cost of the uncertainty model affects the cost of the refinement step of the query processing. These two phases are not independent, since the false-drop rate of the first phase determines the number of objects to be tested in the refinement step. We have also considered the average cost of testing a candidate in the refinement step as follows: In our discrete implementation of the uncertainty models and experiments using a Linux machine equipped with an Intel Pentium III 800MHz and 256MB main memory space, the degree-1 Tornado model (REM) took about 0.7-0.8 microseconds of CPU time to interpolate each boundary point of the uncertainty region, and the degree-2 Tornado model (TUM) required about 5 microseconds. On the other hand, compared to REM, TUM reduced the uncertainty volumes by more than an order of magnitude on average. Understanding the implication of these in actual query performance requires the hardware platform, adopted cache-buffering method, the trajectory data, and the access method.

References

1. Azzalini A., Capitanio, A. Statistical applications of the multivariate skew-normal distribution. *Journal of the Royal Statistical Society*, Series B(61), 1999, 579-602.
2. Azzalini A., Valle, A. D. The multivariate skew-normal distribution. *Biometrika*, 83, 1996, 715-726.
3. Cheng, R., Kalashnikov, D., Prabhakar, S. Evaluating Probabilistic Queries over Imprecise Data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9), 2004, 1112-1127.
4. Hornsby, K., Egenhofer, M. J. Modeling Moving Objects over Multiple Granularities, *Annals of Mathematics and Artificial Intelligence*, 36(1-2), 2002, 177-194.
5. O'Rourke, J. Computational Geometry in C, 2nd ed., Cambridge University Press, 1998.
6. Pfoser, D., Jensen, C. S. Capturing the Uncertainty of Moving-Objects Representations. *In Proc Int. Conf. on Scientific and Statistical Database Management*, 1999, 123-132.
7. Pfoser, D., Jensen, C. S. Querying the Trajectories of On-Line Mobile Objects. *In Proc. ACM MobiDE International Workshop on Data Engineering for Wireless and Mobile Access*, 2001, 66-73.
8. Prager, S. D.: Environmental Contextualization of Uncertainty for Moving Objects. *In: Proc. GeoComputation.* Ann Arbor, Michigan, 2005.
9. R Development Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2004.
10. Sistla, P. A., Wolfson, O., Chamberlain, S., Dao, S. Querying the Uncertain Position of Moving Objects. *Temporal Databases: Research and Practice*, 1997, 310-337.
11. Trajcevski, G., Wolfson, O., Zhang, F., Chamberlain, S. The Geometry of Uncertainty in Moving Object Databases. *In Proc. Int'l Conf. on Extending Database Technology*, 2002, 233-250.
12. Wolfson, O., Sistla, P. A., Chamberlain, S., Yesha, Y. Updating and Querying Databases that Track Mobile Units. *Distributed and Parallel Databases*, 7(3), 1999, 257-387.
13. Yu, B., Kim, S. H., Bailey, T., Gamboa, R. Curve-Based Representation of Moving Object Trajectories. *IEEE International Database Engineering and Applications Symposium*, 2004, 419-425.