On-line Data Reduction and the Quality of History in Moving Objects Databases

Goce Trajcevski*
Dept. of EECS
Northwestern University
Evanston, Il 60208
goce@ece.northwestern.edu

Hu Cao
Dept. of CS
University of Illinois at Chicago
Chicago, Il 60607
hcao2@cs.uic.edu

Peter Scheuermann†
Dept. of EECS
Northwestern University
Evanston, Il 60208
peters@ece.northwestern.edu

Ouri Wolfson‡
Dept. of CS
University of Illinois at Chicago
Chicago, Il 60607
wolfson@cs.uic.edu

Dennis Vaccaro
Defense Systems Division
Northrop Grumman Corp.
Rolling Meadows, Il 60614
dennis.vaccaro@ngc.com

ABSTRACT

In this work we investigate the quality bounds for the data stored in Moving Objects Databases (MOD) in the settings in which mobile units can perform an on-board data reduction in real time. It has been demonstrated that line simplification techniques, when properly applied to the large volumes of data pertaining to the past trajectories of the moving objects, result in substantial storage savings while guaranteeing deterministic error bounds to the queries posed to the MOD. On the other hand, it has also been demonstrated that if moving objects establish an agreement with the MOD regarding the (im)precision tolerance, significant savings can be achieved in transmission when updating the location-in-time information. In this paper we take a first step towards analyzing the quality of the "history in making" in MOD, by correlating the (impact of the) agreement between the server and the moving objects for on-line updates in real time with the error-bounds of the data that becomes a representation of the past trajectories as time evolves.

Categories and Subject Descriptors

H.2.4 [**Database Management Systems**]: Moving Objects Databases

*Research supported by the Northrop Grumman Corp., contract: P.O.8200082518

†Research partially supported by NSF grant IIS-0325144

‡Research supported by NSF Grants 0326284, 0330342, ITR-0086144, 0513736, 0209190

MobiDE'06, June 25, 2006, Chicago, Illinois, USA.

General Terms

Data reduction algorithms

Keywords

Trajectories, Spatio-Temporal Data Reduction, Dead-Reckoning

1. INTRODUCTION AND MOTIVATION

An important aspect of any application that intends to provide some form of Location-Based Services (LBS)[25] is the ability to store and retrieve the data pertaining to many objects whose whereabouts-in-time information constantly changes. Efficient management and query processing of the mobile entities has spurred a large amount of research efforts and generated many results in the field known as Moving Objects Databases (MOD) [15].

Traditionally, there are three main models for representing the future motion plans of the moving objects and, as a consequence of the adopted model, handle the processing of the continuous spatio-temporal queries:

1. At one extreme is the model in which the objects periodically send their *(location,time)* updates to the MOD server (leftmost part in Figure 1). Due to the frequency of the updates, intelligent methodologies are needed that will avoid constant re-evaluation of the pending continuous queries, while still ensuring the correctness of their answers [21, 20]. In order to balance the efficiency of the query processing with keeping the MOD up-to-date, recent works have also addressed "lazy" updating mechanisms [34].

2. In the "middle-land" is the model in which the moving objects are assumed to periodically send *(location, time, velocity)* updates to the MOD server [26], as illustrated by the middle portion of Figure 1. Efficient algorithms for processing continuous queries in MOD under this model were presented in [17, 18], and

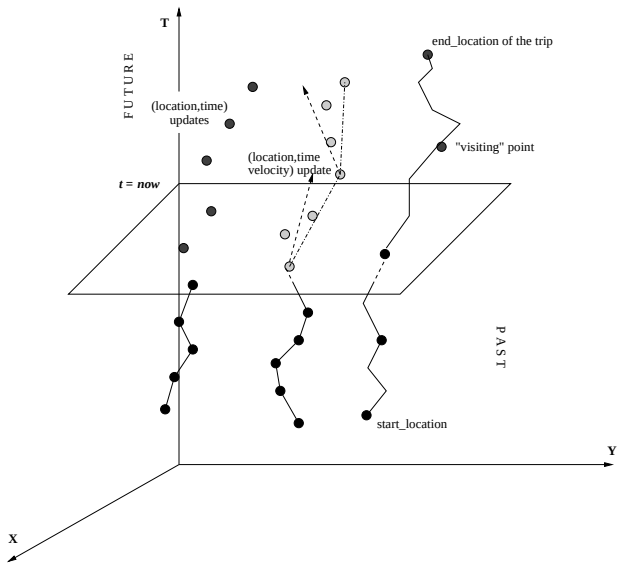

Figure 1: Modeling the motion plans of moving objects

[13, 14] addressed the distributed processing of such queries, by delegating some of the responsibilities to the moving objects themselves. One peculiar feature of this model is that in-between two consecutive updates, the objects are allowed to deviate from the expected route which is calculated using the *velocity* parameter of the most recent update, for as long as the deviation is within certain tolerance bounds [33]. This is illustrated by the shaded circles in the middle portion of Figure 1, which show the actual locations-in-time, as opposed to the expected ones that would be along the dotted arrowed line. We will elaborate this in more details in Section 2.

3. The other extreme model is the one in which the entire future motion plan of a given object is represented as a *trajectory* (rightmost portion in Figure 1). Under this model, each object is assumed to initially transmit to the MOD server the information about its *start_location*, *end_location*, and *start_time* of the trip, plus (possibly) a set of "to-be-visited" points. Using the information available from the electronic maps, plus the knowledge about the distribution of the traffic patterns in a given time-period, the server will apply an $A*$-like [11] variant of the time-aware Dijkstra's algorithm to generate the optimal travel plan [32]. A trajectory is essentially a sequence of 3D points (2D geography + time) of the form (x_1, y_1, t_1), (x_2, y_2, t_2), ..., (x_n, y_n, t_n), where $t_1 < t_2 < \ldots t_n$ and in-between two points the object is assumed to move along a straight line and with a constant speed. The peculiarity of this model is that it enables answering continuous queries pertaining to the further-future, however, the consequence is that a disturbance of the traffic patterns in a small geographic region may affect the correctness of the queries in widely-dispersed areas and at different time-intervals [8, 29].

An important observation, which is part of the motivation for this work, is that when it comes to the *past* of the objects' motion all three models, in a sense, converge, and represent it as a trajectory (bottom part of Figure 1). These past trajectories are important for various reasoning purposes. For example, a delivery company may be interested in a query such as: *"Retrieve all the trucks that had an average delivery-delay of more than 15 min. within the region of 3 miles around downtown Chicago, at least 3 days within the past two weeks"*. Similarly, a cellular service provider may be interested in a query like: *"Retrieve the motion patterns of the users that have changed the base station more than 5 times within any 24 interval, during a period of two weeks"*. An algebra with a complete set of operators for various argument signatures, and algorithms for processing continuous queries for the past trajectories were presented in [4, 19]. However, one significant problem that arises is the *size* of the storage needed for the trajectories' data. Namely, assuming that an update of the size 12B is generated by each mobile user (vehicles, cellular subscribers, etc.) every 20 seconds in a large geographic area, then the the storage requirements for the daily activities of 10 million objects would be approximately 1TB. To deal with this, spatio-temporal data reduction techniques were proposed in [5] that provided large storage savings and ensured bounded error on the uncertainty [30] of the trajectories, as well as the answers to the popular spatio-temporal queries.

At the heart of the motivation for this work is the observation that the results in [5] introduced techniques which are applicable to the *complete*-past trajectories. However, in many applications the whereabouts-in-time information of the moving objects arrives in a stream-like manner, as the time evolves. Thus, in order to apply techniques proposed in [5], one has to wait until the completion of the trips which, in a sense, still imposes the storage overhead. Hence, it is desirable to have some on-line data reduction mechanism which will yield some storage savings throughout the "daily life" of the MOD. However, then the question that arises is what is the quality of the data generated via on-line data reduction? Namely, one would not like a reduced-storage representation that generates too large errors for the typical

MOD queries. Since the results of [5] have established such bounds for data reduction applied to past trajectories, we asked if there can be some correlation between the quality of the data generated on-line with some data-reduction technique(s) applied in real time[1] AND the data representing the historic information in MOD *after* data reduction has been applied to the *complete*-past trajectories?

The main contribution of this work is that we establish such correlation and, in particular, we demonstrate that the dead-reckoning policy [33], when used by the objects whose motion is modelled as a sequence of *(location, time, velocity)* updates, actually generates a special instance of the reduced-storage representation when data reduction techniques are applied to the completed past trajectories. We provide analytical results which link the quality of the on-line created historic data with the data obtained after reduction based on line-simplification techniques has been applied to the complete-past trajectories. The practical implications of our result is as follows: if the MOD server is to wait until the end of the pre-established period (say, a day) for the trajectories to be completed, the above-mentioned 1TB of the storage will already be consumed, and the data reduction techniques will produce post-factum savings. However, if the data reduction is done in an on-line manner, at the end of the day, the storage savings have been achieved, without any other computational overhead. We have experimentally compared the storage savings generated by the two approaches, and our results demonstrate that the on-line data reduction yields savings that are acceptably close to the ones applied to the complete-past trajectories.

The rest of this paper is structured as follows. In Section 2 we recollect some necessary background. Section 3 presents our main results and Section 4 presents the experimental observations. Section 5 positions our work with the related literature and Section 6 concludes the paper and outlines our vision for the future work.

2. PRELIMINARIES

Now we briefly recollect the necessary background and we explain the main ideas behind the two approaches whose correlation is investigated in this work.

2.1 Dead-Reckoning

Dead-reckoning is a policy which essentially represents an agreement between a given moving object and the MOD server regarding the updates transmitted by that particular object. The main idea is that the communication between them can be reduced (consequently, network bandwidth can be spared) at the expense of the imprecision of the data in the MOD representing the object's motion. In order to avoid an unbounded imprecision of the object's whereabouts, the agreement specifies a threshold δ that is a parameter of the policy. The object sends its location and the expected velocity to the MOD server and, as far as the MOD server is concerned, the future trajectory of that object is an infinite ray originating at the update point and obtained by extrapolation from the velocity vector. The information that the MOD server has is the *expected trajectory* of the moving object. However, each moving object is aware about its actual location by periodically sampling it (e.g., using an on-board GPS). For as long as its actual location at a given time t_i does not deviate by more than δ from the location that the MOD estimates at t_i using the information previously transmitted, the object does not transmit any new updates. When the actual distance deviates by more then δ from its location on the expected trajectory, the object will send another *(location, time, velocity)* update.

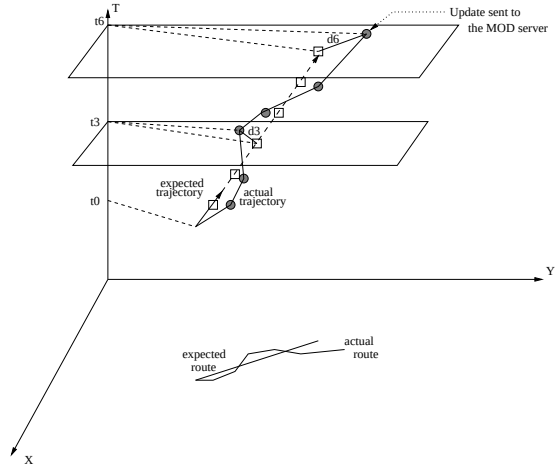

Figure 2: Distance-based Dead-Reckoning Policy

What we described is commonly known as a *distance-based* dead reckoning, and we illustrate it in Figure 2. At time t_0 the object sent its location and the predicted velocity (arrowed line) to the MOD server. The dashed line extending the vector indicate the expected trajectory of the moving object and the squares along it indicate the object's positions at 6 time instances, as estimated by the MOD, while the shaded circles indicate the actual positions of the object. Typically, the actual trajectory is obtained by connecting the GPS points with straight line-segments, assuming that in-between two updates, the object was moving with a constant speed. As illustrated, at t_6 the distance between the actual position and the MOD-estimated one exceeds the threshold agreed upon ($d_6 > \delta$) and the object sends a new update, at which point the MOD changes the expected trajectory, based on that update. Thus, at t_6, the MOD server actually performs two tasks: 1. corrects its own "knowledge" about the recent past and approximates the actual trajectory between t_0 and t_6 with a straight line-segment, which defines the *actual simplification* of the trajectory; and 2. generates another infinite ray corresponding to the future-expected trajectory, starting at the last update-point, and extrapolating the newly received velocity vector.

We reiterate that one of the benefits of the dead-reckoning policy is the reduction of the communication between the object and the MOD server, thus reducing the bandwidth consumption. Various trade-offs between the update costs and the (impacts on the) imprecision of the MOD data for several different dead reckoning policies are investigated in [33].

2.2 Line Simplification

The line simplification problem problem has been extensively studied by the Computational Geometry community, and its essential characteristics can be described as follows.

[1]Essentially, with the evolution of time, every new clock-tick shifts the present "now" into past.

Given a polyline PL_1 with vertices $\{v_1, v_2, \ldots, v_n\}$ in a respective k-dimensional Euclidean space, and a tolerance ε, construct another polyline PL'_1 with vertices $\{v'_1, v'_2, \ldots, v'_m\}$ in the same space, such that $m \leq n$ and for every point $P \in PL_1$ its distance from PL'_1 is smaller than a given threshold: $dist(P, PL'_1) \leq \varepsilon$. The value of $dist(P, PL'_1)$ actually depends on the distance-function chosen, and for many geometrically-intuitive functions, it denotes the distance $\overline{P, P'}$, between P and a particular point $P' \in PL'_1$. For example, the shortest distance between a point M and a line-segment QB in the Euclidian 2D space is the distance MM' where M' is either the perpendicular projection of M on the linde defined by Q and R, or (in case the projection is outside the segment QR) the closest end-point (Q or R) to M.

In case $\{v'_1, v'_2, \ldots, v'_m\} \subseteq \{v_1, v_2, \ldots, v_n\}$, PL'_1 is a *strong* simplification of PL_1, which is the case that we consider in this paper; otherwise, when $\{v'_1, v'_2, \ldots, v'_m\}$ is an arbitrary subset of the points in PL_1 (not necessarily vertices), PL'_1 is called a *weak* simplification. There are two distinct facets of the minimal line simplification problem: 1. Given PL and ε, minimize the number of points m in PL' (known as min-# problem) [6], and 2. Given PL and the "budget" m of the points in PL', minimize the error ε (known as min-ε problem). Optimal algorithms for both problems in 2D settings are given in [6], and for 3D and higher dimensions in [3], and the common *Euclidian distance* in the respective spaces was used as a distance function.

It seems natural to adapt the results in [3, 6] for the 3D space to trajectories in MOD. However, as demonstrated in [5], a caution needs to be exercised when using the *distance function*. Namely, if the *time* is simply treated as a Z-axis and the Euclidean distance in 3D is used in the algorithms for line simplification, the reduced version of the trajectory may yield an *unbounded* error for the answers to the popular spatio-temporal queries, such as, range, (k) nearest neighbor, (semi)join. To alleviate this and guarantee a deterministic error-bound, in [5] the authors introduced the distance function called three-dimensional time_uniform distance, denoted by E_u, which can be specified as follows. Let $p_i = (x_i, y_i, t_i)$, $p_j = (x_j, y_j, t_j)$, such that $t_i < t_j$, and $p_s = (x_s, y_s, t_s)$. The E_u distance between p_s and the 3D line segment $\overline{p_i p_j}$ is $E_u = \sqrt{(x_s - x_c)^2 + (y_s - y_c)^2}$ where $p_c = (x_c, y_c, t_c)$ is the unique point on $\overline{p_i p_j}$ which has the same *time*-value as p_s (i.e., $t_c = t_s$). In other words, the time_uniform distance is the 2D Euclidean distance between p_s and the 3D point on $\overline{p_i p_j}$ at time t_s. Observe that, in case $t_s \notin [t_i, t_j]$, E_u is undefined.

Throughout the rest of this work, we will assume that E_u distance is used.

3. THE QUALITY ASPECTS OF ON-LINE DATA REDUCTION

Now we proceed with presenting the main results of this work.

Assume that a given moving object o_n has established the agreement with the MOD server that a distance-based dead-reckoning policy *(ddr)* with the threshold δ will be used for the updates. Let t_0 denote the beginning time of o_n's trip and let t_c ($t_c > t_0$) denote the current time. Assume that the polyline corresponding to the actual trajectory of o_i between t_0 and t_c is $Tr = \{(x_0, y_0, t_0), (x_1, y_1, t_1), \ldots, (x_i, y_i, t_i), \ldots, (x_c, y_c, t_c)\}$. Further, to simplify the argument assume that along its trip during the time-interval $[t_0, t_c]$, o_n has already sent k updates to the MOD server at times t_{j1}, t_{j2}, ..., $t_{jk} = t_c$. Thus, as far as the MOD is concerned, the past motion plan of the object o_n is a polyline corresponding to the trajectory $Tr' = \{(x_0, y_0, t_0), (x_{j1}, y_{j1}, t_{j1}), \ldots, (x_{jk}, y_{jk}, t_{jk})\}$, which is the *actual simplification* obtained by consecutivelly updating the MOD in accordance with the *ddr* policy, and $(x_{jk}, y_{jk}, t_{jk}) = (x_c, y_c, t_c)$. Now, we have the following:

THEOREM 1. If Tr' *is an actual simplification obtained using the* ddr *policy,* then *it is equivalent to a strong simplification of* Tr *obtained using the distance function* E_u *and with a tolerance threshold* $\varepsilon \leq 2 \cdot \delta$.

Proof: Firstly, observe that since $\{(x_0, y_0, t_0), (x_{j1}, y_{j1}, t_{j1}), \ldots, (x_{jk}, y_{jk}, t_{jk})\} \subset \{(x_0, y_0, t_0), (x_1, y_1, t_1), \ldots, (x_i, y_i, t_i), \ldots, (x_c, y_c, t_c)\}$, then Tr' is indeed a strong simplification of Tr (c.f. Section 2).

The rest of the proof is by induction on the value of k – the number of updates sent by o_n to the MOD server in accordance with the *ddr* (equivalently, the time t_c at which the last update is sent) and an illustration is provided in Figure 3.

Base Case: ($t_c = t_0$) Since the trajectory has only one point, the claim trivially follows.

Inductive Hypothesis: Assume that the statement is true for a value of t_c up to which k updates are sent by the moving object o_n.

Inductive Step: Suppose that the $(k+1)$-st update is sent at the current time $t_c = now$ and the actual trajectory-segment of o_n, as obtained with by the on-board GPS between t_{jk} and t_c is $Tr_k = \{(x_{k1}, y_{k1}, t_{k1}), (x_{k2}, y_{k2}, t_{k2}), \ldots, (x_{kl}, y_{kl}, t_{kl}), \ldots, (x_{km}, y_{km}, t_{km})\}$, where $t_{k1} = t_{jk}$ and $t_{km} = t_c$, illustrated by the points $A = (x_{k1}, y_{k1}, t_{k1})$ and $B = (x_c, y_c, t_c)$ in Figure 3 . However, in accordance with the *ddr* policy, at the time-instance t_{k1}, o_n has sent the information $[(x_{k1}, y_{k1}, t_{k1}), \vec{V_{k1}}]$ to the MOD server. Consequently, just before the update, the MOD believes that at t_c, the location of o_n would be $C = (X_{cMOD}, Y_{cMOD})$, such that $X_{cMOD} = x_{k1} + V^x_{k1} \cdot (t_c - t_{k1})$ and $Y_{cMOD} = y_{k1} + V^y_{k1} \cdot (t_c - t_{k1})$, where V^x_{k1} (resp. V^y_{k1}) is the x-component (resp. y-component) of $\vec{V_{k1}}$.

Due to the properties of the *ddr* policy, we have that $\overline{BC} = \delta$ and the points B and C are in a same horizontal plane $t = t_c$. Once it has received the $(k+1)$-st update at t_c, the MOD server will approximate the trajectory-segment of o_n between t_{k1} and t_c with the straight line-segment $\overline{(x_{k1}, y_{k1}, t_{k1})(x_c, y_c, t_c)} = \overline{AB}$ (c.f. Figure 3). Let $D = (x_{kl}, y_{kl}, t_{kl})$ denote the point on the actual trajectory segment of o_n which is the furthest one from $\overline{AB}$, using the E_u is used as a distance function, and let F denote the corresponding point on $\overline{AB}$ at t_{kl}. Also, let $E = (X_{EMOD}, Y_{EMOD}, t_{kl})$ denote the point in which the MOD was expecting o_n to be at t_{kl}, according to the update sent at t_{k1}. Due to the triangular inequality, $\overline{DF} \leq \overline{DE} + \overline{EF}$, with the equality when the co-planar points D, E and F are also collinear. However, in that particular case, we have that $\triangle ABC$ and $\triangle AFE$ are similar triangles and $\frac{\overline{BC}}{\overline{EF}} = \frac{t_c - t_{k1}}{t_{kl} - t_{k1}}$. Hence, $\overline{EF} = \overline{BC} \cdot \frac{t_{kl} - t_{k1}}{t_c - t_{k1}}$ and, since the *ddr* policy was used, we have that $\overline{DE} < \delta$ and $t_{kl} < t_c$, otherwise o_n

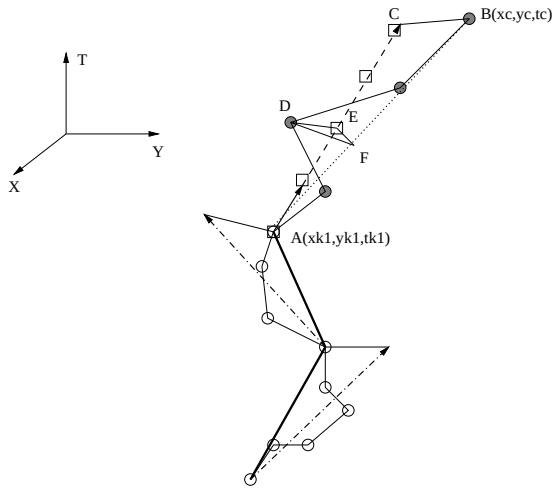

Figure 3: *ddr*-based Data Reduction and Line Simplification

would have sent an update to the MOD server at t_{kl} instead of t_c. Thus, we have the following: $max(DF) \leq \overline{DE}+\overline{EF} < \delta + \frac{t_c - t_{k1}}{t_{kl} - t_{k1}} \cdot \delta < 2 \cdot \delta$. Consequently, at the time t_c when the MOD server has approximated the trajectory segment Tr_k with the line segment $\overline{AB}$, every point on the polyline Tr_k is no further than $\varepsilon = 2 \cdot \delta$ from $\overline{AB}$, which implies that $\overline{AB}$ is a strong simplification of Tr_k. Using the inductive hypothesis, we get that $Tr' \cup \overline{AB}$ is actually a strong simplification of $Tr \cup Tr_k$, when E_u is used as a distance function with tolerance ε. □.

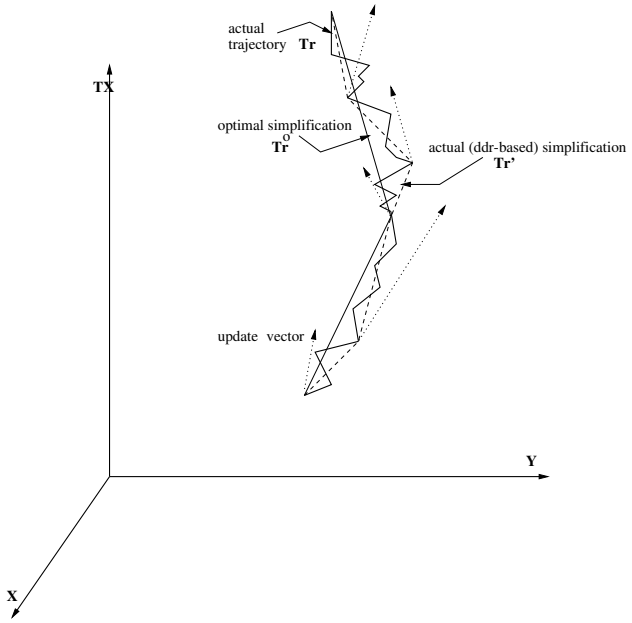

Figure 4: Distance Between Actual and Optimal Simplifications

Although the on-line data reduction using the *ddr* policy generates a strong simplification with respect to the past history of the moving objects' motion, there are two other important aspects of the quality of data that need to be addressed:

1. What are the actual savings in the storage obtained via the on-line simplification of the trajectory? Namely, if the optimal line simplification algorithm [3] is applied offline to the completed past trajectory of a given object, then we know that the number of vertices (equivalently, the size of the reduced trajectory) m is indeed the minimal possible for a given ε (equivalently, δ). However, the *ddr*-based data reduction is not guaranteed to generate a strong simplification with minimal number of vertices for a given ε. Our experiments provide a quantitative comparison of the storage savings obtained via *ddr* policy vs. the optimal line simplification algorithm [3, 6], as well as the heuristic approach known as the Douglas-Peuker algorithm [10, 16], as a function of ε.
2. For a given ε, how "far off" is the actual simplification, obtained on-line via *ddr*, from the optimal one that could be generated at the end of the trip? More specifically, let Tr denote the trajectory corresponding to the entire polyline of the motion of a given object, say, o_n. If the MOD server has Tr available in its entirety, then it can apply the optimal line simplification algorithm and obtain $T^o r$, which is a strong simplification of Tr with minimum number of points for a given tolerance ε. Also, let Tr' denote the actual simplification of the o_n's motion obtained in real time via the *ddr*, with a tolerance threshold $\delta = \frac{\varepsilon}{2}$. An illustration is provided in Figure 4, where the thick solid polyline indicates the complete trajectory Tr, and the dotted-arrow lines indicate the updates sent by the particular moving object. The solid polyline, which has only two segments, represents the optimal simplification of the trajectory $T^o r$, and the dashed polyline represents Tr' – the actual simplification of Tr. Now, the question becomes, how far is Tr' from $T^o r$? Typically (generalizing the discussion in Section 2.2), the distance between two curves is expressed in terms of a *Hausdorff distance*, which can be explained as follows. Let M denote the distance function used to measure the distances in a given space, and let $d_M(P_1, C_2)$ denote the distance between a point, say $P1 \in C_1$, and the curve C_2. Then, the distance between the curves C_1 and C_2 is defined as $D_M(C_1, C_2) = max(d_M(P1, C_2))(\forall P1 \in C_1)$. In our settings, since the distance function used is E_u (c.f. Section

2.2), we are interested in the value[2] of $D_{E_u}(Tr', T^o r)$. As a straightforward consequence of Theorem 1, we have that the distance between the actual simplification and the (offline) optimal one is $D_{E_u}(Tr', T^o r) \leq 2 \cdot \varepsilon$.

We conclude this section with one more observation regarding some other consequences of the spatio-temporal data reduction. Namely, once a line-simplification (or, for that matter, any other data reduction) technique is applied to the past trajectories, their representation in MOD will inevitably need to incorporate some *uncertainty* of the object's whereabouts at a given time. This, in turn, implies that the MOD users will need to be aware about it and, somehow, incorporate it in the (semantics of) the answers to their queries of interest. Linguistics constructs and algorithms for processing MOD queries with the uncertainty model that corresponds to the one obtained when trajectories' data is reduced using line-simplification based techniques, were presented in [30]. Since the actual simplification obtained using the *ddr* policy is at a bounded distance from the optimal simplification applied to the full past trajectories, the results from [5] regarding the the errors of the answers to the spatio-temporal queries can be carried over verbatim. The main advantage of the uncertainty obtained in a manner discussed in this paper, as opposed to the uncertainty introduced when other data reduction techniques are applied to spatio-temporal data (e.g., wavelets [12]) is that the error of the answers to the popular spatio-temporal queries is *bounded* [5]. The consideration of the other uncertainty models for various MOD settings [7, 22] from the perspective of on-line spatio-temporal data reduction, is beyond the scope of this paper and is a subject of our future work.

4. EXPERIMENTAL RESULTS

We used a real trajectory set of 60 GPS trajectories in our experiment. The trajectories were obtained from the GPS receiver carried by one of our lab members during the driving related to his various daily activities (commute, shopping, etc.). Each trajectory represents a trip that occurred in the Cook county and DuPage county of Illinois. Along each trip, the location was sampled every second. The average number of vertices per trajectory is 1465 and the average length of a trajectory is 15.14 miles[3]. In order to generate the *ddr*-based data for the experimental comparisons, we used the following heuristics: initially (and after each subsequent *ddr*-based update), we estimated the velocity vector $\overline{v}$ as an average between the first two velocity vectors, calculated from the set of the first three GPS-sampled points, and assuming constant speed between the consecutive samples.

The data reduction is expressed by the reduction ratio (rr), which is number of vertices of the simplified trajectory / number of vertices of the original trajectory (i.e., the trajectory before simplification). In other words, the storage savings of the simplifications is (1 - rr). For each data reduction experiment we varied the simplification tolerance ε from 0.05 mile to 0.5 mile. All the experiments reported in this paper were performed on a AMD64 3500 machine with 1G B DDR memory, running on Suse Linux.

[2]One may observe (c.f. [2]) that the Hausdorff distance with E_u as a distance measure between two trajectories is equivalent to the Freché distance between the corresponding routes when their respective equations are represented as parameters of time.

[3]see http://cs.uic.edu/ boxu/mp2p/gps_data.html for more details.

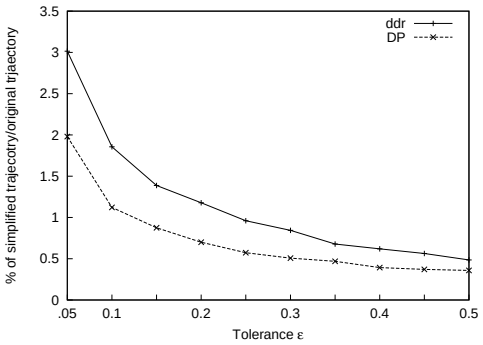

Figure 5: Comparing *ddr* savings vs. Douglas-Peuker savings

Figure 5 presents the comparison between the savings obtained using the *ddr* policy and the ones obtained when the Douglas-Peuker algorithm [10] was applied to the entire set of points representing the completed motion of the objects. The X-axis represents the value of ε used in the data-reduction process and one can observe that even for very small values of ε, the storage savings using the on-line data reduction are comparable to the ones obtained by the Douglas-Peuker algorithm applied to the entire past trajectory.

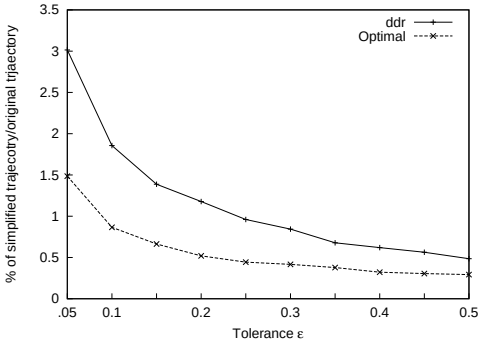

Figure 6: Comparing *ddr* savings vs. Optimal Savings

Similarly, Figure 6 illustrates the comparison of savings between the on-line reduced trajectory using *ddr* and the post-trip reduced trajectory using the variant of the optimal line simplification algorithm [6, 3], adapted to use the E_u distance function. One may observe that the relative storage savings obtained with the *ddr* based data reduction, when compared to the savings obtained using the optimal algorithm, are slightly worse than the *ddr*-based savings compared to the ones obtained by the Douglas-Peuker algorithm. However, the results still demonstrate that substantial savings can be achieved by on-line data reduction, that are reasonably close to the ones generated by the optimal line simplification algorithm, while still guaranteeing a bounded error on the trajectory representation and, consequently, the continuous queries posed to the MOD.

An observation which is in order here is that besides the storage savings demonstrated in Figure 5 and Figure 6 above, one needs to keep in mind that the *ddr*-based data reduction was obtained on-line and, essentially was available at the MOD at any point during its (evolving) history. Meanwhile, in order to generate reduced trajectory data using line simplification based algorithms, one would have to wait until the completion of the trip. Further quantitative analysis regarding various aspects of the benefits of these savings in the processing time is a subject of our future work.

5. RELATED WORK

Historically, the first work to address the line simplification problem was [10], for the purpose of reducing the number of points when displaying geographic features on the maps [31]. Although an improved version (in running time) was subsequently presented [16], in essence, just like the originally proposed algorithm – it yields a heuristic, not an algorithm that will ensure an optimal reduction of a number of points for a given tolerance (and vice-versa). The Computational Geometry community has provided algorithms that guarantee minimal number of points for a given ε, as well as the minimal ε for a given number of points, both for 2D [6] and, more recently, for 3D [3] polylines. However, these works use Euclidean distance in the respective 2D/3D spaces to measure the distance between the points on the original and simplified polylines. As demonstrated in [5], this is not appropriate for the MOD settings because the simplified trajectories' data may yield unbounded errors[4] to the popular MOD queries (range, NN, (semi) join). Based on the results presented in [5], in this work we used the E_u distance function and we demonstrated that a certain level of the simplification-quality of the MOD data can be obtained "on the fly".

Modelling the motion plan of a moving object as a sequence of *(location, time, velocity)* updates was used in [26], which introduced the concept of *dynamic attributes* that need to be updated whenever the expected value of the *velocity* of the object changes. Subsequently, [33] have investigated the trade-offs between communication vs. (im)precision for several variants of a dead-reckoning policy. Efficient algorithms for processing continuous queries in MOD using this type of motion model were investigated in [17, 18]. A paradigm for delegating the processing of the continuous queries to the mobile objects for the purpose of minimizing the communication overhead was introduced in [13, 14], and the works used a variation of the *ddr* dead-reckoning policy for generating velocity updates. However, these works were concerned with optimization problems regarding the query processing for future queries as the *now*-time evolves, and did not address the issue of the quality of the historic data of the MOD. As a particular example, [13, 14] focus on efficient algorithms that pertain to the updates only for the relevant subset of the objects that could impact the result of a particular query[5]. On the other hand, we focused on the correlation between the reduction of the update frequency with a deterministic imprecision bound and the line simplification based properties of the past trajectories in MOD.

[4]A simple intuitive observation of inadequacy is that treating the *time* simply as a Z-axis may justify "traveling back-in-time" when measuring the distance, which is unacceptable in MOD.

[5]Same type of problems were addressed in [21, 20], however, the motion model was assumed to be a stream of *(location,time)* updates

6. CONCLUSIONS AND FUTURE WORK

In this work, we analyzed the impact that an on-line spatio-temporal data reduction using a distance-based dead reckoning policy has on the quality of the data representing the past trajectories of the moving objects. We demonstrated that, when a *ddr* with a threshold δ is used, the evolving-past constructed in this manner actually generates a trajectory that corresponds to a strong line simplification with distance function E_u and threshold $\varepsilon = 2\delta$, applied to the set of points representing the entire trip of a given object. Furthermore, our experiments demonstrated that the storage savings thus obtained are comparable with the ones that would be obtained when the optimal line simplification algorithms [6, 3] are used, and even more comparable with the savings obtained by the Douglas-Peuker heuristics [10, 16], however, both of these methods are applicable on complete-past trajectories.

Our vision for the future work includes several directions of extending the current results: 1. Since the *ddr* assumes that the motion model is the one in which the updates are of the form *(location, time, velocity)*, we would like to investigate an adaptive on-line data reduction for the settings in which the updates are of the form *(location,time)*. This would yield storage savings for the past trajectories with deterministic quality-of-data bounds even for the objects that are not directly involved in the answers of any pending continuous queries [21, 20]. Furthermore, we believe that it would be interesting to make the on-line data reduction process query/context aware, in the sense that different values of the threshold ε can be applied, depending on which queries is a parctular object part of the answer-set; 2. We believe that the on-line data reduction will also have a significant impact on the updates overhead and the maintenance of the underlying indexing mechanism used in MOD [23, 24, 27], especially with the advantage of the deterministic bound for the error/uncertainty [28]; 3. Another interesting problem is to determine whether, and to what extent, the knowledge of the existing infrastructure can be used to improve the benefits of the on-line data reduction while further decreasing the communication between the objects and the MOD server [9]; 4. Finally, we would like to extend our ideas to the settings in which the motion of the objects is not represented as a (piece-wise) linear function of time [1].

7. REFERENCES

[1] C.C. Aggarwal and D. Agarwal. On nearest neighbor indexing of nonlinear trajectories. In *ACM International Conference on Principles of Database Systems (PODS)*, 2003.

[2] H. Alt, K. Knauer, and C. Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1), 2003.

[3] G. Barequet, D. Z. Chen, O. Deascu, M. T. Goodrich, and J. Snoeyink. Efficiently approximating polygonal path in three and higher dimensions. *Algorithmica*, 33(2), 2002.

[4] M. Breunig, C. Türker, M. Böhlen, S. Dieker, R.H. Güting, C. Jensen, L. Relly, P. Rigaux, H.-J. Schek, and M. Scholl. Architectures and implementations of spatio-temporal database management systems. In *Spatio-Temporal Databases – the Chorochronos Approach*. 2003.

[5] H. Cao, O. Wolfson, and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *Journal of Very Large Databases (VLDBJ)*, 2006. accepted, to appear.

[6] W. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments or minimal error. *International Journal of Computational Geometry Applicaitons*, 6, 1996.

[7] R. Cheng and S. Prabhakar. Managing uncertainty in sensor database. *SIGMOD Record*, 32(4), 2003.

[8] H. Ding, G. Trajcevski, and P. Scheuermann. OMCAT: Optimal Maintenance of Continuous queries Answers for Trajectories. In *ACM SIGMOD*, 2006. (demonstration paper).

[9] Z. Ding and R.H.Güting. Managing moving objects on dynamic transportation networks. In *International Conference on Scientific and Statistical Database Management (SSDBM)*, 2004.

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

[11] S. E. Dreyfus. An appraisal of some shortest – path algorithms. *Operations Research*, 17(3), 1969.

[12] M. Garofilakis and P. B. Gibbons. Wavelet Synopses with Error Guarantees. *ACM SIGMOD International Conference on Management of Data*, 2002.

[13] B. Gedik and L. Liu. Mobieyes: Distributed processing of continuous queries on moving objects in a mobile system. In *International Conference on Extending the Database Technology (EDBT)*, 2004.

[14] B. Gedik and L. Liu. Mobieyes: A distributed location monitoring service using moving location queries. *IEEE Transactions on Mobile Computing*, 2006. (accepted, to appear).

[15] R.H. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[16] J. Hershberger and J. Snoeyink. Speeding up the douglas-peuker line-simplification algorithm. In *International Symposium on Spatial Data Handling*, 1992.

[17] G.S. Iwerks, H. Samet, and K. Smith. Continuous k-nearest neighbor queries for continuously moving points with updates. In *International Conference on Very Large Databases (VLDB)*, 2003.

[18] G.S. Iwerks, H. Samet, and K. Smith. Maintenance of spatial semijoin queries on moving points. In *International Conference on Very Large Databases (VLDB)*, 2004.

[19] J.A.C. Lema, L. Forlizzi, R.H. Güting, E. Nardeli, and M. Schneider. Algorithms for moving objects databases. *Computing Journal*, 46(6), 2003.

[20] M.F. Mokbel, X. Xing, M. Hammad, and W.G. Aref. Continuous query processing of spatio-temoral data streams in place. *The GeoInformatica Journal*, 2006. accepted, to appear.

[21] M.F. Mokbel, X. Xiong, and W.G. Aref. SINA: Scalable incremental processing of continuous queries in spatio-temporal databases. In *ACM SIGMOD International Conference on Management of Data*, 2004.

[22] D. Pfoser, N. Tryfona, and C. Jensen. Indeterminacy and spatiotemporal data: Basic definitions and case study. *GeoInformatica*, 9(3), 2005.

[23] S. Prabhakar, Y. Xia, D. Khalashnikov, W. Aref, and S. Hambrusch. Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. *IEEE Transactions on Computers*, 51(10), 2002.

[24] S. Saltenis and C. Jensen. Indexing of moving objects for location-based services. In *International Conference on Data Engineering (ICDE)*, 2002.

[25] J. Schiller and A. Voisard. *Location-based Services*. Morgan Kaufmann Publishers, 2004.

[26] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *13th International Conference on Data Engineering (ICDE)*, 1997.

[27] Y. Tao and D. Papadias. Spatial queries in dynamic environments. *ACM Transactions on Database Systems (TODS)*, 28(2), 2003.

[28] Y.Tao, R. Cheng, X. Xiao, W.K. Ngai, B. Kao, and S. Prabhakar. Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions. *International Conference on Very Large Databases (VLDB)*, 2005.

[29] G. Trajcevski and P. Scheuermann. Reactive maintenance of continuour queries. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8, 2004.

[30] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM Transactions on Database Systems (TODS)*, 29(3), 2004.

[31] R. Weibel. Generalization of spatial data: Principles and selected algorithms. In *Algorithmic Foundations of Geographic Information Systems*. LNCS Springer Verlag, 1998.

[32] O. Wolfson, H. Cao, H. Lin, G. Trajcevski, F. Zhang, and N. Rishe. Management of dynamic location information in domino. In *International Conference on Extending Database Technology (EDBT)*, 2002.

[33] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 7, 1999.

[34] X. Xing and W. Aref. R-trees with update memos. In *International Conference on Data Engineering (ICDE)*, 2006.