Spatio-temporal Data Reduction with Deterministic Error Bounds

Hu Cao
Department of Computer Science,
University of Illinois at Chicago
Chicago, Illinois, USA
hcao2@cs.uic.edu

Ouri Wolfson
Department of Computer Science,
University of Illinois at Chicago
Chicago, Illinois, USA
wolfson@cs.uic.edu

Goce Trajcevski
Department of Computer Science,
University of Illinois at Chicago
Chicago, Illinois, USA
gtrajcev@cs.uic.edu

ABSTRACT

A common way of storing spatio-temporal information about mobile devices is in the form of a 3D (2D geography + time) trajectory. We argue that when cellular phones and Personal Digital Assistants become location-aware, the size of the spatio-temporal information generated may prohibit efficient processing. We propose to adopt a technique studied in computer graphics, namely line-simplification, as an approximation technique to solve this problem. Line simplification uses a distance function in producing the trajectory approximation. We postulate the desiderata for such a distance: it should be sound, namely the error of the answers to spatio-temporal queries must be bounded. We analyze several distances, and prove that some are sound in this sense for some types of queries, while others are not. Interestingly, not a single distance analyzed proves to be sound for all the common spatio-temporal queries, and therefore multi-distance line-simplification is introduced and analyzed. Then we propose an aging mechanism which gradually shrinks the size of the trajectories as time progresses. Finally, we analyze experimentally the effectiveness of line-simplification in reducing the size of a trajectories database.

Categories and Subject Descriptors: H.2.m [Information Systems Applications]: Miscellaneous

General Terms: Theory, Experimentation

Keywords: Moving Objects Database, Line Simplification

1. INTRODUCTION

Location management, i.e. the management of transient location information, is an enabling technology for location based service applications. It is also a fundamental component of other technologies such as fly-through visualization (the visualized terrain changes continuously with the location of the user), context awareness (location of the user determines the content, format, or timing of information delivered), augmented reality (location of both the viewer and the viewed object determines the type of information delivered to viewer), and cellular communication.

We view location management as the problem of managing a set of spatio-temporal points of the form (x, y, t). Such a point indicates that a moving object m was or will-be at geographic location with coordinates (x, y) at time t. These spatio-temporal points may be generated, for example, by a GPS receiver on board m. We will call such point a GPS point, although it may be generated by other means (e.g. PCS network triangulation, a proximity sensor, etc...). Now, consider that a GPS receiver can generate a new (x, y, t) point every second, and that the number of moving objects may be hundreds of millions to billions. Remember also that one is interested in past locations, and planned future locations, and that historical spatio-temporal points may need to be mined for road-network capacity planning, accident replay, municipal transportation planning (e.g. answering a query such as: how many times was bus number 5 late by more than five minutes at a stop in the last year), etc. Thus one can immediately recognize the storage-space problem that location based services applications will face, as well as the computation burden for processing such large amount of information. Additionally, in online tracking where the spatio-temporal points are transmitted from a moving object to a server, this storage problem translates into a bandwidth and power problem.

A key observation that lies at the foundation of this paper is that a GPS point (x, y, t) can be eliminated, and its space saved, if (x, y, t) can be approximated with a reasonable accuracy by interpolating the adjacent (i.e. before and after) GPS points[1]. We formalize this intuition by employing a mechanism based on *line simplification*, that has been studied in computational geometry, cartography and computer graphics. Basically, line simplification approximates a polygonal line by another that is "sufficiently close" (the term will be precisely defined), and has less straight-line segments (or points) and thus takes less storage-space.

The main advantage of line simplification over the most popular lossy compression, namely wavelets[1, 10], is that

[1]Observe that even in the absence of the storage-space concern, location-based applications require interpolation of the location between GPS points. Interpolation is necessary in order, for example, to answer the query: "where was moving object m at 2pm", assuming that a GPS sample for 2pm does not exist.

DIALM–POMC'03, September 19, 2003, San Diego, California, USA.

it provides a deterministic bound on errors. No other lossy compression methods that we are aware of can do that.

Our experimental results indicate that the storage-savings using line simplification is dramatic. Specifically, we used real datasets of moving objects trajectories[2]. The trajectories dataset was obtained from the trace of GPS-points recorded by the shuttle buses on the UCLA campus (we elaborate on our experimental settings and results in Section 5.)[3]. Our experiments indicate that storage-size decreases in an exponential-like manner as the allowed imprecision increases, and an imprecision tolerance of 0.1 miles produces 99% storage savings.

In addition to the storage and processing savings, the attractiveness of line simplification (compared to other data compression techniques such as wavelets) stems from the fact that the approximation carries a given error bound. Namely, the distance between the original trajectory and the approximation is bounded by a parameter of the simplification called the error-tolerance. However, we discovered that, surprisingly, although the approximation error is bounded, the error of the answers to queries[4] may not be bounded. Whether or not it is bounded, depends on the combination of the distance function(or distance for short) used in the approximation, and the spatio-temporal query type. In other words, for some combinations(query-type, distance) the answer-error is bounded (in this case we call the combination *sound*), for others it is not. For example, the Euclidean distance function is not sound for the query "where was moving object m at time t". We provide a comprehensive analysis of the soundness of combinations(query-type, distance).

Then we considered an aging mechanism by which a trajectory is represented by increasingly coarser approximations as time progresses. For example, initially, when the trajectory is stored, it is approximated by a polyline with distance at most 0.1 mile from the original, after 2 months it is approximated by a polyline (which is smaller in size than the first) at distance at most 0.2 miles from the original trajectory, etc. We show that some simplification algorithms are "aging-friendly" (e.g. the Douglas-Peuker heuristic), and some are not (e.g. the optimal simplification algorithm). By aging friendliness we mean that even though the original trajectory is not saved, at every stage we obtain a trajectory that could have been obtained using the larger tolerance from the original trajectory.

We also analyzed experimentally various simplification algorithms. One of the conclusions is that the Douglas-Peuker (DP) algorithm achieves near-optimal savings at a far superior performance.

In summary, the main contributions of this paper are as follows:

- We introduce the concept of soundness of a data compression mechanism.
- We analyze the soundness of several (distance, spatio-temporal query) combinations.
- We quantify experimentally the power of line simplification using different distances and simplification algorithms.
- We analyze the behavior of approximation (simplification) algorithms with respect to data aging, and show that some are well behaved whereas others are not.

The rest of this paper is organized as follows. Section 2 discusses the concept of a trajectory and introduces the problem of trajectory reduction/simplification. Section 3 introduces the concept of soundness and analyzes it with respect to (distance , query type) combinations. Section 4 analyzes simplification algorithms with respect to aging. Section 5 presents our experimental results of trajectory simplification using different distances, tolerances and algorithms. In Section 6 we position our paper with respect to the relevant works, and in Section 7 we provide concluding remarks and directions for future work. In the appendix we provide proof sketches for the theorems.

2. TRAJECTORY REDUCTION

In this section we present basic definitions and we introduce the concept of trajectory reduction by a line simplification. Specifically, in subsection 2.1 we illustrate the nature (and magnitude) of the problems which may arise when storing and processing trajectories, and we introduce the line-simplification approach to address these problems. In subsection 2.2 we discuss several possible distances for this approach.

2.1 The Problem and Line Simplification Solution

Representing the *(location,time)* information of the moving object as a trajectory is a typical approach (c.f. [20, 22, 24]):

Definition 1. *A <u>trajectory</u> is a function $T : [1, n] \to \mathbb{R}^3$ with $n \in \mathbb{N}$ that satisfies the following conditions: (1) $T(1) = (x_1, y_1, t_1)$, $T(2) = (x_2, y_2, t_2)$, ..., $T(n) = (x_n, y_n, t_n)$, such that $t_i < t_{i+1}$ for all $i \in \{1, \dots, n-1\}$; each (x_i, y_i, t_i) is called a <u>vertex</u> of the trajectory T; (2) For each $0 \leq \lambda \leq 1$ and for each $i \in \{1, \dots, n-1\}$, $T(i + \lambda) = (1 - \lambda)T(i) + \lambda T(i + 1)$. For every point (x, y, t) on the trajectory we say that (x, y) is the <u>expected location</u> at time t. The projection of T on the X-Y plane is called the <u>route</u> of T.*

Intuitively, a trajectory defines the location of a moving object in the X-Y plane as an implicit function of time t. The object is at (x_i, y_i) at time t_i. The vertices of a trajectory represent, for example, the readings of the GPS receiver on board a vehicle or other moving object. During each segment $[t_i, t_{i+1}]$ we assume that the object moves along a straight line, at constant speed, from (x_i, y_i) to (x_{i+1}, y_{i+1}). Thus the location of the moving object at a point in time t between t_i and t_{i+1}, $(1 \leq i < n)$, called the expected location at time t, is obtained by a linear interpolation between (x_i, y_i) and (x_{i+1}, y_{i+1}).

An illustration of a trajectory and its route is shown in Figure 1.

Trajectories may impose tremendous storage requirements when the location is sampled frequently. To address this problem, we propose to tradeoff accuracy for efficiency using line simplification. The subject of line simplification has been extensively studied in computational geometry and in

[2]The trajectory of a moving object is the sequence of (x, y, t) points that represents a trip of the object.

[3]We also conducted experiments on a trajectories dataset that consists of 1,000 trajectories describing the motion plans of objects in Chicagoland, which we generated by selecting random source-destination pairs for a trip, on an electronic map. Unfortunately, due to space limitations we will not describe these results here.

[4]i.e. the distance between the answers on the original trajectory and the approximation.

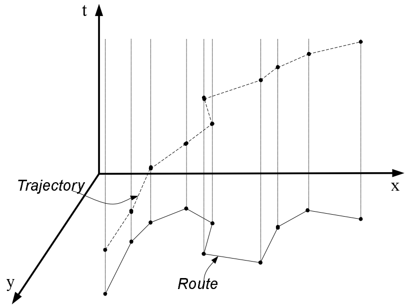


Figure 1: A trajectory and its two dimensional route

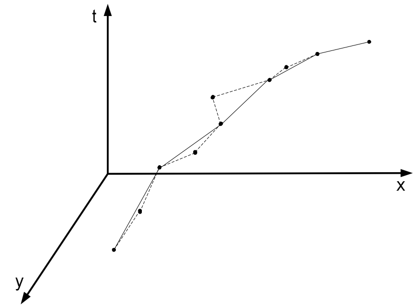


Figure 2: A simplification (solid line) of trajectory in Figure 1

many practical applications such as cartography, computer graphics, image processing [3, 8, 6, 14, 16, 18] since the 1970's. The goal was similar to ours: given a polygonal curve, approximate it by another one which is "not very far" from the original, but has fewer points. In contrast to our present work, these references did not consider the implication of the approximation on query processing.

Now we precisely define the "not very far" statement in the context of trajectories. Let M be the distance between a 3D point and a 3D line. The distance $d_M(p,T)$ between p and a trajectory T is the minimum (among all line segments of T) M-distance between p and a line segment of T. The distance between two trajectories is the *Hausdorff distance*[4] between them. The Hausdorff M-distance from a trajectory T to another trajectory T' is defined as

$$\tilde{D}_M(T,T') = \max_{p\in T} d(p,T')$$

i.e. the Hausdorff distance from T to T' is the maximum distance from a point of T to T'.

The symmetric Hausdorff distance between T and T' (or, for short, the Hausdorff distance between two trajectories) is defined as $D_M(T,T') = \max(\tilde{D}_M(T,T'), \tilde{D}_M(T',T))$; i.e. it is the maximum of the distances from T to T' and from T' to T.

Definition 2. *Let $\{p_1, p_2, \ldots p_n\}$ denote the set of vertices of a given trajectory T. For a subset $\{p'_1, p'_2, \ldots, p'_s\} \subseteq \{p_1, p_2, \ldots p_n\}$, denote by T' the trajectory with these vertices. Let ε be a real number. We say that T' is an <u>ε-simplification</u> of T with respect to M (equivalently, T' is <u>a simplification</u> of T with an <u>M-tolerance</u> ε), denoted by $T' = S(T,\varepsilon,M)$, if $D_M(T,T') \le \varepsilon$.*

Figure 2 shows a simplified trajectory corresponding to the original trajectory depicted on Figure 1.

One comment is that traditionally, the definition of simplification (used in computational geometry, cartography, computer graphics, etc.) considers only the distance from the original trajectory to the simplification, and not the symmetric distance as we do here. However, in databases, since queries operate on the simplifications the symmetric distance has to be bounded. However, for practical purpose the distinction is mute because we have proven that all the distances discussed in this paper are symmetric. Thus, if the distance from T to T' is bounded by ε so is the distance from T' to T. Due to space limitations, this analysis is omitted from this paper.

For a given trajectory T and a tolerance ε, an *optimal ε-simplification* is an ε-simplification with a minimum number of vertices. The optimal ε-simplification can be found using dynamic programming techniques straightforwardly in $O(n^3)$ time, or in quadratic running time using improved algorithms[3, 6, 16]. For better performance, heuristic-based approaches are often used in practice, especially in GIS. Among them, the best known and studied algorithm is Douglas and Peucker's (DP) [8].

2.2 Distance Functions

When simplifying 3D trajectories one needs to carefully consider the choice of the dimensionality and the distance used in the simplification algorithms. For example, one possibility is to simplify the 3D trajectory using the Euclidean distance. Another possibility is to use a 2D simplification by projecting the trajectory onto its 2D route, and then raising back to 3D by considering the time of the vertices in the simplified route. As we will demonstrate shortly (and by experiments later), the choice of distance function and algorithm impacts the amount of storage savings. However, we will also demonstrate in the next section that the choice of the distances affects the "quality" (i.e. error) of the answers to spatio-temporal queries.

In this section, we focus on the distances and discuss their applicability in the line simplification algorithm. Let $p_m = (x_m, y_m, t_m)$ denote a point, and $\overline{p_i, p_j}$ denote the straight line segment between the vertices $p_i = (x_i, y_i, t_i)$ and $p_j = (x_j, y_j, t_j)$ of a trajectory T. The distances between the p_m and the straight line segment $\overline{p_i, p_j}$ are defined as follows:

• E_2 – The two dimensional Euclidean distance, defined as: $E_2(p_m, \overline{p_i p_j}) = \sqrt{(x'_m - x'_c)^2 + (y'_m - y'_c)^2}$, where $p'_c = (x'_c, y'_c)$ is the point on the 2D straight line segment $\overline{p'_i p'_j}$ (i.e. the 2D projection of $\overline{p_i p_j}$) which is closest in terms Euclidean distance to $p'_m = (x'_m, y'_m)$ (the 2D projection of $p_m = (x_m, y_m, t_m)$).

• E_3 – The three dimensional Euclidean distance, defined as:
$E_3(p_m, \overline{p_i p_j}) = \sqrt{(x_m - x_c)^2 + (y_m - y_c)^2 + (t_m - t_c)^2}$
where $p_c = (x_c, y_c, t_c)$ is the point on $\overline{p_i p_j}$ which is closest to $p_m = (x_m, y_m, t_m)$.

• E_u – The three dimensional time_uniform distance is defined when t_m is between t_i and t_j, as follows:
$E_u(p_m, \overline{p_i p_j}) = \sqrt{(x_m - x_c)^2 + (y_m - y_c)^2}$
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_m (i.e. $t_c = t_m$). In other words, the time_uniform distance is the 2D Euclidean distance between p_m and the 3D point on $\overline{p_i p_j}$ at time t_m. This distance func-

tion is defined since, in contrast to the Euclidean distances, it guarantees bounded-error answers to spatial queries on trajectories (see next section).

• E_t – The time distance is defined as: $E_t(p_m, \overline{p_i p_j}) = |t_m - t_c|$, where p_c is the point on the 2D projection on the X-Y plane $\overline{p'_i p'_j}$ which is closest in terms of the Euclidean distance to p'_m, the 2D projection of p_m. In other words, intuitively, to find the time distance between p_m and the line segment proceed as follows. First project both on the X-Y plane, then find the point p'_c on the projected segment which is closest to p'_m, and finally find the difference between the times of p_c and p_m. This distance is defined here since it guarantees bounded-error answers to temporal queries on trajectories.

The distances defined above are illustrated in Figure 3. As a consequence of their respective definitions, the relationships among E_2, E_3, and E_u, are expressed by the following claim:

Claim 1. *Given a 3D point p_m and a line segment $\overline{p_i p_j}$ between two vertices of a trajectory, if t_m is between t_i and t_j, then $E_2(p_m, \overline{p_i p_j}) \leq E_3(p_m, \overline{p_i p_j}) \leq E_u(p_m, \overline{p_i p_j})$.*

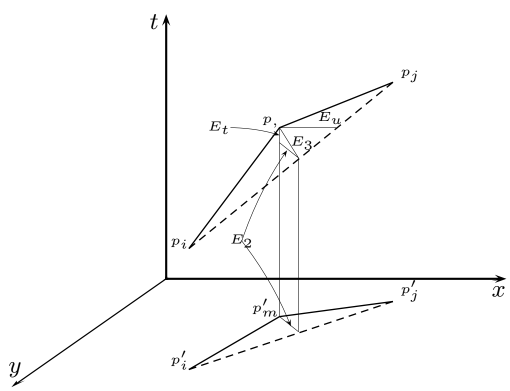


Figure 3: The relationship among the distances.

E_t does not have a straightforward relationship to the other distances. It can be shown that E_t is smaller than the E_u divided by the average speed between p_i and p_j.

Claim 1 implies that when E_2 is used in a simplification with a given tolerance ε, more vertices of a trajectory will be eliminated than when E_3 is used with ε. Similarly, using the E_3 distance will "save" more points than using the E_u distance. More formally, as a consequence of Property 1, we have (Let $\|T\|$ denote the "size", i.e. the number of vertices of a trajectory T):

Corollary 1. *Let T be a trajectory and ε a tolerance. Let $T'_2 = S(T, \varepsilon, E_2)$, $T'_3 = S(T, \varepsilon, E_3)$ and $T'_u = S(T, \varepsilon, E_u)$ denote the respective optimal ε-simplifications. Then $\|T'_2\| \leq \|T'_3\| \leq \|T'_u\|$.*

3. BOUNDED ERROR QUERIES ON SIMPLIFIED TRAJECTORIES

In this section we will analyze the relationship between the distances and the error in the query answers. As we mentioned, our desiderata is to have a bound on the error produced when answering spatio-temporal queries.

In the first subsection we define the types of spatio-temporal queries that we analyze in the rest of this section. In the second subsection we define the notion of soundness for a (query-type, distance) pair, i.e., to produce a bounded-error answer to the query, on a bounded error trajectory-approximation (where the approximation is according to the distance). In the third subsection we analyze several of the above pairs, and conclude that none of the previously defined distances is sound for all types of spatio-temporal queries. Thus we define multidistance trajectory simplification, and show that the combination of E_t and E_u is sound for all the query types, except the join. In the fourth subsection we show that this combination is sound for the join as well.

3.1 Spatio-temporal queries

Most spatio-temporal queries are composed of the following five types of queries, *where_at, when_at, intersect, nearest_neighbor* and *spatial_join*. We introduce the semantics of each one of the operators on a trajectory $T = (x_1, y_1, t_1)$, (x_2, y_2, t_2), ..., (x_n, y_n, t_n), as follows:

• *where_at*(T, t) – returns the expected location (c.f. Definition 1) at time t. If $t < t_1$ or $t > t_n$ then the operator is undefined.

• *when_at*(T, x, y) – returns the time t at which a moving object on trajectory T is expected to be at location (x, y). If the location does not belong to the route of the trajectory, or the moving object visits the location more than once, or is stationary at the location, then the operator is undefined.

• *intersect*(T, P, t_1, t_2) – is *true* if the trajectory T intersects the polygon[5] P between the times t_1 and t_2. (This is also called a spatio-temporal range query).

• *nearest_neighbor*(T, O, t) – The operator is defined for an arbitrary set of trajectories O, and it returns a trajectory T' of O. The object moving according to T', at time t, is closest than any other object of O to the object moving according to T.

• *spatial_join*(O, th) – O is a set of trajectories and the operator returns the trajectory pairs (T_1, T_2) such that their distance (according to some distance functions) is less than the threshold th. The distance used in the join may be different than the distance used for simplification. This operator will be further discussed in the last subsection.

Clearly, the composition of these query types can express more complex queries. For example "Retrieve the 2PM location of the moving objects which will intersect the Parks P_a and P_b between noon and 5PM" (assuming that the parks are represented as polygons).

3.2 Desiderata for Soundness of Distances

When querying simplified trajectories, the answers may deviate from those on the original trajectories. To incorporate trajectory reduction techniques in MOD systems, the imprecision introduced by line simplification must be managed. We introduce a way of doing so, based on the following observation. If the users can predict a priori, namely before data reduction, the maximum error (or imprecision) δ of answers to queries for each given simplification tolerance ε, then the simplification can be restricted to tolerances ε for which the imprecision is acceptable. Observe that in this scheme the maximum imprecision depends on the simplifi-

[5]for simplicity of exposition we will assume throughout this paper that the polygons are convex.

cation tolerance, but not on the individual trajectory. This scheme is possible only if the simplification distance is sound.

Now we explain the notion of soundness. Let $q(T)$ denote the answer of some spatio-temporal query q with respect to a trajectory T. Similarly, let $q(T')$ denote the answer of the same query q when posed to a simplification T'. Intuitively, if $T' = S(T, \varepsilon, E)$, we say that distance function E is sound for q when there exists a bound δ on the distance between the two answers. More precisely, if we let *dist(q(T),q(T'))* denote the distance between the two answers, the soundness of E means that for every ε there exists a δ such that for every trajectory *dist(q(T),q(T'))* $\leq \delta$.

We formalize this notion for each of the queries described above(except for *spatial_join*, which is separately discussed in the following subsection), as follows:

Definition 3. *A distance function E is sound for the respective query if it satisfies the following: For every simplification tolerance ε, there exists a positive number δ, called the answer error bound, such that for every trajectory T and for every simplification $T' = S(T, \varepsilon, E)$:*

- where_at – *For every t for which both T and T' are defined, let $(x, y) = where\_at(T, t)$ and let $(x', y') = where\_at(T', t)$. Then, the distance between (x, y) and (x', y') is bounded by δ, namely $\sqrt{(x' - x)^2 + (y' - y)^2} \leq \delta$.*
- when_at – *Let $t' = when\_at(T', x, y)$, and let $t = when\_at$ (T,x,y). If both t and t' are defined, then $|t - t'| \leq \delta$.*
- intersect – *For any polygon P, if $intersect(T', P, t_1, t_2)$* is true, *then there exists a time $t \in [t_1, t_2]$ such that the expected location of the original trajectory T at time t is no further than δ from $P \cup$ interior of P. Conversely, if $intersect(T', P, t_1, t_2)$* is false, *then for every $t \in [t_1, t_2]$, the expected location of the original trajectory T at time t is either outside P, or, if inside, it is within δ of a side of P (i.e. it does not penetrate P by more than δ).*

Intuitively, this means that if the simplification T' intersects P, then T is not further than δ from P; and if T' does not intersect P, then T does not intersect P, or intersects it "very little". Thus, the user, knowing that the query addresses approximate trajectories, may decide to adjust the polygon P accordingly.

- nearest_neighbor – *Let O be an arbitrary set of trajectories and let $o = nearest\_neighbor(T, O, t)$ and let $o' = nearest\_neighbor(T', O, t)$. Let d_o be the Euclidean distance between o and T at time t, and let $d_{o'}$ be the Euclidean distance between o' and T at time t. Then $|d_o - d_{o'}| \leq \delta$.*

Intuitively, it means that the difference between the distances (o to T) and (o' to T) is bounded by δ. In other words, for any set of trajectories (or moving objects) O, at any time t, the error of the nearest neighbor query is at most δ.

3.3 Soundness of the Distances

Now we inspect the soundness of the distances E_2, E_3, E_u and E_t with respect to the query types. There are 16 possible combinations of distances and query types (four distances and four types). However, we reduce the number of combinations to inspect by studying subsumption relationships among query types and among distances. Then we prove the soundness or unsoundness of the necessary combinations individually. It turns out that no single distance introduced is sound for all the query types. Thus, we introduce simplification with multiple distances and prove that the distance combining E_u and E_t is sound for all query types.

Consider two distances M_1 and M_2. Suppose that for every pair of trajectories T and T', if $M_1(T, T') \leq \varepsilon$, then $M_2(T, T') \leq \varepsilon$. In this case, we say that distance M_1 is *weaker* than M_2, denoted as $M_1 \leq M_2$. The following relationship among distances is a consequence of Claim 1.

Corollary 2. $E_u \leq E_3 \leq E_2$

The following subsumption relationships hold among distances with respect to soundness.

Theorem 1. *For two distances M_1 and M_2, if $M_1 \leq M_2$, then for every query type Q for which M_2 is sound, M_1 is also sound.*

The following subsumption relationships hold among queries with respect to soundness.

Theorem 2. *For any distance function M, if it is sound for the where_at query type, then it is also sound for the intersect and nearest_neighbor types. Furthermore, if M is not sound for the where_at query type, then it is also not sound for the intersect and nearest_neighbor types.*

Theorem 3. *The E_3 distance is not sound for the where_at query type.*

Together with Corollary 2 and Theorem 1, the above theorem implies that

Corollary 3. *The E_2 distance is not sound for the where_at query type.*

Theorem 4. *The E_u distance is sound for the where_at query type. Furthermore, for any simplification tolerance ε, the answer-error-bound of where_at is equal to ε.*

Together with Theorem 2, the above theorem implies

Corollary 4. *The E_u distance is sound for the intersect and nearest_neighbor query types.*

It can also be shown that for the distance E_u, for any simplification tolerance ε, the answer-error-bound of the intersect query type is equal to ε. The bound of the nearest_neighbor query type depends on whether or not the set of trajectories O is simplified; it is ε if O is not simplified, and 2ε if it is.

Theorem 5. *The distance E_u is not sound for the query type when_at.*

Theorem 6. *The distance E_t is sound for the when_at query type. Furthermore, for any simplification tolerance ε, the answer-error-bound of when_at is equal to ε.*

Theorem 7. *The distance E_t is not sound for the where_at query type.*

We can summarize the above results in the following table.

Table 1 indicates that there is not a single distance that is sound for the four spatio-temporal query types we have studied. Thus, to produce bounded-error answers to the four query types, a trajectories database will need to be simplified using a combination of distances. We do so as follows. Given two distance functions between trajectories $M1$ and $M2$, we define the combined distance, denoted $M1 \wedge M2$, as: $M1 \wedge M2(T, T') = \max\{M1(T, T'), M2(T, T')\}$.

The following is easy to prove based on the definitions:

	Where_at	When_at	Intersect	Nearest Neighbor
E_2	No	No	No	No
E_3	No	No	No	No
E_u	Yes	No	Yes	Yes
E_t	No	Yes	No	No

Table 1: The soundness of the distances for spatio-temporal query types.

Claim 2. $M_1 \wedge M_2 \leq M_1$, $M_1 \wedge M_2 \leq M_2$

Consequently:

Claim 3. *For a distance M_1 and a distance M_2, let the set of sound operators of M_1 be $SO(M_1)$ and the set of sound operators of M_2 be $SO(M_2)$. Then the multi-distance $M_1 \wedge M_2$ is sound for all the query types in set $SO(M_1) \cup SO(M_2)$.*

Thus:

Corollary 5. *The distance $E_u \wedge E_t$ is sound for all the spatio-temporal query types. For any simplification tolerance ε, the answer-error-bound is ε for where_at, when_at and intersect, and 2ε for nearest-neighbor.*

In conclusion, the appropriate distance to use in a simplification depends on the type of queries expected on the database of simplified trajectories. If all spatio-temporal queries are expected, then $E_u \wedge E_t$ should be used. If only where_at, intersect, and nearest_neighbor queries are expected, then a more concise approximation with the same answer-error-bound can be achieved by using the E_u distance. If only when_at queries are expected, then the E_t distance should be used.

3.4 Spatial Join

The *spatial join* between trajectories is separately discussed in this section. As mentioned in section 3.1, the definition of spatial join depends on the distance function between trajectories. For example, two of the most common distance functions are the Hausdorff distance (defined in section 2) and the *mean square root error*(MSRE) defined as

$$D(T, T') = \frac{1}{t_e - t_s} \int_{t_s}^{t_e} \sqrt{(x(t) - x'(t))^2 + (y(t) - y'(t))^2} dt$$

where t_s and t_e are the start time and the end time of the comparison time period.

The soundness for the spatial join operation is defined as follows. A distance M is *sound* for the spatial-join operation with a distance function D if it satisfies the following: For every real positive number ε, there exists a real number δ such that for every trajectory T_1 and every simplification $T_1' = (T_1, \varepsilon, M)$ and for every trajectory T_2 and every simplification $T_2' = S(T_2, \varepsilon, M)$, $|D(T_1, T_2) - D(T_1', T_2')| \leq \delta$.

Theorem 8. *Consider a spatial-join with a distance function D. A distance M is sound for the spatial-join if D is a metric and $M \leq D$.*

Together with Corollary 2, Theorem 8 implies

Corollary 6. *E_u is sound for the spatial-joins with the distance function E_2, E_3 or E_u.*

Based on Theorem 8, we get

Theorem 9. *E_u is sound for the spatial-join with the MSRE distance function.*

4. AGING OF THE TRAJECTORIES

Often, the older the information gets, the less precision is necessary. For example, it is possible that the coarseness of the trajectory approximation is allowed to increase as time progresses. In this case a data aging mechanism can be introduced.

Assume, for example, that the error bound on the queries is to be ≤ 0.1 mile after the first month; ≤ 0.2 after the second month; ≤ 0.3 after the third month; etc. Then, one can simplify the original trajectory T to $T' = S(T, 0.1, M)$ after the first month, to $T'' = S(T, 0.2, M)$ after the second month, etc. However, a problem arises. At the beginning of the second month one needs to generate $T'' = S(T, 0.2, M)$, but the original trajectory T does not exist anymore, only $T' = S(T, 0.1, M)$. By simplifying T' can one obtain the trajectory T'', or a same-size trajectory? If so, should one simplify T' by $\varepsilon = 0.1$ or $\varepsilon = 0.2$, or $\varepsilon = 0.2 + 0.1$, or some other value? It turns out that the answer to these questions depends on the simplification algorithm.

Theorem 10. *Let M be a distance, and let $\varepsilon_1 < \varepsilon_2$ be two tolerances. For an arbitrary trajectory T, let T_1'' be an ε_2-simplification of the ε_1-simplification of T; both simplifications are by an algorithm that produces the optimal ε-simplification. Then there are trajectories for which T_1'' is an $(\varepsilon_1 + \varepsilon_2)$-simplification of T but not an ε_2-simplification of T.*

The above theorem indicates the following. Suppose that the allowed bounds on the simplification errors are ε_1 for the first month, ε_2 for the second month, etc. Then the aging procedure after the first month will not work. Specifically, if one simplifies the (once-simplified) trajectories database by ε_1, then no further data reduction will result; if one simplifies it by ε_2 then the resulting database may have trajectories that violate the aging specification, i.e. they may have an error (compared to the original trajectory) that exceeds the allowable ε_2 error. Thus, to use the optimal algorithm, one needs to specify a sequence of tolerances ε_1, ε_2, ε_3, such the input database is simplified by ε_1, the resulting database by ε_2, the resulting database by ε_3, etc. And the approximation errors are bounded by ε_1, $\varepsilon_1 + \varepsilon_2$, $\varepsilon_1 + \varepsilon_2 + \varepsilon_3$, etc., respectively.

Now consider the aging using the DP algorithm[14]. The DP algorithm recursively approximates a given polyline by a "divide and conquer" technique, where the farthest distance point is used to select the divide point in the polyline. Given a *begin_vertex* p_i and an *end_vertex* p_j, if the greatest distance from some vertex p_k to the line segment $\overline{p_i p_j}$ is greater than the tolerance ε, break the trajectory into two pieces at p_k and recursively call the procedure on each of the sub-chains $\overline{p_i p_k}$ and $\overline{p_k p_j}$; Otherwise, the vertices between p_i and p_j are removed from trajectory and this segment is simplified as a straight line $\overline{p_i p_j}$. Thus:

Theorem 11. *Let M be a distance, and let $\varepsilon_1 < \varepsilon_2$ be two tolerances. Then a trajectory T simplified by the DP algorithm (DP-simplified for short) using ε_2 is the same as T DP-simplified using ε_1 first, and subsequently DP-simplified by ε_2. In other words:*
$S(T, \varepsilon_2, M) = S(S(T, \varepsilon_1, M), \varepsilon_2, M))$

The above theorem indicates that the DP algorithm is much more conducive to data-aging in the following sense.

Suppose that the allowed bounds on the simplification errors are ε_1 for the first month, ε_2 for the second month, etc. Then using the DP algorithm one can simplify the input database by ε_1, then the resulting database by ε_2, then the resulting database by ε_3, etc.

5. EXPERIMENTAL STUDY

In this section we give a description of our experimental results and the conclusions based on them. In the first subsection we describe the setting for the experiments, namely the input datasets, the methodology and the environment. Then, in the following subsection, we compare error of the line simplification and the wavelet transform. We also analyze the data reduction obtained by the different distances and by two simplification methods, the DP algorithm and the optimal one. Finally, We compare the execution times of the DP algorithm and the optimal one.

5.1 Datasets and Methodology

The dataset we used in the experiment consists of 38 trajectories constructed from GPS traces. The traces are obtained from the on-board GPS receivers on the UCLA campus shuttle buses. The location was sampled every second[6]. The data was collected on April 24th, 2002. Each trajectory represents the continuous trip of a UCLA campus shuttle-bus on that day. The average number of vertices per trajectory is 7085 and the average length of a trajectory is 16.352 miles.

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. before simplification)*. In other words, the storage savings of the simplification is $1 - rr$. For each data reduction experiment we varied the simplification tolerance ε from 0.05 mile to 1 mile.

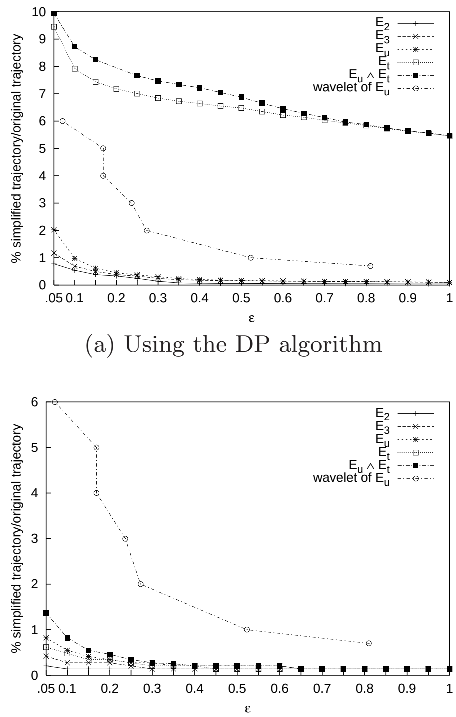
All experiments reported were performed on a Pentium III 866MHz machine with 512MB of SDRAM main memory, running on Suse Linux.

5.2 Experiment Results

	MSRE		E_u	
Ratios	DP	Wavelet	DP	wavelet
1%	0.0267	0.0410	≈0.098	0.522
2%	0.0124	0.0203	≈0.051	0.272
5%	0.0032	0.0076	≈0.013	0.168
10 %	0.0011	0.0035	≈0.004	0.037

Table 2: The average MSRE and E_u errors for varying compression ratios for the UCLA dataset.

First, we compared the errors of wavelets and line simplification as measured by the MSRE and E_u distances. It is important to observe that wavelets does not provide a bound on the error, but we measured the error for every compression ratio obtained by wavelets. We used the Haar wavelets variant[1] and the DP algorithm in the comparison. The wavelets method has been shown to outperform others such as DFT and DCT[1, 13]. The results of this comparison are shown in Table 2. It shows that the error of the DP algorithm is consistently lower than that of wavelets. According to the MSRE, DP is at most 65% of wavelet, and according to E_u it is at most 20%.



(a) Using the DP algorithm

(b) Using the optimal algorithm

Figure 4: The reduction ratio with different tolerances

We investigated the nature of the savings of each of the four distances E_2, E_3, E_u and E_t [7] and the combined simplification of $E_u \wedge E_t$. Figure 4 presents the *average* size of the reduced trajectory as a percentage of that of the original average trajectory, for each one of the distances. The sixth curve represents wavelet compression using E_u. First, for all distances, the reduction ratio monotonically decreases as the value of ε increases. Overall, in all cases, the reduction obtained by the optimal algorithm is several times better than the reduction of the DP heuristic, with the exact value depending on the distance and the tolerance. However, the savings of both methods is over 90%. One can also observe that Corollary 1 is verified by the experiment. Namely, E_2 has better savings than E_3, and E_3 is better than E_u. Additionally, according to both algorithms E_u compresses the trajectories more than $E_t \wedge E_u$; this is also true for E_t. Another observation is that the DP heuristic is not doing well for the E_t and the $E_u \wedge E_t$ distances. Specifically, we see that for the optimal algorithm the savings for these distances ranges from 98.6% to 99.8% depending on the tolerance ε. However, for the DP heuristic, this range is 90.5% to 94%.

Next we compared the time-performance of the optimal and DP algorithms. The time complexities of these algorithms for the various distances are given in Table 3(algorithms for E_2, E_3 and E_u were studied more extensively, and there-

[6]For more information about UCLA shuttle trajectory, please visit http://www.cs.ucla.edu/ cjlai/bustrack/

[7]In order to plot the E_t distance on the same graph as the other distances, the tolerances of the E_t distance are normalized to distance by dividing the time-tolerance ε by the average speed of the trajectory.

	DP	Optimal
E_2	$O(n \log n)$	$O(n^2)$
E_u & E_3	$O(n^2)$	$O(n^2 \log n)$
generic(arbitrary distance)	$O(n^2)$	$O(n^3)$

Table 3: Time complexities of the DP algorithm and the Optimal algorithms.

ε**(mile)**	0.05	0.1	0.2	0.5	1
OP(ms)	13118	32228	42768	42625	42336
DP(ms)	1.535	1.136	0.63	0.641	0.624

Table 4: Comparing the per trajectory average running time of the optimal(OP) and the DP algorithm(DP), using E_2.

fore the time complexities of the generic algorithms were improved). The DP algorithm has a better performance asymptotically in all cases. We have also experimentally compared the running time for the optimal and DP algorithms for the E_2 distance(for the other distances, the optimal algorithms will fall farther behind the DP heuristic). The experimental results are summarized in Table 4. As the results indicate, for the average trajectory in our dataset, the DP algorithm is between 8546 and 67846 times faster than the optimal algorithm, with the advantage of DP increasing with the tolerance ε.

In conclusion, the reduction obtained by the optimal algorithm is several times better than the reduction of the DP heuristic. However, the storage savings of both methods is over 90%. But, the DP algorithm is approximately 10^4 times faster than the optimal algorithm.

6. RELATED WORK

Line simplification has been well-studied from various perspectives: geographic information systems [8, 18]; digital image analysis[15]; and computational geometry[3, 6]. There are two variants of the problem: (1). *min-# problem* – given a tolerance ε, compute an approximation of original polygonal chain (polyline) C, with smallest number of vertices k_{min}; and (2). *min-ε problem* – given a number of vertices k (for the reduced polyline), compute an approximation of the original polyline C with at most k vertices and minimal error ε_{min}. Our approach to the trajectory reduction is a min-# problem, since our goal was to to obtain a reduction which ensures a bound on the error of the answer to the important spatio-temporal queries for all the trajectories in a moving objects database.

Most of the works on line simplification [3, 6] follow the graph-theoretic approach, as introduced by Imai and Iri [16]. The optimal algorithms for simplifying 2D polygonal chains run in $O(n^2)$ time for any Euclidian metrics ($O(n^{4/3+\delta})$ for L_1 and L_∞ metrics[3]). As we have demonstrated, for our problem domain Douglas-Peuker algorithm produces simplification results which are very close to the optimal ones [6] except for E_t, and it has much better running time.

Data compression is a very popular topic in the database research (e.g. [12, 25]). The techniques are targeted towards reduced storage requirements and improved I/O performance. When it comes to generating the answers to the queries, there are two main categories of approaches: 1. The data is decompressed when answering a query [7]; and 2. The compressed data is used to answer the query, and the answer contains some error [11, 5, 10]. Our approach is *lossy* (i.e we do not recover the original trajectories after simplification) and we aim at utilizing the reduced/ simplified trajectories to get faster response. Thus, our results cannot be directly compared to the first category of works above, which decompress the data when answering the queries. As an example of the second category, recently *wavelets* have become a popular paradigm for data reduction which provides fast and "reasonably approximate" answer to queries [5, 10]. The original data is reduced to compact sets of coefficients (*wavelet synopses*) which are used to answer the queries. The main difference with our approach is that these works either do not ensure a bound on the error to the query answers or ensure an *asymptotic/ probabilistic* bounds on the error. Similar observation holds for the works which use *histograms* or *sampling* to compress the data and provide a reasonably accurate answer to the queries ([1] provides a survey of several data reduction techniques). In contrast, in our approach we address the *min-#* problem but we ensure a deterministic bound on the error of the answers to the spatio-temporal queries.

A good survey on location modelling is provided in [21]. Moving objects databases have actively been studied from several aspects: 1. *modelling and querying* [9, 23]; 2. *indexing* in primal or dual space [2, 17]; 3. *uncertainty* and its impact on the queries [19, 22]. However, to the best of our knowledge, none of these works addressed the issue of simplification from the aspect of storage and processing savings.

7. CONCLUSIONS AND FUTURE WORK

In this paper we addressed the problem of spatio-temporal data reduction, particularly the reduction of sets of (x, y, t) records aggregated into trajectories. The data reduction is by line simplification, a technique that guarantees bounds on the error of the approximated trajectories. Experimental results have shown that when an error of 0.1 mile is allowed, the average trajectory is reduced by more than 99%. Unexpectedly, the bounded-error approximation may produce answers to queries for which the error is unbounded. In other words, even though the approximation is bounded-error, there are query types that when posed on this approximation produce answers whose error is unbounded. It turns out that the type of approximations for which this undesirable phenomenon, called unsoundness, arises depends on the distance used to approximate the trajectories and the type of spatio-temporal query. For example, the Euclidean distances (in two and three dimensions) are unsound for the query that asks "where is a particular moving object at a given time", i.e. the query, when posed on the simplified trajectory, may produce an answer which is arbitrarily far from the answer to the same query posed on the original trajectory. In our opinion, soundness is a new important concept in database research. This paper provides a classification of (approximation-distance, query-type) pairs into sound and unsound sets.

We also discussed the aging of trajectories, namely producing increasingly compact (but also coarser) approximations of trajectories over time. Here an interesting phenomenon was discovered, namely that the optimal simplification algorithm (i.e. the one that produces minimum-size trajectories for a given error bound) is "aging-unfriendly" in the sense that it cannot be naturally used in aging. In contrast, the DP heuristic, which provides good but not optimal approximations, is "friendly". This concept of aging-friendliness is explained carefully in section 4, we believe that it will also prove important in other types of approximations. This is the subject of future work.

Finally, we compared experimentally the performance of the optimal algorithm versus the DP heuristic. We have shown that both achieve a data-reduction of at least 90% even for an approximation-error tolerance of 0.05 miles or less, but the optimal algorithm saves over 98%. The exact storage saving of each algorithm depends both, on the distance and on the approximation-error tolerance. However, experimental results show that the DP heuristic on trajectories with thousands of (x, y, t) records is at least 10,000 time faster than the optimal algorithm. We have also shown that savings of line simplification outperforms wavelets.

8. REFERENCES

[1] Special issue on data reduction techniques. In *IEEE Data Engineering*, volume 20. 1998.

[2] A. K. Agarwal, L. Arge, and J. Erickson. Indexing moving points. In *19th ACM PODS Conference*, 2000.

[3] P.K. Agarwal and K. R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discrete & Computational Geometry*, 23:273–291, 2000.

[4] Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation A survey. Technical Report B 96-11, 1996.

[5] Kaushik Chakrabarti, Minos Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Approximate query processing using wavelets. In *VLDB 2000*, Septermber 2000.

[6] W. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *International Journal of Computational Geometry Applications*, 6:50–77, 1996.

[7] Zhiyuan Chen, Johannes Gehrke, and Flip Korn. Query optimization in compressed database systems. In *SIGMOD 2001*, pages 271–282. ACM Press, 2001.

[8] 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.

[9] L. Florizzi, R. H. Gutting, E. Nardelli, and M. Schneider. A data model and data structures for moving objects databases. In *ACM SIGMOD*, 2000.

[10] M. Garofalakis and P. B. Gibbons. Wavelet synopses with error guarantees. In *ACM SIGMOD*, 2002.

[11] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. In *VLDB*, 1997.

[12] G. Graefe and L. D. Shapiro. Data compression and database performance. In *Proc. ACM/IEEE-CS Symp. on Applied Computing*, 1991.

[13] V. Hardle, G. Kerkyacharian, D. Picard, and A. Tsybakov. *Wavelets, Approximation, and Statistical Applications.* Springer, 1998.

[14] J. Hershberger and J. Snoeyink. Speeding up the douglas-peucker line-simplification algorithm. In *the 5th Int. Symp. on Spatial Data Handling*, 1992.

[15] J. D. Hobby. Polygonal approximations that minimize the number of inflections. In *ACM-SIAM SODA*, 1993.

[16] H. Imai and M. Iri. Polygonal approximations of a curve-formulations and algorithms. In *Computational Morphology*, pages 71–86. 1988.

[17] D. Kollios, D. Gunopulos, and V. J. Tsotras. On indexing mobile objects. In *18th ACM PODS Conference*, 1999.

[18] R. McMaster. Automated line generalization. *Cartographica*, 24(2):74–111, 1987.

[19] D. Pfoser and C. Jensen. Capturing the uncertainty of moving objects representation. In *SSDB*, 1999.

[20] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel approaches in query processing for moving object trajectories. In *VLDB*, 2000.

[21] Evaggelia Pitoura and George Samaras. Locating objects in mobile computing. *IEEE TKDE*, 13(4), 2001.

[22] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain. The geometry of uncertainty in moving objects databases. In *the 8th EDBT*, 2002.

[23] M. Vazirgiannis and O. Wolfson. A spatiotemporal model and language for movign objects on road networks. In *MOBIDE*, 2001.

[24] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multi-dimensional trajectories. In *the 18th ICDE*, San Jose, California, 2002.

[25] T. Westmann, D. Kossmann, S. Helmer, and G. Moerkotte. The implementation and performance of compressed databases. *SIGMOD Record*, (3), 2000.

APPENDIX

A. APPENDIX : PROOFS

A.1 Proof Sketch of Theorem 1

Given two distances $M_1 \leq M_2$, for every trajectory T and every tolerance ε_1, if T' is a ε-simplification of T with respect to M_1, i.e. $T' \in S(T, \varepsilon, M_1)$, then $T' \in S(T, \varepsilon, M_2)$. So, if M_2 is sound for Q with error bound $\delta = f(\varepsilon)$, M_1 is also sound for Q with the same error bound δ.

A.2 Proof Sketch of Theorem 2

Assume that M is sound for where_at. For an arbitrary trajectory T and an arbitrary tolerance ε, let T' be an ε-simplification of T with respect to M. If M is sound for *where_at*, then for every point $(x', y', t) \in T'$, there is a point $(x, y, t) \in T$ such that the Euclidean distance between (x, y) and (x', y') is less than ε.

Let P be a polygon. Consider the *intersect(*T', P, t_1, t_2*)* query. If it returns true, then there exists a point (x', y', t') such that $(x', y') \in P \cap T'$ and $t' \in [t_1, t_2]$. Due to the soundness of M for where_at, there is a point $(x, y, t') \in T$ such that (x, y) is ε close to (x', y') . This means that (x, y) is no further than ε from $\{P \cup$ interior of $P\}$.

Conversely, suppose that the intersect query returns false. That means that T' is outside of P. Therefore, every point of T is either outside of P or within ε of a side of P.

Consider the nearest_neighbor query. Let O be an arbitrary set of trajectories. Let $l(o, T, t)$ denote the distance between some trajectory $o \in O$ and the trajectory T at time t. Let o' and o be the nearest_neighbors of T' and T respectively.

Based on the soundness of where_at and the triangle inequality, we have:

$$|l(o', T, t) - l(o', T', t)| \leq \varepsilon \tag{1}$$

$$|l(o, T, t) - l(o, T', t)| \leq \varepsilon \tag{2}$$

Due to the fact that o' is the nearest neighbor of T' at time t and o is the nearest neighbor of T:

$$l(o', T', t) \leq l(o, T', t) \tag{3}$$

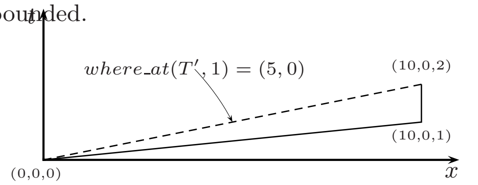
$$l(o, T, t) \leq l(o', T, t) \tag{4}$$

Based on (3), we can open the absolute value in inequality (2) and obtain $l(o', T', t) \leq l(o, T, t) + \varepsilon$. Similarly to this deduction, we obtain $l(o, T, t) \leq l(o', T', t) + \varepsilon$. Putting them together, we obtain $|l(o, T, t) - l(o', T', t)| \leq \varepsilon$.

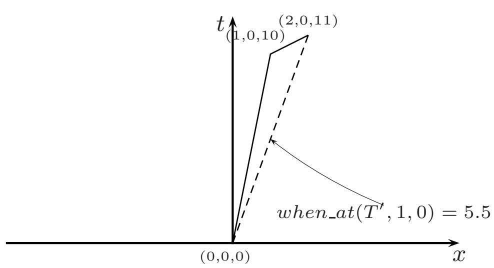
In the above proof, we assume that the trajectories of O are not simplified. If they are, a similar methodology can be used to prove that error of the nearest neighbor query answer is at most 2ε.

Now, assume that M is not sound for where_at. We will show that M is also not sound for the intersect and nearest_neighbor queries. There exists some ε such that for every δ there exist a trajectory T and a time t such that error of the where_at at time t is bigger than δ. We can find a polygon P such that T' is inside P at time t, but T is more than δ away from P. Thus, if E is not sound for where_at, it is also not sound for intersect query.

Similarly, we can prove that at some time t if the error of the where_at is unbounded, the error of the nearest_neighbor is also unbounded.



(a) Counter example for E_3



(b) Counter example for E_u

Figure 5: Counter examples. (The original trajectories are drawn in solid lines and simplifications in dashed lines.)

A.3 Proof Sketch of Theorem 3

Consider the following counterexample. Assume that an object m moves along the x-axis. For every tolerance ε and every answer error bound δ, suppose that the start location and time is (0, 0, 0) and since then m moves 10δ miles in ε minutes then stops there in next ε minutes. Then we have trajectory that represents the motion of m as $T = \langle p_1(0,0,0), p_2(10\delta, 0, \varepsilon), p_3(10\delta, 0, 2\varepsilon)\rangle$. Figure 5(a) shows an instance of this counterexample with $\varepsilon = 1$ and $\delta = 1$. T can be simplified by E_3 and ε as $T' = \langle (0,0,0), (10\delta, 0, 2\varepsilon)\rangle$. Then $dist(where\_at(T, \varepsilon), where\_at(T', \varepsilon))$
$= dist(10\delta, 0), (5\delta, 0)) = 5\delta > \delta$.

A.4 Proof Sketch of Theorem 4

The Euclidean uniform distance E_u requires that any pair of points $\langle (x, y, t)(x', y', t)\rangle$ has a Euclidean distance not greater than ε when they are projected onto the X-Y Euclidean space. Since (x, y) is uniquely determined by t, we have $where\_at(T', t) = (x'(t), y'(t))$ and $where\_at(T, t) = (x(t), y(t))$ for any time t, then
$dist(where\_at(T, t), where\_at(T', t)) \leq \varepsilon$. Let $\delta = \varepsilon$ and we have proven the theorem.

A.5 Proof Sketch of Theorem 5

A counter-example is as follows. Assume that an object m along the x-axis. For every tolerance ε and every answer error bound δ, there exists a trajectory T with three points $\langle p_1(0,0,0), p_2(\varepsilon, 0, 10\delta), p_3(2\varepsilon, 0, 11\delta)\rangle$. Figure 5(b) shows an instance of this counterexample with $\varepsilon = 1$ and $\delta = 1$. Consider an E_u ε-simplification T' consists of two points p_1 and p_3. If we query $when\_at(T, \varepsilon, 0)$, the answer is 10δ, while $when\_at(T', \varepsilon, 0) = 5.5\delta$. So, the distance is $4.5\delta > \delta$.

A.6 Proof Sketch of Theorem 6

Let $t = when\_at(T, x, y)$ and $t' = when\_at(T', x, y)$. By the definition of E_t, every point in every trajectory T and its closet point in the simplification T' are bounded by ε in their time difference. Remember that when_at is defined only when (x, y) is on the route of T and T'. Since T' is a ε-simplification of T according to E_t, $|t - t'| \leq \varepsilon$.

A.7 Proof Sketch of Theorem 7

We prove this theorem using the counter-example of Theorem 3. Clearly, that simplification is also an E_t simplification. However, the answer error is unbound, as we have illustrated in the proof of theorem 3.

A.8 Proof Sketch of Theorem 8

Let ε be an arbitrary positive real number. Consider a spatial join between two arbitrary trajectories T_1 and T_2. Let T_1' and T_2' be the ε-simplifications of T_1 and T_2 with respect to M, i.e. $M(T_1, T_1') \leq \varepsilon$ and $M(T_2, T_2') \leq \varepsilon$. We will show that $|D(T_1, T_2) - D(T_1', T_2')| \leq \varepsilon$ for all the spatial joins whose distance functions satisfy the conditions of the theorem. First, $D(T_1, T_1') \leq \varepsilon$ and $D(T_2, T_2') \leq \varepsilon$ because $M \leq D$. Meanwhile, since D is a metric, $D(T_1', T_2') < D(T_1, T_1') + D(T_1, T_2') \leq D(T_1, T_1') + D(T_1, T_2) + D(T_2, T_2')$, based on the triangle inequality. Thus, $D(T_1', T_2') - D(T_1, T_2) \leq D(T_1, T_1') + D(T_2, T_2') \leq 2\varepsilon$. Similarly, we have $D(T_1, T_2) - D(T_1', T_2') \leq 2\varepsilon$.

A.11 Proof Sketch of Theorem 11

The theorem is trivially true for the trajectory with only two or three vertices. For the trajectory T with $(n > 3)$ vertices, either it is simplified as a straight line, or it is divided at the vertex with furthermost distance and the simplification processes repeat on the two sub-trajectories, using the DP algorithm. Note that the vertex with furthermost distance is still the furthermost one of the simplification, so the simplification on simplification will follow the same division vertex and the same sub-trajectories as those of simplification of the original one. If $\varepsilon_1 < \varepsilon_2$, repeat the process recursively, we get $S(T, \varepsilon_2, E) = S(S(T, \varepsilon_1, E), \varepsilon_2, E))$.