Addressing the Need for Map-Matching Speed: Localizing Global Curve-Matching Algorithms

Carola Wenk Randall Salas
Department of Computer Science
University of Texas at San Antonio
San Antonio, TX 78249-0667, USA
{carola,rsalas}@cs.utsa.edu

Dieter Pfoser
RA Computer Technology Institute
Akteou 11
GR-11851 Athens, Greece
pfoser@cti.gr

Abstract

With vehicle tracking data becoming an important sensor data resource for a range of applications related to traffic assessment and prediction, fast and accurate map-matching algorithms become a necessary means to ultimately utilize this data. This work proposes a fast map-matching algorithm which exploits tracking data error estimates in a provably correct way and offers a quality guarantee for the computed result trajectory. A new model for the map-matching task is introduced which takes tracking error estimates into account. The proposed Adaptive Clipping *algorithm (i) provably solves this map-matching task and (ii) utilizes the weak Fréchet distance to measure similarity between curves. The algorithm uses the error estimates in the trajectory data to reduce the search space (*error-aware pruning*), while offering the quality guarantee of finding a curve which minimizes the weak Fréchet distance to the vehicle trajectory among all possible curves in the road network. Moreover, this work introduces an output-sensitive variant of an existing weak Fréchet map-matching algorithm, which is also employed in the Adaptive Clipping algorithm. Output-sensitiveness paired with error-aware pruning makes Adaptive Clipping the first map-matching algorithm that provably solves a well-defined map-matching task. An experimental evaluation establishes further that Adaptive Clipping is also in a practical setting a fast algorithm that at the same time produces high-quality matching results.*

1 Introduction

With the availability of cheap positioning technology and the penetration of asset tracking applications such as fleet management applications, vehicle tracking data, as a component of floating car data (FCD), becomes an important tool for traffic assessment and prediction. Being a by-product of another application has serious implications for the data quality, e.g., infrequent position samples. Still being able to utilize such data for traffic assessment affords *sophisticated map-matching algorithms* to accomplish the task of matching inaccurate tracking data to a road network. Of critical importance for traffic assessment is the *amount* of available data and its *timeliness*. To provide real-time map-matching, *fast map-matching algorithms* are needed.

This work exploits tracking data error estimates to prune the road network graph in a provably correct manner yielding a fast algorithm and still guaranteeing highly accurate results. A new model for the map-matching task is introduced which takes tracking error estimates into account. The proposed *Adaptive Clipping* algorithm (i) provably solves this map-matching task and (ii) finds a path in the road network that resembles the trajectory of the tracking data based on the weak Fréchet distance measure between the curves [2]. Previous algorithms use the (strong) Fréchet distance [1] or the weak Fréchet distance [3] for the map-matching task, and produce accurate matching results. However, these algorithms are comparatively slow. In fact, due to their quadratic runtimes and storage needs, these algorithms do not scale well at all, which makes them unsuitable for larger road maps.

Moreover, this work introduces an *output-sensitive* variant of the weak Fréchet map-matching algorithm of [3], which is also employed in the Adaptive Clipping algorithm. This output-sensitiveness together with the error-aware pruning yields the Adaptive Clipping algorithm which is the first map-matching algorithm that provably solves a well-defined map-matching task. The competitive quality and running time of Adaptive Clipping is established in an experimental evaluation comparing the algorithm to the existing Incremental algorithm [3].

The outline of this paper is as follows. Section 2 discusses tracking data and its use for the derivation of dynamic

weights, related work with respect to map-matching algorithms, and the Fréchet distance and its use in map-matching algorithms. Section 3 introduces the output-sensitive map-matching algorithm. Section 4 details metadata in the form of tracking data errors, defines a data model incorporating this metadata, and introduces the Adaptive Clipping algorithm as an implementation of the error-aware map-matching task. Section 5 shows the outcome of the experimental evaluation and, finally, Section 6 gives conclusions and directions for future research.

2 Background and Related Work

The outset for this work is to create a map-matching algorithms that allows us to utilize a readily available and untapped data source, floating car data, for means of traffic assessment. This section introduces the data, discusses available map-matching algorithms and gives the necessary background on Fréchet distances as they are used in the following sections.

2.1 Data

For several years, a new technology has been discovered to overcome the issue of using costly stationary sensor networks (loop detectors) for traffic assessment. Floating car data (FCD) is already a well known technology for various ITS (intelligent transportation systems) application and has become an interesting complement to conventional traffic sensors. Floating car data (FCD) refers to using data generated by one vehicle as a sample to assess to overall traffic conditions ("cork swimming in the river"). The most important data component is *the position of the vehicle*. Having large amounts of vehicles collecting such data for a given spatial area such as a city (e.g., taxis, public transport, utility vehicles, private vehicles) will create an accurate picture of the traffic condition in time and space [11]. In all these FCD collection scenarios, this data is produced as a by-product of another application (e.g., fleet management). Being a by-product has serious implications for the data quality, e.g., infrequent position samples. To still be able to use this data, sophisticated map-matching algorithms are needed to provide *accurate matches* of the tracking data to a path in the road network.

Besides traffic assessment, FCD can be used to create a reliable travel time database for a road network used in navigation systems. Per default, only static travel-times derived from road categories and speed limits are used to calculate the fastest or shortest path for a given trip. Dynamic travel times derived from FCD aim at supplying on-the-fly computed speed types. The outdatedness of travel times is of critical importance here. To utilize FCD best, travel times have to be computed on-the-fly as *FCD is collected in real time*.

Based on the above requirements, *fast* and *accurate map-matching algorithms* are needed to utilize vehicle tracking data for traffic assessment and related applications.

2.2 Available Map-Matching Algorithms

Algorithms for map-matching vehicle-tracking data comprise the categories of *incremental algorithms* and *global algorithms*.

Incremental algorithms follow a greedy strategy of sequentially extending a solution from an already matched edge. Greenfeld [8] introduces a map-matching strategy based on distance and orientation that does not assert any further knowledge about the movement besides the position samples. Civilis et al. [5][6] in their work on location update techniques for the tracking of users in location-based services introduce a map-matching algorithm that is based on edge distance and direction similar to [8]. The tracking data itself is obtained by using an active sampling technique based on predicted and measured positions. By controlling the sampling rate, the sampling error can be kept minimal and the map-matching algorithm is presented with an optimal dataset. An approach that augments GPS positioning with other methods such as dead reckoning to reduce the measurement error and to achieve better map-matching results is advocated in [10]. Finally, the Incremental algorithm presented in [3], although based on locally matching geometries additionally uses a constant-depth recursive look-ahead to evaluate path alternatives. This algorithm will be used as a benchmark in Section 5. Incremental algorithms are usually very fast, however due to their heuristic nature they restrict the trajectories in the road network that are considered as a valid match.

Global algorithms consider all possible trajectories in the road network to find amongst them the trajectory that is most similar to the vehicle trajectory. Yin and Wolfson [12] propose an algorithm based on a weighted graph representation of the road network in which the weights of each edge represent the distance of the edge to the trajectory. The matched trajectory in the road network is found by using a Dijkstra shortest-path algorithm for the weighted graph. This algorithm is based on a measure related to the average Fréchet distance, however no overall quality guarantee on the matched curve is given. The authors claim that the algorithm produces high quality matches, however details on the data set, such as type and size are missing. Cao and Wolfson [4] propose an algorithm to find a path in the graph whose Hausdorff distance to the vehicle trajectory is at most a given parameter ε. Although using the Hausdorff distance, a function not well-suited for curves [2], Cao and Wolfson's algorithm seems to be related to the computation

of the weak Fréchet distance, but this fact is not mentioned, and their runtime is a linear factor higher than the algorithms of [1, 3], and no experimental evaluation is given. Alt et al. [1] and Brakatsoulas et al. [3] utilize the strong and the weak Fréchet distance measures to find a path in the road network that closely resembles the trajectory. Although at first sight considering all possible paths seems inefficient, the existing algorithms run in polynomial time. Moreover, the algorithms of [1, 3, 4] are based on a notion of similarity between two curves, which has the advantage that the computed result trajectory comes with a quality guarantee, since it is a curve which minimizes the distance to the vehicle trajectory among all possible curves in the road network.

In terms of quality and speed, global algorithms generally produce high-quality matching results but are slow compared to incremental algorithm. Thus, as we shall see in Section 4, the objective in this work is to *improve the speed* of a global algorithm by *preserving its accuracy*.

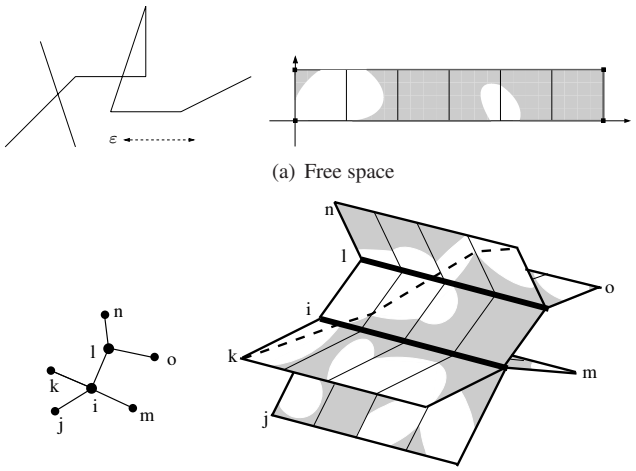
2.3 Fréchet Distances

The algorithm we present in Section 4 is based on the global map-matching algorithms of [1, 3], which employ the Fréchet distance measure for curves. This section will give relevant definitions and results that will be needed in the remainder of this paper. For more details we refer the reader to [1, 2, 3].

The (strong) Fréchet distance for two curves has been proposed by Fréchet [7]. A popular illustration of the Fréchet distance is the following: Suppose a person is walking his dog, the person is walking on the one curve and the dog on the other. Both are allowed to control their speed but they are not allowed to go backwards. Then the *(strong) Fréchet distance* of the curves is the minimal length of a leash that is necessary for both to walk the curves from beginning to end. If both are allowed to go backwards then one obtains the *weak Fréchet distance*

Most algorithms that compute the (weak or strong) Fréchet distance between two curves or which employ those distances in global map-matching algorithms first solve their decision variant: For a fixed $\varepsilon > 0$ decide whether the distance is at most ε or not. Afterwards the minimization problem is solved by applying parametric search or binary search. The algorithms solving these decision problems for a fixed $\varepsilon > 0$ are all based on the notions of the *free space diagram* or the *free space surface* [1].

The free space diagram of a line segment (edge in the road network graph) and a curve (trajectory) encodes which pair of points, one on the line segment and the other on the curve, are at distance at most ε. The axes of a coordinate system are identified with the parameterizations of the curve and the line segment. A white point in the free space diagram encodes a pair of points at distance at most ε, and a black point a distance greater than ε. See Figure 1(a) for an example.



(a) Free space

(b) Free space surface

Figure 1. Free spaces: (a) A free space diagram. (b) A road network (left) and corresponding free space surface (right).

Both the free space diagram of two polygonal curves as well as the *free space surface* of the road network and the trajectory are composed of these segment-curve free space diagrams; for every two incident line segments (in the road network, or in another curve) their individual free space diagrams with the curve are glued together according to the incidence information. Figure 1(b) gives an example road network (left) and its corresponding free space surface for a vehicle trajectory consisting of five position samples (right). The vehicle trajectory is not shown explicitly but implicitly by the white free space area. An example path in the free space from lower left to upper right is drawn dashed.

Deciding whether the weak or strong Fréchet distances are at most ε amounts to finding a path in the white free space area from a lower left corner to an upper right corner. For the strong Fréchet distance the path has to be monotone, for the weak Fréchet distance it can be any path.

3 Output-Sensitive Map-Matching

The concept of output-sensitiveness presents an improvement of existing Fréchet-based map-matching algorithms in that it simplifies existing algorithms and greatly improves average case running times.

3.1 Weak Fréchet Distance and Free Space Graph Optimizations

The traditional approach for solving the decision problem for the weak Fréchet distance (for curves [2] or the map-matching variant [1, 3]) constructs the free space diagram or surface in $\Theta(mn)$ time and space and then runs

a graph traversal algorithm on the free space for a total of $\Theta(mn)$ time. Here m and n are the complexities of the two curves (or the curve and the road map). In this approach the free space is implicitly interpreted as a *free space graph* whose vertices are the white intervals on the boundary of a free space cell, and all intervals incident to the same cell are connected by edges. See Figure 2 illustrations of the free space graph of one free space cell (Left) and of the free space graph of the free space surface of Figure 1(b) (Right).

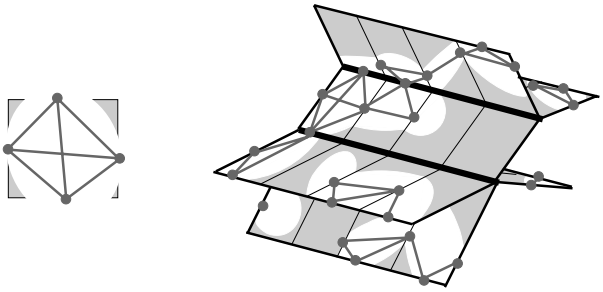


Figure 2. The free space graph of one free space cell (left) and of the free space surface of Figure 1(b) (right).

This *free space graph* encodes all connectivity information of the free space. We modify this graph representation in order to solve the optimization problem directly, without the decision problem: Every pair (e, v) with an edge e on one curve (or on the roadmap) and a vertex v on the other curve (or on the roadmap) defines a possible free space interval, which in turn defines a vertex in the free space graph. For each such pair (e, v) we compute the smallest ϵ for which the free space interval exists, which by definition is the smallest distance from v to e. We weigh the corresponding free space graph vertex with that distance. Let us define the *weight of a path* in the free space graph to be the maximum of the weights of the free space graph vertices it visits. Then a path from the start to the end with minimum weight ϵ^* corresponds to a path in the free space diagram for ϵ^*, and any $\epsilon < \epsilon^*$ for which there is a path in the free space diagram would contradict the definition of ϵ^*. Hence ϵ^* is the optimal ϵ, or, in other words, the weak Fréchet distance. We can compute such a *shortest path* using Dijkstra's shortest-path algorithm in $O(mn \log(mn))$ time and $\Theta(mn)$ space.

The beauty of this algorithm is that it is the same for the computation of the weak Fréchet distance for two curves and for the map-matching task using the weak Fréchet distance, only that in the latter case the underlying free space graph is a bit more complicated.

3.2 Output-Sensitiveness

All previous algorithms for the computation of the strong or the weak Fréchet distance [2] or for the strong or weak Fréchet distance based map-matching algorithms [1, 3] construct and store the whole free space of size $\Theta(mn)$ in advance.

The advantage of the shortest path algorithm on the free space graph as described in Section 3.1 is that it can be implemented to explore and construct necessary portions of the free space graph on the fly during the traversal. We use hash tables to keep track of previously visited free space graph vertices. Assuming that a hash table operation needs $O(1)$ time, this yields an *output-sensitive* algorithm that runs in $O(K \log K)$ time, where K is the size of the *traversed free space*. The *traversed free space* is that part of the free space (or free space graph) that has to be traversed in order to find a shortest path from any start free space graph vertex (e, p_0) to any end free space graph vertex (f, p_n), where e, f are road network edges and p_1 is the first and p_n the last vertex of the GPS curve. Clearly $K = O(mn)$, but since the algorithms stop as soon as a shortest path to the first end vertex has been found, K might be much smaller than $O(mn)$.

Output-sensitive algorithms have the potential to be much more efficient especially for large road maps.

4 Localizing Global Map-Matching Strategies

Additional metadata information related to the tracking data, such as tracking data error, can be exploited to refine the modeling of the map-matching task. In this section, we introduce the *error-aware map-matching task* and the Adaptive Clipping algorithm. The algorithm is a way to localize the global weak Fréchet map-matching algorithm. At the same time it is the first map-matching algorithm which provably solves this well-defined map-matching task.

4.1 Error-Aware Map-Matching

The tracking data is obtained by sampling positions, typically using GPS, to produce data that in database terms is commonly referred to as trajectories. Unfortunately, this data is not precise due to the *measurement error* caused by the limited GPS accuracy, and the *sampling error* caused by the sampling rate [9].

Although the GPS error can be substantial in certain situations (shadowed and reflected signals), with the use of GPS signal augmentation (WAAS, EGNOS) a typical worst-case measurement error is in the range of $10m$ [1]. Sampling

[1] Represents an average value and has to be substituted with worst-case estimates depending on specific measurement scenarios.

movement introduces an error that is directly related to the frequency with which position samples are taken (*sampling rate*). The authors in [9] show that the sampling error is bound by an error ellipse derived from the actual and the maximum traveled distance. Considering a typical sampling rate of $30s$ and a maximum speed of $50km/h$, the traveled distance between position samples can easily reach an error in the range of $150m$ (radius) (cf. [3]). Figure 3 visualizes this error example (worst-case estimate).

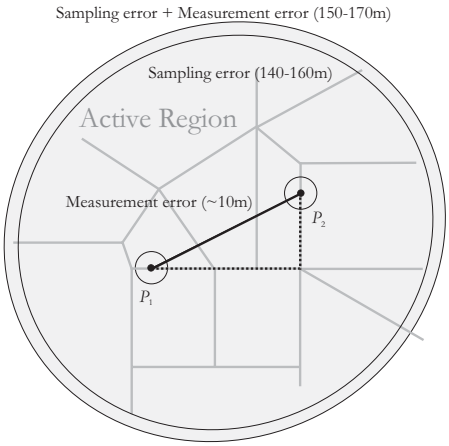


Figure 3. Measurement error, sampling error, and active region.

Usually the vehicle trajectory is modeled as a polygonal curve consisting of a sequence of vertices that are connected by straight-line edges, with the vertices $p_1, \ldots, p_n$ being the sampled GPS positions. Given the two error measures, the positions of the vertices are only accurate up to the measurement error, and the straight-line edges are only accurate up to the sampling error. A more appropriate error-aware representation of the vehicle trajectory is to represent every GPS position with a disk that reflects the measurement error and every two consecutive GPS positions with an error-ellipse that reflects the sampling error (cf. Figure 3). This is a "fuzzy" representation of the trajectory, which describes a region in the plane that contains all possible trajectories where the vehicle could have been, and therefore conveys the true content of the data, as opposed to the polygonal-curve representation, which conveys a false sense of preciseness.

For any two consecutive positions p_i, p_{i+1} let us call $\overline{p_i p_{i+1}}$ an *edge* of the vehicle trajectory. Let the velocity v, the sampling rate r, and the measurement error μ be given. For every p_i we define its *active region* $A(p_i)$ as the disk centered at p_i with radius μ. For every edge $\overline{p_i p_{i+1}}$ we define its *active region* $A(\overline{p_i p_{i+1}})$ as the Minkowski sum of the error ellipse (defined by e, v, and r) with the disk of radius μ (centered at the origin). [2] Intuitively $A(\overline{p_i p_{i+1}})$ is a "thickened" ellipse whose boundary has been thickened by a disk of radius μ (gray-shaded area in Figure 3). $A(p_i)$ contains all positions in the plane which by the measurement error could correspond to p_i, and $A(\overline{p_i p_{i+1}})$ is a combination of measurement and sampling errors and contains all positions in the plane which could correspond to any point in between p_i and p_{i+1}.

The sequence of all active regions $A(p_1), A(\overline{p_1 p_2}), A(p_2), \ldots, A(p_{n-1}), A(\overline{p_{n-1} p_n}), A(p_n)$ is an *error-aware* representation of the vehicle trajectory. See Figure 4 for an example of active regions. Notice that metadata such as known speed on trajectory edges could be used to refine the active regions.

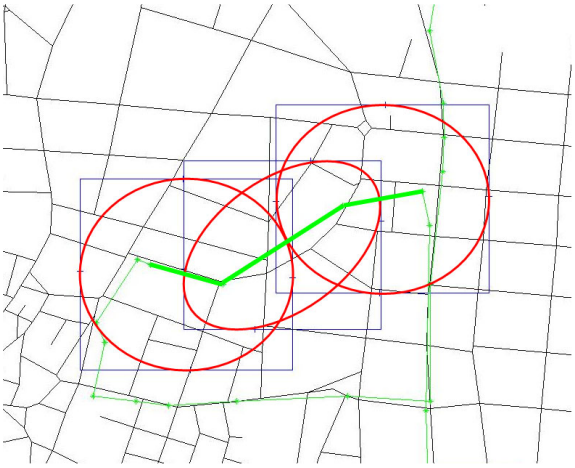


Figure 4. An excerpt of the road network, the trajectory, and three active regions with their bounding boxes.

The following Lemma shows which properties those trajectories in the road network have that could have led to the observed vehicle trajectory.

Lemma 1. *The true trajectory of vertices $q_1, \ldots, q_l$ in the road network that has lead to the vehicle trajectory $p_1, \ldots, p_n$ has the following properties:*

(i) **Start condition:** $q_1 \in A(p_1)$

(ii) **End condition:** $q_l \in A(p_n)$

(iii) **Intersection property:** *The polygonal curve $q_1, \ldots, q_l$ intersects $A(p_i)$ for every $1 \leq i \leq n$.*

[2] The Minkowski sum (also called vector sum) of two point sets $C, D \in \mathbb{R}^2$ is defined as $C \oplus D = \{c + d \mid c \in C, d \in D\}$.

(iv) **Containment property:** *If $q_i \in A(p_j)$ and $q_{i+k} \in A(p_{j+1})$ for $k \geq 0$ then the polygonal curve $q_i, \ldots, q_{i+k}$ has to lie in $A(\overline{p_j p_{j+1}})$.*

Proof. (iii) follows from the measurement error, since every p_i has to have a corresponding point on the polygonal curve $q_1, \ldots, q_l$. (i) and (ii) follow from (iii) and the fact that the curves have corresponding start points and corresponding end points. A sampling ellipse with foci p_j and p_{j+1} contains all positions where the vehicle could have been if it started in p_j and ended in p_{j+1}. Including the measurement error by considering a startpoint $q_i \in A(p_j)$ and an endpoint $q_{i+k} \in A(p_{j+1})$ yields (iv). □

We define the *error-aware map-matching task* as follows:
Given: A road network G and an error-aware representation $A(p_1), A(\overline{p_1 p_2}), A(p_2), \ldots, A(p_{n-1}), A(\overline{p_{n-1} p_n}), A(p_n)$ of the vehicle trajectory.
Task: Find a curve $q_1, \ldots, q_l$ in G that fulfills the properties of Lemma 1.

This formulation of the map-matching task models all known information about the map-matching problem. In the following Section we describe the *Adaptive Clipping* algorithm which solves the error-aware map-matching task.

4.2 Adaptive Clipping Algorithm

Our new *Adaptive Clipping* algorithm that we present in this Section is a way to localize the global weak Fréchet map-matching algorithm. In Theorem 2 we show that in addition it provably solves the error-aware map-matching task as defined in Section 4.1.

The *Adaptive Clipping* algorithm follows an incremental clipping approach: Our output-sensitive weak Fréchet algorithm from Section 3 computes the free space graph, identifies start and end vertices, and then runs Dijkstra's algorithm to find a shortest path from a start to an end vertex. We modify this algorithm to run in stages, each stage corresponding to one trajectory edge. Let the trajectory be $p_1, \ldots, p_n$, and denote by $\overline{p_i p_{i+1}}$ the trajectory edge between p_i and p_{i+1}.

Stage 1.
Dijkstra's algorithm is seeded with the start free space graph vertices (p_1, e), where e is any road network edge (or a part of it) in $A(p_1)$. Then, Dijkstra's algorithm is executed on the part of the free space graph induced by the free space graph vertices (p_1, e), $(v, \overline{p_1 p_2})$, (p_2, e'), where v is any vertex and e any edge in $A(\overline{p_1 p_2})$, and e' is any edge in $A(p_2)$. This computes shortest path values for shortest paths from a start free space graph vertex (p_1, e) to any vertex (p_2, e').

Stage i for $2 \leq i \leq n-1$.
Dijkstra's algorithm is seeded with the shortest path values that have been computed in stage $(i-1)$ for all free space graph vertices (p_i, e'). Then, Dijkstra's algorithm is executed on the part of the free space graph induced by the free space graph vertices (p_i, e), $(v, \overline{p_i p_{i+1}})$, (p_{i+1}, e'), where v is any vertex and e any edge in $A(\overline{p_i p_{i+1}})$, and e' is any edge in $A(p_{i+1})$. This computes shortest path values for shortest paths from a start free space graph vertex (p_1, e) to any vertex (p_{i+1}, e').

Traceback.
All stages store predecessor trees. In the end one shortest path is constructed by tracing back through the predecessor trees of all stages.
See Figure 4 for an illustration of the active regions in the road network.

Clearly, this algorithm computes a shortest path in a restricted free space graph. We claim that it also solves the error-aware map-matching task.

Theorem 2. *The Adaptive Clipping algorithm solves the error-aware map-matching task as defined in Section 4.1. Moreover, amongst all curves in the road network that fulfill the properties of Lemma 1, the Adaptive Clipping algorithm finds a curve with minimum weak Fréchet distance.*

Proof. Let $q_1, \ldots, q_l$ be the curve in the road network that has been computed by the Adaptive Clipping algorithm. First we will show that this curve fulfills all conditions of Lemma 1:

The start vertices of the algorithm are (p_1, e), where e is any road network edge (or a part of it) in $A(p_1)$. Therefore $q_1 \in A(p_1)$, which is condition (i). In stage $n-1$ shortest paths to an end vertex (p_n, e') are computed with e' in $A(p_n)$, which is condition (ii). Condition (iii) follows immediately from noticing that the algorithm visits one vertex (p_i, e) for $e \in A(p_i)$, for every i.

Now assume that $q_i \in A(p_j)$ and $q_{i+k} \in A(p_{j+1})$ for $k \geq 0$. This means that there has to be an edge e in $A(p_j)$ that is incident to q_i such that the algorithm visits the vertex (p_j, e). Similarly, there has to be an edge e' in $A(p_{j+1})$ that is incident to q_{i+k} such that the algorithm visits the vertex (p_{j+1}, e'). In order to reach (p_{j+1}, e') from (p_j, e) the algorithm has to be in stage j in which the only other allowed vertices are $(v, \overline{p_j p_{j+1}})$, (p_{j+1}, e'), where v is any vertex and e any edge in $A(\overline{p_j p_{j+1}})$. This proves (iv), and finishes the proof that the algorithm solves the error-aware map-matching task.

The last claim follows from noticing that the algorithm works just like the weak Fréchet algorithm, only on a restricted free space graph. □

The output-sensitive running time of this algorithm is $O(\sum_{i=1}^{n-1} M_i \log M_i + n)$, where M_i is the number of edges and vertices of the road network in the active region $A(\overline{p_i p_{i+1}})$ whose corresponding freespace graph ver-

tices have been traversed during the algorithm. For $M = \sum_{i=1}^{n-1} M_i$, which is the size of the traversed free space, the runtime can be expressed as $O(M \log M) + n$.

Let m be the total number of edges and vertices in the road network. In general, $M_i \in O(m)$ for each i, which yields $O(mn \log m)$ total runtime. This is, however, a very rough upper bound. In our application the data suggests the following two special cases: (1) It seems that most of the times active regions $A(\overline{pq})$ and $A(\overline{p'q'})$ only intersect if they are adjacent, i.e., if $q = p'$. In this case $\sum_{i=1}^{n-1} M_i = O(m)$ and the total runtime simplifies to $O(m \log m + n)$. (2) In our application it also seems that the parts of the road network in each error ellipse are very small, almost of constant complexity. If each M_i is indeed a constant then the runtime simplifies to $O(n \log n)$.

5 Experimental Evaluation

The objective of this section is to contrast the performance of Adaptive Clipping with the Incremental algorithm [3] in terms of (i) the quality of the map-matching result, (ii) running time, and (iii) database IO operations.

5.1 Setup

The tracking data used in the experiments was obtained by sampling vehicle movements at a rate of 30 seconds. The dataset consists of 27 vehicle trajectories consisting of a total of 15727 position samples. The smallest and the largest trajectory consist of 207 and 1003 edges, respectively [3]. The tracking data was collected in the municipal area of Athens (40x40km) through the years 2000 to 2003. The road network consists of 108000 vertices and 150000 edges.

The implementation platform for the various map-matching algorithms is Java 1.5 in connection with an Oracle 9i database. The algorithms interface with a database using standardized embedded SQL statements making the map-matching algorithms comparable in terms of I/O operations. The road network graph was stored by means of tiles. Thus, only portions of the road network are kept in main memory. Should the map-matching algorithm require an additional network portion, the respective tile is fetched from the database. A LRU buffer scheme is employed to cache road network in main memory. The parameters with respect to the Oracle DBMS are a block size of 6KB and the database cache size of 192MB. Using the tiling approach, the road network graph was partitioned into $16 \times 16 = 256$ tiles and with a LRU buffer size of 50 tiles. The tiling parameter was established during a brief empirical study. However, further research into identifying the optimal parameter setting for each algorithm is necessary. The trajectory data is available by means of text files. For a map-matching task, an entire trajectory is kept in main-memory.

[3]The vehicle tracking data was supplied by Emphasis Telematics, a co-operating telematics company and fleet management service provider.

The Adaptive Clipping algorithm assumes a maximum speed of $v = 80km/h$ and a measurement error of $8m$. We approximate $A(e)$ by its axis-aligned bounding box, and $A(p_i)$ by the intersection of the bounding boxes of $A(\overline{p_{i-1}p_i})$ and $A(\overline{p_ip_{i+1}})$.

Although a Java implementation of the Global algorithm [3] exists, its running time for the dataset used in the following experiments was very long (several hours for one trajectory) and did not permit us to perform a respective evaluation. However, a brief study involving a smaller network showed that its map-matching accuracy (cf. Section 5.2) was in almost all cases equal to that of Adaptive Clipping.

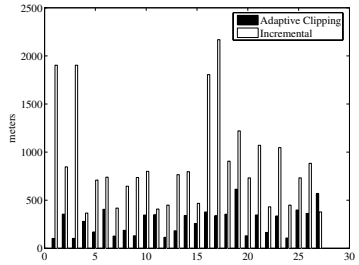
In all charts that follow below, the sorting order of the trajectories is according to the running time of the Adaptive Clipping algorithm for the respective matching result (cf. Figure 7(a)).

5.2 Accuracy

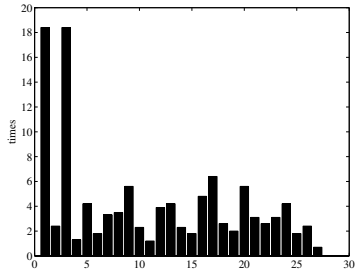
To compare the two algorithms, map-matching results for the tracking data were evaluated using the (i) weak Fréchet distance, (ii) the strong Fréchet distance (c.f. Section 2.3) and (iii) the *average Fréchet distance* with sampling distance $2m$ (cf. [3]). The average Fréchet distance computes an average of the distance between matched points (in contrast to the weak or strong Fréchet distances which compute the maximum).

All 27 data sets contain noise in the sampling rate (i.e., two consecutive samples are significantly more than 30 seconds apart), and the road network apparently lacks roads that actually do exist and the vehicle traversed. We cope with this noise by chopping noise regions off the data sets (online, during the execution of the map-matching algorithm), which results in each data set now being a set of sub-trajectories. Overall, the 27 trajectories comprise 399 sub-trajectories, the smallest number of sub-trajectories was 4, the largest 23, and the average 15. Notice that we still match all the trajectory positions, we just do not consider certain trajectory edges to be part of the data set. We generalize the weak and strong Fréchet distances to this setting by taking the maximum of the distances for each sub-trajectory. The average Fréchet distance takes the average of all individual distances.

The Adaptive Clipping algorithm computes along with the result curve the weak Fréchet distance based on the *restricted* free space graph. This distance is, since it considers only a subset of reparametrizations, greater or equal to the weak Fréchet distance between the trajectory and the result curve in the road network. Interestingly, out of the 399 sub-

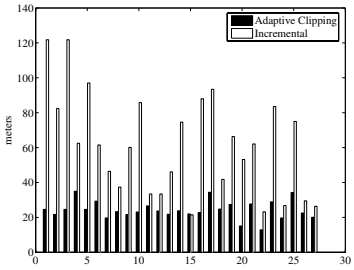


(a) absolute

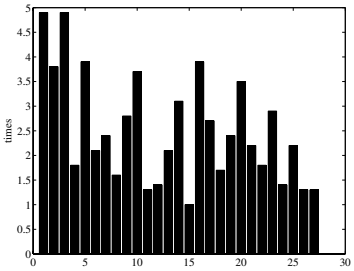


(b) relative

Figure 5. Weak Fréchet distance quality measure



(a) absolute



(b) relative

Figure 6. Average Fréchet distance quality measure

trajectories only 3 have a smaller weak Fréchet distance. Also, in only one case the Fréchet distance is greater than the weak Fréchet distance. Hence, the tracking data exhibits special properties in which all three distances happen to almost always coincide. For this reason, we consider in the following experiments only the weak Fréchet distance and the average Fréchet distance to determine the quality of the respective matching results.

Figures 5 and 6 determine the quality of the matching result by means of the weak Fréchet and the average Fréchet distance, respectively. Figures 5(a) and 6(a) give the absolute distances, whereas the relative quality is shown in Figures 5(b) and 6(b). For the relative distance, the Adaptive Clipping algorithm is the benchmark. What can be readily observed is that the Adaptive Clipping algorithm produces matching results of superior quality. The measured weak Fréchet distance for the Incremental algorithm is an average of 4 times larger (worst case 18 times). Using the average Fréchet distance, it is still on average 1.5 times larger (worst case 5 times). The absolute distances in Figures 5(a) and 6(a) indicate that the trajectory and the matched curve exhibit (i) a worst case distance (weak Fréchet) of up to $2000m$ and $600m$ and (ii) an average distance (average Fréchet) of up to $120m$ and $35m$ for the Incremental and the Adaptive Clipping algorithm, respectively. This rather large distances for the Incremental algorithm are the result of a poor match for the intital trajectory points (cf. also [3]).

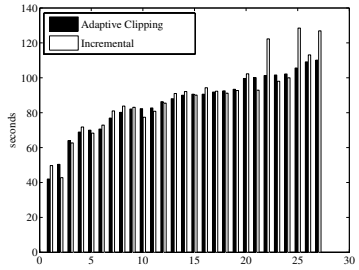
5.3 Speed

The speed of computation is measured in terms of (i) the algorithmic running time and (ii) the number of database IO operations.

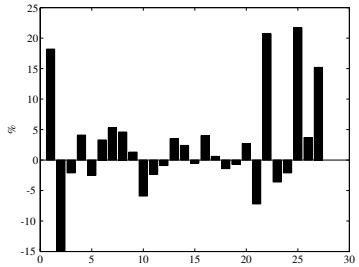
Figures 7 and 8 give the overall running times and time attributed to database access, respectively. Again, absolute and relative times (percentages) are shown, with the Adaptive Clipping algorithm serving as a benchmark for the Incremental algorithm.

An important objective for this work was to use it for real-time map-matching of tracking data. Figure 7(a) shows that matching the smallest trajectory, which consists of 207 position samples takes $40s$ ($50s$). With a sampling rate of $30s$, this data was collected over a period of $6210s$ (1h 40min). Thus, the sampling rate could be as low as $0.2s$ ($0.25s$) (for 207 samples, the collection period would then be equal to the running times of the map-matching algorithms) for the Adaptive Clipping (Incremental algorithm) still to be able to keep up with the incoming tracking data stream.

In comparing the two algorithms, surprisingly in almost all cases, Adaptive Clipping runs faster than the Incremental algorithm (up to 20%). According to the asymptotic running times of the algorithms, Adaptive Clipping - $O(n \log n)$ and Incremental algorithm - $O(n)$ [3], with n being the number of trajectory edges, the opposite was expected. However, the $O(n)$ time of the Incremental algorithm absorbs the lo-

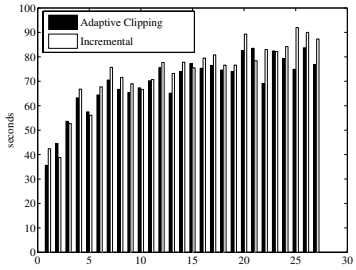


(a) absolute

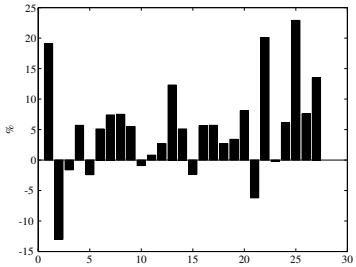


(b) relative

Figure 7. Total running times of the map-matching algorithms



(a) absolute



(b) relative

Figure 8. Running times attributed to database access

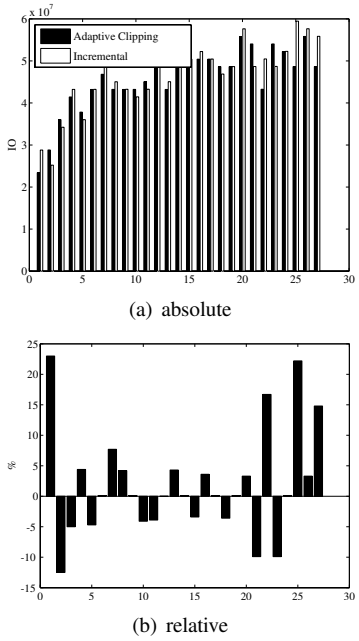
cal look-ahead cost (constant factor), which in practice significantly affects the running time.

A large portion of the running time (75% to 85%) is attributed to database access. Hence, this time largely determines the overall running time. However, interesting deviations (cf. trajectories 10 to 15) exist. When Adaptive Clipping is slower or equal in terms of database access time, the overall relative running time of Adaptive Clipping is even worse, i.e., the algorithm draws its performance advantage largely from a more intelligent choice of which portion of the road network to load from the database. The number of logical IO operations is shown in Figure 9. Logical IO is the sum of the number of buffers read from disk and from the memory cache. Using a buffer cache size of 192MB the number of disk accesses was kept low. To force disk accesses, we conducted an experiment with a smaller cache size (1MB). The charts were similar to what is presented in 9 and, most importantly, did not present different performance characteristics relative between the trajectories and in relation the running times of Figure 8. The number of IO operations show a more balanced picture between the two algorithms but still confirm the performance advantage of Adaptive Clipping. Comparing Figures 9(b) and 8(b) shows that the running time attributed to database access is not solely determined by the number of IO operations but also by additional factors related to query execution.

Overall, the experiments show that Adaptive Clipping with respect to the Incremental algorithm (i) produces better matching results by (ii) in most cases having a lower running time. Both algorithms are further well suited to perform real-time map-matching for tracking data collected at a typical sampling rate of $30s$. The sampling rate could be as low as $0.2s$ to still fulfill the real-time requirement.

6 Conclusions and Future Work

Map-matching algorithms are an enabling technology to the use of vehicle tracking data for traffic assessment and related applications such as routing. To possibly include a large variety of datasets and to guarantee the timeliness of the data, the key objective in this work is to provide a *fast* and *accurate map-matching algorithm*. We present the *Adaptive Clipping* algorithm, which combines the best of both worlds of previously introduced algorithms. It is *fast* in that (i) it intelligently uses tracking data metadata in the form of error estimates to prune the road network graph in a provably correct manner and (ii) employs an output-sensitive optimization for Fréchet-based algorithms. It is *accurate* since it (i) provably solves the error-aware map-matching task, and (ii) offers the quality guarantee of finding a curve which minimizes the weak Fréchet distance to the vehicle trajectory among all possible curves in the road network. The performance study establishes the running time of Adaptive Clipping to be lower than that of the fastest available algorithm, the Incremental algorithm. Its output-sensitive asymptotic running time is



(a) absolute

(b) relative

Figure 9. Number of database IO operations

$O(M \log M + n)$, where $1 \leq M \leq mn$ is the size of the traversed free space. Also, the quality of the matching result is in all cases superior to that of the Incremental algorithm. Overall, Adaptive Clipping is a fast algorithm producing high-quality matching results and thus can be used in a real-time tracking data collection scenario.

The directions for future work are as follows. Having reached a certain level of maturity, Adaptive Clipping should be applied as a map-matching algorithm in a life data collection scenario. Also, given that in such a scenario typically many concurrent data streams exist, we have to evaluate of how well the overall architecture (DBMS and algorithm) scales with concurrent map-matching threads. With respect to algorithmic improvements, we will explore and evaluate the use of spatial range queries to speed up the computation of the active regions. Moreover, we plan on utilizing additional FCD data components as metadata in the algorithm, e.g., use collected speed information for the definition of the active regions. We are currently working on assessing the quality of various map-matching algorithms by using artificially generated trajectories to conduct the ultimate quality test of comparing the result trajectories of the various map-matching algorithms with the actual original curve.

Acknowledgments

This research is supported in part by the the IXNILATHS project funded by the Greek General Secretariat of Research and Technology, and the Faculty Research Award Program at the University of Texas at San Antonio.

References

[1] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *J. of Algorithms*, 49:262–283, 2003.

[2] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[3] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. 31st VLDB Conference*, pages 853–864, 2005.

[4] H. Cao and O. Wolfson. Nonmaterialized motion information in transport networks. In *Proc. 10th ICDT conf.*, pages 173–188, 2005.

[5] A. Civilis, C. S. Jensen, J. Nenortaite, and S. Pakalnis. Efficient tracking of moving objects with precision guarantees. In *Proc MobiQuitous conf.*, pages 164–173, 2004.

[6] A. Civilis, C. S. Jensen, and S. Pakalnis. Techniques for efficient road-network-based tracking of moving objects. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):698–711, 2005.

[7] M. Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del circolo Mathematico di Palermo*, 22:1–74, 1906.

[8] J. Greenfeld. Matching GPS observations to locations on a digital map. In *Proc. 81th Annual Meeting of the Transportation Research Board*, Washington, DC, 2002.

[9] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In *Proc. 6th SSD conf.*, pages 111–132, 1999.

[10] M. Quddus, W. Ochieng, L. Zhao, and R. Noland. A general map matching algorithm for transport telematics applications. *GPS Solutions Journal*, 7(3):157–167, 2003.

[11] R.-P. Schaefer, K.-U. Thiessenhusen, and P. Wagner. A Traffic Information System by Means of Real-time Floating-car Data. In *Proc. ITS World Congress*, Chicago USA, 2002.

[12] H. Yin and O. Wolfson. A weight-based map matching method in moving objects databases. In *Proc. 16th SSDBM conf.*, pages 437–438, 2004.