International Journal of
Geo-Information

MDPI

Article

Uncertainty-Based Map Matching: The Space-Time Prism and *k*-Shortest Path Algorithm

Bart Kuijpers [1,*], Bart Moelans [1,2], Walied Othman [1] and Alejandro Vaisman [3]

1 UHasselt–Hasselt University and transnational University Limburg, Databases and Theoretical Computer Science Research Group, Agoralaan, Diepenbeek 3590, Belgium; walied.othman@gmail.com
2 VikingCo, Hasselt 3500, Belgium; bart.moelans@vikingco.com
3 Instituto Tecnológico de Buenos Aires 1106, Argentina; avaisman@itba.edu.ar
* Correspondence: bart.kuijpers@uhasselt.be; Tel.: +32-476-741-939

Academic Editor: Wolfgang Kainz
Received: 5 September 2016; Accepted: 4 November 2016; Published: 10 November 2016

Abstract: Location-aware devices can be used to record the positions of moving objects for further spatio-temporal data analysis. For instance, we can analyze the routes followed by a person or a group of people, to discover hidden patterns in trajectory data. Typically, the positions of moving objects are registered by GPS devices, and most of the time, the recorded positions do not match the road actually followed by the object carrying the device, due to different sources of errors. Thus, matching the moving object's actual position to a location on a digital map is required. The problem of matching GPS-recorded positions to a road network is called map matching (MM). Although many algorithms have been proposed to solve this problem, few of them consider the uncertainty caused by the absence of information about the moving object's position in-between consecutive recorded locations. In this paper, we study the relationship between map matching and uncertainty, and we propose a novel MM algorithm that uses space-time prisms in combination with weighted k-shortest path algorithms. We applied our algorithm to real-world cases and to computer-generated trajectory samples with a variety of properties. We compare our results against a number of well-known algorithms that we have also implemented and show that it outperforms existing algorithms, allowing us to obtain better matches, with a negligible loss in performance. In addition, we propose a novel accuracy measure that allows a better comparison between different MM algorithms. We applied this novel measure to compare our algorithm against existing algorithms.

Keywords: map matching; trajectory and moving object data; space-time prism; *k*-shortest path algorithm

1. Introduction

GPS devices can be used not only as navigational tools, but also for recording the position of moving objects (MOs) for further spatio-temporal data analysis [1]. For instance, we can analyze the routes followed by a single MO or a group of MOs, to discover hidden patterns in trajectory data. However, GPS-obtained coordinates will not always match the road followed by, for instance, a car or a pedestrian, due to different sources of errors. Besides the typical measurement errors, in real-world data, we can find traffic jams (producing many consecutive points with a very small spatial gap) or large gaps between two consecutive measured points, produced by some interference in the satellite signal (for instance, when moving in tunnels or urban tunnels). Thus, matching the user's actual position with a location on a digital map is required. The general problem of matching GPS-recorded positions to road segments in a road network is called map matching (MM) [2].

Many algorithms have been proposed to solve the MM problem, although none of them considers the uncertainty caused by the absence of information about the MO's position in between to consecutively recorded locations. A common problem in moving object databases (MOD) is the reconstruction of a trajectory from a trajectory sample [3]. Linear interpolation between sample points, which assumes that objects move at constant minimal speed, is a classical solution. A more realistic model is based on the notion of uncertainty. For example, space-time prisms can be used to model the unknown, but possible, positions of an MO between sample points, by using background information, like speed limitations. A space-time prism acts as an envelope that encompasses all possible locations that a MO can have visited between consecutive measured sample points, given a speed limit.

In this paper, we study the relationship between MM and uncertainty. Our main contribution is a novel MM algorithm that uses space-time prisms combined with weighted *k*-shortest path algorithms [4]. This algorithm is applicable to a wide range of trajectory sample types. In short, it first selects parts of the road network by computing, for each pair of consecutive sample points, the road segments that the MO could have moved on (using the bounding boxes of the projections of space-time prisms). Then, for each sample point, the algorithm computes the closest road segment and assigns scores to each road segment. Finally, the algorithm computes, within the limited road network (as determined in the first step), the *k*-shortest paths (taking the shortest path with the highest score computed in the second step).

We applied our algorithm to real-world cases, as well as to computer-generated trajectory samples with a variety of properties. Observations are taken at small regular intervals and at larger and irregular intervals, to capture the classical distinction between low sampling and high sampling rates in trajectory data. We also compared our results against a number of well-known algorithms that we have implemented. Our results show that the incorporation of uncertainty in the MM process leads to obtaining better matchings. This positive outcome largely compensates the longer running times, which remain within reasonable limits, however. It is worth noting that we have produced a great variety of experimental scenarios that show the applicability of our method in a wide range of situations. This is relevant given that, as stated in the survey by Hashemi and Karimi [5], most work in the field does not clearly specify the testing environment, making it difficult to compare proposals. We believe that having implemented ourselves the most classic algorithms and comparing these implementations against our proposal in a wide variety of scenarios help to make our results more reliable.

As a second contribution, we address the problem of accurately comparing the performance of MM algorithms against each other. We review possible properties of trajectory samples and techniques to obtain datasets with these properties. We also discuss a number of existing methods to measure the accuracy of an MM algorithm over these different kinds of datasets. We propose a novel accuracy measure called curve-and-length-accuracy that, combining the strengths of those methods, does not suffer from their drawbacks. We applied this novel measure to compare our algorithm against existing MM algorithms, as described above.

Organization

The remainder of the paper is organized as follows. In Section 2, we present an overview of the main existing MM algorithms. In Section 3, after reviewing the notion of space-time prisms, we present our MM algorithm. Section 4 introduces our novel method to evaluate MM algorithms, which we use, in Section 5, to evaluate our algorithm against existing proposals. We conclude in Section 6.

2. An Overview of Existing Approaches to Map Matching

We mentioned above that, typically, when the position of an MO is monitored using GPS, a large portion of the recorded space-time points fall outside the actually followed road network, due, in part, to measurement errors. However, there are also other problems with real-world data, like traffic jams and large gaps between measured space-time points. These gaps appear due to several reasons, e.g.: (a) a faulty GPS signal; (b) an interruption of the communication; or (c) ambiguous data. Case (a) may occur due to the bad reception of the coordinates, which is quite unlikely to occur. In Case (b), an interruption of the communication between the GPS satellite and the MO's device may be due, e.g., to a densely-forested area, a tunnel or high buildings. Case (c) may appear when roads run parallel, and GPS data are not precise enough to decide which is the correct road. All of these types of problems require mapping the measured space-time points to the road network. The following classical definition or description of the MM problem is due to White, Bernstein and Kornhauser: An object is moving along a finite system (or set) of streets, $\overline{N}$. A location-aware device such as GPS provides an estimate for the vehicle's location at a finite number of points in time, denoted by $\{0, 1, \ldots, t\}$. The vehicle's actual location at time t is denoted by $\overline{P}^t$, and the estimate is denoted P^t. Map matching is the process of determining the street in $\overline{N}$ that contains P^t; that is, to determine the street that the vehicle is on at time t [2]. Many algorithms have been proposed to address this problem. We study some of them next.

2.1. Classification of Map Matching Algorithms

Two orthogonal classification of MM algorithms are usually found in the literature: offline versus online algorithms and low sampling rate versus high sampling rate algorithms. Online MM algorithms are devised to be used in a real-time situation, where the object is still on the move. An example is a GPS route planner in a driving car. In offline MM algorithms, the complete trajectory data of an MO is available, and generally, algorithms for this type of data give better results.

Regarding the second classification above, the rate at which GPS trajectory data are sampled depends on the source of the data and ranges from one sample point per second up to any rate the application requires. Usually, more than one sample point per two minutes is used as the lower limit to be considered high sampling. However, this makes it possible for data to be misclassified. We propose the following, novel way that uses the number of samples per road segment, for classifying data as having a low or high sampling rate: if there is at least one sample per road segment composing a trajectory between two given points, we assume the sampling rate is high. Otherwise, the data have a low sampling rate. For this classification, we calculate a confidence area around each GPS point and limit the road network to the road segments that fall within the confidence area. This is done by drawing a circle around the sample point, with a diameter equal to the possible measurement error. In practice, this is often in the range of twenty-five meters. Figure 1 (left) shows an example of high sample rate trajectory data. The right-hand side of the figure shows an example of low sampling rate trajectory data.

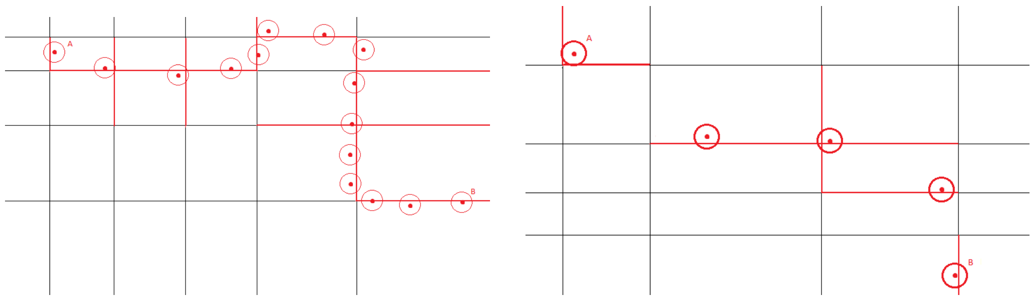

Figure 1. Left: An example of a high sampling rate. The circles around GPS points represent the confidence area. A trajectory from starting Point A to end Point B exists, such that at least one point falls in each segment. **Right**: An example of a low sampling rate. There is no trajectory from Point A to B.

2.2. A Brief Review of Map Matching Algorithms

Another way of classifying MM algorithms defines three kinds of them: (a) geometric algorithms; (b) topological algorithms; and (c) probabilistic algorithms. We also discuss other approaches not falling into these three categories. We only give details concerning the algorithms we use in our experimental setting (see Section 5) to compare against our proposed algorithm.

2.2.1. Geometric Algorithms

These algorithms use the shape of the road segments and ignore how road segments are connected. We can classify them as "point-to-point" and "point-to-curve (with or without topological information)" algorithms. We refer to [6] and [2] for details.

In point-to-point matching, each space-time sample point is simply mapped to the nearest node (or vertex) in the road network.

Point-to-curve matching is similar to point-to-point matching, but space-time points are matched to the closest road segment, instead of the closest node in the road network. There are several problems with these types of algorithms. First, sampling errors may occur in the data (e.g., when two consecutive space-time points are farther apart in time than they should be). Furthermore, the inaccuracy of GPS data may produce a recorded space-time point that is closer to a road, which is not in the actual trajectory. This typically occurs when there are parallel roads close to each other or in the vicinity of crossings. Furthermore, these algorithms only look at geometric information and do not take into account to which road segment the previous point was matched. Point-to-curve matching with topological information uses connectivity information to match points to a road segment [2]. The topology of the road network is used, whenever possible, to determine candidate nodes. Only road segments that are directly reachable from the current segment are considered. However, if the algorithm has low confidence in the previous match, it will switch to the point-to-curve matching algorithm. In [2], confidence is reached when the error is less than 0.15 km and two times the average error.

2.2.2. Topological Algorithms

These algorithms consider the way in which nodes are connected within the road network. Generally speaking, they give a weight to each road segment and find the best path considering these weights. We discuss some of them and refer the reader to the references for the rest. A comprehensive review of MM algorithms can be found in [5].

The hidden Markov model models a process considering many possible states [7]. Applied to MM, a state could be a road segment, and each transition between road segments can be assigned some probability. The state measurements are the measured sample points. Examples of this method can be found in [8] and [9].

Greenfeld's algorithm [10] only uses the coordinate information of the object and does not use any knowledge about the expected traveling route, traveling speed and heading. It first computes an initial match (using the point-to-point algorithm); then, to determine to what road segment a space-time point will be matched, a weight is calculated for several candidate road segments.

Brakatsoulas et al. [11] proposed two kinds of algorithms: incremental and global ones. The former match the position samples sequentially, considering only the connected part of the road network. To improve algorithms like the Greenfield one, mentioned above, the local look-ahead is used, to explore which branch would be the best match, instead of simply looking for the best edge. The global algorithm presented in [11] tries to find a curve (that is, a number of connected road segments) in the road network that is as close as possible to the followed trajectory (that is, the curve formed by a sequence of GPS points). As a measure of distance between two curves, the Fréchet distance or weak Fréchet distance is used in this case. The Fréchet distance is usually explained by the analogy of a person walking her/his dog using a leash. The person is walking on one curve, while the dog is walking on another one. Both may vary in speed or stop, but they are not allowed to walk back. The strong Fréchet distance is the length of the shortest leash length that makes it possible to traverse both curves from start to end. The weak Fréchet distance allows traversing backwards on the curves. The problem with global algorithms is that, due to their nature, they consider the whole network, producing good matches, but at the expense of slow running execution times. Thus, in [12], an adaptive clipping algorithm is proposed, which tries to improve the running time of the global algorithm, by using a worst-case motion estimate to prune the road network. An error estimation is used, based on the maximum speed the object can move at, using an error ellipse.

Other advanced algorithms are based on Kalman filters [13]. For example, in [14], an "extended Kalman filter" is used to linearize a trajectory. In Li et al. [15], a topological MM algorithm is presented, where data from GPS, DRsensors and DEM are first integrated. Candidate segments are connected to the previous one, using proximity and heading as weights to find the correct road segment.

2.2.3. Probabilistic Algorithms

These algorithms use a confidence region around each sample point to compute a matching point on the road network. This confidence region is based on the error variances that space-time points typically have. As a representative example, we discuss the Ochieng, Quddus and Noland algorithm [16]. Like many other algorithms, this algorithm consists of an initial mapping process, which finds an initial match, and a subsequent mapping process. Like in the space-time prisms algorithm discussed later, an area is calculated around each space-time point. The main difference is that this algorithm uses probability concepts to immediately select each road match, whereas the space-time prism algorithm attributes a weight to each segment and then uses *k*-shortest paths to find matches (we explain the *k*-shortest paths algorithm below). Road segments that are in the confidence region are considered the candidate segments. If there is only one such segment, it is selected. If not, link connectivity, heading information, closeness to the segment in question and historical information are used to select one. The algorithm considers that the object remains on the same road segment until a junction is reached or a turning maneuver is detected. Whenever this occurs, the initial matching process is called again, and a new road segment is chosen. The detection of a turning maneuver is based on the speed of the object, the time it takes to perform the turn and the change in heading angle. After matching a space-time point to a road segment, an estimation of where the object is located on the road segment is calculated.

Algorithms based on fuzzy logic [17] were first proposed by Zhao [18]. His method only matches the space-time points to a nearby road network segment. Syed and Cannon improve this approach by using more inputs and matching the space-time point to a location on the road segment [19]. Quddus further improves this algorithm by using even more fuzzy inputs and by also using connectivity information and the historical trajectory of the object [20]. The latter algorithm is very similar to Greenfeld's, but the weight is calculated using fuzzy logic.

2.3. Algorithms for Data with a Low Sampling Rate

Most of the algorithms above require a high sampling rate to produce meaningful results. Since not all data have a high enough sampling rate, algorithms that can handle low sampling rate data are needed. We comment on some of them separately, although they can fit into one of the categories mentioned above.

One of the most representative works in this category is the ST-matching algorithm [21]. This algorithm first generates candidate points and, then, through spatial (using the distance between a space-time point and a road segment) and temporal (using the average speed between consecutive points) analysis, gives weights to each candidate point. Then, the path with the highest weight is selected. Measurement probabilities are also calculated, and instead of modeling noise with a zero-mean Gaussian distribution, a normal distribution is used. The transition probability is calculated as in the hidden Markov model.

Li et al. [22] propose a so-called multi-track MM algorithm (opposite the traditional single-track MM). The rationale behind this method is the observation that in many cases, moving objects move following the same pattern, so they apply MM techniques to a group of trajectories in a sample, and not individually. In other words, they match simultaneously a collection of trajectories to a map. The authors report some experimental preliminary results, with data samples having an interval between one and five minutes, but the characteristics of the network and samples are not completely specified, which makes it difficult to compare against other work. The authors also report important work to be done (e.g., defining an appropriate point-to-curve distance).

Tang et al. [23] present a dynamic programming approach, combined with space-time paths (rather than space-time prisms) to handle uncertainty trajectory data, although we think that the experiments are not conclusive about the method's efficiency, particularly under low-sampling rate conditions.

Chen et al. [24] also proposed a multi-criteria dynamic programming algorithm for MM with low frequency data, using the shortest path procedure to find the candidate road segments between two consecutive GPS points.

Other works worth being mentioned are the ones by Wang et al. [25], Morikawa et al. [26], Rahmani et al. [27], Hunter et al. [28] and Zeng et al. [29]. For the sake of space, we do not further comment on them.

It is worth noting that all of the works cited above consider nominal time intervals to distinguish between a low and a high sampling rate data. Opposite this, in Section 2.1, we provide a more realistic measure for this issue, which takes into account the topology of the network.

3. An Uncertainty-Based Map Matching Algorithm

In the previous section, we gave a broad overview of existing MM algorithms. In this section, we propose a novel algorithm that exploits the uncertainty caused by an MO's unknown location between sampled space-time points, by using background information (such as speed limits). We study the relation between MM and uncertainty and propose an algorithm that combines weighted *k*-shortest paths with space-time prisms.

3.1. Introduction to Space-Time Prisms

Often, in practical applications, more is known about measured trajectories than merely some (for instance, GPS-collected) sample points $(\mathrm{t}_i, \mathrm{x}_i, \mathrm{y}_i)$, $i = 1, 2, ..., N$. For instance, background knowledge, like a physically- or law-imposed speed limit v_i at location $(\mathrm{x}_i, \mathrm{y}_i)$, might be available. Such a speed limit might even be time dependent, e.g., some streets might have different speed limits during the day and the night or during rush hours. The speed limits that hold between two consecutive sample points can be used to model the uncertainty of an MO's location between sample points. For modeling uncertainty, Pfoser and Jensen [30] and, later, Egenhofer et al. [31,32] introduced the notion of beads in the MOD literature. Before, Wolfson used cylinders to model uncertainty [33] (see also [3]). Beads were already conceptually known in the time geography of Hägerstrand in the 1970s under the name space-time prisms [34]. In this paper, we use this more traditional name. In the area of GIS, space-time prisms were also studied by Miller [35].

The space-time prism between two consecutive sample points is defined as the collection of time-space points where the MOs could have passed, given the (local) speed limitation. Now, we make this more formal. Let S be a (measured) trajectory sample $\{(\mathrm{t}_0, \mathrm{x}_0, \mathrm{y}_0), (\mathrm{t}_1, \mathrm{x}_1, \mathrm{y}_1), ..., (\mathrm{t}_N, \mathrm{x}_N, \mathrm{y}_N)\}$, with $t_0 < t_1 < \cdots < t_N$. In the space-time prism model, for each pair $(\mathrm{t}_i, \mathrm{x}_i, \mathrm{y}_i), (\mathrm{t}_{i+1}, \mathrm{x}_{i+1}, \mathrm{y}_{i+1})$, with $0 \leq i < N$, in the sample S, the corresponding space-time prism depends on a maximal velocity value v_i of the MO between those two locations. We know that, given the speed limitation v_i, at a time t, $\mathrm{t}_i \leq t \leq \mathrm{t}_{i+1}$, the object's distance to $(\mathrm{x}_i, \mathrm{y}_i)$ is at most

$\mathsf{v}_i(t - \mathsf{t}_i)$, and its distance to $(\mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ is at most $\mathsf{v}_i(\mathsf{t}_{i+1} - t)$. The spatial location of the object is therefore somewhere in the intersection of the disc with center $(\mathsf{x}_i, \mathsf{y}_i)$ and radius $\mathsf{v}_i(t - \mathsf{t}_i)$ and the disc with center $(\mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ and radius $\mathsf{v}_i(\mathsf{t}_{i+1} - t)$. We can therefore say that the space-time prism consists of all points (t, x, y) in the time-space space that satisfy the following three constraints: $\mathsf{t}_i \leq t \leq \mathsf{t}_{i+1}$; $(x - \mathsf{x}_i)^2 + (y - \mathsf{y}_i)^2 \leq \mathsf{v}_i^2(t - \mathsf{t}_i)^2$ and $(x - \mathsf{x}_{i+1})^2 + (y - \mathsf{y}_{i+1})^2 \leq \mathsf{v}_i^2(t - \mathsf{t}_{i+1})^2$. Geometrically speaking, the last two equations define the intersection of an upward and a downward cone in time-space space, as is illustrated in Figure 2. The chain of space-time prisms connecting consecutive trajectory sample points is called a space-time prism chain. Sometimes, the term bead is used for space-time prism and lifeline necklace for space-time prism chain [31]. For more details on the geometric properties of space-time prisms, we refer to [36].

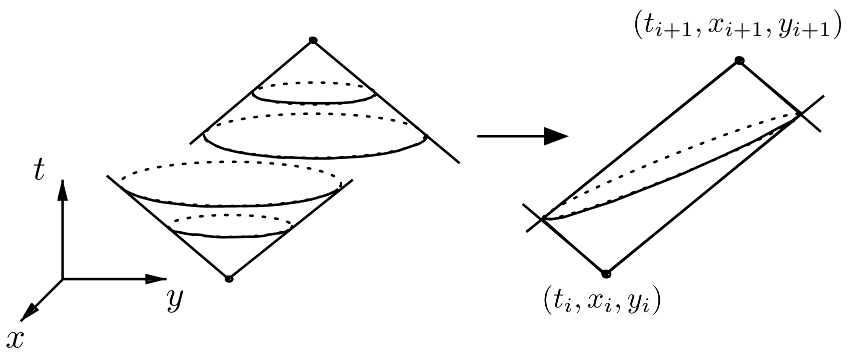

Figure 2. An example of a space-time prism as the intersection of an upward and a downward cone.

3.2. Using Space-Time Prisms for Map Matching

The main contribution of this paper is a novel MM algorithm that uses a combination of techniques for handling uncertainty in trajectory databases. More precisely, we propose to use space-time prisms in combination with weighted k-shortest paths algorithms. To incorporate space-time prisms efficiently into our algorithms, we use some geometrical simplifications. The projection of a space-time prism on the two-dimensional spatial component of time-space space is an ellipse with focal points $(\mathsf{x}_i, \mathsf{y}_i)$ and $(\mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ and semi-major axis $\frac{\mathsf{v}_i(\mathsf{t}_{i+1} - \mathsf{t}_i)}{2}$. However, for efficiency and easy expressibility in SQL, we compute a bounding box of this ellipse (with respect to the x- and y-axes). The following sections describe how this bounding box of the projection of a space-time prism can be determined and how it can be used in practice, respectively.

3.2.1. Computation of the Projection of a Space-Time Prism and Its Bounding Box

We define the bounding box of the ellipse, that is the spatial projection of a space-time prism in the plane, as the smallest rectangle, with sides parallel to the x- and y-axes, that encloses the ellipse. Alternatively, we could say that the bounding box of an ellipse is (the border of) the rectangle that is the Cartesian product of the projection of the ellipse on the x-axis, with the projection of the ellipse on the y-axis. The theorem in this section shows how the bounding box of an ellipse in the plane can be determined, given the focal points (x_1, y_1) and (x_2, y_2) of the ellipse and its semi-major axis $L > 0$. In this theorem, we call the x-values of the left and right vertical sides of the bounding box, X_1 and X_2, and the y-values of the lower and upper horizontal sides of the bounding box, Y_1 and Y_2. Therefore, the bounding box of the ellipse is (the border of) the set $[X_1, X_2] \times [Y_1, Y_2]$. For the proof of this theorem, we refer to Appendix A. Although this theorem is quite elementary, in textbooks, only the case where the axes of the ellipse are parallel to the coordinate axes can usually be found. Figure A1 in Appendix A gives an illustration of an ellipse and its bounding box.

Theorem 1. *Let (x_1, y_1) and (x_2, y_2) be points (which represents the foci) in $\mathbf{R}^2$, and let $L > 0$ be a real number (which represents the semi-major axis). Let (x_c, y_c) be the mid-point of the foci and d_f be the distance between them. If $x_1 \neq x_2$, the bounding box $[X_1, Y_1] \times [X_2, Y_2]$ is given by:*

- $X_1 = x_c - \sqrt{\frac{s^2\ell^2+L^2}{1+s^2}}$, $Y_1 = y_c - \sqrt{\frac{s^2L^2+\ell^2}{1+s^2}}$,
- $X_2 = x_c + \sqrt{\frac{s^2\ell^2+L^2}{1+s^2}}$, $Y_2 = y_c + \sqrt{\frac{s^2L^2+\ell^2}{1+s^2}}$,

where $\ell = \frac{1}{2}\sqrt{4L^2 - d_f^2}$ (the semi-minor axis) and $s = \frac{y_1-y_2}{x_1-x_2}$. If $x_1 = x_2$, then $X_1 = x_c - \ell$, $X_2 = x_c + \ell$, $Y_1 = y_c - L$ and $Y_2 = y_c + L$ (with $L = \ell$ if $y_1 = y_2$). □

3.2.2. Using Bounding Boxes of the Projection of the Space-Time Prisms in Map Matching

Now, we give two examples of how the bounding box of the projection of the space-time prism can help to limit the number of road segments that have to be considered in the MM process. Let us first explain how we model road networks.

Definition 1. A road network RN is a graph embedding in $\mathbf{R}^2$ of a labeled (directed) graph given by a finite set of vertices $\mathsf{V} = \{(x_i, y_i) \in \mathbf{R}^2 \mid i = 1, \ldots, N\}$ and a set of edges $\mathsf{E} \subseteq \mathsf{V} \times \mathsf{V}$ that are labeled with a speed limit. Vertices are embedded in $\mathbf{R}^2$ by the points that have their coordinates, and edges are embedded as straight line segments between the embedded vertices. These edge embeddings may intersect, to model bridges and tunnels. These straight line segments are called road segments. □

The left-hand side of Figure 3 shows the bounding box of a space-time prism projection computed for two points A and B, that are 13 m apart. The (projection of the) space-time prism is extremely large in this case, because the traveling distance from A to B is 35 s (possibly due to a traffic jam). The projection of the space-time prism represents the region where the car could have been when it would travel for 35 s at the maximum speed of the highway (120 km/h). This results in a huge bounding box of the projection of a space-time prism that contains many streets. On the other hand, the right-hand side of Figure 3 shows the projection of a space-time prism for two points A and B, which are 11 m apart, with a travel time of one second. Here, the bounding box of the projection of the space-time prism includes only two road segments. These examples illustrate how the use of space-time prisms to prune the streets or road segments that have to be considered in the MM process can vary. In the second example, the benefit is much larger than in the first example.

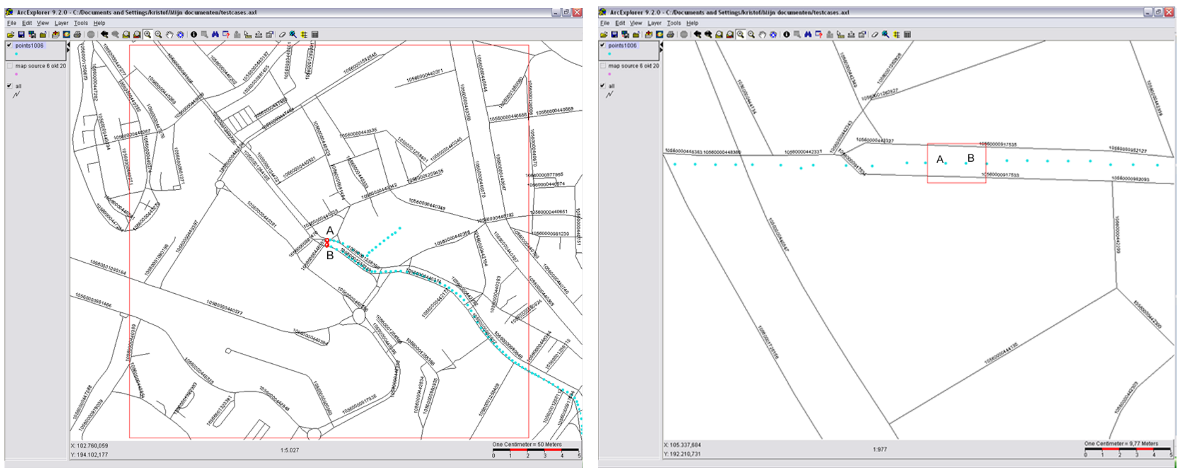

Figure 3. An example of the bounding box for two consecutive sample points with a time gap of 35 s (on the **left**) and for two consecutive points with a time gap of one second (on the **right**).

The use of the bounding box of the projection of the space-time prism, rather than the ellipse itself, has a practical motivation. In the end, road networks are stored in GIS databases using the predicate `POINT(x,y)`, which is standardized by OpenGIS within the well-known text format. To verify that a road network point is in the bounding box $[X_1, Y_1] \times [X_2, Y_2]$, the following SQL query can be used:

SELECT `vertex` FROM `roadnetwork`
WHERE $X_1 \leq$ `vertex.`x AND `vertex.`$x \leq X_2$ AND $Y_1 \leq$ `vertex.`y AND `vertex.`$y \leq Y_2$

This query is simple and efficient, due to the spatial index on the vertices in a road network.

3.2.3. An Algorithm for *k*-Shortest Path Routing

We start this section explaining the k-shortest path routing, well-known in the field of networks and which we use in the sequel. It does not only find a shortest path between two points, but also $k-1$ other paths in increasing order of cost. Here, k is a parameter that indicates the number of shortest paths to be found. For our purposes, we adapt Yen's algorithm to rank the k loopless shortest paths [4]. This algorithm first computes the shortest path between two vertices using the A^*-algorithm [37]. Then, it takes the n-th vertex in the shortest path, starting with $n = 1$ and increasing n until $n = k - 1$, and calculates the shortest path from the n-th vertex to the end vertex, called a spur path. The path from the start vertex to the n-th vertex is called a root path. Two restrictions are placed on a spur path of a vertex: (a) it must not pass through any vertex on the root path of that vertex (to ensure that the paths are loopless); and (2) it must not branch from the current vertex on any edge used by a previously found k-shortest path. The second condition means that the spur path cannot start with an edge that is already in a previously found shortest path. For example, if we have already found the shortest paths $A \to B \to C \to D$ and $A \to B \to E \to F \to D$, and we have $A \to B$ as a current root path, then we cannot use the edges $B \to C$ or $B \to E$, because these would result in an already found path. If a new spur path is found, it is appended to the root path for that vertex, to form a complete path from start to end vertex. Example 1 illustrates the use of Yen's algorithm.

Example 1. Consider the road network of Figure 4 and assume that all road segments (arrows) are equally weighted. We want to find the shortest path from A to D. It is clear that the shortest path is $A \to B \to C \to D$, so we include this path in the result path. Now, we look for other paths starting from the shortest one. We start with root path A and look for a path from A to D that is not already in the result list. The only possible path, not including the edge from A to B, is $A \to E \to F \to G \to H \to D$. We add this path to the result list (which would now be the result for $k = 2$). Now, we start with $A \to B$ as the new root path and find $A \to B \to F \to G \to H \to D$. These are all of the possible paths, and the algorithm ends. □

$$\begin{tikzcd}
A \arrow[r] \arrow[d] & B \arrow[r] \arrow[d] & C \arrow[r] & D \\
E \arrow[r, dash] & F \arrow[r, dash] & G \arrow[r] & H \arrow[u]
\end{tikzcd}$$

Figure 4. A road network for Example 1.

There are several special cases of MM inputs that are difficult to handle just using a shortest path algorithm. For example, Figure 5 depicts an MO that has followed the road indicated in thin lines (three sample points are shown). Algorithms based on shortest path computation would likely choose the thick line road (shown in red). We call this problem the triangle problem. Suppose, for instance, that scores of 6, 5 and 6 are given to Road Segments 1, 2 and 3, respectively. Dijkstra's algorithm would decide between the segments $1 + 2$ and 3 and choose Segment 3 (the one with smallest score), although we know that the path $1 + 2$ is the actual matching path. We solve this by tweaking Dijkstra's algorithm to take the highest-scored path, by working with one divided by the total score of a path, rather than by using the total score. In Section 3.2.4, we discuss how to solve this issue by assigning a (calculated) weight to each road segment.

Calculating the spur paths from each vertex has complexity $O(N)$ and using Dijkstra's shortest path algorithm [38], $O(N^2 + |E|)$ with $|E|$ the number of road segment (edges). The A^*-algorithm has an $O(N^2)$ upper time complexity bound. Since the time complexity for both Dijkstra's algorithm and the A^*-algorithm is $O(N^2)$ and, in the worst-case, we have to compute it for all N vertices, we obtain $O(N^3)$ as the complexity of the algorithm.

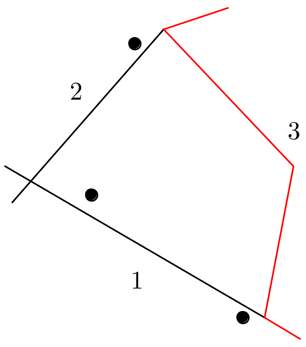

Figure 5. A problem with shortest path approaches.

3.2.4. Description of the Space-Time Prisms Map Matching Algorithm

Now, we show how we can use the k-shortest path algorithm and avoid, for instance, problems like the one in Figure 5, by using the notion of space-time prisms (introduced in Section 3.1) and by adding weights to the edges in the road network. Additionally, space-time prisms allows us to use datasets that contain outliers, since the weight that the related edges receive in such cases is negligible. First, the algorithm computes the road segments closest to the recorded space-time sample points, as follows. For each two consecutive space-time points, we compute the bounding box of the projection of their space-time prism, as explained in Section 3.2. Then, we give weights to the road segment, in a way such that the road segment closest to a given space-time sample point gets weight m, where m is the number of road segments to which we want to give a weight. The second closest road segment will have weight $m - 1$, continuing until the m-th closest road segment, which receives weight one. We remark that only the road segments included in the bounded box are taken into account, avoiding including roads that are very unlikely to have been followed. Finally, the k-shortest paths are calculated, as well as each path's total weight. The path with the highest weight is selected as the map-matched path.

Example 2. An example of this algorithm, with a maximum weight of three, can be found in Table 1 and Figure 6. We describe next how the algorithm works, assuming that the space-time prisms have already been computed (that is, we know which edges are relevant). Starting from point A, we assign a weight to each road segment according to the closeness of the segment to this point. Therefore, the road segment with $id = 1$ receives a score of three, and the road segments with $id = 3$ and $id = 4$ receive weights of two and one, respectively. These weights can be found in the third column of Table 1 (that means, the weight for each segment and Point A). Thus, A will likely be matched to the road segment with $id = 1$. Next, we consider Point B, for which the weights are 3 (for the segment with $id = 4$), 2 (for Segment 1) and 1 (for Segment 3), according to the closeness of Point B to those segments. The total weights for Segments 1, 3 and 4 will be, respectively, 5 (that is, $3+2$), 3 (that is, $2+1$) and 4 (that is, $1+3$). Since Segment 1 is farther from all other points (considering three to be the maximum weight), the weight of this segment will remain as five. We continue in this way until all points have been analyzed and the weights assigned. Table 1 shows the outcome of the weighting algorithm. The rightmost column gives the total weight for each segment. Let us suppose that we have chosen $k=2$. Given the weights calculated above, we now compute the two shortest paths. The first path we obtain is $1 \rightarrow 4 \rightarrow 10 \rightarrow 11 \rightarrow 16$; the second one is $1 \rightarrow 3 \rightarrow 5 \rightarrow 10 \rightarrow 11 \rightarrow 16$. Although the second one has more edges, its total weight (44) is smaller than that of the first one (46), which is, indeed, the actual route followed by the MO. Note that this weight is computed as the sum of the weights of the involved segments (that is, $5+7+13+12+9=46$). $\square$

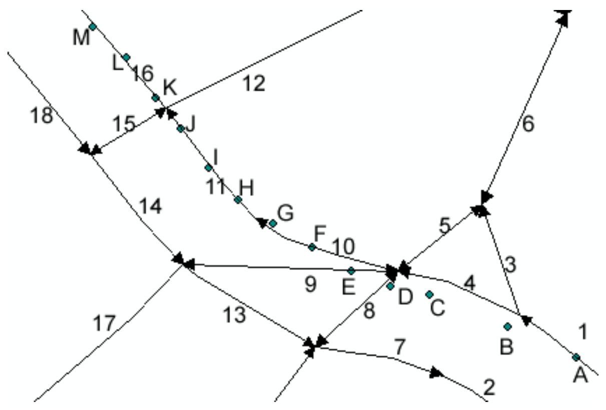

Figure 6. The symbols A, ..., M represent sample points, and the symbols 1, ..., 18 are identifiers of road segments.

Table 1. An example of the algorithm for the assignment of weights.

Id	Init	A	B	C	D	E	F	G	H	I	J	K	L	M
1	0	3	5	5	5	5	5	5	5	5	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	2	3	3	3	3	3	3	3	3	3	3	3	3
4	0	1	4	7	7	7	7	7	7	7	7	7	7	7
5	0	0	0	2	2	2	2	2	2	2	2	2	2	2
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	4	5	5	5	5	5	5	5	5	5
9	0	0	0	0	2	5	7	8	9	9	9	9	9	9
10	0	0	0	0	1	3	6	9	11	13	13	13	13	13
11	0	0	0	0	0	0	1	3	6	9	12	12	12	12
12	0	0	0	0	0	0	0	0	0	1	3	4	5	5
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	1	3	5	5
16	0	0	0	0	0	0	0	0	0	0	0	3	6	9
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In summary, our MM algorithm proceeds as follows:

Step 1. Select the relevant parts of the road network by calculating, for each pair of consecutive points, which road segments the MO could have driven on (using the bounding boxes of the projections of space-time prisms, as described in Section 3.2).

Step 2. For each sample point, compute the closest road segment, as described in Section 3.2.4, and assign scores to each road segment. A score for a segment s is computed adding up the weights of all of the segments that match s.

Step 3. Compute, within the limited road network determined in Step 1, the k-shortest paths, and select, as output, the shortest path with the maximum total weight. If we obtain two paths with the same weight, the first path would be selected.

We remark that, sometimes, simply adopting a starting and ending vertex from the road segments closest to the first and last time-space point, respectively, may not result in a correct match. This is illustrated in Figure 7. Here, the road segment closest to $p1$ would be $R1$, although the trajectory clearly follows $R2$. The ending point will be matched to $R2$, eventually preventing finding a route for this trajectory. To solve this problem and taking into account that the maximum measurement error is about 10 m, the algorithm looks at all of the possible starting segments (instead of only one) within a circle with a radius of five meters around the first time-space point and selects all road segments that intersect with this circle. The same procedure is followed for the end segment. Then, the algorithm looks for possible k-shortest paths between all of the possible start segments and end segments within these boundaries (that is, the circles).

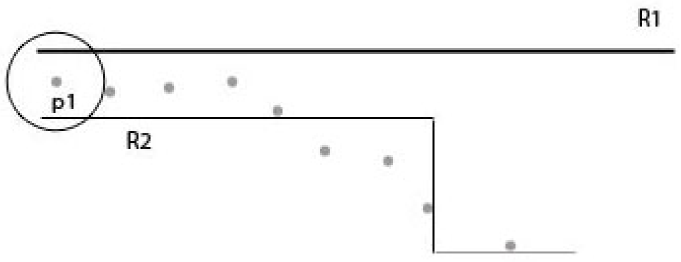

Figure 7. An example of an ambiguous trajectory start.

4. An Improved Evaluation of Map Matching Algorithms

In Section 2, we discussed the most relevant MM algorithms in the literature. In this section, we give a new method to compare accurately the performance of such algorithms against each other. We first discuss data sources and properties, and then, we present the accuracy measures that are commonly used to evaluate the result of MM algorithms.

4.1. Data Sources and Properties

Among the sources of data used to apply MM algorithms we have: (a) human-labeled data (this is, the case when a person identifies herself/himself, doing a manual MM that can be considered 100% correct); (b) computer-generated data (obtained through automatic methods); and (c) unknown source data.

For Method (b), for example, using an existing road network, we randomly select two nodes in this network, and using a k-shortest path algorithm, k paths are generated. From this collection of paths, one is randomly selected and chosen as the real trajectory produced by an MO. The trajectory sample is generated by producing points close to the road segment (within a certain distance). The number of these points is based on the chosen sampling rate. This method is used in our implementation. For Method (c), it may happen that the actual trajectory followed for a trajectory sample is not known. Given that, in order to evaluate a particular MM algorithm, we need a "correct" matching, we can generate the correct labeling using another MM algorithm. Given that the accuracy of this second algorithm is also not known, the results obtained from comparing the first to the second algorithm may be inaccurate and have limited meaning.

MM algorithms are very sensitive to the characteristics of the samples used as input. Among these characteristics, we will consider the following: (a) sampling rate (average number of space-time points per second of a trajectory sample); (b) road length; (c) sampling errors (in the points or due to noise in the sample); (d) length of the route; (e) road density (number of road segments per area of the map).

4.2. Methods to Measure the Quality of a Map Matching Algorithm

There are multiple ways to measure the correctness or accuracy of an MM algorithm on a given trajectory sample in a given road network. We list the most important ones below.

1. Accuracy by length: This measure (see [21]) is computed as the total length of correctly matched road segments divided by the total length of the matched trajectory. We see that a long road (implying a big error) would be more important than a short road.

2. Accuracy by number: This method (see [21]) uses the number of correctly matched road segments. Thus, a mismatch of a long road segment is treated in the same way as a mismatch of a short road segment. It is computed as the number of correctly matched road segments divided by the number of road segments in the matched trajectory. This method works well when the number of correctly matched segments is more important than the correctly matched length, e.g., an algorithm with high accuracy by number will be appropriate to apply to a city center.

3. Accuracy by length minus erroneous road: This method (see [2]) uses the total length of the erroneously added and removed road segments. It has the same benefits as accuracy by length. The main difference is that incorrectly added road segments have a bigger penalty. Since this measure does not give us an idea of how accurate the calculated trajectory is, we combine it with the accuracy by length measure, giving the following formula for accuracy by length minus erroneous road: we take the value:

$$\frac{\text{length of correctly matched road segments} - d_- - d_+}{\text{total length of the matched trajectory}},$$

if this value is positive, and we take zero, otherwise.

4. Weak, average and strong Fréchet distance: This method [11,12] uses the Fréchet distance measure(s) between two curves. The average Fréchet distance is the average of the weak and strong Fréchet distances, explained in Section 2. A benefit of using the Fréchet distance method is that we look at the curve itself, instead of at the road segments. A matched trajectory might have a low accuracy with the other measures, but can still be very close to the actual path. Using this distance method allows us to calculate how close these paths are. However, we could get a high accuracy score for a path that does not have any road segment in common with the original path (for instance, a parallel path). Assume A and B are two curves; the Fréchet distance is formally defined as:

$$F(A,B) = \inf_{\alpha,\beta} \max_{t\epsilon[0,1]} \left\{ d(A(\alpha(t)), B(\beta(t))) \right\},$$

where reparametrizations $\alpha, \beta : [0,1] \to [0,1]$ of A and B are continuous, non-decreasing surjections. In the above formula, d is the standard Euclidean distance function. The weak Fréchet distance removes the non-decreasing requirement.

4.3. Problems in Measuring Accuracy

The above methods suffer from some problems in certain settings. A first example occurs when an MM algorithm selects incorrect road segments, which are parallel to the actual trajectory. The accuracy by length, accuracy by number and the length of erroneous road quality measures give no score to segments that are parallel to the correct path or even penalize those segments. Thus, the matched trajectories that are parallel to the correct trajectory, but still very close to it, get a too low score. Quality measures that use the curve distance between the correct trajectory and the matched trajectory (for example, the Fréchet distance) handle this case better, since they do not look at which exact road segment was matched. On the other hand, quality measures using curve distance suffer from another problem when the MM algorithm selects too many segments (for instance, in addition to each correct segment, a parallel segment is incorrectly selected). This causes the matched trajectory to be twice as long as the correct one. Quality measures using curve distance would give this trajectory a high score, since each selected segment is either on the correct curve or very close to it. However, this is inaccurate, due to the large number of incorrectly selected road segments.

4.4. A New Accuracy Measure: CL-Accuracy

We propose a new quality measure to address the problems mentioned in Section 4.3. We combine the strengths of some of the measures discussed above. In short, we compute a curve distance between the correct and the matched trajectory, and then, we take the lengths of both trajectories into account. We denote this new method the curve-and-length-accuracy, or CL-accuracy, for short.

To compute this new measure, we first calculate a score between zero and 100 for each segment selected by the MM algorithm. For this, we compute the Euclidean distance to the closest segment on the correct trajectory, with a maximum of 100 m. This limit is based on the accuracy of current GPS devices (if a point is more than 100 m away, it is almost certainly an outlier). The total score for the matched trajectory is calculated by adding up all of the segment scores and then using the following formula:

$$\text{score} = \frac{\text{maxScore} - \text{score}}{\text{maxScore}}.$$

The term maxScore in the above formula is the maximum score possible for the matched trajectory (100 times the number of segments). This formula computes a score for the curve distance between the correct and matched trajectory. A score of 100% means that each segment of the matched trajectory is on the correct trajectory. A score of 0% means that each segment is 100 m or more away from the correct trajectory. Next, we take the length of both trajectories into account, multiplying the score above by the quotient between the lengths of the smallest and largest trajectories. If O is the original trajectory and M its map matched version, then O can be longer or shorter than M (depending on the algorithm that is used). Since we want to obtain a similarity measure in the range 0% to 100%, if $|O| > |M|$, we use $\frac{|M|}{|O|}$ instead. As a consequence, if, for instance, M is 10% longer than O, this is penalized in exactly the same way as if O were 10% longer than M. With this procedure, a matched trajectory that is twice as long as the correct trajectory only gets half the score.

5. Experimental Comparison of Map Matching Algorithms

We now present a number of tests of different MM algorithms on a variety of trajectory sample datasets. The aim is to establish which type of MM algorithm works best on a certain type of trajectory sample. We have implemented a number of existing algorithms and compared these with our own uncertainty-based MM algorithm (presented in Section 3.2.4). We have selected algorithms from each category discussed in Section 2, in order to compare our own approach to representatives of each category. The algorithms that we selected for the geometric analysis category are the curve-to-point algorithm and the curve-to-curve algorithm, but we have made a few changes in them: we use Dijkstra's algorithm when two consecutive matched segments are not connected. This ensures there are no gaps in the calculated trajectory. We have selected Greenfeld's algorithm for the topological analysis category, since this was one of the first algorithms of its kind, and many other algorithms resemble this method. For the probabilistic category, we have implemented the algorithm by Ochieng, Quddus and Noland. For the low sampling rate category, we have implemented the ST-matching algorithm [21]). Finally, we have also implemented our space-time prisms combined with *k*-shortest paths algorithm. Although there are some more recent algorithms proposed, most of them do not provide conclusive evidence of effectiveness, and we believe (and this follows from the literature [5]) that the ones we chose are mature enough and have been proven through time to be effective for different cases. For the space-time prisms algorithm, in the experiments, we have considered (based on previously performed experiments) a maximum weight of $m = 50$, $k = 10$ (note that k is an upper bound, rather than a constant) and a maximum speed of 120 km/h.

In Section 5.1, we describe the tests of these algorithms on trajectory samples generated by a GPS-equipped device. In Section 5.2, we show the tests on computer-generated trajectory samples. Finally, in Section 5.3, we combine all of these results and formulate a conclusion on the discussed MM algorithms. All tests were run on a Macbook with a 2.16-GHz Intel Core 2 Duo processor and 1 GB RAM.

5.1. Tests on Human Labeled Data

For the tests on human labeled data, we worked with trajectory samples generated by a GPS-equipped device that are manually labeled. In these data, timestamps, speed and heading information are included. The data used for this test are twenty trajectory samples from the police force of Ghent, Belgium. The trajectory samples were located in the city and collected by intervention cars of the police. Each sample contains approximately 400 space-time points, comprising a length of about 4 km. They also contain some data gaps, which are usually caused by driving through tunnels (where GPS reception is absent). In Figure 8, we show the average result on the twenty trajectory samples from the police dataset. An overview of average running times is given in Table 2.

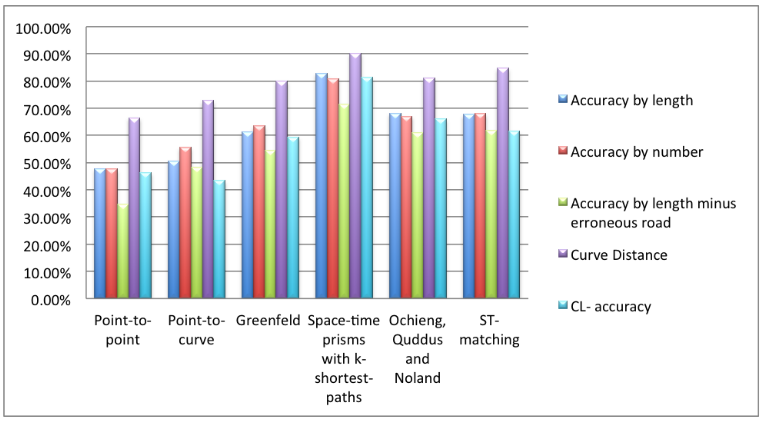

Figure 8. Average results of the map matching (MM) algorithms on 20 trajectory samples of the Ghent police dataset.

Table 2. Average running time in seconds of the MM algorithms for the dataset of the Ghent police.

Algorithm	Average on 20 Samples
Point-to-point	77,725
Point-to-curve	699,015
Greenfeld	168,845
Space-time prisms with k-shortest paths	6015
Ochieng, Quddus and Noland	721,515
ST-matching	7,612,535

At first glance, geometric analysis algorithms perform the worst of all, which makes sense, since they do not look at anything but the location of the space-time point. Information about where previous points were matched or how the road network is connected is all ignored. The other types of algorithms all seem to perform reasonably well. For the average case, our space-time prisms combined with k-shortest paths algorithm appears to perform the best. In Section 5.2, we discuss tests of these algorithms for many different test cases to figure out which algorithm works best in several different scenarios. Running times are similar for all of the tested algorithms, although, of course, given its simplicity, Greenfield's performs best, at the expense of accuracy. In general, all of these algorithms can run in real time on a route planner. The exception is the ST-matching algorithm (76 s on average), which cannot be run in real time if the sampling rate is too high. This is an expected result, as we commented on in Section 2, given that it is a global algorithm.

It may seem counter-intuitive that the geometric analysis algorithms perform a little bit slower than most of the other algorithms. This is due to a modification we have done in these algorithms. In our implementation of the geometric analysis algorithms, we use Dijkstra's shortest path algorithm when two consecutive matched segments are not connected. This ensures that there are no gaps in the calculated trajectory. Without this change, these algorithms would run substantially faster than they do now, but still give worse results.

5.2. Tests on Computer-Generated Data

In this section, we use data produced by a trajectory sample generator. Using this generator, we can produce trajectory samples with different properties discussed in Section 4. We divide the study into high and low sampling rates.

5.2.1. High Sampling Rate

Here, the time difference between two consecutive sample points is less then 10 s.

Simulated Trajectory Samples with No Measurement Errors

For this test, we created trajectory samples with space-time points being recorded every two seconds and lengths of a few kilometers. Figure 9 (left) shows the average result on ten computer-generated trajectories. We can see that, as expected, all of the algorithms could detect the correct path reasonably well, with the exception of the point-to-point algorithm, which selected too many segments, resulting in a score lower than the rest. The other algorithms can be said to be accurate enough to be used with this type of data.

Simulated Trajectory Samples with Measurement Errors

For this test, we created trajectory samples with space-time points with an interval between three and 10 s, with an error between zero and 15 m for each point. Figure 9 (right) shows the average test result on ten trajectories. The algorithms using geometric analysis perform poorly in this case, since they generate many false positives. The other types of algorithms perform as on data without errors and generally have no problem handling these errors.

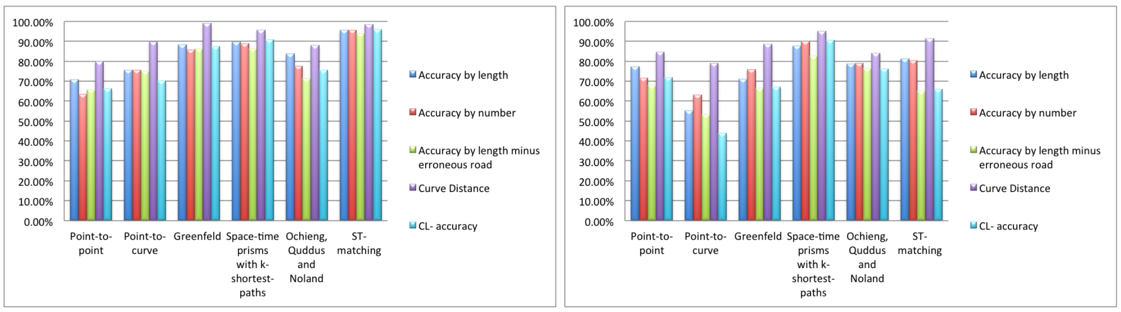

Figure 9. Average results of the MM algorithms on ten computer-generated trajectory samples with no (**left**) and with (**right**) measurement errors.

Simulated Trajectory Samples with Measurement Errors and Outliers

Data collected with a GPS-equipped device can not only contain measurement errors, but also outliers. We produced samples to simulate this situation. Figure 10 (left) shows the average result on ten trajectory samples with a high sampling rate (three to 10 s between space-time points), with measurement errors (between zero and 15 m per point) and with outliers (one to three outliers per dataset, ranging from 10 to 250 m large). We can see that the geometric algorithms cannot handle outliers very well, because they compute the shortest path between each of two consecutive points and add this route to the calculated trajectory. Thus, many road segments leading to the outliers are incorrectly added to the calculation, resulting in a poor score. The ST-matching algorithm has a similar method of calculating the trajectory and also has a poor score caused by this. The space-time prisms with the k-shortest paths algorithm performs very well in this case and gets a perfect score in the above test.

Simulated Trajectory Samples with Measurement Errors and Gaps

When passing through tunnels, there is no connection with GPS satellites to calculate an MO's location, causing gaps in the trajectory samples. We have generated trajectory samples with gaps, to simulate this situation, and then ran our tests. We generated 10 trajectory samples with a high sampling rate (three to 10 s between points), measurement errors (between zero and 15 m per point) and gaps (one to three gaps per sample, ranging from 50 to 200 m in size). The average result of the MM algorithm on those samples is shown in Figure 10 (right). In general, gaps did not cause a problem for any of the algorithms, although ST-matching was unable to calculate a trajectory for two of the datasets. Our space-time prism with *k*-shortest paths algorithm and the algorithm by Ochieng et al. performed very well here.

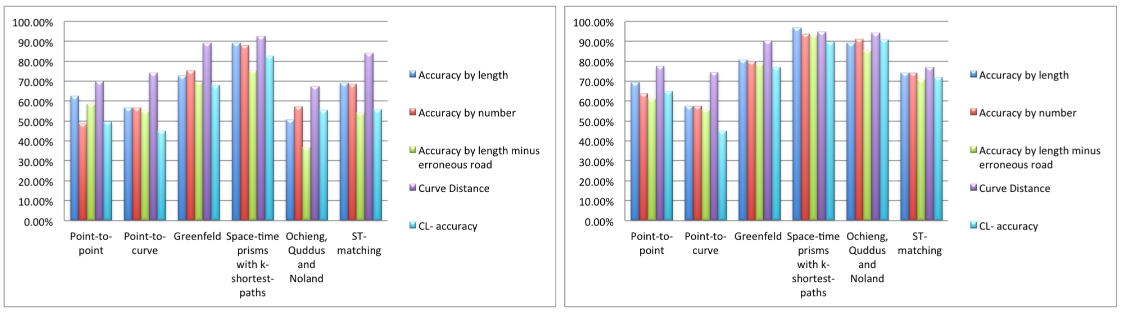

Figure 10. Average results of MM algorithms on ten computer-generated trajectory samples with measurement errors and outliers (**left**) and with measurement errors and gaps (**right**).

Simulated Trajectory Samples on a Highway with GPS Errors

For this test, we simulated trajectory samples on a highway, instead of near or in a town. On a highway, there are less possible road segments from which to choose. Figure 11 (left) shows the average results on ten simulated trajectory samples with a high sampling rate (three to 10 s between GPS points), with measurements errors (between zero and 15 m per point). The poor score of most algorithms can be explained by the fact that algorithms often choose parallel roads, combined with non-highway roads near the highway. The actual trajectory or a parallel trajectory was selected in each case, but due to the selection of many incorrect road segments, they still got a poor score. Exceptions to this are the space-time prism with *k*-shortest paths and the probabilistic algorithm by Ochieng et al., which worked quite well (the latter being the best of all).

Simulated Trajectory Samples on a Long Distance with Measurement Errors

The datasets used in this section so far only contained medium-sized trajectory samples. Since some of the implemented algorithms are incremental, errors at the beginning of the algorithm may cause a big difference in the subsequent calculations. We now use trajectory samples containing several thousand of space-time points. The average result on ten trajectory samples with a high sampling rate (three to 10 s between space-time points), with measurement errors (between zero and 15 m per point) and several times more space-time points than in the previous samples is shown in Figure 11 (right). We can see that, in general, all algorithms performed well in this case.

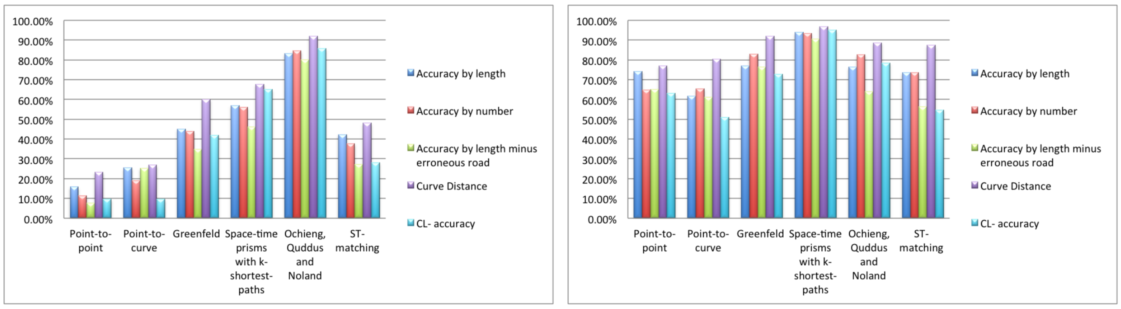

Figure 11. Average results of the MM algorithms on ten computer-generated trajectories simulating driving on a highway with GPS errors (**left**) and on ten long computer-generated trajectories with measurement errors (**right**).

Remark 1. We remark that the poor score of the probabilistic algorithms is caused by the fact that the algorithm gets stuck when selecting one specific road segment, since the alternative has a very low score and is never selected. Due to this problem, this algorithm does not progress further than this road segment. This causes a bad score. In the other tests, no such problems were encountered.

5.2.2. Low Sampling Rate

Now, we discuss samples where sample points are much longer than 10 seconds apart.

Simulated Trajectory Samples with No Measurement Errors

The trajectory samples used in this test follow the same trajectory as the ones used in the analogous case in Section 5.2.1. We generated ten trajectories with a low sampling rate (60 to 150 s between consecutive space-time points) and with no measurement errors. The results are shown in Figure 12 (left). As expected, the ST-matching algorithm, specifically designed for MM data with low sampling rates, scores the best. The geometric analysis algorithms and the space-time prism with *k*-shortest paths algorithm work well in most cases. However, when the sampling rate drops to very low values, the space-time prism with *k*-shortest paths algorithm fails to generate a path, which occurred twice in our tests. These particular cases are not shown in the figure, but had an impact on the overall average results, and in this case, the graph in Figure 12 (left) penalizes the space-time prisms algorithm. Greenfeld's algorithm and the probabilistic algorithm performed poorly on data with a low sampling rate. The reason is that these algorithms only pick a maximum of one road segment per space-time point, and thus, not enough road segments are selected.

Simulated Trajectory Samples with Measurement Errors

Here, also the trajectory samples follow the same trajectory as the ones used in the analogous case in Section 5.2.1. We generated ten trajectory samples with a low sampling rate (60 to 150 s between space-time points) and with measurement errors (between zero and 15 m per point). Figure 12 (right) shows the average results. We see that introducing errors in the data does not affect the results in a relevant way. The ST-matching algorithm and the geometrical analysis algorithm perform very well. On the contrary, Greenfeld's algorithm, the space-time prism with *k*-shortest paths algorithm and the probabilistic algorithm perform poorly.

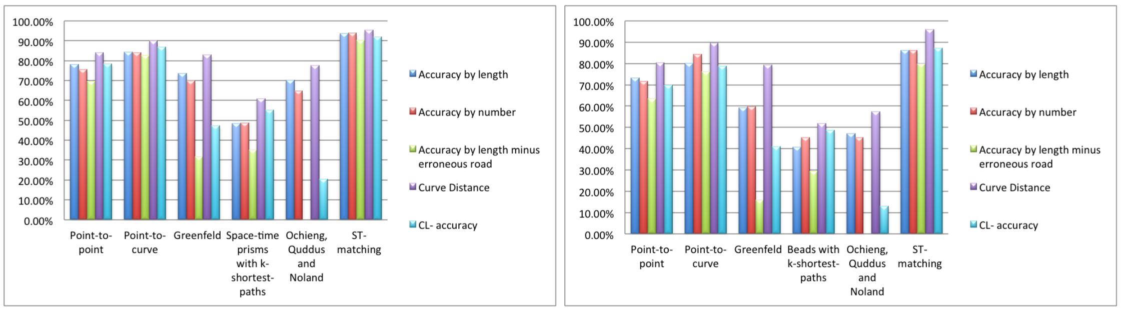

Figure 12. Average results of the MM algorithms on ten computer-generated trajectories with a low sampling rate (**left**) and no measurement errors and measurement errors (**right**).

5.3. Discussion

We now recapitulate and aggregate the above results and briefly discuss which algorithm is best suited in different situations. As an overview, we show in Figure 13 the average matching errors of all tests on the dataset with multiple trajectory samples. In Figure 14, we present the average running time per case and, in the last column, an overall average.

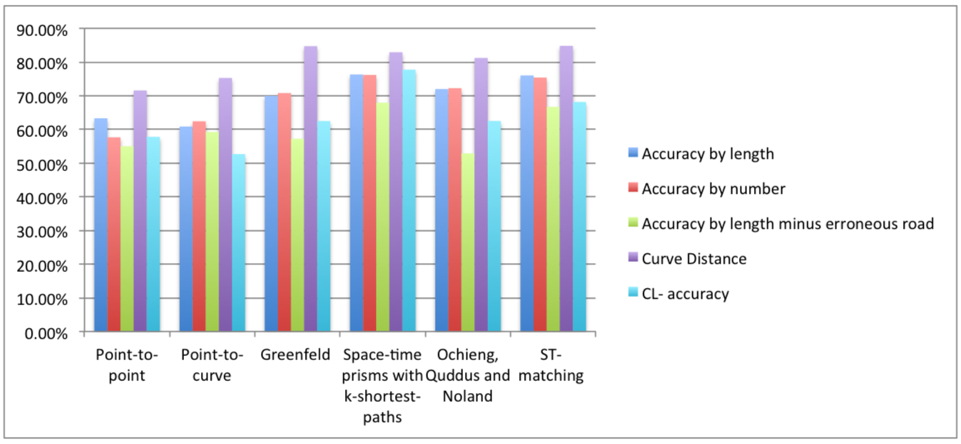

Figure 13. Average of the results from Figures 8–12.

In most cases, the geometric analysis algorithms (point-to-point and point-to-curve) returned the least accurate results, although they run faster than the rest, since this type of algorithm requires less calculations. This is, of course, relevant for devices with low processing power.

The probabilistic algorithms by Ochieng, Quddus and Noland also performed well in nearly every test case. The algorithm just had problems with long trajectories or trajectories with a low sampling rate, and performed substantially better than the other algorithms when the data are located on a highway. The ST-matching algorithm, an algorithm specifically created for data with a low sampling rate, had varying results on data with a high sampling rate. Due to these varying results and the long run-time required to run this algorithm, we think it should be avoided for use on data with a high sampling rate. For data with a low sampling rate, however, the algorithm proves to calculate a very close match to the trajectory in a reasonable running time.

The space-time prism with *k*-shortest paths algorithm works well in most cases. The exception occurs when there are many outliers in the data: due to the way the algorithm computes the path, outliers cause small hiccups in the calculation, which causes the algorithm to slow down when encountering an outlier. It also has problems when the sampling rate is very low. However, in general, the space-time prism with *k*-shortest paths algorithm is competitive with the algorithms that perform well and, in many cases, outperforms them, showing also a reasonable performance in many different scenarios. We remark that even if it works better under a high data sampling rate, it also works well for low sampling rate situations (if the rate is not too low).

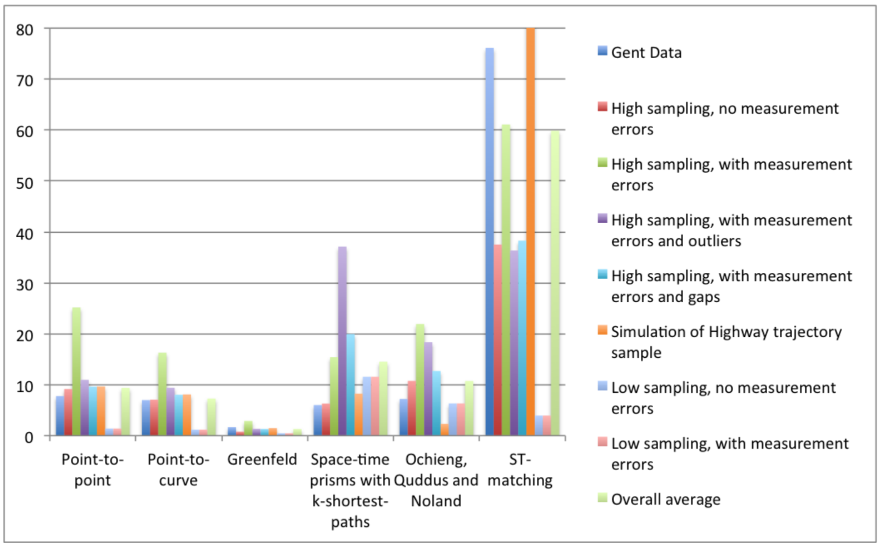

Figure 14. Average running times in seconds of the results of the data shown in Figures 8–12, where, for visibility, we cut the orange line at 80 s.

6. Conclusions and Open Problems

We have presented a novel algorithm, which accounts for the uncertainty of the sample points in a trajectory, to solve the MM problem. Using the notion of space-time prisms, together with weighted *k*-shortest path algorithms, results in an algorithm that, in addition to being applicable to a wide range of trajectory sample types, in most cases outperforms well-known MM algorithms. We support this claim by implementing those algorithms and applying a novel method for measuring the accuracy of an MM algorithm.

Historically, the area of MM has been driven, mostly, by experimental results. Algorithms are often compared by means of tests over datasets chosen or produced in an ad hoc fashion. Further, in the MM field, there is no repository of existing implementations. Therefore, in this paper, we have compared our space-time prism with *k*-shortest paths algorithm with existing MM algorithms that we have implemented ourselves, probably missing some optimizations. Thus, we believe that a general, flexible benchmark for the evaluation of MM algorithms is missing and is a suitable target for future work.

Acknowledgments: A short description of the algorithm, presented here, was published as a poster paper at a conference [39]). This conference publication lacks any kind of evaluation and experimental comparison with other algorithms. Alejandro Vaisman was supported by a travel grants from Hasselt University (Korte verblijven–inkomende mobiliteit, BOF15KV13 and BOF16KV09). He was also partially supported by PICT-2014 Project 0787.

Author Contributions: All authors contributed to the design of the map matching method and of the experiments; Bart Kuijpers and Bart Moelans performed the experiments and analyzed the results; all authors contributed to the writing of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Computation of the Projection of a Space-Time Prism and Its Bounding Box

The theorem in this section extends Theorem 1. It shows how the bounding box $[X_1, Y_1] \times [X_2, Y_2]$ of an ellipse in the plane can be determined, given the focal points (x_1, y_1) and (x_2, y_2) of the ellipse and the semi-major axis $L > 0$ of the ellipse. Figure A1 gives an illustration of an ellipse and its bounding box.

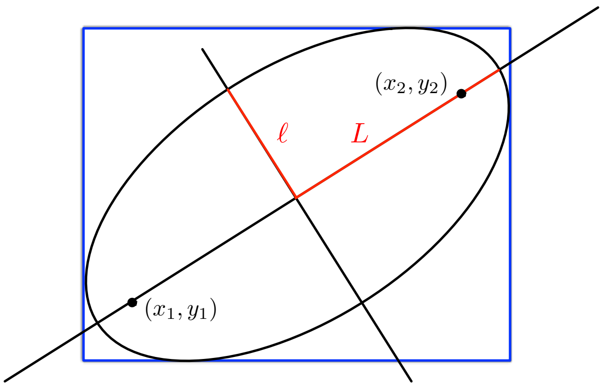

Figure A1. An ellipse with focal points (x_1, y_1) and (x_2, y_2), semi-major axis L and semi-minor axis ℓ (in red). Its bounding box is shown in blue.

We remark that the major axis, $2L$, expresses twice the longest distance from the center of the ellipse to a point on the ellipse. Loosely, we will also use the terms major axis and minor axis (see below) to indicate the actual lines that carry these lengths.

To contain the length of expressions, we introduce some abbreviations. The center of the ellipse is the point

$$(x_c, y_c) = (\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}).$$

The distance between the foci is abbreviated by d_f, that is

$$d_f = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

The semi-minor axis of the ellipse is denoted by ℓ. Therefor, 2ℓ is the length of the minor axis of the ellipse.

To determine ℓ, we look at Figure A2, where a and b are points on the intersection of the ellipse with its major and minor axis, respectively. If we think of the rope-drawing construction of the ellipse, then we can see that:

$$d_f + d((x_1, y_1), a) + d((x_2, y_2), a) = d_f + d((x_1, y_1), b) + d((x_2, y_2), b),$$

since both the left and the right side in this equality equal the length of the rope, needed to draw the ellipse. We also know that $d((x_1, y_1), a) + d((x_2, y_2), a) = 2L$, the length of the major axis.

Furthermore, we know that $d((x_1, y_1), b) = d((x_2, y_2), b) = H$, since we have an equilateral triangle, if we call H the length of the line segment connecting a focal point with b (that is, the length of the hypotenuse of the triangle formed by a focal point, the center of the ellipse and b). Therefore, we get $d_f + 2L = d_f + 2H$ or $L = H$. Pythagoras' theorem, applied to the triangle formed by the center of the ellipse, the focal point (x_1, y_1) and the point b, then gives $H^2 = (\frac{d_f}{2})^2 + \ell^2$. Since $L = H$, we obtain

$$\ell = \frac{1}{2}\sqrt{4L^2 - d_f^2}.$$

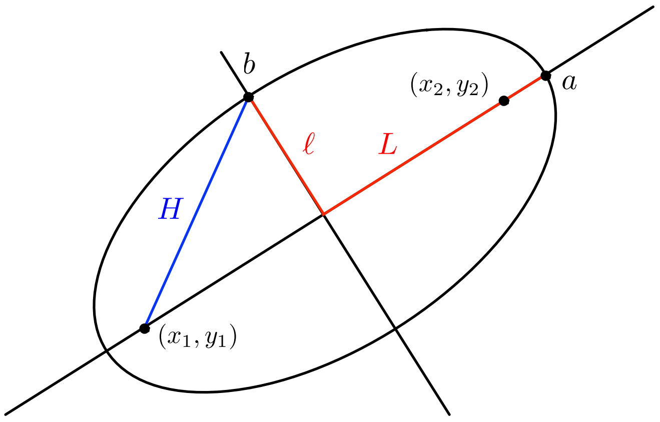

Figure A2. An ellipse with focal points (x_1, y_1) and (x_2, y_2), semi-major axis L and semi-minor axis ℓ (in red) and points a and b on the major and minor axis, respectively. The length H of the line segment connecting a focal point with b is indicated in blue.

Finally, if $x_1 \neq x_2$, we abbreviate the slope of the line connecting (x_1, y_1) and (x_2, y_2) by s, that is:

$$s = \frac{y_1 - y_2}{x_1 - x_2}.$$

The goal of the following theorem and its main contribution is to give an explicit description of the bounding box of an ellipse in the plane, given the focal points (x_1, y_1) and (x_2, y_2) of the ellipse and the semi-major axis $L > 0$ of the ellipse. Since the axes of the ellipse are not necessarily parallel to the coordinate axes, we also derive a formula that describes the ellipse (since in textbooks, such formulas are generally only given for "standard" ellipses).

The following theorem is a detailed version of Theorem 1. It distinguishes between four configurations of the foci of the ellipse and additionally gives the equation of the ellipse.

Theorem 2. *Let (x_1, y_1) and (x_2, y_2) be points in $\mathbf{R}^2$, and let $L > 0$ be a real number. The equation of the ellipse with foci (x_1, y_1) and (x_2, y_2) and semi-major axis L and its bounding box is given by the following expressions:*

- *If $(x_1, y_1) = (x_2, y_2)$, then:*
$$(x - x_c)^2 + (y - y_c)^2 = L^2$$
is the equation of the ellipse, and the bounding box is given by $X_1 = x_c - L$, $X_2 = x_c + L$, $Y_1 = y_c - L$ and $Y_2 = y_c + L$;
- *If $x_1 = x_2$ and $y_1 \neq y_2$, then:*
$$\frac{(x - x_c)^2}{\ell^2} + \frac{(y - y_c)^2}{L^2} = 1$$
is the equation of the ellipse, and the bounding box is given by $X_1 = x_c - \ell$, $X_2 = x_c + \ell$, $Y_1 = y_c - L$ and $Y_2 = y_c + L$;

- *If $x_1 \neq x_2$ and $y_1 = y_2$, then:*
$$\frac{(x - x_c)^2}{L^2} + \frac{(y - y_c)^2}{\ell^2} = 1$$
is the equation of the ellipse, and the bounding box is given by $X_1 = x_c - L$, $X_2 = x_c + L$, $Y_1 = y_c - \ell$ and $Y_2 = y_c + \ell$;
- *If $x_1 \neq x_2$ and $y_1 \neq y_2$, then:*
$$\frac{(y - y_c - s(x - x_c))^2}{\ell^2} + \frac{(s(y - y_c) + (x - x_c))^2}{L^2} = (1 + s^2)$$
is the equation of the ellipse, and the bounding box is given by:
  - $X_1 = x_c - \sqrt{\frac{s^2\ell^2+L^2}{1+s^2}}$,
  - $X_2 = x_c + \sqrt{\frac{s^2\ell^2+L^2}{1+s^2}}$,
  - $Y_1 = y_c - \sqrt{\frac{s^2L^2+\ell^2}{1+s^2}}$ *and*
  - $Y_2 = y_c + \sqrt{\frac{s^2L^2+\ell^2}{1+s^2}}$.

Proof. Let (x_1, y_1) and (x_2, y_2) be points in $\mathbf{R}^2$, and let $L > 0$ be a real number. In each of the four cases of the theorem, we first want to find the equation of the ellipse with foci (x_1, y_1) and (x_2, y_2) and semi-major axis L and then determine its projections on the x- and the y-axis.

The first three cases are trivial (high-school geometry). Only the forth case requires some work. We remark that Case 3 coincides with Case 4 for $s = 0$.

Case 1. We assume $(x_1, y_1) = (x_2, y_2)$. In this case, the ellipse is a circle with center $(x_c, y_c) = (x_1, y_1) = (x_2, y_2)$ and radius L. It is given by the equation:

$$(x - x_c)^2 + (y - y_c)^2 = L^2.$$

The bounding box of this circle is determined by $X_1, X_2 = x_c \pm L$ and $Y_1, Y_2 = y_c \pm L$.

Case 2. We assume $x_1 = x_2$ and $y_1 \neq y_2$. In this case, we have an ellipse where the major axis is in the direction of the y-axis, and the minor axis is in the direction of the x-axis. The equation of this ellipse is:
$$\frac{(x - x_c)^2}{\ell^2} + \frac{(y - y_c)^2}{L^2} = 1.$$
The bounding box of this circle is determined by $X_1, X_2 = x_c \pm \ell$ and $Y_1, Y_2 = y_c \pm L$.

Case 3. We assume $x_1 \neq x_2$ and $y_1 = y_2$. In this case, we have an ellipse where the major axis points in the direction of the x-axis, and the short axis points in the direction of the y-axis. The equation of this ellipse is:
$$\frac{(x - x_c)^2}{L^2} + \frac{(y - y_c)^2}{\ell^2} = 1.$$
The bounding box of this circle is determined by $X_1, X_2 = x_c \pm L$ and $Y_1, Y_2 = y_c \pm \ell$.

Case 4. We assume $x_1 \neq x_2$ and $y_1 \neq y_2$. Therefore, for the slope s, we have $s \neq 0$.

The line "F" connecting the foci (x_1, y_1) and (x_2, y_2) has equation $F(x, y) = 0$, where:

$$F(x, y) = y - y_c - \frac{y_1 - y_2}{x_1 - x_2}(x - x_c) = y - y_c - s(x - x_c).$$

The line "P", perpendicular to F and through (x_c, y_c), has equation $P(x, y) = 0$, where:

$$P(x, y) = \frac{y_1 - y_2}{x_1 - x_2}(y - y_c) + (x - x_c) = s(y - y_c) + (x - x_c).$$

The ellipse with foci (x_1, y_1) and (x_2, y_2) and semi-major axis L then has equation $E(x,y) = 0$, with $E(x,y) = \frac{F(x,y)^2}{A^2} + \frac{P(x,y)^2}{B^2} - 1$ or:

$$E(x,y) = \frac{(y - y_c - s(x - x_c))^2}{A^2} + \frac{(s(y - y_c) + (x - x_c))^2}{B^2} - 1,$$

with $A, B > 0$.

We find B by requiring that the two intersection points of the ellipse with the major axis are at distance $2L$ from each other. These intersection points are the solutions of the system of equations $E(x,y) = 0 \wedge F(x,y) = 0$. From $F(x,y) = 0$, we get $y - y_c = s(x - x_c)$. If we use this equality in $E(x,y) = 0$, we get $(1+s^2)^2(x - x_c)^2 = B^2$ or $x = x_c \pm \frac{B}{1+s^2}$ for the x-coordinates of the two intersection points. The corresponding y-coordinates are $y = y_c \pm \frac{sB}{1+s^2}$. If we set the distance between the points $(x_c - \frac{B}{1+s^2}, y_c - \frac{sB}{1+s^2})$ and $(x_c + \frac{B}{1+s^2}, y_c + \frac{sB}{1+s^2})$ to $2L$, we obtain $B^2 = L^2(1+s^2)$.

Similarly, we find A by requiring that the solutions of the system $E(x,y) = 0 \wedge P(x,y) = 0$ are 2ℓ apart. If we set the distance between the points $(x_c - \frac{sA}{1+s^2}, y_c + \frac{A}{1+s^2})$ and $(x_c + \frac{sA}{1+s^2}, y_c - \frac{A}{1+s^2})$ to 2ℓ, we obtain $A^2 = \ell^2(1+s^2)$.

$$\begin{cases} A^2 &= \ell^2(1+s^2) \\ B^2 &= L^2(1+s^2), \text{ and} \end{cases}$$

Therefore, the equation of the ellipse with foci (x_1, y_1) and (x_2, y_2) and semi-major axis L is given by the equation $E(x,y) = 0$, where:

$$E(x,y) = \frac{(y - y_c - s(x - x_c))^2}{\ell^2} + \frac{(s(y - y_c) + (x - x_c))^2}{L^2} - (1+s^2).$$

To determine the bounding box of this ellipse, we consider the vector:

$$\left(\frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}\right),$$

which for a point (x_0, y_0) on the ellipse (that is, for which $E(x_0, y_0) = 0$) gives the direction perpendicular to the ellipse, when evaluated in (x_0, y_0).

When we set $\frac{\partial E}{\partial x} = 0$, this perpendicular is in the direction of the y-axis. The equation $\frac{\partial E}{\partial x} = 0$ is:

$$s(y - y_c)(\ell^2 - L^2) + (x - x_c)(s^2L^2 + \ell^2) = 0$$

or:

$$x - x_c = \frac{s(L^2 - \ell^2)}{s^2L^2 + \ell^2}(y - y_c),$$

which determines a line that intersects the ellipse in the two points. When we substitute $x - x_c$ from the equation of this line in the equation of the ellipse, we obtain the lower and upper bounds of the bounding box:

$$Y_1 = y_c - \sqrt{\frac{s^2L^2 + \ell^2}{1+s^2}},$$

and:

$$Y_2 = y_c + \sqrt{\frac{s^2L^2 + \ell^2}{1+s^2}}.$$

When we set $\frac{\partial E}{\partial y} = 0$, the perpendicular to the ellipse is in the direction of the x-axis. The equation $\frac{\partial E}{\partial y} = 0$ is:

$$(y - y_c)(L^2 + s^2\ell^2) + (x - x_c)s(\ell^2 - L^2) = 0$$

or:

$$y - y_c = \frac{s(L^2 - \ell^2)}{s^2\ell^2 + L^2}(x - x_c),$$

which determines a line that intersects the ellipse in the two points. When we substitute $y - y_c$ from the equation of this line in the equation of the ellipse, we obtain the left and right bounds of the bounding box:

$$X_1 = x_c - \sqrt{\frac{s^2\ell^2 + L^2}{1 + s^2}},$$

and:

$$X_2 = x_c + \sqrt{\frac{s^2\ell^2 + L^2}{1 + s^2}}.$$

This completes the proof.

References

1. Giannotti, F.; Pedreschi, D. *Mobility, Data Mining and Privacy—Geographic Knowledge Discovery*; Springer: Berlin, Germany, 2008.
2. White, C.E.; Bernstein, D.; Kornhauser, A.L. Some map matching algorithms for personal navigation assistants. *Transp. Res. Part C Emerg. Technol.* **2000**, *8*, 91–108.
3. Güting, R.H.; Schneider, M. *Moving Objects Databases*; Morgan Kaufmann: San Francisco, CA, USA, 2005.
4. Yen, J.Y. Finding the lengths of all shortest paths in N-node nonnegative-distance complete networks using $\frac{1}{2}N^3$ additions and N^3 comparisons. *J. ACM* **1972**, *19*, 423–424.
5. Hashemi, M.; Karimi, H.A. A critical review of real-time map-matching algorithms: Current issues and future directions. *Comput. Environ. Urban Syst.* **2014**, *48*, 153–165.
6. Bernstein, D.; Kornhauser, A.L. *An Introduction to Map Matching for Personal Navigation Assistans*; Technical Report; Transportation Research Board: Washington, DC, USA, 1998.
7. Eddy, S.R. What is a hidden Markov model? *Nat. Biotechnol.* **2004**, *22*, 1315–1316.
8. Newson, P.; Krumm, J. Hidden Markov map matching through noise and sparseness. In Proceedings of the 17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, New York, NY, USA, 4–6 November 2009.
9. Krumm, J.; Letchner, J.; Horvitz, E. Map matching with travel time constraints. In Proceedings of the Society of Automotive Engineers (SAE) 2007 World Congress, Detroit, MI, USA, 3–6 April 2007.
10. Greenfeld, J.S. Matching GPS observations to locations on a digital map. In Proceedings of the Transportation Research Board 81st Annual Meeting, Washington, DC, USA, 13–17 January 2002.
11. Brakatsoulas, S.; Pfoser, D.; Salas, R.; Wenk, C. On map-matching vehicle tracking data. In Proceedings of the 31st International Conference on Very Large Data Bases, Toronto, ON, Canada, 31 August–3 September 2005.
12. Wenk, C.; Salas, R.; Pfoser, D. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In Proceedings of the 18th International Conference on Scientific and Statistical Database Management, Vienna, Austria, 3–5 July 2006.
13. Kalman, R.E. A new approach to linear filtering and prediction problems. *Trans. ASME J. Basic Eng.* **1960**, *82*, 35–45.
14. Quddus, M.A.; Zhao, L.; Ochieng, W.Y.; Noland, R.B. An extended Kalman Filter algorithm for integrating GPS and low-cost dead reckoning system data for vehicle performance and emissions monitoring. *J. Navig.* **2003**, *56*, 257–275.
15. Li, L.; Quddus, M.; Zhao, L. High accuracy tightly-coupled integrity monitorin algorithm for map matching. *Transp. Res. Part C Emerg. Technol.* **2013**, *36*, 13–26.
16. Quddus, M.A.; Ochieng, W.Y.; Noland, R.B. Map matching in complex urban road networks. *Braz. Jm Cartogr. Revista Brasil. Cartogr.* **2003**, *55*, 1–18.
17. Zadeh, L.A. Fuzzy sets. *Inf. Control* **1965**, *8*, 338–353.
18. Zhao, Y. *Vehicle Location and Navigation Systems: Intelligent Transportation Systems*; Navtech Seminars and GPS Supply: Alexandria, VA, USA, 1997.

19. Syed, S.; Cannon, M.E. Fuzzy logic based map matching algorithm for vehicle navigation system in urban canyons. In Proceedings of the 2004 National Technical Meeting of the Institute of Navigation, Monterey, CA, USA, 26–28 January 2004.
20. Quddus, M.A. High Integrity Map Matching Algorithms for Advanced Transport Telematics Applications. Ph.D. Thesis, Imperial College, London, UK, 2006.
21. Lou, Y.; Zhang, C.; Zheng, Y.; Xie, X.; Wang, W.; Huang, Y. Map-matching for low-sampling-rate GPS trajectories. In Proceedings of the 17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, Seattle, WA, USA, 4–6 November 2009.
22. Li, Y.; Huang, Q.; Kerber, M.; Zhang, L.; Guibas, L. Large-scale joint map matching of GPS traces. In Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, New York, NY, USA, 5–8 November 2013.
23. Tang, J.; Song, Y.; Miller, H.J.; Zhou, X. Estimating the most likely space–time paths, dwell times and path uncertainties from vehicle trajectory data: A time geographic method. *Transp. Res. Part C Emerg. Technol.* **2016**, *66*, 176–194.
24. Chen, B.Y.; Yuan, H.; Li, Q.; Lam, W.H.K.; Shaw, S.; Yan, K. Map-matching algorithm for large-scale low-frequency floating car data. *Int. J. Geogr. Inf. Sci.* **2014**, *28*, 22–38.
25. Wang, W.; Jin, J.; Ran, B.; Guo, X. Large-scale freeway network traffic monitoring: A map-matching algorithm based on low-logging frequency GPS probe data. *J. Intell. Transp. Syst.* **2011**, *15*, 63–74.
26. Miwa, T.; Kiuchi, D.; Yamamoto, T.; Morikawa, T. Development of map matching algorithm for low frequency probe data. *Transp. Res. Part C Emerg. Technol.* **2016**, *22*, 132–145.
27. Rahmani, M.; Koutsopoulos, H. Path inference from sparse floating car data for urban networks. *Transp. Res. Part C Emerg. Technol.* **2013**, *30*, 41–54.
28. Hunter, T.; Abbeel, P.; Bayen, A.M. The path inference filter: Model-based low-latency map matching of probe vehicle data. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 507–529.
29. Zeng, Z.; Zhang, T.; Li, Q.; Wu, Z.; Zou, H.; Gao, C. Curvedness feature constrained map matching for low-frequency probe vehicle data. *Int. J. Geogr. Inf. Sci.* **2016**, *30*, 660–690.
30. Pfoser, D.; Jensen, C.S. Capturing the uncertainty of moving-object representations. In *Lecture Notes in Computer Science*; Güting, R.H., Papadias, D., Lochovsky, F.H., Eds.; Springer: Berlin, Germany, 1999; pp. 111–132.
31. Egenhofer, M.J. Approximation of geospatial lifelines. In *SpadaGIS, Workshop on Spatial Data and Geographic Information Systems*; Bertino, E.; Floriani, L.D., Eds.; University of Genova: Genova, Italy, 2003.
32. Hornsby, K.; Egenhofer, M.J. Modeling moving objects over multiple granularities. *Ann. Math. Artif. Intell.* **2002**, *36*, 177–194.
33. Wolfson, O. Moving objects information management: The database challenge. In *Lecture Notes in Computer Science*; Halevy, A.Y., Gal, A., Eds.; Springer: Berlin, Germany, 2002; pp. 75–89.
34. Hägerstrand, T. What about people in regional science? *Papers Reg. Sci. Assoc.* 1970, *24*, 7–21.
35. Miller, H.J. A measurement theory for time geography. *Geogr.l Anal.* **2005**, *37*, 17–45.
36. Othman, W. Uncertainty Management in Trajectory Databases. Ph.D. Thesis, Hasselt University, Hasselt, Belgium, 2009.
37. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107.
38. Dijkstra, E.W. A nnte on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271.
39. Ghys, K.; Kuijpers, B.; Moelans, B.; Othman, W.; Vangoidsenhoven, D.; Vaisman, A.A. Map matching and uncertainty: An algorithm and real-world experiments. In Proceedings of the 17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, Seattle, DC, USA, 4–6 November 2009.