A Force-Directed Approach for Offline GPS Trajectory Map Matching

Efstratios Rappos
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud
University of Applied Sciences of Western Switzerland (HES-SO)
Yverdon-les-Bains, Switzerland
efstratios.rappos@heig-vd.ch

Stephan Robert
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud
University of Applied Sciences of Western Switzerland (HES-SO)
Yverdon-les-Bains, Switzerland
stephan.robert@heig-vd.ch

Philippe Cudré-Mauroux
eXascale Infolab
University of Fribourg
Fribourg, Switzerland
pcm@unifr.ch

ABSTRACT

We present a novel algorithm to match GPS trajectories onto maps offline (in batch mode) using techniques borrowed from the field of force-directed graph drawing. We consider a simulated physical system where each GPS trajectory is attracted or repelled by the underlying road network via electrical-like forces. We let the system evolve under the action of these physical forces such that individual trajectories are attracted towards candidate roads to obtain a map matching path. Our approach has several advantages compared to traditional, routing-based, algorithms for map matching, including the ability to account for noise and to avoid large detours due to outliers in the data whilst taking into account the underlying topological restrictions (such as one-way roads). Our empirical evaluation using real GPS traces shows that our method produces better map matching results compared to alternative offline map matching algorithms on average, especially for routes in dense, urban areas.

CCS CONCEPTS

• **Information systems** → **Geographic information systems**;
• **Theory of computation** → Computational geometry;

KEYWORDS

Map matching, force-directed algorithms, GPS trajectory, road map, offline routing

ACM Reference Format:
Efstratios Rappos, Stephan Robert, and Philippe Cudré-Mauroux. 2018. A Force-Directed Approach for Offline GPS Trajectory Map Matching. In *26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18), November 6–9, 2018, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3274895.3274919

SIGSPATIAL '18, November 6–9, 2018, Seattle, WA, USA

ACM ISBN 978-1-4503-5889-7/18/11…$15.00
https://doi.org/10.1145/3274895.3274919

1 INTRODUCTION

Map matching is the process of mapping a geospatial trajectory obtained from a GPS receiver onto a given road network. As the coordinates obtained from these devices are not always precise, in dense road networks the task of matching these onto a real map is not trivial. Several candidate roads may exist in close proximity and a map matching algorithm must ensure that the resulting path on the road network is plausible and that physical constraints (e.g., one-way streets, obstacles) are respected.

Map matching has been studied for over a decade [25] and a large collection of algorithms exist with varying degrees of complexity and accuracy. Existing algorithms can be divided into two broad categories: i) online or real-time algorithms, where the algorithm has to determine the likely position on a map given the history of previous points, for example on a vehicle equipped with a GPS navigation device, and ii) offline algorithms, where the entire trajectory is known in advance and the algorithm has to adjust the trajectory points a posteriori such that they represent on a map the likely route taken by the vehicle.

The present article considers offline map matching. This problem has received less focus than its real-time counterpart as it is not useful for real-time navigation. However, in many applications, such as logistics and supply chain management, the analysis of vehicle trajectories is done a posteriori once the vehicles have returned to the depot, where a map matching algorithm is used to correct measurement errors by the GPS receivers and produce a trajectory that lies completely on a real map network. One of the main differences with the online case is the inclusion of the entire trajectory in the analysis, which provides additional information on the likely route taken.

We propose a novel approach which borrows methods from force-directed graph design to direct the map matching strategy, improving on the existing map matching literature. The algorithms used in force-directed graph design aim to produce an elegant visualization of a graph topology (vertices and edges) on a plane [5, 32]. To achieve this, each vertex is assumed to repel each other whilst each edge can expand or contract freely, and these forces are modeled using concepts from physics, such as electrical repulsion or spring forces. The system is then simulated on a computer where the edges and vertices are allowed to move according to physical laws and, after a number of iterations, a visually elegant layout of the graph is obtained. These approaches are presented in detail in Section 3.

In this article we use similar techniques to achieve high precision map-matching results. The road network of the map is assumed to exert a force field and every vertex of the trajectory is attracted to the field in such a way that after a number of iterations the trajectory is closely matched to, or 'snapped', onto the road network. Using this approach, the force exerted is linked to the distance between the trajectory and the road, so the trajectory will be preferentially attracted to nearby roads and, in addition, the direction of the force is linked to the angle between the road and the trajectory, enabling one-way streets to be correctly represented (if a trajectory travels in the opposite way near a one-way road, it results in a repelling force and this road is avoided). These two features produce an effective map matching algorithm. To our knowledge this is the first method which uses force-directed algorithms for map matching.

2 RELATED WORK ON MAP MATCHING METHODS

The aim of map matching is to convert, based on some known map data, a list of GPS points into a trajectory (series of roads or links) denoting the most likely route traveled by the vehicle or moving object. Over the past years, many map matching algorithms have been proposed in the literature, both for real-time and for offline map matching, which cover a number of different types of applications and input data. A comprehensive review of over 30 map matching algorithms can be found in [25]. The authors classify the analytical approaches used in the algorithms into 'geometrical', which use proximity-based methods, 'topological', which use the notions of connectivity between the links (one-way roads, connectivity and reachability information), 'probabilistic', which further use information about the quality or accuracy of the GPS signal (typically obtained from the GPS sensor), and 'advanced', which use more specific methods such as Kalman filters, hidden Markov chains, timing information (e.g., to predict the exiting from a tunnel) and other application-specific approximation techniques. Typically the underlying map network is known, however some researchers [4, 15, 19, 34] have developed approximation techniques to generate an unknown underlying map or to perform map matching without reference to a known map topology by observing the clustering of trajectories.

Recently, improvements on these methods have been proposed, such as an efficient buffer topological algorithm to detect bicycle paths in Bologna [26], or a score-based matching for car trajectories in Zurich [21]. The ACM SIGSPATIAL 2012 competition [1, 13, 18, 28, 30, 31, 35] requested participants to determine a fast map matching algorithm for use in real-time systems; the focus of the competition was on algorithm speed since the competition used only ten vehicle trajectories and the provided instances were relatively easy to solve (good quality GPS points on a not very dense road network). The authors of [17] use a geometric distance measure to determine the nearby roads and then apply a Dijkstra-based algorithm to select those roads which satisfy the topological restrictions of the map. In a different direction, [14] select their road segments using an optimization method which takes advantage of cases where many users drove along a similar route, similar to trajectory clustering methods.

More complex approaches can also be found in the literature, the most noteworthy of which is the voting-based map matching algorithm [38] where the most likely path is determined by the relative mutual influence between pairs of points, taking into account at the same time the temporal information (timestamps) of the GPS points.

Among the probabilistic approaches, a method proposed in [2] ranks all topologically possible trajectories based on a calculated probability, which is a generalization of earlier hidden Markov chain or Viterbi map matching methods. This approach is extended in [23] where the number of turns in the resulting trajectory is taken into consideration and optimized using inverse reinforced learning. A similar comprehensive search method is used in [37] where a heuristic search algorithm is used to find and score each possible trajectory, and in [33] for real-time map matching.

Another interesting approach is the one presented in [27], where the authors do not really perform map-matching but aim to 'correct' GPS trajectories by interpolation so that the resulting traces are closer to the real route taken, using a clustering algorithm which compares trajectories between them. This concept is similar to our proposed force-directed algorithm where we also 'correct' the raw GPS points, but we do so by considering the interaction of a particular trajectory with the underlying road network instead of comparing trajectories between them.

The most commonly used approaches for map matching which combine both 'geometrical' and 'topological' methods are routing-based methods. This means that the map matching problem is converted into a routing problem, where in its simplest version the trajectory is divided in smaller segments, the endpoints of which are then matched onto a road (for example, moving each endpoint to the nearest point on the road), and the intermediate points of the segment are replaced by a routing calculation of the shortest possible route from one endpoint to the other, taking into account the road layout. This approach produces good results as it ensures that the produced route is close to the original points and that the route will be plausible, in the sense that it is guaranteed to lie on an existing road and all the topological restrictions will be satisfied. The most popular implementations of route-based algorithms for map matching are GraphHopper [6], and MapBox [20] both of which use shortest-distance routing directed by weights derived from the GPS trajectory to find a match.

However, routing based methods are not fool-proof. They operate under the assumption that the driver who produced the GPS trace was driving on a shortest-distance fashion between periodically-sampled segments of the route, so short, circular loops within each segment (taken for example by taxis) will not be matched correctly.

Our proposed method takes these routing-based methods one step further, adding an element of the 'probabilistic' map matching techniques: we use a force-directed algorithm to adjust, or correct, the obtained raw GPS points before applying a routing-based method, resulting in a more accurate match.

3 COMPLEXITY AND EVALUATION OF MAP MATCHING ALGORITHMS

The nature of the map matching problem presents some unique challenges. First of all, the difficulty of the task can vary significantly: if the trajectories obtained correspond to a rural setting (e.g., on an isolated highway) the task can be very easy or trivial, as there may only be one possible candidate road for the path taken. Conversely in a city center setting, the challenge is much harder as the road network is more dense. Similarly, the frequency of sampling of the GPS points is important, as recording one point every second will make map matching easier than, say, recording every minute. Finally, the quality of the GPS signal, the GPS receiver used and the underlying map are also important as a good quality trajectory will result in points that are closer to coordinates of the real road, making the task much easier.

Although the density of the underlying map is a key factor in terms of determining the difficulty of the map matching problem and therefore the performance of a map matching algorithm, other elements that influence this are the quality of the GPS data and the mode of transport, which determines the speed of the vehicle. Bicycle and pedestrian trajectories are easier to match for a given sampling frequency as the object does not move much between successive GPS points.

In terms of the underlying fixed road network map, which is necessary for most map matching algorithms, researchers tend to use data from the freely available OpenStreetMap service [22], which has a good coverage of road networks for most cities around the world.

In order to evaluate the performance of map matching algorithms, the obvious step is to compare the produced trajectory with the 'ground truth', i.e., the actual trajectory taken by the vehicle. However, this approach can only be used in limited circumstances, as the ground truth is typically not available in most cases of large scale data collections (as it requires navigating along a predefined route or significant manual input to record the precise route taken). Some researchers derive the ground truth by manually matching some trajectories by sight, for example in [14] experienced human drivers were asked to trace, based on their experience, the 'ground truth' of a random subset of 100 trajectories among their dataset of Beijing taxi traces. Other researchers [36] create their own datasets by driving along a very small number of predefined routes (four). These approaches have many drawbacks, namely the fact that the chosen routes are defined in advance by the researchers, that human discretion is required to ascertain the route taken and that a large number of trajectories cannot be matched by hand.

In order to mitigate these limitations and use trajectories for which no ground truth exists, some authors [17, 21, 24] propose distance-based metrics based on minimizing the distance between the GPS trajectory and the route produced by the algorithm. This suggests that map matches that are close to the original points are considered to be more accurate than those which are farther away. Some alternatives for the evaluation of map matching algorithms in the absence of ground truth include the comparison of the length of the original trajectory compared the the length of the matched route [26]. Finally, an interesting approach [24], although with limited practical use, is to collect two sets of GPS data, one of low quality used as the input trace and a second trace of high quality data used as a proxy for the ground truth.

4 FORCE-DIRECTED GRAPH DRAWING METHODS

Graph visualization is a well-researched field, as graph structures appear in many areas and graph drawing on a 2-dimensional plane quickly becomes challenging as the size of the graph increases. Recently, a number of approaches in this area have focused on 'force-directed' methods which can automatically 'draw' large graphs on a plane [5, 7, 8, 11, 29, 32]. A comprehensive overview of such algorithms is presented in [12] and some interesting, more recent variations appear in [9, 10, 16]. In these methods, a directed or undirected graph is modeled as a system of particles with forces acting between them and a compelling visual result is achieved when the particles are placed in such a way as to achieve a force equilibrium.

An example of the input and output of a force-directed graph drawing algorithm from [5] is shown in Figure 1. One can see how the forces between the edges in 2-dimensions force the graph to spread out into a symmetrical equivalent representation.

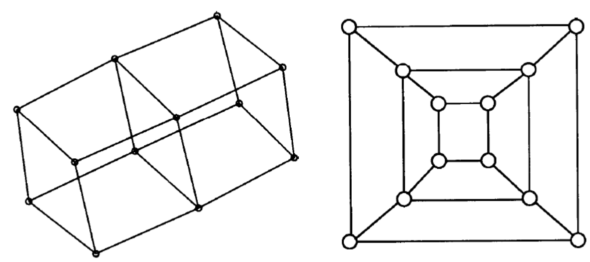

Figure 1: Example input and output graphs using a force-directed graph drawing algorithm from [5]

In general, all force-directed graph drawing algorithms consider repelling forces between non-adjacent vertices that are inversely proportional to the distance d between the vertices (c/d), or to the square of the distance (c/d^2) in order to reduce the strength of the force between distant vertices and yield a faster convergence. The edges are modeled as spring forces that can both expand and contract around an 'ideal' length, although pure spring forces (proportional to the displacement of the spring) are considered too strong and are usually replaced by the logarithm of the displacement ($c_1 \log(d/c_2)$), where $d = c_2$ is the desired 'ideal' length between vertices (often defined as the square root of the total drawing area divided by the number of vertices to ensure an even spread on the drawing area).

As the computational simulation of a system of particles under the laws of physics is computationally intensive, an approximation is always used, where the force applied to each vertex is calculated in turn, and then the vertex is displaced by a small amount in the direction of the combined net force before the process iterates. Other refinements that have been proposed in the literature include modified force formulas, computation of various parameters based on some further graph characteristics (e.g., the diameter of the graph) and the introduction of a cooling coefficient (based on the field of simulated annealing in optimization) where the displacement of the vertices gradually becomes smaller to ensure that once a good configuration is found no major modifications to the layout occur.

In the next section we present an algorithm that uses the techniques of force-directed graph drawing outlined in this section to perform a map matching of GPS trajectories onto a map.

5 A FORCE-DIRECTED MAP MATCHING ALGORITHM

We consider GPS trajectories defined by N points $P_1, P_2, \ldots, P_N$ where a point is defined by its position in space and time $P_i = (lat_i, lon_i, t_i)$. We do not use information about the quality of the signal or receiver (accuracy and precision of the GPS receiver) as this is typically not included in many GPS production systems. The fixed road network or map is represented by a directed graph $G = (V, E)$ of intersection points and straight line segments (links) between them, which are obtained from the public-domain mapping provider OpenStreetMap [22].

The key idea of force-directed map matching is to consider an 'electrical current' that passes through each edge E of the road network and results in an attractive or repulsive force on each point of a given trajectory. We set the magnitude of this force F_e as follows:

- inversely proportional to the distance d between the point P and the road edge E,
- proportional to the cosine of the angle θ between the road edge E and the trajectory at P,
- proportional to the length l of E.

We explain the rationale behind each choice in turn: the first point specifies that trajectory points should be attracted more strongly by nearby roads, which is sensible. The second point relates the force to the angle between the trajectory and the road. The force should be at its maximum when the trajectory and the road are parallel to each other, it should reduce to zero when the two are perpendicular to each other and then become negative (repulsive) when the trajectory and the road point in opposite directions. The last requirement is necessary as the edges in the underlying road network are not of equal length. Without this constraint, if a road on the map was split into two edges, it would result in doubling the force on point P. Adjusting for the length of each edge E avoids the trajectory being pulled towards areas with high road density (many small roads).

The direction of the force is taken to be either: (i) perpendicularly towards the edge E or (ii) towards the midpoint of the edge E. Note that unlike graph drawing algorithms, no forces are operating between individual vertices except for neighboring vertices as described below.

We also assume that spring forces apply on each edge (P_{i-1}, P_i) of a given trajectory. These were set at the same standard log-distance formula as used in graph drawing:

$$F_s = c_1 \log(d/c_2)$$

where c_1, c_2 are constants and d is the length of the spring. We have set the natural length d to be equal to the length of the trajectory segment (P_{i-1}, P_i) as we assume that the distance between points on the true trajectory will be similar to the observed distance. The forces between the points can be attractive or repulsive and are applied in the direction of the edge (P_{i-1}, P_i).

Once all the forces are calculated for each point i, each point is moved in turn by a distance proportional to the net force

$$\Delta \mathbf{x}_i = c_4 \left(\sum_E \mathbf{F}_e + \sum_{P_{i-1}, P_{i+1}} \mathbf{F}_s \right)$$

The key parameters of our algorithm are summarized in Table 1. We experimented with variations of the distance and force formulas as suggested in the force-directed graph drawing literature until we found an ideal combination for the strengths of the electrical and spring forces. The values that we used for the final algorithm are denoted by a star '*' in the table. Regarding our choices, we can comment that an electrical force proportional to the road edge length l is too strong for quick convergence and replaced it with $\sqrt{l}$. Furthermore, the repulsive forces when the road and trajectory are pointing in opposite directions had to be significantly reduced to ensure that the trajectory is still attracted to nearby roads with the correct orientation. We also note that because of the sharp decrease in the magnitude of the electrical forces with the distance as well as for computational efficiency, we only include edges which are within 100 meters from the current point in the calculation of the electrical forces.

A pseudocode of our force-directed algorithm is given in Figure 2. Once the trajectory is read, we use the force-directed method to attract the trajectory points towards the roads on the map. After a number of iterations, the trajectory will be close to a plausible map match. As a final step in order to convert the points into road segments, it is necessary to apply an algorithm to place the obtained points exactly on a map road. This algorithm can simply be to place each point to the nearest road segment that has the correct alignment (in the case of one-way roads), or alternatively, it could be an implementation of a traditional route-based algorithm. In our implementation and numerical experimentation we chose the latter method for a number of reasons that are explained in the following section.

```
Read GPS points of trajectory T
for t:=1 to iterations do
    for all points of the trajectory P_i ∈ T do
        Calculate total force F_e, F_s on the point P_i
        Update position Δx of point P_i
    end for
end for
Finalize position by placing the modified points exactly on the
map
```

Figure 2: Pseudocode of the force-directed map matching algorithm

6 EXPERIMENTAL EVALUATION

This section presents the results of an experimental evaluation of our force-directed map matching algorithm applied on a large

Table 1: Summary of the algorithm parameter values tested

variable	selection used	alternatives tested
d	*	(i) perpendicular distance d_p from P to line defined by edge AB (ii) distance from P to segment AB (d_s): – perpendicular distance, if projection of P lines inside segment AB – minimum distance to endpoint of AB, otherwise (iii) distance d_m of P to midpoint of AB
θ	*	(i) $\cos(\angle E, P_{i-1} - P_{i+1})$, (ii) $(\cos(\angle E, P_{i-1} - P_i) + \cos(\angle E, P_i - P_{i+1}))\,/2$, (average of cosines) (iii) $\cos\left((\angle E, P_{i-1} - P_i + \angle E, P_i - P_{i+1})/2\right)$ (cosine of average)
edge repulsion	*	(i) if $\cos(\theta) < 0$ replace $\cos(\theta)$ with $\cos(\theta)/2$ (ii) if $\cos(\theta) < 0$ replace $\cos(\theta)$ with -0.001 (iii) if $\cos(\theta) < 0$ replace $\cos(\theta)$ with 0
force direction	*	(i) perpendicular from P towards the edge E (ii) direction from P towards the midpoint of E
force value F_e	*	$F_e = c\sqrt{l}\cos(\theta)/d$ $F_e = c\sqrt{l}\cos(\theta)/d^2$ $F_e = c\sqrt{l}\cos(\theta)d_p/d^2$ $F_e = c\sqrt{l}\cos(\theta)/(d \cdot d_m)$
F_s	*	$F_s = c_1 \log(d/c_2)$

number of GPS trajectories where we compare its performance with other state-of-the art map matching algorithms. We consider a large dataset of taxi trajectories created in 2014 in Rome [3]. This data consists of timestamped latitude/longitude data corresponding to nearly 500,000 km of driving carried out by taxi drivers equipped with a GPS tracking device on a tablet computer.

We chose this dataset for specific reasons: The road network in Rome is very dense, not grid-like with many short and irregularly shaped roads, many obstacles and one-way streets. (The average road segment length in our Rome map data was 29 meters.) This means that the map matching problem on this layout is more complex than a similar problem on city with a grid layout and large blocks. Moreover, this dataset records a trajectory as one GPS point every 15 seconds (so pretty infrequently) which increases the trajectory ambiguity, and the need for a proper trajectory correction.

In line with most articles in the literature, we obtained the underlying map road network for Rome from OpenStreetMap [22] and filtered the road network to include only roads that are open to car traffic. The road network (corresponding to the assumed force field passing through each road segment) that was obtained is shown in Figure 3, while Figure 4 shows the distribution of the length of the road segments in the same area.

In the implementation of our force-directed algorithm, we used the same route-based algorithm as Graphhopper for the last step to convert the final positions of the modified points into road segments. This choice was made for three reasons: first, we had to produce a path that can be correctly identified using its underlying OpenStreetMap name in order to compare it to the map matching produced by the routing-based algorithm; using the Graphhopper tool to do so is the obvious solution. Secondly, using Graphhopper as the last step in our method also demonstrates the superiority of our method compared to route-based methods, since if we produce a better map matching result this cannot be due to particularities or limitations in the Graphhopper implementation, as these would be present in our results too. The third reason was a practical one: with this step in place we do not need to wait until the force-directed algorithm converges (sometimes slowly, depending on the choice of the algorithm parameters) towards the final matched path; instead we can terminate our algorithm after a number of iterations when a good enough approximate match is found, and then post-process this result to obtain a feasible path. In other words, we compare the difference of performing a map match by a routing algorithm (such as Graphhopper) directly on the input data, against performing the same algorithm on data points that have been first displaced towards specific roads by our force-directed algorithm.

The GPS data was cleaned and divided into distinct trajectories in the same way as the authors in [3]: when an anomaly is detected (defined as a speed of over 50 km/h), we look at the total duration of the anomaly. For anomalies under 42 sec we simply delete the incorrect GPS points; for anomalies between 42 sec and 8 min we delete the points and replace with intermediate points based on linear interpolation; for anomalies over 8 min or consecutive points over 8 min apart we assume that this is due to a break implying the end of a trajectory and the start of a new one. We further removed trajectories with fewer than 10 points or totaling less than 8 mins as they are too short for useful map matching. Finally, we also excluded a small number of trajectories which lie outside our chosen reference grid of latitude (41.8001, 41.9859) and longitude (12.382189, 12.608782). This approach resulted in a total of 18,111 trajectories containing over 16 million points (1.3GB data) and a total distance of 467,875 km in 37,517 hours.

Figure 3: The road network of central Rome corresponding to the electrical force field

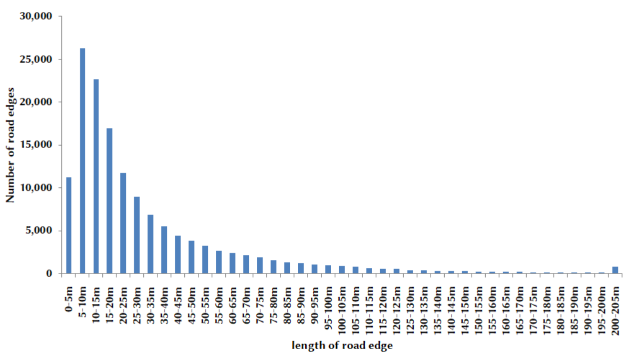

Figure 4: Distribution of OpenStreetMap road edge lengths for Rome

In line with [26] we also excluded from the comparative evaluation trajectories totaling less than 300 meters in length and those for which the length index (defined in [26] as the ratio of the total length of the calculated trajectory divided by the total length of the original trajectory) is outside of the range [0.8, 1.2]. Our investigation showed that a large difference in trajectory length is due to undocumented particularities of the map network, for example around the touristic Piazza di Spagna area which in reality can be driven through by taxis but which is recorded on the map we used as a pedestrian-only area forcing any map-matching algorithm to take a long detour. Removing these trajectories resulted in a total of 11,154 trajectories, containing 7.1 million points and a total distance of 199,398 km driven over 16,753 hours. Figure 5 shows a histogram of the distribution of the lengths of the trajectories used in this analysis.

The implementation of our algorithm was done in Java on a machine with 16GB of RAM and four CPU cores. We compared our

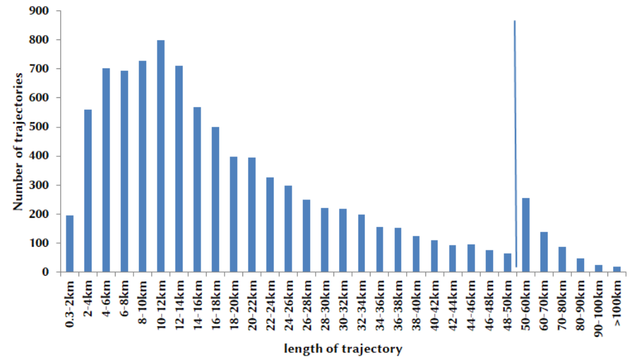

Figure 5: Distribution of trajectory lengths

algorithm to the popular routing-based map matching algorithm Graphhopper.

An example of the algorithm output is shown in Figure 6. The original data is shown in blue and the routing-based map match is shown in purple. When we carry out the force-directed map matching algorithm the trajectory is modified to the green trajectory and the resulting map-match is shown in black. It is quite evident that, even using one iteration, the force-directed algorithm produces a better matching route.

A second example depicting a trajectory with a loop is shown in Figure 7. We note that the routing-based map matching algorithm fails to detect the apparent loop in the trajectory, which is successfully identified once the trajectory points are modified under our force-directed algorithm.

7 RESULTS AND DISCUSSION

In order to evaluate the results of the proposed map matching algorithm, we used two metrics found in the literature for the comparative evaluation of map matching algorithms in the absence of the ground truth. It is worth noting that all evaluation metrics without ground truth will have some limitations since there is no fool-proof method of comparing a map match produced by one algorithm with one produced by another algorithm. Nonetheless, these metrics measure elements that can be reasonable deemed to feature in bad matches, such as the path being too far away or its length being too different to the original trajectory, and therefore can be used to assess the quality of map matching.

We first used the method proposed by [26] to evaluate map matches for bicycle paths in Bologna: we calculate the length index I_L, which is defined by dividing the length of the matched route R by the line-interpolated length of the GPS trace:

$$I_L = \frac{\sum_R L_a}{\sum_i (P_{i-1}, P_i)}$$

and assume that the closer this index is to 1 the better the match. In other words, it is assumed that a good map matching algorithm will produce a path with length similar to the length obtained from the GPS points. Although this is not necessarily true, it provides a good approximation by penalizing algorithms which produce paths are too short or too long for the trajectory, for example paths containing a lot of detours or that omit loops of the trajectory.

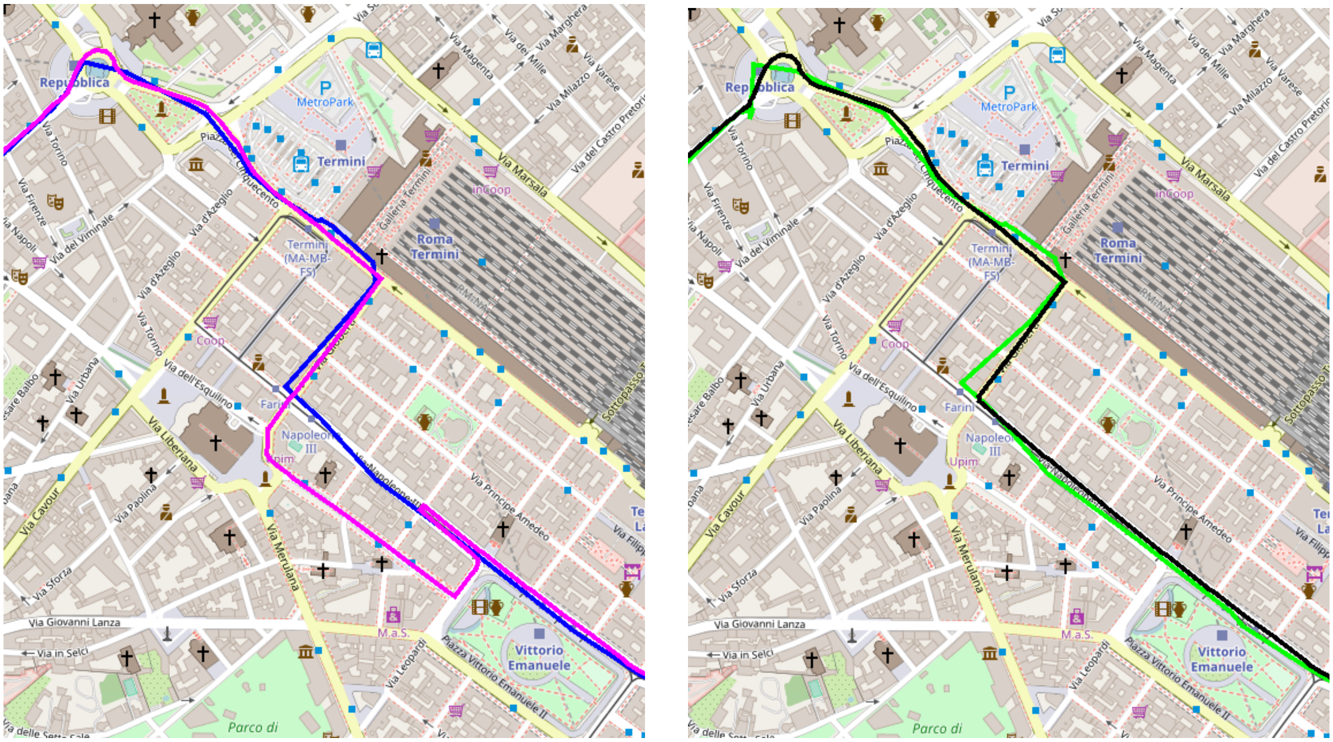

Figure 6: Using the routing-based map matching algorithm, the GPS trajectory (blue) is matched to the purple path. Under the proposed force-directed algorithm, the path is perturbed (green, after one iteration) yielding eventually a more plausible route (black)

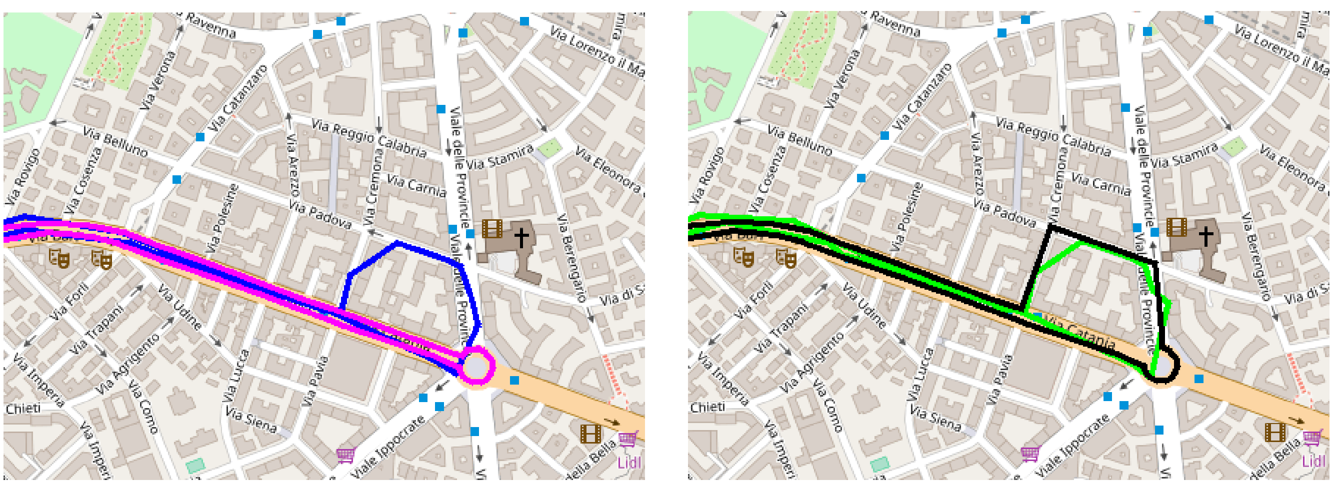

Figure 7: An example of map matching for a trajectory containing a loop

The results of using this metric are shown in Table 2. We note that the force-directed algorithm results in an index which is closer to 1 and therefore produces a better match than the route-based map matching method. A distribution of the index according to the length of the original trajectory and the number of points in the trajectory are shown in Figure 8 and Figure 9 respectively. We can observe that the force-directed algorithm performs consistently better, except for very short trajectories. There is little difference in the distribution of the length index according to the number of iterations used in the algorithm.

The second evaluation metric used is linked to the average absolute error of the calculated path compared to the original GPS points. This method has been used in [21]. For each GPS point of the original trajectory we define its distance to the matched path

Table 2: Comparison between routing-based and force-directed map matching

	route-based	force-directed algorithm, # of iterations				
	algorithm	1	5	10	15	20
Method 1: Length index	1.114	1.113	1.096	1.084	1.079	1.079
Method 2: Avg. error (meters)	18.34	18.29	15.34	14.50	14.25	14.21
Computational time (sec)	0.22	1.03	4.10	7.04	8.80	13.52

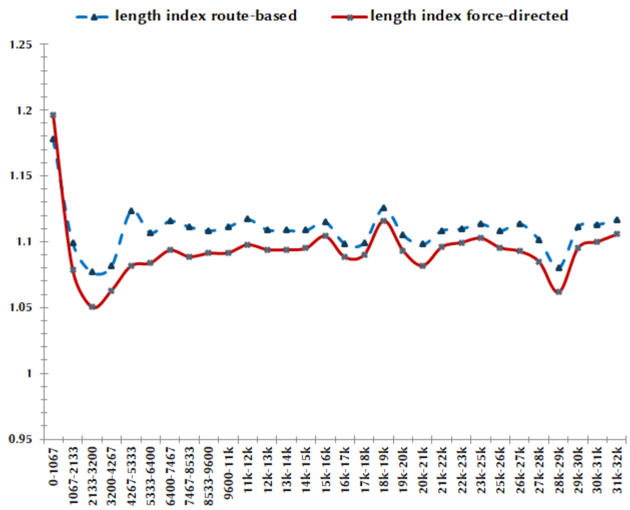

Figure 8: Distribution of the length index by the length of the trajectory

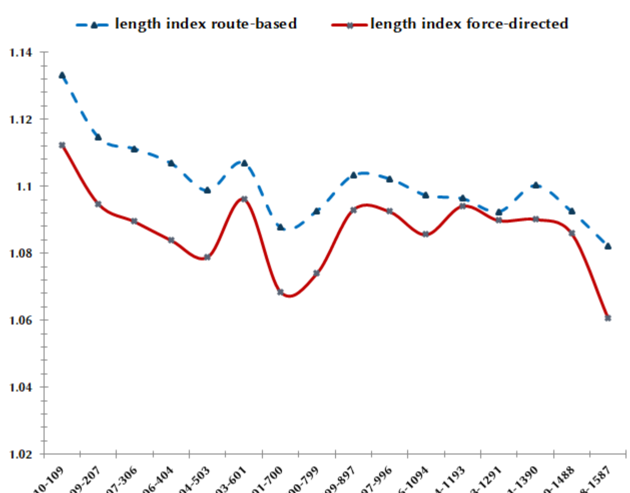

Figure 9: Distribution of the length index by the number of points in the trajectory

as the minimum of the distances of the point to the line segment of the matched path. The distance of a point to the segment is defined as the perpendicular distance if the projection of the point to the line segment lies between the endpoints of the segment, otherwise it is defined as the minimum distance to the endpoints:

$$d_s = d(P, AB) = \begin{cases} d(P, P'), & \text{if } P' \in [AB] \\ \min\{d(P, A), d(P, B)\}, & \text{otherwise.} \end{cases}$$

where P' is the projection of P on the line AB. The average error is then defined as the average distance of each GPS point to the matched path:

$$\text{Avg Error} = \frac{1}{N} \sum_i \min_{e \in E} d(P_i, M_e)$$

In other words, the average error can be considered as the average, along the trajectory, transverse distance between the trajectory and the matched route and a smaller error denotes a better match.

The results using this method are shown in Table 2 and the distribution of this metric by the length and the number of points of the trajectory are shown in Figure 10 and Figure 11. We note that the force-directed algorithm also produces better results using this evaluation method than traditional map matching, and the performance improves as the number of iterations of the force-directed algorithm increases. This trend continues until approximately 20 iterations, when the average error stabilizes and remains around 23% better than routing-based method on average. This suggests that a longer running time of the force-directed algorithm does not produce better results, and only a small number of iterations is needed to perturb the trajectory points sufficiently for a good, potentially optimal, match to be found.

It is also worth noting that in 12% of the trajectories both algorithms produced the same matching path, reflecting our observation that for several trajectories there can only be one or very few plausible routes and the task of finding a map match is easier.

In terms of computational time, Table 2 and Figure 12 show the average computational time taken by each of the two algorithms, measured in seconds of elapsed clock time. We note that while the routing based algorithm is able to transform one GPS trajectory into a sequence of roads in less than one second, the force-directed one takes significantly more time, on average 13.52 seconds and up to 17 seconds for the trajectories over 30km. This is because of the large number of interactions that have to be taken into account during the calculation of the forces between the trajectory and the road network. The computational time of the force-directed algorithm increases linearly with the number of points in the trajectory. The slight decrease in computational time for trajectories over 30km long is due to the small number of trajectories in this range and to

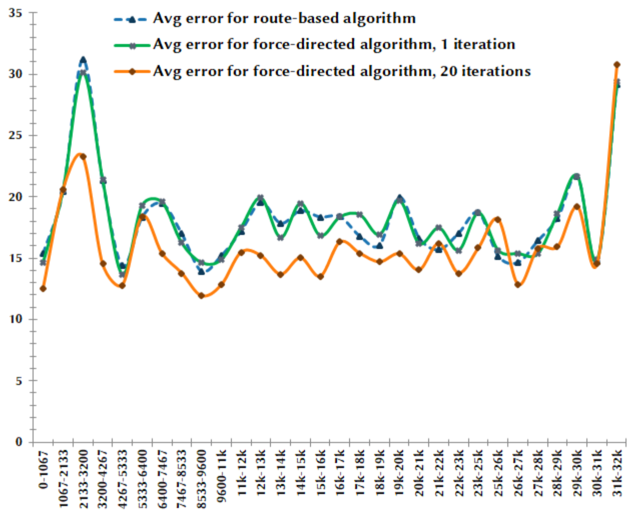

Figure 10: Distribution of the average error in meters by the length of the trajectory

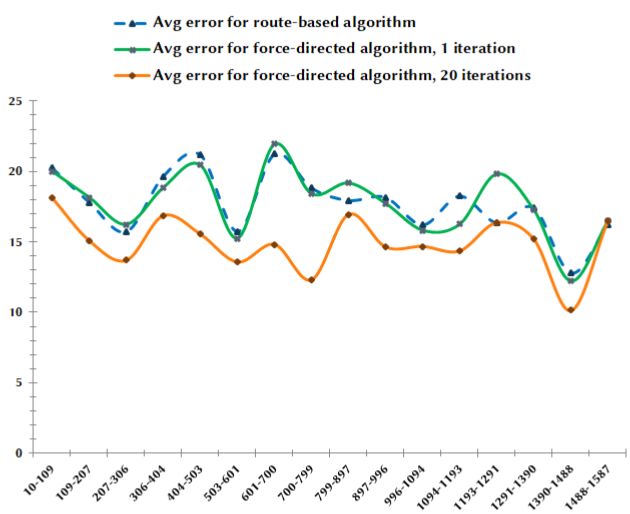

Figure 11: Distribution of the average error in meters by the number of points in the trajectory

the fact that many of these trajectories were on fast motorways, resulting in fewer GPS points than was typical for their length.

The increase in computational time to under 20 seconds per trajectory poses no practical limitations, since the algorithm is designed for offline processing of trajectories and is an acceptable price to pay if it results in more accurate road matches.

Under the two evaluation metrics considered the proposed algorithm performs better in terms of the quality of the produced path than the baseline routing-based map matching algorithm, at the expense of increased computational time, although as mentioned earlier, all evaluation metrics have limitations in in the absence of ground truth data.

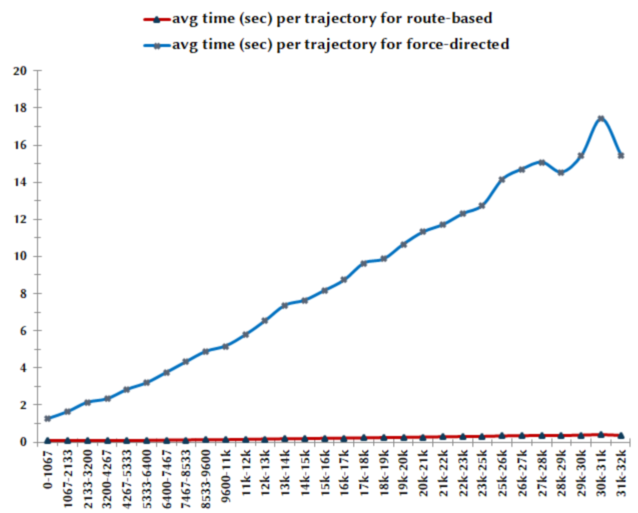

Figure 12: Average computational time taken by each algorithm by trajectory length

8 CONCLUSIONS AND FUTURE WORK

This paper presented a novel algorithm that can be used to match trajectories obtained from GPS receivers onto a known map network. The algorithm borrows techniques used in force-directed graph drawing in order to incrementally perturb the trajectory and make it converge onto a good, likely path whilst at the same time ensuring that topological limitations such as one-way streets are satisfied. The interactions between the trajectory points and the underlying road network are modeled by a physical system evolving under the influence of physical-like forces which were described in detail in this work.

Numerical experimentation using real trajectories in a dense, urban road network demonstrates that the proposed method produces better map matching paths than routing-based map matching alone, providing a framework for the use of force-directed algorithms in related domains such map construction through the clustering of multiple related trajectories and real-time map matching.

The future work in this direction includes the evaluation of the algorithm using new data, including datasets which contain the ground truth, and the development of more reliable metrics of evaluation, for example using a version of the Fréchet distance which can better measure the similarity of spatio-temporal trajectories. We are also working to further explore the optimal values of the parameters of the algorithm, such as the optimal number of iterations. Equally, a comparison of the performance of the proposed approach with other relevant implementations, in particular [2, 23, 35, 38] and commercial software [20], is under investigation. The performance of the algorithm in difficult constellations and uncommon layouts, such as fly-overs or roads separated vertically and multilane matching (which are very uncommon in the Rome dataset used for this paper) remains to be evaluated. Finally, the use of the GPS temporal information (timestamps) to determine some of the parameters of the algorithm has the potential to further improve the accuracy in the case of sparse data.

REFERENCES

[1] Mohamed Ali, John Krumm, Travis Rautman, and Ankur Teredesai. 2012. ACM SIGSPATIAL GIS Cup 2012. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*. ACM, New York, NY, USA, 597–600. https://doi.org/10.1145/2424321.2424426

[2] Michel Bierlaire, Jingmin Chen, and Jeffrey Newman. 2013. A probabilistic map matching method for smartphone GPS data. *Transportation Research Part C: Emerging Technologies* 26 (2013), 78–98. https://doi.org/10.1016/j.trc.2012.08.001

[3] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. 2014. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from http://crawdad.org/roma/taxi/20140717.

[4] Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristan, Rodrigo I. Silveira, Frank Staals, and Carola Wenk. 2017. Clustering Trajectories for Map Construction. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'17)*. ACM, New York, NY, USA, Article 14, 10 pages. https://doi.org/10.1145/3139958.3139964

[5] Thomas M. J. Fruchterman and Edward M. Reingold. 1991. Graph drawing by force-directed placement. *Software: Practice and Experience* 21, 11 (1991), 1129–1164. https://doi.org/10.1002/spe.4380211102

[6] Graphhopper. 2017. Map-matching based on graphhopper. https://github.com/graphhopper/map-matching

[7] Stefan Hachul and Michael Jünger. 2007. Large-graph layout algorithms at work: an experimental study. *Journal of Graph Algorithms and Applications* 11, 2 (2007), 345–369.

[8] David Harel and Yehuda Koren. 2002. A fast multi-scale method for drawing large graphs. *Journal of Graph Algorithms and Applications* 6, 3 (2002), 179–202.

[9] Danny Holten and Jarke J. van Wijk. 2009. Force-directed Edge Bundling for Graph Visualization. In *Proceedings of the 11th Eurographics / IEEE - VGTC Conference on Visualization (EuroVis'09)*. The Eurographs Association & John Wiley & Sons, Ltd., Chichester, UK, 983–998. https://doi.org/10.1111/j.1467-8659.2009.01450.x

[10] Yifan Hu. 2006. Efficient, High-Quality Force-Directed Graph Drawing. *The Mathematica Journal* 10, 1 (2006), 37–71.

[11] Tomihisa Kamada and Satoru Kawai. 1989. An algorithm for drawing general undirected graphs. *Inform. Process. Lett.* 31, 1 (1989), 7–15. https://doi.org/10.1016/0020-0190(89)90102-6

[12] Stephen G. Kobourov. 2013. Force-directed graph algorithms. In *Handbook of Graph Drawing and Visualization*, Roberto Tamassia (Ed.). Chapman & Hall/CRC, Boca Raton, FL, USA, Chapter 12, 383–408.

[13] Roy Levin, Elad Kravi, and Yaron Kanza. 2012. Concurrent and Robust Topological Map Matching. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*. ACM, New York, NY, USA, 617–620. https://doi.org/10.1145/2424321.2424431

[14] Yang Li, Qixing Huang, Michael Kerber, Lin Zhang, and Leonidas Guibas. 2013. Large-scale Joint Map Matching of GPS Traces. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'13)*. ACM, New York, NY, USA, 214–223. https://doi.org/10.1145/2525314.2525333

[15] Yang Li, Yangyan Li, Dimitrios Gunopulos, and Leonidas Guibas. 2016. Knowledge-based Trajectory Completion from Sparse GPS Samples. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '16)*. ACM, New York, NY, USA, Article 33, 10 pages. https://doi.org/10.1145/2996913.2996924

[16] Chun-Cheng Lin and Hsu-Chun Yen. 2012. A new force-directed graph drawing method based on edge-edge repulsion. *Journal of Visual Languages and Computing* 23 (2012), 29–42. https://doi.org/10.1016/j.jvlc.2011.12.001

[17] Mick Chang-Heng Lin, Fu-Ming Huang, Po-Ching Liu, Yu-Hsiang Huang, and You-shan Chung. 2016. Dijkstra-Based Selection for Parallel Multi-lanes Map-Matching and an Actual Path Tagging System. In *Intelligent Information and Database Systems*, Ngoc Thanh Nguyen, Bogdan Trawiński, Hamido Fujita, and Tzung-Pei Hong (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 499–508. https://doi.org/10.1007/978-3-662-49390-8_49

[18] Kuien Liu, Yaguang Li, Fengcheng He, Jiajie Xu, and Zhiming Ding. 2012. Effective Map-matching on the Most Simplified Road Network. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*. ACM, New York, NY, USA, 609–612. https://doi.org/10.1145/2424321.2424429

[19] Huajian Mao, Wuman Luo, Haoyu Tan, Lionel M. Ni, and Nong Xiao. 2012. Exploration of Ground Truth from Raw GPS Data. In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing (UrbComp '12)*. ACM, New York, NY, USA, 118–125. https://doi.org/10.1145/2346496.2346515

[20] MapBox. 2017. https://www.mapbox.com

[21] Fabrice Marchal, J. Hackney, and Kay W. Axhausen. 2005. Efficient map-matching of large Global Positioning System data sets: Tests on a speed-monitoring experiment in Zürich. *Transportation Research Record: Journal of the Transportation Research Board* 1935 (2005), 93–100. https://doi.org/10.3141/1935-11

[22] OpenStreetMap. 2017. https://www.openstreetmap.org

[23] Takayuki Osogami and Rudy Raymond. 2013. Map Matching with Inverse Reinforcement Learning. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI '13)*. AAAI Press, Beijing, China, 2547–2553. http://dl.acm.org/citation.cfm?id=2540128.2540495

[24] Mohammed A. Quddus, Robert B. Noland, and Washington Y. Ochieng. 2005. Validation of map matching algorithms using high-precision positioning with GPS. *Journal of Naviagation* 58, 2 (2005), 257–271. https://doi.org/10.1017/S0373463305003231

[25] Mohammed A. Quddus, Washington Y. Ochieng, and Robert B. Noland. 2007. Current map-matching algorithms for transport telematics applications: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies* 15, 5 (2007), 312–328. https://doi.org/10.1016/j.trc.2007.05.002

[26] Joerg Schweizer, Silvia Bernardi, and Federico Rupi. 2016. Map-matching algorithm applied to bicycle global positioning system traces in Bologna. *IET Intelligent Transport Systems* 10, 4 (2016), 244–250. https://doi.org/10.1049/iet-its.2015.0135

[27] Fabrício A. Silva, Clayson Celes, Azzedine Boukerche, Linnyer B. Ruiz, and Antonio A.F. Loureiro. 2015. Filling the Gaps of Vehicular Mobility Traces. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '15)*. ACM, New York, NY, USA, 47–54. https://doi.org/10.1145/2811587.2811612

[28] Renchu Song, Wei Lu, Weiwei Sun, Yan Huang, and Chunan Chen. 2012. Quick Map Matching Using Multi-core CPUs. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*. ACM, New York, NY, USA, 605–608. https://doi.org/10.1145/2424321.2424428

[29] Kozo Sugiyama and Kazuo Misue. 1995. Graph drawing by the magnetic spring model. *Journal of Visual Languages and Computing* 6, 3 (1995), 217–231. https://doi.org/10.1006/jvlc.1995.1013

[30] Youze Tang, Andy Diwen Zhu, and Xiaokui Xiao. 2012. An Efficient Algorithm for Mapping Vehicle Trajectories Onto Road Networks. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*. ACM, New York, NY, USA, 601–604. https://doi.org/10.1145/2424321.2424427

[31] Fernando Torre, David Pitchford, Phil Brown, and Loren Terveen. 2012. Matching GPS Traces to (Possibly) Incomplete Map Data: Bridging Map Building and Map Matching. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*. ACM, New York, NY, USA, 546–549. https://doi.org/10.1145/2424321.2424411

[32] Chris Walshaw. 2001. A Multilevel Algorithm for Force-Directed Graph Drawing. In *Graph Drawing*, Joe Marks (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 171–182.

[33] Guanfeng Wang and Roger Zimmermann. 2014. Eddy: An Error-bounded Delay-bounded Real-time Map Matching Algorithm Using HMM and Online Viterbi Decoder. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '14)*. ACM, New York, NY, USA, 33–42. https://doi.org/10.1145/2666310.2666383

[34] Suyi Wang, Yusu Wang, and Yanjie Li. 2015. Efficient Map Reconstruction and Augmentation via Topological Methods. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '15)*. ACM, New York, NY, USA, Article 25, 10 pages. https://doi.org/10.1145/2820783.2820833

[35] Hong Wei, Yin Wang, George Forman, Yanmin Zhu, and Haibing Guan. 2012. Fast Viterbi Map Matching with Tunable Weight Functions. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*. ACM, New York, NY, USA, 613–616. https://doi.org/10.1145/2424321.2424430

[36] Christopher E. White, David Bernstein, and Alain L. Kornhauser. 2000. Some map matching algorithms for personal naviation assistants. *Transportation Research Part C: Emerging Technologies* 8, 1 (2000), 91–108. https://doi.org/10.1016/S0968-090X(00)00026-7

[37] Dongdong Wu, Tongyu Zhu, Weifeng Lv, and Xin Gao. 2007. A Heuristic Map-Matching Algorithm by Using Vector-Based Recognition. In *2007 International Multi-Conference on Computing in the Global Information Technology (ICCGI '07)*. IEEE, Guadeloupe City, Guadeloupe, 18–18. https://doi.org/10.1109/ICCGI.2007.3

[38] Jing Yuan, Yu Zheng, Chengyang Zhang, Xing Xie, and Guang-Zhong Sun. 2010. An Interactive-Voting Based Map Matching Algorithm. In *2010 Eleventh International Conference on Mobile Data Management*. IEEE, Kansas City, MO, USA, 43–52. https://doi.org/10.1109/MDM.2010.14