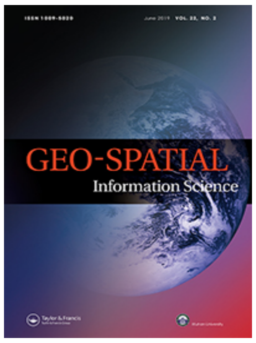


Geo-spatial Information Science

ISSN: 1009-5020 (Print) 1993-5153 (Online) Journal homepage: https://www.tandfonline.com/loi/tgsi20

A map-matching algorithm dealing with sparse cellular fingerprint observations

Andrea Dalla Torre, Paolo Gallo, Donatella Gubiani, Chris Marshall, Angelo Montanari, Federico Pittino & Andrea Viel

To cite this article: Andrea Dalla Torre, Paolo Gallo, Donatella Gubiani, Chris Marshall, Angelo Montanari, Federico Pittino & Andrea Viel (2019) A map-matching algorithm dealing with sparse cellular fingerprint observations, Geo-spatial Information Science, 22:2, 89-106, DOI: 10.1080/10095020.2019.1616933

To link to this article: https://doi.org/10.1080/10095020.2019.1616933

GEO-SPATIAL INFORMATION SCIENCE
2019, VOL. 22, NO. 2, 89–106
https://doi.org/10.1080/10095020.2019.1616933

A map-matching algorithm dealing with sparse cellular fingerprint observations

Andrea Dalla Torre[a], Paolo Gallo[b], Donatella Gubiani[c], Chris Marshall[d], Angelo Montanari[b], Federico Pittino[a] and Andrea Viel[b]

[a]Cellular Product Center, u-blox Italia SpA, Trieste, Italy; [b]Department of Mathematics, Computer Science and Physics, University of Udine, Udine, Italy; [c]Center for Information Technologies and Applied Mathematics, University of Nova Gorica, Nova Gorica, Slovenia; [d]Cellular Technology, u-blox UK Ltd, Reigate, UK

ABSTRACT
The widespread availability of mobile communication makes mobile devices a resource for the collection of data about mobile infrastructures and user mobility. In these contexts, the problem of reconstructing the most likely trajectory of a device on the road network on the basis of the sequence of observed locations (map-matching problem) turns out to be particularly relevant. Different contributions have demonstrated that the reconstruction of the trajectory of a device with good accuracy is technically feasible even when only a sparse set of GNSS positions is available. In this paper, we face the problem of coping with sparse sequences of cellular fingerprints. Compared to GNSS positions, cellular fingerprints provide coarser spatial information, but they work even when a device is missing GNSS positions or is operating in an energy saving mode. We devise a new map-matching algorithm, that exploits the well-known Hidden Markov Model and Random Forests to successfully deal with noisy and sparse cellular observations. The performance of the proposed solution has been tested over a medium-sized Italian city urban environment by varying both the sampling of the observations and the density of the fingerprint map as well as by including some GPS positions into the sequence of fingerprint observations.

ARTICLE HISTORY
Received 16 November 2018
Accepted 19 March 2019

KEYWORDS
Map-matching algorithm; trajectory; cellular fingerprint; Hidden Markov Model

1. Introduction

Nowadays, the widespread availability of mobile communication makes the mobile phone a resource that may collect information about urban mobility and cellular networks at a very low cost. It can be exploited, for instance, to determine the approximate position of a phone user on the basis of information about the signal of received cells only, if there are previous georeferenced data to match against. Moreover, the collection of sequences of such observations relative to a given mobile phone allows one to guess the path followed by the user carrying that phone. As a matter of fact, reconstructing the path of a moving device is an important task in many application fields, ranging from location-based services to asset tracking and fleet management.

As people do not usually move randomly, but follow the underlying transportation infrastructures, there is the chance to use information about road networks to improve both position detection and trajectory reconstruction. The map-matching problem is mainly about reconstructing the most likely trajectory of a device across a road network given a sequence of observed locations. A variety of approaches to such a problem can be found in the literature. For a survey on them see, for instance, (Quddus, Ochieng, and Noland 2007). They differ in various respects and make use of different technologies. In particular, they exhibit a different trade-off among cost, energy consumption, and accuracy.

The problem has been extensively studied in the case of observations that include location information from a Global Navigation Satellite System (GNSS), like the Global Positioning System (GPS) (Thiagarajan et al. 2011; Zhuang, Kim, and Singh 2010; Paek et al. 2011; Li et al. 2014). However, whenever GPS data are not available, locations must be inferred on the basis of accessible information. In this work, we focus on solutions based on cellular networks, where signal fingerprinting provides a complementary/alternative source of information with respect to GPS. Compared to those based on GPS, solutions based on signal fingerprinting also guarantee a lower energy consumption. For such a reason, despite the fact that the accuracy in positioning that they guarantee is, in general, lower, in energy constrained environments they are often the preferred choice. Moreover, since in most of the considered systems a communication capability is

CONTACT Donatella Gubiani donatella.gubiani@ung.si

This paper is an extended version of the paper "Map matching with sparse cellular fingerprint observations" from 2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS), DOI: 10.1109/UPINLBS.2018.8559918 an additional requirement, using a single device (the cellular module) for both communication and positioning turns out to be a cost-effective solution.

Fingerprint positioning systems are based on the observation of the received signals, whose strength, which depends on the current location of the receiver device, may be different (Chen et al. 2006; Benikovsky, Brida, and Machaj 2010). In general terms, a fingerprint can be defined as the set of the values of the signal strengths of the observable transmitters. In the specific case of cellular networks, a fingerprint is the collection of the values of the signal strengths of the available cell towers. One of the fundamental modules of a fingerprint positioning system is the database, that stores observations taken at known locations. When a device asks the system for an estimation of its current position, it compares the fingerprint of the device, that is, the observed signals, with the set of past observations recorded in the database. To execute such a comparison, the system has to distinguish among the signals coming from different cells (at any given time, signals coming from multiple cells are usually received by a device). Since each cell broadcasts a set of identifying parameters, it can be easily recognized (Hoy 2015).

While fingerprint positioning is more energy efficient than an approach using GPS, its overall performance strongly depends on the quantity and distribution of available fingerprints. A problematic case for map matching is given by scenarios where observations are temporally sparse, i.e. the frequency at which devices collect data is so low that a few minutes, or even more, may elapse from one observation to the next. Observations can be temporally sparse to reduce energy consumption, but also to satisfy specific constraints on data storage or transmission. Solutions that work well with temporally dense data become less effective when applied to temporally sparse data. The problem of map matching when data are spatially noisy and temporally sparse has been addressed in the GPS setting. In order to deal with the limited amount of information, probabilistic algorithms are exploited that return the most likely trajectory with a certain degree of accuracy. One of the most effective solutions, which makes use of Hidden Markov Models (HMM), is reported in (Newson and Krumm 2009). It has been exploited and further improved in subsequent contributions (Osogami and Raymond 2013; Oran and Jaillet 2013; Jagadeesh and Srikanthan 2015).

In this paper, we go a step further and propose a solution to the problem of reconstructing the trajectory of a device when data are spatially noisy and temporally sparse using only cellular fingerprints. The worst scenario is probably the one in which, due to a low sampling rate, limited temporal information is available and spatial information has a low accuracy.

To deal with sparse observations in the form of cellular fingerprints, we design an HMM-based map-matching algorithm that works offline. It consists of four main steps. It starts with (i) a preprocessing step, followed by (ii) the generation of the sets of states and (iii) the definition of the set of transitions between them. Once the model has been generated, (iv) the most likely map-matched trajectory is extracted from it. To compare fingerprints, the algorithm makes use of a machine learning method based on decision tree ensembles, that turns out to be competitive against state-of-the-art fingerprint comparison functions (Viel et al. 2018).

To demonstrate that the proposed algorithm can reconstruct trajectories even with fingerprint observations that intrinsically have a low spatial accuracy, we tested it in the urban environment of a medium-sized Italian city. Moreover, in order to evaluate its robustness, we applied it to a variety of scenarios obtained by (artificially) changing the sampling of the observations and the density of the fingerprint map. We completed the experimentation by checking its behaviour over mixed sequences of GPS and fingerprints.

The paper is organised as follows. In Section 2, we briefly analyse related work on cellular fingerprinting positioning systems, the map-matching problem, and the application of HMM to it. Then, in Section 3, we illustrate the proposed algorithm, with a special attention to the applied methods. The considered dataset is described in Section 4. In Section 5, we show the algorithm at work on real data, and then, in Section 6, we evaluate the experimental results. Finally, in Section 7, we summarise the achieved results and outline future work directions.

2 Related work

Before presenting details and outcomes of the proposed solution, we give a short account of the relevant literature. We first focus on positioning systems, that make use of cellular fingerprints (Section 2.1), and then we discuss the map-matching problem, and how it can be addressed by means of HMM (Section 2.2).

2.1. *Cellular fingerprint positioning systems*

Cellular fingerprint positioning systems aim at estimating the current position of a device by comparing its fingerprint with the fingerprints of known positions stored in the system database. Such a comparison of the observed fingerprint with the recorded ones is a crucial component of the behaviour of the system. As the devices communicate over a network, fingerprints always provide information about the serving cell and its signal strength. Additional information about the set of neighbour cells, possibly including the strength of their signals, is sometimes taken into consideration as well. In the best case, all the physical parameters of all the observed cells are available.

Various metrics have been used to evaluate the distance between pairs of fingerprints in order to identify, among the fingerprints stored in the database, the most similar ones (the underlying assumption is that the more similar the fingerprints are, the closer are their positions). The most commonly used metric is the Euclidean distance (Bahl and Padmanabhan 2000). Others metrics proposed in literature are the Spearman Correlation (Zekavat and Buehrer 2011), which measures similarity between two ordered sets using ranking, and Hyperbolic Fingerprinting (Kjærgaard and Munk 2008), that compares signal strength between pairs of bacon. An innovative approach that makes use of decision tree ensembles has been developed in (Viel et al. 2018).

A variety of localisation methods that exploit information from cellular networks have been proposed in the literature (Deblauwe 2008). The distinguishing features of signal fingerprinting are that (i) it does not require any assistance from the network operators and (ii) it does not impose any significant modification to the network infrastructure. Signal fingerprinting has been successfully used in various contexts, including Wi-Fi and cellular networks, and indoor and outdoor environments. As for Wi-Fi networks, the RADAR system (Bahl and Padmanabhan 2000) proves that Wi-Fi access points can be exploited to achieve accurate indoor location estimations. As for cellular networks, GSM fingerprinting in an outdoor context has been investigated in (Chen et al. 2006). In such a contribution, it clearly emerges that the quality and the quantity of the fingerprints available for the position estimation significantly affect the outcomes of the process. Cellular fingerprints have also been paired with information coming from other sensors to increase the performance of a positioning system. This is the case, for instance, with (Aly and Moustafa 2013), where the authors make use of signals coming from cellular networks to reset the accumulated error of a system exploiting inertial sensors.

In general, a variety of factors influence the performance of fingerprint positioning. The density of cells is one of them: usually, the higher the density, the higher the precision is. With a high density, indeed, it becomes easier to differentiate the fingerprints, even in small areas. An important role is played also by the quantity and the distribution of the observations recorded in the database: a highly populated fingerprint map generally produces more accurate estimations. A particularly advantageous characteristic of the fingerprinting method, that distinguishes it from other cellular positioning ones, like, for instance, those based on the Timing of Arrival, is its tolerance toward signal fluctuations and multi-path effects, which is a very helpful feature in urban environments, where also GPS is mostly at a disadvantage. A significant drawback of the method is that a sampling of the territory is needed to obtain a map of signal fingerprints, tagged with the GPS position of the collecting site. However, once the fingerprint map has been produced, the position estimation can be obtained in a very fast way, even in those situations where GPS can take a long time to get a fix.

In this paper, fingerprint comparison is done by means of decision trees. Decision trees are a popular method for many supervised machine learning tasks. Their success is due to the fact they are easily interpretable and quite efficient, during both the learning and the prediction phases. On the negative side, the predictions obtained by means of decision trees are usually less accurate than those that can be achieved with other methodologies. Moreover, decision trees have a tendency to over-fit training data. A way to significantly improve the accuracy of the method is to combine a set of different decision trees into a so-called *Random Forest* (Breiman 2001). This solution builds a set of trees during the training phase, and then it outputs the average prediction of the individual trees as the result. Such a technique has been exploited to estimate fingerprint similarity in (Viel et al. 2018), by making use of an ensemble of decision trees. Basically, the similarity between a pair of fingerprints is assessed by means of a set of features, which includes the number of common cells and other parameters related to signal strength. An experimental evaluation of the method has shown a significant improvement in accuracy compared to other methods such as, for instance, the one based on the Euclidean distance. The method has also other advantages over traditional ones. By using features that characterize the differences between fingerprints, rather than the fingerprints themselves, the model is not specific to the area in which it was generated, that is, at least in principle, it is not strictly necessary to train a different model for each region. Moreover, the model provides a similarity measure in meters, which is an advantage compared to other metrics in the literature that report a similarity value devoid of an actual spatial interpretation.

2.2. The map-matching problem

Map matching aims at integrating positioning data with spatial information about the road network. The problem of map matching has been largely explored in the literature and various algorithms have been proposed, which differ from each other in the available sensors, sampling rate, road network model, and application field.

There is not a widely accepted classification of the proposed solutions. Some possible criteria have been suggested in (Quddus, Ochieng, and Noland 2007). Considering the way observations are processed, one can distinguish between online (or real time) and offline algorithms. In the first case, the observations are elaborated incrementally, in the second one, there is a post-processing phase that involves all the observations of a path at once. Alternatively, one can distinguish among geometrical, topological, and probabilistic map-matching algorithms. A partition can also be done on the basis of the frequency of observations, that allows us to split existing solutions into two classes: low frequency and high frequency. Finally, there exist some GPS-based solutions, that are respectively based on probabilistic theory, Kalman filter, fuzzy logic, and belief theory, which are not easily ascribable to any of the above classes (Quddus, Ochieng, and Noland 2007).

A critical analysis of several map-matching solutions, which pay a special attention to their performance, can be found in (Hashemi and Karimi 2014). Its outcomes can be summarised as follows: (i) the choice of the map-matching algorithm to use in a specific application strongly depends on the assumptions and the expected accuracy, (ii) there are few map-matching algorithms that fully exploit topology networks, taking into consideration parameters like, for instance, direction and turn-restriction, and (iii) there exists a trade-off between accuracy and simplicity of map-matching algorithms.

2.2.1. *HMM-based map matching*

The Hidden Markov Model (HMM) is a statistical model designed to deal with sequential data with an underlying hidden structure. It has a number of applications in domains such as language processing, speech recognition, and biological analyses (Juang and Rabiner 1991; Jurafsky and Martin 2009). Recently, HMM techniques have been successfully exploited in traffic monitoring and positioning (Aly and Moustafa 2015; Thiagarajan et al. 2009).

In the last years, it has been successfully applied in the context of map-matching trajectories based on GPS observations, which are, at the same time, spatially noisy and temporally sparse. The proposed solutions extensively used HMM to describe systems whose states cannot be accessed directly, but only through external observations. Indeed, HMM turns out to be particularly useful when matching a location observation, especially, for example, in the case of cellular observations, for situations when the closest road on the map cannot be distinguished reliably (Algizawy, Ogawa, and El-Mahdy 2017).

Map-matching algorithms based on HMM build a probabilistic model that, on the basis of the available observations, describes the routes that a device may have followed. States can be viewed as a set of candidate road segments for each observed location, while transitions are interpreted as paths between pairs of road segments. States and transitions are paired with an emission and a transition probability, respectively. The emission probability expresses the likelihood that the measurement was done in that location, while the transition probability is the probability that the path was taken. Most HMM methods proposed in literature heavily rely upon the computation of those probabilities. As an example, in CTrak (Thiagarajan et al. 2011), HMM was exploited for the mapping of trajectories in a scenario based on cellular fingerprints. Once HMM is built, the most likely sequence of states is computed by means of the Viterbi algorithm (Forney 1973; Newson and Krumm 2009).

HMM-based map-matching algorithms are usually applied once all the location data are collected to generate the full path. An on-line alternative is to use a sliding window method, where the path is computed excluding the n-th latest observations (Lou et al. 2009). This approach is well suited for real-time applications, but, unfortunately, it considerably reduces the accuracy of the algorithm. In (Goh et al. 2012), a different approach that makes use of a variable-width sliding window is outlined, which guarantees that the optimal solution is still found.

Finally, the problem of dealing with scenarios where the positions in the sequence are sparsely distributed and affected by noise is addressed in (Newson and Krumm 2009). The proposed HMM-based map-matching algorithm has been experimentally evaluated with a GPS trace across Seattle, adding artificial noise in observations and removing points to obtain a sparse trace, and it proved itself quite successful. For this reason, it has been largely reused and improved by a number of authors in subsequent contributions.

2.2.2. *A short account of existing contributions*

In the following, we briefly describe the most significant developments of the work by Newson and Krumm.

Lou et al. (2009) use a candidate graph model, in some way similar to the HMM-based solution outline in (Newson and Krumm 2009), that exploits the road network and the spatial constraints on the trajectory. Such a solution was later improved by Yuan et al. (2010), whose proposal takes into account the weighted mutual influences among GPS points by means of a suitable voting mechanism. Osogami and Raymond (2013) make use of the model proposed in (Newson and Krumm 2009), but they apply a different transition probability function, that, in order to estimate the probability of traversing a certain path, takes into consideration the number of turns. A similar approach is followed by Oran and Jaillet (2013), where an alternative transition function, based on a cumulative proximity-weight formulation, is used.

The map-matching problem in the presence of cellular positioning data entered the research agenda only very recently. Existing solutions differ from the one given in the present work in the assumptions about the available data. More precisely, Thiagarajan et al. (2011) propose a fingerprint-based map-matching algorithm that requires a very high sampling rate (about a fingerprint per second) and the availability of accelerometers and magnetometers. Jagadeesh and Srikanthan (2015) develop a variant of the HMM algorithm in (Newson and Krumm 2009). In order to address noisy and sparse smart-phone location data (both Wi-Fi and cellular), that, unlike the case of cellular fingerprints, is already in the form of latitude/longitude pairs, they suitably change the probability functions.

All the above-described methods use data which can be collected by the device itself, e.g. fingerprints. It is worth mentioning that there exists a different class of methods that work with data collected by the cellular operator for purposes generally unrelated to positioning. This is the case, for instance, with call logs, which are available to the infrastructure, but, usually, not to the client modem or handset. Leontiadis et al. (2014) describe a solution that, in order to reconstruct the trajectory of a mobile user, exploits cellular handover logs from the network operator, which contain coarse spatial and temporal information as well as knowledge about the coverage of the cells. A different solution is outlined in a work by Schulze, Horn, and Kern (2015), where, in order to determine the most likely path, one constructs a street graph by using the sequence of visited cells.

3. Methods

In the following, we describe our solution to the problem of reconstructing the trajectory of a device in a scenario where only spatially noisy and temporally sparse data from cellular fingerprints are given. We assume that no GPS data from the device are available, and only rely on (i) basic information about the road network and (ii) cellular fingerprints to be matched against a database of previously-collected fingerprints with an associated GPS position.

The proposed map-matching algorithm exploits a suitable adaptation of the HMM construction. One of its distinctive features is an original technique for the generation of the states of the probabilistic model, starting from the fingerprint observations, that makes use of a machine learning method based on decision tree ensembles (Viel et al. 2018). The algorithm can also be easily adapted to deal with mixed sequences including both fingerprints and GPS fixes.

The map-matching algorithm consists of four main steps (see Figure 1 for a graphical illustration):

(1) *preprocessing*: the first step removes data which might be incorrect or simply redundant for reconstructing the trajectory;
(2) *emission probability*: by exploiting decision trees, for each fingerprint observation in the

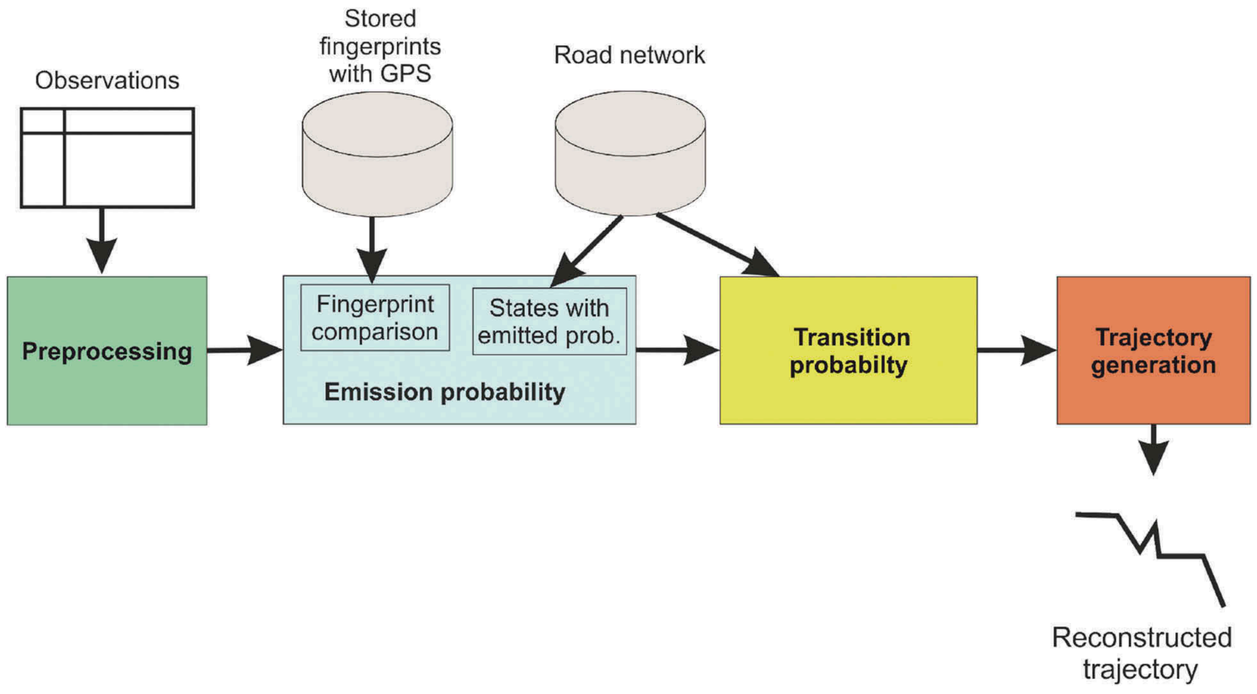


Figure 1. Diagram with the four phases (colored boxes) of the algorithm.

sequence associated with a device, the second step retrieves the most relevant fingerprints in the database and uses them to find the most probable roads nearby;

(3) *transition probability*: the third step connects the roads and assigns them a probability which depends on the given spatial and temporal constraints;

(4) *trajectory generation*: finally, in the fourth step, the most probable trajectory is generated by applying the Viterbi algorithm on HMM.

Most of the above phases are implemented by means of PL/pgSQL (Procedural Language/PostgreSQL) (PostgreSQL Global Development Group 2008). PL/pgSQL is a fully-fledged programming language that allows much more procedural control than plain SQL. For instance, it gives the possibility to use loops and other control structures. As for the emission probability phase, we exploited the machine learning software Weka (Waikato Environment for Knowledge Analysis) (Hall et al., 2009) for the generation of random forests. Details about the data and tools used to compute the shortest path between pairs of points on the road map during transition probability phase are described in Section 4.

3.1. Preprocessing

The preprocessing phase aims at removing data that does not really help in reconstructing the trajectory. As an example, a user travelling around in a city with a mobile phone can acquire a lot of observations even for a short trip. Moreover, he/she can stay almost still for a while, thus recording the same set of visible cells repeatedly. Finally, during such an activity, the measurements may be affected by several errors (a list of typical measurement errors is reported in (Ulm, Widhalm, and Brndle 2015)). In addition, since cellular fingerprints are generally imprecise, using observations which are temporally very close can easily generate some noise, without producing any clear benefit. Last but not least, a huge number of observations means a large HMM, and thus an increase in the time needed to compute the trajectory.

The problem of filtering the collected trip data has been extensively investigated in the literature. However, it is not the main focus of the present work, and thus we limit ourselves to remove observations that are temporally close to each other (below 10 seconds). More complex filtering criteria can be possibly adopted without affecting the algorithm in any significant way.

3.2. Emission probability

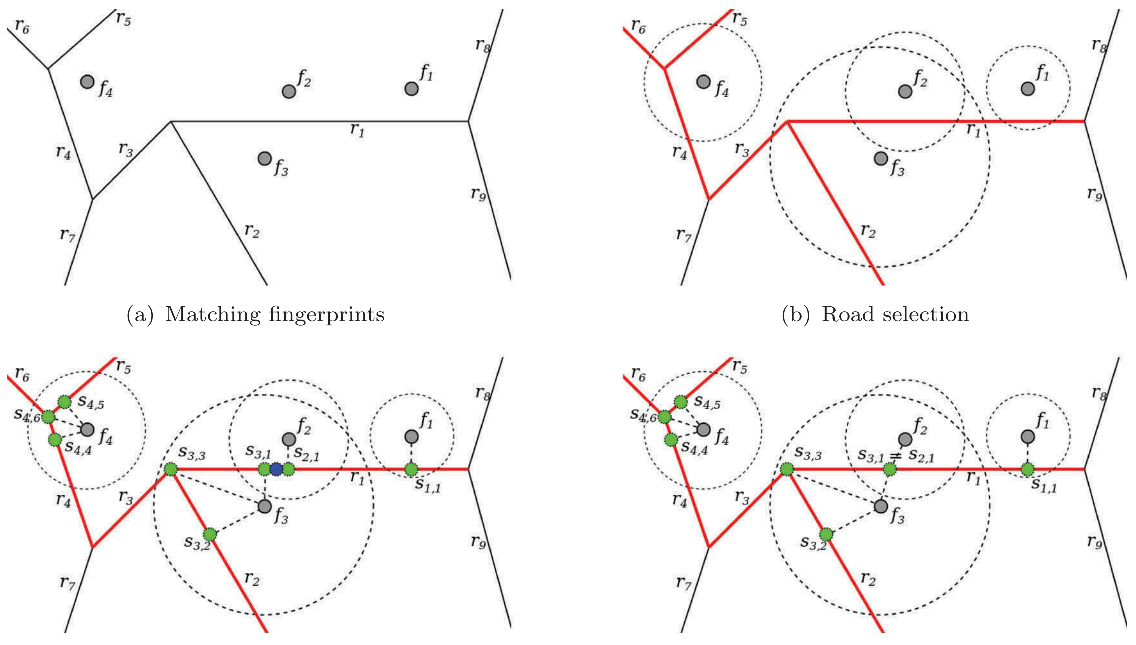
Once the unnecessary observations have been removed, the next step consists of building an HMM in order to compute the most likely trajectory followed by the device on the map. HMM-based map-matching algorithms work by constructing a probabilistic model, basically a graph, which represents the possible routes that a device may have taken on the basis of the available observations. For every location observation, a set of candidate road segments are identified, which becomes a set of hidden states of the model. Two states obtained by subsequent observations are linked together by a transition, which represents a path between the two road segments. This representation exploits the connectivity of the road network.

The HMM building process starts with the identification of the hidden states and the computation of their emission probabilities. The overall process can be split into two phases, as shown in Figure 1: (i) comparison of fingerprints and (ii) selection of the states and assignment of an emitted probability to each of them.

The similarity between a pair of fingerprints is assessed as follows. Initially, for each pair of fingerprints, a set of features, like the number of common cells and other characteristics related to the signal strength, is extracted. These features are then fed into a Random Forest model, as proposed in (Viel et al. 2018), trained to report the distance in meters between the locations at which the fingerprints were observed. The output of the comparison phase is the set of fingerprints which are most similar to the observed one. An example is given in Figure 2, where the fingerprints closest to the input one are labelled as f_1,f_2,f_3, and f_4 (Figure 2(a)).

In HMM, each input fingerprint (observation) generates several states, which basically represent candidate points on the roads. A probability must be associated with any such state, which represents the probability of the device being in that place at that step of the trajectory. Since in cellular fingerprints spatial information is not explicit, the computation of the states is more complex than in the case of GPS observations.

Once we have identified the most similar stored fingerprints, the next step consists of the identification of the set of candidate road segments in the map by exploiting information about the road network. As a preliminary step, road segments longer than a certain threshold are split into multiple segments. Then, the output of the Random Forest, which is expressed in meters, is used as an upper bound to the search radius for the roads nearby (plus some tolerance). This makes the method more robust in the case of a lacking fingerprint map. As shown in Figure 2(b), f_1 and f_2 are related to r_1, f_3 is related to r_1, r_2, and f_4 is related to r_4, r_5, and r_6. It is worth pointing out that roads r_7, r_8, and r_9 do not intersect any search circle. Notice also that the search radii



(a) Matching fingerprints (b) Road selection

(c) Points to merge (d) Merging complete

Figure 2. An example of the process of computing the emission probability. It starts with the matching of closest fingerprints using RF (a). Then, it uses the RF distance in meters to select nearby roads for each matched fingerprint (b). Next, the closest points on the roads (dotted circle in green) for each fingerprint (solid circle in grey) are computed (c). Finally, there is a check for close enough points, which are merged together, producing the insertion of blue point in (c), thus obtaining the final set of points (d).

around fingerprints are, in general, different, as they depend on the distance returned by the Random Forest algorithm.

Next, for each fingerprint, we determine the set of closest points on the corresponding roads computed by the previous step, which become the states of HMM. This is done on the basis of the Euclidean distance between the coordinates of the stored fingerprints and the road segments. As an example, in Figure 2(c), the fingerprint f_3 produces the states $s_{3,1}$, $s_{3,2}$, and $s_{3,3}$ on the roads r_1, r_2, and r_3, respectively, while the fingerprints f_1 and f_2 respectively produce the states $s_{1,1}$ and $s_{2,1}$ both belonging to the road r_1. Since similar fingerprints are generally close to each other, there can be multiple states on the same road segment. This is the case, for instance, with $s_{2,1}$ and $s_{3,1}$ in Figure 2 (c), which are related to two very close fingerprints for the same observation (input fingerprint). If they are close enough, they are merged into one single point (the blue point in Figure 2(c)), which finally becomes a single state of the model (points $s_{2,1} = s_{3,1}$ in Figure 2(d)). Such an approximation is fairly natural as, given the limited precision of fingerprinting locations, it is not necessary to discriminate between close locations. Moreover, it reduces the number of states and, consequently, the running time. Finally, since we are interested in reconstructing the trajectory of the device, if an intermediate point is shifted along the same road, the final trajectory remains mostly unchanged.

The probability of each state of the model is essentially given by the distance reported by the Random Forest for the fingerprint that generated it. Formally, the emission probability for a state s_i, generated by a fingerprint observation o_j, is determined as follows. First, we compute the values:

$$p(o_j|s_i) = \max_k d_{RF}(o_j,f_k) - d_{RF}(o_j,f_c) \qquad (1)$$

where d_{RF} is the distance returned by the Random Forest model, f_c is the fingerprint that generated s_i, and each f_k is a fingerprint close to o_j. The values are then suitably normalised to obtain a proper probability measure. As a result, the states generated by a closer fingerprint have a higher probability.

In the case of a state which has been generated by merging multiple points, the probability is given by the fingerprint at the shortest distance. We would like to remark that we decided to take the lowest value, and not the sum, as otherwise, we would take into account the density of the fingerprint map in the area, which does not depend on the current position of the device. Of course, this would be a reasonable alternative if the fingerprint map was collected in a crowd-sourced effort and density was due to the inherent popularity of a location. Taking the average into consideration, instead, could increase the influence of noise. Hence, using the one at the shortest distance seems a reasonable choice, considering also that the model tends to overestimate rather than to underestimate the distance (Viel et al. 2018).

Last but not least, even though the proposed algorithm has been designed to work with cellular fingerprints, its structure makes it easy to adapt it to the case of mixed sequences of fingerprints and GPS observations. In such a case, following the approach outlined in (Newson and Krumm 2009), each GPS measurement is matched to the nearest point of the nearby roads (up to a certain range, that is, 60 m). In comparison to fingerprinting, it can be seen as a simplified case, because high precision coordinates are already available. These points become states of the model and a probability is assigned to them according to a Gaussian distribution which gives a higher probability to the road segments closer to the GPS point. The deviation is given by the accuracy reported by the GPS module.

More precisely, the emission probability for a state s_j obtained by a GPS observation o_i is defined as:

$$p(o_j|s_i) = \frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{d^2}{2\sigma^2}} \tag{2}$$

where d is the distance between the point p_j, from which s_j is obtained, and the GPS measurement o_i. Usually, the standard deviation σ is set to a fixed value, obtained by estimating the common noise of a GPS fix. However, using the accuracy reported by the GPS module could be arguably more accurate, since it varies with each measurement. Another difference between GPS and fingerprint observations, in the way we use them, is the range for searching road segments around observations. For GPS observations, we can choose a fixed distance (60 m), considering that the precision given by GPS is generally higher and the probability outside that range should be minimal in any case. Finally, as in the base case, road segments longer than a certain threshold are split into multiple segments. We use different thresholds for GPS and cellular fingerprints, 10m and 100m respectively, taking into account their different accuracy.

3.3. Transition probability

States and transitions are respectively paired with an emission and a transition probability. The former gives the likelihood that the measurement was taken in that location, the latter represents the probability that the path was taken.

As illustrated in Figure 3, in HMM, pairs of states generated by consecutive measurements are linked by a transition, that represents the path on the road network that connects the road segments corresponding to those states. The path is computed by the Dijkstra algorithm suitably configured to return the shortest route.

To assign a probability to transitions, we adopted the probability function proposed in (Jagadeesh and Srikanthan 2015). Here, we only give a general account of it, and refer the reader to (2015) for details. It has a spatial component, inspired by Newson and Krumm (2009), and a temporal component, that deals with the difference between the time actually elapsed from the first measurement to the second one and the estimated travel time.

The spatial component of the probability function is given by the difference between the great-circle distance among the points on the road segments associated with the states and the length of the path that connects them following the road network. Given two states s_i and s_j, which have been obtained from two consecutive observations, its value is computed as follows:

$$\frac{D_d(s_i, s_j) - D_s(s_i, s_j)}{\Delta t} \tag{3}$$

where the function D_d returns their driving distance, the function D_s computes their straight-line distance, and Δt reports the time actually elapsed between the two observations. The rationale behind such

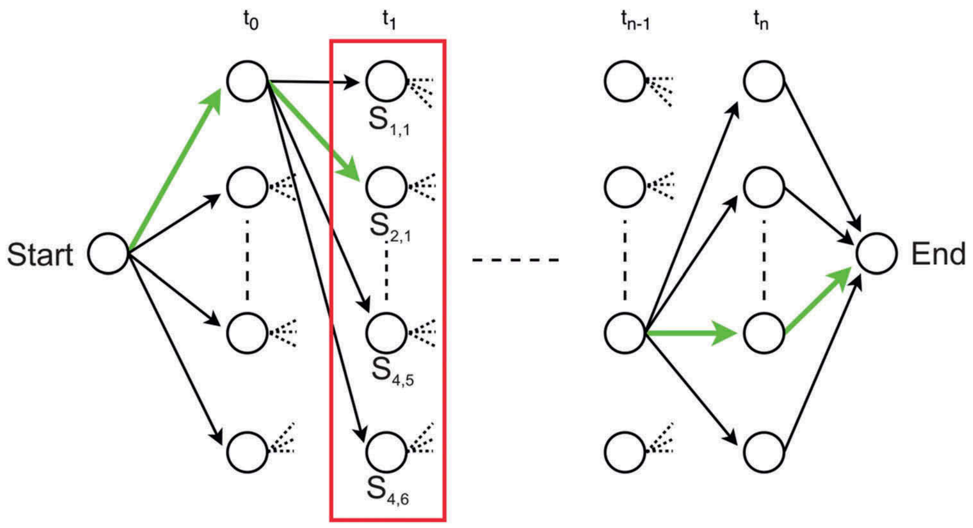


Figure 3. In HMM, the map-matched path is computed by choosing a state and a transition for each observation. States from Figure 2 are represented inside the red box. A possible path is highlighted in bright green.

a component of the function is that straight paths are more likely than tortuous ones and thus they are given a higher probability.

Since we are managing sparse and noisy observations, it is clearly advantageous to exploit as much as possible information coming from the road network. As an example, on the basis of the length and the speed limits of the roads, it is possible to approximate the time necessary to travel a path.

The temporal component of the probability function assigns a lower probability to paths which require a time higher than the actual travel time given by the timestamps of the trajectory measurements:

$$\frac{\max((T_e(s_i, s_j) - \Delta t), 0)}{\Delta t} \tag{4}$$

where the function T_e computes the estimated driving time between the two states and Δt reports the time actually elapsed between the two observations. As a matter of fact, when the estimated time is far higher than the actual travel time, the path is discarded since it is unrealistic (and would have been assigned a very low probability anyway). Notice that the opposite case, that is, the case in which the device takes more time than expected, is not penalised, as a long travel time may reflect bad traffic conditions. The two equations are then combined as shown in (Jagadeesh and Srikanthan 2015).

Finally, map data generally include a classification of road segments as highway, pathway, and so on. Whenever the current travelling method is known, the map matching algorithm makes use of it to simplify the transition computation by ignoring those roads that cannot be used by the vehicle under consideration. This does not only improve the accuracy of the method, but it also improves the running time, as it reduces the search space of the Dijkstra algorithm.

3.4. Generating the trajectory

Once HMM is built, we can obtain a map-matched trajectory as the result of the merging of the more likely paths between each pair of consecutive states. To this end, we apply to well-known Viterbi algorithm (Forney 1973; Newson and Krumm 2009).

An example is given in Figure 3, where the state to the left is the starting point for each possible sequence. For each observation in the sequence (from time t_0 to time t_n), there exists a set of candidate states. With reference to the example in Figure 2, the states associated with the observation taken at time t_1 are collected in column inside the red box. A sample path from the starting node to the ending one is coloured in bright green.

It is worth pointing out that there is a chance that synthesising the trajectory turns out to be impossible. This happens, for instance, whenever the set of states, relative to a certain observation, with an incoming transition and the set of those with an outgoing transition are disjoint (as we already observed, some transitions could have been discarded because they require an unreasonable time to be travelled). There are three ways to cope with this unfortunate situation: keep all the transitions, remove all the states related to that observation from the model, or split the trajectory into two shorter ones. We opted for the second approach, because such a situation is an indication that those states are probably too far from the correct position and that explains the unreasonable time. In any case, this is a last resort method, and in fact it has been necessary just in one case in our evaluation.

4. Fingerprint data collection

In this section, we describe the collection of data, the way in which they have been acquired, and the tools that we used to test the proposed map-matching algorithm against a real scenario.

Data consist of a set of fingerprints paired with GPS location information collected in an urban environment. We chose to evaluate the proposed solution in an urban scenario, because it is characterized by quite complex road networks from which one generates a large number of road candidates. The presence of a lot of buildings, interfering with the signal propagation, is an additional challenging factor to cope with. In comparison, reconstructing a trajectory in rural scenarios seems to be more straightforward and less error-prone. Moreover, to evaluate the robustness of the proposed solution, starting from the collected urban dataset, we generated and tested different scenarios by simply removing sets of fingerprints.

Data have been collected in the historical centre of Udine, a medium-sized Italian city, in order to prove that the trajectory of a moving device can be reconstructed on the basis of a set of fingerprint observations with an intrinsically low spatial accuracy. The overall data collection process took several days using a Sony Xperia Z3 Compact phone equipped with a basic Android application developed for the purpose. In fact, data collection can be done with any cellular device capable of recording logical and physical parameters of visible cells, not necessarily a mobile phone. Different devices may obviously experience better (or worse) signal reception. However, these differences do not significantly affect the overall performance of the system as long as the same device is used for both sampling and testing.

As shown in Figure 4, the process of data collection involved over 4000 GSM fingerprints. Each collected observation includes both cellular fingerprints and GPS-provided locations at the time of capture. Let us take a closer look at the data. Table 1 reports a sequence of 10 fingerprints from the collected dataset. The first three columns respectively contain the identifier of the fingerprint (*ID*), the serving cell identifier (*Serving*), and the capture time (*Time*). The fourth column (*Cell Signature*) contains the set of all visible cells, including the serving one, while the fifth column records the signal strengths (*Signal Strength*). It is worth pointing out that, even if not listed in the table, each record also includes other typical GPS information, such as position, altitude, speed, bearing, number of satellites, and accuracy.

In order to evaluate the proposed algorithm and its performance, besides a set of fingerprints with the associated GPS fixes, it was necessary to collect some sparse fingerprint data, devoid of GPS information, describing a few paths. To this purpose, we extracted from the entire set of observations three paths between different points of interests in Udine, respectively path A, path B, and path C (Figure 5). Those traces were originally collected at a high frequency sample rate, with just a few seconds between each observation; in order to simulate the case of sparse observations, we artificially degraded the sequences to obtain lower sampling periods of approximately 30, 60, and 120 seconds. In such a way, we have been able to obtain new traces with different levels of sampling rate, that allowed us to measure the impact of the changes in the sampling period on the outputs of the proposed solution. An account of the distinctive characteristics of the considered traces is given in Table 2.

All data were collected by bike. There are various motivations for such a choice. First of all, it is a common way of travelling, especially in small- and medium-sized cities. Second, it basically represents

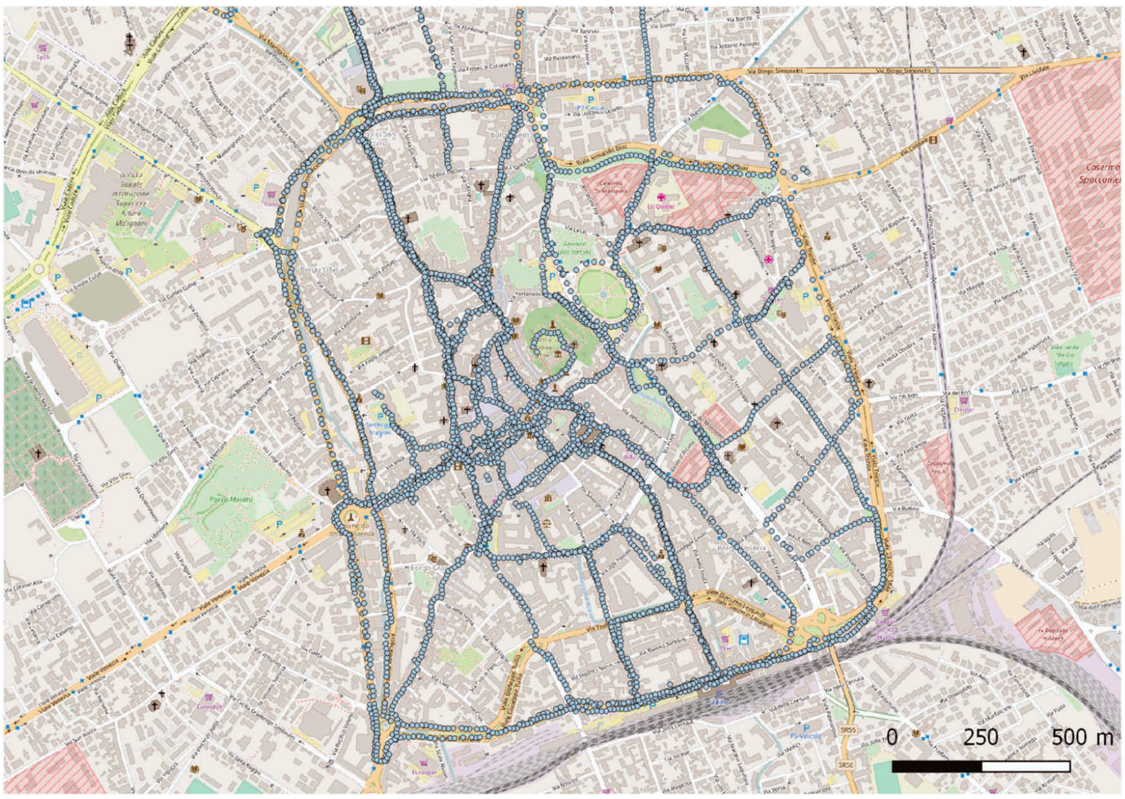
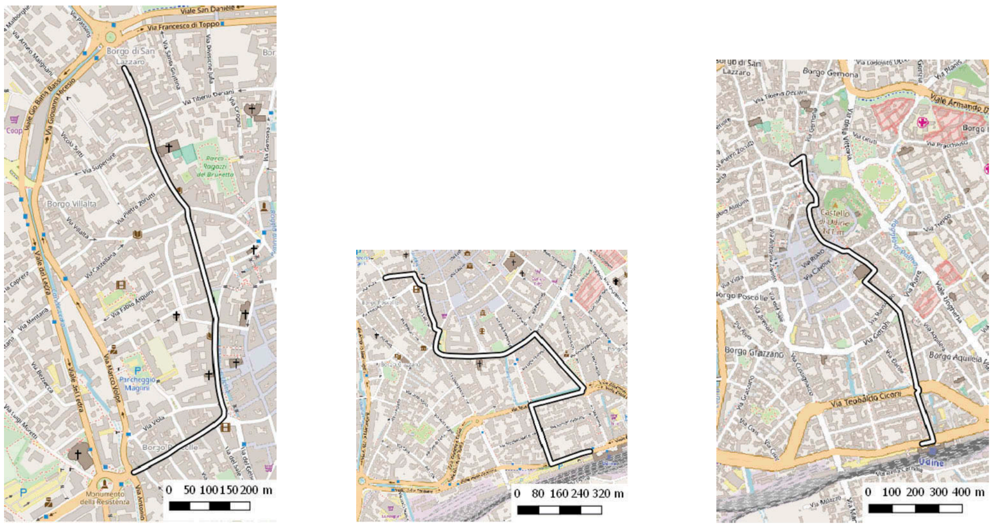


Figure 4. Fingerprint observations collected in the historical centre of Udine, an Italian city. Map Data © OpenStreetMap contributors, CC BY-SA.

Table 1. An example of the collected fingerprints (GPS data are omitted).

ID	Serving	Time	Cell Signature	Signal Strength
00	73896	17:46:03	73895,73896,74004,74091,75838	13,18,10,9,12
01	73896	17:46:18	73895,73896,74004,74091,75838,75841	15,19,12,13,13,12
02	73896	17:46:21	73895,73896,74004,74056,74091,75841	14,18,12,13,14,12
03	73896	17:46:25	73895,73896,74004,74056,74091,75861	13,19,12,12,13,12
04	73896	17:46:28	73895,73896,74004,74091,75838,75861	14,18,12,11,11,13
05	73896	17:46:31	73895,73896,74004,74091,75838,75861	14,18,12,10,11,13
06	73896	17:46:35	73895,73896,74004,74091,74685,75861	14,17,12,11,10,13
07	73896	17:46:39	73895,73896,74004,74056,74091,75861	14,18,14,13,14,13
08	73896	17:46:42	73895,73896,74004,74056,74091,75861	15,19,14,15,14,12
09	73896	17:46:46	73895,73896,74056,74091,75838,75841	14,22,12,13,13,12
10	73896	17:46:48	73896,74091,75838,75841	21,13,14,14



(a) Trace A (b) Trace B (c) Trace C

Figure 5. The three paths between different points of interests collected in the city of Udine © OpenStreetMap contributors, CC BY-SA.

Table 2. Characteristics of the traces used in the evaluation.

Trace	A	B	C
Length (m)	1227	1846	1722
Time (min)	12	9	6
Observations	112	111	101
Observations (30s)	22	16	13
Observations (60s)	12	8	7
Observations (120s)	6	5	4

a middle ground between car driving and walking. On the one hand, a cyclist is normally slower than a car, apart from the case of abnormal traffic situations, and thus the travelled distance is shorter, which helps map matching in the presence of sparse information. On the other hand, a cyclist has more freedom of movement, being able to ride his/her bike in several streets and paths that a car simply cannot access. Finally, as pointed out in Section 3, knowledge about the travelling method can be used to restrict the possible routes, resulting in an improvement in both running time and accuracy. The fact that all the paths were traversed by bike allowed us to exclude during computations some of the roads, like the ones classified as foot-way or steps.

The above data was imported in a PostgreSQL (PostgreSQL Global Development Group 2008) relational DataBase Management System (DBMS). To deal with spatial data, we used its spatial extension PostGIS (PostGIS Project Steering Committee 2018). Information about the road network, for the area under consideration, was obtained from OpenStreetMap. Once information about the relevant points on the road was computed, the additional library pgRouting for PostgreSQL (pgRouting Community 2018) has been exploited to determine the shortest paths between pairs of points of the road network. In particular, we used Dijkstra's algorithm to compute the shortest paths between road points representing the states in the model.

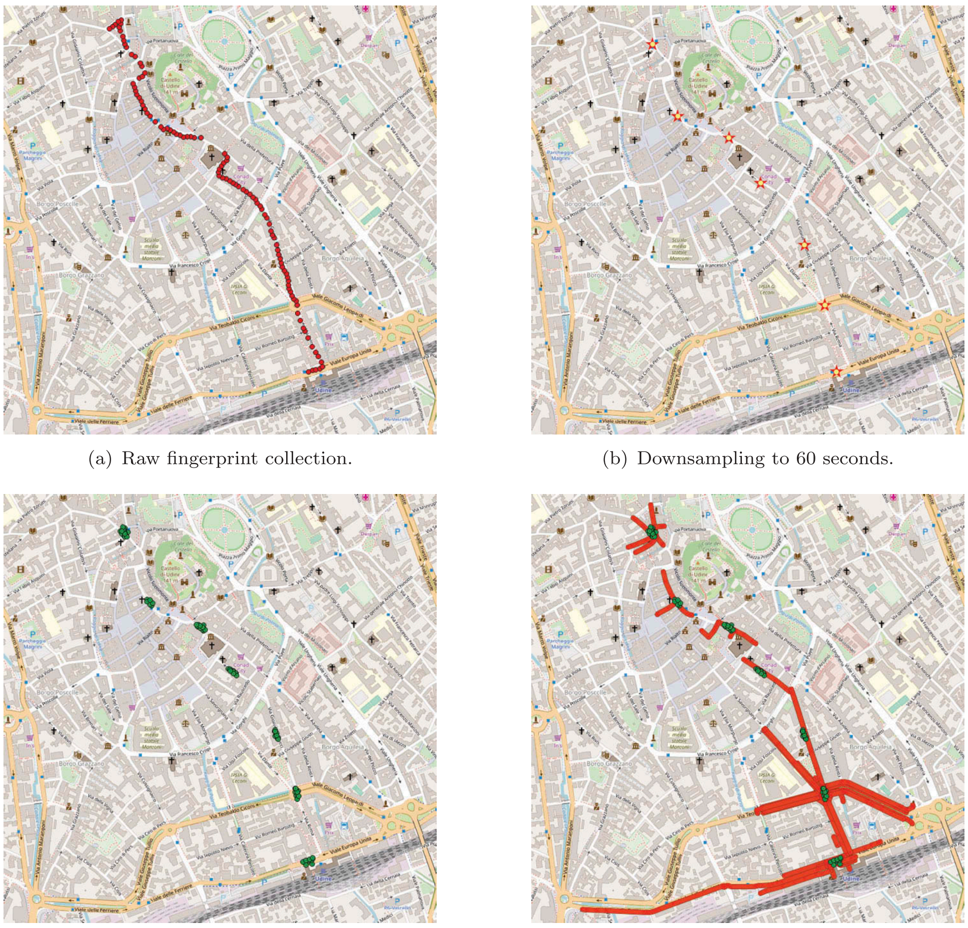
5. The execution of the proposed map-matching algorithm

The algorithm described in Section 3 has been executed on the dataset given in Section 4 (Figure 5), including three extracted test paths and a set of GPS-tagged observations. For each path, we considered different sampling frequencies.

To explain how the algorithm works, we illustrate in detail some of its steps with respect to one of the testing paths, namely, the trace C. The trace is composed of 122 fingerprints, collected at the locations shown in Figure 6(a) (red points). To simulate the sparse observation scenario, we retain one observation per minute. The resulting picture is depicted in Figure 6(b) (red-yellow stars), which only contains 8 observations. Obviously, the GPS positions of path C are not considered by the algorithm; they are only used to show how it works.

The first step involves the preprocesssing of the sequence of observations for removing the unnecessary or erroneous data. This is crucial step, especially for high sampling rates, for which observations which are unnecessarily close together may generate too many states. In the considered example, however, where the sampling period is 60 seconds, such a filtering step turned out not to be necessary.

Then, each fingerprint from the processed sequence is compared to those stored into the database using the Random Forest model to identify the most similar ones (Section 3). The training of the Random Forest was done over the fingerprint dataset by taking a random subset of all the observations and



(a) Raw fingerprint collection.

(b) Downsampling to 60 seconds.

(c) Fingerprint comparison.

(d) Roads for state selection.

Figure 6. Some intermediate steps of the processing on the trace C, leaving only one observation every 60 seconds. Map Data © OpenStreetMap contributors, CC BY-SA.

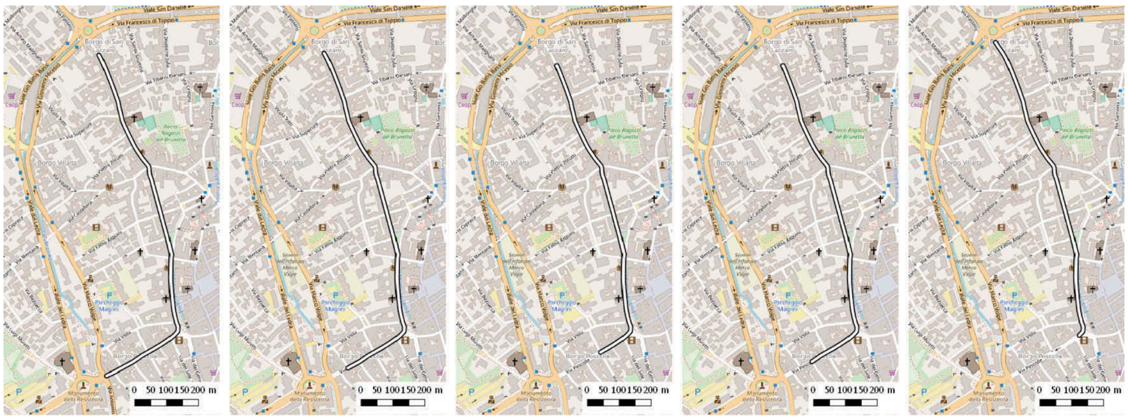
coupling them with other fingerprints to obtain pairs. Then, from each pair of fingerprints, we extracted the features that are fed to the tree, which is trained to predict the distance in meters between the positions where they were collected. The result of the comparison process is shown in Figure 6(c), where the fingerprints closest to observations are highlighted as green points.

On the basis of the estimated distances, the algorithm extracts the candidate roads, which are highlighted in Figure 6(d). Next, matching points on the road segments are computed, thus obtaining the states of HMM with the associated emitted probabilities. As a matter of fact, all the considered observations ended up with at least one state.

Then, in order to assign a transition probability to each pair of consecutive states, the algorithm searches for the shortest route between each state and all the next ones in the sequence.

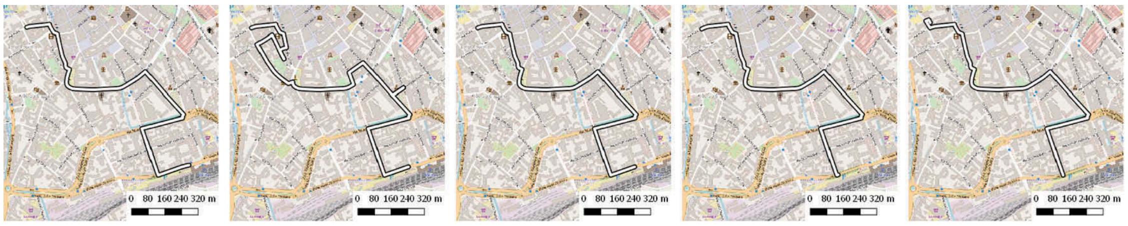
The last step of the algorithm is the reconstruction of the trajectory by means of the generated HMM. This is done by executing the Viterbi algorithm on the model, which produces the most likely path among the states. The path generated by the algorithm in the considered scenario (Figure 6(b)) is shown in Figure 9(d).

The results of the execution of the algorithm on the three paths at different sampling rates are reported in Figures 7, 8, and 9. Trace A is the shortest and straightest one among the three traces. In Figure 7, it is possible to see that the ground truth and the generated trajectories are quite similar for all the sampling frequencies. In particular, notice that the reduction of the frequency did not affect the shape of the trajectory, apart from shortening the path thanks to the initialization of the algorithm. Trace B, reported in Figure 8, is more twisted and longer than trace A. The path generated with sampling intervals at 30 seconds is the one that fits best to the original track. Increasing the sample rate probably introduces noise that produces some deviations from the original path as a result of trying to include some outliers. On the other hand, reducing the sample rate up to 120 seconds did not degrade too much the reconstruction of the trajectory. Trace C, depicted in Figure 9, goes from the train station to a university building, passing through the city centre. It is longer than the other two traces. In this case, when moving from 30 seconds sampling onward to 120 seconds, the generated path fits quite well with respect to the ground truth. Increasing the sampling frequency over 30 seconds produces minor outliers that can be explained as a detour to reach some



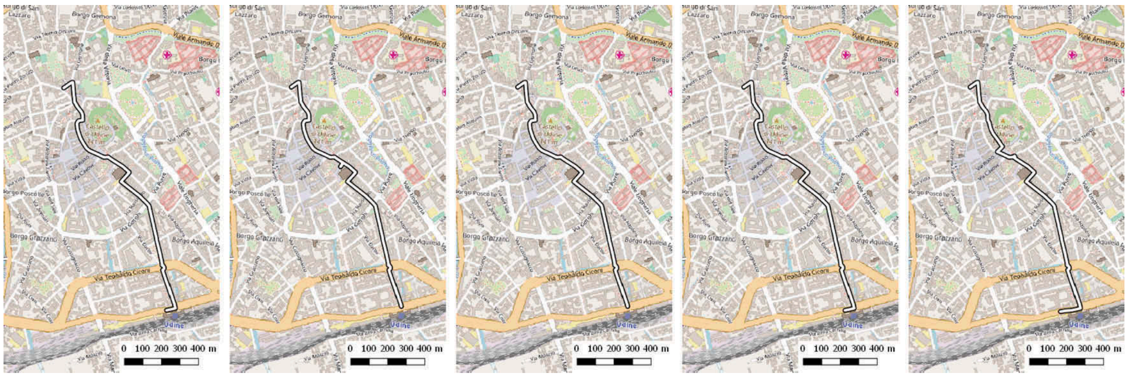
(a) Ground truth (b) Sampling 0s (c) Sampling 30s (d) Sampling 60s (e) Sampling 120s

Figure 7. Path A: ground-truth and map-matched trajectory at various samplings. The path goes from the top to the bottom. Map Data © OpenStreetMap contributors, CC BY-SA.



(a) Ground truth (b) Sampling 0s (c) Sampling 30s (d) Sampling 60s (e) Sampling 120s

Figure 8. Path B: ground-truth and map-matched trajectory at various sampling rates. The path goes from the top to the bottom. Map Data © OpenStreetMap contributors, CC BY-SA.



(a) Ground truth (b) Sampling 0s (c) Sampling 30s (d) Sampling 60s (e) Sampling 120s

Figure 9. Path C: ground-truth and map-matched trajectory at various samplings. The path goes from the bottom to the top. Map Data © OpenStreetMap contributors, CC BY-SA.

candidate points quite close to the trajectory, but not really on the original path.

We would like to conclude the section with some remarks about the computational efficiency of the positioning algorithm we exploited, which is one of distinctive features of the proposed solution. As shown in detail in Viel et al. (2018), in comparison to the classic Euclidean one, the Random Forest positioning method is an order of magnitude slower in computing a single distance estimation, but it turns out to be more accurate. Moreover, the time required by the Random Forest to compute the estimation (one millisecond) is definitely exceeded by the cost of querying the database to retrieve the relevant fingerprints, leading to an overall cost of about 30 milliseconds.

6. Experimental evaluation

In order to evaluate the behaviour of the proposed algorithm on test data, we used some metrics to analyse the accuracy and assess whether it changes when lowering the sampling rate (Section 6.1).

Moreover, to investigate the influence of changes in the quantity and the distribution of the stored fingerprints on the behaviour of the system, we also simulated the removal of some of them from the original GPS-tagged fingerprint dataset (Section 6.2). Finally, we tested a mixed solution where fingerprint observations are randomly enriched with sparse GPS observations (Section 6.3).

6.1. Accuracy results

The accuracy evaluation of the algorithm was done using two standard metrics, namely, precision and recall, respectively denoted by P and R, which are defined as follows:

$$P = \frac{\text{length of the correct matched segments}}{\text{length of the generated trajectory}} \tag{5}$$

$$R = \frac{\text{length of the correct matched segments}}{\text{length of the ground truth trajectory}} \tag{6}$$

These metrics are commonly used in map matching as they focus on the similarity between the generated trajectory and the one actually followed by the device, rather than looking at the position where each observation is matched (Jagadeesh and Srikanthan 2015; Thiagarajan et al. 2011).

Table 3. Accuracy of the map-matched trajectory using fingerprint observations.

Trace	Sampling time (s)	P (%)	R (%)
A	0	100	96
	30	100	88
	60	100	92
	120	100	80
B	0	62	80
	30	100	97
	60	99	90
	120	96	90
C	0	98	95
	30	100	95
	60	100	99
	120	83	85

Precision and recall values are reported in Table 3. Interestingly, degrading the sequences by decreasing the sampling does not always change their values significantly. In a few cases, such a decrease produces the opposite effect, like in path B where the lowest precision (62%) is obtained when the sampling is the highest. The effect is even more evident in Figure 10, which describes the impact of the changes in the sampling period of the sequences on the recall. Notice that this supports the choice of removing redundant fingerprints during the preprocessing step.

We conclude by commenting on the spatial characteristics of the generated paths. In almost all the cases, the most noticeable error in the trajectories is that they result in shortening of the ends. This is probably due to the fact that a shorter trajectory usually has a higher probability in the model, and can thus favour this kind of behaviour at the extremes of the path.

6.2. Influence of the fingerprint map size

Let us now briefly discuss the influence of the fingerprint map size on map matching. The algorithm described in the previous sections makes use of a fingerprint database which maps quite densely the city centre of Udine, obtaining a map-matched trajectory very similar to the one actually followed by the device. However, assuming a similar density for a larger region can be unrealistic, as there can certainly be areas which have a lower number of fingerprints stored in the database.

To simulate such a scenario, where a sparse fingerprint map is available, the map-matched trajectories were recomputed multiple times, using increasingly larger random subsets of all the fingerprints available in the database. We set three thresholds at 50%, 70% and 100% of the full database. In Table 4, we show the resulting precision and recall

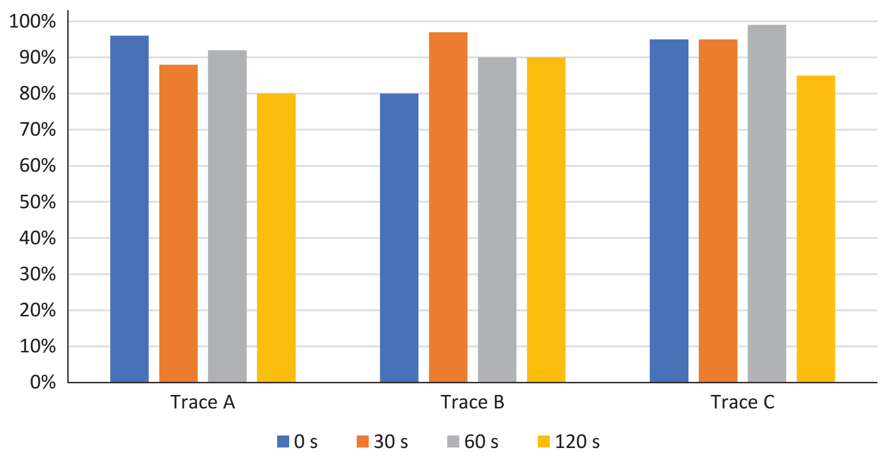


Figure 10. How increasing the sampling time impacts on the recall measures for the different traces.

Table 4. Accuracy of the map-matched trajectory with different subsets of fingerprints.

		Trace A		Trace B		Trace C	
Sampling time (s)	Subset (%)	*P* (%)	*R* (%)	*P* (%)	*R* (%)	*P* (%)	*R* (%)
	50	100	100	71	83	98	95
0	75	100	100	65	80	98	95
	100	100	96	62	80	98	95
	50	100	92	80	76	100	90
30	75	100	88	64	83	100	95
	100	100	88	100	97	100	95
	50	100	88	99	90	73	59
60	75	100	92	73	90	100	99
	100	100	92	99	90	100	99
	50	99	80	66	83	87	86
120	75	99	80	95	90	87	85
	100	100	80	98	90	83	85

values. It is worth pointing out that removing random fingerprints from the map is stronger than removing them uniformly. The random removal of fingerprints is indeed better for testing the robustness of the algorithm in the case of a fingerprint map with low density.

To make it easier to follow, Figure 11 shows the recall values for increasingly larger subsets of the fingerprint map, maintaining the sampling period fixed at 120 seconds. Looking at these values, it is possible to notice that there is no strong correlation between the fingerprint map size and the accuracy of the trajectory. Even with just half of the fingerprint map available, the accuracy of the generated trajectory remains basically unchanged. In most cases, the reported values are very similar and, in certain situations, a larger subset produced a worse trajectory.

6.3. Using mixed GPS and fingerprint measurements

The proposed map-matching algorithm is flexible enough to deal also with mixed sequences of cellular fingerprints and GPS observations. In general, obtaining a GPS fix is battery-wise more costly than a passively collected fingerprint. However, GPS observations are particularly useful when the device is moving in an area with no cellular radio coverage, or when there are no matching fingerprints to compare within the database.

Hence, we decided to check whether a small number of GPS observations can improve the accuracy of the map-matched trajectories. To this end, we randomly substituted about a third of the fingerprints of each sequence with the corresponding GPS positions and run the map-matching algorithm on the modified data. Table 5 shows the accuracy results in the case of the mixed sequence.

In principle, GPS observations should work as high-precision anchors reducing the uncertainty in HMM. Hence, the expected result was a better estimation of the trajectory. As a matter of fact, that was not the case: compared to the results in Table 3, the improvement obtained from a mixed sequence is modest. Moreover, in most cases, the precision and recall results do not change. We can interpret such an outcome as a further confirmation of the quality of the output of the algorithm when applied to sequences of fingerprints only. The added GPS points would have indeed changed completely the resulting trajectories if they were off from the real path. As

Table 5. Accuracy of the map-matched trajectory using mixed GPS/fingerprint observations.

Trace	Sampling time (s)	*P* (%)	*R* (%)
A	0	94	96
	30	100	88
	60	100	92
	120	100	88
B	0	70	85
	30	100	97
	60	100	90
	120	77	97
C	0	100	95
	30	100	95
	60	96	99
	120	87	81

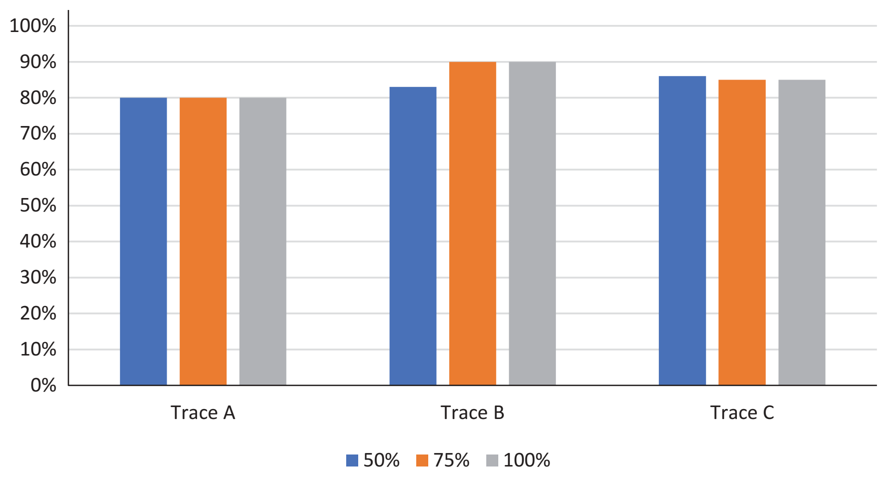


Figure 11. Recall measures at 120 seconds of sampling for the different traces while taking increasingly larger subsets of the fingerprint map.

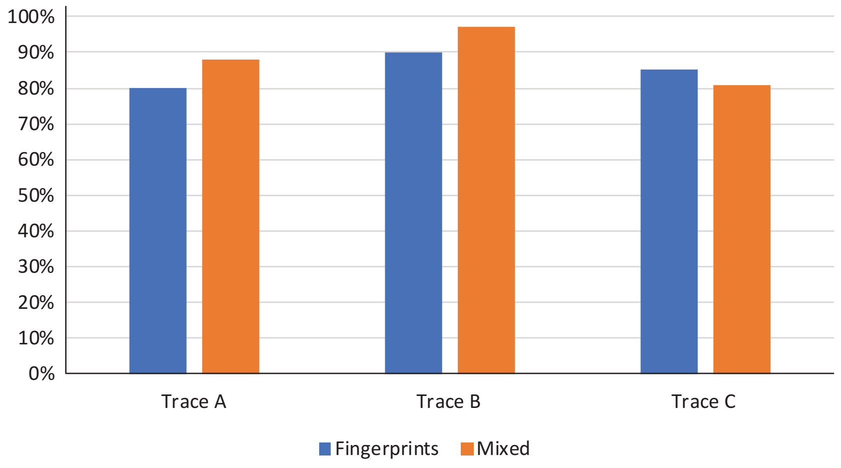


Figure 12. Recall measures at 120 seconds of samplings for mixed sequence (in orange) and fingerprint only sequence (in blue).

depicted in Figure 12, the Recall for the 120 seconds sampling with mixed map matching is even lower in the trace C. This is probably related to the outcomes of adding a new GPS point, that forces a detour.

7. Conclusions

In this paper, we proposed an original map-matching algorithm aiming at coping with sparse cellular fingerprint observations. The algorithm tries to reconstruct the path of mobile devices exploiting only the signals received from the cellular networks. It presents some similarities to recent research contributions focusing on the problem of figuring out the path followed by a device on the basis of sparse GPS observations. However, it only exploits cellular fingerprint observations. Cellular fingerprints, unlike GPS measurements, do not contain explicit spatial information and thus cannot be easily mapped into geographical coordinate pairs. The main contribution of the paper is a novel technique for the generation of HMM states exploiting cellular fingerprints, without the need of converting them into a single location. To assess the similarity among fingerprints, it makes use of a state-of-the-art machine learning approach based on decision tree ensembles.

The proposed map-matching algorithm was tested on the urban environment of the city centre of Udine, a medium-sized Italian city. The outcomes of the experimental evaluation showed that it is actually possible to reconstruct the trajectory of a moving device with high accuracy using just sparse fingerprint cellular observations. Moreover, it showed that increasing progressively the sampling interval for the sequence of observations, the accuracy of the map-matched trajectory does not exhibit a significant worsening. A possible explanation is that the accuracy of fingerprinting is not directly influenced by the sparsity of observations. Indeed, in our testing scenario, we experienced that the trajectory of a device can be reconstructed by means of a small number of observations, a higher amount of possibly noisy measurements being often useless. The evaluation phase also tested the impact of varying the density of the fingerprint map on the generated path. The experimental results demonstrated that it is not strictly necessary to have a dense fingerprint map to obtain a good accuracy. Indeed, even using a small subset (even half) of the available data, the generated trajectory showed an accuracy similar to the one obtained by using the full fingerprint map. Finally, we considered the case in which one deals with sparse observations mixing fingerprints and GPS fixes. The experiments showed that, in most of the cases, the insertion of a small number of GPS observations did not change the overall resulting trajectory, which was always quite aligned to the ground truth.

As for future work, we are thinking of further testing the proposed map-matching algorithm in other scenarios, e.g. on larger areas. The experimental evaluation involved a small portion of a city, and the algorithm took a reasonable amount of resources to compute a trajectory. It would be interesting to check whether the algorithm scales well when copying with a larger area and a high number of nodes to process. Since the experimental results made use of observations taken from a single device, it would be interesting to evaluate also the impact of the use of a dataset where contributions are collected by a set of different devices. Finally, the availability of data coming from several devices makes it possible to compare the generated trajectory to the previously stored ones. Such a possibility has been already explored in the literature in the GPS case, while, to the best of our knowledge, it has not been systematically addressed in the case of cellular fingerprints.

Notes on contributors

Andrea Dalla Torre got his PhD at the University of Padova (Italy) in 2004 and he is currently manager in the

Location Service Team in u-blox Italia S.p.A. His work is focusing on the positioning feature in cellular modems.

Paolo Gallo obtained his MSc in Computer Science from the University of Udine (Italy) in 2009. Currently, he is research fellow at the University of Udine. His main research interests are in data science and geographical information systems.

Donatella Gubiani received her PhD at the University G. d'Annunzio of Chieti-Pescara (Italy) in 2008 and she is currently a researcher in Computer Science at the University of Nova Gorica (Slovenia). Her research interests mainly focus on data science and spatiotemporal databases.

Chris Marshall graduated from Cambridge University in 1980, and then gained his PhD at Imperial College, London (UK). His research and development interests have spanned wireless and positioning technologies, and he is currently a senior principal engineer at u-blox in the UK.

Angelo Montanari got his PhD at the University of Amsterdam (The Netherlands) on 1996 and he is currently full professor of Computer Science at the University of Udine (Italy). His major research interests are in formal methods, AI, and spatiotemporal databases.

Federico Pittino received his PhD in co-tutelle between the University of Udine (Italy) and the Institut Polytechnique de Grenoble (France) in 2015. He has worked at u-blox on localisation algorithms and is currently a researcher at Carinthian Tech Research (Austria) focusing on machine learning, AI, and data science.

Andrea Viel received his PhD at the University of Udine (Italy) in 2019 and he is currently working at u-blox Italia S.p.A. on location-based services. His research interests are in data science and spatiotemporal databases.

ORCID

Donatella Gubiani http://orcid.org/0000-0001-7684-5200
Federico Pittino http://orcid.org/0000-0003-3248-5391

References

Algizawy, E., T. Ogawa, and A. El-Mahdy. 2017. “Real-Time Large-Scale Map Matching Using Mobile Phone Data.” *ACM Transactions on Knowledge Discovery from Data* 11 (4): 52:1–52:38.

Aly, H., and Y. Moustafa. 2013. “Dejavu: An Accurate Energy-Efficient Outdoor Localization System.” In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM-GIS)*, Orlando, USA, November 5–8.

Aly, H., and Y. Moustafa. 2015. “semMatch: Road Semantics-Based Accurate Map Matching for Challenging Positioning Data.” In *Proceedings of the 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS)*, Seattle, Washington, November 3–6.

Bahl, P., and V. N. Padmanabhan. 2000. “RADAR: An in-Building RF-based User Location and Tracking System.” In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Tel Aviv, Israel, March 26–30.

Benikovsky, J., P. Brida, and J. Machaj. 2010. “Localization in Real GSM Network with Fingerprinting Utilization.” In *Proceedings of the 2nd International Conference on Mobile Lightweight Wireless Systems (MOBILIGHT)*, Barcelona, Spain, May 10–12.

Breiman, L. 2001. “Random Forests.” *Machine Learning* 45 (1): 5–32.

Chen, M. Y., T. Sohn, D. Chmelev, D. Haehnel, J. Hightower, J. Hughes, A. LaMarca, F. Potter, I. Smith, and A. Varshavsky. 2006. “Practical Metropolitan-Scale Positioning for GSM Phones.” In *Proceedings of the 8th International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, Orange County, CA, September 17–21.

Deblauwe, N. 2008. *GSM-based Positioning: Techniques and Applications*. Belgium: Vrije Universiteit Brussel .

Forney, G. D. 1973. “The Viterbi Algorithm.” *Proceedings of the IEEE* 61 (3): 268–278.

Goh, C. Y., J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet. 2012. “Online MapMatching Based on Hidden Markov Model for Real-Time Traffic Sensing Applications.” In *Proceedings of the 15th International Conference on Intelligent Transportation Systems (ITSC)*, Anchorage, AK, September 16–19.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. “The WEKA Data Mining Software: An Update.” *ACM SIGKDD Explorations Newsletter* 11 (1): 10–18.

Hashemi, M., and H. A. Karimi. 2014. “A Critical Review of Real-Time Map-Matching Algorithms: Current Issues and Future Directions.” *Computers, Environment and Urban Systems* 48: 153–165.

Hoy, J. 2015. *Forensic Radio Survey Techniques for Cell Site Analysis*. Chichester: John Wiley & Sons.

Jagadeesh, G. R., and T. Srikanthan. 2015. “Probabilistic Map Matching of Sparse and Noisy Smartphone Location Data.” In *Proceedings of the 18th International Conference on Intelligent Transportation Systems (ITSC)*, Las Palmas de Gran Canaria, Spain, September 15–18.

Juang, B. H., and L. R. Rabiner. 1991. “Hidden Markov Models for Speech Recognition.” *Technometrics* 33 (3): 251–272.

Jurafsky, D., and J. H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. New Jersey: Prentice Hall.

Kjærgaard, M. B., and C. V. Munk. 2008. “Hyperbolic Location Fingerprinting: A Calibration- Free Solution for Handling Differences in Signal Strength (Concise Contribution).” In *Proceedings of the 6th International Conference on Pervasive Computing and Communications (PerCom)*, Hong Kong, March 17–21.

Leontiadis, I., A. Lima, H. Kwak, R. Stanojevic, D. Wetherall, and K. Papagiannaki. 2014. “From Cells to Streets: Estimating Mobile Paths with Cellular-Side Sata.” In *Proceedings of the 10th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, Sydney, Australia, December 2–5.

Li, X., X. Zhang, K. Chen, and S. Feng. 2014. “Measurement and Analysis of Energy Consumption on Android Smartphones.” In *Proceedings of the 4th International Conference on Information Systems and Technologies (ICIST)*, Valencia, Spain, March 22–24.

Lou, Y., C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. 2009. “Map-Matching for Low- Sampling-Rate GPS Trajectories.” In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic*

Information Systems (ACM-GIS), Seattle, WA, November 4–6.

Newson, P., and J. Krumm. 2009. “Hidden Markov Map Matching through Noise and Sparseness.” In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM-GIS)*, Seattle, WA, November 4–6.

Oran, A., and P. Jaillet. 2013. “An HMM-based Map Matching Method with Cumulative Proximity-Weight Formulation.” In *Proceedings of the 2nd International Conference on Connected Vehicles and Expo (ICCVE)*, Las Vegas, NV, December 2–6.

Osogami, T., and R. Raymond. 2013. “Map Matching with Inverse Reinforcement Learning.” In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, Beijing, China, August 3–9.

Paek, J., K.-H. Kim, J. P. Singh, and R. Govindan. 2011. “Energy-Efficient Positioning for Smartphones Using Cell-ID Sequence Matching.” In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Washington, DC, June 28-July 1.

pgRouting Community. 2018. “pgRouting Project.” Accessed 16 November 2018. https://pgrouting.org

PostGIS Project Steering Committee. 2018. “PostGIS Spatial and Geographic Objects for PostgreSQL.” Accessed 16 November 2018. http://www.postgis.net

PostgreSQL Global Development Group. 2008. “PostgreSQL.” Accessed 16 November 2018. http://www.postgresql.org

Quddus, M. A., W. Y. Ochieng, and R. B. Noland. 2007. “Current Map-Matching Algorithms for Transport Applications: State-Of-The Art and Future Research Directions.” *Transportation Research Part C: Emerging Technologies* 15 (5): 312–328.

Schulze, G., C. Horn, and R. Kern. 2015 “Map-Matching Cell Phone Trajectories of Low Spatial and Temporal Accuracy.” In *Proceedings of the 18th International Conference on Intelligent Transportation Systems (ITSC)*, Las Palmas de Gran Canaria, Spain, September 15–18.

Thiagarajan, A., L. Ravindranath, H. Balakrishnan, S. Madden, L. Girod. 2011. “Accurate, Low-Energy Trajectory Mapping for Mobile Devices.” In *Proceedings of the 8th Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, MA, March 30-April 1.

Thiagarajan, A., L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. 2009. “VTrack: Accurate, Energy-Aware Road Traffic Delay Estimation Using Mobile Phones.” In *Proceedings of the 7th Conference on Embedded Networked Sensor Systems (SenSys)*, Berkeley, CA, November 4–6.

Ulm, M., P. Widhalm, and M. Brndle. 2015. “Characterization of Mobile Phone Localization Errors with OpenCellID Data.” In *Proceedings of the 4th International Conference on Advanced Logistics and Transport (ICALT)*, Valenciennes, France, May 20–22.

Viel, A., A. Brunello, A. Montanari, and F. Pittino. 2018. “An Original Approach to Positioning with Cellular Fingerprints Based on Decision Tree Ensembles.” In *Proceedings of the 14th International Conference on Location Based Services (LBS)*, Zurich, Switzerland, January 15–17.

Yuan, J., Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun. 2010. “An Interactive-Voting Based Map Matching Algorithm.” In *Proceedings of the 11th International Conference on Mobile Data Management (MDM)*, Kansas City, MO, May 23–26.

Zekavat, R., and R. M. Buehrer. 2011. *Handbook of Position Location: Theory, Practice and Advances*. 1st ed. New Jersey: Wiley-IEEE Press.

Zhuang, Z., K.-H. Kim, and J. P. Singh. 2010. “Improving Energy Efficiency of Location Sensing on Smartphones.” In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, June 14–18.