The path inference filter: model-based low-latency map matching of probe vehicle data

Timothy Hunter, Pieter Abbeel, and Alexandre M. Bayen

Abstract

We consider the problem of reconstructing vehicle trajectories from sparse sequences of GPS points, for which the sampling interval ranges between 10 seconds and 2 minutes. We introduce a new class of algorithms, collectively called *path inference filter* (PIF), that maps streaming GPS data in real-time, with a high throughput. We present an efficient Expectation Maximization algorithm to train the filter on new data without ground truth observations. The path inference filter is evaluated on a large San Francisco taxi dataset. It is deployed at an industrial scale inside the *Mobile Millennium* traffic information system, and is used to map fleets of vehicles in San Francisco, Sacramento, Stockholm and Porto.

1 Introduction

The paradigm of connected vehicles, the progressive integration of smartphones and car infrastructure, the rise of Web 2.0, and the progressive emergence of automation onboard vehicles have created a very fertile ground for GPS data sources from probe vehicles to be collected at an unprecedented scale. Yet, the lack of ubiquitous connectivity, the parsimonious use of bandwidth and smartphone battery limits, and the lack of system reliability often makes this data sparse. In particular, it is very common common today for GPS traces to be sampled at very low frequencies (on the order of minutes), leading to challenges in using this data. While some studies [13] predict an 80% penetration of the cellphone market by GPS enabled devices by 2015, it is not clear that this figure will translate into ubiquitous GPS data for traffic information systems. Furthermore, millions of fleet vehicles today produce

Timothy Hunter · Pieter Abbeel · Alexandre M. Bayen
Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, e-mail: tjhunter,pabbeel,bayen@eecs.berkeley.edu

GPS traces sampled at low frequencies, see for example [4], and the paradigm of connected (and automated) vehicles does not automatically translate into high fidelity GPS traces.

Two of the common problems which occur when dealing with these GPS traces are the correct mapping of these observations to the road network, and the reconstruction of the trajectories of the vehicles from these traces. This problem is similar to numerous other problems also encountered in robotics, for example tracking of *people of interest* (POI) by the military, in which POIs disappear between sparse samples and analysts must reconstruct the path followed.

We present a new class of algorithms, called the *path inference filter*, that solve this problem in an efficient way. There are two difficulties associated with this problem. First, there may be many possible projections of the noisy position onto the map. Second, the number of possible paths between consecutive pairs of positions is potentially very high (given that in urban environments, a vehicle could typically travel several blocks between measurements). Instead of handling both problems separately, as is commonly done in traffic modeling studies (which leads to significant pitfalls), the method solves both problems at once, which increases the efficiency of the algorithm and uses the conjunction of both unknowns (projections and paths) in a unified manner to resolve this inference problem.

Specific instantiations of this algorithm have been deployed as part of *Mobile Millennium*, which is a traffic estimation and prediction system developed at UC Berkeley [2]. *Mobile Millennium* infers real-time traffic conditions using GPS measurements from drivers running cell phone applications, taxicabs, and other mobile and static data sources. This system was initially deployed in the San Francisco Bay area and later expanded to other locations. In addition to GPS information, the data points collected by *Mobile Millennium* contain other attributes such as heading, speed, etc. We will show how this additional information can be integrated in the rest of the framework presented into this article.

In the case of high temporal resolution (typically, a frequency greater than an observation per second), some highly successful methods have been developed for continuous estimation [16]. In the case of low frequency sampled data, simple deterministic algorithms to reconstruct trajectories fail due to misprojection or shortcuts (Figure 1). Such shortcomings have motivated our search for a principled approach that jointly considers the mapping of observations to the network and the reconstruction of the trajectory.

Related work Researchers started systematic studies after the introduction of the GPS system to civilian applications in the 1990s [12]. Early geometry projection approaches were later refined to use more information such as heading and road curvature. This greedy matching, however, leads to poor trajectory reconstruction since it does not consider the path leading up to a point. New deterministic algorithms emerged to directly match partial trajectories to the road by using the topology of the network [6], and were soon

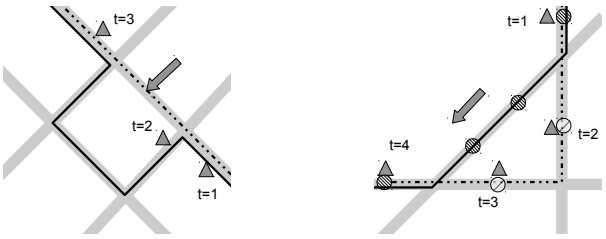

Fig. 1 Examples of failures when using intuitive algorithms to project each GPS measurement to the closest link. The raw GPS measurements are the triangles, the true trajectory is the dashed line, and the reconstructed trajectory is the continuous line. The hashed circles are the states chosen by this reconstruction algorithm. *(a)* The trajectory is reconstructed using a closest-point, shortest path algorithm. Due to GPS noise, the point at $t = 2$ is closer to the orthogonal road and forces the algorithm to add a wring left turn. *(b)* Example of error when considering the shortest paths with multiple potential projections.

expanded into probabilistic frameworks. A number of implementations were explored, among other particle filters [7], Kalman filters [11], and Hidden Markov Models [3].

Two types of information are missing in a sequence of GPS readings: the exact location of the vehicle on the road network when the observation was emitted, and the path followed from the previous location to the new location. These problems are correlated. The aforementioned approaches focus on high-frequency sampling observations, for which the path followed is extremely short (less than a few hundred meters, with very few intersections). In this context, there is usually a dominant path that starts from a well-defined point, and Bayesian filters accurately reconstruct paths from the observations [11, 16]. When sampling rates are lower and observed points are further apart, however, a large number of paths are possible between two points. Researchers have recently focused on efficiently identifying these correct paths and have separated the joint problem of finding the paths and finding the projections into two distinct problems. The first problem is path identification and the second step is projection matching [3, 15]. Some interesting approaches mixing points and paths that use a voting scheme have also recently been proposed [19].

Contributions of the article The *path inference filter* is a probabilistic framework that aims at recovering trajectories and road positions from low-frequency probe data in real time. The performance of the filter degrades gracefully as the sampling frequency decreases, and it can be tuned to different scenarios (such as real time estimation with limited computing power or offline, high accuracy estimation).

The filter falls into the general class of *Conditional Random Fields* [9]. Our framework can be decomposed into the following steps:

- *Map matching*: each GPS measurement from the input is projected onto a set of possible candidate states on the road network.

- *Path discovery:* admissible paths are computed between pairs of candidate points on the road network.
- *Filtering*: probabilities are assigned to the paths and the points using both a stochastic model for the vehicle dynamics and probabilistic driver preferences learned from data.

The path inference filter presents a number of compelling advantages over the work found in the current literature:

1. The approach presents a general framework grounded in established statistical theory that encompasses numerous approaches presented in the literature. In particular, it combines information about paths, points and network topology in a single unified notion of *trajectory.*
2. Nearly all work on Map Matching is segmented into (and presents results for) either high-frequency or low-frequency sampling. After training, the path inference filter performs competitively across all frequencies.
3. As will be seen in Section 3, most existing approaches (which are based on Hidden Markov Models) do not work well at lower frequencies due to the *selection bias problem.* Our work directly addresses this problem by performing inference on a Random Field.
4. The path inference filter can be used with complex path models such as the ones used in [3]. In the present article, we restrict ourselves to a class of models (the exponential family distributions) that is rich enough to provide insight in the driving patterns of the vehicles. Furthermore, when using this class of models, the learning of new parameters leads to a convex problem formulation that is fast to solve. In addition, this algorithm efficiently learns parameters in an unsupervised setting.
5. With careful engineering, it is possible to achieve high mapping throughput on large-scale networks. The implementation in *Mobile Millennium* achieves an average throughput of hundreds of GPS observations per second on a single core in real time. Furthermore, the algorithm scales well on multiple cores and has achieved average throughput of several thousands of points per second on a multicore architecture.
6. The path inference filter is designed to work across the full spectrum of accuracy versus latency. As will be shown, approximate 2-lagged smoothing is nearly as precise as full smoothing: we can still achieve good mapping accuracy while delaying computations by only one or two time steps.

2 Path discovery

The road network is described as a directed graph $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ in which the nodes are the street intersections and the edges are the streets, referred to in the text as the *links* of the road network. Every location on the road network is completely specified by a given link l and a non-negative offset o on this link. At any time, the *state* x of a vehicle consists of its location on the road

network and some other optional information such as speed, or heading. For our example, we consider that the state is simply the location on one of the road links (which are directed): $x = (l, o)$.

From GPS points to discrete vehicle states The points are mapped to the road following a Bayesian formulation. This process is represented by a probability distribution $\omega(g|x)$ that, given a state x, returns a probability distribution over all possible GPS observations g. Such distributions ω will be described in Section 3. Additionally, we may have some *prior knowledge* over the state of the vehicle, coming for example from previous GPS observations. This knowledge can be encoded in a *prior distribution* $\Omega(x)$. Under this general setting, the state of a vehicle, given a GPS observation, can be computed using Bayes' rule: $\pi(x|g) \propto \omega(g|x)\,\Omega(x)$. The letter π will define general probabilities, and their dependency on variables will always be included. This posterior distribution is usually complicated, owing to the mixed nature of the state (links and offsets). Instead of working with the posterior density directly, we represent it by a discrete distribution over a few well-chosen representative locations: for each link l_i, one or more states from this link are elected to represent the posterior distribution of the states on this link $\pi(o|g, l = l_i)$. In our model, the distribution over the offsets is unimodal and can be approximated with a single sample taken at the most likely offset: $\forall l_i, \quad o^*_{i_{\text{posterior}}} = \text{argmax}_o \ \pi(o|g, l = l_i)$, which implicitly depends on the prior distribution Ω. In practice, the prior is slowly varying comparatively to the length of the link, and can be considered constant as a first approximation on each link. This is why we can approximate $o^*_{i_{\text{posterior}}}$ by the most likely offset with respect to the observation distribution: $o^*_{i_{\text{posterior}}} \approx o^*_{\text{obs}} = \text{argmax}_o\, \omega(x = (o, l_i)\,|g)$. This approximation is illustrated in Figure 2.

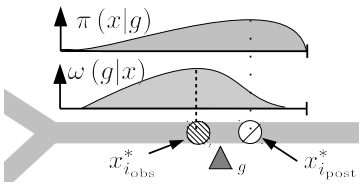

Fig. 2 Example of a measurement g on a link. The GPS measurement is the triangle denoted g.

In practice, the probability distribution $\pi(x|g)$ decays rapidly, and can be considered overwhelmingly small beyond a certain distance from the observation g. Links located beyond a certain radius need not be considered valid projection links, and may be discarded.

In the rest of the article, the boldface symbol $\mathbf{x}$ will denote a (finite) collection of states associated with a GPS observation g that we will use to represent the posterior distribution $\pi(x|g)$, and the integer I will denote its cardinality: $\mathbf{x} = (x_i)_{1:I}$. These points are called *candidate state projections for the GPS measurement* g. These discrete points will then be linked together through trajectory information that takes into account the trajectory and the dynamics of the vehicle.

From discrete vehicle states to trajectories At each time step t, a GPS point g^t is observed. This GPS point is then mapped onto I^t different candidate states denoted $\mathbf{x}^t = x^t_1 \cdots x^t_{I^t}$. Because this set of projections is

finite, there is only a finite number J^t of paths that a vehicle can have taken while moving from some state $x_i^t \in \mathbf{x}^t$ to some state $x_{i'}^{t+1} \in \mathbf{x}^{t+1}$. We denote the set of *candidate paths* between the observation g^t and the next observation g^{t+1} by $\mathbf{p}^t = \left(p_j^t\right)_{j=1:J^t}$. Each path p_j^t goes from one of the projection states x_i^t of g^t to a projection state $x_{i'}^{t+1}$ of g^{t+1}. There may be multiple pairs of states to consider, and between each pair of states, there are typically several paths available (see Figure 3). Lastly, a *trajectory* is defined by the succession of states and paths, starting and ending with a state: $\tau = x_1 p_1 x_2 \cdots p_{t-1} x_t$ where x_1 is one element of $\mathbf{x}^1$, p_1 of $\mathbf{p}^1$, and so on.

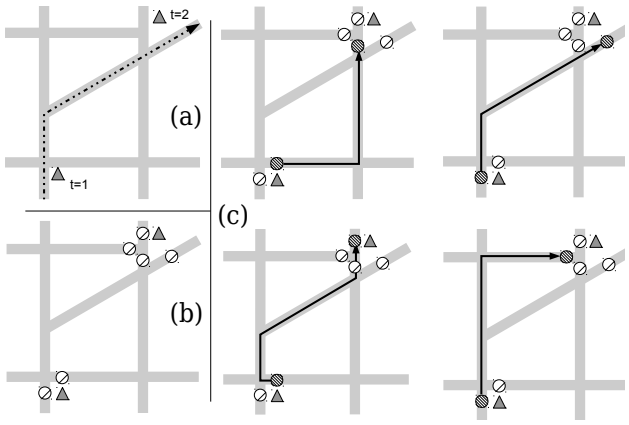

Fig. 3 Example of path exploration between two observations. On the upper left corner, the true trajectory and two associated GPS observations. On the lower left corner, the set of candidate projections associated with each observation. On the right, a few examples of computed paths.

Due to speed limits leading to upper bounds on achievable travel times on the network, there is only a finite number of paths a vehicle can take during a time interval Δt. Such paths can be computed using standard graph search algorithms. An algorithm that performs well in practice is the A* algorithm [8]. The cost metric we use here is the expected travel time on each link, and the heuristic is the shortest geographical distance, properly scaled so that it is an admissible heuristic. Due to the noise in the GPS observations, some failure cases need some special care. The supplementary materials available online (see the first page of the article) provide additional technical information about some ways to deal with these failure cases.

3 Discrete filtering using a Conditional Random Field

We have reduced the trajectory reconstruction problem to a discrete selection problem between candidate projection points interleaved with candidate paths. We now apply a probabilistic framework to infer the most likely trajectory τ^* or the marginal conditionals $\pi\left(x^t|g^{1:T}\right)$ and $\pi\left(p^t|g^{1:T}\right)$.

We endow the set of all possible paths on the road network with a probability distribution. The *transition model* η describes a distribution $\eta(p)$ defined over all possible paths p across the road network. This distribution is not a distribution over actually observed paths as much as a model of the *preferences* of the driver when given the choice between several options. In order to make the problem tractable, we introduce some conditional independences between states and paths: a path p^t is independent of all other paths and states given x^t and x^{t+1}, and a state x^t is completely known when either the preceding path p^{t-1} or the subsequent path p^t are known.

Each path must start at the start state and must end at the end state. We formally express the compatibility between a state x and the start and end states of a path p with the compatibility functions $\underline{\delta}$ and $\bar{\delta}$:

$$\underline{\delta}(x,p) = \begin{cases} 1 & \text{if } p \text{ starts at } x \\ 0 & \text{otherwise} \end{cases} \qquad \bar{\delta}(p,x) = \begin{cases} 1 & \text{if } p \text{ ends at } x \\ 0 & \text{otherwise} \end{cases}$$

Given a sequence of observations $g^{1:T} = g^1 \cdots g^T$ and an associated trajectory $\tau = x^1 p^1 \cdots x^T$, we define the *unnormalized score,* or *potential*, of the trajectory as:

$$\phi\left(\tau | g^{1:T}\right) = \left[\prod_{t=1}^{T-1} \omega\left(g^t | x^t\right) \underline{\delta}\left(x^t, p^t\right) \eta\left(p^t\right) \bar{\delta}\left(p^t, x^{t+1}\right)\right] \times \omega\left(g^T | x^T\right)$$

The non-negative function ϕ is called the *potential function.* When properly scaled, the potential ϕ defines a probability distribution over all possible trajectories, given a sequence of observations: $\pi\left(\tau | g^{1:T}\right) = Z^{-1}\phi\left(\tau | g^{1:T}\right)$. The variable $Z = \sum_\tau \phi\left(\tau | g^{1:T}\right)$, called the *partition function*, is the sum of the potentials over all the compatible trajectories.

The potential function ϕ defines an unnormalized distribution over all trajectories. Such a probabilistic framework is a *Conditional Random Field* (CRF) [9], for which efficient inference algorithms exist.

Fig. 4 Illustration of the graphical models used in the path inference filter (CRF, left), and commonly found in the literature (HMM, right), defined over a trajectory $\tau = x^1 p^1 x^2 p^2 x^3$ and a sequence of observations $g^{1:3}$.

The case against the Hidden Markov Model approach. The classical approach to filtering in the context of trajectories is based on *Hidden Markov Models* (HMMs), or their generalization, *Dynamic Bayesian Networks* (DBNs) [10]: a sequence of states and trajectories form a trajectory,

and the coupling of trajectories and states is done using transition models $\hat{\pi}(x|p)$ and $\hat{\pi}(p|x)$. See Figure 4 for a representation.

This results in a chain-structured directed probabilistic graphical model in which the path variables p^t are unobserved. Depending on the specifics of the transition models, $\hat{\pi}(x|p)$ and $\hat{\pi}(p|x)$, probabilistic inference has been done with Kalman filters [11], the forward algorithm or the Viterbi algorithm [3], or particle filters [7].

Fig. 5 Example of a failure case when using a Hidden Markov Model: the solid black path will be favored over all the other paths.

Hidden Markov Model representations, however, suffer from the *selection bias problem*, first noted in the labeling of words sequences [9], which makes them not the best fit for solving path inference problems. Consider the example trajectory $\tau = x^1 p^1 x^2$ observed in our data, represented in Figure 5. For clarity, we consider only two states x_1^1 and x_2^1 associated with the GPS reading g^1 and a single state x_1^2 associated with g^2. The paths $\left(p_j^1\right)_j$ between x^1 and x^2 may either be the lone path p_1^1 from x_1^1 to x_1^2 that allows a vehicle to cross the Golden Gate Park, or one of the many paths between Cabrillo Street and Fulton Street that go from x_2^1 to x^1, including p_3^1 and p_2^1. In the HMM representation, the transition probabilities must sum to 1 when conditioned on a starting point. Since there is a single path from x_2^1 to x^2, the probability of taking this path from the state x_1^1 will be $\hat{\pi}\left(p_1^1|x_1^1\right) = 1$ so the overall probability of this path is $\hat{\pi}\left(p_1^1|g^1\right) = \hat{\pi}\left(x_1^1|g^1\right)$. Consider now the paths from x_2^1 to x_1^2: a lot of these paths will have a similar weight, since they correspond to different turns and across the lattice of streets. For each path p amongst these N paths of similar weight, Bayes' assumption implies $\hat{\pi}\left(p|x_2^1\right) \approx \frac{1}{N}$ so the overall probability of this path is $\hat{\pi}\left(p|g^1\right) \approx \frac{1}{N}\hat{\pi}\left(x_2^1|g^1\right)$. In this case, N can be large enough that $\hat{\pi}\left(p_1^1|g^1\right) \geq \hat{\pi}\left(p|g^1\right)$, and the remote path will be selected as the most likely path.

Due to their structures, all HMM models will be biased towards states that have the least expansions. In the case of a road network, this can be pathological; this phenomenon is observed around highways for example. Our model, which is based on CRFs, does not have this problem since the renormalization happens just once and is over all paths from start to end, rather than renormalizing for every single state transition independently.

Trajectory filtering and smoothing Computing the most likely trajectory $\tau^* = \text{argmax}_\tau\ \pi\left(\tau|g^{1:T}\right)$ is a particular instantiation of a standard dynamic programing algorithm called the *Viterbi algorithm* [5]. The posterior probability $\bar{q}_i^t$ of the vehicle being at the state $x_i^t \in \mathbf{x}^t$ at time t, given all the observations $g^{1:T}$ of the trajectory, is defined up to some scaling factor as:

$$\overline{q}_i^t \propto \pi\left(x_i^t | g^{1:T}\right) = \frac{1}{Z} \sum_{\tau = x^1 \cdots p^{t-1} x_i^t p^t \cdots x^T} \phi\left(\tau | g^{1:T}\right)$$

A natural choice is to scale the distribution $\bar{q}_i^t$ so that the probabilistic weight of all possibilities is equal to 1: $\sum_{1 \leq i \leq I^t} \bar{q}_i^t = 1$. The quantity $\bar{q}_i^t$ has a clear meaning: it is the probability that the vehicle is in state x_i^t, when choosing amongst the set $(x_{i'}^t)_{1 \leq i \leq I^t}$, given all the observations $g^{1:T}$. For each time t and each path index $j \in [1 \cdots J^t]$, we also introduce the discrete distribution over the paths at time t given the observations $g^{1:T}$: $\overline{r}_j^t \propto \pi\left(p_j^t | g^{1:T}\right)$ which are scaled so that $\sum_{1 \leq j \leq J^t} \overline{r}_j^t = 1$.

Finding the most likely trajectory is one of two classical inference problems [10]. The other one is to find for each time what the probability distribution over states is. When doing so conditioned on all past and future observations (in our application the GPS measurements $g^{1:T}$), it is called smoothing, when doing so conditioned on all past measurements only it is called filtering. Filtering can be achieved by a forward pass, and smoothing can be achieved by both the same forward pass as done for filtering and a backward pass and then combining their results [14].

The forward-backward algorithm as applied to our CRF is presented in Algorithm 1. The path inference algorithm has time complexity $O(TUV)$ with U the maximum number of paths at one time step, and V the maximum number of paths originating from a single point (which is usually small). We refer the reader to the supplementary material for a more complete overview of the algorithm. Furthermore, the path inference algorithm can be adapted to pure filtering or lagged-smoothing scenarios to get timely estimates as new data comes in. We also provide a reference implementation of all variations [1].

Observation model We now describe the observation model ω. The observation probability is assumed to only depend on the distance between the point and the GPS coordinates. We take an isoradial Gaussian noise model $\omega(g|x) = \frac{1}{\sqrt{2\pi}\sigma} \left(-\frac{1}{2\sigma^2} \mathrm{d}(g,x)^2\right)$ in which the function d is the distance function between geocoordinates. In a first approximation, the standard deviation σ is assumed to be constant over all the network. This model works surprisingly well despite some known limitations such as urban canyoning. A more robust model, such as the exponential distribution, can also be used. It would be interesting to see how the choice of a more robust model affects the quality of the output.

Driver model The driver model assigns a weight to any path on the road network. We consider a model in the *exponential family*, in which the weight distribution over any path p only depends on a selected number of features $\varphi(p) \in \mathbb{R}^K$ of the path: $\eta(p) \propto \exp\left(\mu^T \varphi(p)\right)$. Feature functions considered in the present article include the length of the path, the mean speed and travel time, the number of stop signs and signals, and the number of turns to

Algorithm 1 Description of the forward-backward recursion algorithm

Given a sequence of observations $g^{1:T}$, a sequence of sets of candidate projections $\mathbf{x}^{1:T}$ and a sequence of sets of candidate paths $\mathbf{p}^{1:T-1}$:
Initialize the forward state distribution:
$\forall i = 1 \cdots I^1 \colon \overrightarrow{q}_i^{\,1} \leftarrow \omega\left(x_i^1 | g^1\right)$
For every time step t from 1 to $T-1$:
Compute the forward probability over the paths:
$\forall j = 1 \cdots J^t : \overrightarrow{r}_j^{\,t} \leftarrow \eta\left(p_j^t\right)\left(\sum_{j:\underline{\delta}\left(x_i^t, p_j^t\right)=1} \overrightarrow{q}_i^{\,t}\right)$
Compute the forward probability over the states:
$\forall i = 1 \cdots I^{t+1} : \overrightarrow{q}_i^{\,t+1} \leftarrow \omega\left(x_i^{t+1} | g^{t+1}\right)\left(\sum_{j:\bar{\delta}\left(p_j^t, x_i^{t+1}\right)=1} \overrightarrow{r}_j^{\,t}\right)$
Initialize the backward state distribution
$\forall i = 1 \cdots I^T \colon \overleftarrow{q}_i^{\,T} \leftarrow 1$
For every time step t from $T-1$ to 1:
Compute the forward probability over the paths:
$\forall j = 1 \cdots J^t : \overleftarrow{r}_j^{\,t} \leftarrow \eta\left(p_j^t\right)\left(\sum_{j:\bar{\delta}\left(p_j^t, x_i^{t+1}\right)=1} \overleftarrow{q}_i^{\,t+1}\right)$
Compute the forward probability over the states:
$\forall i = 1 \cdots I^t \colon \overleftarrow{q}_i^{\,t} \leftarrow \omega\left(x_i^{t+1} | g^{t+1}\right)\left(\sum_{j:\underline{\delta}\left(x_i^t, p_j^t\right)=1} \overleftarrow{r}_j^{\,t}\right)$
For every time step t:
$\forall j = 1 \cdots J^t : \overline{r}_j^t \leftarrow \overrightarrow{r}_j^{\,t} \cdot \overleftarrow{r}_j^{\,t}$
Normalize $\overline{r}^t$
$\forall i = 1 \cdots I^t \colon \overline{q}_i^t \leftarrow \overrightarrow{q}_i^{\,t} \cdot \overleftarrow{q}_i^{\,t}$
Normalize $\overline{q}^t$
Return the set of vectors $\left(\overline{q}^t\right)_t$ and $\left(\overline{r}^t\right)_t$

the right or to the left. The distribution is parametrized by a vector $\mu \in \mathbb{R}^K$, called the *behavioral parameter vector*.

Training The procedure detailed so far requires the calibration of the observation model and the path selection model by setting some values for the weight vector μ and the standard deviation σ. There is a striking similarity between the state variables $x^{1:T}$ and the path variables $p^{1:T}$ — especially between the forward and backward equations introduced in Algorithm 1. Consider $\epsilon = \sigma^{-2}$ and θ the stacked vector of the desired parameters $\theta = \left(\epsilon\, \mu^T\right)^T$. One then realizes that the potential ϕ defines an exponential family distribution over the variables $\mathbf{x}^{1:T}, \mathbf{p}^{1:T-1}$, and that the vector θ is the natural parameter of the distribution. It ensues that maximizing the likelihood of observations with respect to θ is a convex problem [17]. Furthermore, the elements of the objective function (gradient, Hessian) can be efficiently computed using dynamic programming. The complete derivations can be found in the supplementary materials, and our reference implementation [1] provides an implementation of the likelihood maximization procedure.

Usually, only the GPS observations $g^{1:T}$ are available; the choices of x_i^t and p_j^t are hidden. In this case, we estimate a good value of θ using the *Expectation Maximization* (EM) algorithm, in which we take the expectation over the each possible choice x_i^t and p_j^t. This is done by computing the marginal conditional

probabilities $\vec{q}_i^t = \pi\left(x_i^t | g^{1:T}; \theta\right)$ and $\vec{r}_i^t = \pi\left(p_j^t | g^{1:T}; \theta\right)$, using the forward-backward algorithm (Algorithm 1).

4 Results from field operational test

We evaluate the path inference filter with two datasets collected from the same source (taxi cabs): a smaller set at high frequency, called "Dataset 1", and a larger dataset sampled at 1 minute for which we do not know ground truth, called "Dataset 2". For the collection of Dataset 1, ten San Francisco taxicabs were fit with high frequency GPS (1 second sampling rate) during a two-day experiment. Together, they collected about 200,000 measurement points.

The second dataset consists of one day of one-minute samples of 600 taxis from the same fleet, which represents 600,000 GPS points, collected the same month.

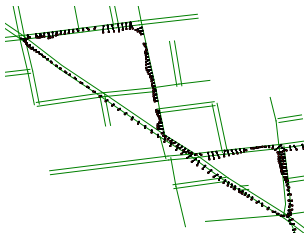

Fig. 6 Example of points collected in "Dataset 1", in the Russian Hill neighborhood in San Francisco. The (red) dots are the GPS observations (collected every second), and the green lines are road links that contain a state projection. The black lines show the most likely projection of the GPS points on the road network.

Experiment design The path inference filter was first run using hand-tuned parameters on all the samples to build a set of ground truth trajectories. The trajectories were then downsampled to different temporal resolutions (2 seconds to two minutes) to test the filter in different configurations. We performed cross-validation to test the impact of the sampling rate, the computing strategy ("online" or pure filtering, "1-lag" and "2-lag" for fixed-lagged smoothing, Viterbi and "offline" or smoothing). We also tested different models: some deterministic models ("Hard closest point" [6], "Closest point", and "Shortest path" [18]) were selected as baselines. In addition, we ran two models from from the exponential family:

- "Simple": A simple model that considers two features that could probably have been tuned by hand with some effort (but were learned in our experiments): the length of the path and the distance of a point projection to its GPS coordinate.
- "Complex": A more complex model with a more diverse set of ten features, which makes manual tuning impractical. In addition to the two features of the simple model, this model in particular includes the number of signals, turns and stop signs on the path, the class of the road and some average speed information.

These two models were also trained in a supervised setting, leading to the "MaxLL-Simple" and "MaxLL-Complex" models, respectively. The simple model was trained using Expectation Maximization on "Dataset 1" and lead

to "EM-Simple". Due to convergence issues, the EM algorithm was used on the complex model using "Dataset 2". This set of parameters is presented under the label "EM-Complex". From a practical perspective, all the deterministic models can be well approximated using the Simple model by setting some large or small values to its parameters.

These models were evaluated under a number of metrics:

- The proportion of path and point misses: for each trajectory, it is the number of times the most likely path or the most likely state was not the true one, divided by the number of time steps in the trajectory.
- The log-likelihood of the true point projection and of the true path projection. This is indicative of how often the true point or the true path is identified by the model.
- The entropy of the path distribution and of the point distribution. This statistical measure indicates the confidence assigned by the filter to its result. A small entropy (close to 0) indicates that one path is strongly favored by the filter against all the other ones, whereas a large entropy indicates that all paths or points are considered equal.

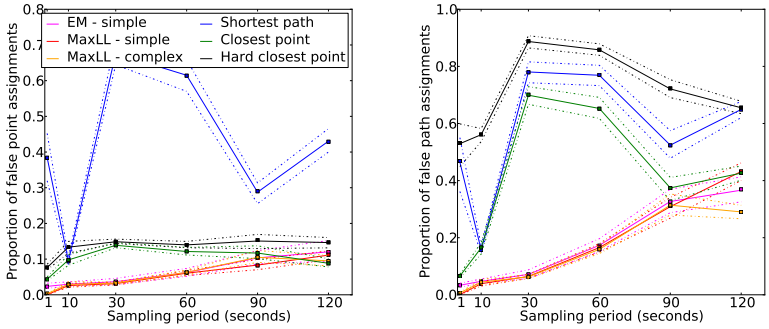

Fig. 7 Proportion of point misses (left) and path misses (right) using trajectory reconstruction (Viterbi algorithm) for different sampling rates, as a percentage of incorrect point reconstructions for each trajectory (positive, smaller is better). The solid line denotes the median, the squares denote the mean and the dashed lines denote the 95% confidence interval. The black curve is the performance of a greedy reconstruction algorithm, and the colored plots are the performances of probabilistic algorithms for different features and weights learned by different methods.

Results Given the number of parameters to adjust, we only present the most salient results here.

The raw accuracy of the filter, in terms of point misses and path misses, is presented in Figure 7. As expected, the error rate is 0 for high frequencies (low sampling period): all the points are correctly identified by all the algorithms. In the low frequencies (high sampling periods), the error is still low (around 10%) for the trained models, and also for the greedy model ("Hard closest point"). For sampling rates between 10 seconds and 90 seconds, trained models ("Simple" and "Complex") show a much higher performance compared to

untrained models ("Hard closest point", "Closest point" and "Shortest path"). In higher sampling regions, there are significantly more paths to consider and the error increases substantially. Nevertheless, the probabilistic models still perform very well: even at 2 minute intervals, they are able to recover about 75% of the true paths.

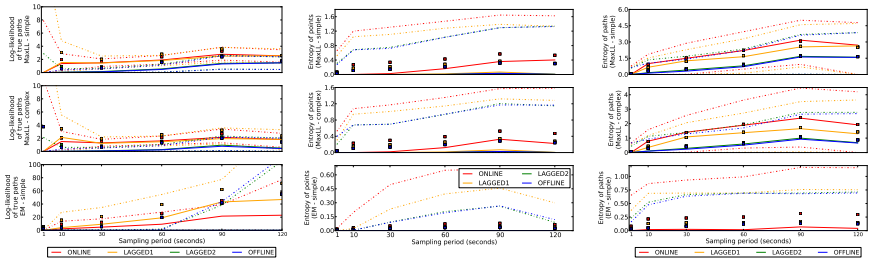

Fig. 8 On the left, negative of the log likelihood of true paths for different strategies and different sampling rates (positive, lower is better). On the center, entropy of point distributions and on the right, entropy of path distributions. The solid line denotes the median, the squares denote the mean and the dashed lines denote the 90% confidence interval.

We now turn our attention to the resilience of the models, i.e. how they perform when they make mistakes. We use two statistical measures: the (log) likelihood of the true paths, and the entropy of the distribution of points or paths (Figure 8). Note that in a perfect reconstruction with no ambiguity, the log likelihood would be zero. Interestingly, the log likelihoods appear very stable as the sampling interval grows: our algorithm will continue to assign high probabilities to the true projections even when many more paths can be used to travel from one point to the other. The performance of the simple and the complex models improves greatly when some backward filtering steps are used, and stays relatively even across different time intervals.

The paths inferred by the filter are also never dramatically different: at two minute time intervals (for which the paths are 1.7km on average), the returned path spans more than 80% of the true path on average. Using the complex model decreases this relative error by more than 15% . We refer the reader to the supplementary materials for a longer discussion on the performance of the filter.

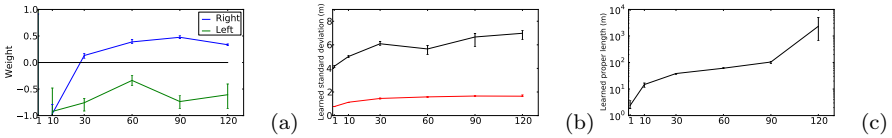

Fig. 9 Examples of learned feature weights: le..ght turn preferences (a), standard deviation (b), and characteristic length (c). The error bars indicate the complete span of values computed for each time.

In the case of the complex model, the weights can provide some insight into the features involved in the decision-making process of the driver. In particular, for extended sampling rates (t=120s), some interesting patterns appear. For example, drivers show a clear preference to turn on the right as opposed to the left, as seen in Figure 9. This may be attributed in part to the difficulty in crossing an intersection in the United States.

Unsupervised learning results The filter was also trained for the simple and complex models using Dataset 2. This dataset does not include true observations but is two orders of magnitude larger than Dataset 1 for the matching sampling period (1 minute). The unsupervised training finds some weight values similar to those found with supervised learning. The magnitude of these weights is larger than in the supervised settings, however. Indeed, during the E step, the algorithm is free to assign any sensible value to the choice of the path. This may lead to a self-reinforcing behavior and the exploration of a bad local minimum.

As Table 10 shows, however, a large training dataset puts unsupervised methods on par with supervised methods as far as performance metrics are concerned.

	False points	False paths
EM-Complex (large)	**5.8±0.4%**	16.0±1.0%
EM-Simple (large)	6.3±0.4%	17.1±0.9%
EM-Simple	6.2±0.6%	17.3±1.3%
MaxLL-Complex	6.2±0.3%	**15.8±0.7**
MaxLL-Simple	6.1±0.3%	16.5±0.7

Fig. 10 Proportion of true points and true paths incorrectly identified, for different models evaluated with 1-minute sampling (lower is better).

Key findings Our algorithm can reconstruct a sensible approximation of the trajectory followed by the vehicles analyzed, even in complex urban environments. In particular:

- An intuitive deterministic heuristic ("Hard closest point") dramatically fails for paths at low frequencies, less so for points. It should not be considered for sampling intervals larger than 30 seconds.
- A simple probabilistic heuristics ("closest point") gives good results for either very low frequencies (2 minutes) or very high frequencies (a few seconds) with more 75% of paths and 94% points correctly identified. However, the incorrect values are not as close to the true trajectory as they are with more accurate models ("Simple" and "Complex").
- For the medium range (10 seconds to 90 seconds), trained models (either supervised or unsupervised) have a greatly improved accuracy compared to untrained models, with 80% to 95% of the paths correctly identified by the former.
- For the paths that are incorrectly identified, trained models ("Simple" or "Complex") provide better results compared to untrained models (the output paths are closer to the true paths, and the uncertainty about which paths may have been taken is much reduced). Furthermore, using

a complex model ("Complex") improves these results even more by a factor of 13-20% on all metrics.
- For filtering strategies: online filtering gives the worst results and its performance is very similar to 1-lagged smoothing. The slower strategies (2-lagged smoothing and offline) outperform the other two by far. Two-lagged smoothing is nearly as good as offline smoothing, except in very high frequencies (less than 2 second sampling) for which smoothing clearly provides better results. We thus recommend two-lagged smoothing for most applications.
- Using a trained algorithm in a purely unsupervised fashion provides an accuracy as good as when training in a supervised setting - assuming enough data is available. The models produced by EM are equally good in terms of raw performance (path and point misses), but they may be overconfident.

5 Conclusions

We have presented a novel class of algorithms to track moving vehicles on a road network: the *path inference filter*. This algorithm first projects the raw points onto candidate projections on the road network and then builds candidate trajectories to link these candidate points. An observation model and a driver model are then combined in a Conditional Random Field to infer the most probable trajectories.

The algorithm exhibits robustness to noise as well as to the specificities of driving in urban road networks. It is competitive over a wide range of sampling rates (1 second to 2 minutes) and greatly outperforms intuitive deterministic algorithms. Furthermore, given a set of ground truth data, the filter can be automatically tuned using a fast supervised learning procedure. Alternatively, using enough GPS data with no ground truth, it can be trained using unsupervised learning. Experimental results show that the unsupervised learning procedure compares favorably against learning from ground truth data.

This algorithm supports a range of trade-offs between accuracy, timeliness and computing needs. It extends the current state of the art [20, 19] in its most accurate setting. The results are supported by the theoretical foundations of Conditional Random Fields. Because no standardized benchmark exists, the authors have released an open-source implementation of the filter to foster comparison with other methodologies using other datasets [1].

The authors have written an industrial-strength version in the Scala programming language, deployed in the *Mobile Millennium* system. This multi-core version maps GPS points at a rate of several thousands of GPS points per second for the San Francisco Bay area and other large urban areas.

References

1. Supporting code for the path inference filter. `https://github.com/tjhunter/Path-Inference-Filter/`.
2. A. Bayen, J. Butler, A. Patire, and et al. Mobile Millennium final report. Technical report, University of California, Berkeley, CCIT Research Report UCB-ITS-CWP-2011-6, 2011.
3. M. Bierlaire and G. Flötteröd. Probabilistic multi-modal map matching with rich smartphone data. In *STRC 2011*, 2011.
4. The Cabspotting program. `http://cabspotting.org/`.
5. G.D. Forney Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.
6. J.S. Greenfeld. Matching GPS observations to locations on a digital map. In *81th Annual Meeting of the Transportation Research Board*, 2002.
7. F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.J. Nordlund. Particle filters for positioning, navigation, and tracking. *Signal Processing, IEEE Transactions on*, 50(2):425–437, 2002.
8. P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
9. J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
10. K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California at Berkeley, 1994.
11. W.Y. Ochieng, M. Quddus, and R.B. Noland. Map-matching in complex urban road networks. *Revista Brasileira de Cartografia*, 2(55), 2009.
12. M.A. Quddus, W.Y. Ochieng, L. Zhao, and R.B. Noland. A general map matching algorithm for transport telematics applications. *GPS solutions*, 7(3):157–167, 2003.
13. D.L. Schrank, T.J. Lomax, and Texas Transportation Institute. *2009 Urban mobility report*. The Texas A&M University, 2009.
14. K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden Markov model structure for information extraction. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.
15. Arvind Thiagarajan, Lenin S. Ravindranath, Katrina LaCurts, Sivan Toledo, Jakob Eriksson, Samuel Madden, and Hari Balakrishnan. Vtrack: Accurate, energy-aware traffic delay estimation using mobile phones. In *7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Berkeley, CA, November 2009.
16. S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.
17. M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.
18. C. E. White, D. Bernstein, and A. L. Kornhauser. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies*, 8(1-6):91–108, 2000.
19. J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.Z. Sun. An interactive-voting based map matching algorithm. In *Mobile Data Management (MDM), 2010 Eleventh International Conference on*, pages 43–52. Ieee, 2010.
20. Y. Zheng and M.A. Quddus. Weight-based shortest-path aided map-matching algorithm for low-frequency positioning data. In *Transportation Research Board 90th Annual Meeting*, 2011.