Physically Consistent Map Matching

Physically Consistent Map Matching

Bram Custers
Wouter Meulemans
[b.a.custers|w.meulemans]@tue.nl
TU Eindhoven
The Netherlands

Marcel Roeloffzen
Bettina Speckmann
[m.j.m.roeloffzen|b.speckmann]@tue.nl
TU Eindhoven
The Netherlands

Kevin Verbeek
k.a.b.verbeek@tue.nl
TU Eindhoven
The Netherlands

ABSTRACT

An important data source for traffic analysis is GPS trajectory data. However, due to measurement inaccuracies, such data does not necessarily align well with data describing the road network. Hence, GPS data typically needs to be aligned with the road network before further analysis can take place; this process is called map matching. The challenges in map matching are exacerbated when trajectories are sparse (have a low measurement frequency); we then need to fill the gaps between the measurements with a realistic estimate of the actual route between two points. As vehicle movement is subject to physical constraints (such as acceleration and speed bounds), an estimated route should be physically consistent. We present a method that creates physically consistent estimated routes to perform map matching for sparse trajectories.

CCS CONCEPTS

• **Information systems → Geographic information systems**.

KEYWORDS

Map matching, GPS trajectories, physical consistency

ACM Reference Format:
Bram Custers, Wouter Meulemans, Marcel Roeloffzen, Bettina Speckmann, and Kevin Verbeek. 2022. Physically Consistent Map Matching. In *The 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '22), November 1–4, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3557915.3560991

1 INTRODUCTION

Understanding mobility and traffic is vital to the management of transportation infrastructure. To understand and analyze these phenomena, we need good data that captures them. GPS trajectories are an important source of information for such analyses. However, to work well with GPS trajectories, some processing is needed to overcome inaccuracies, misalignment, and missing data. In particular, for human mobility in the form of traffic, the road network defines and constrains vehicle movement, allowing us to infer local information by considering where the vehicle was driving in the network. Hence we need to find the driven route in the network for a GPS trajectory; this process is known as *map matching*.

SIGSPATIAL '22, November 1–4, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9529-8/22/11.
https://doi.org/10.1145/3557915.3560991

There are multiple challenges in map matching. Firstly, the individual GPS measurements may be noisy: GPS receivers have varying accuracy, and the noise of the signal is further exacerbated by urban infrastructure such as tall buildings, overpasses, and tunnels. Secondly, trajectories may be incomplete: individual samples or entire time ranges can be missing from the collected trace. And finally, trajectories may have low temporal sampling frequency, for example, to conserve battery power or to improve privacy.

To overcome these issues, algorithms have been developed in previous work that try to fill the gaps in sparse GPS trajectories. These algorithms generally try to infer from the context where on the road network the measured locations should be and aim to estimate reasonable network paths between these locations. The contextual data is often attached to the road network: estimated travel times or speed limits may be recorded on road segments. Alternatively, contextual data on the trajectory measurements themselves, such as recorded velocity and heading, may be used to guide the algorithms.

Contributions. In this paper, we present a new map-matching algorithm that leverages physical constraints on vehicles. In particular, we assume that a vehicle has bounded acceleration (and deceleration). In addition, we assume that a vehicle is subject to legal constraints: it must adhere to speed limits on the road network. Our algorithm finds a solution space of routes that are feasible under the given speed limits and acceleration bounds. Our routes have hence a guaranteed high quality. We select the final route based on geometric length and road-type changes.

Related work. Map matching is a well-studied topic. We refer the reader to recent surveys for a comprehensive overview [2, 7]; here we highlight the results that our most relevant in our context.

The most commonly used map-matching algorithms are based on Hidden Markov Models (HMMs). In these approaches a group of candidate points that lie on the road network are selected and candidate paths are created to connect them. The algorithm itself is incremental and maintains the "best" paths to each of the candidate points of a measured location. At each stage these paths are extended to the next set of candidates points and again the best path to each candidate is stored. Which paths are best is decided through a set of probabilities based on various measures. Differences between approaches occur based on how probabilities are assigned to different candidates. The simplest and most common factor is the distance of a candidate to its measured location. A second common factor is to consider the length of the path to each candidate. One could aim to have this distance be close to the distance between the measured locations [12] or to have it match with the time between measurements based on speed limits [5]. Other options include considering the number of turns [13].

Outside of the incremental HMM method one can also consider speed bounds in a more global manner. In the ST-matching method

as proposed by Lou *et al.* [10] a graph is constructed that models paths between candidates. This graph is then weighted based on discrepancies between the time it would take to traverse a path and the time differences between the measurements. One can then find a globally good path through this graph. This approach can also be extended to include measured directions of movement [3, 6].

We ensure that the paths found are physically consistent to provide a quality guarantee on our output. That is, the paths produced by our algorithm are physically realizable within the given acceleration bound and speed limits. Using physics to model driver behavior is a common technique in trajectory processing and GIS analysis in general [11]. In its most simple form, one leverages the velocity of a vehicle to determine whether a certain distance can be traveled in a given time. In a GIS context, such a construction is commonly referred to as the space-time prism. Depending on the type of problem at hand, this can be sufficient. In other problems where the kinetic properties of a vehicle play a larger role, a more complex model can be useful, where acceleration is taken into account [4, 8, 9]. For graphs Ardizonni *et al.* [1] describe how to find a fastest path under speed and acceleration bounds. This is similar to what we are interested in, however, in our case we are not interested in the fastest path, but in a path that is physically consistent with the measured locations and times. Additionally, the algorithm proposed by Ardizonni *et al.* does not run in polynomial time for general road networks.

2 PRELIMINARIES

A (GPS) measurement $p = (x, y, t)$ is a tuple (x, y, t), containing the measured location $(x, y) \in \mathbb{R}^2$ and the time t at which the measurement was taken. We then define a *trajectory* as a sequence of measurements $T = \langle p_1, \ldots, p_n \rangle$. We consider a *road network* to be a graph $\mathcal{G} = (V, E)$, embedded in $\mathbb{R}^2$ using straight lines for edges in E. We assume that each edge on the road network has an attribute indicating the local speed bound and a road type.

We consider two measurements *consistent* [4] if the vehicle could travel in the time span between the measurements from the first location to the second, considering properties such as maximum speed, acceleration, or turning rate. The actual movement satisfying such physical properties is then a *witness* to the consistency.

A model that considers only speed limits is *concatenable* [4], meaning that we can decide consistency between measurements independently, and combine them without violating consistency. Here, we consider a more refined model that includes acceleration (and deceleration) bounds. This model is *conditionally concatenable* [4], meaning that we must know boundary conditions: how the witness arrives at a measurement (in our case, at which speed) influences if or how we can reach the next measurement.

3 CONSISTENT MAP MATCHING

Our overall algorithm for map matching based on consistency is designed similarly to a Hidden Markov Model. Our method maps a trajectory T to a given road network $\mathcal{G}$ as follows:

(1) We generate c candidate locations on $\mathcal{G}$ for each of the measurements in T. These candidate locations are the nearest point on the c nearest road segments in $\mathcal{G}$.
(2) We generate a consistent route for every candidate location of each measurement to each candidate location of the next measurement, if such consistent routes indeed exists. Since we have bounded acceleration, it matters at which speed the vehicle is driving at the former candidate location, to decide how it can continue to the next. As such, we must keep track of intervals of speeds at which the vehicle could have reached the candidate location, to ensure consistency of the entire route.
(3) We trace back through our generated consistent routes from candidate location to candidate location, to reconstruct the overall route.

3.1 Computing a consistent route

A given route. How can we check whether a given route, travelling through candidate locations, is indeed consistent? That is, does a witness exist that demonstrates that these candidate locations can be visited at exactly the given times, while adhering to the local speed bounds and global acceleration bounds? Overall, we use an approach akin to that presented by Custers *et al.* [4]: knowing the interval(s) of speeds at which the vehicle could travel at one candidate location, we *propagate* under consistency to the next candidate location to find new interval(s). However, where Custers *et al.* [4] consider a global speed bound, our speed bound varies along the route. Hence we need a more involved propagation algorithm.

To propagate, our input is four-fold: (1) a route through the network – a sequence of edge lengths $\langle \ell_1, \ldots, \ell_m \rangle$ and speed limits $\langle v_1, \ldots, v_m \rangle$; (2) an interval of possible initial speeds $[v_\downarrow, v_\uparrow]$; (3) a total time Δt between the candidate locations; (4) an interval of realistic acceleration $[a_\downarrow, a_\uparrow]$. We wish to determine the interval $[v_{\downarrow,\mathrm{f}}, v_{\uparrow,\mathrm{f}}]$ of final speeds that the vehicle may travel at, when it reaches the end of the route, while taking exactly Δt time and adhering to the local speed limits and global acceleration bounds. Note that we use speed instead of velocity, assuming that the vehicle always travels forwards along the route. Moving backwards is in principle possible, but unlikely to be done in such amounts as to affect consistency between candidate locations – driving back and forth can be mimicked by standing still after all.

SIS diagrams. We introduce a *space-inverse-speed* diagram (SIS diagram) as a tool to reason about witnesses. In a SIS diagram, we plot the inverse speed of the vehicle as a function of space. By our assumption of only moving forward, this is indeed a function. In the limit of moving arbitrarily slowly, the vehicle can also stand still. We use the inverse speed, such that the integral of this function is exactly the travel time. To match intuition of the function in the diagram visually going up to increasing speed, we draw the vertical axis downwards, with zero (of inverse speed, so infinite speed) at the top. The travel time is then the area above the curve.

The local speed bounds translate to horizontal segments in the SIS diagram. At the transition, there is a jump in the speed limit, but due to the acceleration bounds, the vehicle cannot discretely change its speed. To strictly adhere to limits, acceleration or deceleration to reach the new speed bound must happen in the segment that has a higher speed bound. Hence, considering also acceleration bounds as well as the maximum initial speed $v_\uparrow$, we obtain a continuous function $U(x)$ that indicates the maximum achievable speed at

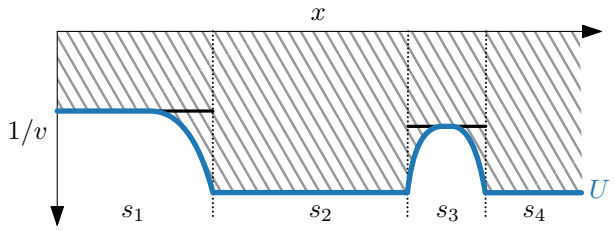

Figure 1: Example SIS diagram. Shown are four segments s_1, s_2, s_3, s_4, with the blue line indicating the maximum achievable speed U. The hatched area indicates the travel time to traverse these segments at maximum speed and acceleration. Witnesses must stay on or below U.

distance x along the route, measured from the starting point. Any witness must stay below or on this function in the diagram. An example is given in Fig. 1.

The function U is defined by pieces of movement following extremes; also in our later proofs, we use extreme movement. We distinguish the following types.

- The vehicle travels at maximum local speed v as defined by $\mathcal{G}$. This corresponds to a horizontal line in a SIS diagram.
- The vehicle accelerates maximally, at a rate of $a_\uparrow$. This corresponds to a decreasing curve (going up in the drawing) in a SIS diagram.
- The vehicle decelerates maximally, accelerating at a rate $a_\downarrow$. This corresponds to an increasing curve (going down in the drawing) in a SIS diagram.

Computing propagation. What follows is a sequence of lemmata, to establish eventually that we can propagate an interval of speeds from one candidate location to the next, for a given route, in time linear in the complexity of the route.

Lemma 3.1. *For a route of m segments and an initial speed interval $[v_\downarrow, v_\uparrow]$, we can compute U in $O(m)$ time.*

Proof sketch. We build U incrementally, storing the pieces of the curve in a stack, keeping track of the current speed v_c of U, which is initially $v_\uparrow$, the maximum initial speed of the route. For each segment in the route, we add its maximum speed to the upper envelope, with an accelerating curve if its speed bound exceeds the current ending speed of U, or a decelerating curve if the speed bound decreases. Note that the latter case, the curve is added before the road segment under consideration, and may therefore require changes to U. Effectively, this is removing the current head of U until it fits. Since each segment adds at most two curves to U, each segment takes amortized $O(1)$ time, and thus the entire construct runs in linear time. □

Corollary 3.2. *We can test whether a witness taking at most Δt time exists for a given route in linear time.*

Lemma 3.3. *If a route admits both a witness with total time less than Δt and a witness with total time more than Δt, then a witness exists that takes exactly Δt time.*

Proof sketch. We can linearly interpolate between the two assumed witness to construct one that has exactly a total time of Δt; since speed and acceleration in this witness are a convex combination of the assumed witnesses, it must match the speed and acceleration bounds as well. □

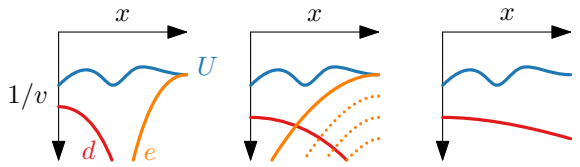

Figure 2: Computing $v_{\uparrow,\mathrm{f}}$: if d and e do not intersect and reach speed zero, U determines the maximum speed at the end of the route (left); otherwise we find an ending speed by shifting e such that we achieve a total time Δt (middle); though such may not exist if d does not reach zero (right).

Lemma 3.4. *Given a route, total time Δt and an interval $[v_\downarrow, v_\uparrow]$ of initial speeds, we can compute the interval $[v_{\downarrow,\mathrm{f}}, v_{\uparrow,\mathrm{f}}]$ of final speeds for all witnesses in linear time.*

Proof sketch. We construct upper envelope U via Lemma 3.1, and derive the fastest total travel time. If this time exceeds Δt or U does not exist, then no consistent witness exists. Otherwise, we find $v_{\uparrow,\mathrm{f}}$, by decelerating for some distance to then maximally accelerate towards the speed bound. By constructing a witness that has the highest possible final speed and with total travel time at least Δt, we can use Lemma 3.3 with such a witness and U to establish that this is indeed $v_{\uparrow,\mathrm{f}}$. This witness must be one of a few cases (see Fig. 2 for examples) in constant time.

In the SIS diagram, if the decelerating curve d starting at $v_\downarrow$ and the accelerating curve e ending at the final speed of U intersect, we conceptually shift e downwards (that is, we reduce the final speed) until the witness that d and e together describe has total time at least Δt. The required shift can be computed directly using a closed-form expression in constant time.

If they do not intersect, we need to consider whether both curves reach speed zero (in which case we can reach the maximal speed of U), or whether e reaches $x = 0$ above $v_\downarrow$ (in which case we follow a similar strategy as before, by shifting e downwards).

Computing $v_{\downarrow,\mathrm{f}}$ follows a similar rationale: first maximally accelerate, then follow U, finally maximally decelerate. □

Theorem 3.5. *Given a trajectory T whose measurements lie on a route P through a road network, we can compute in linear time whether P is consistent with T given global acceleration bounds and local speed bounds.*

Proof. We begin by partitioning P into the subroutes that reflect the pieces from one measurement to the next in T; this can be done trivially in linear time. Now, we initialize an interval of speeds from zero to the local maximum speed at the first measurement in T. We use Lemma 3.4 to repeatedly propagate this interval to an interval of speeds at the next measurement, for the given subroute. This takes linear time in the complexity of the subroute, and thus linear time overall. We conclude that P is consistent with T if this interval is never empty. □

Finding a route. Our goal is to find a route, rather than just validating a given one. However, generalizing our method above to a find a route in a network is problematic. Our approach relies heavily on velocity being a function of space (distance along the given route), avoiding a need for explicit consideration of when we arrive at a specific location along the route – this is captured implicitly by the area above the curve after all. For a given route, this simply means that we assume the vehicle did not drive backwards.

In a network we need an explicit consideration of intermediate vertices in order to aggregate information about different routes through $\mathcal{G}$, if we are to avoid enumerating all routes. However, this implies that we must consider not only the speed at a vertex, but also how long it took the vehicle to get to this vertex. That is, the information we need goes from 1-dimensional (interval of speeds, at a fixed time) to 2-dimensional (areas in a "speed-time" diagram). Propagation of such information is cumbersome at best, even if we assume some form of order between the vertices (e.g., the spanned subnetwork is a DAG). But generally, this subnetwork will not have a clear order, and we may visit two vertices in either order in a route; even if we assume a simple route and thereby forbid revisiting a vertex in a route, computationally, we may need to consider vertices multiple times, since the order of traversal matters.

Hence, we opt to use another assumption: the vehicle moves along some route that is "reasonable": a fast route, though not necessarily the fastest. With this assumption in mind, we thus generate k fastest alternative routes between two candidate locations, using purely the speed bounds. Specifically, we use the k-shortest paths algorithm [14] with edge weights corresponding to traversal times of the edges. For each alternative route, we consider consistency and propagate intervals, using the method described above. Though we could also consider computing the fastest path with acceleration bounds [1], the parameter used in the presented algorithm is higher than a trivial constant, leading to impractical running times.

Fragmentation. When combining feasible intervals from multiple paths between two measurements, the resulting intervals at the second measurement may be disjoint. This phenomenon is referred to as fragmentation of the speed interval [4]. Just as [4] we can mimic free movement through a fully connected road network, and hence fragmentation is also in the worst case at least linear, and at most exponential – though the exponential term now relies on the number of paths through the network, rather than the number of measurements (since we do not skip any measurements). In our map matcher, we combine intervals into a single interval if they overlap for further propagation, though we do store the intervals separately to support reconstructing the route.

4 CONCLUSION AND DISCUSSION

We developed a method that allows us to map-match sparse trajectories, in such a way that the resulting route adheres to constraints on local speed and global acceleration. In the future we plan to experimentally validate our algorithmic results. Below we briefly review some possible avenues for additional future work.

We use only acceleration in our model, disregarding that taking sharp turns requires a lower speed. Though we could model a turning rate explicitly, it seems reasonable to assume that turns in the road network can be taken, if the vehicle travels at a sufficiently low speed. Such an approach can easily be modeled in our method, injecting zero-length edges at turns with a speed limit depending on the angle of the turn.

Our approach relies on driving at maximum speed, maximally accelerating and maximally decelerating, to decide consistency. Typically, this results in a variety of possible ways to have driven the reconstructed route. If we are to estimate the actual location (and thereby speed and acceleration) of the vehicle at all times, we are looking for the "most reasonable" witness and may consider minimizing e.g. the variation in acceleration, or the deviation from the local speed bounds.

Furthermore, it may be interesting to consider the "inverse" problem of map matching with physical consistency, in order to augment a road network or assess its quality: if we are given a route and acceleration bounds, what would reasonable speed limits on the network be? Or similarly, how much do the given speed limits need to change to accommodate a consistent route?

ACKNOWLEDGMENTS

B. Custers is supported by HERE Technologies & NWO (628.011.005).

REFERENCES

[1] Stefano Ardizzoni, Luca Consolini, Mattia Laurini, and Marco Locatelli. 2022. Solution algorithms for the Bounded Acceleration Shortest Path Problem. *IEEE Trans. on Automatic Control* 0, 0 (2022), 1–8. https://doi.org/10.1109/TAC.2022.3172169

[2] Pingfu Chao, Yehong Xu, Wen Hua, and Xiaofang Zhou. 2020. A survey on map-matching algorithms. In *Proc. Australasian Database Conference (LNISA 12008)*. Springer, New York City, 121–133. https://doi.org/10.1007/978-3-030-39469-1_10

[3] Mingliang Che, Yingli Wang, Chi Zhang, and Xinliang Cao. 2018. An enhanced Hidden Markov map matching model for floating car data. *Sensors* 18, 6, Article 1758 (2018), 19 pages. https://doi.org/10.3390/s18061758

[4] Bram Custers, Mees Van De Kerkhof, Wouter Meulemans, Bettina Speckmann, and Frank Staals. 2021. Maximum Physically Consistent Trajectories. *ACM TSAS* 7, 4, Article 17 (2021), 33 pages. https://doi.org/10.1145/3452378

[5] Chong Yang Goh, Justin Dauwels, Nikola Mitrovic, Muhammad Tayyab Asif, Ali Oran, and Patrick Jaillet. 2012. Online map-matching based on hidden Markov model for real-time traffic sensing applications. In *Proc. 15th Int. IEEE Conf. on Intelligent Transportation Systems*. IEEE, New York City, 776–781. https://doi.org/10.1109/ITSC.2012.6338627

[6] Yu-Ling Hsueh and Ho-Chian Chen. 2018. Map matching for low-sampling-rate GPS trajectories by exploring real-time moving directions. *Information Sciences* 433 (2018), 55–69. https://doi.org/10.1016/j.ins.2017.12.031

[7] Zhenfeng Huang, Shaojie Qiao, Nan Han, Chang-an Yuan, Xuejiang Song, and Yueqiang Xiao. 2021. Survey on vehicle map matching techniques. *CAAI Trans. on Intelligence Technology* 6, 1 (2021), 55–71. https://doi.org/10.1049/cit2.12030

[8] Bart Kuijpers, Harvey J Miller, and Walied Othman. 2017. Kinetic prisms: incorporating acceleration limits into space–time prisms. *IJGIS* 31, 11 (2017), 2164–2194. https://doi.org/10.1080/13658816.2017.1356462

[9] Jed A Long. 2016. Kinematic interpolation of movement data. *IJGIS* 30, 5 (2016), 854–868. https://doi.org/10.1080/13658816.2015.1081909

[10] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *Proc. 17th ACM SIGSPATIAL GIS*. ACM, New York City, 352–361. https://doi.org/10.1145/1653771.1653820

[11] Harvey J Miller. 2017. Time geography and space-time prism. *Int. encyclopedia of geography: People, the earth, environment and technology* 1 (2017), 1–19. https://doi.org/10.1002/9781118786352.wbieg0431

[12] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *Proc. 17th ACM SIGSPATIAL GIS*. ACM, New York City, 336–343. https://doi.org/10.1145/1653771.1653818

[13] Takayuki Osogami and Rudy Raymond. 2013. Map Matching with Inverse Reinforcement Learning. In *Proc. 23rd Int. Joint Conf. on Artificial Intelligence*. ACM, New York City, 2547–2553.

[14] Jin Y Yen. 1971. Finding the k shortest loopless paths in a network. *Management Science* 17, 11 (1971), 712–716. https://doi.org/10.1287/mnsc.17.11.712