Maximum Physically Consistent Trajectories

Maximum Physically Consistent Trajectories

BRAM CUSTERS, TU Eindhoven, the Netherlands
MEES VAN DE KERKHOF, Utrecht University, the Netherlands
WOUTER MEULEMANS and BETTINA SPECKMANN, TU Eindhoven, the Netherlands
FRANK STAALS, Utrecht University, the Netherlands

Trajectories are usually collected with physical sensors, which are prone to errors and cause outliers in the data. We aim to identify such outliers via the physical properties of the tracked entity, that is, we consider its physical possibility to visit combinations of measurements. We describe optimal algorithms to compute maximum subsequences of measurements that are consistent with (simplified) physics models. Our results are output-sensitive with respect to the number k of outliers in a trajectory of n measurements. Specifically, we describe an $O(n \log n \log^2 k)$-time algorithm for 2D trajectories using a model with unbounded acceleration but bounded velocity, and an $O(nk)$-time algorithm for any model where consistency is "concatenable": a consistent subsequence that ends where another begins together form a consistent sequence. We also consider acceleration-bounded models that are not concatenable. We show how to compute the maximum subsequence for such models in $O(nk^2 \log k)$ time, under appropriate realism conditions. Finally, we experimentally explore the performance of our algorithms on several large real-world sets of trajectories. Our experiments show that we are generally able to retain larger fractions of noisy trajectories than previous work and simpler greedy approaches. We also observe that the speed-bounded model may in practice approximate the acceleration-bounded model quite well, though we observed some variation between datasets.

CCS Concepts: • **Theory of computation → Computational geometry**;

Additional Key Words and Phrases: Outlier detection, algorithms, physics models, experiments

ACM Reference format:
Bram Custers, Mees van de Kerkhof, Wouter Meulemans, Bettina Speckmann, and Frank Staals. 2021. Maximum Physically Consistent Trajectories. *ACM Trans. Spatial Algorithms Syst.* 7, 4, Article 17 (June 2021), 33 pages.
https://doi.org/10.1145/3452378

1 INTRODUCTION

Trajectories—sequences of time-stamped locations representing the motion of an entity—are among the most frequently collected types of spatio-temporal data. Consequently, there are myriad analysis techniques that use trajectories as their input. However, many ways to collect trajectories

This research was supported by the Dutch Research Council (NWO) under Project No. 639.023.208. B. Custers and M. van de Kerkhof are supported by HERE Technologies and NWO under Project No. 628.011.005; B. Speckmann is partially supported by NWO under Project No. 639.023.208.

Authors' addresses: B. Custers, W. Meulemans, and B. Speckmann, Department of Mathematics and Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands; emails: {b.a.custers, w.meulemans, b.speckmann}@tue.nl; M. van de Kerkhof, Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands; email: m.a.vandekerkhof@uu.nl; F. Staals, Utrecht University, the Netherlands; email: f.staals@uu.nl.

2374-0353/2021/06-ART17
https://doi.org/10.1145/3452378

involve physical sensors, which are prone to errors. For example, GPS readings notoriously stray far from their real location in urban canyons, resulting in trajectories with multiple significant outliers. These outliers pose problems for many analysis techniques such as clustering or grouping, and they skew the results of statistical methods. Hence, it is common practice to try to eliminate outliers in a preprocessing step.

There are a variety of methods to remove outliers. Some techniques, such as smoothing or averaging the data, have a possibly negative impact on the complete trajectory. Others, such as map matching, are applicable only to trajectories that can be expected to coincide with a road network. In this article, we focus on *outlier detection*, that is, we describe algorithms that identify outliers, which are subsequently removed from the trajectory.

Specifically, we aim to identify outliers via the physical properties of the moving (real-world) entity. We consider two measurements within a trajectory to be *consistent* for a particular physics model, if the corresponding entity could have traveled between the two measured locations in the time between the two measurements. In this article, we present optimal algorithms to compute maximal consistent subtrajectories according to different (simplified) physics models. Before describing our results in more detail, we first introduce the necessary notation and formally state the problem.

Notation. A trajectory T is a mapping from time to space that represents the motion of an entity. However, recording devices typically record the position, and any other relevant information, of the entity at discrete moments in time. Hence, a trajectory T is usually stored as a sequence of time-stamped *measurements* $\langle p_1, \ldots, p_n \rangle$. A measurement p_i represents the position of the entity at time t_i and may contain additional information such as its velocity v_i and its acceleration a_i at time t_i. The measurements are ordered by timestamp, so $t_i < t_j$ if and only if $i < j$.

Let v^- be the minimum speed, or velocity, that the entity can achieve, and let v^+ be the maximum speed that the entity can achieve. Similarly, let a^- and a^+ be the minimum and maximum possible acceleration. These speed and acceleration bounds represent physical bounds, and thus the entity cannot exceed them at any time, even in between consecutive time stamps t_i and t_{i+1}. The actual continuous motion of an entity is assumed to be a continuous path $\pi : [t, t'] \to \mathbb{R}^d$ over time interval $[t, t']$ through d-dimensional space (typically, $d = 2$). We say that a path π *adheres to* the physics model if it never exceeds the bounds. For example, the speed is always in $[\mathrm{v}^-, \mathrm{v}^+]$ and the acceleration is always in $[a^-, a^+]$. A sequence of measurements $T = \langle p_1, \ldots, p_n \rangle$ is *consistent* with the physics model, denoted $C(T)$, if and only if there exists at least one *witness*: a path $\pi : [t_1, t_n] \to \mathbb{R}^d$ such that (i) for all $i \in \{1, \ldots, n\}$, $\pi(t_i)$ coincides with location p_i, and (ii) π adheres to the physics model. We sometimes write $C(p_i, p_j)$ instead of $C(\langle p_i, p_j \rangle)$.

We use *subtrajectory* or *subsequence* of a trajectory T to refer to a subset of the measurements in the same order as in T; note that these measurements do not need to be consecutive in T.

Formal problem statement. Given a trajectory T and a physics model, compute a maximum-size subsequence S of T such that S is consistent with the given model. When S has size ℓ, the trajectory T contains $k = n - \ell$ erroneous measurements or *outliers*.

Concatenability. Regardless of the physics model, if a sequence T is consistent, then so is any subsequence S of T. But we cannot necessarily construct a consistent subsequence from smaller ones: the concatenation $\langle p_1, \ldots, p_n = q_1, \ldots, q_m \rangle$ of two consistent subsequences $T = \langle p_1, \ldots, p_n \rangle$ and $U = \langle q_1, \ldots, q_m \rangle$ with $p_n = q_1$ is not necessarily consistent. We call a physics model *concatenable* if this is the case. An example of a concatenable model is one that limits only the speed of the entity. Concatenable models generally allow more efficient algorithms.

Not all physics models are concatenable: for example, a model limiting both the speed and the acceleration is not concatenable. See Figure 1 (left): both $T = \langle p_1, p_2 \rangle$ and $U = \langle p_2, p_3 \rangle$ are

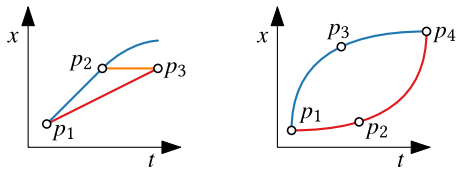

Fig. 1. (Left) In an acceleration-bounded model $\langle p_1, p_2, p_3 \rangle$ is not consistent, even though $\langle p_1, p_2 \rangle$ and $\langle p_2, p_3 \rangle$ (and even $\langle p_1, p_3 \rangle$) are. (Right) A consistent subtrajectory through p_2 (red) may require a different speed at p_4 than a subtrajectory that includes p_3 (blue).

consistent, but $\langle p_1, p_2, p_3 \rangle$ is not. The main problem is that sequences U and T essentially require the entity to have two different speeds at p_2. The two subtrajectories U and T are concatenable in the acceleration-bounded model, under the condition that they have the same speed at their common measurement. To capture this, we define a notion of *conditional consistency*, denoted $C(T \mid \gamma)$, in which a trajectory T is consistent, provided that it has a witness satisfying condition γ. In case $C(T \mid \gamma)$ and $C(U \mid \gamma')$ imply that the concatenation of T and U is consistent, we say that the physics model is *conditionally concatenable*. Hence, the model with bounded acceleration is conditionally concatenable, using the condition that the speed at the common measurement is the same. The speeds attainable at a certain measurement may depend on the subtrajectory so far, as illustrated in Figure 1 (right).

Results and organization. We present three algorithms and the results of computational experiments investigating the efficacy of our methods. Specifically, in Section 2, we describe a simple, optimal algorithm that runs in $O(nk)$ time for any concatenable physics model allowing $O(1)$ consistency checks between two measurements. We then describe a more efficient algorithm that runs in $O(n \log n \log^2 k)$ time for the speed-bounded model in Section 3. Our final algorithm, described in Section 4, uses an acceleration-bounded model, that can optionally also bound the speed. This algorithm runs in $O(nk^2 \log k)$ time under mild assumptions, that are validated by our experiments. We also present a variant of this algorithm that introduces slack in the physics model to obtain an efficient approximate algorithm that achieves the given worst-case running time without assumptions.

In Section 5, we discuss the results of a series of computational experiments on real-world data using our open-source implementation.[1] Specifically, we compare the quality of our algorithms to simple greedy approaches and conclude that our algorithms are more reliable, especially for trajectories with more than minor levels of noise. We also observe that the speed-bounded model approximates the acceleration-bounded model, though there is some dependency on the dataset. Finally, we also briefly investigate how sensitive our results are to the model parameters: though speed bound is quite sensitive, the acceleration bounds have comparatively little influence on the number of outliers detected. We conclude with a discussion of our results in Section 6.

Related work. Outlier detection is necessary to cope with imprecise data. Hence, many different methods have been developed for various contexts. A general survey of outlier detection is given in Reference [9]; see also Reference [8] for a survey focusing on data with a temporal component, including trajectories. For trajectories, outlier detection has mainly focused on finding outlying trajectories in sets of trajectories [7, 11, 13, 20] and not on finding outlying measurements in one trajectory. At a glance, detecting outlying measurements resembles trajectory simplification and trajectory smoothing, both well-studied topics: refer to Reference [21] for a survey. However,

[1] Released as part of the MoveTK library, https://movetk.win.tue.nl/.

simplification generally aims to minimize the number of measurements while still accurately describing the trajectory: this typically retains outliers as these are "salient."

Physics models are often used in trajectory processing. Kalman filtering [12], for example, is based upon a linear model for physical motion; its extensions handle more complex, nonlinear models. Note, however, that Kalman filtering changes the measurement positions rather than selecting a consistent subset. In a similar vein, physics models are used to reconstruct trajectories from data, replacing subtrajectories that cannot be physically realized with ones that can [14, 18]. Here, unrealistic subtrajectories are detected using a local time window, sliding over the trajectory.

Given a trajectory and physics model, we aim to determine the maximum number of measurements that can be explained through a path adhering to the model. As such, our problem bears some resemblance to two other problems: computing a longest common subsequence and map matching. The former asks to compute the maximum subsequence of two strings [3] and has also been used to compute trajectory similarity [19]. Contrasting our approach, longest common subsequence requires that both trajectories are known. The latter, map matching, is the problem of determining the driven route through a street network, given a noisy trajectory. Myriad algorithms exist, e.g., References [1, 15]; see Reference [21] for an overview. Dealing with noise naturally arises in this application. Though we do not investigate this here, explicit outlier removal before map matching may improve results of simple and faster algorithms; postprocessing map matching results using our methodology may give rise to more realistic results. However, the primary difference is that our method does not rely on knowledge of the street network: the space of potential paths is defined implicitly and as such our methodology is more broadly applicable to movement that does not follow a predefined network (pedestrians, ships, airplanes).

2 CONCATENABLE CONSISTENCY MODEL

We assume an arbitrary concatenable physics model that allows consistency checks between two measurements in $O(f(n))$ time for some function f; typically, $f(n) = O(1)$. We follow the methodology of the Imai-Iri line-simplification algorithm [10]. Let $G = (V, A)$ be a directed acyclic graph with a vertex v_i for each measurement p_i of T and an edge from v_i to v_j if and only if $C(p_i, p_j)$. This graph has $O(n^2)$ edges; each can be tested in $O(f(n))$ time. By concatenability, a path in G describes a consistent subsequence. Since G is directed and acyclic, we compute a longest path in G, and thus a maximum consistent subsequence of T, in $O((|V| + |A|)f(n)) = O(n^2 f(n))$ time.

We now develop an output-sensitive variant of this algorithm. Rather than constructing the full graph, we build a subgraph in which each vertex v has at most one incoming edge (u_v, v). In particular, u_v and v are the last measurements of a longest consistent subsequence T_v ending in v. Let ℓ_v denote the length of T_v.

Theorem 2.1. *Consider a concatenable physics model that allows checking the consistency of a pair of measurements in $O(f(n))$ time. A maximum consistent subsequence of a trajectory T with n measurements can be computed in $O(nkf(n))$ time, where k is the number of outliers.*

Proof. We handle the measurements in chronological order, maintaining a linked list $\mathcal{L}$ that stores, for each handled measurement v, the value ℓ_v and the predecessor u_v in T_v. $\mathcal{L}$ is ordered by the lengths ℓ_v in descending order. For a new measurement w, we traverse $\mathcal{L}$ to find the first measurement v consistent with w. Since $\mathcal{L}$ is ordered, we have thus found a longest consistent subsequence of length $\ell_v + 1$ ending in w. We now walk backwards in $\mathcal{L}$ and add w to the appropriate place in the list.

After we have handled all measurements, the maximum consistent subsequence can be retrieved in $O(n - k)$ time, by starting at the head of the list and following the predecessor pointers. For each of the $n - k$ measurements that end up in the longest subsequence, we perform one successful

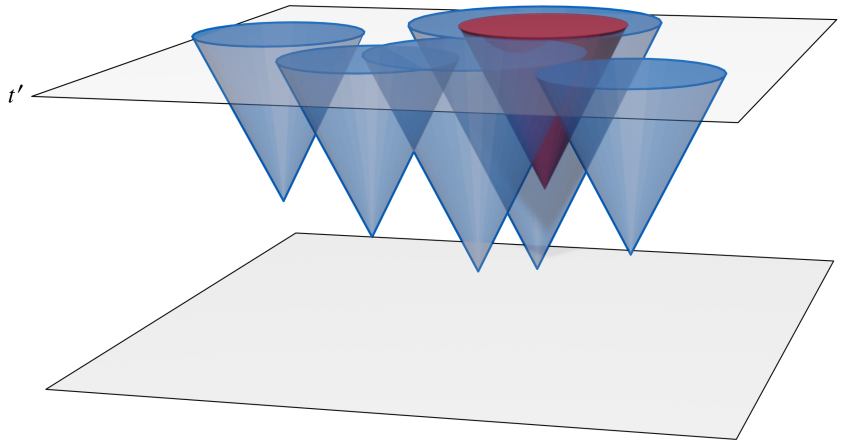

Fig. 2. Each measurement defines a reachable region (a cone), that intersects the plane at time t' in a disk. These disks define an AWVD. A measurement p_j is consistent with an earlier measurement p_h if (and only if) its cone (shown in red) is contained in the cone of p_h.

check preceded by at most k failed checks, and for the k outliers we perform at most n checks. This gives a total of $O(nkf(n))$ time performing the checks. The time for inserting a measurement in $\mathcal{L}$ can be charged to the number of checks it performs: this takes $O(nk)$ time in total. Hence, the total running time for the algorithm is $O(nkf(n))$. □

As the speed-bounded model allows to check consistency between two measurements in constant time, we thus obtain the following running time for this model. We can, however, improve upon this algorithm in case the trajectory has many outliers, as discussed in the next section.

Corollary 2.2. *For the speed-bounded model, a maximum consistent subsequence of a trajectory T with n measurements can be computed in $O(nk)$ time, where k is the number of outliers.*

3 THE SPEED-BOUNDED MODEL IN 2D

We now consider the speed-bounded model, with maximum speed v^+, and trajectories in $\mathbb{R}^2$. We present an $O(n \log n \log^2 k)$-time algorithm to find a maximum consistent subtrajectory in this model. To this end, we develop an insertion-only data structure that, given a measurement q, can determine the length of a maximum consistent subsequence ending at q in $O(\log^3 n)$ time. Insertions are supported in $O(\log^3 n)$ time. By incrementally building the data structure in chronological order, we can determine the maximum consistent trajectory in $O(n \log^3 n)$ time. With a more careful analysis this can be improved to $O(n \log n \log^2 k)$ time.

3.1 A Consistency Data Structure

Let P be a subset of measurements from T, and let $\hat{t}$ be the time of the last measurement in P. We develop a data structure $\mathcal{D}$ that can efficiently answer *consistency-queries* on P. That is, for a given new query measurement q occurring at time $t \geq \hat{t}$, we can test whether there is a measurement in P consistent with q. We view the measurements in P as points in $\mathbb{R}^3$, with the third axis being time, that is, $p_i = (x_i, y_i, t_i)$. Measurements p_j, with $j > i$, that are consistent with p_i must lie inside a cone that starts at p_i and has radius $v^+(t - t_i)$ at time $t \geq t_i$; see Figure 2. We call this cone the *reachable region* of p_i; testing whether p_j is in the reachable region of p_i takes $O(1)$ time.

To determine if a measurement q at time $t_q \geq \hat{t}$ is consistent with any measurement of P, we use an *additively weighted Voronoi diagram* (AWVD) [5, 6]. Given a set of disks with centers $\{v_1, \ldots, v_l\}$

and radii $\{r_1, \ldots, r_l\}$, this diagram partitions the plane into cells $\{c_1, \ldots, c_l\}$ associated with the disks, such that for any point $v \in c_i$ it holds that $d(v, v_i) - r_i \leq d(v, v_j) - r_j$, for all $v_j \neq v_i$. Here, d is a distance measure (in our case the Euclidean distance), and equality holds only on boundaries between cells.

We construct an AWVD on the measurements in P by using the locations as the centers and picking $r_i = \mathrm{v}^+(t' - t_i)$ for every measurement for some arbitrary $t' > t_n$. Observe that a measurement p_j is consistent with p_h if the reachable region of p_j at t' is inside the reachable region of p_h at t' (see Figure 2). We preprocess the AWVD for point location queries. Let $\mathcal{D}$ denote the resulting data structure, which we refer to as a *consistency data structure*. We can now query $\mathcal{D}$ with a new measurement $q = p_q$, giving us a previous measurement p_c and a distance s_c between the disks (p_q, r_q) and (p_c, r_c), given by $s_c = d(p_q, p_c) - r_q - r_c$, where d measures the Euclidean distance between the points in the plane, that is, ignoring the temporal component. The following lemma then gives us that $\mathcal{D}$ can be used to answer consistency queries.

Lemma 3.1. *Let $\mathcal{D}$ be a consistency data structure on a set P of measurements and let $q = p_q$ be a query measurement, resulting in measurement p_c on $\mathcal{D}$. If $s_c \leq -2r_q$ for the resulting distance s_c of p_c with p_q, then p_c is consistent with q. Otherwise, no measurement in P is consistent with q.*

Proof. We denote $w_j = (x_j, y_j)$ for compactness. Let $p_q = (x_q, y_q, t_q)$ be the qth measurement in a trajectory and let $\mathcal{D}$ be the consistency data structure on a subset $P \subseteq \{p_1, \ldots, p_{q-1}\}$. Let p_c be the measurement associated with the found cell in $\mathcal{D}$ containing (x_q, y_q) and let $s_c = d(w_q, w_c) - r_q - r_c$.

Since $s_c = d(p_q, p_c) - r_q - r_c$, we can rewrite the inequality $s_c \leq -2r_q$ to $d(p_q, p_c) \leq r_c - r_q$, Applying the definition of r_c and r_q, the right-hand side can be rewritten to $\mathrm{v}^+(t' - t_c) - \mathrm{v}^+(t' - t_q) = \mathrm{v}^+(t_q - t_c)$. In other words, we have that the distance between the measurements is at most the maximum distance that the object can travel in the given time span. Hence, p_q is consistent with p_c.

Analogously, if the given inequality does not hold, then the distance between the two measurements is larger than what the object can cover when traveling at maximum speed: They are not consistent.

To show that no other measurement in P can be consistent, observe that the definition of the AWVD gives us that $d(p_q, p_c) - r_c \leq d(p_q, p_k) - r_k$ for all $p_k \in P \setminus \{p_c\}$. Subtracting r_q from both sides, we get that $d(p_q, p_c) - r_c - r_q = s_c \leq d(p_q, p_k) - r_k - r_q$. Thus, if $s_c > -2r_q$, we must also have that $d(p_k, p_c) > r_k - r_q$. Thus, all measurements p_k are not consistent with measurement p_q. □

We can construct the AWVD for a set of m measurements and preprocess this AWVD for point-location queries in $O(m \log m)$ time [5, 6]. The resulting data structure uses $O(m)$ space, and can answer point-location queries in $O(\log m)$ time. Since a single consistency check takes constant time, we can also answer consistency queries in $O(\log m)$ time.

3.2 Supporting Insertions

Next, we describe how to extend our consistency data structure to support insertions. Testing whether a measurement is consistent with any previous measurement of a subsequence of T is a decomposable search problem. Thus, we use the approach by Bentley and Saxe [2] to turn our consistency data structure into an efficient insertion-only data structure.

For a set of m measurements, we maintain $O(\log m)$ instances of our static data structure $\mathcal{D}_1, \ldots, \mathcal{D}_{O(\log m)}$ (see Figure 3). Every measurement is in one of these $O(\log m)$ data structures. Data structure $\mathcal{D}_i$ has size 2^i. On insertion, we create a new $\mathcal{D}_1$ with the inserted measurement.

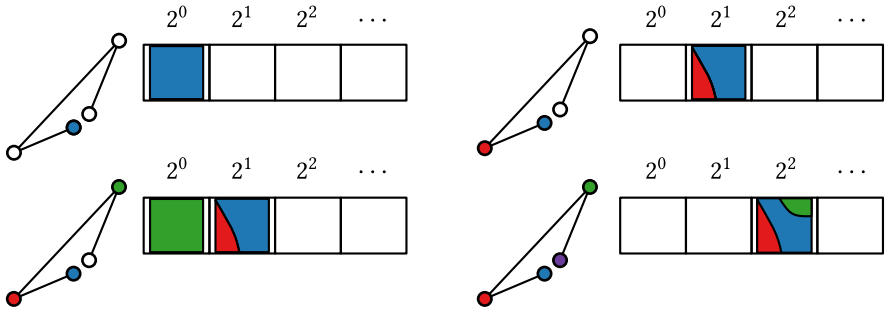

Fig. 3. Inserting elements using Bentley-Saxe. The colored measurements in the trajectory are the elements in the insertion-only consistency data structure.

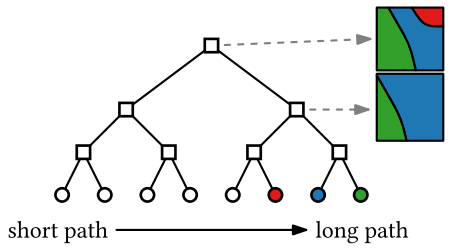

Fig. 4. Data structure for maximum subsequence queries.

When we get two data structures of same size 2^i, we remove both and replace them by a single data structure of size 2^{i+1}. We repeat this process until all data structures have a unique size. To answer a query, we simply query all $O(\log m)$ data structures.

The above construction together with the consistency query structure gives $O(\log^2 m)$ time for a query and $O(\log^2 m)$ amortized time for an insertion. These bounds can be made worst case as well [17]. We summarize our results in the following lemma.

Lemma 3.2. *There is a consistency data structure $\mathcal{D}$ that can store a subset P of m measurements from T and can answer consistency queries for query points q at time $t \geq \hat{t}$, in $O(\log^2 m)$ time, and supports insertions in $O(\log^2 m)$ time. Here, $\hat{t}$ denotes the time of the last measurement currently in P. The data structure uses $O(m)$ space.*

3.3 Maximum Subsequence Queries

We now use the data structure from Lemma 3.2 to build a dynamic data structure that, for a new query measurement $q = p_q$ can determine the length ℓ_q of a longest consistent subsequence $T_q \subseteq P$ ending at q. We store the measurements in $p \in P$ in the leaves of a balanced binary tree $\mathcal{T}$, ordered by the length ℓ_p of the longest consistent subsequence ending in p. Each internal node v with right child r corresponds to a subset $P_v \subseteq P$, and stores the minimum ℓ_p, with $p \in P_r$, occurring in its right subtree, and a consistency data structure $\mathcal{D}_v$ built on the set P_r (see Figure 4).

Given a query measurement q, we find a measurement $u \in P$ consistent with q with maximum length ℓ_u. It then follows that a maximum-length consistent subsequence T_q ending in q has length $\ell_u + 1$, and that u is the predecessor of q in T_q. To find u, we start at the root v and query $\mathcal{D}_v$ to test whether any measurement in the right subtree is consistent with q. If so, then we repeat the process in the right child. If not, then we move to the left child. This way, we get the longest-path measurement that is consistent with our query measurement q.

To insert a new measurement q, we find the leaf corresponding to length ℓ_q and insert q in the appropriate associated data structures of all ancestors along this root to leaf path. To keep the tree $\mathcal{T}$ balanced, we implement it using a BB[α] tree [4, 16]. When a subtree rooted at a node v becomes unbalanced, we rebuild it and its associated data structures from scratch.

With the lemma below, we prove that this data structure can be implemented efficiently.

Lemma 3.3. *There is a data structure $\mathcal{T}$ that can store a subset P of m measurements from T and that can find, given a query measurement q at time $t_q \geq \hat{t}$, (the length ℓ_q of) a longest consistent subtrajectory T_q ending in q in $O(\log^3 m)$ time. The data structure uses $O(m \log m)$ space and supports insertions in $O(\log^3 m)$ amortized time. Here, $\hat{t}$ denotes the time of the last measurement currently in P.*

Proof. To answer a query, we follow a path from the root down to a leaf, and query the associated data structure at each node. Each such query takes $O(\log^2 m)$ time (Lemma 3.2), and thus the total query time is $O(\log^3 m)$. Since each measurement is stored in the associated data structure of $O(\log m)$ nodes, the total space use is $O(m \log m)$, and the total direct cost of an insertion is $O(\log^3 m)$. To prove the lemma, we need to bound the costs due to rebalancing operations.

Assume an insertion of a node triggers a rebalance operation for a subtree v of m_v elements, and $C(m_v)$ the total construction time: the time that it takes to acquire all elements in the subtree and construct a perfectly balanced tree (with its associated data structures). By the BB[α] definition, $(1-2\alpha)m_v - 2$ insertions must have occurred in v to trigger the rebalance operation. This implies that the amortized rebalance time per insertion is $\frac{C(m_v)}{(1-2\alpha)m_v - 2}$.

Consider the tree after rebalancing. At height $h > 0$, we have intermediate nodes, each requiring a data structure $\mathcal{D}$ constructed on $O(2^h)$ elements. There are $O(m_v/2^h)$ nodes at height h. In total, to construct nodes with height h, we require $O((m_v/2^h)2^h \log^2(2^h)) = O(m_v h^2)$ time. Let $H = O(\log m_v)$ be the height of the entire subtree; summing up the construction time of all heights in the subtree gives the total construction time $C(m_v) = \sum_{h=1}^{H} O(m_v h^2) = O(m_v H^3) = O(m_v \log^3 m_v)$. Amortized, this construction cost and hence the entire insertion time is $O(\log^3 m)$.[2] □

3.4 Maximum Consistent Subtrajectories

To compute a maximum-length consistent subtrajectory of T, we process all measurements in chronological order. For each, we simply query the data structure from Lemma 3.3, and then insert it. This results in an $O(n \log^3 n)$-time algorithm. Next, we show that we can improve this to $O(n \log n \log^2 k)$, where k is the number of outliers.

Lemma 3.4. *For two consistent measurements p_i and p_j with $i < j$, the reachable region for p_j for all $t > t_j$ is contained in the reachable region of p_i.*

Proof. In the three-dimensional space (with the third dimension being time), the reachable region of each measurement is an upward cone starting at the measurement, with slope v^+. As p_j is consistent with p_i, the former lies inside the latter's cone. As their direction and slope are the same, the cone for p_j is thus contained in the cone for p_i.

We can equally see this in two-dimensional space. Consider an arbitrary time $t > t_j$. A hypothetical measurement p^* at time t consistent with p_j must be within distance $\mathrm{v}^+(t - t_j)$. Since p_i and p_j are consistent, we know that their distance is at most $\mathrm{v}^+(t_j - t_i)$. Through triangle inequality, we thus know that the distance between p_i and p^* is at most $\mathrm{v}^+(t - t_j) + \mathrm{v}^+(t_j - t_i) = \mathrm{v}^+(t - t_i)$. This readily implies that p^* is consistent with p_i as well. □

[2]Note that in our improved bound in Theorem 3.5 the total reconstruction time $C(m_v)$ is simply $O(m_v \log m_v \log^2 k)$, as rebuilding the associated data structure of a node takes $O(m_v \log^2 k)$ time rather than $O(m_v \log^2 m_v)$ time.

From the definition of the AWVD, we know that if a disk c_1 is strictly inside another disk c_2, then c_1 will have an empty associated cell in the diagram. Combining this with Lemma 3.4 shows that, any subset of $m \geq 1$ measurements thus produces a diagram with at most $\min(m, k)$ cells. Hence, a static consistency data structure uses only $O(\min(m, k))$ space, and querying it requires $O(\log(\min(m, k)))$ time. When we insert a new measurement p_j into our insertion-only data structure, we first query the data structure to decide whether p_j is consistent with some earlier measure p_i. If so, then we simply discard p_j rather than inserting it; even when inserting additional points, the cell of p_i will contain that of p_j. The query and insertion time therefore both become $O(\log^2 \min(m, k))$.

It now follows that the associated data structure $\mathcal{D}_v$ of every node in $v \in \mathcal{T}$ has size only $O(\min(n_v, k))$, thus querying it requires only $O(\log^2 k)$ time, and thus $O(\log n \log^2 k)$ time in total. Similarly, inserting a new measurement takes amortized $O(\log n \log^2 k)$ time.

Theorem 3.5. *Given a 2D trajectory T with n measurements, of which k are outliers, we can compute a maximum consistent subsequence of T for the speed-bounded model in $O(n \log n \log^2 k)$ time.*

4 THE ACCELERATION-BOUNDED MODEL

We now consider 1D trajectories where each measurement is of the form $p_i = (x_i, t_i)$. We assume a physics model where both velocity and acceleration are restricted. The velocity must lie in the range $[\mathrm{v}^-, \mathrm{v}^+]$ for constants $\mathrm{v}^-, \mathrm{v}^+$. In addition, the acceleration must lie in the range $[a^-, a^+]$ for constants a^-, a^+. For simplicity, we assume $a^- < 0$ and refer to deceleration as acceleration with a negative value.

For this acceleration-bounded model, we can still test in constant time if two points p_i and p_j are consistent: we can check if the distance between the two measurements can be traveled using velocities that lie in the range $[\mathrm{v}^-, \mathrm{v}^+]$. If there exists a velocity at p_i such that the required velocity at p_j can be reached by accelerating, then the pair $\langle p_i, p_j \rangle$ is consistent. Recall, however, that a physics model that limits acceleration is not concatenable: there may be a triplet of measurements $\langle p_1, p_2, p_3 \rangle$ for which the measurements are pairwise consistent, but the entire sequence is not (see Figure 1 (left) for an example). Hence, we cannot use the algorithm described in Section 3.

In Section 4.1, we describe a dynamic programming algorithm that explicitly computes the velocities achievable at every measurement and, using these velocities, finds the length of a maximum-length consistent subtrajectory. In Section 4.2, we show how to retrieve the actual consistent trajectory. The running time of the dynamic program and of the retrieval procedure depends on the maximal fragmentation of the velocity intervals, which can arise during the DP. In Section 4.3, we first argue that this number can be as large as $\Omega(n)$ for a linear number of sets of velocities. It is easy to see that the maximal fragmentation is at most $O(2^n)$; however, it is unlikely that this bound would ever be reached in practice. In the following, we consider an acceleration-bounded model with some slack in the acceleration bounds, modeling real-world imprecision. This slack allows us to prove a linear upper bound for the fragmentation of any set of velocities. Finally, in Section 4.4, we explain how to extend the acceleration-bounded model to dimensions greater than one.

4.1 Computing the Maximum Length of a Physically Consistent Subtrajectory

A subtrajectory T is generally not concatenable with another subtrajectory T' under the acceleration-bounded model, but is conditionally concatenable with T' when the velocities at measurements that they have in common are the same (see Section 1). Intuitively, this follows from the fact that a bound on the acceleration prevents (discontinuous) jumps in velocity. Based on this, we observe the following:

Observation 1. *A (sub)trajectory $S = \langle p_1, \ldots, p_m \rangle$ is consistent in the acceleration-bounded model if and only if there are velocities $\langle v_1, \ldots, v_m \rangle$ such that for all $i \in \{1, \ldots, m-1\}$, we have that $C(p_i, p_{i+1} \mid \mathrm{v}_i = v_i, \mathrm{v}_{i+1} = v_{i+1})$.*

Observation 1 implies that our problem has an optimal substructure. Suppose we have found all subtrajectories of some length ℓ that are consistent. If we now want to know whether a subtrajectory of length $\ell + 1$ exists, then we have to determine only whether there is a measurement p' such that the observation holds for one of the subtrajectories when we add p' at the end. That is, there should be witness paths for both the subtrajectory and the trajectory between the last measurement of the subtrajectory and p', that have a common velocity at the last measurement of the subtrajectory. Hence, we can apply the dynamic programming paradigm to find the optimal length for which a subtrajectory is physically consistent.

More formally, for each measurement p_i and each possible length $\ell \in \{1, \ldots, n\}$, we maintain the set of velocities $\mathcal{I}(\ell, i)$ such that for every velocity $v \in \mathcal{I}(\ell, i)$, a subsequence $S = \langle \ldots, p_i \rangle$ ending at p_i of length ℓ exists that is physically consistent and has velocity v at p_i, so that $C(S \mid \mathrm{v}_i = v)$. Let ℓ^* be the maximum value, such that a measurement p_i exists for which some set $\mathcal{I}(\ell^*, i)$ is non-empty. It follows that the maximum consistent subtrajectory of T has length ℓ^*.

Given the set of possible velocities $\mathcal{I}(\ell, h)$ at p_h, we can then determine whether a consistent subsequence of length $\ell + 1$ exists that ends at a later measurement p_i by using the conditional concatenability property: If we find velocities $v_h \in \mathcal{I}(\ell, h)$ and $v \in [\mathrm{v}^-, \mathrm{v}^+]$ such that $C(p_h, p_i \mid \mathrm{v}_h = v_h \wedge \mathrm{v}_i = v)$, then a consistent subsequence $\langle \ldots, p_h, p_i \rangle$ of length $\ell + 1$ exists. Hence, we obtain the following recurrence for $\mathcal{I}(\ell, i)$:

$$\mathcal{I}(\ell, i) = \begin{cases} \emptyset, & i < \ell, \\ \{v \mid \exists h : C(p_h, p_i \mid \mathrm{v}_i = v)\}, & \ell = 2, \\ \{v \mid \exists h : C(p_h, p_i \mid \mathrm{v}_h \in \mathcal{I}(\ell - 1, h), \mathrm{v}_i = v)\}, & \ell > 2. \end{cases}$$

Moreover, we prove in Lemma 4.1 below, that when the entity directly travels from p_i to p_j, and leaves p_i with velocity v_i, the possible velocities with which it can arrive at p_j form a connected interval. It follows that the sets $\mathcal{I}(\ell, i)$ are actually sets of intervals.

Lemma 4.1. *Let p_i and p_j be measurements with $t_i < t_j$, and let $v_1 \leq v \leq v_2$ be velocities. If $C(p_i, p_j \mid \mathrm{v}_j = v_1)$ and $C(p_i, p_j \mid \mathrm{v}_j = v_2)$, then we also have $C(p_i, p_j \mid \mathrm{v}_j = v)$.*

Proof. $C(p_i, p_j \mid \mathrm{v}_j = v_1)$ and $C(p_i, p_j \mid \mathrm{v}_j = v_2)$ imply that there are two witnesses: paths $\pi_1(t)$ and $\pi_2(t)$ between p_i and p_j that travel $\Delta x = x_j - x_i$ distance, obey the physics model and have velocity v_1, respectively, v_2 at p_j. Let $a_1(t)$ and $a_2(t)$ denote the acceleration functions describing these paths.

The traveled distance Δx between t_i and t_j using any acceleration function $a'(t)$ and velocity v' at p_j is given by

$$\Delta x = (t_j - t_i)\left(v' - \int_{t_i}^{t_j} a'(t)\mathrm{d}t\right) + \int_{t_i}^{t_j}\int_{t_i}^{t} a'(t')\mathrm{d}t'\mathrm{d}t. \tag{1}$$

Any new path π^*, which we create using convex combinations $v = \beta v_1 + (1-\beta)v_2$ and $a(t) = \beta a_1(t) + (1-\beta)a_2(t)$ for $\beta \in [0, 1]$, travels exactly the same distance by linearity of the integrals. Since π^* was created via convex combinations, we also know that it satisfies the velocity and acceleration constraints, since its velocity and acceleration always lie between the original velocities and accelerations at any time t in $[t_i, t_j]$. Hence, π^* is a witness that implies $C(p_i, p_j \mid \mathrm{v}_j = v_1 + (1-\beta)v_2)$ for any $\beta \in [0, 1]$. □

Lemma 4.2 below shows how to propagate a single speed interval from p_i to p_j in constant time. The problem is clearly computable, and has $O(1)$ input complexity: two measurements and a

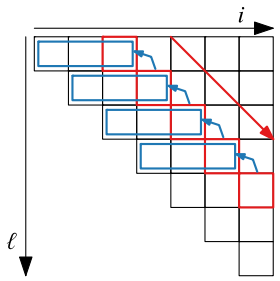

Fig. 5. The order for computing $\mathcal{I}(\ell, i)$.

single interval of velocities. As such, the lemma readily follows. The complete proof of Lemma 4.2, including a derivation of the precise propagation function, requires some lengthy mathematical derivations, which we relegate to Appendix A as to not break the flow of the article.

Lemma 4.2. *Let p_i and p_j be two measurements with $i < j$, and let I be an interval of velocities at p_i. The interval $I' = \{v \mid C(p_i, p_j \mid v_i \in I \wedge v_j = v)\}$ of achievable velocities can be computed in $O(1)$ time.*

Let now $\delta(\ell, i)$ denote the number of intervals in $\mathcal{I}(\ell, i)$. We refer to $\delta(\ell, i)$ as the *fragmentation* of $\mathcal{I}(\ell, i)$. Let $\delta_{\max}$ be the maximum fragmentation over all ℓ and i. Using the recurrence for $\mathcal{I}$ defined earlier, we can compute all values $\mathcal{I}(\ell, i)$ using dynamic programming. We compute the $\mathcal{I}(\ell, i)$ values by increasing distance k' from the diagonal, and stop once there are no more reachable speeds. That is, we start by computing all $\mathcal{I}(i, i)$, for increasing i. Observe that these values correspond to having $k' = 0$ outliers. Once we have all sets $\mathcal{I}(i - k', i)$ for some k', we continue with the $\mathcal{I}(i - (k' + 1), i)$ sets (see Figure 5). Let k be the number of outliers in a maximum-length consistent subtrajectory, then all sets of speed intervals $\mathcal{I}(i - (k + 1), i)$ will be empty. Hence, the algorithm finishes after at most $k + 1$ "rounds." To compute a single entry $\mathcal{I}(i - k', i)$, we have to propagate the speed intervals from at most k other entries (since all sets $\mathcal{I}(\ell, i)$ with $\ell > i$ are also empty). It follows that in total, this procedure takes $O(nk^2 \cdot P)$ time, where P is the time required to propagate all speed intervals in some set $\mathcal{I}(\ell', i)$ to $\mathcal{I}(\ell, j)$. Every set $\mathcal{I}(\ell, i)$ contains at most $\delta_{\max}$ intervals, which we keep in sorted order. Propagating a single interval takes constant time (see Lemma 4.2), and merging it with the intervals already in $\mathcal{I}(\ell, i)$ then takes $O(\log \delta_{\max})$ time.

Theorem 4.3. *Let T be a 1D trajectory with n measurements. Under the acceleration-bounded model, the maximum length of a physically consistent subtrajectory of T can be computed in $O(nk^2 \delta_{\max} \log \delta_{\max})$ time using $O(nk\delta_{\max})$ space, where k denotes the number of outliers and $\delta_{\max}$ the maximum fragmentation.*

4.2 Retrieving the Physically Consistent Subtrajectory

The dynamic program computes the length ℓ^* of a maximum consistent subsequence. Generally, keeping track of the choices made in a dynamic program allows easy recovery of the actual answer, that is, the actual subsequence. However, we need slightly more, as we join overlapping intervals and thus no longer store which previous measurements led to parts of that interval—generally there may not be only one measurement for an interval.

We could opt for storing a minimum cover of the interval in a cell instead, which we can easily obtain while computing the union. However, this increases memory requirements. Alternatively, we can also use "backpropagation." That is, we extract S itself using the speed intervals in the sets $\mathcal{I}(\ell^*, i)$. We take an interval $I \in \mathcal{I}(\ell^*, i)$ and use an inverse propagation to find a measurement p_h such that $\mathcal{I}(\ell^* - 1, h)$ has a nonempty interval of speeds at which the interval of $\mathcal{I}(\ell^*, i)$ is reachable. We repeat this backpropagation, until the start of the subsequence is reached.

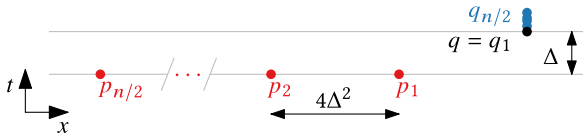

Fig. 6. Instance with $\Omega(n)$ disjoint speed intervals at c.

To do this efficiently, we leverage that the intervals in $\mathcal{I}(\ell^* - 1, h)$ are sorted by the dynamic program already. Thus, we use backpropagation in $O(1)$ time by Lemma 4.2 to find the velocity interval I' at p_h. We then find whether one of the intervals in $\mathcal{I}(\ell^* - 1, h)$ intersects I' using binary search in $O(\log \delta_{\max})$ time. Thus, computing the subtrajectory after the dynamic program takes $O((n-k)\log \delta_{\max})$ time.

4.3 Bounding the Maximum Fragmentation

The running time of the dynamic program described in Section 4.1 depends on the maximum fragmentation $\delta_{\max}$, that is, the maximum number of intervals in any set of velocities $\mathcal{I}(\ell, i)$. Recall that $\mathcal{I}(\ell, i)$ may contain more than one velocity interval (see Figure 1 (right)). We argue in the following lemma that the fragmentation of a linear number of sets $\mathcal{I}(\ell, i)$ may even be $\Omega(n)$.

Lemma 4.4. *There is a 1D trajectory T with n measurements such that $\Omega(n)$ sets of velocities $\mathcal{I}(\ell, i)$ have fragmentation $\Omega(n)$.*

Proof. We construct a trajectory $T = \langle p_1, \ldots, p_{n/2}, q_1, \ldots, q_{n/2}\rangle$ such that for parameters $v^- = 0$ and $a = a^+ = -a^- = 1$, we get $\Omega(n)$ speed intervals at each point p_j, $j > n/2$.

Let $\Delta > 0$ be some real number. We place the points at $p_i = (-4i \cdot \Delta^2, 0)$, for $i \in \{1, \ldots, n/2\}$, and $q_j = (0, \Delta)$, for $j \in \{1, \ldots, n/2\}$ (see Figure 6). We can ensure that all points have unique time stamps by offsetting them by some arbitrarily small time. This construction ensures that a consistent subtrajectory cannot use two points p_i and p_j simultaneously. We now claim that every point p_i together with point $q = q_1$ generates a consistent subtrajectory $\langle p_i, q\rangle$ for which the possible speeds at q are given by the interval $I_i = [v_i - \Delta, v_i + \Delta]$ with $v_i = 4i\Delta$. Observe that these intervals are all pairwise disjoint. Since the other points q_j are arbitrarily close to q, the same argument shows that we get $\Omega(n)$ speed intervals at those points.

Since $a = 1$ and the time between p_i and q is short, the velocity that the entity has at p_i must be similar to its velocity at q. If the speed at p_i differs too much from the velocity at q, then the entity cannot actually reach q: it will either travel too little or too far. Next, we formalize this argument.

To derive a contradiction, assume that there is a consistent subtrajectory in which an entity travels from p_j, with $j \neq i$, to q and arrives at q with speed $v \in I_i$. Since $v^- = 0$, the distance that any entity can and has to travel to go from p_j to q is exactly $4j\Delta^2$. The entity covers this in Δ time, and hence its average speed must be $4j\Delta$. Since $a = 1$, it then follows that at any time in the time interval $[0, \Delta]$ its speed lies in the range $[4j\Delta - \Delta, 4j\Delta + \Delta]$.

The entity achieves speed $v \in I_i = [v_i - \Delta, v_i + \Delta]$ at q. So, we have $4j\Delta - \Delta \leq v \leq v_i + \Delta$. Using that $v_i = 4i\Delta$, we get $j \leq i + \frac{1}{2}$. As i and j are natural numbers, we get $j \leq i$. Symmetrically, we have $v_i - \Delta \leq v \leq 4j\Delta + \Delta$, and get $i \leq j$. Combining these results gives $i = j$: a contradiction.

Note that in this construction all consistent subtrajectories have length two. We can easily achieve length $\ell > 2$ by prefixing the construction with a common trajectory of length $\ell - 2$; this prefix provides sufficient time between its last point and the points p_i, to allow the entity to achieve all speeds v_i at p_i. □

It is relatively easy to see that the fragmentation $\delta(\ell, i)$ is at most $O(2^i)$, since any fixed subsequence of $\langle \ldots, p_i\rangle$ yields only a single interval (refer to Lemma 4.1). To realize such a large number of intervals, they have to be packed ever more closely to the minimum or maximum al-

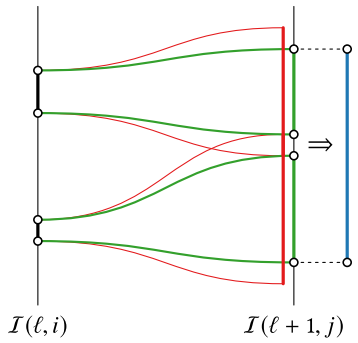

Fig. 7. Propagation using the slacked model, from p_i to p_j. Green indicates standard propagation and red slacked propagation. The result of merging the intervals is indicated in blue.

lowed speed threshold. It seems unlikely that this behavior will appear in realistic settings, and hence we expect that the fragmentation is much smaller in practice. Below, we hence describe an acceleration-bounded model, which introduces some slack in the parameters a^- and a^+, which models real-world imprecision.

An acceleration model with slack. In a real-world setting, we can assume that there is some error in the parameters a^- and a^+, which bound the acceleration. To model this error, we introduce a slack parameter $\varepsilon > 0$ for the acceleration bounds. Specifically, let $\Delta a = a^+ - a^-$ denote the difference between minimum and maximum acceleration. For our slacked bounds, we add $\frac{\varepsilon}{2}\Delta a$ to a^+ and $-\frac{\varepsilon}{2}\Delta a$ to a^-. During the dynamic program, we first propagate intervals as usual, using the actual bounds a^- and a^+. Then, we also propagate using the slacked bounds $a^+ + \frac{\varepsilon}{2}\Delta a$ and $a^- - \frac{\varepsilon}{2}\Delta a$. This is illustrated in Figure 7: the green intervals are the result of standard propagation and the red intervals are the result of slacked propagation. If two slacked intervals intersect, then we merge the corresponding standard intervals and use the merged interval for future propagation (in Figure 7 the blue interval is the result of the merge).

In the following, we give an upper bound on the size of any set of intervals $\mathcal{I}(\ell, i)$ as a function of ε. To do so, we estimate the number of disjoint intervals that can occur after propagation. First, note that at both the minimum and the maximum velocity, standard intervals can degenerate to a point. Slacked intervals, however, always have non-zero size. Consider now an interval $[\mathrm{v}, \mathrm{w}]$, which slack-propagates to a slacked interval $[\mathrm{v}_s, \mathrm{w}_s]$ of minimum size. This implies that in fact $\mathrm{v} = \mathrm{w}$, that is, the input interval degenerates to a point. We want to determine the minimum separation between v and any other input interval whose slacked propagation touches $[\mathrm{v}_s, \mathrm{w}_s]$. To this end, we compute the largest input velocity v_{hi}, which slack-propagates to v_s. The separation is now given by $|\mathrm{v} - \mathrm{v}_{hi}|$. We can now compute a coarse upper bound for the number of intervals by dividing the complete input range $\Delta a \Delta t$ (see Appendix A) by the separation:

$$\delta_{\text{prop}} = \frac{\Delta a \Delta t}{|\mathrm{v} - \mathrm{v}_{hi}|}.$$

Solving for δ_{prop} results in

$$\delta_{\text{prop}} = \left\lfloor \frac{\sqrt{2}}{2\varepsilon\sqrt{\frac{1}{\varepsilon}+1}\left(\sqrt{2\sqrt{2}\sqrt{\frac{1}{\varepsilon}+1}-1}-1\right)} \right\rfloor = O(\varepsilon^{-1/4}),$$

which proves Theorem 4.5 below.

Theorem 4.5. *Let T be a 1D trajectory with n measurements. Under the slacked acceleration-bounded model, the maximum fragmentation δ_{max} for any set of velocities $\mathcal{I}(\ell, i)$ is $O(n\varepsilon^{-1/4})$.*

4.4 Extending to Higher Dimensions

The algorithm described above works for one-dimensional data. This may be realistic in some scenarios: For example, if we track contestants in a race along a predefined route, then the known route defines an approximately one-dimensional space. However, in most cases, movement is in two or even three dimensions. There are various ways of generalizing the acceleration-bounded model.

There are two standard 2D "interpretations" of our algorithm: either we use the Euclidean distance between the points, or we consider the Euclidean length of the path through all intermediate measurements. In our view, the former is more suitable as we aim to remove outliers that could greatly affect distances in the latter.

Yet, assuming a linear motion between two measurements is unrealistic as well. Thus, we use the Euclidean distance between measurements only as a lower bound; the upper bound is the Euclidean distance multiplied by a constant λ. Note that an upper bound can also be derived from the current speed and acceleration bounds, but we use our simpler model in the experiments below. To propagate a velocity interval, we use the distance lower bound to determine the minimum velocity at the next measurement, and the upper bound for the maximum velocity.

Of course, the models above assume that the tracked object may turn arbitrarily fast. Effectively, this means that positive or negative velocity becomes meaningless as we can instantaneously rotate from one to the other. We thus set the minimal velocity to zero. However, the direction of movement cannot be changed arbitrarily fast in reality, especially at higher speeds. Though we can easily define various physics models to address this issue, this would require more complex algorithms: We need to know more than just speed for the propagation and thus must generalize from intervals to higher-dimensional regions.

5 EXPERIMENTS

We introduced various algorithms for computing maximum consistent subsequences of a trajectory, according to different physics models, specifically a speed-bounded and an acceleration-bounded model. The algorithms for the former are simpler and faster than for the latter. However, the acceleration-bounded model is more accurate. Through a series of experiments, we investigate the quality of our algorithms and the trade-off between them.

5.1 Algorithms

We use the following seven algorithms in our experiments. The first two refer to our optimal output-senstive algorithms described above, their running time depending on the number of outliers. Additionally, we use three comparison algorithms to investigate the quality of our methods with respect to simpler algorithms. These algorithms are two variants of an incremental greedy algorithm (under both physics models) and a local greedy method (under the speed-bounded model). We implemented all algorithms in C++; these implementations are open source and available as part of the MoveTK library.[3]

[OSB] Optimal Speed-bounded. This algorithm implements the method of Section 2, under the speed-bounded model.

[3] https://movetk.win.tue.nl/.

[OAB] Optimal Acceleration-bounded. This algorithm implements the method of Section 4. We use the 2D generalization, using $\lambda = 1.5$: the upper bound on the traveled distance is 1.5 times the Euclidean distance between two measurements.

[GSB/GAB] Greedy Speed/Acceleration-bounded. We greedily build a consistent subsequence by testing whether the next considered measurement is consistent with the last measurement in the current subsequence under the **speed-bounded model (GSB)** or **acceleration-bounded model (GAB)**. For GAB, we use the propagation technique of OAB to maintain an interval of speeds—the next measurement is consistent if the interval after propagation is nonempty. These methods run in $O(n)$ time.

[SGSB/SGAB] Smart Greedy Speed/Acceleration-bounded. We keep track of multiple subsequences simultaneously. We append the next measurement to each subsequence ending in a consistent measurement; if no such subsequence exists, the measurement starts a new subsequence. The longest subsequence is returned. These methods run in $O(n^2)$ time.

[LGSB] Local Greedy Speed-bounded. Zheng [21] points to the only other method that uses a speed bound for outlier detection. However, neither survey nor the references therein give a detailed description of this heuristic method. We hence compare against our interpretation of the sketch provided by Zheng [21]. We construct a graph with a vertex per measurement. Two vertices are connected if their measurements are successive in the original trajectory and they are consistent according to the speed bound. A measurement is added to the output, if and only if its vertex is in a connected component of a user-specified size; we set this value to 3 in our experiments. Note that this local heuristic does not guarantee that the complete output is consistent according to the speed bound. This method runs in $O(n)$ time.

5.2 Datasets

We use three real-world datasets in our experiments. They are based on GPS measurements in different modes of transport, at different locations and different times.

[MB] Mountain-bike trips. This dataset consists of 1,214 trajectories of mountain-bike trips of a single cyclist in several European countries from 2012 to 2019.

[HR] The Hague-Rotterdam. This dataset provided by HERE[4] consists of 5,000 trajectories of cars and trucks in the region of The Hague-Rotterdam (the Netherlands), on a single day in January 2019.

[LA] Los Angeles. This dataset provided by HERE[4] consists of 78,658 trajectories of cars and trucks in the metropolitan area of Los Angeles, CA (USA), on a single day in September 2018.

All trajectories in the datasets have at least 10 measurements. General statistics of these datasets are provided in Table 1, along with our parameter settings per dataset, which are based on the nature and location of the general dataset; note that v^- is always set to 0 to allow the tracked object to remain stationary.

5.3 Comparing Algorithms and Models

In our analysis of the results, we look primarily at relative lengths, that is, the ratio of the number of measurements with respect to the input size. Thus, a result that filters k outliers and keeps $n - k$

[4] https://www.here.com/.

Table 1. Summary of the Complexities and Speeds of Trajectories per Dataset

	trajectories	complexity			speed (km/h)		model parameters		
		mean	maximum	stddev	mean	stddev	v^+ (km/h)	a^- (m/s^2)	a^+ (m/s^2)
MB	1,214	3,377.1	22,426	2,643.4	18.8	10.9	35.0	−3.24	1.62
HR	5,000	424.9	8,925	545.6	62.3	43.0	125.0	−10.00	10.00
LA	78,658	304.4	38,719	1,082.0	55.8	1,557.7	129.6	−10.00	10.00

The final columns list the default model parameters used throughout the experiment.

Table 2. Mean, 99 Percentile, and Maximum Running Time in Milliseconds (ms), Unless Indicated Otherwise

	MB			HR			LA			All datasets		
	mean	99%	max	mean	99%	max	mean	99%	max	mean	99%	max
OSB	5.32	31.25	203.12	1.01	15.62	62.50	0.37	15.62	359.38	0.48	15.62	359.38
GSB	2.37	15.62	15.62	0.32	15.62	15.62	0.20	15.62	31.25	0.24	15.62	31.25
SGSB	210.26	1,812.50	7,750.00	4.15	78.12	468.75	2.27	15.62	8,593.75	5.35	78.12	8,593.75
LGSB	2.59	15.62	31.25	0.30	15.62	15.62	0.21	15.62	31.25	0.25	15.62	31.25
OAB	7,194.19	89.1 s	1,074.6 s	71.03	1,640.62	14.7 s	127.04	171.88	1,754.6 s	224.83	1,156.25	1,754.6 s
GAB	4.22	15.62	31.25	0.45	15.62	15.62	0.35	15.62	46.88	0.41	15.62	46.88
SGAB	95.37	921.88	2,656.25	2.35	31.25	234.38	1.47	15.62	4,843.75	2.86	46.88	4,843.75

Running times are shown per dataset, and over all datasets. Note that the imbalance in dataset size skews the mean over all datasets strongly to the mean of the LA dataset.

measurements has a relative length of $\frac{n-k}{n} \in [0, 1]$. In the remainder, we simply use length to refer to relative length. We start, however, with a brief consideration of efficiency.

Efficiency. Table 2 provides performance statistics per algorithm and dataset in terms of running time, as performed on a HP Elitedesk 800 g2 TWR (Intel Core i5-6500 CPU at 3.20 GHz; 16 GB of RAM; 64-bit Windows 10 Enterprise). Overall, the trend between the algorithms per dataset is roughly the same. We see differences between datasets—specifically MB with respect to LA and HR—which are simply a result of the increased trajectory complexity within the MB dataset. We see that our OSB is competitive with GSB and even considerably faster than SGSB. As we may expect from the theoretical analysis, OAB is very slow in comparison to the other algorithms, yet the greedy alternatives are comparatively fast.

The main questions to investigate are thus twofold: (1) Is the speed-bounded model able to achieve reasonable results, compared to the acceleration-bounded model? (2) How do the faster greedy approaches compare in terms of quality with respect to the optimal algorithms. We first investigate the latter question before turning to the former.

Speed-bounded model. We have three algorithms that strictly adhere to the speed-bounded model: OSB, GSB, and SGSB (see left two columns of Figure 8). As OSB computes optimal results, GSB and SGSB cannot result in longer subsequences. For the MB dataset, we observe that GSB and SGSB perform very similarly in terms of the number of outliers detected. For the HR and LA datasets, we see larger differences, especially for GSB. Table 3 shows the ratio between OSB and GSB/SGSB according to different brackets of OSB. These numbers indicate that a vast majority of trajectories has less than 10% outliers, and that in such cases the results are on average not much different. The more outliers are present, the more pronounced the difference between our optimal result and the greedy results becomes.

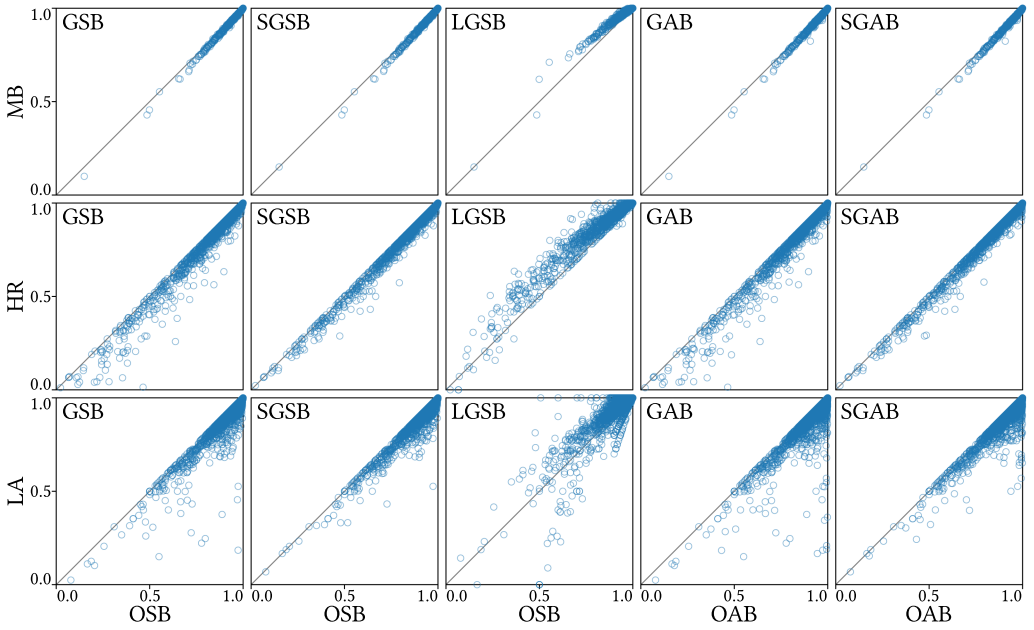

Fig. 8. Comparing the various algorithms. Each axis represents the (relative) length. Top row: MB data; middle row: HR data; bottom row: LA data. First three columns: comparison of OSB with GSB, SGSB, and LGSB; last two columns: comparison of OAB with GAB and SGAB.

Table 3. Mean and Standard Deviation of the Ratio between Greedy Strategies and Optimal Strategies, Split by Bins of the Optimal Length ("Length" Row)

	MB				HR				LA			
Length	**0.0–0.6**	**0.6–0.8**	**0.8–0.9**	**0.9–1.0**	**0.0–0.6**	**0.6–0.8**	**0.8–0.9**	**0.9–1.0**	**0.0–0.6**	**0.6–0.8**	**0.8–0.9**	**0.9–1.0**
Size	0.07%	0.20%	0.57%	99.17%	3.14%	4.58%	5.96%	86.32%	0.33%	2.06%	7.00%	90.61%
GSB	0.87 ± 0.14	0.97 ± 0.01	0.98 ± 0.01	1.00 ± 0.01	0.83 ± 0.20	0.95 ± 0.07	0.98 ± 0.03	1.00 ± 0.01	0.87 ± 0.17	0.93 ± 0.11	0.97 ± 0.05	1.00 ± 0.01
SGSB	0.95 ± 0.06	0.97 ± 0.01	0.98 ± 0.01	1.00 ± 0.01	0.93 ± 0.07	0.97 ± 0.04	0.99 ± 0.02	1.00 ± 0.01	0.94 ± 0.08	0.96 ± 0.05	0.98 ± 0.03	1.00 ± 0.01
LGSB	1.10 ± 0.20	1.08 ± 0.02	1.05 ± 0.01	1.01 ± 0.01	1.22 ± 0.28	1.11 ± 0.07	1.05 ± 0.03	1.00 ± 0.01	1.05 ± 0.48	1.04 ± 0.16	1.05 ± 0.05	1.00 ± 0.01
Size	0.07%	0.21%	0.70%	99.02%	3.14%	4.64%	5.94%	86.32%	0.33%	2.06%	7.33%	90.28%
GAB	0.98 ± 0.06	0.97 ± 0.01	0.98 ± 0.01	1.00 ± 0.01	0.83 ± 0.21	0.95 ± 0.07	0.98 ± 0.03	1.00 ± 0.01	0.87 ± 0.17	0.93 ± 0.11	0.97 ± 0.04	1.00 ± 0.01
SGAB	1.10 ± 0.27	0.97 ± 0.01	0.98 ± 0.01	1.00 ± 0.01	0.93 ± 0.08	0.97 ± 0.04	0.98 ± 0.02	1.00 ± 0.01	0.93 ± 0.09	0.96 ± 0.06	0.98 ± 0.03	1.00 ± 0.01

The "size" row indicates the percentage of trajectories in the corresponding length bin. in GSB, SGSB, and LGSB are compared to OSB; GAB and SGAB to OAB.

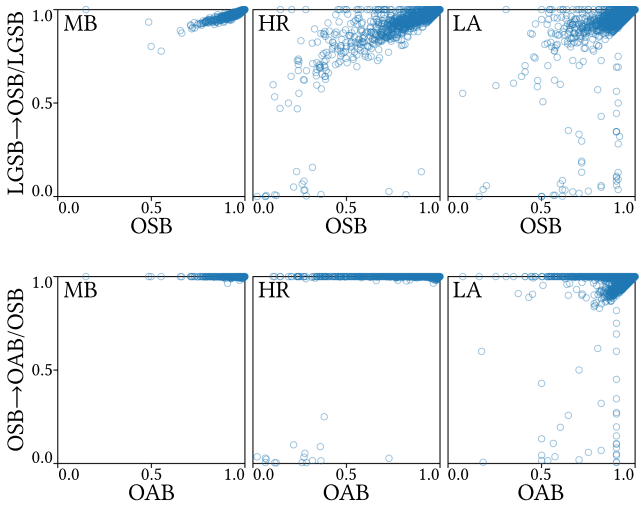

Fig. 9. Postprocessing to ensure a stricter physics model. Left column: MB data; middle column: HR data; right column: LA data. Top row: comparison of LGSB→OSB with OSB; bottom row: comparison of OSB→OAB with OAB.

OSB is thus more reliable, as it gives optimal results. When there are few outliers, this algorithm is close to linear, and thus we may expect less of a performance loss compared to the simpler methods. Indeed, we see that in terms of running time, OSB (0.48 ms on average per trajectory) performs similarly as the GSB (0.24 ms) and is actually faster than SGSB (5.35 ms). When there are many outliers, the extra time spent may be well worth the effort to obtain the maximum consistent subsequence.

Acceleration-bounded model. Referring to Figure 8 and Table 3, we observe the same patterns between OAB and GAB/SGAB as above for the speed-bounded variants, but the differences are more pronounced. However, it must be noted that the computation times behave much differently. Although the number of intervals in a single cell never exceeds 2 for almost all trajectories (with a maximum of 4), the computation time of OAB (224.8 ms on average per trajectory) is significantly higher than GAB (0.41 ms) and SGSB (2.86 ms). Thus, OAB seems practical mostly for cases where processing speed is not a primary concern: for example, because much longer offline computations are expected afterwards, or because the trajectory lengths are limited.

Local strategy. The LGSB method can also be compared to OSB. However, because this method does not ensure that the entire subsequence adheres to the physics model, it may be the case that LGSB yields a longer sequence than OSB. This is quite structurally the case (see third column in Figure 8), with more pronounced effects for a large number of outliers (see Table 3, LGSB row). This indicates that the local strategy for determining outliers is not quite suitable for capturing the actual constraints of the physics model.

We further investigate by postprocessing the results of LGSB by OSB (LGSB→OSB). That is, we find the longest consistent subsequence of the LGSB result. If LGSB would work perfectly, then no outliers are filtered in this postprocessing step. The more outliers are found in the LGSB result, the more violations of the physics model the LGSB result exhibits. The top row of Figure 9 shows the results; note that the vertical axis shows (relative) length of the final result with respect to the length without postprocessing rather than (relative) length with respect to the input. We see again that the results depend on the number of outliers in the trajectory, but overall the difference may

be quite pronounced: LGSB→OSB on average has 8.75% less measurements than LGSB for cases with OSB length less than 0.9. The dataset also has an effect: MB has less variance than the other two datasets.

Comparing models. Since any acceleration-bounded path in our setting is also a speed-bounded path, OSB cannot detect more outliers than OAB. That is, OSB results can be interpreted in the acceleration-bounded model, and we can investigate how well the model inherently meets the acceleration bound. We follow the same approach in comparing LGSB to OSB above, postprocessing OSB results by OAB (OSB→OAB) to determine how many outliers the OSB result still includes according to the stricter model.

The bottom row of Figure 9 shows the results. We clearly see that that only few measurements are filtered in the OAB postprocessing step for all three datasets. This pattern is strongest in MB (0.09% classified as outliers on average) and HR (0.04% on average), even for more noisy trajectories. For the LA dataset, slightly more measurements are filtered (1.74% on average), but interestingly, this seems mostly the case for the less noisy trajectories. These averages are based on the cases with OAB length less than 0.9.

We may conclude that generally the speed-bounded model is capable of getting quite realistic results even for the acceleration-bounded case, while avoiding the computational complexity. It is interesting that there seems to be slightly different behaviors between the two vehicle datasets: this raises the question whether differences in traffic and driving behavior make acceleration more important in certain environments than in others.

5.4 Sensitivity of Model Parameters

The physics models have a few parameters to capture what is considered feasible movement through space and time. Here, we look at how sensitive the results are to changing the parameter values. Following our observations from the previous section, we focus on the speed-bounded model, which effectively has one parameter: the maximum speed v^+, but we also briefly investigate the effect of the detour factor as well as the acceleration bound in OAB.

Procedure. Our analysis for each parameter follows the same procedure: We vary the parameter systematically from very restrictive values to very generous values, running the optimal algorithm for the model under consideration on all data. We then consider how the length of the result varies with this parameter.

To allow for summarizing the results, we operationalize the sensitivity σ for a single trajectory as the maximum of the difference between relative length and the difference between two parameter values, over all (consecutive) pairs of parameter values. We refer to the the two parameters that result in the maximum the *sensitive range* ρ of that trajectory; we use the mean value of the range to compute summary statistics. The unit of σ is thus the inverse of the unit of the parameter, but we generally omit this indication. Intuitively, the sensitivity is the "slope" when plotting the relative length as a function of the parameter, which we refer to as a *profile*. We illustrate these functions for each case using a selection of the trajectories for each dataset, consider summary statistics over all trajectories, and investigate the relation between the sensitivity and the sensitivity range.

Note that our choice of step size in varying the parameter inherently limits the maximum sensitivity that can be obtained to the reciprocal of the step size. For example, steps of 2 km/h in varying the speed bound v^+, limits the sensitivity to 0.5, which would indicate jumping from length 0 to 1 between two values of v^+. In degenerate, constructed inputs this can indeed be realized—in fact, any arbitrarily large sensitivity can achieved in theory. Consider a hypothetical trajectory of n measurements along a straight line, sampled every second, with a distance between consecutive measurements a distance c. The length of the optimal result for any $v^+ < c$ is then

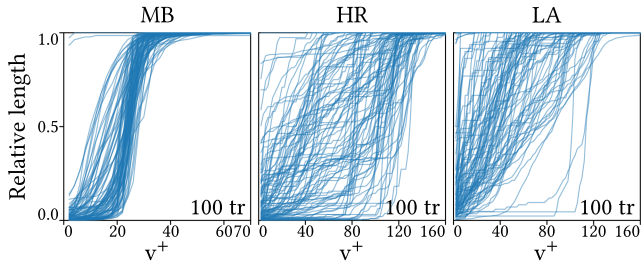

Fig. 10. Profile of the speed bound: length of OSB as a function of v^+, for 100 random trajectories for each dataset.

Table 4. Sensitivity σ of the Speed Bound v^+

dataset	mean	stddev	min	99%	max
MB	0.093	± 0.049	0.013	0.243	0.368
HR	0.059	± 0.042	0	0.244	0.418
LA	0.047	± 0.033	0	0.180	0.458

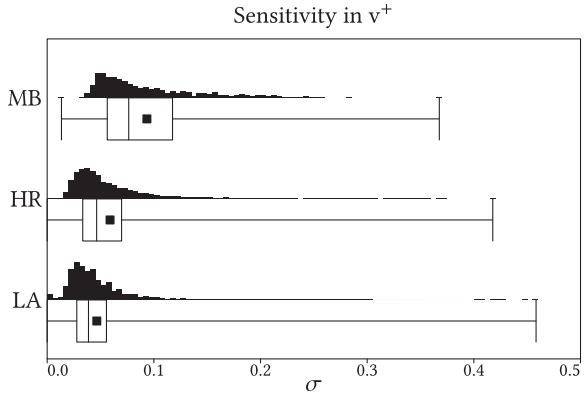

Fig. 11. Sensitivity σ of the speed bound v^+ per dataset.

$1/n$, as no pair is consistent. However, the length for $v^+ \geq c$ is 1. Thus, for $v^+ = c$ and $v^+ = c - \varepsilon$, we obtain a sensitivity of $(1 - 1/n)/\varepsilon$. For ε approaching zero, this thus tends to infinity. Thus, we focus on the practical slope of these curves, using some reasonable sampling of the domain of the parameter.

Speed bound. We run our OSB algorithm, using a speed bound v^+ from 2 to 70 km/h (MB), from 2 to 160 km/h (HR and LA), in steps of 2 km/h. Figure 10 provides a random sample of the resulting profiles. As we can see, many trajectories follow the roughly the same pattern of a few steep increases at different speed bounds. We attribute this to different behavior of the moving entity. For MB, this behavior is fairly consistent, with a high sensitivity around 21.5 km/h (average sensitivity range). For the other data sets, this is less clear, likely reflecting different driving behavior due to local speed limits, which varies between trajectories but also within a single trajectory.

Table 4 and Figure 11 show summary statistics of the sensitivity for the three datasets. We see that the sensitivity can be quite high in extreme cases: changing the parameter by 1 km/h

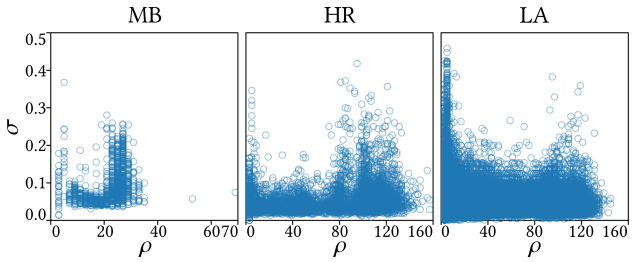

Fig. 12. Scatterplot relating the sensitivity σ of v^+ to (the mean of) the sensitivity range ρ. Each circle represents a single trajectory.

may change the relative length by almost 0.46. On average, the sensitivity is considerably lower. However, these results still show that careful selection of the model parameters is important: too low values result in measurements being identified as outliers unjustly, but setting them too high might leave too many outliers undetected.

With Figure 12, we look at the relation between the sensitivity and the sensitivity range. We roughly see the same pattern for each of the datasets: A number of trajectories have their relatively large sensitivity at low speeds, followed by another peak at higher speeds. Potentially, this separates the trajectories into different cases of actual behavior; for example, cars drive at different speeds in residential areas, provincial roads, and highways—if a trajectory falls mostly within one of the categories, it is reasonable to expect the largest sensitivity to occur at that speed. Under this hypothesis, we see that we used quite a reasonable bounds on the maximum speed, that is, values slightly higher than the sensitivity range for majority of the trajectories.

Acceleration bound. The acceleration-bounded model has, as the name suggests, parameters controlling the allowed acceleration and deceleration, a^+ and a^-, respectively. Due to the high computational cost of OAB, we restrict our attention to only six values combinations of a^+ and a^- per dataset. The parameters we selected for defaults reflect fairly extreme capabilities: limits of racing cars (HR and LA) and estimates of well-trained cyclists (MB). To investigate the effect of these parameters, we thus reduce these parameter values, to reflect settings of "normal" and "slow" behavior in terms of acceleration and deceleration. Specifically, we test the following six combinations for each dataset: $a^+ \in \{2, 4, 10\}$ and $a^- \in \{-2, -10\}$ (HR and LA); $a^+ \in \{0.8, 1.2, 1.62\}$ and $a^- \in \{-2, -3.24\}$ (MB). Each of these values is expressed in m/s^2.

We can now study sensitivity of the one parameter by fixing the other parameter to each of its values. A sample of the resulting profiles are shown in Figure 13 and Figure 14. We immediately see that there is very little effect of the acceleration or deceleration bound. As these are not a random sample, but actually the profiles with highest sensitivity, this tells us that these parameters are of little influence.

The summary statistics over all trajectories further confirm this, as shown in Figure 15. Considering our chosen set of parameters, the sensitivity in a^+ can be at most 0.5 (HR and LA) or 1.67 (MB); for a^- these maxima are 0.125 (HR and LA) and 0.81 (MB). What we observe, however, is that the actual sensitivity is significantly lower—also foregoing the need for further refine the tested parameter values. The strongest sensitivity observed is 0.066 for a^+ and 0.01 for a^-. This is, however, an "extreme" with medians and averages laying much closer to zero.

One observation to be made is that the sensitivity for the maximum acceleration in the MB seems to be slightly higher, though still much lower. This is possibly caused by the nature of the data: a mountain biker may accelerate and decelerate more strongly, compared to regular traffic.

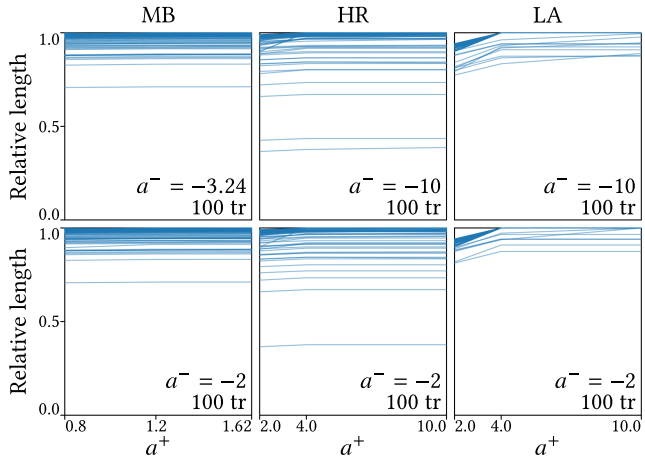

Fig. 13. Profiles of the acceleration bound: length of OAB as a function of a^+, for the 100 most sensitive trajectories for each dataset.

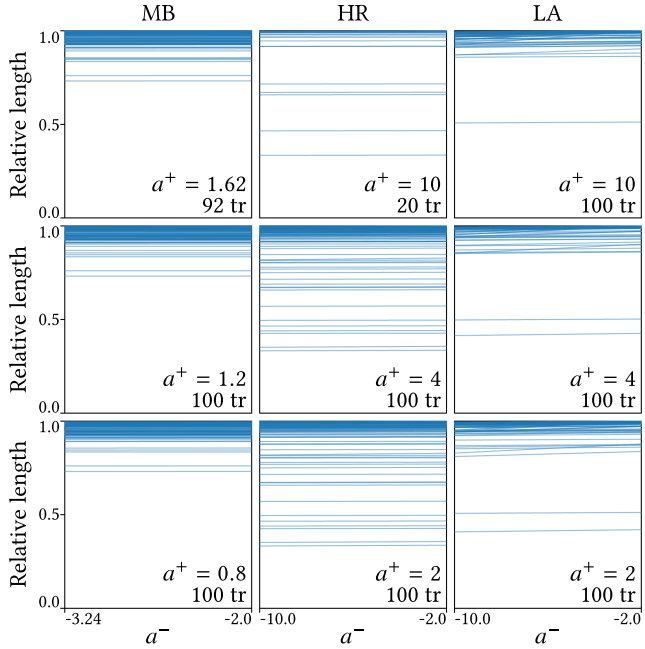

Fig. 14. Profiles of the deceleration bound: length of OAB as a function of a^-, for the 100 most sensitive trajectories for each dataset.

In Figures 16 and 17, we show histograms for the sensitivity, split by the sensitivity range. In these charts, we omit all trajectories that have sensitivity zero—the number of remaining trajectories is indicated per dataset. Notably, we see that MB has relatively few trajectories with zero sensitivity, whereas for the other datasets this is the majority of trajectories. Again, we attribute this to the different nature of mountain biking.

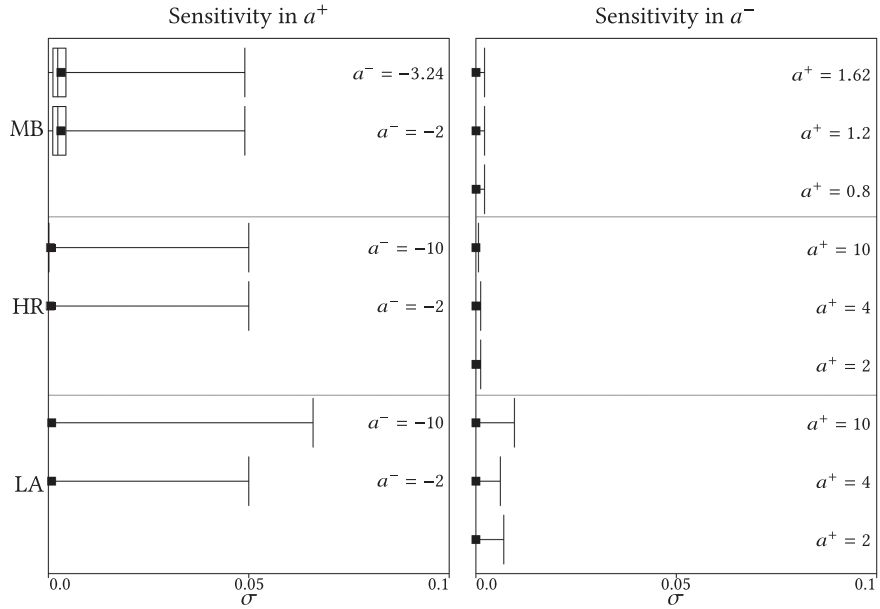

Fig. 15. Sensitivity of a^+ (left) and a^- (right) per dataset and value of the other parameter. Note that the technical maximum sensitivity would be significantly higher, but this does not occur – the horizontal scale has been adjusted.

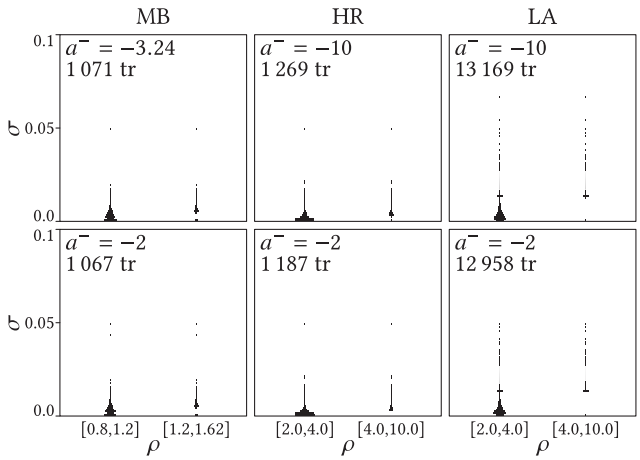

Fig. 16. Histogram relating sensitivity of a^+ and the sensitivity range. Trajectories with sensitivity 0 have been omitted. Note that the technical maximum sensitivity would be significantly higher, but this does not occur—the horizontal scale has been adjusted.

In light of the little dependence on the acceleration bounds, we assume that the speed bounds used by the acceleration model, in terms of sensitivity, behave similarly as for the speed-bounded model. More importantly, these results further support our conclusion from Section 5.3: the speed-bounded model provides realistic results even for the acceleration-bounded model.

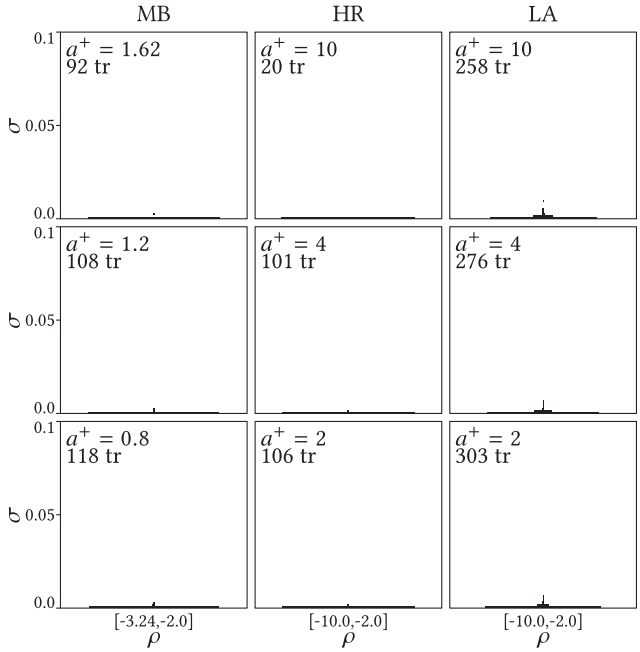

Fig. 17. Histogram of sensitivity of a^-, of the single sensitivity range tested. Trajectories with sensitivity 0 have been omitted. Note that the technical maximum sensitivity would be significantly higher, but this does not occur—the horizontal scale has been adjusted.

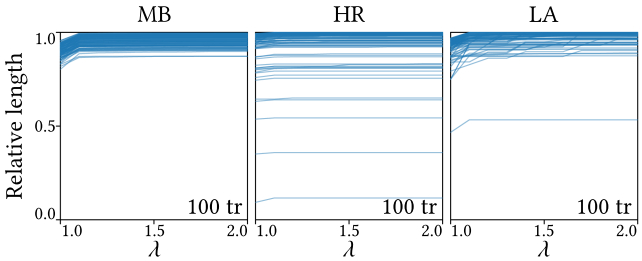

Fig. 18. Profiles of the detour factor: length of OAB as a function of λ, for the 100 most sensitive trajectories for each dataset.

Detour factor. The OAB algorithm uses a detour factor λ, to determine how much distance can at most be traveled between two measurements, which is λ times the Euclidean distance. In other experiments, this is fixed to 1.5, but here we investigate how much this parameter may influence the results. We run the OAB algorithm using λ from 1 to 2, with increments of 0.1.

Refer to Figures 18 and 19. We observe that detour factor λ has very little influence in general, with most trajectories not being influenced by λ at all: 52 of 1,214 for MB, 4,049 of 5,000 for HR, and 74,500 of 78,658 for LA. The detour factor is likely to help in cases where turns are made at relatively high speed: The Euclidean distance might be too short to slow down and reach the next measurement at the right time—but adding some slack gives enough space to travel between two somewhat close points at high speed. Thus, this factor can be expected to be of less influence for trajectories with high sampling frequency or without turns are relatively high speed. This

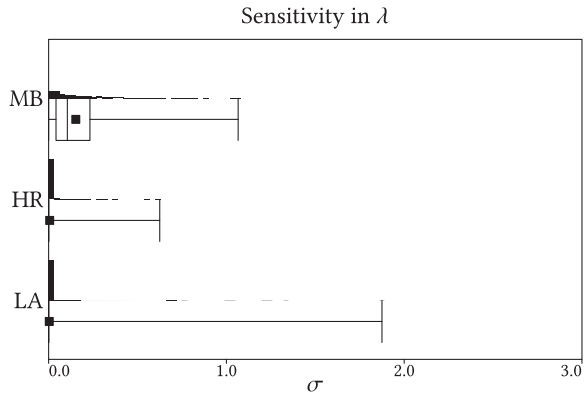

Fig. 19. Sensitivity of λ per dataset. Note that the technical maximum sensitivity would be 10, but this does not occur—the horizontal scale has been adjusted.

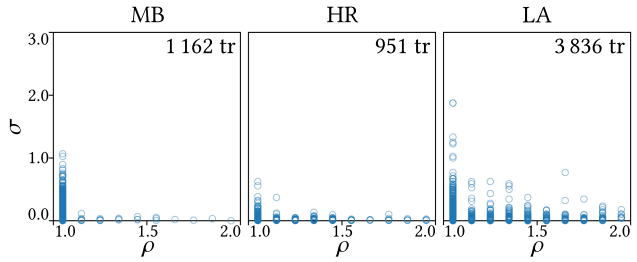

Fig. 20. Scatterplot relating the sensitivity σ of λ to (the mean of) the sensitivity range ρ. Each circle represents a single trajectory. Trajectories with sensitivity 0 have been omitted. Note that the technical maximum sensitivity would be 10, but this does not occur—the vertical scale has been adjusted.

may explain why the mountain-bike dataset exhibits more sensitivity than the other two vehicle datasets.

Figure 20 relates the sensitivity to the sensitivity range. We observe that the highest sensitivity is found in the sensitivity range $[1, 1.1]$, and generally a trend of higher sensitivity at lower values of λ. This supports our suggestion above as to the cause of the low sensitivity.

Most of our trajectories have relatively high sampling frequency and as such the sensitivity is low. The question is how these observations generalize to low sampling frequencies. This will likely depend strongly on the object being tracked. If it travels frequently at nearly maximum speed, then sensitivity may be high. However, if the general speed is significantly lower, then the admitted variation in the reconstructed speed may already be sufficient to avoid sensitivity in the detour factor.

6 DISCUSSION

Results. Our results indicate that our optimal algorithms outperform simple greedy strategies, either in quality of the results, running time, or both. Noise levels and other characteristics do influence these results, and our methods are particularly effective for dealing with large amounts of noise. The example in Figure 21 (top) illustrates a case where the OAB algorithm computes a longer sequence, compared to SGAB: the cause is that a few erroneous measurements lead this greedy algorithm to make a sequence that prevents it from selecting many measurements later.

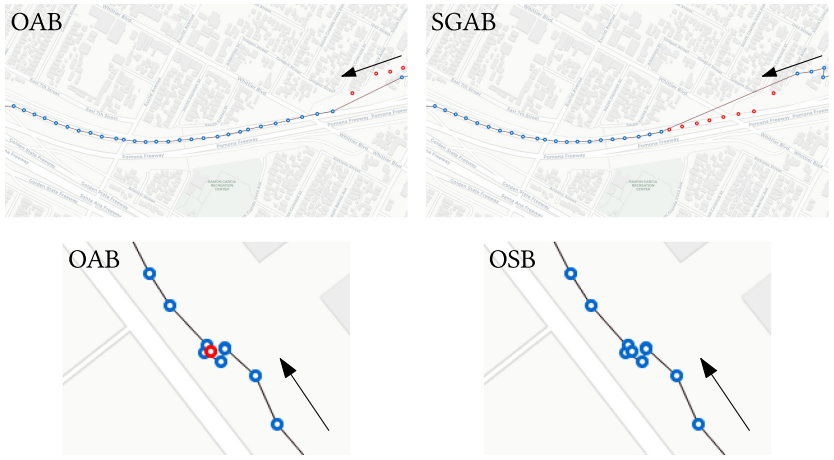

Fig. 21. Example trajectory where OAB differs from SGAB (top) and OSB (bottom). Arrows indicate the direction of travel, blue markers are measurements that are part of the consistent subsequence computed, and red markers indicate the corresponding outliers. Both trajectories are from the LA dataset. Base maps from OpenStreetMap.

Furthermore, the results suggest that the quality difference between speed-bounded models and acceleration-bounded models is small. This must be considered carefully though, as there is an effect of social or geographic environment. Figure 21 (bottom) shows a case where the OSB algorithm detects fewer outliers, though the difference is only minor. Contrasting the previous comparison, this implies that OAB performed better than OSB: OSB fails to capture the outlier that is not physically realizable in the stronger acceleration-bounded model. That is, there is not enough time to realistically decelerate and accelerate to capture the full near-stationary measurements.

The selection of parameters influences the results, but this is mostly the case for the maximum speed. Acceleration and detour factor for our OAB algorithm tend to have minimal effect on the number of outliers detected, though we observed variation between the types of moving entities. Figure 22 shows a sequence of results for different speed bounds, for a trajectory from each dataset. Increasing the speed bound leads to fewer outliers – but possibly less realistic behavior, if the bound is set too high. The effect of lowering the speed bound is that corners tend to be cut by marking outliers, to lower the traveled distance to one that is achievable within the speed bound.

Context. By design, we do not consider the use of other data, such as a road network that a vehicle is driving on. However, such data opens up various potential avenues for further research. For example, given a road network, we may be able to more accurately assess the travel distance or limit it to a few likely candidates, rather than using the Euclidean distance. For OSB and OAB, this is straightforwardly included into the algorithm. For our faster algorithm under the speed-bound model, however, this is not quite the case, as the AWVD is no longer directly applicable, but there may be potential to generalize the approach.

Beyond assessing distances more accurately, additional data could also be used to define more accurate physics models. Our current models are fairly simple, and use only few parameters to define global thresholds on the maximum speed and acceleration. However, such thresholds may actually depend on the environment. For example, expected maximum speed for driving in a car is different on the highway than it is in an urban environment. Similarly, cycling uphill or downhill affects maximum speed. Ideally, physics models and, by extension, outlier-detection algorithms

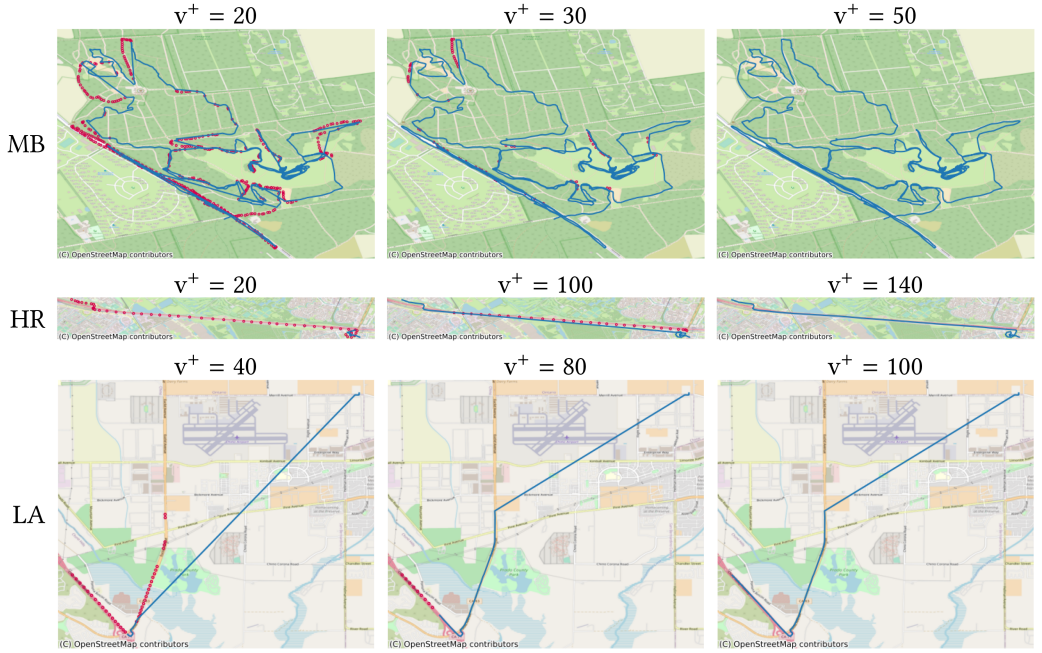

Fig. 22. Example results, using different speed bounds v^+. One trajectory for each dataset is shown. Blue line represents the resulting trajectory, with red dots marking the outliers. Base maps from OpenStreetMap.

should accommodate for such variations, as this allows for more efficacious processing of heterogeneous trajectories that travel through different environments.

Including contextual factors will make the models more accurate and realistic, but a crisp decision boundary (movement is or is not physically possible) may no longer exist. Instead, we may want to define that a car can violate speed limits, but the severity and duration affect how likely the behavior is. Future work could explore "behavioral models" that describe expected movement more closely, including context, and are more robust by allowing deviations from the model, thereby reducing parameter sensitivity.

Enhancing other techniques. There are many other forms of trajectory processing and analysis techniques, such as clustering, map matching, and segmentation. Such techniques may be complemented or enhanced by applying physics models to define possible or realistic behavior. For example, a map-matching algorithm could include considerations of whether its result is physically realizable, or clustering may be done based on what physical behavior would be necessary to realize certain trajectories. We leave exploring such complementarity of techniques to future work, but our results presented here provide a framework and methods that may be integrated into such enhanced techniques.

A DERIVATION OF THE PROPAGATION FUNCTION

For our algorithm under the acceleration-bounded model, we need to determine the minimum and maximum speed for which the moving entity can arrive at a measurement p_j, when starting at a measurement p_i with some given velocity. For our asymptotic analysis, we already argued that this is possible in linear time (Lemma 4.2). Here, we show how to precisely compute this interval, providing the exact formula for propagation and thus proving that this is indeed computable. We do

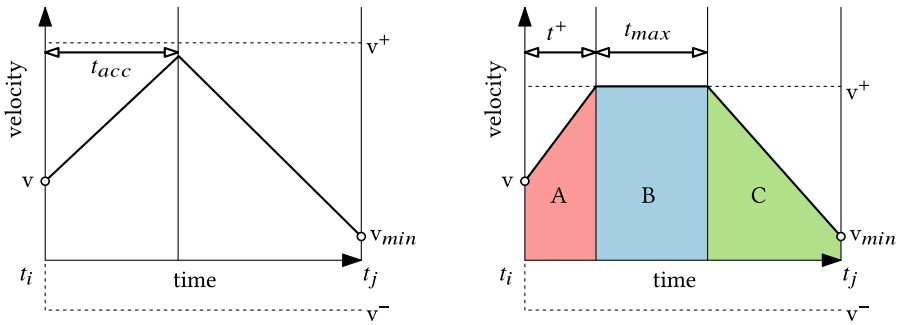

Fig. 23. (Left) Velocity-time diagram for the behavior where one maximally accelerates for a time t_{acc} and then maximally decelerates to obtain the lowest possible speed v_{min}. (Right) Velocity-time diagram for the behavior of accelerating to the maximum velocity v^+ in time t^+, then maintaining this velocity for t_{max} time, and finally maximally decelerating to get the minimum velocity.

so for the minimum velocity; computing the maximum velocity is symmetrical. This minimum velocity $v_{min} \in \mathbb{R}$ is the smallest value such that $C(p_i, p_j \mid v_i = v, v_j = v_{min})$ for two measurements p_i, p_j and some initial $v \in \mathbb{R}$. Note that these velocities may be negative, indicating the direction of movement in the 1D space.

We need particular behaviors of the moving entity to accomplish this minimum velocity. These behaviors are determined by the travel time $\Delta t = t_j - t_i$ and distance $\Delta x = \|p_j - p_i\|$ between p_i and p_j as well as the initial velocity v: We want to travel the distance between p_i and p_j within the given time, while minimizing the velocity at p_j.

Checking consistency. First, we determine whether *any* velocity can be obtained, that is, whether it is physically possible to travel distance Δx in Δt time, starting with velocity v. That is, we must test whether $C(p_i, p_j \mid v_i = v)$ holds. The maximum distance Δx^+ that can be traveled, is obtained by accelerating until reaching the maximum velocity v^+ and then maintaining that speed. Accelerating to maximum speed takes $t^+ = \frac{v^+ - v}{a^+}$ time. If $t^+ < \Delta t$, then the maximum velocity is achieved, and we can express Δx^+ as $(v + v^+)t^+/2 + (\Delta t - t^+)v^+$. Otherwise, this behavior accelerates maximally for the entire duration, in which case $\Delta x^+ = (2v + a^+\Delta t)\Delta t/2$. Analogously, we find an expression for the minimum distance Δx^- that can be traveled. As we can use a convex combination of achieve any traveled distance between these two extremes, we know that $C(p_i, p_j \mid v_i = v)$ if and only if $\Delta x^- \leq \Delta x \leq \Delta x^+$.

Note that the above test indicates consistency, then we can look for a minimal (and maximal) velocity. If this consistency does not hold, then we know that no velocity can be reached at all. For the remainder, we assume that $C(p_i, p_j \mid v_i = v)$ is indeed true.

Finding the minimum velocity. We now strive to find the necessary behavior that results in the minimum velocity, v_{min}, assuming we have already confirmed the basic consistency described above. To visualize this behavior, we look at the velocity-time diagrams for the moving entity; see the examples in Figure 23. The curve in this diagram represents the velocity as a function of time. In this diagram, we need that the total area under the curve is exactly the traveled distance Δx. The speed bounds v^-, v^+ are now represented as allowed minimum and maximum values for the curve. The bounds on the acceleration a^-, a^+ translate to the minimum and maximum slope the curve can have at any time.

We can distinguish a number of situations where we need different behavior to get to the minimum velocity, depending on whether we travel at maximal velocity intermediately. If we can reach

p_j by first maximally accelerating for some time t_{acc} and then maximally decelerating the rest of the time, then this gives us the lowest possible velocity (left diagram in Figure 23). We can, however, encounter the case where the maximum velocity we reach with this behavior exceeds v^+. In this case, we can accomplish the minimum velocity by accelerating to v^+ in time t^+, retaining this speed for some time t_{max} and then maximally decelerating (right diagram in Figure 23) for the remaining time. However, if the result of the appropriate situation above violates the minimum velocity bound v^-, then we can conclude that $\mathrm{v}_{min} = \mathrm{v}^-$. Detailed proofs that the first two behaviors indeed give the minimum velocity are given in Appendices A.1 and A.2.

To compute the minimum velocity for the first and second case, we will use the equation of motion in 1D, given by

$$\Delta x = \mathrm{v}\Delta t + \int_{t_i}^{t_j}\int_{t_i}^{t} a(t')\mathrm{d}t'\mathrm{d}t. \tag{2}$$

This describes the aforementioned requirement that the area under the curve in the velocity-time diagram is the distance Δx between p_i and p_j. To find the minimum velocity for the first two described situations, we fill in the shape for the acceleration $a(t)$ and determine the minimum velocity, given by

$$\mathrm{v}_{min} = \mathrm{v} + \int_{t_i}^{t_j} a(t)\mathrm{d}t. \tag{3}$$

Maximally accelerate, then maximally decelerate. We first consider the situation where we do not reach the velocity bound when maximally accelerating. In this first situation, our acceleration function is equal to

$$a(t) = \begin{cases} a^+, & t_i \le t \le t_i + t_{acc} \\ a^-, & t_i + t_{acc} < t \le t_j \end{cases}. \tag{4}$$

What remains is to determine t_{acc}. We do this by solving the 1D equation of motion (Equation (2)) for the distance Δx between the measurements. We fill in the acceleration function and integrate to get

$$\Delta x = \mathrm{v}\Delta t + \frac{1}{2}a^+ t_{acc}^2 + (\mathrm{v} + a^+ t_{acc})(\Delta t - t_{acc}) + \frac{1}{2}a^-(\Delta t - t_{acc})^2. \tag{5}$$

We can now solve this quadratic equation for t_{acc}. To simplify notation, we use $\Delta a = a^+ - a^-$ and $\bar{\mathrm{v}} = \Delta x/\Delta t$, that is, the average required velocity to travel the distance in the given amount of time. We pick the root of the solution such that the resulting t_{acc} is in $[t_i, t_j]$ and get

$$t_{acc} = \Delta t - \sqrt{\frac{\Delta t}{\Delta a}}\sqrt{a^+\Delta t + 2(\mathrm{v} - \bar{\mathrm{v}})}. \tag{6}$$

With this value, we can now determine the minimum velocity. We fill in Equation (3) and get

$$\begin{aligned} \mathrm{v}_{min} &= \mathrm{v}(t_i + \Delta t) = \mathrm{v} + t_{acc}a^+ + (\Delta t - t_{acc})a^- \\ &= \mathrm{v} + \Delta t a^+ - \sqrt{\Delta a \Delta t}\sqrt{a^+\Delta t + 2(\mathrm{v} - \bar{\mathrm{v}})}. \end{aligned} \tag{7}$$

Note that this situation applies only if we do not exceed the velocity bounds when accelerating and decelerating. So, we require that

$$\mathrm{v} + a^+ t_{acc} \le \mathrm{v}^+, \qquad \mathrm{v}_{min} = \mathrm{v} + t_{acc}a^+ + (\Delta t - t_{acc})a^- \ge \mathrm{v}^-. \tag{8}$$

Accelerating to maximum velocity. We now consider the situation where we reach the speed bound v^+. We accelerate for some t^+ time, until we are moving with velocity v^+, then we retain

that velocity for some time t_{max}, and finally we maximally decelerate to get the minimum velocity. We can describe this behavior with the following acceleration function:

$$a(t) = \begin{cases} a^+, & t_i \leq t \leq t_i + t^+, \\ 0, & t_i + t^+ < t \leq t_i + t^+ + t_{max}, \\ a^-, & t_i + t^+ + t_{max} < t \leq t_j. \end{cases} \tag{9}$$

We now need to determine t^+ and t_{max}. As before, $t^+ = \frac{\mathrm{v}^+ - \mathrm{v}}{a^+}$ indicates the time needed to accelerate from v to v^+.

With the equation for t^+, we can now determine t_{max} by again solving the 1D equation of motion in Equation (2) with our new acceleration function. This gives us the following equation of motion, and solution for t_{max}:

$$\Delta x = \mathrm{v}t^+ + \frac{1}{2}a^+(t^+)^2 + \mathrm{v}^+(\Delta t - t^+) + \frac{1}{2}a^-(\Delta t - t_{max} - t^+)^2, \tag{10}$$

$$t_{max} = \Delta t - t^+ - \sqrt{\frac{3a^+}{a^-}(t^+)^2 + \frac{2\Delta t(\bar{\mathrm{v}} - \mathrm{v}^+)}{a^-}}. \tag{11}$$

Using the above, we now find the minimum velocity, by filling in Equation (3):

$$\mathrm{v}_{min} = \mathrm{v}^+ + (\Delta t - t^+ - t_{max})a^- = \mathrm{v}^+ + \sqrt{\frac{3a^+}{a^-}(t^+)^2 + \frac{2\Delta t(\bar{\mathrm{v}} - \mathrm{v}^+)}{a^-}}a^-. \tag{12}$$

This case is applicable only if $t_{max} > 0$ and the resulting $\mathrm{v}_{min} \geq \mathrm{v}^-$.

Achieving v^-. The extreme behavior of the previous two cases achieve the lowest possible speed, without violating v^+, a^+, or a^-. We can readily choose between the two cases, by comparing t_{acc} with t^+: If the former is at most the latter, then the first case applies; otherwise, the second case applies. However, the result may still violate the physics model, but only v^-. That is, if the computed v_{min} is below v^-. Our claim is that, in such a case, v_{min} is actually equal to v^-. Intuitively, the previous cases in fact achieve a velocity that is too low: We thus have slack to use less extreme behavior intermittently, such as standing still (v = 0) for a certain time.

Consider the behavior of the previous cases. If we follow the behavior but maintain the minimal velocity bound, then we have too much area under the curve: We overshoot our traveled distance. We can compensate for this by accelerating less extremely or maintaining a velocity below v^+. Since some velocity is obtainable, we know that there is sufficient slack and can indeed achieve v^-, if the previous cases would violate the velocity bound of v^-.

A.1 Proof: Achieving Minimum Velocity without Attaining Velocity Bounds

We now show that the behavior of maximally accelerating, followed by maximally decelerating indeed gives the minimum velocity, provided that the velocity bounds $\mathrm{v}^-, \mathrm{v}^+$ are never exceeded during this behavior. Without loss of generality, we further assume that $t_i = 0$ to simplify the exposition, and thus $t_j = \Delta t$.

The equation of 1D motion (Equation (2)) must satisfied for $a(t)$, with additional constraints that $a(t) \in [a^-, a^+]$ for any $t \in [0, \Delta t]$. We then want to minimize the velocity at p_j, as given by Equation (3).

We represent the function $a(t)$ as follows:

$$a(\phi, t) = a^- + (a^+ - a^-)\phi(t) = a^- + \Delta a \psi(t), \tag{13}$$

where $\phi : [0, \Delta t] \to [0, 1]$. This way, the acceleration bounds are trivially satisfied by the function.

Let $a(\psi, t)$ be the function representing maximal acceleration up to some time $t_{acc} \in [0, \Delta t]$ and then maximum deceleration until t_j. In addition, assume that the traveled distance is satisfied by this function. We can represent $\psi(t)$ by

$$\psi(t) = \begin{cases} 1 & t \leq t_{acc} \\ 0 & t > t_{acc} \end{cases}. \tag{14}$$

Let $\psi'(t)$ be a function given by $\psi(t) + \delta\psi(t)$ where $\delta\psi(t)$ is a perturbation on the function, such that $a(\psi', t)$ still travels the required distance, but differs in at least one value t from $a(\psi, t)$. We now show that the velocity at p_j for this perturbed function is always greater than the velocity produced by $\psi(t)$.

By assumption, both travel the same Δx distance in Δt time. Thus, filling in Equation (2) for both gives us the following equality and its simplification:

$$\mathrm{v}\Delta t + \int_0^{\Delta t}\int_0^t (a^- + \Delta a\psi(t'))\mathrm{d}t'\mathrm{d}t = \mathrm{v}\Delta t + \int_0^{\Delta t}\int_0^t (a^- + \Delta a(\psi(t') + \delta\psi(t'))\mathrm{d}t'\mathrm{d}t, \tag{15}$$

$$\int_0^{\Delta t}\int_0^t \delta\psi(t')\mathrm{d}t'\mathrm{d}t = 0. \tag{16}$$

We know that the velocity at p_j is given by Equation (3). We can now look at the difference $\Delta\mathrm{v}$ between this velocity for $\psi'(t)$ and for $\psi(t)$. This difference is given by

$$\Delta\mathrm{v} = \Delta a \int_0^{\Delta t} \delta\psi(t)\mathrm{d}t. \tag{17}$$

Now, if the minimum velocity at p_j for $a(\psi', t)$ is smaller than the minimum velocity for $a(\psi, t)$, then this would imply that $\Delta\mathrm{v}$ is negative for the corresponding $\delta\psi$ function.

By definition, the value of $\delta\psi(t)$ is non-positive for $t < t_{acc}$ and non-negative for $t > t_{acc}$, as otherwise the acceleration bounds would be violated. Equation (16) implies that $\int_0^{t_{acc}} \delta\psi(t)\mathrm{d}t < 0$ and $\int_{t_{acc}}^{\Delta t} \delta\psi(t)\mathrm{d}t > 0$. Equivalently, the integral $\int_0^t \delta\psi(t)\mathrm{d}t$ is non-increasing for the interval $[0, t_{acc}]$ and non-decreasing for the interval $[t_{acc}, \Delta t]$.

Assume for a contradiction that $\Delta\mathrm{v}$ is negative for some $\delta\psi$, that is, $\int_0^{\Delta t} \delta\psi(t)\mathrm{d}t < 0$. This automatically implies that $\int_0^t \delta\psi(t')\mathrm{d}t' \leq 0$ for any t in the interval, due to the non-decreasing and non-increasing properties of the integration interval. But then the integral of Equation (16) is by definition negative, which means that $a(\psi', t)$ does not travel Δx distance. Hence, we must have that $\Delta\mathrm{v} \geq 0$. Observe that $\Delta\mathrm{v} = 0$ only if $\delta\psi(t)$ is zero.

To prove that maximally decelerating and then accelerating results in the maximum velocity follows a similar argumentation.

A.2 Proof: Achieving Minimum Velocity When Attaining Velocity Bounds

We now prove that if we maximally accelerate to the velocity bound v^+ in time t^+, retain this speed for time t_{max}, and then maximally decelerate, this indeed gives us the lowest possible velocity, if the previous situation does not apply—that is, $t^+ \leq t_{acc}$, and we never reach the velocity lower bound v^-. Without loss of generality, we assume that $t_i = 0$ and $t_j = \Delta t$.

We follow the argumentation as described in the previous section. We again describe the acceleration behavior using $a(\psi, t)$ for a to be defined function $\psi(t)$. We assume that $a(\psi, t)$ travels the required distance given the initial velocity. But now, the behavior of this case yields a slightly

different function for ψ, do describe $a(\psi, t)$:

$$\psi(t) = \begin{cases} 1, & t \leq t^+, \\ \alpha, & t^+ < t \leq t^+ + t_{max}, \\ 0, & t > t^+ + t_{max}. \end{cases} \tag{18}$$

Here, $\alpha = -\frac{a^-}{\Delta a}$ indicates an acceleration of zero. For brevity, we call the three time regions with the different behaviors A, B, and C, see Figure 23.

We again perturb $\psi(t)$ to get a function $\psi'(t) = \psi(t) + \delta\psi(t)$. Here, we again assume that $\delta\psi(t)$ is not the zero function. To obey the acceleration bounds, we observe that $\delta\psi(t)$ is non-positive in region A and non-negative in region C, which implies that $\int_0^t \delta\psi(t)\mathrm{d}t$ is non-increasing in region A and non-decreasing in region C.

For region B, we observe that $\psi(t)$ describes the behavior that results in the highest possible velocity in that region. So, the velocity at a time t in region B that results from $\psi'(t)$ can be at most the velocity obtained via $\psi(t)$. The velocity at any time t is given by

$$\mathrm{v}(\phi, t) = \mathrm{v} + ta^- + \Delta a \int_0^t \phi(t)\mathrm{d}t \tag{19}$$

for acceleration function $a(\phi, t)$. Thus, we can now formalize the observation as $\mathrm{v}(\psi', t) \leq \mathrm{v}(\psi, t)$ for all $t \in B$. Using the definition of $\mathrm{v}(\phi, t)$ and simplifying, we obtain that

$$\int_0^t \delta\psi(t) \leq 0 \tag{20}$$

should hold for all $t \in B$. Since we want that ψ and ψ' travel the same distance Δx in Δt time, we again get the identity

$$\int_{t_i}^{t_j} \int_{t_i}^{t} \delta\psi(t)\mathrm{d}t'\mathrm{d}t = 0, \tag{21}$$

as was shown in Appendix A.1. Similar to the argumentation in Appendix A.1, we look at the difference in minimum speed $\Delta\mathrm{v}$ at t_j:

$$\Delta\mathrm{v} = \Delta a \int_{t_i}^{t_j} \delta\psi(t)\mathrm{d}t. \tag{22}$$

Again, $\psi'(t)$ has a lower minimum velocity if $\Delta\mathrm{v}$ is negative. For this to happen, $\int_{t_i}^{t_j} \delta\psi(t)\mathrm{d}t$ has to be negative.

Now, assume for a contradiction that for some $\delta\psi$, $\Delta\mathrm{v}$ is less than zero, such that the minimum velocity using $a(\psi', t)$ is less than that from $a(\psi, t)$. From Equation (20), we see that $\int_0^t \delta\psi(t)\mathrm{d}t$ is non-positive for all t in regions A and B. In particular, at the end of region B, the integral is non-positive. We distinguish two cases.

The integral is zero. Suppose the integral $\int_0^t \delta\psi(t)\mathrm{d}t$ is zero at the end of region B. Then, since the integral is non-decreasing in region C as established before, we cannot have that $\int_0^{\Delta t} \delta\psi(t)\mathrm{d}t$ is less than zero: $\Delta\mathrm{v}$ is non-negative, which gives a contradiction.

The integral is negative. Suppose now that $\int_0^t \delta\psi(t)\mathrm{d}t$ is negative at the end of region B. If we want $\Delta\mathrm{v}$ to be negative, then this requires that the $\int_0^{\Delta t} \delta\psi(t)\mathrm{d}t$ is negative. Since the integral is non-decreasing in region C, we must have that the integral is negative everywhere in region C to accomplish this. But then the traveled distance for $\psi(t)$ and $\psi'(t)$ is not the same, since Equation (21) is less than zero. This again gives a contradiction.

From the previous argumentation, we can conclude that for any choice of $\psi'(t)$ that satisfies the traveled distance requirement, the minimum velocity at p_j is at least the minimum velocity obtained by using $\psi(t)$.

ACKNOWLEDGMENTS

The authors thank Kevin Verbeek for fruitful discussions on the topic of this article and HERE Technologies for providing the HR and LA datasets. This research was initiated at the 4th Workshop on Applied Geometric Algorithms (AGA 2018) in Langbroek, The Netherlands.

REFERENCES

[1] Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. 2003. Matching planar maps. *J. Algor.* 49, 2 (2003), 262–283.
[2] Jon Louis Bentley and James B. Saxe. 1980. Decomposable searching problems I. Static-to-dynamic transformation. *J. Algor.* 1, 4 (1980), 301–358.
[3] Lasse Bergroth, Harri Hakonen, and Timo Raita. 2000. A survey of longest common subsequence algorithms. In *Proceedings of the 7th International Symposium on String Processing and Information Retrieval*. 39–48.
[4] Norbert Blum and Kurt Mehlhorn. 1978. On the average number of rebalancing operations in weight-balanced trees. *Theoret. Comput. Sci.* 11 (1978), 303–320.
[5] Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi. 1986. Optimal point location in a monotone subdivision. *SIAM J. Comput.* 15, 2 (1986), 317–340.
[6] Steven Fortune. 1987. A sweepline algorithm for Voronoi diagrams. *Algorithmica* 2, 1–4 (1987), 153.
[7] Yong Ge, Hui Xiong, Zhi-hua Zhou, Hasan Ozdemir, Jannite Yu, and Kuo Chu Lee. 2010. Top-eye: Top-k evolving trajectory outlier detection. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. 1733–1736.
[8] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. 2014. Outlier detection for temporal data: A survey. *IEEE Trans. Knowl. Data Eng.* 26, 9 (2014), 2250–2267.
[9] Victoria Hodge and Jim Austin. 2004. A survey of outlier detection methodologies. *Artific. Intell. Rev.* 22, 2 (2004), 85–126.
[10] Hiroshi Imai and Masao Iri. 1988. Polygonal approximations of a curve—Formulations and algorithms. *Comput. Morphol.* 6 (1988), 71–86.
[11] Jae-Gil Lee, Jiawei Han, and Xiaolei Li. 2008. Trajectory outlier detection: A partition-and-detect framework. In *Proceedings of the 24th International Conference on Data Engineering*. 140–149.
[12] Wang-Chien Lee and John Krumm. 2011. Trajectory preprocessing. In *Computing with Spatial Trajectories*, Y. Zheng and X. Zhou (Eds.). Springer, 3–33.
[13] Xiaolei Li, Jiawei Han, Sangkyum Kim, and Hector Gonzalez. 2007. Roam: Rule- and motif-based anomaly detection in massive moving object data sets. In *Proceedings of the SIAM International Conference on Data Mining*. 273–284.
[14] Marcello Montanino and Vincenzo Punzo. 2015. Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns. *Transport. Res. B: Methodol.* 80 (2015), 82–106.
[15] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *Proceedings of the 17th ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL'09)*. 336–343.
[16] Jürg Nievergelt and Edward M. Reingold. 1973. Binary search trees of bounded balance. *SIAM J. Comput.* 2, 1 (1973), 33–43.
[17] Mark H. Overmars and Jan van Leeuwen. 1981. Worst-case optimal insertion and deletion methods for decomposable searching problems. *Inform. Process. Lett.* 12, 4 (1981), 168–173.
[18] Vincenzo Punzo, Maria Teresa Borzacchiello, and Biagio Ciuffo. 2011. On the assessment of vehicle trajectory data accuracy and application to the Next Generation SIMulation (NGSIM) program data. *Transport. Res. C: Emerg. Technol.* 19, 6 (2011), 1243–1262.
[19] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. 2002. Discovering similar multidimensional trajectories. In *Proceddings of the 18th International Conference on Data Engineering*. 673–684.
[20] Guan Yuan, Shixiong Xia, Lei Zhang, Yong Zhou, and Cheng Ji. 2011. Trajectory outlier detection algorithm based on structural features. *J. Comput. Info. Syst.* 7, 11 (2011), 4137–4144.
[21] Yu Zheng. 2015. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.* 6, 3 (2015), 29.

Received August 2020; accepted February 2021