Maximum Physically Consistent Trajectories

Bram Custers
b.a.custers@tue.nl
TU Eindhoven, the Netherlands

Mees van de Kerkhof
m.a.vandekerkhof@uu.nl
Utrecht University, the Netherlands

Wouter Meulemans
w.meulemans@tue.nl
TU Eindhoven, the Netherlands

Bettina Speckmann
b.speckmann@tue.nl
TU Eindhoven, the Netherlands

Frank Staals
f.staals@uu.nl
Utrecht University, the Netherlands

ABSTRACT

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 which 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**.

KEYWORDS

outlier detection, algorithms, physics models, experiments

ACM Reference Format:
Bram Custers, Mees van de Kerkhof, Wouter Meulemans, Bettina Speckmann, and Frank Staals. 2019. Maximum Physically Consistent Trajectories. In *27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '19), November 5–8, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3347146.3359363

SIGSPATIAL '19, November 5–8, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6909-1/19/11…$15.00
https://doi.org/10.1145/3347146.3359363

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 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, 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 paper 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 travelled between the two measured locations in the time between the two measurements. In this paper 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 $\pi\colon [t, t'] \to \mathbb{R}^d$ 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 $[v^-, v^+]$ 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 $\pi\colon [t_1, t_n] \to \mathbb{R}^d$ 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 *concatable* 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 Fig. 1 (left): both $T = \langle p_1, p_2 \rangle$ and $U = \langle p_2, p_3 \rangle$ are 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 (see Fig. 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 which 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. 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. We conclude with a discussion of our results in Section 6.

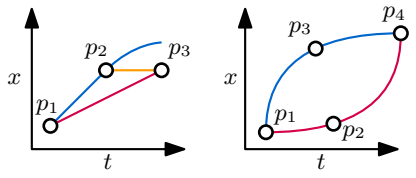

Figure 1: (Left) In an acceleration-bounded model $\langle p_1, p_2, p_3 \rangle$ is not consistent, even though $\langle p_1, p_2 \rangle$ and $\langle p_2, p_3 \rangle$ (and even $\langle p_1, p_3 \rangle$) are. (Right) A consistent subtrajectory through p_2 (red) may require a different speed at p_4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

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 [8]; see also [7] 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 [6, 10, 11, 17], 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 [18] for a survey. However, 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, 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 [12, 15]. 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 (LCSS) and map matching. The former asks to compute the maximum subsequence of two strings [3] and has also been used to compute trajectory similarity [16]. Contrasting our approach, LCSS 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. [1, 13]; see [18] 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 methododology 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 [9]. 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 $C(p_i, p_j)$. 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 sketch. We maintain a linked list of the already processed measurements, sorted by the length of the consistent subsequence ending at that measurement. For a new measurement p, we traverse this list to find the first measurement q consistent with p, that is, with $C(q, p)$. We then insert p with T_p having length $1 + T_q$. Each of the $n - k$ measurements in the longest result traverses only $O(k)$ measurements in the list, while each of the k outliers traverses $O(n)$ elements. As all other operations take $O(f(n))$ time, the algorithm takes $O(nkf(n))$ time. □

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 if 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, i.e. $p_i = (x_i, y_i, t_i)$. 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 $t \geq t_i$. 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 $t_i \geq \hat{t}$ is consistent with any measurement of P we use an *additively weighted Voronoi diagram* (AWVD). Given a set of circles 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 circles, such that for any point $v \in c_i : d(v, v_i) - r_i \leq d(v, v_j) - r_j$, for all $v_j \neq v_i$. Here, d is a distance measure (in our case the Euclidean distance), and the 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 $r_i = v^+(t' - t_i)$ 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'. 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 $q = p_i$, giving us a previous measurement p_c and a distance s_c between the circles (p_i, r_i) and (p_c, r_c), given by $s_c = d(p_i, p_c) - r_i - r_c$. 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 $q = p_i$ be a query measurement. If $s_c \leq -2r_i$ for the resulting distance s_c of p_c with p_i on $\mathcal{D}$, p_c is consistent with q. Otherwise, no measurement in P is consistent with q.*

Proof sketch. If the inequality holds, it immediately follows that the reachable region of p_i is inside that of p_c, so p_i and p_c are consistent. Otherwise, p_c is not consistent, and the AWVD definition tells us that all other distances are larger, so no other measurement can be consistent. □

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 [4, 5]. 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 Fig. 2). 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. 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}.

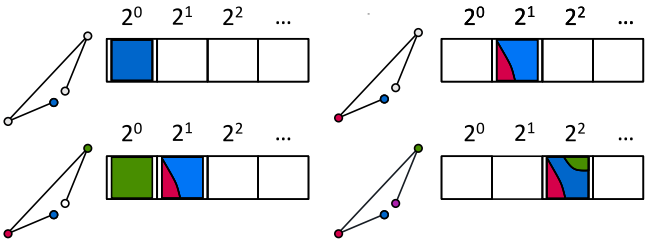

Figure 2: Inserting elements using Bentley-Saxe. The colored measurements in the trajectory are the elements in the insertion only consistency data structure.

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 [14]. 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 $q = p_i$ 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 $p \in P_r$, occuring in its right subtree, and a consistency data structure $\mathcal{D}_v$ built on the set P_r (see Fig. 3).

Given a query measurement q, we find a measurement $u_q \in P$ 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_q is the predecessor of q in T_q. To find u_q 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, we repeat the process in the right child. If not, we move to the left child. This way we get the longest-path measurement that is consistent with our query measurement q. Since querying an associated consistency data structure $\mathcal{D}_v$ takes $O(\log^2 m)$ time, and we do $O(\log m)$ such queries, the total query time is $O(\log^3 m)$.

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. When a subtree rooted at a node v becomes unbalanced, we rebuild it and its associated data structures from scratch. The amortized cost of these rebalancing operations can be shown to be $O(\log^3 m)$.

Lemma 3.3. *There is a data structure $\mathcal{T}$ that can store a subset P of m measurements from T and that given a query measurement q at time $t \geq \hat{t}$ can find (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.*

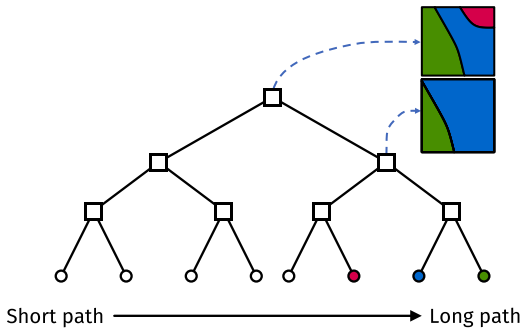

Figure 3: Data structure for maximum subsequence queries.

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. *If $C(p_i, p_j)$ for $j > i$, then the reachable region for p_j for all $t > t_j$ is always contained in the reachable region of p_i.*

Proof sketch. Measurement p_j is consistent with measurement p_i if its reachable region is strictly contained inside the reachable region of p_i at time t_j. For time beyond t_j, the regions grow with the same speed. Thus, the smallest distance between the regions remains constant, which implies that the reachable region of p_j remains inside the region of p_i for higher t. □

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 $\min(k, m)$ cells. Hence, a static consistency data structure uses only $O(\min(k, m))$ 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 test if p_j is consistent with some earlier measure p_i. If so, 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 the velocity and acceleration at any time are restricted. We require the velocity to be in the range $[\mathrm{v}^-, \mathrm{v}^+]$ for some chosen constants $\mathrm{v}^-, \mathrm{v}^+$. In addition, we require the acceleration to be in the range $[a^-, a^+]$ where again a^-, a^+ are chosen constants. For simplicity, we will assume $a^- < 0$.

For this physics model, we can still test if two points p_i and p_j are consistent in constant time: we can check if the distance between the measurements can be travelled by a velocity that lies in the velocity range $[\mathrm{v}^-, \mathrm{v}^+]$. Then, if there exists a velocity at p_i such that the required velocities can be reached by accelerating, the pair is consistent. However, when considering more than two points, this approach is not applicable as 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. This implies that consistent sequences are not concatenable, and hence the previous algorithms are not applicable. Instead, we develop a new algorithm that explicitly computes the speeds achievable at every measurement, and uses this to find a maximum-length consistent subtrajectory.

4.1 Computing speed intervals

The following observation provides the basis for our algorithm.

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

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 $S = \langle \ldots, p_i \rangle$ of length ℓ exists with $C(S \mid \mathrm{v}_i = v)$. Let ℓ^* be the maximum value for which there is a measurement p_i for which $\mathcal{I}(\ell, i)$ is non-empty. It follows that the maximum consistent subtrajectory of T has length ℓ^*.

Given the set of possible speeds $\mathcal{I}(\ell, h)$ 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 $\mathcal{I}(\ell, i)$.

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

Moreover, we prove 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.

Lemma 4.1. *Let p_i, 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 $\pi_1(t), \pi_2(t)$ 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.

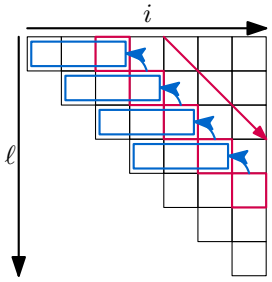

Figure 4: The order for computing $\mathcal{I}(\ell, i)$ in our dynamic program.

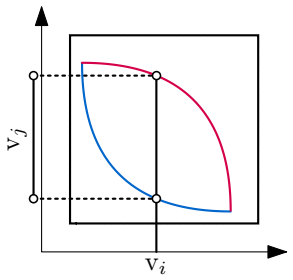

Figure 5: Speed v_i maps to a speed interval at v_j; assumes minimum or maximum speed is not reached.

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

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

If we take a convex combination $v = \beta v_1 + (1-\beta)v_2$ for $\beta \in [0, 1]$ and similarly $a(t) = \beta a_1(t) + (1-\beta)a_2(t)$ to create a new path, we see that we get the exact same travelled distance by linearity of the integrals. Since we took a convex combination, we know that the path still satisfies the velocity and acceleration constraints, since the velocity and acceleration for the new path will always lie between the original velocities and accelerations at any time t in $[t_i, t_j]$. Hence, the new path is a witness that implies $C(p_i, p_j \mid \mathrm{v}_j = v_1 + (1-\beta)v_2)$ for any $\beta \in [0, 1]$ – this concludes the proof. □

Hence, the sets $\mathcal{I}(\ell, i)$ actually form sets of intervals. With some slight abuse of notation, we treat them as such in the remainder of the paper. Let $\delta(\ell, i)$ 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. Next, we establish how to propagate a single speed interval from p_i to p_j.

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

Proof Sketch. Let v and v' be the endpoints of I. We can express the minimum and maximum speed achievable at p_j, assuming that the speed at p_i is v or v'. This gives us the minimum and maximum speed at p_j. □

Using the above recurrence, we can then 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 Fig. 4). 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 (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. *Given a 1D trajectory T with n measurements. Under the velocity and 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 with $O(nk\delta_{\max})$ space, with k the number of outliers and $\delta_{\max}$ the maximum fragmentation.*

4.2 A bound on the size of $\mathcal{I}(l,i)$

The running time of the dynamic program sketched in the previous section depends on the maximum fragmentation, i.e., the maximum number of intervals in a set $\mathcal{I}(\ell,i)$. We observe that $\mathcal{I}(\ell,i)$ may contain more than one velocity interval, see Fig. 1 (right). As we show in the following lemma, the fragmentation may even be $\Omega(n)$.

Lemma 4.4. *There is a one-dimensional trajectory T with n vertices in which $\Omega(n)$ vertices have fragmentation $\Omega(n)$.*

Proof. We construct a trajectory $T = \langle p_1,\ldots,p_{n/2},c_1,\ldots,c_{n/2}\rangle$ such that for parameters $v^- = 0$ and $a = a^+ = -a^- = 1$, we get $\Omega(n)$ speed intervals at each point c_j, $j \in 1,\ldots,n/2$.

Let $\Delta > 0$ be some real number. We place the points at $p_i = (-4i\cdot\Delta^2, 0)$, for $i = 1,\ldots,n$, and $c_j = (0,\Delta)$, for $j = 1,\ldots,n$ (see Fig. 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 $c = c_1$ generates a consistent subtrajectory $\langle p_i, c\rangle$ for which the possible speeds at c 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 c_j are arbitrarily close to c, the same argument shows that we get $\Omega(n)$ speed intervals at those points.

Since $a = 1$ and the time between p_i and c is short, the speed that the entity has at p_i must be similar to its speed at c. If the speed at p_i differs too much from the speed at c, then the entity cannot actually reach c: 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 c and arrives at c 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 c is exactly $4j\Delta^2$. The entity covers this in Δ time, and hence its average speed must be

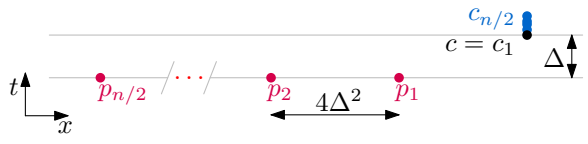

Figure 6: Instance with $\Omega(n)$ disjoint speed intervals at c.

$4j\Delta$. 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 c. So, we have $4j\Delta - \Delta \le v \le v_i + \Delta$. Using that $v_i = 4i\Delta$ we get $j \le i + \frac{1}{2}$. As i and j are natural numbers, we get $j \le i$. Symmetrically, we have $v_i - \Delta \le v \le 4j\Delta + \Delta$, and get $i \le j$. 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 can achieve all speeds v_i at p_i. □

An upper bound on the fragmentation $\delta(\ell,i)$ is $O(2^i)$, since any fixed subsequence of $\langle\ldots,p_i\rangle$ yields only a single interval (refer to Lemma 4.1). We expect that the fragmentation is much smaller in practice and examine how many disjoint intervals we can get in the propagation function (see Fig. 5). The exponential behavior follows from the many very small intervals that can be present at the extremes of the function. Hence, we introduce some slack in the parameters a^-, a^+. We then merge intervals whenever the corresponding intervals in the slacked model overlap.

Let $\Delta a = a^+ - a^-$. We add $\frac{\varepsilon}{2}\Delta a$ to a^+ and $-\frac{\varepsilon}{2}\Delta a$ to a^-. We can give a rough upper bound on the number of intervals a measurement can contribute to the intervals of a later measurement by determining how large the smallest possible input interval can be and dividing the entire possible input range by this size. First, we compute the input velocities that correspond to the smallest possible output range. This range is the input range corresponding to the output range at the extremes of the input range under the slacked model. Then we divide the entire input range by this range. As at most $O(n)$ interval sets are propagated, we now have $\delta_{\max}(\varepsilon) = O(n\varepsilon^{-1/4})$.

4.3 Retrieving the 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.

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.

4.4 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, 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 which 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. 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 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++.

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

[OAB] Optimal Acceleration-Bounded. This algorithm implements the method of Section 4. We use the 2D generalization with an upper bound on the travelled distance of 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 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 must run in $O(n)$ time.

[SGSB/SGAB] Smart Greedy Speed/Acceleration-Bounded. We keep track of multiple greedy 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 [18] points to the only other method that uses a bound on the speed 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 [18]. 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 σ. We use $\sigma = 3$ in our experiments. Note that this local heuristic does not guarantee that the complete output is consistent according to the speed bound.

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. All trajectories in the data sets have at least 10 measurements.

[MB] Mountain-bike trips. This dataset consists of 1 214 trajectories of mountain-bike trips in several European countries from 2012 to 2019. On average, the trajectories have 3377.1 measurements (standard deviation 2643.4) and average speed 18.8 km/h, (standard deviation 10.9 km/h). We set $v^- = 0$ km/h, $v^+ = 35$ km/h. For the acceleration-bounded model, we additionally use $a^- = -3.24$ m/s^2, $a^+ = 1.62$ m/s^2.

[HR] The Hague-Rotterdam. This dataset provided by HERE[1] 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. On average, the trajectories have 424.9 measurements (standard deviation 545.6) and average speed 62.3 km/h, (standard deviation 43.0 km/h). We set $v^- = 0$ km/h, $v^+ = 125$ km/h. For the acceleration-bounded model, we additionally use $a^+ = -a^- = 10$ m/s^2.

[LA] Los Angeles. This dataset provided by HERE[1] 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. On average, the trajectories have 304.4 measurements (standard deviation 1082.0) and average speed 55.8km/h (standard deviation 1557.65 km/h). For the LA dataset, we set $v^- = 0$ km/h, $v^+ = 129.6$ km/h. For the acceleration-bounded model, we additionally use $a^+ = -a^- = 10$ m/s^2.

[1] http://www.here.com/

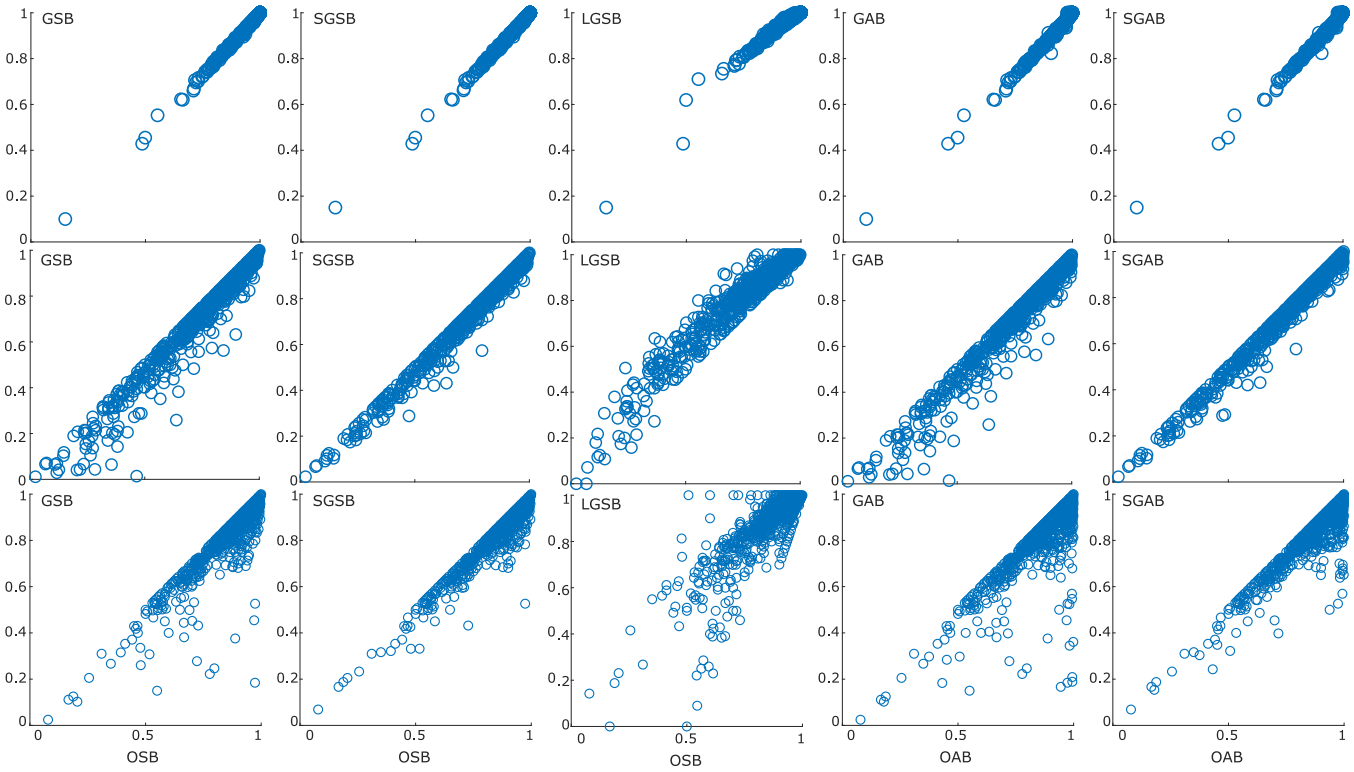

Figure 7: Comparing the various algorithms. Each axis represents the (relative) length. Top row: MB data; middle row: HR data; bottom row: LA data. First three columns: comparison of OSB with GSB, SGSB and LGSB; last two columns: comparison of OAB with GAB and SGAB.

Table 1: Mean and standard deviation of the ratio between greedy strategies and optimal strategies, split by bins of the optimal length ("length" row). 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.

	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

5.3 Results

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 $n - k$ 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.

Speed-bounded model. We have three algorithms that strictly adhere to the speed-bounded model: OSB, GSB and SGSB (see Fig. 7 (left two columns)). 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 1 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.

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 (0.50 ms on average per trajectory) performs similarly as the GSB (0.25 ms) and is actually faster than SGSB (3.17 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. Refering to Fig. 7 and Table 1, 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 (232.82 ms on average per trajectory) is significantly higher than GAB (0.40 ms) and SGSB (5.54 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 Fig. 7), with more pronounced effects for a large number of outliers (see Table 1, 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, 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 left column of Fig. 8 shows the results; note that the vertical axis shows not (relative) length with respect to the input, but with respect to the result without postprocessing. 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.

The right column of Fig. 8 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

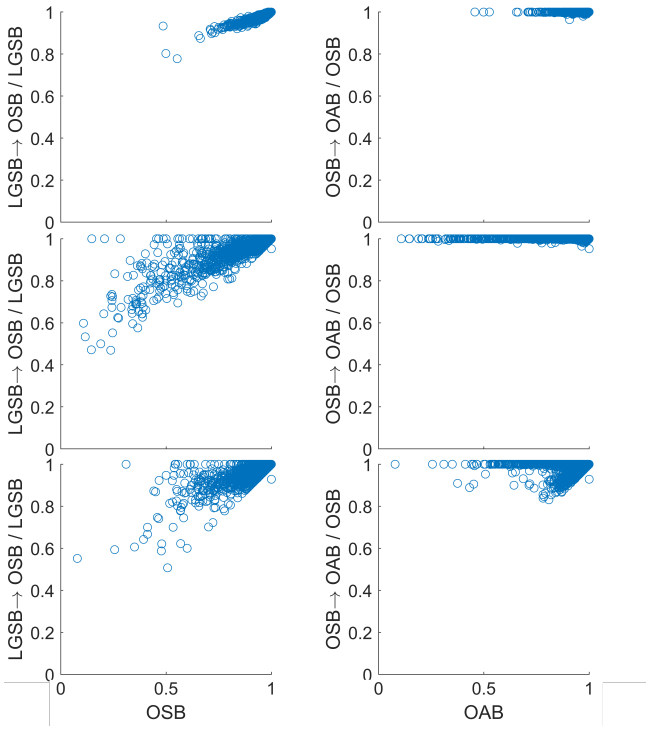

Figure 8: Postprocessing to ensure a stricter physics model. Top row: MB data; middle row: HR data; bottom row: LA data. First column: comparison of LGSB→OSB with OSB; second column: comparison of OSB→OAB with OAB.

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.

Parameter sensitivity. The physics models have a few parameters to capture what is realistically possible 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.

We run our OSB algorithm on all data again, but now increment the speed bound from 0 km/h to 70 km/h (MB), from 0 km/h to 160 km/h (HR) and from 0 km/h to 160 km/h (LA), in steps of 5 km/h. We operationalize sensitivity as the largest change in the resulting length between two consecutive steps (its unit is thus relative length per km/h, or h/km). That is, if we plot the length of the result as a function of the speed bound, this represents the steepest slope. Such a plot is illustrated in Fig. 9.

Table 2 shows summary statistics for the three datasets on this sensitivity. We see that the effect of the parameter may be quite

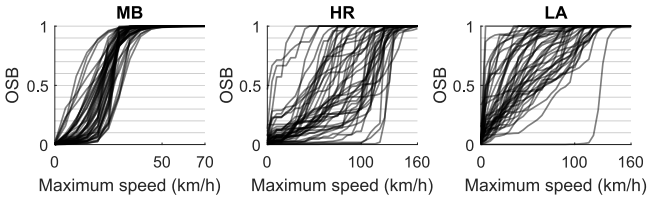

Figure 9: Length of OSB as a function of maximum speed, for 50 random trajectories for each dataset.

Table 2: Parameter sensitivity (relative length per km/h).

dataset	mean	stddev	min	max
MB	0.076	± 0.032	0.032	0.200
HR	0.047	± 0.029	0	0.200
LA	0.053	± 0.038	0	0.200

pronounced in extreme cases: changing the parameter by 1 km/h may change the number of outliers by almost 20%. On average, the sensitivity is approximately only a third of that. 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.

6 DISCUSSION

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 results also suggest that the quality difference between speed-bounded models and acceleration-bounded models are small. This must be considered carefully though, as there is an effect of social or geographic environment.

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 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.

ACKNOWLEDGMENTS

The authors would like to thank Kevin Verbeek for fruitful discussions on the topic of this paper, 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, supported by the Netherlands Organisation for Scientific Research (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.

REFERENCES

[1] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262–283, 2003.
[2] J. L. Bentley and J. B. Saxe. Decomposable searching problems I. static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.
[3] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *Proceedings of the 7th International Symposium on String Processing and Information Retrieval*, pages 39–48, 2000.
[4] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.
[5] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2(1-4):153, 1987.
[6] Y. Ge, H. Xiong, Z.-h. Zhou, H. Ozdemir, J. Yu, and K. C. Lee. Top-eye: Top-k evolving trajectory outlier detection. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1733–1736, 2010.
[7] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.
[8] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.
[9] H. Imai and M. Iri. Polygonal approximations of a curve—formulations and algorithms. *Computational Morphology*, pages 71–86, 1988.
[10] J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. In *Proceedings of the 24th International Conference on Data Engineering*, pages 140–149, 2008.
[11] X. Li, J. Han, S. Kim, and H. Gonzalez. Roam: Rule-and motif-based anomaly detection in massive moving object data sets. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 273–284, 2007.
[12] M. Montanino and V. Punzo. Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns. *Transportation Research Part B: Methodological*, 80:82–106, 2015.
[13] P. Newson and J. Krumm. Hidden Markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 336–343, 2009.
[14] M. H. Overmars and J. van Leeuwen. Worst-case optimal insertion and deletion methods for decomposable searching problems. *Information Processing Letters*, 12(4):168–173, 1981.
[15] V. Punzo, M. T. Borzacchiello, and B. Ciuffo. On the assessment of vehicle trajectory data accuracy and application to the Next Generation SIMulation (NGSIM) program data. *Transportation Research Part C: Emerging Technologies*, 19(6):1243–1262, 2011.
[16] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *Proceddings of the 18th International Conference on Data Engineering*, pages 673–684, 2002.
[17] G. Yuan, S. Xia, L. Zhang, Y. Zhou, and C. Ji. Trajectory outlier detection algorithm based on structural features. *Journal of Computational Information Systems*, 7(11):4137–4144, 2011.
[18] Y. Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology*, 6(3):29, 2015.