Maximum Physically Consistent Trajectories

Bram Custers[1], Mees van de Kerkhof[2], Wouter Meulemans[1], Bettina Speckmann[1], and Frank Staals[2]

1 **TU Eindhoven, the Netherlands**
[b.a.custers,w.meulemans,b.speckmann]@tue.nl
2 **Utrecht University, the Netherlands**
[m.a.vandekerkhof,f.staals]@uu.nl

Abstract

We study the problem of detecting outlying measurements in a GPS trajectory. Our method considers the physical possibility for the tracked object to visit combinations of measurements, using simplified physics models. We aim to compute the maximum subsequence of the measurements that is consistent with a given physics model. We give an $O(n \log^3 n)$ time algorithm for 2D-trajectories in a model with unbounded acceleration but bounded velocity, and an output-sensitive algorithm for any model where consistency checks can be done in $O(1)$ time and consistency is transitive.

1 Introduction

The use of GPS trajectories in computing has greatly increased in recent years, with many algorithms and applications using them as input for analysis. However, since trajectories are created by performing real-world measurements there is always some amount of spatial error. Thus, no matter the application, it is important to preprocess trajectories to try and reduce the impact measurement errors have on the result. There are several types of preprocessing that can be applied to a trajectory to limit the impact of errors. We focus on *outlier detection*, where we find and remove data points with a large distance to the rest of the data. These outliers are likely to have been the result of large measurement errors.

We approach the problem from a physics perspective. Trajectories track real-world objects with real-world physical properties. By considering if the tracked object could physically travel between its measured locations in the given time interval we can find outlying measurements. We call a subsequence of the measurements of a trajectory *consistent* if there is a path the object could have taken that visits the points in order and does not violate the constraints of (is consistent with) the physics model. Our problem is then to find the maximum consistent subsequence[1] of the trajectory.

Contributions. We give an output-sensitive $O(nd)$ time algorithm for finding the maximum consistent subsequence of a trajectory, where d is the number of outliers, assuming a physics model that allows consistency checks for a pair of measurements in $O(1)$ time and has transitive consistency. For the physics model where the object has bounded speed but no other restrictions we additionally give an $O(n \log^3 n)$ time algorithm.

Related work. Outlier detection is part of any application that must cope with imprecise data. Hence, many different methods have been developed for many different contexts. A general survey of outlier detection methodologies is given in [6]. Gupta et al. [5] give a survey for outlier detection in data with a temporal component, including trajectories.

[1] This subsequence does not need to be a consecutive subsequence.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Outlier detection for trajectories has mainly been studied as finding outlying trajectories in a data set of trajectories, rather than finding outlying measurements in a single trajectory [4, 8, 9, 13]. Although detection of outlying measurements in a single trajectory has not been studied as such, the problem shows similarities to trajectory simplification and trajectory smoothing, both well-studied topics. We refer to [14] for a survey.

2 Definitions & Notation

f A trajectory $T = \langle p_1, p_2, \ldots, p_n \rangle$ is an ordered sequence of n measurements, each of which contains at least a location $\mathbf{x}_i$ and timestamp t_i. Additional data such as acceleration and velocity may also be present. The measurements are ordered by their timestamp, so for any pair (p_i, p_j) with $i < j$ we have that $t_i < t_j$.

We use physics models to describe possible states that can be reached in the future, given some measurement at a given time. Under a particular model, a subsequence $\langle p_i, \ldots, p_j \rangle$ is *consistent* iff a path exists that connects all measurements and obeys the constraints of the physics model. We will denote this by $C(p_i, \ldots, p_j)$. We will restrict ourselves to models where existence of such a path per measurement pair can be checked in $O(1)$. In addition, we require that the consistency is transitive. That is, for all $i < j < k$ we have that

$$C(p_i, p_j) \wedge C(p_j, p_k) \iff C(p_i, p_j, p_k). \tag{1}$$

For example, a model that bounds only the maximum speed meets the above criteria. We use the generic model in Section 3 and the speed-bounded model in Section 4. An example of a model without transitivity is one where both speed and acceleration are bounded. Here, the situation can occur that reaching p_j from p_i requires a velocity at t_j that makes reaching p_k within the physics model impossible, while the separate pairs are consistent.

3 Output-sensitive algorithm for models with transitive consistency

We assume the generic physics model with an $O(1)$-time consistency check and transitive consistency. First, we observe that we can follow a similar methodology to the Imai-Iri simplification algorithm [7]. Let $G = (V, A)$ be a directed acyclic graph with a vertex for each measurement of T and an arc from a vertex v_i to v_j if $C(p_j, p_i)$. This graph has $O(n^2)$ complexity and we can test for the existence of each arc in constant time: construction takes $O(n^2)$ as well. Since consistency is transitive, a path in G corresponds to a consistent subsequence. We can determine the longest path, and thus the maximum consistent subsequence, in this graph in $O(|V| + |A|) = O(n^2)$ time.

We now show that we can further use transitivity to obtain an output-sensitive variant of this algorithm. Rather than constructing the full DAG, we build a subgraph where each vertex has at most one incoming arc coming from the vertex with the longest subsequence that is consistent with it. This is captured in the theorem below.

▶ **Theorem 1.** *Consider a physics model that allows checking the consistency of a pair of measurements in constant time, and where consistency between measurements is a transitive relation. The maximum consistent subsequence of a trajectory $T = \langle p_1, p_2, \ldots, p_n \rangle$ can be computed in $O(nd)$ time, where d is the number of outliers.*

Proof. We handle the measurements in chronological order. We maintain a linked list that stores all previously handled measurements, sorted in decreasing order of the largest length of a path that ends in that measurement. For a new measurement, we go down this list in

order, stopping as soon as we find a measurement that is consistent with the new one. We then know that the length of the longest path ending in the new measurement is equal to the length of the path ending in this previous measurement plus 1. During the list traversal, we maintain the first occurrence of the latest unique path length, so that we can insert the new element in O(1) time. After we have handled all measurements in this way we know the longest path. Each of the $(n-d)$ measurements that ends up in the longest path requires only 1 successful check preceded by at most d failed checks, and the d outliers require at most n checks, giving $O(nd)$ total checks which is also the time bound. ◀

4 Subquadratic algorithm for the speed-bounded model

Here we consider the speed-bounded model for 2D trajectories. We denote the maximum speed by s_{max}. A subsequence S is consistent under this model if a path $r(t) = (x(t), y(t))$ exists with $||\frac{d}{dt}r(t)|| < s_{max}$ and $(x(t_i), y(t_i)) = (x_i, y_i)$ for all $p_i \in S$.

We develop an insertion-only data structure that, given a measurement q, can determine the length k of the maximum 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.

4.1 Consistency data structure

We may view the measurements as points in 3D space, with the third axis being time, i.e. $p_i = (x_i, y_i, t_i)$. For all $t > t_i$, measurements that are consistent with p_i must lie inside a cone, starting at p_i with radius $s_{max}(t-t_i)$ at time $t \geq t_i$. This representation of consistency bears some similarity to the space-time prisms [10].

To determine if a measurement p_i is consistent with any measurement of a set P of previous measurements, we use an *additively weighted Voronoi diagram* (AWVD) for the elements. Given a set of points $\{r_1, \ldots, r_n\}$ and weights $\{w_1, \ldots, w_n\}$, this diagram partitions the plane into cells $\{c_1, \ldots, c_n\}$ associated with the points, such that for any point $r' \in c_i : d(r', r_i) - w_i \leq d(r', r_j) - w_j$, for all $r_j \neq r_i$. Here, d is a distance measure (in our case the Euclidean distance), and the equality holds only on boundaries between cells.

We use the locations of the measurements in the set P for $\{r_i\}$ and for the weights, we pick $s_{max}(t'-t_i)$ for every measurement for some $t' > t_n$. This results in the construction depicted in Figure 1. On this AWVD, we construct a point location structure $\mathcal{D}$. A consistency query with p_i on $\mathcal{D}$ consists of two steps: first, we retrieve the element r_c such that cell c_c contains the location of p_i; then, we return p_c if it is consistent with p_i, or otherwise the query "fails".

▶ **Lemma 2.** *Let $\mathcal{D}$ be a consistency data structure on a set P of measurements. If a consistency query with p_i fails on $\mathcal{D}$, no measurement of P is consistent with p_i.*

Proof sketch. The definition of AWVD tells us that the distance between the found location in $\mathcal{D}$ and the query point, minus the points weight is at most the same value for any other location. Using the strict inequality we get from inconsistency, we can derive that all other locations in $\mathcal{D}$ are also inconsistent. ◀

We can construct the AWVD in $O(m \log m)$ time on m elements, where the complexity of the AWVD is $O(m)$ [3]. We may then construct a point location structure in $O(m \log m)$ time on the AWVD that allows for querying in $O(\log m)$ time in which cell a point is situated [2]. As a consistency check costs $O(1)$ time, we obtain the lemma below.

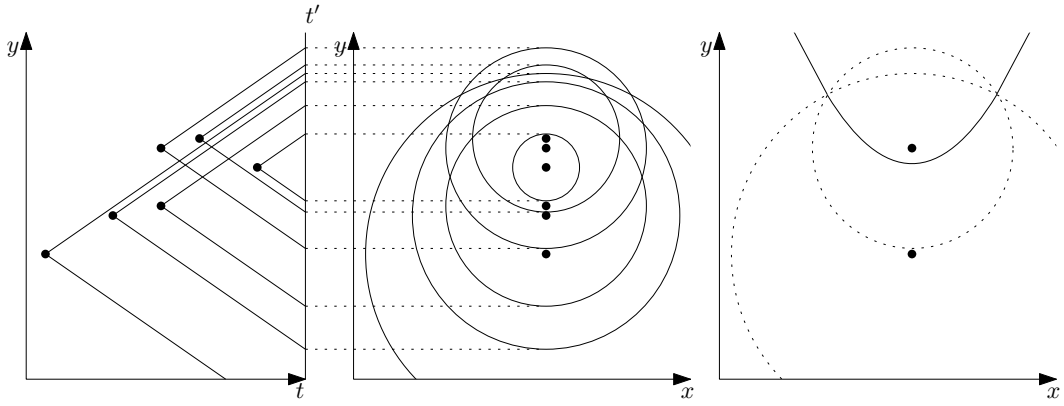

Figure 1 Example of the AWVD on measurements with equal x-coordinates. (Left) The y-coordinate of the measurements in time and the maximum speed cones progressing in time. (Middle) A cross-section of the (x, y, t) cones at time t' with accompanying centers (x, y). (Right) The corresponding additively weighted Voronoi diagram.

▶ **Lemma 3.** *Let P be a subset of T with $|P| = m$. We can construct a data structure $\mathcal{D}$ for querying consistency of a later measurement with any of the measurements in P in $O(m \log m)$ time. The data structure supports a consistency query in $O(\log m)$ time.*

4.2 Insertion-only consistency data structure

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 Bently and Saxe [1] 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 $\mathcal{D}_1, .., \mathcal{D}_{O(\log m)}$. 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}. We repeat this process until no duplicates exist. 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. By the results of Overmars and Van Leeuwen [12], the insertion time can be made a worst-case running time.

▶ **Lemma 4.** *We can modify the consistency data structure $\mathcal{D}$ to be insertion-only, providing a $O(\log^2 m)$ query time and $O(\log^2 m)$ insertion time in the worst-case.*

4.3 A BB[α] tree for maximum subsequences

We combine processed measurements and the consistency data structure in a BB[α] tree [11]. We store the measurements in the leaves along with the length of the maximum consistent subsequence ending at the measurement, k. In the intermediate nodes we keep an insertion-only consistency structure as described before, built on all measurements in the subtree. The leaves are kept sorted on k in ascending order.

We use the BB[α] tree to keep the tree of depth $O(\log n)$, while avoiding tree rotations at insertions which require full recomputation of the intermediate node data structures.

Instead, we rebuild a subtree when it becomes sufficiently unbalanced, which gives us a better amortized running time.

We can query the tree with a measurement p to determine a predecessor q with largest k such that $C(q, p)$: at each level, we query the right child data structure or leaf measurement if a measurement is consistent with p. If so, we traverse the right subtree, otherwise the left subtree. We continue until we reach a leaf, giving us the largest k with associated measurement that is consistent with the query measurement. Since we query $O(\log n)$ intermediate nodes, this process takes $O(\log^3 n)$ time.

After a query, we can insert the measurement p with value $k+1$ in the tree. We insert the measurement in the consistency data structures of all ancestor nodes and possibly rebalance the tree by rebuilding an unbalanced subtree. Using the BB[α] tree and the insertion-only data-structures, this takes $O(\log^3 n)$ amortized time.

▶ **Lemma 5.** *Consider a BB[α] tree storing an insertion-only consistency structure at each internal node. We can insert a new node into the tree, rebalance it and rebuild the necessary consistency structures in $O(\log^3 n)$ amortized time per insertion.*

4.4 Running-time analysis

To determine the maximum consistent subsequence, we need to process all measurements, querying for their predecessor in the tree and then inserting it into the tree. Finally, we query for the largest k value. By maintaining pointers from measurements to their predecessors during processing, we obtain the maximum consistent subsequence. Since we do a query and an insert per measurement, the algorithm runs in $O(n \log^3 n)$ as stated before.

▶ **Theorem 6.** *Given a trajectory T with n measurements, we can determine the maximum consistent subsequence of T for the 2D speed-bounded model in $O(n \log^3 n)$ time.*

5 Acknowledgement

Research on the topic of this paper 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.

References

1 J. L. Bentley and J. B. Saxe. Decomposable searching problems i. static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.
2 H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.
3 S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2(1-4):153, 1987.
4 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.
5 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.
6 V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.
7 H. Imai and M. Iri. Polygonal approximations of a curve—formulations and algorithms. *Computational Morphology*, pages 71–86, 1988.

8 J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. In *Proceedings of the IEEE 24th International Conference on Data Engineering*, pages 140–149, 2008.

9 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.

10 H. J. Miller. Time geography and space-time prism. *International Encyclopedia of Geography: People, the Earth, Environment and Technology*, pages 1–19, 2017.

11 J. Nievergelt and E. M. Reingold. Binary search trees of bounded balance. *SIAM Journal on Computing*, 2(1):33–43, 1973.

12 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.

13 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.

14 Y. Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.