# Segmentation of Trajectories on Non-Monotone Criteria

Boris Aronov* Anne Driemel† Marc van Kreveld† Maarten Löffler† Frank Staals†

**Abstract**

In the trajectory segmentation problem we are given a polygonal trajectory with $n$ vertices that we have to subdivide into a minimum number of disjoint segments (subtrajectories) that all satisfy a given criterion. The problem is known to be solvable efficiently for *monotone* criteria: criteria with the property that if they hold on a certain segment, they also hold on every subsegment of that segment [4]. To the best of our knowledge, no theoretical results are known for non-monotone criteria.

We present a broader study of the segmentation problem, and suggest a general framework for solving it, based on the *start-stop diagram*: a 2-dimensional diagram that represents all valid and invalid segments of a given trajectory. This yields two subproblems: (*i*) computing the start-stop diagram, and (*ii*) finding the optimal segmentation for a given diagram. We show that (*ii*) is NP-hard in general. However, we identify properties of the start-stop diagram that make the problem tractable, and give polynomial-time algorithm for this case.

We study two concrete non-monotone criteria that arise in practical applications in more detail. Both are based on a given univariate attribute function $f$ over the domain of the trajectory. We say a segment satisfies an *outlier-tolerant criterion* if the value of $f$ lies within a certain range for at least a given percentage of the length of the segment. We say a segment satisfies a *standard deviation criterion* if the standard deviation of $f$ over the length of the segment lies below a given threshold. We show that both criteria satisfy the properties that make the segmentation problem tractable. In particular, we compute an optimal segmentation of a trajectory based on the outlier-tolerant criterion in $O(n^2 \log n + kn^2)$ time, and on the standard deviation criterion in $O(kn^2)$ time, where $n$ is the number of vertices of the input trajectory and $k$ is the number of segments in an optimal solution.

*Dept. of Computer Science and Engineering, Polytechnic Institute of NYU, United States; aronov@poly.edu

†Dept. of Information and Computing Sciences, Utrecht University, the Netherlands; {a.driemel, m.j.vankreveld, m.loffler, f.staals}@uu.nl

## 1 Introduction

Trajectory data is ubiquitous today. Researchers in different fields study the movements of animals [3, 5, 11], hurricanes [15], traffic [12], or other moving objects [9] by analyzing their spatio-temporal trajectories. Movement data is often collected using GPS, and large sets of such data have been collected in the last few years. A trajectory is usually modeled as a continuous function from a time interval (the domain) to the plane, but is collected as a discrete sequence of time-stamped locations. By linearly interpolating the locations we obtain a continuous piecewise-linear curve as the image of the function. We can derive *attributes* from this data: functions defined on the domain of the trajectory. For example speed, heading, or acceleration. In some applications, trajectories may also have additional data stored with each time stamp. Some of these attribute functions, such as *speed* and *heading*, are piecewise constant when we assume linear interpolation.

**Segmentation.** An important analysis task is to segment a trajectory: partition the input trajectory into a minimum number of *segments* such that each segment satisfies a given criterion [13]. A criterion is usually defined based on an attribute of the trajectory. For example, for the attribute "speed," we could segment the trajectory such that the minimum and maximum speed within any segment differ by at most $h$ km/h, or by at most a factor of two. Figure 1 shows an example segmentation of a trajectory into segments of similar speed. This way, a segment of the trajectory represents a contiguous subtrajectory for which a property is stable in some sense. This aids the automated analysis, since such segments are often meaningful features of the trajectory. In the analysis of the trajectories of birds, for example, the goal is to extract the stretches where a certain activity is observed, such as soaring, directional flight, sleeping, etc. For more examples, see also [14, 17]. One may draw the comparison to the segmentation of an image into contiguous patches of similar color or texture. In these problems one is also interested in decomposing the image into meaningful portions, which represent objects shown [10].

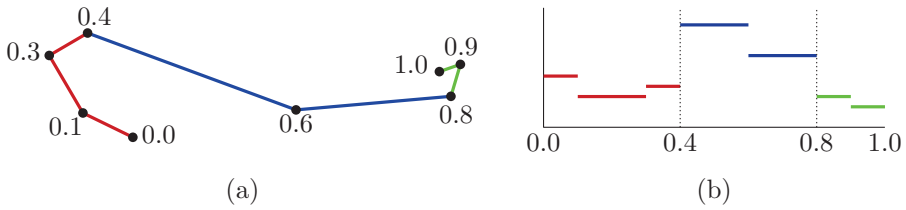


Figure 1: (a) A trajectory, obtained by linearly interpolating seven locations measured at irregular times. (b) The speed along the trajectory is a piecewise-constant function. We may divide the trajectory into three segments with "similar" speed: $[0.0, 0.4], [0.4, 0.8], [0.8, 1.0]$.

Depending on the coarseness of the data, we distinguish two segmentation problems: *continuous* and *discrete*. If the data has a fine resolution and the interpolation between two consecutive points is truthful, we want to segment continuously along the interpolated curve. In practice, it may happen that long irregular gaps exist between consecutive data points. For example, when tracking migrating birds, a GPS receiver will be programmed to use a lower frequency to save battery time and storage space as soon as migration starts. In this case, one may want to resort to a discrete segmentation, which groups consecutive data points into subsets (segments) that satisfy the criterion.

We address both continuous and discrete segmentation in this paper. Algorithmically, continuous segmentation is much more challenging than discrete segmentation. For discrete segmentation we study a more general weighted version of the problem. One could think of different applications of the weighting. For example, if the sampling of a trajectory was at irregular time intervals, one could weight the data point by the total duration that the measurement was valid. Studying this slightly more general problem does not influence the running times of our algorithms.

Previous research on similar segmentation problems has been done in animal movement studies [9, 14] and time-series analysis [2, 6, 16], see also [18]. These solutions provide no guarantees for individual segments in the segmentation. Instead, they are either heuristics or they optimize a global error criterion when a desired number of segments is specified. In the latter case, dynamic programming is a common approach [13]. The one exception is the research of Buchin *et al.* [4], see the discussion below.

**Monotonicity.** A criterion $C$ is *monotone* if it has the property that if $C$ is satisfied on some segment $S$ of a trajectory, then it is also satisfied on any subsegment $S' \subseteq S$. Buchin *et al.* [4] provide a framework for segmenting trajectories based on a monotone criterion or a combination of several monotone criteria. Many criteria are monotone by nature. However, if the trajectory data is noisy, or a type of behavior is briefly interrupted, it is desirable to weaken the monotonicity requirement. For example, instead of requiring that the difference in speed within a segment is at most $h$, we could require that this is the case for at least 95% of the time within each segment. A short burst in speed does not cause more segments in the segmentation with such a criterion. Another example is using a threshold for the standard deviation of speed within a segment.

**Our results.** To segment a trajectory based on non-monotone criteria, we introduce the notion of the *start-stop diagram*. This allows us to split the problem into two sub-problems: computing the start-stop diagram and computing an optimal segmentation for a given start-stop diagram. We show that the latter problem is NP-hard in general. However, a polynomial-time solution exists if the start-stop diagram has certain properties. We identify these properties in Section 3 and present algorithms to compute an optimal segmentation. In Section 4, we consider combining multiple criteria and we see how this affects the properties of the start-stop diagram. In Section 5 we present lower bound constructions which show the tightness of some of our upper bounds. Section 6 considers the issue of computing the start-stop diagram; we show how to efficiently compute the start-stop diagram for two specific criteria. One such criterion requires that the minimum and maximum values of a piecewise-constant function $f$ on each segment have at most a given difference, while allowing a certain percentage of outliers. The second criterion requires that on each segment the standard deviation of $f$ is below a certain threshold on each segment. We show that these criteria satisfy the properties that make the segmentation problem tractable, and in both cases we provide solutions to the discrete and the continuous segmentation problem.

### 1.1 Discrete vs. continuous segmentation

As we will see, the continuous segmentation problem is much harder than the discrete version. Therefore, the

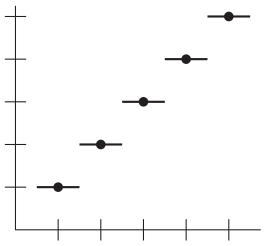

Figure 2: A discrete segmentation input (points) and its continuous analogue (line segments).

algorithm for continuous segmentation is also slower than the one for discrete segmentation. So perhaps a natural question is if continuous segmentation has advantages over discrete segmentation.

We now show that this is indeed the case: comparable situations can give a significantly different number of segments in an optimal segmentation. Consider the discrete function $f(i) = i$ for all $i \in \{1, \ldots, n\}$ and the very similar continuous function $f(x) = \lfloor x + \frac{1}{2} \rfloor$ for $x \in [\frac{1}{2}, n + \frac{1}{2}]$ (see Figure 2 for $n = 5$).

Suppose we segment on standard deviation with threshold 0.499. Then the discrete segmentation must take each index separately, because two consecutive indices give a standard deviation of 0.5 and this only gets higher if we take longer segments. The continuous segmentation can take as the first segment the first constant part with $f(x) = 1$ and nearly all of the second part with $f(x) = 2$. Similarly, the second segment can take the small remaining part where $f(x) = 2$, the entire part where $f(x) = 3$, and nearly all of the part where $f(x) = 4$ (but slightly less than before). The construction scheme leads to nearly half as many segments in the minimal continuous segmentation as in the minimal discrete segmentation.

## 2 General Approach

We study the problem of segmenting a trajectory with respect to a criterion. In the continuous case, this problem can be defined as follows. We define a trajectory $T$ as a function from an interval $I = [0, 1]$ to $\mathbb{R}^2$ (or $\mathbb{R}^d$) and a subtrajectory, also called *segment*, $T[a, b]$ as the function restricted to the subinterval $[a, b] \subseteq I$. A criterion $C$ is a function $C\colon I \times I \to \{True, False\}$, which is defined on all possible segments of $T$. We say an interval $[a, b] \subseteq I$ *satisfies* a criterion $C$ if $C(a, b) = True$; in this case we call the segment *valid*. A partitioning of $I$ (or of $T$) into non-overlapping segments whose union covers $I$ is called a *segmentation*. A segmentation of *size* $k$ can be denoted by its segments $[\tau_0, \tau_1], [\tau_1, \tau_2], \ldots, [\tau_{k-1}, \tau_k]$; $\tau_0 = 0$ and $\tau_k = 1$. A segmentation is *valid* if and only if all of its segments are valid, and *segmenting* a function refers to partitioning into valid segments. We say a valid segmentation is *minimal* (optimal) with respect to $C$ if its size is minimum; the segmentation problem is to compute a valid minimal segmentation. We will often omit the word "valid" because only valid segmentations are useful.

A criterion is often based on an *attribute function* $f\colon I \to \mathbb{R}$ that has the same domain $I$ as $T$, for example the function that maps time to speed at that time. A segmentation of $I$ based on the function $f$ trivially induces a segmentation of $T$ (see Figure 1), therefore we may use the term "segmenting $f$" to denote the resulting segmentation. We assume that $f$ is piecewise constant and we call the points where the value of $f$ changes the *breakpoints*. The portions between consecutive breakpoints, where $f$ stays constant, are called *pieces*.

The discrete segmentation problem is defined analogously. Here, a function $f\colon U \to \mathbb{R}$, where $U = \{1, \ldots, n\}$ is an index set, is given. A segmentation is a partition of $U$ into disjoint contiguous subsequences called *discrete segments*. The discrete segmentation problem on a given criterion is to compute a segmentation into a minimum number of valid discrete segments. We will address the *weighted* version: $f$ is given as a sequence $S = s_1, \ldots, s_n$ of pairs $s_i = (v_i, w_i)$, where $v_i$ is the value of piece $i$ and $w_i$ is its weight. Values and weights influence the criterion and therefore the validity of segments.

### 2.1 The start-stop diagram

To compute a minimal segmentation of $f$ we define the *start-stop diagram*. Consider the parameter space of the set of subintervals of $I$. For any candidate segment $[a, b]$, we associate the start parameter $a$ with the horizontal axis and the stop parameter $b$ with the vertical axis in the diagram. For continuous segmentation, any point $(a, b)$ in this diagram, with $a < b$, represents a candidate segment. Thus, the set of candidate segments is represented by the points in the upper left triangle of the unit square. The set of points which represent candidate segments that satisfy the criterion defines the *free space* in the start-stop diagram. The remaining points constitute the *forbidden space*. A segmentation of $I$ into a sequence of segments $[\tau_0, \tau_1], [\tau_1, \tau_2], [\tau_2, \tau_3], \ldots, [\tau_{k-1}, \tau_k]$; $\tau_0 = 0$ and $\tau_k = 1$, corresponds to a *staircase* in the start-stop diagram whose convex vertices correspond to the points $(\tau_i, \tau_{i+1})$ and whose concave vertices lie on the main diagonal. Note that a segmentation is valid if and only if all convex vertices lie in the free space (see Figure 3(a) for an example). The size of a staircase, that is, the number of convex vertices, corresponds to the size of

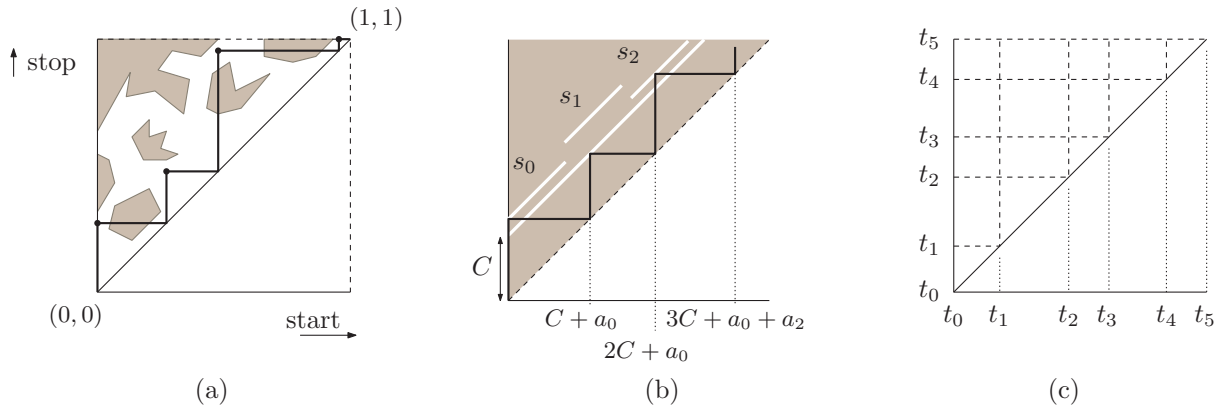


Figure 3: (a) The start-stop diagram and a valid segmentation into four segments. (b) Reduction from subset sum to the abstract segmentation problem. The staircase shows the sum $a_0 + a_2$. (c) The subdivision of the start-stop diagram into cells according to the breakpoints of $f$.

the segmentation. The start-stop diagram is reminiscent of the free space diagram used to compute the Fréchet distance [1]; however, there one is interested in a path that stays inside the free space entirely.

For discrete segmentation, we consider the *start-stop matrix*, a simplification of the start-stop diagram that is an $n \times n$ matrix $B$ with Boolean values, where the entry $B(i, j)$ at the $i$th column and $j$th row from below corresponds to the value of the criterion function at $(i, j)$.[1]

### 2.2 Segmenting in the discrete case

In the discrete case it is relatively easy to compute a minimal segmentation by using dynamic programming. In fact, if one treats $B(\cdot, \cdot)$ as the adjacency matrix of an unweighted directed acyclic graph, a minimal segmentation corresponds precisely to a shortest path from 1 to $n$ and can be found in $O(n^2)$ time [7].

THEOREM 2.1. *Given an $n \times n$ start-stop matrix, one can check if a minimal discrete segmentation exists, and if so compute it, in $O(n^2)$ time and space.*

### 2.3 Segmenting in the continuous case

Continuous segmentation is much harder. In the *abstract segmentation problem*, we are given a decomposition $D$ of the triangle spanned by $(0, 0)$, $(0, 1)$ and $(1, 1)$ into points, $a$-monotone bounded-degree curve segments (with horizontal axis $a$), and faces, each of which is marked either *free* or *forbidden*. Let $n$ be the total complexity of $D$. In the resulting start-stop diagram, we need to find a staircase that represents a valid, minimal segmentation. We show that even testing the existence of a valid segmentation is already NP-hard.

THEOREM 2.2. *The abstract segmentation problem is NP-hard.*

*Proof.* We reduce from SUBSET-SUM. Let $a_0, \ldots, a_{n-1}$ be a set of positive integers and let $B$ be the desired subset sum. We generate an instance of the abstract segmentation problem by constructing $n+1$ line segments in the start-stop diagram; these line segments are precisely the free space.

Let $A = \sum_{i=0}^{n-1} a_i$ and $C = A + 1$. We generate a start-stop diagram of size $[0, (n+1)C + B] \times [0, (n+1)C + B]$. One line segment $s$ of the free space has its endpoints at $(0, C)$ and at $(nC + B, (n + 1)C + B)$ (see Figure 3(b)).

For each $a_i$ we create a line segment $s_i$ with endpoints $(iC, (i + 1)C + a_i)$ and $(iC + A, (i + 1)C + a_i + A)$. This segment lies $a_i$ units vertically above the first line segment $s$ in the start-stop diagram. The placement of the segments is such that each staircase must choose to have its first convex vertex on $s$ or $s_0$, its second convex vertex on $s$ or $s_1$, and so on. Each subsequent step places the concave vertex $C$ units further along the diagonal if $s$ is chosen, and $C + a_i$ units if $s_i$ is chosen. Each staircase that ends at $((n + 1)C + B, (n + 1)C + B)$ has exactly $n+1$ convex vertices, the last one necessarily on $s$. Furthermore, we can end at $((n+1)C+B, (n+1)C+B)$ if and only if the chosen $s_i$ are such that the corresponding values $a_i$ sum up to $B$.

The length of the segments $s_i$ is chosen so that we can select $s_i$ no matter what segments have been picked

[1] Usually a matrix is indexed by row first and column second. We switch the indices to be consistent with the continuous case, and because it feels more natural in light of the application.

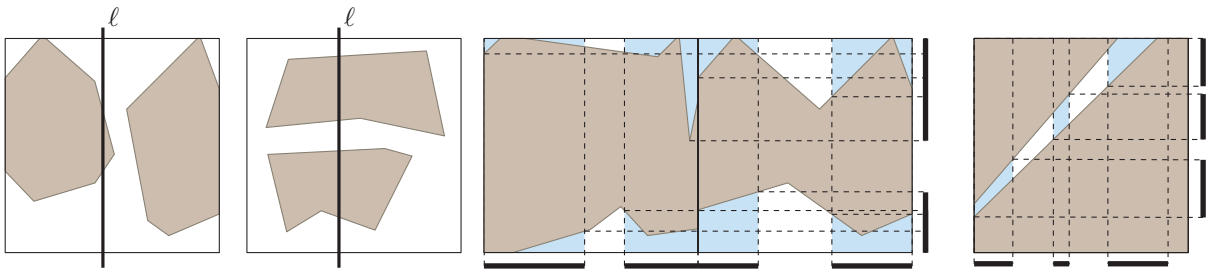


Figure 4: (a) A verticonvex cell. (b) A tunnel cell. (c) Any row of verticonvex cells produces at most two $k$-reachable intervals. (d) A tunnel cell can produce more than two $k$-reachable intervals.

before. At the same time, the step size of at least $C$ in every step makes sure that we cannot make more than one step on the same $s_i$. Clearly the reduction is polynomial. □

## 3 Solving the Segmentation Problem in the Continuous Case

Even though the general continuous segmentation problem is NP-hard, we identify some properties of the start-stop diagram that make the problem tractable.

In the following, we assume that there exists a grid decompositon of size $n \times n$ or smaller of the start-stop diagram. When $f$ is a piecewise-constant function there is a natural decomposition of the start-stop diagram into a grid by the breakpoints of $f$ (see Figure 3(c)). In this case each cell of the grid corresponds to a set of candidate segments where the start and stop points lie on fixed pieces of $f$. The cells incident to the diagonal are triangular and represent candidate segments with start and stop points lying in the same piece.

**Computing the reachable space.** Recall that a minimal segmentation corresponds to a minimum-link staircase in the start-stop diagram. We define the *$k$-reachable space* $R_k$ as the set of points on the diagonal that can be reached from $(0,0)$ by a valid staircase of size at most $k$. We identify the (connected components of the) reachable space $R_k$ with a set of *intervals* on $I$. The number of intervals is the *complexity* of the reachable space. For a fixed $k$, the *incoming $k$-reachable intervals* of a cell $C_{ij}$ are the intervals of $R_k$ intersected with the $i$th column. We call the union of vertical (resp. horizontal) lines intersecting the interval $X$ the vertical slab (resp. horizontal slab) induced by an interval $X$. Consider the valid staircases of size at most $(k+1)$ which have their last convex vertex in such a vertical slab $V_X$. We call the connected components of the set of points on the diagonal that can be reached by such a staircase the *outgoing $(k+1)$-reachable intervals* produced by $X$.

The following algorithm describes an iterative procedure to compute $R_i$, for $i = 1, 2, \ldots$, starting with $R_0 = \{(0,0)\}$. The algorithm stops when it has found the smallest value of $i$ for which $R_i$ contains point $(1,1)$. Let this be $k$. An actual minimum-link staircase, and hence a minimal segmentation, can then be extracted from the sets $R_0, .., R_k$.

**Algorithm** ReachableSpace($S$)
*Input.* A start stop-diagram $S$
*Output.* The sets of reachable space $R_0, .., R_k$, where $k$ is the smallest $i$ such that $R_i$ contains an interval with point $(1,1)$.
1. $i \leftarrow 0; R_0 \leftarrow \{(0,0)\}$
2. **while** $(1,1) \notin R_i$ **do**
3. **for each** interval $X$ in $R_i$ **do**
4. Consider the intersection of the free space in $S$ with the vertical slab induced by $X$ and project it horizontally back to the diagonal.
5. The union of $R_i$ and these projections, over all $X$, forms $R_{i+1}$.
6. $i \leftarrow i + 1$
7. **return** $R_0, .., R_i$

We will make this algorithm more concrete when proving Theorem 3.2 and we will see that the running time depends on the shape and the distribution of the forbidden space in each cell. Note, for example, that a monotone criterion yields a monotone curve in the start-stop diagram. The region above the curve is the forbidden space, and the region below the curve is the free space. We call an object *verticonvex* if and only if its intersection with any vertical line $\ell$ is at most a single interval.[2] A start-stop diagram cell (or row) is

[2] A verticonvex region is also called "$x$-monotone" in the

*verticonvex* when the forbidden region within it is (see Figure 4(a)). We call a cell a *tunnel cell* if any vertical line $\ell$ intersects the forbidden space in that cell in at most two intervals (see Figure 4(b) for an example). We will show that for a given $n \times n$ start-stop diagram we can compute a minimal segmentation in $k$ segments in $O(kn^2)$ time if all cells are verticonvex, and in $O(k^2n^2)$ time if each row contains at most one tunnel cell and all other cells are verticonvex. If one allows cells where a vertical line can intersect the forbidden space in three or more intervals (i.e., double-tunnels), the problem becomes NP-hard as seen in the construction in the proof of Theorem 2.2. We will later see that even if we have only tunnel cells, but more than one tunnel cell in each row, we may create a reachable space of exponential complexity.

### 3.1 Verticonvex cells only

Consider the algorithm ReachableSpace. The intersection of a vertical line with the free space in a verticonvex cell consists of at most two (possibly empty) intervals: one connected to the top, and one connected bottom of the row. The horizontal projection onto the diagonal preserves this property, and the union of the projections of this type also consists of at most two intervals (see Figure 4(c)).

Observation 3.1. *For any $k$ and any row, the reachable space $R_k$ produced by the incoming intervals in the verticonvex cells in the row consists of at most two intervals. One of these intervals is connected to the top of the row, and the other one is connected to the bottom of the row.*

We now prove:

Theorem 3.2. *Given an $n \times n$ start-stop diagram in which the forbidden space in each cell is verticonvex and has constant complexity, we can compute a minimal segmentation in $O(kn^2)$ time, where $k$ is the size of a minimal segmentation.*

*Proof.* Assume that for the given instance of the problem a valid segmentation exists. We specialize the procedure used in algorithm ReachableSpace. Recall that the reachable space can be encoded as a set of intervals, each of which corresponds to a connected component on the diagonal. Since all cells are verticonvex, Observation 3.1 implies that the $k$-reachable space consists of $O(n)$ such intervals and as such we can store it as a set of $O(n)$ values. Given the reachable space $R_i$ we compute the reachable space $R_{i+1}$ in a row by row manner. By Observation 3.1 the $(i+1)$-reachable space restricted to any row consists of two intervals, one connected to the top and one to the bottom. In order to compute these two intervals we maintain the highest reachable point and lowest reachable point while iterating over the cells in the current row and handling the incoming $i$-reachable intervals to every cell. Observation 3.1 implies that there are at most two incoming $i$-reachable intervals to any cell, since it is nothing more than the $i$-reachable space restricted to the column that contains the cell. Furthermore, the complexity of the forbidden space in each cell has constant complexity. Therefore, we spend constant time per cell during this process and $O(n^2)$ time in total for computing $R_{i+1}$ from $R_i$. We perform this step until the top-right corner $(1,1)$ is contained in $R_{i+1}$. Since this happens for $i+1=k$, this takes $O(kn^2)$ time. An optimal segmentation can be extracted by standard dynamic programming methods. □

### 3.2 At most one tunnel per row

Our algorithm ReachableSpace is efficient when all rows in the start-stop diagram are verticonvex. If a row is not verticonvex, it may produce more than two intervals. Nonetheless, we show that if the start-stop diagram is not too complex—specifically, if each row contains at most one tunnel cell—the problem can still be solved efficiently. We need the following lemma to prove Lemma 3.2.

Lemma 3.1. *Let $C$ be a tunnel cell, and let $F \subseteq C$ be the forbidden space. An incoming interval $X$ can produce at most one outgoing interval $I$ such that (i) $I$ is neither incident to the top nor the bottom of the row; and (ii) $I$ does not intersect a horizontal line through a vertex of $F$.*

*Proof.* Assume, for a contradiction, that $X$ can produce two intervals with this property. Let them be denoted $I = [i_1, i_2]$ and $J = [j_1, j_2]$, where $i_2 < j_1$ (see Figure 5). Let $H_I$ denote the horizontal slab induced by $I$, define $H_J$ similarly, and let $V_X$ denote the vertical slab induced by $X$. Since $I$ does not intersect a horizontal line through a vertex of $F$, $H_I$ does not contain any vertices. The same holds for $H_J$.

Now, consider the interval $I$. Since $X$ produced outgoing interval $I$, there must be at least one point $p$ of free space in $V_X \cap H_I$. Since there are no vertices in $H_I$, either $p$ lies on a curve $\gamma$ of free space, or in a face of free

literature; we choose to use the new term to avoid confusion with "monotone criteria."

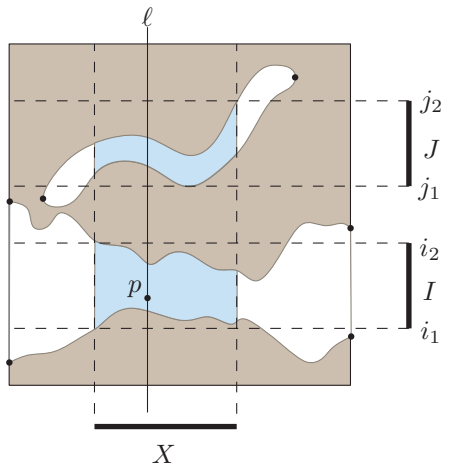


Figure 5: Illustration to the proof of Lemma 3.1. The vertices of the forbidden space are also shown.

space that is bounded by curves. Since the curves are $a$-monotone, they extend to the left and right outside $V_X$; in the second case this means the face that contains $p$ restricted to $V_X$ is bounded by two curves $\gamma_1$ and $\gamma_2$. It must be that $\gamma, \gamma_1, \gamma_2 \cap V_X \subseteq H_I$; otherwise, $I$ would be larger.

We can make the same argument about the interval $J$ and obtain one or two boundary curves as described above that stay inside $H_J$ when intersected with $V_X$. Clearly, they must also intersect a vertical line $\ell \subseteq V_x$ inside $H_J$. This implies that $\ell$ intersects $F$ at least three times: once below $i_1$, once between $i_2$ and $j_1$ and once above $j_2$. Therefore, $C$ is not a tunnel cell. □

LEMMA 3.2. *Let $C$ be a tunnel cell, and let $F \subseteq C$ be the forbidden space. Suppose $C$ has $m$ incoming $k$-reachable intervals and $u$ is the complexity of the union of the outgoing $(k+1)$-reachable intervals produced by them. Then $u \leq c + m + 2$, where $c$ is the number of vertices of $F$.*

*Proof.* We draw a horizontal line $h_v$ through every vertex $v$ of $F$. Let $H$ denote this set of lines. We charge every interval in the union of the outgoing intervals that intersects such a line $h_v$ to $v$. This way, each vertex gets charged at most once, since two intervals that would charge the same line cannot be disjoint. Thus, we have at most $c$ charges of this type.

By Lemma 3.1, each incoming interval can produce at most one outgoing interval that does not intersect a line of $H$ and is neither incident to the top nor to the bottom of the row. Therefore, we can charge those to the incoming intervals with at most $m$ charges.

The remaining intervals in the union are either incident to the top or the bottom of the row, and of those there can be at most two in total. This completes the proof. □

LEMMA 3.3. *In an $n \times n$ start-stop diagram in which (i) the forbidden space in each cell has constant complexity, (ii) each row contains at most one tunnel cell, and (iii) the remaining cells are verticonvex, $R_k$ consists of at most $O(kn)$ intervals.*

*Proof.* Consider the $k$-reachable space $R_k$ restricted to one row of the grid. It consists of the endpoints of all valid staircases that have their last convex vertex in a cell of this row.

We distinguish two types of staircases: (a) those that have their last convex vertex in a verticonvex cell and (b) those that have it in a tunnel cell. By Observation 3.1, the $k$-reachable space that is formed by the staircases of type (a) has constant complexity.

As for the staircases of type (b), they all have their last convex vertex in the same cell since there is only one tunnel cell per row. By Lemma 3.2, and since the forbidden space in every cell has constant complexity, the $k$-reachable space formed by those staircases consists of $m + O(1)$ intervals, where $m$ is the complexity of the $(k-1)$-reachable space $R_{k-1}$ restricted to the column that the tunnel cell lies in.

Thus, we have the following recurrence for the complexity of the $k$-reachable space restricted to one row: $C(k) \leq C(k-1) + O(1)$. Clearly, $C(1) \leq 3$, and therefore, we have that $C(k) = O(k)$. Since there are $n$ rows, the lemma follows. □

Since the complexity of the forbidden space is constant in each cell, we can list a set of $m$ incoming intervals and produce the at most $m + O(1)$ outgoing intervals in $O(m)$ time. Using essentially the same algorithm as before we then obtain:

THEOREM 3.3. *Given an $n \times n$ start-stop diagram in which (i) the forbidden space in each cell has constant complexity, (ii) each row contains at most one tunnel cell, and (iii) the remaining cells are all verticonvex, we can compute a minimal segmentation in $O(k^2 n^2)$ time, where $k$ is the size of a minimal segmentation.*

## 4 Combining Criteria

We can also consider segmenting a trajectory based on multiple criteria. For example, we want segments such

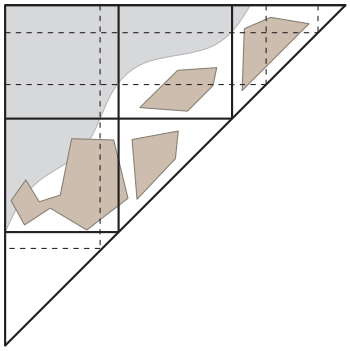

Figure 6: A refined grid.

that two criteria hold simultaneously, or where at least one criterion holds. We can combine criteria into new ones by taking conjunctions, disjunctions, and negations, and in general we can build any Boolean combination this way.

Again, we assume that there exists a grid decompositon of size $n \times n$ or smaller of the start-stop diagram for each of the criteria. To obtain the start-stop diagram for the criterion $C_1 \wedge C_2$ we simply overlay their grids and take the union of the forbidden space $F_1$ of $C_1$ and the forbidden space $F_2$ of $C_2$. Similarly, the forbidden space for $C_1 \vee C_2$ is the intersection of $F_1$ and $F_2$. Can we still solve the segmentation problem efficiently on the resulting start-stop diagram?

Buchin *et al.* [4] observe that the conjunction or disjunction of monotone criteria is again monotone. Similarly, observe that the start-stop diagram of a disjunction of two *verticonvex criteria*, i.e., criteria for which the start-stop diagram contains only verticonvex cells, again contains only verticonvex cells. Hence the disjunction of two verticonvex criteria is again a verticonvex criterion. Since a monotone criterion is also verticonvex, the disjunction of a monotone criterion with a verticonvex criterion results in a verticonvex criterion as well.

For the conjunction of two verticonvex criteria we take the union of their forbidden spaces. Unfortunately, this can lead to non-verticonvex cells. However, we can show that if one of the criteria is monotone, we can slightly modify the grid of the combined start-stop diagram such that it contains at most one tunnel cell in each row and the remaining cells are verticonvex. By Theorem 3.3 we can therefore still compute a minimal segmentation efficiently.

THEOREM 4.1. *Let $C_1$ be a monotone criterion, let $C_2$ be a verticonvex criterion, and let the overlay of their grids have size $n \times n$. If the complexity of the forbidden space in every cell is constant, then we can compute a minimal segmentation with respect to the criterion $C_1 \wedge C_2$ in $O(k^2n^2)$ time, where $k$ is the size of a minimal segmentation.*

*Proof.* We argue that there exists an $O(n) \times O(n)$ grid decomposition of the start-stop diagram of $C_1 \wedge C_2$ such that $(i)$ every row contains at most one tunnel cell, and $(ii)$ the remaining cells are verticonvex. The claim then follows from Theorem 3.3.

Recall that a monotone criterion corresponds to an $ab$-monotone curve $\gamma$ in the start-stop diagram (with axes $a$ and $b$), in which all points above the curve are forbidden space, and all points below the curve are free space. We subdivide the grid such that $\gamma$ intersects at most one cell in any row, and at most one cell in any column (see Figure 6). Since $\gamma$ is $ab$-monotone it intersects an original grid line at most once; this means we add at most $O(n)$ grid lines. The complexity of the forbidden space in each cell is constant.

Clearly, the cells of the refined grid that are not intersected by $\gamma$ are verticonvex: everything above $\gamma$ is forbidden space, and the forbidden space in other cells originates only from a verticonvex criterion. The cells that are intersected by $\gamma$ are tunnel cells since every vertical line intersects the forbidden space at most twice: at most once for the forbidden space resulting from the verticonvex criterion, and at most once for the forbidden space above $\gamma$. The claim now follows from Theorem 3.3. □

Note that essentially the same approach can also be used to solve the problem for the conjunction of a verticonvex criterion and the negation of a monotone criterion:

COROLLARY 4.1. *Let $C_1$ be a monotone criterion, let $C_2$ be a verticonvex criterion, and let the overlay of their grids have size $n \times n$. If the complexity of the forbidden space in every cell is constant, then we can compute a minimal segmentation with respect to the criterion $\neg C_1 \wedge C_2$ in $O(k^2n^2)$ time, where $k$ is the size of a minimal segmentation.*

## 5 Lower Bounds

The running time in Theorem 4.1 (and Theorem 3.3) is a factor $k$ worse than that in Theorem 3.2. We now give an example of an $n \times n$ start-stop diagram resulting from the conjunction of a monotone and a verticonvex criterion in which the complexity of the $k$-reachable space is $\Omega(kn)$. In this start-stop diagram, every cell

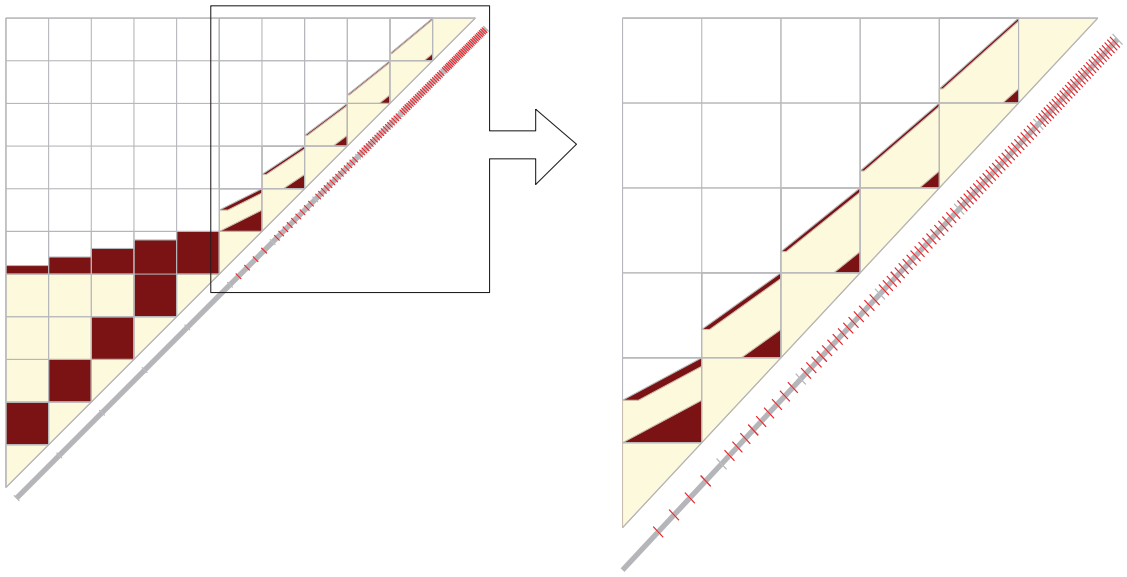

Figure 7: A start-stop diagram that results in a $k$-reachable space of complexity $\Omega(kn)$. Note that the forbidden space is shown in light colors and the free space is shown in dark colors in this figure. In particular, in white the forbidden space resulting from the monotone criterion, in lightyellow the forbidden space resulting from the verticonvex criterion, and in red the free space.

has constant complexity, every row contains at most one tunnel cell and all the remaining cells are verticonvex. This shows that the extra factor $k$ in the above theorems is necessary in the worst case. Though this construction shows we cannot hope to solve the abstract problem any faster, we do not expect this behavior to appear in practice.

THEOREM 5.1. *There exists an $n \times n$ start-stop diagram such that (i) the forbidden space is the union of the region above a monotone curve and one verticonvex region per cell (ii) the forbidden space in each cell is polygonal and has constant complexity, and (iii) the complexity of the reachable space $R_i$ after $i$ steps is $\Omega(in)$ for all $i < n/2$.*

*Proof.* For a given $k$, we construct a start-stop diagram of size $n = 2k + 1$ where the complexity of the reachable space in a single cell can be quadratic, and moreover, the total complexity is $\Omega(k^2)$. The construction is illustrated in Figure 7.

The construction consists of two parts. The first part consists of the first $k + 1$ columns, where we define the free space as follows. For $1 \leq i \leq j \leq k$, the cells $C_{ij}$ are completely free if $i + 1 = j$, and completely forbidden otherwise. In each cell $C_{i(k+1)}$, the bottom $i/k$ fraction is free, and the rest is forbidden. Cell $C_{(k+1)(k+1)}$ is completely forbidden, as are all cells where $i \leq k$ and $j \geq k+2$. The cells in the $k+1$th row are used to create a situation in the $k$th interval where the $i$-reachable space is different for each $i = 1, \ldots, k$, and the $i$-reachable space is contained in the $j$-reachable space for any $i < j$.

The second part consists of the last $k + 1$ columns (overlapping one column with the first part), and is used to duplicate this set of intervals $k$ times. Only cells $C_{(k+1+i)(k+2+i)}$ contain free space, for $0 \leq i \leq k$. Any other cells are completely forbidden. Consider a cell $C_{(k+1+i)(k+2+i)}$ and map it to the unit square. We create a triangle of free space in the lower right, with coordinates $(\frac{i}{i+1}, 0)$, $(1, 0)$, and $(1, \frac{i+1}{i+2})$. Furthermore, we create a tunnel of free space cutting diagonally through the remaining forbidden space at the top. This tunnel has its vertices at $(0, \frac{i+1}{i+2})$, $(1, 1)$, $(\frac{i+2}{(i+1)^2}, \frac{i+1}{i+2})$ and $(1, 1 - \frac{1}{i+1})$. The triangular piece of free space is used to take the last groups of $k$ intervals from the $k+i$th column and project it to the first part of the next column, while the line-shaped part of the free space copies all $i$ groups of $k$ intervals once, leading to $k^2$ intervals in the final cell. □

Next, we show that if we can have even more tunnel cells, then the reachable space can have an exponential complexity.

THEOREM 5.2. *There exists an $n \times n$ start-stop diagram such that (i) the forbidden space in each cell has constant complexity, (ii) each row contains at most two tunnel*

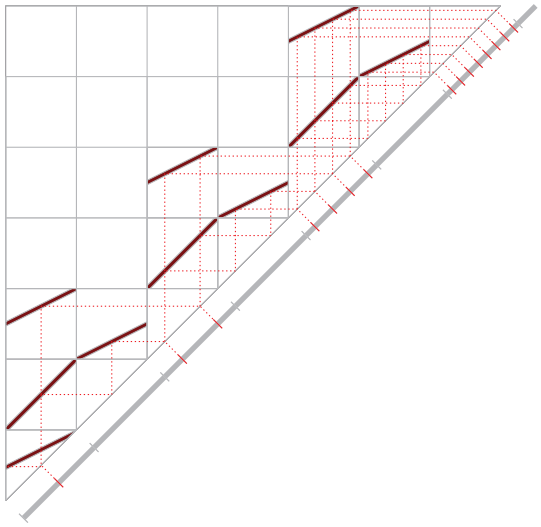

Figure 8: A start-stop diagram with at most two tunnels per row that results in a $2k$-reachable space of complexity $\Omega(2^k)$. Note that the forbidden space is shown in white in this figure.

*cells, (iii) the remaining cells are all verticonvex, and (iv) the complexity of the reachable space $R_i$ after $i$ steps is $\Omega(2^i)$, for all $i < n/2$.*

*Proof.* For a given $k$, we construct a start-stop diagram of size $n = 2k + 1$ where the complexity of the reachable space in a single cell can be exponential. The construction is illustrated in Figure 8. The construction consists of $k$ copies of a gadget that uses three tunnel cells, which effectively doubles the complexity of the reachable space in a single pass. □

## 6 Computing the Start-stop Diagram

In previous sections we focused on computing minimal segmentations for a given start-stop diagram or start-stop matrix. We now discuss two specific criteria, prove that they are verticonvex, and explain how to construct the corresponding start-stop diagram or start-stop matrix efficiently. For both the discrete- and the continuous segmentation problem this leads to polynomial-time algorithms. The criteria are defined on a piecewise-constant function $f$. We also show that if $f$ is piecewise-linear then the start-stop diagram may contain more than one tunnel cell per row.

### 6.1 The outlier-tolerant criterion

For a given piecewise-constant function $f$, we can segment $f$ such that in each segment $[a, b]$ the minimum and maximum values differ by at most $h$. To accomodate outliers we require that only a fraction $\rho$, for a given constant $0 \leq \rho \leq 1$, of a segment $[a, b]$ must have its values within a range of extent $h$, and a fraction $1 - \rho$ of $[a, b]$ may have any value.[3] We present efficient algorithms for computing the start-stop matrix and the start-stop diagram for this criterion. To obtain a segmentation algorithm for the continuous case, we further show that the forbidden space within each cell of the start-stop diagram is verticonvex.

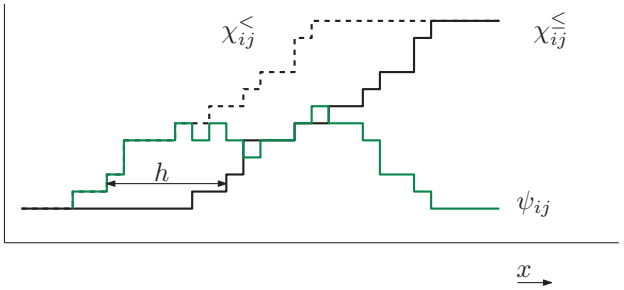


Figure 9: Obtaining the function $\psi_{ij}$ from $\chi^{\leq}_{ij}$ and $\chi^{<}_{ij}$ shifted by $h$ to the left (dashed).

#### 6.1.1 Discrete case

We consider $f$ as a sequence $s_1 = (v_1, w_1), \ldots, s_n = (v_n, w_n)$ of (value, weight) pairs. We first give a formal definition of the criterion and investigate its structure. Let $S_{ij}$ be the contiguous subsequence $s_i, \ldots, s_j$ and let $\lambda_{ij}$ be the total weight of its elements. Let $S^{\leq}_{ij}(x) = \{s_k \in S_{ij} \mid v_k \leq x\}$ denote the set of elements in $S_{ij}$ with value at most $x$, and let $\chi^{\leq}_{ij}(x) = \sum_{s_k \in S^{\leq}_{ij}(x)} w_k$ be the total weight of these elements. We define $S^{<}_{ij}(x)$ and $\chi^{<}_{ij}(x)$ analogously. The total weight of the elements in $S_{ij}$ with a value in the range $[x, x + h]$ is then

$$\psi_{ij}(x) = \chi^{\leq}_{ij}(x + h) - \chi^{<}_{ij}(x). \tag{6.1}$$

Our goal is now to segment $f$ so that, for each segment $S_{ij}$, we have $\max_x \psi_{ij}(x)/\lambda_{ij} \geq \rho$. Let $B$ be the start-stop matrix. To test the value $B_{ij}$, we could compute an explicit representation of $\chi^{\leq}_{ij}$ and an explicit representation of $\chi^{<}_{ij}$ shifted by $h$ to the left, and then take their difference (see Figure 9). This takes $O((j-i)\log(j-i))$ time. Since there are $O(n^2)$ cells the total amount of time required to compute the start-stop matrix is $O(n^3 \log n)$.

[3] Using the same algorithm we can also handle the following similar criterion by using the logarithm of the function. For a given piecewise-constant function $f$, we can segment $f$ so that in each segment $[a, b]$ there is a portion $V$ of total length at least $\rho(b-a)$ such that $w_{\max}/w_{\min} \leq h$, where $w_{\min}$ and $w_{\max}$ are the minimum and maximum function values that occur in $V$.

It is however not necessary to recompute the functions from scratch for each cell. Increasing $j$ by one corresponds to raising the functions $\chi_{ij}^{\leq}$ and $\chi_{ij}^{<}$ by $w_{j+1}$ for all values $x \geq v_{j+1}$. Therefore we have:

(6.2)
$$\psi_{i(j+1)}(x) = \begin{cases} \psi_{ij}(x) + w_{j+1} & \text{if } x \in [v_{j+1} - h, v_{j+1}] \\ \psi_{ij}(x) & \text{otherwise} \end{cases}$$

We now store $\psi_{ij}$ as a data structure that allows us to query the maximum of $\psi_{ij}$ on any given interval, and can be updated efficiently to represent $\psi_{i(j+1)}$. The data structure we use is an augmented segment tree that supports both operations in $O(\log n)$ time.[4]

**Data structure to compute the start-stop matrix.** We associate each piece $s_j = (v_j, w_j)$ of $f$ with an interval $I_j = [v_j - h, v_j]$ of *weight* $w_j$. By Equation (6.2) it now follows that the value of $\psi_{ij}(x)$ is equal to the total weight of the intervals from $I_1, .., I_j$ that contain $x$. Hence, we can represent $\psi_{ij}$ using the set of intervals $I_1, .., I_j$. We now describe an augmented segment tree $T$ that stores these weighted intervals.

Let $u_1, .., u_m$ be the endpoints of all intervals in $I_1, .., I_n$ in sorted order. Internally, a segment tree $T$ stores the elementary intervals $[u_i, u_{i+1}]$ in this order in the leaves of a balanced binary tree [8]. The internal nodes store values that allow searching on $u$-value. Since we know the endpoints of the intervals of $\psi_{ij}$ for all $i$ and $j$ in advance, we can initialize $T$ with the set of $O(n)$ endpoints and all weights set to zero. Therefore, no rebalancing has to be done when adding or removing an interval stored in the tree; only the values stored in the nodes which relate to weights will change.

Each node $\nu$ in a segment tree has an associated range $r_\nu$, and an associated set $\mathcal{I}_\nu$ of intervals. The range $r_\nu$ is the union of the elementary intervals stored in (the leaves of) the subtree rooted at $\nu$, and $\mathcal{I}_\nu$ is a subset of intervals stored in the tree. An interval $I$ occurs in $\mathcal{I}_\nu$ if and only if $I$ contains $r_\nu$ but not the range of $\nu$'s parent node [8]. We now augment $T$ such that each node $\nu$ stores the total weight $A_\nu$ of the intervals associated with $\nu$.

So for a tree $T$ representing the function $\psi_{ij}$ we can obtain the value of $\psi_{ij}$ at $x$ by searching for the elementary interval containing $x$ and sum over the $A$-values on the search path.

[4]The segments that are stored in this standard data structure as described in [8] are not to be confused with the segments of the segmentation of a trajectory.

A second augmentation provides us with a way to determine the maximum of $\psi_{ij}$ in a given interval in $O(\log n)$ time. This is done by storing a value $B_\nu$ at every node $\nu$ that is the maximum sum of all $A$-values on a path from $\nu$ to a leaf in the subtree rooted at $\nu$.

When querying $T$ for the maximum of the function $\psi_{ij}$ on a given interval $[a, b]$, we walk along the two paths from the root to the elementary intervals containing $a$ and $b$ and maintain the maximum of the $B$-values stored in the roots of the subtrees in between the two paths. This takes time linear in the number of nodes visited, hence we can find the maximum of $\psi_{ij}$ on $[a, b]$ in $O(\log n)$ time.

When inserting an interval $I_j = [v_j - h, v_j]$ with weight $w_j$, updating $A$- and $B$-values can be done in $O(\log n)$ time. The $A$-value needs to be increased by $w_j$ in the $O(\log n)$ nodes $\nu$ for which $I_j \in \mathcal{I}_\nu$. This operation is standard. The $B$-values have to be updated only in the nodes along the path from the root to the nodes where the $A$-values have been modified. Since those nodes lie along the two paths from the root to the elementary intervals storing $v_j - h$ and $v_j$, this can be done in $O(\log n)$ time overall.

LEMMA 6.1. *The start-stop matrix can be computed in $O(n^2 \log n)$ time.*

*Proof.* We fill in the matrix $B$ by testing the validity of subsequences of $S$. A single column of $B$ corresponds to testing the validity of $S_{ii}, S_{i(i+1)}, \ldots, S_{in}$. This can be done in the given order (bottom-up in a column) in $O(n \log n)$ time by using the data structure described above.

Assume that we have a tree $T$ representing $\psi_{ij}$ and that we have determined whether $S_{ij}$ is valid. Then we insert $I_{j+1}$ with weight $w_{j+1}$ in the augmented tree, and perform a query to determine $\max_x \psi_{i(j+1)}(x)$. We also compute $\lambda_{i(j+1)}$ from $\lambda_{ij}$ by adding $w_{j+1}$. Now the test $\max_x \psi_{i(j+1)}(x)/\lambda_{i(j+1)} \geq \rho$ determines whether $S_{i(j+1)}$ is valid or not. □

We then use the algorithm from Theorem 2.1, and conclude:

THEOREM 6.1. *Given a piecewise-constant function $f$ with $n$ breakpoints, a threshold value $h > 0$, and a ratio $\rho \in [0, 1]$, we can compute a minimal discrete segmentation for the condition that, on a fraction of length at least $\rho$ of a segment, the difference between the maximum and minimum function value is at most $h$, in $O(n^2 \log n)$ time.*

### 6.1.2 Continuous case

For continuous segmentation, we are allowed to cut $f$ at any two points $a, b \in [0,1]$. So we need to determine the forbidden space in the start-stop diagram. The breakpoints of the function $f$ decompose the start-stop diagram in a grid. We now prove that the forbidden space within a cell $C = [\underline{a}, \overline{a}] \times [\underline{b}, \overline{b}]$ of this grid can be described by four linear inequalities of the following form, where $\psi_1^*, \ldots, \psi_4^*$ are constant values specific to the cell.

$$(b-a)\rho > \begin{cases} \psi_1^* + (\overline{a} - a) + (b - \underline{b}) \\ \psi_2^* + (\overline{a} - a) \\ \psi_3^* + (b - \underline{b}) \\ \psi_4^* \end{cases} \tag{6.3}$$

LEMMA 6.2. *For any cell $C = [\underline{a}, \overline{a}] \times [\underline{b}, \overline{b}]$ of the start-stop diagram, there exist values of $\psi_1^*, \ldots, \psi_4^*$, such that the forbidden space in $C$ is the intersection of four halfplanes with the cell, described by the inequalities in (6.3).*

*Proof.* We use an approach similar to that used in the previous section. Let

$$\chi^{\leq}(x, a, b) = \big|\{t \mid t \in [a, b] \wedge f(t) \leq x\}\big|$$

denote the length of the portion of $[a, b]$ on which the value of $f$ is at most $x$, let $\chi^{<}$ be defined analogously, and let

$$\psi(x, a, b) = \chi^{\leq}(x + h, a, b) - \chi^{<}(x, a, b). \tag{6.4}$$

For any point $(a, b) \in [\underline{a}, \overline{a}] \times [\underline{b}, \overline{b}]$, the corresponding segment $[a, b]$ is invalid if and only if $\max_x \psi(x, a, b) < (b - a)\rho$, i.e. for all possible intervals $I = [x, x + h]$, the fraction of $[a, b]$ on which the function value lies in $I$ is smaller than $\rho$.

Note that the candidate segment corresponding to $(a, b)$ contains the segment defined by $(\overline{a}, \underline{b})$ and let $\psi_C$ be the function $\psi$ restricted to the cell $C$. We can rewrite $\psi_C$ with respect to $\psi(x, \overline{a}, \underline{b})$ as follows. Within the cell, decreasing $a$ by some value $\Delta_a$ corresponds to raising the functions $\chi^{\leq}$ and $\chi^{<}$ by $\Delta_a$ for all input values which are greater or equal to $f(a)$. By the definition of $\psi$ in Equation (6.4) above, this implies that $\psi_C$ increases by $\Delta_a$ only on the interval $I_a = [f(a) - h, f(a)]$. Similarly, increasing $b$ by $\Delta_b$ results in increasing $\psi_C$ by $\Delta_b$ on $I_b = [f(b) - h, f(b)]$. Letting $\Delta_a = \overline{a} - a$ and $\Delta_b = b - \underline{b}$, we can rewrite $\psi_C$ as follows:

$$\psi_C(x, a, b) = \begin{cases} \psi(x, \overline{a}, \underline{b}) + \Delta_a + \Delta_b & \text{if } x \in I_a \cap I_b \\ \psi(x, \overline{a}, \underline{b}) + \Delta_a & \text{if } x \in I_a \setminus I_b \\ \psi(x, \overline{a}, \underline{b}) + \Delta_b & \text{if } x \in I_b \setminus I_a \\ \psi(x, \overline{a}, \underline{b}) & \text{otherwise} \end{cases} \tag{6.5}$$

For each of the above cases, the maximum of the function over $x$ depends on $\psi(x, \overline{a}, \underline{b})$. For each maxima, $\psi(x, \overline{a}, \underline{b})$ is a value no longer depending on $x$, $a$, or $b$. Let these values (constants) be $\psi_1^*, \ldots, \psi_4^*$. It follows that each of the above cases yields a linear inequality in $a$ and $b$. The values of $(a, b)$ for which each of these inequalities holds true form the forbidden space. Thus, the forbidden space in this cell can be described by the intersection of the corresponding four halfplanes. □

LEMMA 6.3. *The start-stop diagram can be computed in $O(n^2 \log n)$ time.*

*Proof.* As in the discrete case, we can represent the function $\psi_{ij} = \psi(\cdot, \overline{a}, \underline{b})$ in a data structure, where $(\overline{a}, \underline{b})$ is the lower right corner of the current cell. Again, we maintain this data structure while traversing the cells of the grid bottom up within a column and reinitialize it for every column. We use exactly the same data structure as before. By Lemma 6.2, the forbidden space in a cell is described by Equation (6.3). We now need four queries to compute the values of $\psi_1^*, \ldots, \psi_4^*$, since these are the maxima of the function $\psi_{ij}$ on the intervals in Equation (6.5). And hence, we get the free space of a cell $C$. This means we can compute the start-stop diagram in $O(n^2 \log n)$ time. □

Clearly, Lemma 6.2 implies the forbidden space within each cell of the start-stop diagram is verticonvex and has constant complexity, so we can invoke Theorem 3.2. We conclude:

THEOREM 6.2. *Given a piecewise-constant function $f$ with $n$ breakpoints, a threshold value $h > 0$, and a ratio $\rho \in [0, 1]$, we can compute a minimal segmentation for the condition that on a fraction of at least $\rho$ of a segment, the difference between the maximum and minimum function value is at most $h$ in $O(n^2 \log n + kn^2)$ time, where $k$ is the size of a minimal segmentation.*

**Piecewise-linear functions.** If the attribute function $f$ is piecewise linear, then the grid induced by the breakpoints of $f$ may contain (many) tunnel cells. Assume that we want to segment the attribute function $f$

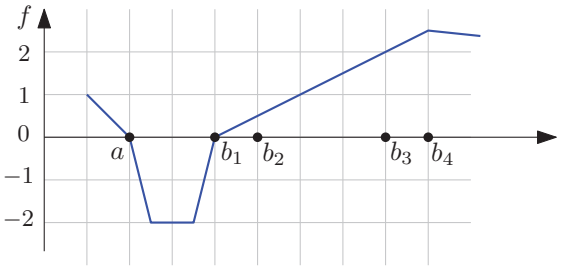


Figure 10: A piecewise-linear attribute function that leads to tunnel cells.

depicted in Figure 10 on the outlier-tolerant criterion with $h = 2 + \varepsilon$, for some $0 \leq \varepsilon \leq 0.1$ and $\rho = 2/3$, i.e. the difference between any two values is not more than approximately 2 but we can disregard 1/3 of the segment. For this criterion, the candidate segments $[a, b_2]$ and $[a, b_3]$ are valid, as well as any candidate segment $[a, b]$ for $b_1 < b < b_2$. However, there exist candidate segments $[a, b]$ for $b_2 < b < b_3$ and for $b_3 < b < b_4$ which are not valid. Since $b_1$, $b_2$, $b_3$, and $b_4$ lie on the same piece of the function $f$, this implies that the vertical line at $a$ intersects the forbidden space in this cell twice, once below $b_3$ and once above $b_3$.

It is now easy to see that we can create more than one tunnel cell per row: we simply use the above construction with points $a^- = a - \delta$ and $a^+ = a + \delta$, for some arbitrarily small $\delta$, as starting point of the segments. Since $a$ is a break point these points lie on different pieces of $f$, and hence we get two tunnel cells. In both cases the end points of these segments lie on the piece $[b_1, b_4]$, so the cells lie in the same row.

This example suggests that solving the problem for piecewise-linear attribute functions efficiently calls for a different approach.

## 6.2 The standard deviation criterion

Another important non-monotone criterion that we consider involves the standard deviation of an attribute function. In this section we show how to compute a minimal segmentation of a piecewise-constant function $f$ where each segment has standard deviation not exceeding a given threshold value. Let $\mu(a, b) = \int_a^b f(y)\,\mathrm{d}y/(b - a)$ denote the mean value on a candidate segment $[a, b]$. The standard deviation $\sigma(a, b)$ is given by

$$\sigma(a, b) = \sqrt{\frac{\int_a^b (f(x) - \mu(a, b))^2\,\mathrm{d}x}{b - a}}\,.$$

### 6.2.1 Discrete case

In the discrete segmentation problem, we are allowed to partition $f$ only where its value changes. Recall that $f$ is given as a sequence $S = s_1, \ldots, s_n$ of pairs $s_i = (v_i, w_i)$, where $v_i$ is the value of piece $i$ and $w_i$ is its weight.

LEMMA 6.4. *The start-stop matrix can be computed in $O(n^2)$ time.*

*Proof.* To compute the start-stop matrix $B$, we need to test whether the standard deviation of a segment is below the allowed threshold. We could compute this in time linear in the length of a segment. However, we can also maintain the mean $\mu_{ij}$ and standard deviation $\sigma_{ij}$ of a weighted sequence $S_{ij}$. We can then compute the mean $\mu_{i(j+1)}$ and the standard deviation $\sigma_{i(j+1)}$ for $S_{i(j+1)}$ in constant time using $\mu_{ij}$ and $\sigma_{ij}$. This implies that we can fill $B$ in constant time per cell and thus quadratic time in total. □

Once we have $B$ we can compute a minimal segmentation in $O(n^2)$ time (Theorem 2.1). Hence:

THEOREM 6.3. *For any value $h > 0$, a minimal discrete segmentation where each segment has standard deviation at most $h$ can be computed in $O(n^2)$ time.*

### 6.2.2 Continuous case

In the continuous case, $a$ and $b$ can have any real value in $I$. Setting the standard deviation $\sigma(a, b)$ equal to the constant threshold value $h$, we obtain the functional description of the boundaries of the forbidden space in the start-stop diagram:

$$\sigma(a, b) = h \iff \int_a^b (f(x) - \mu(a, b))^2\,\mathrm{d}x = (b - a) \cdot h^2.$$

Further algebraic manipulations, using the fact that $f$ is piecewise constant, give a cubic expression in $a$ and $b$. Hence, the boundaries of the forbidden space within each cell of the start-stop diagram are piecewise-cubic curves. This allows us to prove the following lemma:

LEMMA 6.5. *Every cell of the start-stop diagram is verticonvex.*

*Proof.* On a vertical line, the start point of the candidate segment is fixed. Let it be $\tilde{a}$. Inside a single cell, the stop point has a single constant function value $f(b) = c$. Therefore, varying $b$ implies including more or less of $c$ in the values whose standard deviation is considered.

Imagine $b$ is at its low end of the cell, and let it increase to its high end. Then the mean $\mu(\tilde{a}, b)$ tends monotonically towards $c$. There are four cases to distinguish:

1. Initially $\sigma(\tilde{a}, b) < h$ and $|\mu(\tilde{a}, b) - c| < h$. Then the whole intersection of the cell and the vertical line is allowed.
2. Initially $\sigma(\tilde{a}, b) > h$ and $|\mu(\tilde{a}, b) - c| > h$. Then the lower end of the intersection is forbidden, but possibly, at some value of $b$ it becomes allowed.
3. Initially $\sigma(\tilde{a}, b) < h$ and $|\mu(\tilde{a}, b) - c| > h$. Then the lower end of the intersection is allowed, but possibly, it becomes forbidden and possibly later allowed again.
4. Initially $\sigma(\tilde{a}, b) > h$ and $|\mu(\tilde{a}, b) - c| < h$. Then the lower end of the intersection is forbidden, but possibly, at some value of $b$ it becomes allowed.

In other words, we have the property that if for some $b$ the candidate segment becomes allowed, then it stays allowed. This is true for the following reason. At the value $\tilde{b}$ of $b$ when $[\tilde{a}, b]$ becomes allowed, we have $\sigma(\tilde{a}, \tilde{b}) = h$ and $|\mu(\tilde{a}, \tilde{b}) - c| < h$. For increasing $b$ the average $\mu(\tilde{a}, b)$ will tend monotonically towards $c$. So for larger $b$ we have $|\mu(\tilde{a}, b) - c| < h$, and therefore $\sigma(\tilde{a}, b) < h$. This proves the lemma. □

LEMMA 6.6. *The start-stop diagram can be computed in $O(n^2)$ time.*

*Proof.* The forbidden space in each cell has constant complexity, and given the description of $\sigma$ for cell $C$ we can compute $\sigma$ for neighbors of $C$ in $O(1)$ time. □

Using Theorem 3.2, we then obtain:

THEOREM 6.4. *Given a piecewise-constant function $f$ with $n$ breakpoints and a threshold value $h > 0$, we can compute a minimal segmentation for the criterion that the standard deviation of a segment is at most $h$ in $O(kn^2)$ time, where $k$ is the size of a minimal segmentation.*

**Piecewise-linear functions.** As with the outlier-tolerant criterion, if $f$ is piecewise linear then the grid decomposition of the start-stop diagram induced by the breakpoints of $f$ may contain (many) tunnel cells. Consider for example the function $f$ depicted in Figure 11. For both the candidate segments $[a, b_2]$ and $[a, b_4]$ the mean is close to 0.5 and the standard deviation is close to 0.25. However, there exists a value $b_2 < b < b_4$, such that the mean and the standard deviation of $[a, b]$ are smaller than 0.25. For example in Figure 11 the standard deviation at $b_3$ is below 0.2. So if we choose $h = 0.25$ the cell $[0, b_1] \times [b_1, b_5]$ is a tunnel cell: the vertical line at $a$ intersects the forbidden space twice, once below $b_3$ and once above it. We can extend this example to construct two tunnel cells as before. Therefore, we do not expect that there exists a straightforward extension of our algorithm to this case.

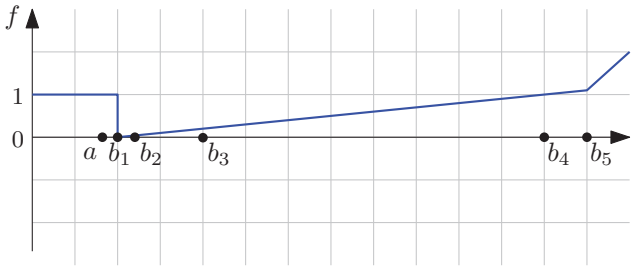


Figure 11: A piecewise-linear attribute function that leads to tunnel cells.

## 7 Conclusions

In this paper we analyzed the problem of segmenting a trajectory for non-monotone criteria. For monotone criteria near-linear time algorithms are known [4]. We showed that also for certain non-monotone criteria polynomial-time solutions exist. In particular, our approach uses the start-stop diagram and we identify properties of this diagram that make the problem tractable. Furthermore, we proved that the abstract segmentation problem based on this diagram is NP-hard.

For two concrete non-monotone criteria we presented efficient algorithms to compute the start-stop diagram. We also showed that the resulting diagrams indeed have the properties that allow for efficient segmentation. As a result, we can compute an optimal segmentation of a trajectory based on the outlier-tolerant criterion in $O(n^2 \log n + kn^2)$ time, and on the standard deviation criterion in $O(kn^2)$ time, where $n$ is the number of vertices of the input trajectory and $k$ is the number of segments in an optimal solution. These two criteria are relevant in practice, since they are more robust against noise than related monotone criteria.

A complete characterization of the start-stop diagrams that allow for efficient segmentation is still open. We started with this characterization, but more can perhaps be said. Furthermore, the analyses of the running

times for the outlier-tolerant criterion and the standard deviation criterion are based on the assumption that the attribute function is piecewise constant. While many second-order attributes, such as speed and heading, are piecewise constant due to the linear interpolation of the trajectory, it would also be interesting to see whether similar results can be obtained for, e.g., piecewise-linear functions. The examples given in Sections 6.1 and 6.2 seem to indicate that the methods developed in this paper do not immediately apply if the attribute functions are piecewise linear.

Finally, our two-step approach of first computing the start-stop diagram, and then solving the segmentation problem, inherently requires at least $O(n^2)$ time and space. An intriguing open problem is whether a subquadratic solution may be possible.

## Acknowledgments

Work on this paper has been partially supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.021.123, 612.001.022, and 612.065.823, and by EU Cost Action IC0903 (MOVE). Work on this paper by B.A. has been partially supported by NSA MSP Grant H98230-10-1-0210 and by NSF Grants CCF 08-30691 and CCF 11-17336.

## References

[1] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geometry Appl.*, 5:75–91, 1995.

[2] A. Anagnostopoulos, M. Vlachos, M. Hadjieleftheriou, E. Keogh, and P. Yu. Global distance-based segmentation of trajectories. In *Proc. 12th Conference on Knowledge Discovery and Data Mining*, pages 34–43, 2006.

[3] P. Bovet and S. Benhamou. Spatial analysis of animals' movements using a correlated random walk model. *J. Theoretical Biology*, 131(4):419–433, 1988.

[4] M. Buchin, A. Driemel, M. J. van Kreveld, and V. Sacristan. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *J. Spatial Information Science*, 3(1):33–63, 2011.

[5] C. Calenge, S. Dray, and M. Royer-Carenzi. The concept of animals' trajectories from a data analysis perspective. *Ecological Informatics*, 4(1):34 – 41, 2009.

[6] P. Chundi and D. J. Rosenkrantz. Segmentation of time series data. In J. Wang, editor, *Encyclopedia of Data Warehousing and Mining*, pages 1753–1758. 2009.

[7] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill, 2008.

[8] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[9] S. Dodge, R. Weibel, and E. Forootan. Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. *Computers, Environment and Urban Systems*, 33(6):419–434, 2009.

[10] K. Fu and J. Mui. A survey on image segmentation. *Pattern Recognition*, 13(1):3–16, 1981.

[11] E. Gurarie, R. D. Andrews, and K. L. Laidre. A novel method for identifying behavioural changes in animal movement data. *Ecology Letters*, 12(5):395–408, 2009.

[12] X. Li, X. Li, D. Tang, and X. Xu. Deriving features of traffic flow around an intersection from trajectories of vehicles. In *Proc. 18th International Conference on Geoinformatics*, pages 1–5. IEEE, 2010.

[13] R. Mann, A. Jepson, and T. El-Maraghi. Trajectory segmentation using dynamic programming. In *Proc. 16th International Conference on Pattern Recognition (ICPR)*, volume 1, pages 331–334, 2002.

[14] R. Nathan, W. Getz, E. Revilla, M. Holyoak, R. Kadmon, D. Saltz, and P. Smouse. A movement ecology paradigm for unifying organismal movement research. *Proc. National Academy of Sciences*, 105:19052–19059, 2008.

[15] A. Stohl. Computation, accuracy and applications of trajectories – a review and bibliography. *Atmospheric Environment*, 32(6):947 – 966, 1998.

[16] E. Terzi and P. Tsaparas. Efficient algorithms for sequence segmentation. In *Proc. 6th SIAM International Conference on Data Mining*, pages 314–325, 2006.

[17] B. van Moorter, D. R. Visscher, C. L. Jerde, J. L. Frair, and E. H. Merrill. Identifying movement states from location data using cluster analysis. *J. Wildlife Management*, 74(3):588–594, 2010.

[18] X. Zhou, S. Shekhar, P. Mohan, S. Liess, and P. K. Snyder. Discovering interesting sub-paths in spatiotemporal datasets: A summary of results. In *Proc. 19th ACM SIGSPATIAL Conference on Advances in Geographic Information Systems*, pages 44–53, 2011.