JOURNAL OF SPATIAL INFORMATION SCIENCE
Number 3 (2011), pp. 33–63 doi:10.5311/JOSIS.2011.3.66

RESEARCH ARTICLE

Segmenting trajectories: A framework and algorithms using spatiotemporal criteria*

Maike Buchin[1], Anne Driemel[2], Marc van Kreveld[3], and Vera Sacristán[4]

[1]Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands
[2]Dept. of Information and Computing Sciences, Utrecht University, The Netherlands
[3]Dept. of Information and Computing Sciences, Utrecht University, The Netherlands
[4]Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain

Received: July 19, 2011; returned: October 10, 2011; revised: November 16, 2011; accepted: December 10, 2011.

Abstract: In this paper we address the problem of segmenting a trajectory based on spatiotemporal criteria. We require that each segment is homogeneous in the sense that a set of spatiotemporal criteria are fulfilled. We define different such criteria, including location, heading, speed, velocity, curvature, sinuosity, curviness, and shape. We present an algorithmic framework that allows us to segment any trajectory into a minimum number of segments under any of these criteria, or any combination of these criteria. In this framework, a segmentation can generally be computed in $O(n \log n)$ time, where n is the number of edges of the trajectory to be segmented. We also discuss the robustness of our approach.

Keywords: spatial and spatiotemporal information systems, computational geometry, moving objects analysis, trajectory analysis, segmentation

1 Introduction

Due to technologies such as GPS tags, trajectories are collected on a large scale. A trajectory represents the locations of a moving object over a certain time interval. Typically, a trajectory is collected by recording the location at a number of moments in time. Such a

*This article is a revised and expanded version of a paper that appeared at ACM SIGSPATIAL 2010 [9].

sequence of points with time stamps can then be interpreted as a continuous motion path of a moving object by some form of interpolation. The recording of locations can be done at regular or irregular intervals. Due to noise in the GPS data certain locations can be unreliable or missing. In some applications the recording is speed-dependent, with higher sampling rates if the moving object is faster.

With the existence of large collections of trajectory data, the analysis of such data has also taken a flight. Examples are detecting flocking behavior in trajectories of animals [7], analyzing the trajectories of several shoppers in a supermarket [25], and determining commuting patterns in the trajectories of people [8].

The analysis task we discuss in this paper is *segmenting* a trajectory. The segmentation problem for a trajectory is to partition it into a (typically small) number of pieces, which are called *segments*. (Note that *segment* refers to *subtrajectory* and not the mathematical *line segment*.) The idea of segmentation is to obtain segments where movement characteristics inside each segment are uniform in some sense. Movement characteristics are for example, speed, heading, or curviness, or any combination of such characteristics. Segmentation aids in understanding the behavior of animals from animal trajectories, it helps to find and analyze patterns in movement of sports players, and is important for content-based motion retrieval tasks.

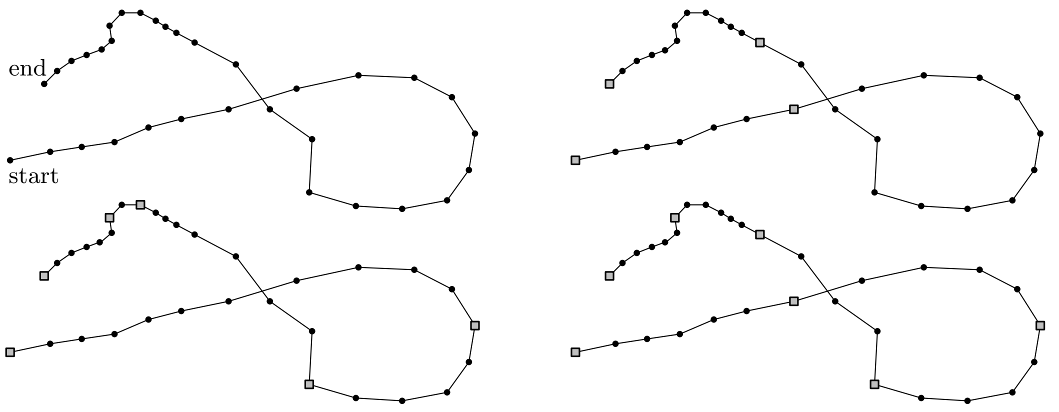

Figure 1: A trajectory and three segmentations of it: cutting points are indicated by squares.

Figure 1 illustrates the segmentation problem using different criteria. To the top left, a possible trajectory is given, based on points sampled with equal time intervals. This means that the length of each edge is proportional to the (average) speed along that edge. The heading along an edge is the direction from the earlier endpoint to the later endpoint. To the top right, a segmentation is shown where the criterion is that within each segment, the speed cannot differ more than a factor 2. Hence, a segment cannot contain two edges of which one is more than twice as long as the other. To the bottom left, a segmentation by heading is shown, where the criterion is that within each segment, the direction of motion differs by at most $90°$. To the bottom right, a segmentation is shown where both criteria are used in conjunction. In all three segmentations, the number of segments used is minimum (3 for speed, 5 for heading, and 6 for speed and heading simultaneously).

Related work Segmentation is generally understood as the partitioning of data into homogeneous, possibly meaningful pieces. This is true for segmentation in image processing [16, 34] and DNA sequence segmentation [6], for instance. In image processing, segmentation is the partitioning of a digital image into regions of pixels by similarity in color and intensity. These regions are often meaningful parts of the image, and therefore segmentation can help in image analysis.

Segmentation is also common in time-series analysis [2, 10, 19]. Again the objective is to partition the data into homogeneous pieces. The objective may be for compact or high-level representation, for reduction of noise, and eventually for understanding or prediction of the behavior of the time-series data. These papers treat segmentation as a simplification or clustering problem, where a segment basically is a continuous section of the time series (the trajectory) that could be replaced by a single edge or median point in the resulting segmentation. The choice of the error criterion determines in which sense the characteristics of the input are kept. It is also common to assume that the number of segments to be used is given, and a global error function should be optimized [27, 35]. The most common techniques used for segmentation of time-series data include curve fitting [27], dynamic programming [22, 35], and various heuristics [2, 40]. The running time is generally $O(n^2k)$, where n is the length of the time series and k is the maximal number of segments in the output, see also [3].

For the processing of geographic trajectories of moving objects, segmentation has also been studied [15, 40]. The objectives are generally the same as for time-series data segmentation. Some papers discuss segmentation of trajectories for the purpose of high-level representation, or semantic annotation [19, 42]. Here the goal is it also to have as much homogeneity within each segment as possible while using only few segments in total. Alternatively, a given set of templates are used to match to parts of the trajectory to realize semantic annotation. These papers focus more on the semantics of spatiotemporal trajectories.

Our approach to segmentation Corresponding to the most common view of segmentation, in this paper we view trajectory segmentation as the problem of partitioning a trajectory into parts (segments) that are sufficiently homogeneous. We formalize this as follows. For each relevant movement characteristic we define an *attribute function* that specifies a value at every point in time, where the trajectory is defined. For instance, attributes can be speed, heading, and curvature. Then we define *criteria* that specify that within any single segment, the attribute values at all points within a segment are sufficiently similar. For instance, the speeds should be at most 30% different within each segment. This implies that we will have a guaranteed similarity of each incorporated attribute within each segment. Within the limitations imposed by requiring similar attribute values, we minimize the number of segments used in the segmentation. Interestingly, this optimization problem can be solved efficiently in many situations. Our algorithmic framework shows that this is the case: optimal segmentation requires at most $O(n \log n)$ time for a trajectory consisting of n edges if we use simple criteria relating to speed, heading, location, curvature, sinuosity, curviness, or any combination of these. One of the main advantages of our approach is that we can compute an optimal segmentation using multiple criteria at once.

Application areas Segmentation is used as an important step in a variety of applications. Not surprisingly, the desired properties requested towards a segmentation often depend on the application domain and the exact problem at hand. We mention a few application scenarios that we consider relevant to our setting.

Bird ecologists study recurrent patterns and behavioral changes in animal tracking data to explain the individual's behavior during foraging or migration [18, 37]. It is common to split and annotate the trajectories into distinct segments, depending on the birds activity, e.g., directional flight, soaring, circling, etc. Often, this task is performed manually by a domain expert, but the process could be automated using a segmentation algorithm, given a formal definition of these activity states, as in [36] for example. In fact, a common approach in animal movement analysis assumes that the individual movement is a response to a combination of internal states, physiological constraints, and environmental factors [18, 23, 24], which can be modeled using a state-space framework [30]. For many models, the information that defines a particular state will be encoded in the spatiotemporal trajectory or is available through context data. The different states of a process can be defined manually using domain knowledge, or sometimes they can be identified using cluster analysis [37].

State-space models are an important tool used in many research areas. Scientists study the trajectories generated by such a model to find out more about the underlying dynamic system. In order to simplify this analysis, the trajectories are sometimes preprocessed into semantic label sequences, a technique which is called symbolic time-series analysis [33, 39]. We think that segmentation can help in this analysis by producing more sophisticated label sequences.

Another application is the task of context recognition for mobile phone applications. Modern mobile devices carry sensors for acceleration, noise level, luminosity, humidity, etc. Online segmentation algorithms for the time series composed of this data can help to adapt the user interface and increase the usability [22].

Finally, the solutions to the segmentation problem described in this paper can be used to detect outliers. An outlier can be considered noise or relevant information. In both cases it is desirable to detect it in a trajectory. An outlier can (arguably) be defined as a short section on the trajectory that represents a behavior which deviates from its context, i.e., before and after this section. If we can identify certain attributes which have to be homogeneous along the noise-free parts of the trajectory, then segmentation based on these attributes will reveal the outliers as very short segments.

Overview of this paper In Section 2 we define a trajectory and the interpretation of it as a representation of the motion path of a moving object. We also define the trajectory segmentation problem and when criteria are *monotone*. In Section 3 we discuss the basic attributes location, heading, speed, and velocity, and which criteria for them can be used for segmentation. In Section 4 we present an algorithmic framework that allows us to efficiently segment according to a criterion provided that two basic procedures can be given, and the criterion is *monotone*. We prove that for any monotone criterion we can use a greedy strategy for segmentation to obtain an optimal solution, and hence we can avoid dynamic programming. In Section 5 we show that for simple criteria relating to location, heading, speed, and velocity, efficient algorithms exist to implement the basic functions. Using the results of the previous section this immediately implies efficient algorithms for segmentation on such a criterion. We proceed to show that multiple criteria can be combined using conjunctions, disjunctions, or linear combinations within our framework. The segmentation according to these combined criteria remains optimal and is equally efficient.

In Section 6 we consider more complex attributes like curvature, sinuosity, and curviness, and possible criteria for them. We will show that segmenting optimally and efficiently on these criteria is possible as well, with the same approach. In Section 7 we show that shape-fitting criteria can also be used in our framework. These criteria do not result in homogeneity of some attribute value within each segment, but achieve homogeneity in a different manner. In Section 8 we discuss robustness in relation to segmentation, and show that it can be handled on different levels.

Additional material in this version Part of this work has been published previously in a conference proceedings version, see [9]. The additional material provided in the current version can be summarized as follows. We included sections which were previously omitted due to space requirements and we give full proofs to all the claims that were posed in the previous version together with an extended discussion at several places. In particular, the paragraph on application areas in Section 1 has been added in this version and a section on robustness, see Section 8. We give mathematical formulations of different ways to combine criteria and show how they can be integrated in the framework, see Section 5.4. Furthermore, we discuss a completely new type of criterion based on shape in Section 7.

2 Preliminaries

We define two types of trajectories, discrete and continuous (piecewise-linear). In both cases, the representation of the trajectory usually follows from the way it is collected: a discrete sample of time-space positions. A piecewise-linear trajectory is a continuous motion path, while a discrete trajectory is defined only at a series of discrete points in time.

Definition 1. *A* discrete trajectory *τ is a mapping from a series of time stamps $t_0, t_1, \ldots, t_n$ to the plane (or a higher-dimensional space). For any time stamp t_i, we denote the location in the plane at time t_i by $\tau(t_i)$. For any two times $t_i, t_j \in \{t_0, \ldots, t_n\}$ with $t_i \leq t_j$, we denote the subtrajectory of τ from time t_i to time t_j by $\tau[t_i, t_j]$.*

Definition 2. *A* piecewise-linear trajectory *τ is a continuous mapping from a time interval $[t_0, t_n]$ to the plane (or a higher-dimensional space). For some sequence of time steps $t_0 < t_1 < \cdots < t_n$, the locations at these times are given as $v_i = \tau(t_i)$ for $0 \leq i \leq n$, and $v_0, v_1, \ldots, v_n$ are the* vertices *of τ. The location $\tau(t)$ for $t \in [t_i, t_{i+1}]$ is the linear interpolation over time of $\tau(t_i)$ and $\tau(t_{i+1})$, that is, the point $\frac{t_{i+1}-t}{t_{i+1}-t_i} \cdot \tau(t_i) + \frac{t-t_i}{t_{i+1}-t_i} \cdot \tau(t_{i+1})$. We define $e_i = \overline{v_{i-1}v_i}$ for $1 \leq i \leq n$ to be the* edges *of τ. For any two times $t, t' \in [t_0, t_n]$ with $t \leq t'$, we denote the subtrajectory of τ from time t to time t' by $\tau[t, t']$.*

Note that a subtrajectory of a discrete trajectory is a discrete trajectory itself, and the same holds true for a piecewise-linear trajectory.

For a discrete trajectory one does not wish to assume any location between the known, measured locations at the time stamps $t_0, \ldots, t_n$. This may be due to under-sampling of the trajectory: locations would be unreliable and therefore not so useful. In practice this can happen, for example, when tracking migrating birds over long distances and time stretches. It is common practice to choose a very low tracking frequency in case of a limited energy supply of the tracking devices, which are carried by the birds.

If we want to compute a segmentation of a discrete trajectory based on attributes like speed or heading, we must be provided with such data at the time stamps $t_0, \ldots, t_n$. For example, heading information may be available if a GPS measurement and compass reading is done at each time stamp, and speed at a time stamp may be obtained by two GPS measurements in quick succession. Otherwise, we must compute an attribute at a time stamp t_i by using the known locations at times t_{i-1}, t_i, and t_{i+1}.

For a piecewise-linear trajectory we do assume a location at each time in $[t_0, t_n]$. Attributes of the piecewise-linear trajectory can be defined, like speed, heading, curvature, etc. Location can also be seen as an attribute that has two (or three) coordinates: at any time t, the location is given by $\tau(t)$. Since our trajectories are piecewise-linear, and location is assumed to be linearly interpolated between $\tau(t_i)$ and $\tau(t_{i+1})$, the attribute speed is constant on every edge of the trajectory. At vertices, it changes to a new speed value. The same is true for the attribute heading. Attributes like location and curvature (for a suitable definition, given later in this paper) are not constant on edges of the trajectory.

In this paper we are mostly concerned with segmentation of piecewise-linear trajectories, and we will use the term "trajectory" for any piecewise-linear trajectory. Our solutions to the segmentation problem can be applied to the discrete setting with minor modifications. In general, the problem is easier for discrete trajectories. When sampling is sufficiently dense for the type of data, piecewise-linear trajectories are in line with the nature of movement, because a moving object has its location change in a continuous manner.

2.1 The segmentation problem

A *segmentation* of a trajectory is a partitioning into a number of parts called segments. The idea of a segmentation is that each segment satisfies certain criteria. A segmentation is *optimal* if it is a partitioning into a minimum number of segments while satisfying the criteria.

We distinguish between *discrete* and *continuous* segmentation. Discrete segmentation is the partitioning of a discrete trajectory into segments, where a segment consist of one or more consecutive time stamps. Continuous segmentation is the partitioning of a continuous (piecewise-linear) trajectory into segments, where a segment consists of a subtrajectory which can start and end anywhere on edges. The segments have disjoint interiors, and the end of one segment is the start of the next segment. For example, $\tau[t_0, t], \tau[t, t'], \tau[t', t_n]$ is a segmentation of $\tau[t_0, t_n]$ into three segments if $0 < t < t' < t_n$.

Continuous segmentation is algorithmically more challenging than discrete segmentation. Still we will show that continuous segmentation can be solved efficiently with a relatively simple approach. The remainder of this paper is mainly concerned with continuous segmentation, but our results can be adapted (in fact, simplified) to discrete segmentation.

2.2 Relative versus absolute criteria

Criteria typically concern attributes of the trajectory that are defined at any time t, and bounds on the values of these. An example of a relative criterion is a bound on the standard deviation of the values of an attribute within a subtrajectory, whereas an upper and a lower bound on the mean or average value is an absolute criterion. Note that, consequently, absolute criteria are not always defined. However, it is always possible to segment using relative criteria.

Observe that using *absolute* boundary values, like $10\,\mathrm{km/h}$, $20\,\mathrm{km/h}$, $50\,\mathrm{km/h}$, and $100\,\mathrm{km/h}$ for speed might lead to *oversegmentation*. Using these absolute values a trajectory that has speeds 51–51–98–98–103–103 km/h along its edges is segmented into more pieces than desired. Also, it would segment a trajectory with speeds 49–51–49–51 km/h along its edges into four segments. Hence, we will only use criteria based on *relative values within a segment* in this paper.

Another illustrative example can be given using the location attribute. An absolute criterion for location would categorize the positions of the trajectory into zones with fixed boundaries. One of the known issues with this approach is the so-called *modifiable area unit problem* (MAUP) [12]. A relative criterion is described in Section 3. Here, we say the criterion is satisfied for a given subtrajectory, if the positions on this subtrajectory fit into any disk of a given radius, which can be centered anywhere in space.

2.3 Monotone criteria

Definition 3. *A criterion is* monotone *if for any subtrajectory $\tau' \subseteq \tau$, we have that if τ' satisfies the criterion, then any subtrajectory $\tau'' \subseteq \tau'$ also satisfies the criterion.*

Monotonicity is a requirement which guarantees the soundness of the segmentation. In classical image processing, for instance, the segmentation problem is modeled using monotone criteria [16, 34].

The monotonicity condition implies that if we have any subtrajectory for which the criterion is not satisfied, then extending the subtrajectory cannot satisfy the criterion. There are many examples of criteria that are monotone. In particular, criteria that bound the *range* (or maximum extent) of an attribute are always monotone. The range can be bounded by bounding the difference of the extreme values (for example, the difference in minimum and maximum speed is at most 20 km/h), or by bounding the ratio of the extreme values (for example, the maximum speed is at most 1.5 times as high as the minimum speed).

An example of a criterion that is *not monotone* is specifying that the standard deviation of some attribute is below a given value in each segment. It is possible that the standard deviation decreases by extending a segment. Another criterion which is not monotone is described in Section 8: if we allow the criterion to not be met for short durations, we obtain a criterion that is more robust against noise, but it is not monotone if these durations are relative to the length of the segment. By extending the segment, the tolerance duration is also extended and therefore the criterion which was not satisfied before, may become satisfied.

In Section 5.4 we show that there are simple ways of combining criteria, such as conjunctions, disjunctions, and linear combinations, which preserve the monotonicity. In Section 7 and Section 8 we describe monotone criteria that are robust against noise.

3 Basic spatiotemporal attributes and criteria

Next, we discuss some specific, basic attributes and possible criteria on them that can be used for segmentation. These attributes are location, heading, and speed. We also discuss velocity, the combination of heading and speed. For all, straightforward criteria exist that are monotone. Furthermore, the criteria that we give do not use fixed, absolute values of the attribute to bound segments, and hence they avoid oversegmentation.

Location The location of a point on a trajectory is given by its coordinates in a spatial reference system. Since trajectories are continuous motion paths of objects, the location changes continuously over time, along the trajectory. We give two criteria based on location. Both will—intuitively—bound the distance between points within one segment. One criterion based on location states that each location in a segment in a segmentation should be within a fixed-size neighborhood of some (unspecified) location. Geometrically, for any segment, a disk of fixed size must exist that covers the locations of that segment completely. Clearly, the *disk criterion* is monotone: if some subtrajectory can be covered by a fixed-size disk, then any part of it can also be covered by such a disk (namely, by the same disk). Alternatively, we could impose the criterion that no two points within one segment are more than some given distance apart. Geometrically, we bound the diameter within a segment's locations. The *diameter criterion* is also monotone.

Heading The heading at any moment in time is the direction in which the moving object is traveling at that moment. It can be specified by an angle in the range $[0, 2\pi)$ according to some reference system, or it is UNDEFINED if the object is not moving. For example, purely north has angle 0 and purely east has angle $+\pi/2$. For trajectories represented by vertices and edges, the heading has a straightforward definition on each edge, but not at vertices. To define heading at a vertex, we can for instance choose it to be the same as the heading on the edge just before or after the vertex. We should choose it to make sure that no segment consists of a single vertex.

A straightforward criterion for heading is that in each segment, the heading lies within an angular range of some pre-specified size α, or it is UNDEFINED. In other words, for each segment, all of its edges have length zero and the heading is UNDEFINED, or there exists an angle β such that all edges in the segment have angles in the range $[\beta, \beta + \alpha]$. This range should be interpreted modulo 2π, as heading has a circular scale. The angle α must be chosen smaller than π; a reasonable value might be $\pi/3$ but it depends on the application. This *angular range criterion for heading* is monotone.

Speed The speed at any moment in time is well-defined on any edge since we assume constant speed on any edge. At any vertex, the speed can be chosen the same as the speed on the edge just before or after the vertex. A straightforward criterion for speed is that on any segment, the difference (or ratio) of the maximum and minimum speed is at most some given value. This *difference criterion for speed* (or *ratio criterion for speed*) is monotone.

Velocity The velocity at any location on τ is captured by a vector whose length is speed and whose direction is heading. Previously we bounded speed and heading separately, but we can also define a criterion directly on velocity. To this end, consider the *velocity vector plane*, where any point p represents the vector from the origin O to p. The origin O represents the null vector. Since all points on any single edge of τ have the same velocity, they are represented by the same point in the velocity vector plane.

Let α be some fixed angle in $[0, \pi]$ that is specified by the criterion. An *α-wedge* is a wedge whose apex is at O and that has opening angle α in the velocity vector plane. The *disk criterion for velocity* specifies that there exists a disk inside an α-wedge such that for each segment of the segmentation and for any location on an edge in that segment, its velocity vector has its representing point in that disk. Note that if the representing points fit in a disk that lies inside a wedge of opening angle α, we can always enlarge the disk to be tangent to the bounding rays of the wedge and keep the points representing the velocity vectors inside.

The criterion is quite similar to the angular range criterion for heading combined with the ratio criterion for speed using a conjunction, for which the shape is a sector of an annulus centered at O in the velocity vector plane. We include it to show that our framework can handle a variation like this as well. Figure 2 illustrates the two criteria.

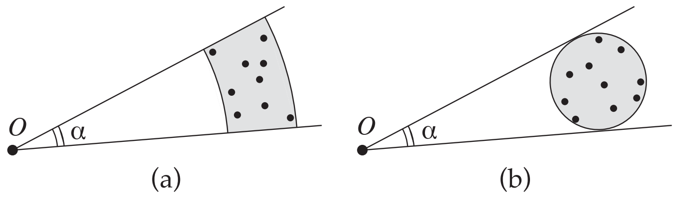

Figure 2: The points represent velocity vectors starting at O. (a) Speed and heading are bounded independently. (b) Speed and heading are bounded in conjunction with the disk criterion.

4 Algorithmic framework

We show that a trajectory with n edges can be segmented optimally and in $O(n \log n)$ time if the following conditions are fulfilled on the criterion that is used.

- The criterion is *monotone*.
- There is an $O(m)$ time test to decide whether a subtrajectory $\tau[t_i, t_j]$ satisfies the criterion (where $m = j - i + 1$).
- If subtrajectory $\tau[t_i, t_{j-1}]$ satisfies the criterion but subtrajectory $\tau[t_i, t_j]$ does not, there is an $O(m \log m)$ time method to maximize $q \in [t_{j-1}, t_j]$ such that subtrajectory $\tau[t_i, q]$ satisfies the criterion.
- The number of segments in the output is $O(n)$.

The first condition is necessary for the optimality of the segmentation, whereas the other three conditions are needed for the running time. The third condition is not needed for discrete segmentation and the fourth condition is trivially fulfilled for discrete segmentation. In the extreme case that the number of segments in the output is superlinear in n, this number will be an additional term in the running time. This will not occur in practice, since segmentations that have more segments than the trajectory has vertices are typically not useful.

4.1 Greedy strategy

A greedy strategy for segmentation starts at the beginning of the trajectory and makes the first segment as long as possible, until the segment would not satisfy the criterion anymore. The remainder of the trajectory is then segmented in the same way, always choosing the longest possible next segment. If a segmentation problem uses a criterion that is monotone, then a greedy method can be used to efficiently find an optimal solution. We show this next.

Theorem 4. *Given a trajectory τ and a monotone criterion for segmentation that should be satisfied, a greedy strategy yields an optimal solution to the segmentation problem.*

Proof. Let $t_0 = a_0, \ldots, a_k = t_n$ be the sequence of times where the greedy segmentation yields segment boundaries, and assume that an optimal segmentation has segment boundaries at times $t_0 = b_0, \ldots, b_l = t_n$. Let b_i be the first time with $b_i > a_i$ (if no such i exists then $k \leq l$ and the greedy strategy is optimal as well). Since $b_{i-1} \leq a_{i-1}$, the i-th greedy segment starts no earlier than the optimal segmentation. But then, by monotonicity, $\tau[a_{i-1}, b_i]$ must satisfy the criterion as well because $[a_{i-1}, b_i] \subseteq [b_{i-1}, b_i]$, and the greedy strategy would have extended $[a_{i-1}, a_i]$ at least up to b_i. Hence, $b_i > a_i$ leads to a contradiction with the greedy strategy. □

4.2 Algorithm outline

With slight abuse of terminology, we use *vertex* also for a time stamp like t_i, and we use *edge* also for an elementary time interval like $[t_i, t_{i+1}]$. The algorithm to compute an optimal segmentation on a given criterion has a rather simple structure. Starting at the end time s of the last completed segment (for the first segment we let $s = t_0$), we find the longest segment that satisfies the criterion by determining a vertex t_j so that $\tau[s, t_{j-1}]$ satisfies the criterion but $\tau[s, t_j]$ does not. We proceed by maximizing the part of the edge $[t_{j-1}, t_j]$ that still satisfies the criterion to determine the latest end time of the current segment. (If the goal was to compute a discrete segmentation, then we end the current segment at t_{j-1} and start the next segment at t_j.)

Suppose that we have two routines. One is an algorithm TEST($\tau[s, q]$) which returns true if the subtrajectory $\tau[s, q]$ satisfies the criterion, and false otherwise. The other is an algorithm FURTHEST($\tau[s, t_j]$) where t_{j-1} is the last vertex for which $\tau[s, t_{j-1}]$ satisfies the criterion, and returns the last time in $[t_{j-1}, t_j]$ that satisfies the criterion. FURTHEST is not needed in case of discrete segmentation and for attributes that are constant on edges.

Incremental method The most simple implementation for a greedy strategy is incremental. Let t_{i+1} be the first vertex strictly after s, the end time of the last segment. We incrementally call TEST($\tau[s, t_{i+1}]$), TEST($\tau[s, t_{i+2}]$), ..., until a test fails. If this happens for the test TEST($\tau[s, t_j]$), then we run FURTHEST($\tau[s, t_j]$). The efficiency of this implementation depends on whether we can efficiently perform TEST($\tau[s, t_{i+a+1}]$) if we already know that TEST($\tau[s, t_{i+a}]$) returns true. For the monotone criteria given for heading and speed, we can test every next vertex and edge in $O(1)$ time and this results in a linear running time. More generally, using this method one can segment a trajectory with n edges optimally in time $O(n)$ with respect to a range criterion of any single-valued attribute, if FURTHEST takes $O(m)$ time on a subtrajectory of m edges.

For the disk and diameter criteria for location, the attribute is not single-valued. For an efficient incremental solution, we would need an algorithm that efficiently maintains the diameter or smallest enclosing disk under insertions. Such an algorithm exists for the problem of deciding whether a disk of given radius exists that contains the points, under insertions and deletions given off-line with $O(\log m)$ update time [21], but not for the diameter version. In any case, the method we describe next is simpler, equally or more efficient, and more general.

Double-and-search method We next present a different implementation of the greedy strategy that is guaranteed to work in $O(n \log n)$ time for many criteria. We present this result in a general theorem later in this section. The implementation uses the doubling search technique which combines an exponential and a binary search. The pseudocode for the implementation is given in Algorithm 1.

Suppose we have segmented a trajectory up to a time s, and let t_{i+1} be the first vertex strictly after s. Initialize $a \leftarrow 1$. Then we call TEST($\tau[s, t_{i+a}]$). If successful, we double the value of a and repeat. The loop ends if the test is not successful or $i + a > n$. In the latter case we call TEST($\tau[s, t_n]$). If successful, the whole remainder of the trajectory is the last segment and we stop.

If $a = 1$, we know $j = i + 1$. Else $a = 2^b$ for some $b \geq 1$ and we know that $j \in [i + a/2,\ \min(i + a, n)]$. In this interval of size at most $a/2$, we perform a binary search using TEST to make decisions to determine the exact value of j. When j is determined, we call FURTHEST($\tau[s, t_j]$) to determine the latest time q on the edge $[t_{j-1}, t_j]$ such that the criterion for the segment up to q is still satisfied. The time q is then used as the new starting time s in the computation of the next segment.

This technique is akin to a technique also used in [1].

Algorithm 1 Simple algorithmic framework for trajectory segmentation.

```
// Input: $\tau = (v_0, t_0), \ldots, (v_n, t_n)$
$i \leftarrow 0$; $s \leftarrow t_0$; $S_{opt} \leftarrow \emptyset$;
while ($s \neq t_n$)
{
    // Phase 1: find first vertex $v_j$ that "does not fit"

    $a \leftarrow 1$;
    while ($i + a \leq n$ && TEST($\tau[s, t_{i+a}]$))
            { $a \leftarrow 2a$; }
    $j \leftarrow$ Binary search in $[i + \lfloor a/2 \rfloor, \min(i + a, n)]$,
            s.t. TEST($\tau[s, t_{j-1}]$) = true $\wedge$ ($j = n$ $\vee$ TEST($\tau[s, t_j]$) = false);

    // Phase 2: find latest time $q$ on edge $[t_{j-1}, t_j]$

    $q \leftarrow$ FURTHEST($\tau[s, t_j]$);
    $S_{opt} \leftarrow S_{opt} \cup \tau[s, q]$; // Next segment found
    $s \leftarrow q$; $i \leftarrow j - 1$;
}
```

Theorem 5. *For a trajectory with n edges, if* TEST *takes $T(m)$ time for a subtrajectory with m edges and* FURTHEST *takes $F(m)$ time for a subtrajectory with m edges, then optimal segmentation takes $O(T(n) \log n + F(n))$ time (assuming $T(m)$ and $F(m)$ are at least linear in m and at most polynomial in m and the number of segments in the output is linear in n).*

Proof. Suppose the optimal number of segments is h, let $m_1, \ldots, m_h$ be the numbers of edges (fully or partially) spanned by the segments which are computed by our algorithm. Then we have $\sum_{i=1}^{h} m_i \leq n + h - 1$.

During the computation of one segment of unknown size m, TEST is called on subtrajectories that have size smaller than $2m$. The number of times it is called in the while-loop is bounded by the maximal a such that $2^a \leq 2m$, i.e., $a \leq \log 2m$, since the test value is doubled until it reaches the first number that is greater than m. After this, a binary search on the interval of size at most m will perform a maximum of $O(\log m)$ calls to TEST on subtrajectories of size smaller than $2m$. Computing a segment spanning m edges therefore takes at most $O(T(2m) \log m + F(m))$ time. Note that FURTHEST is called only once at the end.

We conclude that the algorithm takes $\sum_{i=1}^{h}(T(2m_i) \log m_i + F(m_i))$ time in total. Since $T(m)$ is at least linear, $T(m) \log m$ is at least linear as well, and since $F(m)$ is also linear, $\sum_{i=1}^{h}(T(2m_i) \log m_i + F(m_i)) \leq T(n+h) \log(n+h) + F(n+h)$. Since $h = O(n)$ and $T(m)$ and $F(m)$ are at most polynomial, we have $T(n+h) \log(n+h) + F(n+h) = O(T(n) \log n + F(n))$. □

5 Segmentation using basic attributes

In this section we apply the framework to single criteria and multiple criteria in combination. It is sufficient to provide efficient implementations of the two routines TEST and FURTHEST, and then our framework implies the efficient algorithm for optimal segmentation. We first discuss single criteria involving only one attribute, and then we discuss combinations of criteria.

5.1 Univariate attribute criteria

For any of the monotone criteria listed earlier for heading and speed, we can show that TEST can be performed in linear time, and Theorem 5 yields $O(n \log n)$ time solutions for segmentation. Since these attributes are constant along the edges, we do not need the procedure FURTHEST.

In general, we can also apply the framework to univariate attribute functions that are not piecewise constant over the edges, assuming that a small set of requirements are met. We denote the attribute function by $\phi(t)$. Its definition will generally depend on a sequence of analytic functions $\psi_1(t), \ldots, \psi_k(t)$ that become valid in this order at certain points in time.

For example, consider the attribute speed. Suppose that we did not only record positions of our moving objects, but also the current speed at each time stamp. Then we could choose to interpolate speed on $[t_i, t_{i+1}]$ by linearly interpolating the measured values at t_i and t_{i+1}. Now the analytic functions $\psi_i(t)$ are linear functions (instead of constant), and our greedy approach will segment on edges (instead of at vertices). We could also use higher-degree polynomials to interpolate speed, which would be necessary to make speed a differential function in time. In Section 6 we will see examples of more complex analytic functions, and also cases where the number of analytic functions k needed to define $\phi(t)$ is larger than n.

We prove a general result on segmenting a trajectory on a univariate criterion that satisfies some conditions on the computations needed on $\psi_1(t), \ldots, \psi_k(t)$.

Theorem 6. *Let τ be a trajectory with n edges, and let an attribute function $\phi(t)$ be defined for any $t \in [t_0, t_n]$ by analytic functions $\psi_1(t), \ldots, \psi_k(t)$, with $k = O(n)$. If for any $1 \leq i \leq k$,* *we can (i) evaluate $\psi_i(t)$ in time $O(1)$, (ii) compute the minima and maxima of $\psi_i(t)$ over the interval on which it is defined in time $O(1)$, and (iii) evaluate $\psi_i^{-1}(y)$ in time $O(1)$, then an optimal segmentation with respect to a range criterion of this attribute can be computed in $O(n \log n)$ time after preprocessing.*

Proof. Assumption (ii) implies that each $\psi_i(t)$ has $O(1)$ extrema in the interval on which it is defined. We add these extrema as time stamps and vertices to τ, and note that τ still has $O(n)$ vertices and edges in total. Adding time stamps at the extrema of each $\psi_i(t)$ causes $\phi(t)$ to be monotone (increasing or decreasing) on each edge. The functions $\psi_1(t), \ldots, \psi_k(t)$ can each be associated with $O(1)$ consecutive edges. With this adaptation, TEST can be used as before, and we need to evaluate ϕ only at vertices of τ to determine if a subtrajectory $\tau[s, t_j]$ satisfies the range criterion of the attribute.

The routine FURTHEST($\tau[s, t_j]$) requires the computation of the inverse function ϕ_j^{-1} on some interval $[t_{j-1}, t_j]$, to find the last moment of time up to where the segment still satisfies the criterion. We evaluate ϕ at s and the vertices $t_{i+1}, \ldots, t_{j-1}$ to obtain the maximum and minimum values realized up to t_{j-1}. If function ψ_j is increasing, then we use the minimum realized value on $\tau[s, t_{j-1}]$ to compute the maximum allowed value Max of ψ_j. The correct result of FURTHEST is the time $\psi_j^{-1}(Max)$. Symmetrically, if function ψ_j is decreasing, then we use the maximum realized value on $\tau[s, t_{j-1}]$ to compute the minimum allowed value Min of ψ_j. The correct result of FURTHEST is the time $\psi_j^{-1}(Min)$. By assumption (iii), we can compute the inverse of ψ_j in constant time.

Now, Theorem 5 implies that optimal segmentation takes $O(n \log n)$ time after preprocessing, using the same algorithm as before. □

Theorem 7. *An optimal segmentation using the angular range criterion for heading, and the ratio or difference criterion for speed, can be computed in $O(n \log n)$ time for a trajectory with n edges.*

5.2 Location criteria

Suppose that we require for each segment that the locations are within a disk of radius r. Then TEST($\tau[t_i, t_j]$) can easily be implemented in linear time using known, simple algorithms for smallest enclosing disk on the points $v_i, \ldots, v_j$ [14, 28, 38]. Now, we need to use FURTHEST to determine the latest time that still satisfies the location criterion. For this task, a simple and efficient algorithm exists because it is a so-called *LP-type problem*. LP-type problems can be solved using *randomized incremental construction*, a powerful technique that can solve various optimization problems with a surprisingly simple implementation [11, 14, 17, 38]. Let P be the set of points $p_1, \ldots, p_m$; assume $p_1, \ldots, p_{m-1}$ fit inside some radius r disk but not $p_1, \ldots, p_m$. Apply a similarity transformation to P so that $p_{m-1} = (0, 0)$ and $p_m = (0, 1)$. Now the problem of finding the disk with given radius that contains p_{m-1} and the largest portion of $\overline{p_{m-1}p_m}$ is transformed into the problem of finding a disk with some (different) radius r' that contains $P \setminus \{p_m\}$ (after transformation) and has the rightmost possible intersection with the x-axis.

Either the optimal solution is realized by a disk whose center is the center point of the line segment connecting the intersection of the disk with the positive x-axis and one of the points of $P \setminus \{p_m\}$, or it is realized by a disk that has two points on its boundary, see Figure 3. In the latter case, the two points on the boundary and the intersection point form a triangle that contains the center of the disk.

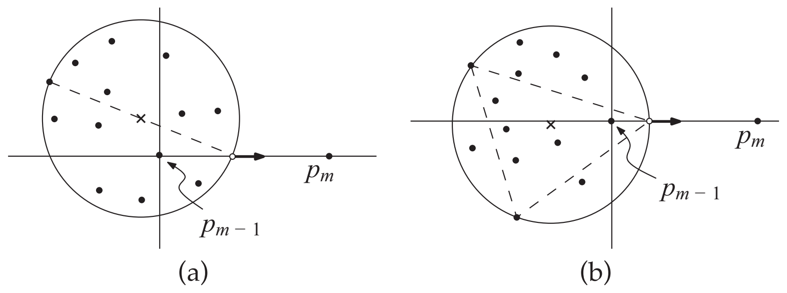

Figure 3: Optimal solution determined by (a) one point, or (b) two points. The arrow to p_m illustrates the optimization.

The pseudocode for solving the problem is given in Algorithm 2; the returned result must be transformed back by the inverse of the affine transformation. Note the similarity to the code for smallest enclosing disk or linear programming in constant dimensions [14].

Algorithm 2 Randomized incremental construction for FURTHEST **for the disk criterion for location.**

```
// Input: A set $P$ with $m-1$ points, and a radius $r$
Let $(p_1, \ldots, p_{m-1})$ be the points of $P$ in random order;
$D_1 \leftarrow$ OPTDISK$(r, p_1)$;
for $h \leftarrow 2$ to $m-1$
{
    if $D_{h-1}$ contains $p_h$
            { $D_h \leftarrow D_{h-1}$; }
    else $D_h \leftarrow$ OPTDISK$(r, p_1, \ldots, p_{h-1})$ with the
                condition that $p_h$ is on $D_h$'s boundary
}
return the rightmost point on the $x$-axis and in $D_{m-1}$
```

Algorithm 2 uses a routine OPTDISK which can be implemented as follows. Note that a disk of fixed radius whose boundary contains a point p_h can only pivot around this point, and has just one degree of freedom which is angular. Every point in $p_1, \ldots, p_{h-1}$ restricts the angle by some angular interval, and the common intersection of these is again an angular interval that is computed in $O(m)$ time by a single for-loop. Over this angular interval we can optimize the disk easily in $O(1)$ time. Therefore, using the standard efficiency analysis for randomized incremental construction [14], Algorithm 2 takes linear expected time, where the expectation is only over the randomization performed within the algorithm. That is, we do not assume a certain probability distribution on the input.

Lemma 8. *Given a subtrajectory $\tau[t_i, t_j]$ such that $v_i, \ldots, v_{j-1}$ fit in a disk of radius r but $v_i, \ldots, v_j$ do not fit in any disk of radius r, the problem of computing a disk of radius r that contains $v_i, \ldots, v_{j-1}$ and the largest possible part of e_j is LP-type.*

Proof. We will show that the equivalent, transformed problem is an LP-type problem. Since the radius of the disk is fixed, an optimal solution for P is already determined by up to 2 points of P, the *basis* of the problem, which shows that the combinatorial dimension of the problem is 2. We need to show the two requirements *monotonicity* and *locality*.

Monotonicity is trivially satisfied: if a point of the set is removed, then the disk still covers the remaining points, and therefore the solution for the new problem instance cannot be worse (meaning: less of the positive x-axis covered). To ensure locality we need to show the following: if a point set G yields an optimal point with x-coordinate x_G and a subset $F \subset G$ also yields $x_F = x_G$, then the two disks are identical. Let D_G, D_F be the disks corresponding to G and F, respectively. Assume for the sake of contradiction that $D_F \neq D_G$. Clearly all points in F lie inside $D_F \cap D_G$ because both are enclosing disks. The two disks have boundaries that pass through $(x_G, 0)$, since they have the same optimal solution. Let q be the other intersection point of the boundaries of D_F and D_G. Now we can pivot D_F around q while increasing $D_F \cap D_G$, which necessarily results in an intersection point of the boundary of D_F with the positive x-axis with higher x-coordinate, a contradiction. □

Next we consider the diameter criterion for location. For any set of n points in the plane, the diameter can be computed in $O(n \log n)$ time as follows. First, sort the points on x-coordinate in $O(n \log n)$ time. Second, construct the convex hull of the sorted set of points in $O(n)$ time by a Graham scan [14]. Third, perform a rotating calipers step (visit antipodal pairs) on the convex hull in $O(n)$ time to find the furthest pair of points, see [32]. Hence, we can implement TEST to run in $O(m \log m)$ time on a subtrajectory of m edges.

We can implement FURTHEST to run in $O(m)$ time. The maximum allowed diameter d will be realized by one point among $\tau(s), v_{i+1}, \ldots, v_{j-1}$ and a point on the edge e_j. The desired point on e_j has distance exactly d to one point from $\tau(s), v_{i+1}, \ldots, v_{j-1}$ and a smaller distance to the other points. Hence, we simply compute for each point the point on the edge e_j that is at distance d. From the points on e_j, we choose the one closest to v_{j-1} and return it.

Theorem 9. *An optimal segmentation using the disk criterion for location can be computed in $O(n \log n)$ time for a trajectory with n edges, and using the diameter criterion in $O(n \log^2 n)$ time.*

Remark: We will show in Section 7 that we can also segment optimally using the diameter criterion in $O(n \log n)$ time. This requires an extra algorithmic idea that we postpone for now.

5.3 Velocity criteria

We next turn our attention to velocity without using speed and heading separately. Recall that this problem must be solved in the velocity vector plane, where points represent the velocity vectors of the edges of τ. Note that the origin O represents the null vector. Let α be the fixed opening angle of the wedge with apex at the origin O specified by the disk criterion; we denote such a wedge by α-wedge. Note that we cannot simply compute the minimum enclosing disk and check if the smallest wedge that contains it has angle at most α: a slightly larger disk that is further from O may have a smaller opening angle, see Figure 4(a). Instead, we consider α-wedges only, and will compute the *smallest* disk that is twice tangent to an α-wedge. We can show that this is an LP-type problem, and hence TEST for the disk criterion for velocity can be solved in linear expected time using randomized incremental construction. The algorithm may give a smallest disk that is twice tangent to an α-wedge, and then we return true, or no smallest disk exists (and therefore no disk at all), and then we return false as the result of TEST.

The following lemma describes center points of disks that contain a fixed point on their boundary while remaining twice tangent to some α-wedge, see Figure 4(c).

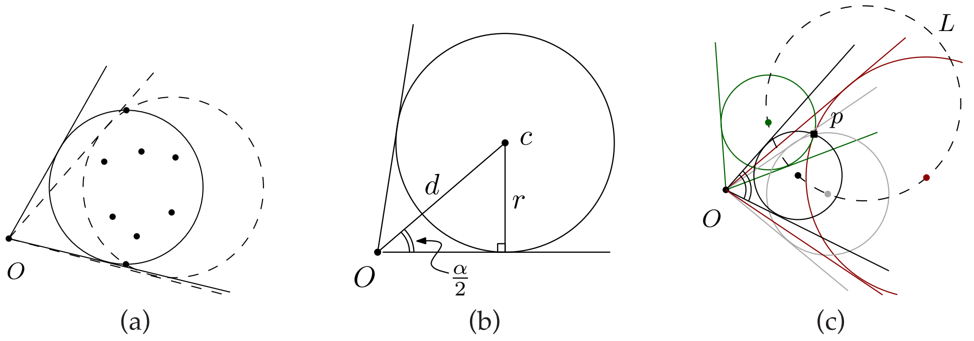

Figure 4: (a) Computing the tangents to the minimum enclosing circle is not guaranteed to give the wedge with smallest angle. (b) The radius r and the distance d of the circle center to O are directly related via $d\sin(\alpha/2) = r$. (c) The circles pivoting at a fixed point p while remaining tangent to an α-wedge have their center on the circle L.

Lemma 10. *For any point p, the locus of the center points of disks that are twice tangent to an α-wedge and which contain p is a disk.*

Proof. For any disk inside and twice tangent to the α-wedge, denote its center by $c = (c_1, c_2)$ and radius by r, and let $d = d(c, O)$. Notice that $r = d\sin(\alpha/2)$ for all such disks, see Figure 4(b). Hence, $p = (p_1, p_2)$ lies inside the disk if and only if

$$(p_1 - c_1)^2 + (p_2 - c_2)^2 \leq r^2 = d^2 \sin^2(\alpha/2) = (c_1^2 + c_2^2)\sin^2(\alpha/2).$$

This is equivalent to

$$\left(c_1 - \frac{p_1}{\cos^2(\alpha/2)}\right)^2 + \left(c_2 - \frac{p_2}{\cos^2(\alpha/2)}\right)^2 \leq (p_1^2 + p_2^2)\frac{\sin^2(\alpha/2)}{\cos^4(\alpha/2)}$$

which means that c lies inside the disk centered at the point $(\frac{p_1}{\cos^2(\alpha/2)}, \frac{p_2}{\cos^2(\alpha/2)})$ with radius $\sqrt{(p_1^2 + p_2^2)}\frac{|\sin(\alpha/2)|}{\cos^2(\alpha/2)}$. See Figure 4(c) for an illustration. □

Lemma 11. *The problem of computing the smallest disk twice tangent to and inside an α-wedge is LP-type.*

Proof. If the velocity vector representations (points) of the edges $e_i, \ldots, e_j$ of the subtrajectory can be covered by a disk that is twice tangent to an α-wedge, then the center point of this disk lies in the intersection of the disks of the type described in 10. We can compute these disks $D_i, \ldots, D_j$ in $O(m)$ time, where $m = j - i + 1$. The radius of a disk tangent to a wedge of apex O and opening angle α is related to the distance d of its center to O by $d\sin(\alpha/2) = r$, see Figure 4(b). Therefore, the *smallest* disk that covers the points and is twice tangent to an α-wedge has its center at the point on the boundary of $D_1 \cap \cdots \cap D_n$

and is closest to O; its radius is uniquely defined by the position of its center point. A basis either consists of one disk D_i, in which case the point on the boundary of D_i that is closest to O is the center point that realizes the smallest disk, or the basis consists of two disks D_i and D_j, in which case the center point of the smallest disk is the intersection point of the boundaries of D_i and D_j that is closest to O. Therefore the basis computation is trivial, and the combinatorial dimension of the problem is again 2.

The monotonicity criterion is trivially satisfied: the intersection of a set of disks G is still contained in the intersection if we remove a disk D_i from the set. Therefore a valid solution for G is also a solution for $G \setminus D_i$ and the closest center point cannot be further from O. To show that the locality criterion is satisfied, we argue as follows. Let $F \subset G$ be a subset of the disks in G, and assume for a contradiction that F and G define different closest center points c_F and c_G that are at the same distance from O. Let $R = \bigcap_{D \in G} D$, then $c_F, c_G \in R$. By convexity of R, the whole line segment connecting c_F and c_G lies in R. But then the midpoint is in R and it is closer to O than c_F and c_G, a contradiction. □

The lemma above immediately implies that TEST for the disk criterion for velocity can be made to run in linear expected time. Since velocity is constant over edges of τ, we do not need FURTHEST for this criterion. We obtain:

Theorem 12. *An optimal segmentation using the disk criterion for velocity can be computed in $O(n \log n)$ time for a trajectory with n edges.*

5.4 Combinations of segmentation criteria

In this section we show that there are several simple ways to combine segmentation criteria of various attributes. The resulting combined criterion is monotone if the separate criteria are monotone. In our framework, using a combined criterion for segmentation, made up of a constant number of criteria, has the same asymptotic running time as each separate criterion. We present two different ways of combining criteria, namely boolean combinations and linear combinations. The criteria to be combined can be the ones for location, heading, speed, and velocity, but also the criteria for more complex attributes to be presented later in this paper.

Definition 13. *Let $\Phi_1, \ldots, \Phi_k$ be a set of monotone criteria, where $\Phi_i : [t_0, t_n] \times [t_0, t_n] \rightarrow \{True, False\}$. We call any combination of conjunctions (e.g., $\Phi_i \wedge \Phi_j$) and disjunctions (e.g., $\Phi_i \vee \Phi_j$) of these criteria a* boolean combination criterion.

For example, we may allow a segment in a segmentation to have any speed and heading as long as the diameter of its locations is at most 2 km (criterion Φ_1), or a segment may have any location as long as the difference criterion for heading is 30 degrees (criterion Φ_2) and the difference criterion for speed is 20 km/h (criterion Φ_3). The boolean combination would be $\Phi_1 \vee (\Phi_2 \wedge \Phi_3)$. One can imagine that a segment satisfying Φ_1 could indicate local inspection or foraging behavior, whereas a segment satisfying $\Phi_2 \wedge \Phi_3$ could indicate various forms of directed travel, also segmented by speed.

Theorem 14. *Let $\Phi_1, \ldots, \Phi_c$ be a constant number of monotone criteria, and assume that for each of the Φ_i, an algorithm for* TEST *runs in $O(m)$ time and an algorithm for* FURTHEST *runs in $O(m \log m)$ time on a subtrajectory of m edges. Then we can compute an optimal segmentation using any boolean combination criterion of these criteria in $O(n \log n)$ time, for a trajectory of n edges.*

Proof. Let Φ denote an arbitrary boolean combination criterion of the Φ_i. Let I be a a set of subsets S of the index set $\{1, \dots, c\}$, such that

$$\Phi_{CNF} = \bigvee_{S \in I} \bigwedge_{i \in S} \Phi_i$$

is the conjunctive normal form of Φ. It can be computed in constant time, since c is constant. We first show that Φ_{CNF} is a monotone criterion. If subtrajectory τ' does not satisfy the criterion Φ_{CNF}, then for every $S \in I$, there must be an Φ_i, such that $i \in S$ and τ' does not satisfy the criterion Φ_i. By the monotonicity of Φ_i, it also holds that τ'' does not satisfy Φ_i if $\tau' \subseteq \tau''$. Therefore, τ'' does not satisfy Φ_{CNF}. We observe that Φ_{CNF} has constant size, since the index set $\{1, \dots, c\}$ has constant size and therefore only a constant number of different subsets exist that are candidates for $S \in I$ and furthermore, each such subset S has constant size.

To prove the efficiency, we will argue that we can give algorithms for TEST and FURTHEST with respect to Φ that run in $O(m)$ time and $O(m \log m)$ time, respectively, for a subtrajectory of m edges. Theorem 5 then implies the claim. Recall that we required that there is an algorithm for TEST for each of the Φ_i that runs in $O(m)$ time. Let τ' be any such a subtrajectory. We invoke the test function on τ' for each of the Φ_i. This takes $O(cm) = O(m)$ time. Let a_i be the outcome of the test for Φ_i. Now, the outcome of TEST on τ' for Φ can be determined by evaluating $\bigvee_{S \in I} \bigwedge_{i \in S} a_i$ in constant time. Similarly, for FURTHEST, given a subtrajectory $\tau[s, t_j]$ such that $\tau[s, t_{j-1}]$ satisfies Φ but $\tau[s, t_j]$ does not, we can run the respective algorithms for each of the Φ_i separately. This computation takes $O(cn \log n) = O(n \log n)$ time. Let the outcome be the values $t'_1, \dots, t'_c$. We have that the furthest point on the edge $\tau[t_{j-1}, t_j]$, such that the subtrajectory satisfies Φ, is defined by $t' = \max_{S \in I} \min_{i \in S} t'_i$. □

Another way to combine different criteria is by linear combinations. In this way, two segmenting criteria may be combined such that less difference is allowed in one of them if there is already a significant difference in the other one, and vice versa. For example, we may allow a segment to have speed values different by at most 20 km/h if the heading is constant, and also allow it to have a maximum heading difference of at most 40 degrees if the speed is constant, by means of a linear combination of these two extremes. This would also allow intermediate values such as a speed difference of 10 km/h and a heading difference of 20 degrees.

Definition 15. *Given a set of univariate attribute functions $\phi_1, \dots, \phi_c$, and real coefficients $a_1, \dots, a_c$, consider the function $C(s, q)$ of a subtrajectory $\tau[s, q]$ defined as*

$$C(s, q) := \sum_{1 \leq i \leq c} a_i \max_{\substack{s \leq t \leq q \\ s \leq t' \leq q}} (\phi_i(t) - \phi_i(t')).$$

We call the criterion that is satisfied for $\tau[s, q]$ iff $C(s, q) \leq \delta$, for a given threshold value δ, a linear combination criterion*.*

If ϕ_1 is speed in km/h and ϕ_2 is heading in degrees, then the example above would use $a_1 = 2$, $a_2 = 1$, and $\delta = 40$, for the linear combination criterion.

Provided that certain computations are possible in an efficient manner on the attribute functions to be combined and on the function $C(s, q)$ as well, we can segment using a linear combination criterion in $O(n \log n)$ time. For typical attribute functions, these requirements will hold, but it must be verified for the combination of attributes to be used.

Theorem 16. *Given a constant number of univariate attribute functions $\phi_1, \ldots, \phi_c$, of which each $\phi_i(t)$ is defined for any $t \in [t_0, t_n]$ by analytic functions $\psi_{i,1}(t), \ldots, \psi_{i,k_i}(t)$, with $k_i = O(n)$. We can compute an optimal segmentation using any linear combination criterion with respect to these attributes in $O(n \log n)$ time for a trajectory with n edges, if the following requirements are met. For any $1 \leq i \leq c$ and for any $1 \leq j \leq k_i$, we can*

(i) evaluate $\psi_{i,j}(t)$ in time $O(1)$,
(ii) compute the minima and maxima of $\psi_{i,j}(t)$ over the interval on which it is defined in time $O(1)$, and
(iii) evaluate $f^{-1}(\delta)$ in time $O(1)$, where $f(q) := \sum_{1 \leq i \leq c} a_i(b_i + p_i \psi_{i,l_i}(q))$ for constants $b_1, \ldots, b_c$, $p_i \in \{1, -1\}$ and $1 \leq l_i \leq k_i$ and f monotone increasing.

Proof. We first show that the function $C(s, q)$ is monotone decreasing in s and monotone increasing in q. For any $1 \leq i \leq c$, we have

$$\max_{\substack{s \leq t \leq q \\ s \leq t' \leq q}} (\phi_i(t) - \phi_i(t')) = \max_{s \leq t \leq q} \phi_i(t) - \min_{s \leq t' \leq q} \phi_i(t'). \tag{1}$$

Let $f_i(s, q) := a_i(\max_{s \leq t \leq q} \phi_i(t) - \min_{s \leq t' \leq q} \phi_i(t'))$. Then $C(s, q) = \sum_{1 \leq i \leq c} f_i(s, q)$. For any given values $0 < s < q < q' < t_n$, we have that $f_i(s, q) \leq f_i(s, q')$, since the interval $[s, q']$ is a superset of the interval $[s, q]$. This implies that the function f_i is monotone increasing in the second parameter. Since the sum of monotone increasing functions is monotone increasing, we also have that $C(s, q)$ is monotone increasing in the second parameter. By a similar argument, it follows that $C(s, q)$ is monotone decreasing in the first parameter. As a consequence, the linear combination criterion is monotone.

We will now describe how to apply the framework using a similar approach as the one used in the proof of Theorem 6. For each $1 \leq i \leq c$ and $1 \leq j \leq k_i$ we add the extrema of the functions $\psi_{i,j}(t)$ in the interval on which they are defined as time stamps and vertices to τ. We also do this for the points where the domain of $\psi_{i,j}$ ends and the domain of $\psi_{i,j+1}$ starts. By Assumption (ii), each $\psi_{i,j}(t)$ has $O(1)$ extrema in the interval on which it is defined and there are $\sum_{1 \leq i \leq c} k_i = O(cn)$ of these functions. Therefore, τ still has $O(n)$ vertices and edges in total. Now, the algorithm for TEST for a subtrajectory $\tau[s, q]$ with m edges only has to determine the minimum and maximum of each attribute function ϕ_i over the interval $[s, q]$, which can be done in $O(cm)$ time. Then, $C(s, q)$ can be determined using Equation 1 in constant time. Therefore, we can give an algorithm for TEST that runs in $O(m)$ time. We can also give an algorithm for FURTHEST. For this procedure, we are given a subtrajectory $\tau[s, t_j]$ such that $\tau[s, t_{j-1}]$ satisfies the criterion but $\tau[s, t_j]$ does not. On the interval $[t_{j-1}, t_j]$, we can determine an analytical representation of the function $C(s, q)$ for fixed s. By construction, each attribute function ϕ_i is defined by one function ψ_{i,l_i} over the interval $[t_{j-1}, t_j]$ and this function is either monotone increasing or monotone decreasing over this interval. In the first case, $\min_{s \leq t \leq q} \phi_i(t)$ is constant for $q \in [t_{j-1}, t_j]$, in the second case this holds true for $\max_{s \leq t \leq q} \phi_i(t)$. Let b_i denote this constant. We have that over $[t_{j-1}, t_j]$, $C(s, q)$ for fixed s, is the sum of terms of the type $a_i(b_i - \psi_{i,l_i}(q))$ and the type $a_i(\psi_{i,l_i}(q) - b_i)$. By assumption (iii), we can evaluate the inverse of this function at value δ in $O(1)$ time, which gives the furthest point on the edge $\tau[t_{j-1}, t_j]$, such that the criterion is satisfied. By Theorem 5, we can compute an optimal segmentation in time $O(n \log n)$. □

6 Attributes with a neighborhood

While location, heading, speed, and velocity are perhaps the most basic attributes that can be defined for points on a trajectory, there are several others that may be useful for the segmentation problem. We study the attributes curvature, sinuosity, and curviness, and possible criteria for these that can be used in our algorithmic framework.

The attributes we discuss next require a neighborhood for their definition. A neighborhood of a point $\tau(t)$ is a subtrajectory that contains $\tau(t)$. For example, to define curviness at a location $\tau(t)$ on τ, we need to measure the total angular change for an interval around $\tau(t)$. We assume for now that this neighborhood always exists. There are three simple ways to obtain such a neighborhood:

k-Vertex neighborhood: The subtrajectory from the k-th vertex before $\tau(t)$ until the k-th vertex after $\tau(t)$ (counting $\tau(t)$ itself only once if it is a vertex, and clipped at the start and end of τ if necessary).

d-Space neighborhood: The subtrajectory from the location at distance d before $\tau(t)$ until the location at distance d after $\tau(t)$ (measured along the trajectory, and clipped at the start and end of τ if necessary).

$\hat{t}$-Time neighborhood: The subtrajectory in between time $t - \hat{t}$ and time $t + \hat{t}$ (clipped at the start and end of τ if necessary).

A vertex neighborhood may be appropriate only if the sampling is regular and no data is missing, otherwise, vertex neighborhoods are not meaningful. In case τ was sampled with regular time intervals, a time neighborhood is the continuous version of a vertex neighborhood. A vertex neighborhood, however, changes abruptly at the vertices, while time neighborhoods always change continuously.

An attribute that uses a neighborhood to define a value of the trajectory at a time t is based on some subtrajectory of τ. Let us denote by t^{left} and t^{right} the start and end times of this subtrajectory.

Notice that for times t near t_0 or t_n, the times t^{left} or t^{right} may lie outside the time span of the trajectory. This may cause an attribute value to be undefined. We can solve this in various different ways. For example, we could extrapolate the first and last edge of the trajectory sufficiently far, or we could use t_0 instead of t^{left} if $t^{\text{left}} < t_0$ as the start of the neighborhood, and use t_n instead of t^{right} when $t^{\text{right}} > t_n$ as the end of the neighborhood.

6.1 Computing the attribute functions

When the segmentation uses a criterion based on an attribute that requires a neighborhood, it may take more than a constant amount of time to even evaluate the attribute at some time t. The number of time stamps from $t_0, \ldots, t_n$ between t^{left} and t^{right} will influence the efficiency. Fortunately, we can pre-process the trajectory so that we can evaluate the attribute value at every time efficiently. How to do this is further described below.

The three locations $\tau(t^{\text{left}})$, $\tau(t)$, and $\tau(t^{\text{right}})$ generally lie on three different edges of τ, although these could be the same. When t increases, t^{left} and t^{right} also increase, and $\tau(t^{\text{left}})$, $\tau(t)$, and $\tau(t^{\text{right}})$ advance along τ. Since each of t^{left}, t, and t^{right} goes past the times $t_0, \ldots, t_n$ of the vertices of τ at most once during the whole increase of t from t_0 to t_n, the total number of triples of edges on which $\tau(t^{\text{left}})$, $\tau(t)$, and $\tau(t^{\text{right}})$ lie is at most $3n$. For the attributes discussed in this section, it holds that as long as $\tau(t^{\text{left}})$, $\tau(t)$, and $\tau(t^{\text{right}})$ lie on the same triple of edges, an attribute value of interest is constant or is given by a well-defined, analytic function of t. We denote these analytic functions by $\psi_1(t), \ldots, \psi_k(t)$, where $k \leq 3n$. The time intervals over which the $\psi_1(t), \ldots, \psi_k(t)$ are defined form a partition of $[t_0, t_n]$, so together they define the attribute of a function $\phi(t)$ for all of τ.

We compute these functions as follows. Perform a sweep with t from t_0 to t_n, and simultaneously keep track of t^{left} and t^{right}. Whenever t^{left} or t^{right} is on one of the times $t_0, \ldots, t_n$ that define vertices of τ, add a time stamp t and vertex $\tau(t)$ to the trajectory (unless t is on one of the times $t_0, \ldots, t_n$ itself already). For the time interval up to t from the previous time stamp before t, determine and store the analytic function that gives the attribute value for any time in that time interval. How this is exactly done depends on the attribute value definition, but for the examples we discuss in this section (curvature, sinuosity, and curviness), preprocessing takes only linear time in total.

6.2 Examples: curvature, sinuosity, and curviness

Next we discuss the attributes curvature, sinuosity, and curviness and show that we can segment optimally with respect to a range criterion for each attribute in each case. Recall that a range criterion bounds the maximum range, or extent, of an attribute by a maximum allowed difference or ratio, in the case of univariate attributes. Curvature, sinuosity, and curviness are univariate.

Curvature Discrete curvature estimators have been studied widely to deal with the fact that the standard curvature definition is not suitable for piecewise-linear curves [20, 26]. Different definitions of standard curvature exist in differential geometry which are all equivalent. The corresponding definitions of discrete curvature are not equivalent, however. The commonly used methods for discrete curvature can be classified into three categories: those that use Gaussian smoothing, those that use curve or circle fitting, and three-point estimators. All of these methods define the curvature at a point based on certain points in a neighborhood of this point, implicitly or explicitly.

Three-point estimators define the curvature at a point p using the three points $p^{\text{left}} = \tau(t^{\text{left}}), p = \tau(t), p^{\text{right}} = \tau(t^{\text{right}})$. We now discuss three definitions of this type, as described in [26], and how they can be used in our framework.

Firstly, the curvature can be defined as the rate of change of the direction of the tangent vector. A first definition of discrete curvature therefore uses the turning angle of the directed line segments connecting the three points, see also Figure 5(a):

$$\kappa(p) = \frac{\angle(p^{\text{left}}, p, p^{\text{right}})}{\|p - p^{\text{left}}\| + \|p^{\text{right}} - p\|}, \text{ where } \angle(p^{\text{left}}, p, p^{\text{right}}) = \arccos \frac{< p - p^{\text{left}}, p^{\text{right}} - p >}{\|p - p^{\text{left}}\| + \|p^{\text{right}} - p\|} \tag{2}$$

defines the turning angle. When this definition of discrete curvature is used, each of the analytic functions $\psi_1(t), \ldots, \psi_k(t)$ is given by an expression similar to the one for κ. Instead of using p^{left}, p, and p^{right}, we use the positions of these points as a function of t to get the analytic functions that together define the curvature along τ.

Secondly, the curvature can be based on the inverse of the radius of the osculating circle. The second definition of discrete curvature approximates this circle by the circle passing through the three points, which is the circumcircle of the triangle formed by them, see also

Figure 5(b):

$$\kappa(p) = \frac{2|(p - p^{\text{left}}) \times (p^{\text{right}} - p)|}{\|p^{\text{right}} - p\| \cdot \|p - p^{\text{left}}\| \cdot \|p^{\text{left}} - p^{\text{right}}\|} \tag{3}$$

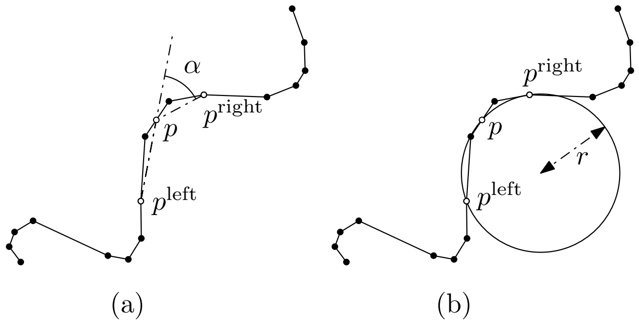

Figure 5: Discrete curvature by (a) turning angle, and (b) osculating circle.

Thirdly, the curvature is also the length of the second derivative vector if the curve is parametrized by arc length. The third definition of discrete curvature takes the discrete derivatives $\tau'(p^{\text{right}}) = p^{\text{right}} - p$ and $\tau''(p^{\text{right}}) = (p^{\text{right}} - p) - (p - p^{\text{left}})$, plugged into the curvature formula:

$$\kappa(p) = \frac{\tau'(p^{\text{right}}) \times \tau''(p^{\text{right}})}{\|\tau'(p^{\text{right}})\|^3}. \tag{4}$$

The attribute functions $\psi_1(t), \ldots, \psi_k(t)$ for discrete curvature according to all three definitions satisfy the three requirements stated in Theorem 6. When p^{left}, p, and p^{right} are on some triple of edges, the time t defining p gives rise to a time t^{left} that defines p^{left} and linearly depends on t. The analogous statement is true for p^{right}. Hence, we can express the curvature at p as an analytic function in t whose form can be stated easily. The function has $O(1)$ extrema on its interval that can be determined in $O(1)$ time, it can be evaluated in $O(1)$ time, and its inverse can be computed in $O(1)$ time as well. Hence we obtain:

Corollary 17. *An optimal segmentation using a range criterion for discrete curvature as defined in Equations (2), (3), or (4) using time, space or vertex neighborhoods can be computed in $O(n \log n)$ time for a trajectory with n vertices.*

Sinuosity The sinuosity of a point on a path refers to the amount of bending or winding of the path in the neighborhood of this point. The term is used in the analysis of rivers and coastlines [29, 31], but also to describe trajectories of moving animals [4, 5]. There appears to be no standard definition for sinuosity. If $\widehat{\tau} = \tau[t^{\text{left}}, t^{\text{right}}]$ is the subtrajectory that represents the neighborhood of $\tau(t)$, then some authors define the sinuosity at t as the arc length of $\widehat{\tau}$ divided by $\|\tau(t^{\text{left}}) - \tau(t^{\text{right}})\|$. We will refer to this as the *detour sinuosity*. Another existing definition of sinuosity is the angular range of heading, as defined in Section 3, divided by the arc length of $\widehat{\tau}$. We will refer to this as *winding sinuosity*. Note that both definitions measure a slightly different winding behavior, as illustrated in Figure 6(a). While winding sinuosity distinguishes the top two curves from the bottom curve, detour sinuosity distinguishes the bottom two curves from the top curve. Other definitions include the deviation of the edges of $\widehat{\tau}$ from the line segment connecting $\tau(t^{\text{left}})$ and $\tau(t^{\text{right}})$. We focus on the first two definitions and show how they can be used in our framework.

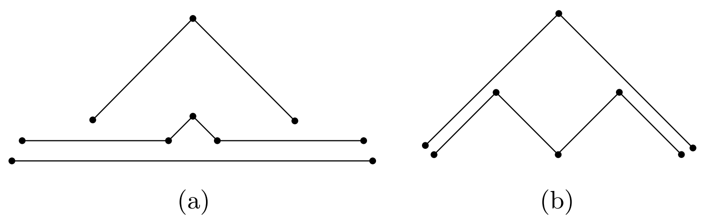

Figure 6: (a) Segments that are distinguished differently by winding and detour sinuosity. (b) Segments that are distinguished by curviness, but not by winding or detour sinuosity. (c) Segments that are distinguished by winding and detour sinuosity, but not by curviness.

Corollary 18. *An optimal segmentation using a range criterion for winding or detour sinuosity using time, space or vertex neighborhoods can be computed in $O(n \log n)$ time for a trajectory with n edges.*

Proof. We claim that Theorem 6 applies in each of the cases. If vertex neighborhoods are used, the elementary functions are piecewise constant, both for winding sinuosity as well as detour sinuosity. If we use space neighborhoods, then the arc length of $\widehat{\tau}$ is constant for all $\tau(t)$. As such, the winding sinuosity at a point depends only on the directions of the sequence of edges covered by its neighborhood. Therefore the elementary attribute functions are constant-valued functions in this case too. Similarly, detour sinuosity depends only on the distance between the two endpoints of $\widehat{\tau}$ if we use space neighborhoods. Within the range of one elementary attribute function these endpoints move on straight lines, and the function of their distance can be expressed as the square-root of a polynomial of degree two, which fulfills the conditions of Theorem 6. If we use time neighborhoods, the arc length of $\widehat{\tau}$ is a linear function, since speed is constant on the edges of the trajectory. We can apply Theorem 6 since the composition of a linear function with the discussed functions still satisfies the conditions. □

Curviness In the context of the discussed sinuosity definitions we propose another measure that captures the winding of a path near a point. Let $v_i, \dots, v_j$ be the subsequence of vertices strictly inside the neighborhood of $\tau(t)$, then we define the curviness at $\tau(t)$ as the total angular change, divided by the arc length of the subtrajectory that is the neighborhood of $\tau(t)$. We define total angular change as

$$\sum_{h=i}^{j} |\angle(v_{h+1} - v_h,\ v_h - v_{h-1})|,$$

where $\angle(v, w) \in (-\pi, \pi]$ is the angle of w with respect to v, interpreted as vectors $\vec{ow}$ and $\vec{ov}$. Again, this definition measures a different winding behavior than sinuosity, a fact illustrated in Figure 6(b) and (c). The proof of the following corollary is analogous to the proof Corollary 18.

Corollary 19. *An optimal segmentation using a range criterion for curviness using time, space, or vertex neighborhoods can be computed in $O(n \log n)$ time for a trajectory with n edges.*

7 Shape fitting criteria

In this section we discuss a criterion that is satisfied if the subtrajectory is similar to a certain type of shape. This shape could be a line or a circle, or it could be a more complex shape. The similarity could be defined in different ways. In the cases that we discuss below it is the directed Hausdorff distance from a subtrajectory τ' to any line ℓ:

$$\min_{\text{line } \ell} \max_{p \in \tau'} \min_{q \in \ell} \|p - q\| .$$

Requiring that a subtrajectory has a small distance to a line is a similar but more robust criterion than the angular range criterion for heading. For example, if a moving object is traveling in one direction, but due to an error one measurement is located just behind the previous location, then segmentation by heading would require the current segment to end, while segmentation by line fitting would not. If one requires a similarity to a circle, this could be a robust replacement for the criteria that we described for curvature.

Surprisingly, this type of criterion is monotone and therefore we can apply our framework. However, depending on the complexity of the shape and the degrees of freedom allowed in the similarity transformation, the computations necessary for TEST and FURTHEST can be quite expensive. We discuss how to apply our framework in the most simple case, where we compare the subtrajectory to a line. We show how to compute an optimal segmentation in $O(n \log n)$ time in this case.

Definition 20. *Given a subtrajectory τ' and a value δ, we say that τ' satisfies a* line fitting criterion *with threshold δ if there exist two parallel lines L and M at distance δ such that any point on τ' lies below or on L and above or on M.*

Note that even though we claim that this criterion is more robust, it may also have some disadvantages. The line fitting criterion does not detect a very sharp turn of angle close to π, for example. A significant part of the trajectory before the turn and after the turn could be close to some line, which causes the parts before and after the turn to be in the same segment.

Theorem 21. *An optimal segmentation using a line fitting criterion can be computed in $O(n \log n)$ time for a trajectory with n edges.*

Proof. Clearly, the criterion is monotone. If there exist two parallel lines L and M of distance δ which enclose a subtrajectory τ', then these two lines also enclose any subtrajectory of τ'.

Now, we describe algorithms for TEST and FURTHEST that each run in $O(m \log m)$ time on a subtrajectory of m edges. For TEST we can use an algorithm to compute the width of a point set, because a subtrajectory satisfies the line fitting criterion with threshold δ if and only if the width of its vertices is at most δ. The width can be computed using the rotating calipers algorithm [32] after constructing the convex hull.

For FURTHEST, we are given a subtrajectory $\tau[s, t_j]$ such that $\tau[s, t_{j-1}]$ satisfies the criterion but $\tau[s, t_j]$ does not. We need to compute the furthest point $r = \tau(t)$ on the edge $\tau[t_{j-1}, t_j]$, such that $\tau[s, t]$ satisfies the criterion. The width of a point set is always determined by three of its points: one point on one line and two points on the other line. In our setting, r is one of these points, and the width of $\{\tau(s), \ldots, \tau(t_{j-1})\} \cup \{r\}$ is exactly δ. Let L and M be the two lines that realize this situation. We have two cases, either the other two

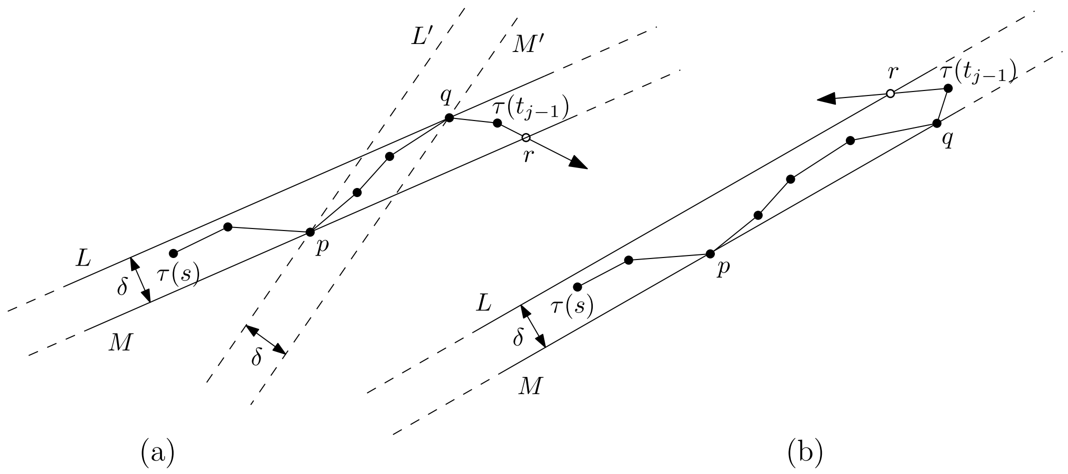

Figure 7: The two cases for the optimal configuration for FURTHEST using a line fitting criterion: (a) The furthest point r shares a line with p or q. (b) The furthest point r is on the opposite line of p and q.

points p and q lie on the same line and r on the other line, or p and q lie on different lines and r shares the line with one of them. See Figure 7 for an example of the two cases.

In the first case it must be that p and q are consecutive vertices on the boundary of the convex hull of $\tau[s, t_{j-1}]$. The point r is the intersection point of the edge $\tau[t_{j-1}, t_j]$ and the line at distance δ from the line through p and q. There are $O(m)$ candidate points r since there are $O(m)$ pairs of consecutive points on the convex hull.

In the second case points p and q must be antipodal on the convex hull of $\tau[s, t_{j-1}]$. This means that there must be two parallel lines, one containing p and one containing q, that contain $\tau[s, t_{j-1}]$ (or, equivalently, its convex hull). It is well-known that a set of m points has $O(m)$ antipodal pairs and they can be computed in $O(m)$ time by a rotating calipers algorithm after the convex hull has been computed [32]. We test every antipodal pair p and q and observe the following: If p and q are at distance less than δ, then they cannot be on the lines L and M that determine the width together with point r. If p and q are at distance greater than δ, there are exactly two pairs of lines that contain p and q and are at distance δ from each other. We call these pairs of lines L and M, and L' and M', see Figure 7(a). Using the edges incident to p and q on the convex hull of $\tau[s, t_{j-1}]$, we can test in constant time whether $\tau[s, t_{j-1}]$ lies between L and M (or L' and M'). If so, we compute r as the intersection point of $\tau[t_{j-1}, t_j]$ and L or M (or L' or M'). Assuming the convex hull was already computed, we handle every antipodal pair in $O(1)$ time.

So for both cases together we get $O(m)$ possible points r on $\tau[t_{j-1}, t_j]$ where we have checked that $\tau[s, t_{j-1}]$ lies in between L and M. We pick the furthest computed point on $\tau[t_{j-1}, t_j]$ over all these valid configurations. This yields a running time of $O(m \log m)$ for FURTHEST. We can apply Theorem 5, which implies that the overall computations take $O(n \log^2 n)$ time, since the number of segments will always be smaller than n with this criterion.

We can improve the running time to $O(n \log n)$ using the following idea. Suppose we are computing the next segment, which has m edges, and m is unknown. In the doubling phase of the algorithm we use TEST as we described it. But as soon as TEST fails for the first time, we use a slightly different method to find the $O(m)$ antipodal pairs of points. Suppose TEST fails on a subtrajectory $\tau(s), v_{i+1}, \ldots, v_{i+a}$. Then we sort these $a+1 = O(m)$ points by x-coordinate in $O(m \log m)$ time. Now we start the binary search phase but with a linear-time version of TEST. Suppose we test $\tau(s), v_{i+1}, \ldots, v_j$. We extract these points in x-sorted order from the sorted sequence $\tau(s), v_{i+1}, \ldots, v_{i+a}$ in $O(m)$ time. Then we use Graham's scan to compute the convex hull in $O(m)$ time as well, and find the antipodal pairs in $O(m)$ time as well. We can now analyze the doubling phase and binary search phase separately as in the proof of Theorem 5, which leads to $O(m \log m)$ time to find the maximal segment starting at $\tau(s)$. In total we spend $O(n \log n)$ time on the segmentation. □

Remark: As announced in Section 5.2, we can compute an optimal segmentation on the diameter criterion in $O(n \log n)$ time, improving upon $O(n \log^2 n)$ time claimed before. The idea is the same as in the last part of the proof above. In the doubling phase we use the $O(m \log m)$ time implementation of TEST. However, when it fails, we deviate from the usual binary search. We sort the $O(m)$ vertices $\tau(s), v_{i+1}, \ldots$ involved in the last TEST by x-coordinate in $O(m \log m)$ time, where m is again the unknown size of the segment we are computing. As described in Section 5.2, from a sorted sequence of vertices we can compute the diameter in only $O(m)$ time.

As mentioned above, we can also define a *circle-fitting criterion*, which is satisfied if the subtrajectory lies within an annulus of constant width. This criterion is also monotone, but it is computationally more expensive. If we are interested in segmentation of a discrete trajectory, we need an algorithm for TEST, which tests if the set of vertices spanned by the segment lie within a constant-width annulus. Furthermore, if we restrict the outer radius of the annulus to be fixed, the criterion is still monotone, but easier to compute. A boolean combination of a constant number of circle-fitting criteria with different outer radii could be used to detect subtrajectories of different curvature. Hence, we could use the algorithm described in [13] which runs in $O(m \log m)$ time, achieving an $O(n \log^2 n)$ time for the overall segmentation algorithm.

8 Robustness

The segmentation process is sensitive to noise and outliers. It is likely that such data problems can cause additional segmentation, or segmentation in a different location. While it is not the purpose of this paper to describe how to deal with noise and outliers successfully in all cases, we describe three conceptually different ways that apply to different parts of our framework. Recall that the segmentation in our framework is based on trajectories, attributes, and criteria. We can choose to make any of these three concepts more robust, which we describe next.

To make a trajectory more robust, we can apply outlier detection and removal, and smoothing for dealing with noise and imprecision. Sometimes the device already takes care of this. GPS often give coordinates that have been smoothed by Kalman filtering, for example.

To make an attribute more robust, for example heading, we can use a neighborhood definition for heading instead of a point-based definition. The heading is then determined by the direction of the vector from the starting point of the neighborhood to the endpoint of the neighborhood. This in essence smooths the attribute values.

To make a criterion more robust, we can allow for short durations where the criterion is not met. These durations can be absolute or relative. For example, a more robust criterion for speed is that within each segment of the segmentation, there is a speed s such that 95% of the time, the speed is in the interval $[s - 3, s + 3]$ m/s. We note that although such a definition of a criterion can be stated formally, the criterion is no longer monotone, so our framework cannot be used directly. However, under certain assumptions on the occurrences of outliers (similar to noise models), criteria that allow outliers and that are monotone can be devised. Since it is beyond the scope of this paper to discuss models for outliers, we do not discuss this possibility any further.

9 Conclusions

This paper presented a general framework that allows efficient and optimal segmentation using various different criteria separately or in combination. Our approach to segmentation is to specify criteria formally that should hold for any attribute within each segment of the segmentation. The resulting algorithms segment a trajectory with n edges optimally in $O(n \log n)$ time for many different criteria. The only requirements on the criteria posed are monotonicity and that there exist efficient algorithms to evaluate these criteria for a given subtrajectory. The property that makes a segmentation of a trajectory optimal in our paper, is that it subdivides the trajectory into as few pieces as possible. This ensures that the algorithm maximizes the length of each individual piece in a global fashion.

We comment that the framework extends directly to segmenting 3-dimensional trajectories. Attributes must be defined based on 3-dimensional coordinates of vertices, and the implementations of TEST and FURTHEST must be adapted to this end. For example, the disk criterion for location becomes a ball criterion. Also, curvature is a more complex attribute in 3-space, described by curvature and torsion [26]. On the other hand, the adaptations needed to segment on speed in the 3-dimensional case are trivial. In general, if TEST and FURTHEST have efficient implementations in the 3-dimensional case, then segmentation is efficient as well.

In this paper we segment a trajectory by partitioning it into as few pieces as possible. Interesting related tasks are computing segmentations with overlap and segmentations that allow for non-homogeneous pieces. We can adapt our segmentation technique to include non-homogeneous pieces by joining short segments and marking these as 'non-homogeneous'. We note that a recent paper [41] considers the problem of finding possibly overlapping pieces of a trajectory that are 'interesting' according to a given measure.

The application of our framework (which we presented using *abstract* spatiotemporal attributes) to domain specific tasks requires the conversion of semantically relevant characteristics of trajectories into attributes related to the ones we considered, but specific for the purpose. This needs to be done in close collaboration with domain experts and is a topic for further research.

Acknowledgments

Maike Buchin is supported by the Netherlands Organisation for Scientific Research NWO (project ACATA) and partially supported by EU Cost action IC0903 (MOVE). Anne Driemel is supported by the Netherlands Organisation for Scientific Research NWO (project RIMGA). Marc van Kreveld is partially supported by EU Cost action IC0903 (MOVE). Vera Sacristán is partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.

References

[1] AGARWAL, P. K., HAR-PELED, S., MUSTAFA, N. H., AND WANG, Y. Near-linear time approximation algorithms for curve simplification. *Algorithmica 42* (2005), 203–219. doi:10.1007/s00453-005-1165-y.

[2] ANAGNOSTOPOULOS, A., VLACHOS, M., HADJIELEFTHERIOU, M., KEOGH, E., AND YU, P. Global distance-based segmentation of trajectories. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (2006), pp. 34–43. doi:10.1145/1150402.1150411.

[3] BELLMAN, R. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM 4* (1961), 284. doi:10.1145/366573.366611.

[4] BENHAMOU, S. How to reliably estimate the tortuosity of an animal's path: straightness, sinuosity, or fractal dimension? *Journal of Theoretical Biology 229* (2004), 209–220. doi:10.1016/j.jtbi.2004.03.016.

[5] BOVET, P., AND BENHAMOU, S. Optimal sinuosity in central place foraging movements. *Animal Behaviour 42*, 1 (1991), 57–62. doi:10.1016/S0003-3472(05)80605-0.

[6] BRAUN, J. V., AND MÜLLER, H.-G. Statistical methods for DNA sequence segmentation. *Statistical Science 13* (1998), 142–162. doi:10.1214/ss/1028905933.

[7] BUCHIN, K., BUCHIN, M., AND GUDMUNDSSON, J. Detecting single file movement. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2008), pp. 288–297. doi:10.1145/1463434.1463476.

[8] BUCHIN, K., BUCHIN, M., GUDMUNDSSON, J., LÖFFLER, M., AND LUO, J. Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry and Applications, special issue on 19th International Symposium on Algorithms and Computation (ISAAC) 21*, 3 (2011), 253–282. doi:10.1142/S0218195911003652.

[9] BUCHIN, M., DRIEMEL, A., VAN KREVELD, M., AND SACRISTAN, V. An algorithmic framework for segmenting trajectories based on spatio-temporal criteria. In *Proceedings of the 18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems* (2010), pp. 202–211. 10.1145/1869790.1869821.

[10] CHUNDI, P., AND ROSENKRANTZ, D. J. *Segmentation of Time Series Data*, 2nd ed. Hershey: IGI Global, 2009, pp. 1753–1758. doi:10.4018/978-1-60566-010-3.ch267.

[11] CLARKSON, K., MEHLHORN, K., AND SEIDEL, R. Four results on randomized incremental constructions. *Computational Geometry: Theory and Applications 3* (1993), 185–212. doi:10.1016/0925-7721(93)90009-U.

[12] DARK, S. J., AND BRAM, D. The modifiable areal unit problem (maup) in physical geography. *Progress in Physical Geography 31* (2007), 471–479. doi:10.1177/0309133307083294.

[13] DE BERG, M., BOSE, P., BREMNER, D., RAMASWAMI, S., AND WILFONG, G. Computing constrained minimum-width annuli of point sets. In *Proceedings of the 5th International Workshop on Algorithms and Data Structures (WADS)* (1997), vol. 1272 of *LNCS*, pp. 392–401. doi:10.1007/3-540-63307-3_77.

[14] DE BERG, M., CHEONG, O., VAN KREVELD, M., AND OVERMARS, M. *Computational Geometry: Algorithms and Applications*, 3rd ed. Springer, Berlin, 2008.

[15] DODGE, S., WEIBEL, R., AND FOROOTAN, E. Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. *Computers, Environment and Urban Systems 33*, 6 (2009), 419–434. doi:10.1016/j.compenvurbsys.2009.07.008.

[16] FU, K., AND MUI, J. A survey on image segmentation. *Pattern Recognition 13*, 1 (1981), 3–16. doi:10.1016/0031-3203(81)90028-5.

[17] GUIBAS, L., KNUTH, D., AND SHARIR, M. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica 7*, 4 (1992), 381–413. doi:10.1007/BFb0032048.

[18] GURARIE, E., ANDREWS, R. D., AND LAIDRE, K. L. A novel method for identifying behavioural changes in animal movement data. *Ecology Letters 12*, 5 (2009), 395–408. doi:10.1111/j.1461-0248.2009.01293.x.

[19] HARGUESS, J., AND AGGARWAL, J. Semantic labeling of track events using time series segmentation and shape analysis. In *Proceedings of the 16th IEEE International Conference on Image Processing (ICIP)* (2009), pp. 4317–4320. doi:10.1109/ICIP.2009.5413671.

[20] HERMANN, S., AND KLETTE, R. A comparative study on 2D curvature estimators. In *Proceedings of the IEEE International Conference on Computing: Theory and Applications (ICCTA)* (2007), pp. 584–589. doi:10.1109/ICCTA.2007.2.

[21] HERSHBERGER, J., AND SURI, S. Off-line maintenance of planar configurations. *Algorithms 21* (1996), 453–475. doi:10.1006/jagm.1996.0054.

[22] HIMBERG, J., KORPIAHO, K., MANNILA, H., TIKANMÄKI, J., AND TOIVONEN, H. Time series segmentation for context recognition in mobile devices. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM)* (2001), pp. 203–210. doi:10.1109/ICDM.2001.989520.

[23] IRISSON, J.-O., LEVAN, A., LARA, M. D., AND PLANES, S. Strategies and trajectories of coral reef fish larvae optimizing self-recruitment. *Journal of Theoretical Biology 227*, 2 (2004), 205–218. doi:10.1016/j.jtbi.2003.10.016.

[24] JONSEN, I. D., FLEMMING, J. M., AND MYERS, R. A. Robust stateñspace modeling of animal movement data. *Ecology 86* (2005), 2874–2880. doi:10.1890/04-1852.

[25] LARSON, J. S., BRADLOW, E. T., AND FADER, P. S. An exploratory look at supermarket shopping paths. *International Journal of Research in Marketing 22*, 4 (2005), 395 – 414. doi:10.1016/j.ijresmar.2005.09.005.

[26] LEWINER, T., GOMES JR., J., LOPES, H., AND CRAIZER, M. Curvature and torsion estimators based on parametric curve fitting. *Computers & Graphics 29* (2005), 641–655. doi:10.1016/j.cag.2005.08.004.

[27] MANN, R., JEPSON, A., AND EL-MARAGHI, T. Trajectory segmentation using dynamic programming. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR)* (2002), pp. 331–334. doi:10.1109/ICPR.2002.1044709.

[28] MATOUŠEK, J., SHARIR, M., AND WELZL, E. A subexponential bound for linear programming. *Algorithmica 16* (1996), 498–516. doi:10.1007/BF01940877.

[29] MUELLER, J. An introduction to the hydraulic and topographic sinuosity indexes. *Annals of the Assoc. of American Geographers 58*, 2 (1968), 371–385. doi:10.1111/j.1467-8306.1968.tb00650.x.

[30] NATHAN, R., GETZ, W., REVILLA, E., HOLYOAK, M., KADMON, R., SALTZ, D., AND SMOUSE, P. A movement ecology paradigm for unifying organismal movement research. *Proceedings of the National Academy of Sciences 105* (2008), 19052–19059. doi:10.1073/pnas.0800375105.

[31] PLAZANET, C. Measurement, characterization and classification for automated line feature generalization. In *Proceedings of the ACSM/ASPRS Annual Convention and Exposition (Auto-Carto 12)* (1995), pp. 59–68.

[32] PREPARATA, F., AND SHAMOS, M. *Computational Geometry, an introduction*. Springer, Berlin, 1985.

[33] RAJAGOPALAN, V., RAY, A., SAMSI, R., AND MAYER, J. Pattern identification in dynamical systems via symbolic time series analysis. *Pattern Recognition 40*, 11 (2007), 2897–2907. doi:10.1016/j.patcog.2007.03.007.

[34] SZELISKI, R. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[35] TERZI, E., AND TSAPARAS, P. Efficient algorithms for sequence segmentation. In *Proceedings of the Sixth SIAM International Conference on Data Mining* (2006).

[36] VAN LOON, E., SACK, J.-R., BUCHIN, K., BUCHIN, M., DE BERG, M., VAN KREVELD, M., GUDMUNDSSON, J., AND MOUNTAIN, D. Results of the break-out group: Gulls Data. In *Dagstuhl Seminar 10491: Representation, Analysis and Visualization of Moving Objects* (2011), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany.

[37] VAN MOORTER, B., VISSCHER, D. R., JERDE, C. L., FRAIR, J. L., AND MERRILL, E. H. Identifying movement states from location data using cluster analysis. *The Journal of Wildlife Management 74*, 3 (2010), 588–594. doi:10.2193/2009-155.

[38] WELZL, E. Smallest enclosing disks (balls and ellipsoids). In *Symposium on New Results and Trends in Computer Science* (1991), vol. 555 of *LNCS*, pp. 359–370. doi:10.1007/BFb0038178.

[39] YAMANO, T., SATO, K., KAIZOJI, T., ROST, J.-M., AND PICHL, L. Symbolic analysis of indicator time series by quantitative sequence alignment. *Computational Statistics & Data Analysis 53*, 2 (2008), 486–495. doi:10.1016/j.csda.2008.08.033.

[40] YOON, H., AND SHAHABI, C. Robust time-referenced segmentation of moving object trajectories. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)* (2008), pp. 1121–1126. doi:10.1109/ICDM.2008.133.

[41] ZHOU, X., SHEKHAR, S., MOHAN, P., LIESS, S., AND SNYDER, P. K. Discovering interesting sub-paths in spatiotemporal datasets: A summary of results. In *19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2011), pp. 44–53.

[42] ZHOU, Y., AND HUANG, T. Bag of segments for motion trajectory analysis. In *Proceedings of the 15th IEEE International Conference on Image Processing (ICIP)* (2008), pp. 757–760. doi:10.1109/ICIP.2008.4711865.