

An Algorithmic Framework for Segmenting Trajectories based on Spatio-Temporal Criteria

Maike Buchin,^{*} Anne Driemel,[†]
Marc van Kreveld[‡]
Dep. of Information and Computing Sciences
Utrecht University
{maike,anne,marc}@cs.uu.nl

Vera Sacristán[§]
Dep. de Matemàtica Aplicada II
Universitat Politècnica de Catalunya
vera.sacristan@upc.edu

ABSTRACT

In this paper we address the problem of segmenting a trajectory such that each segment is in some sense homogeneous. We formally define different spatio-temporal criteria under which a trajectory can be homogeneous, including location, heading, speed, velocity, curvature, sinuosity, and curviness. We present a 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, the segmentation problem can generally be solved in $O(n \log n)$ time, where n is the number of edges of the trajectory to be segmented.

Categories and Subject Descriptors

F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems

General Terms

Algorithms

Keywords

Trajectory segmentation, geometric algorithms

^{*}Supported by the German Research Foundation (DFG) under grant number BU 2419/1-1.

[†]Supported by the Netherlands Organisation for Scientific Research NWO (project RIMGA).

[‡]Partially supported by EU Cost action IC0903 (MOVE).

[§]Partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.

1. INTRODUCTION

Due to technologies such as GPS and RFID 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 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, computing the similarity of the trajectories of several shoppers in a supermarket, and determining commuting patterns in the trajectories of people.

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 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

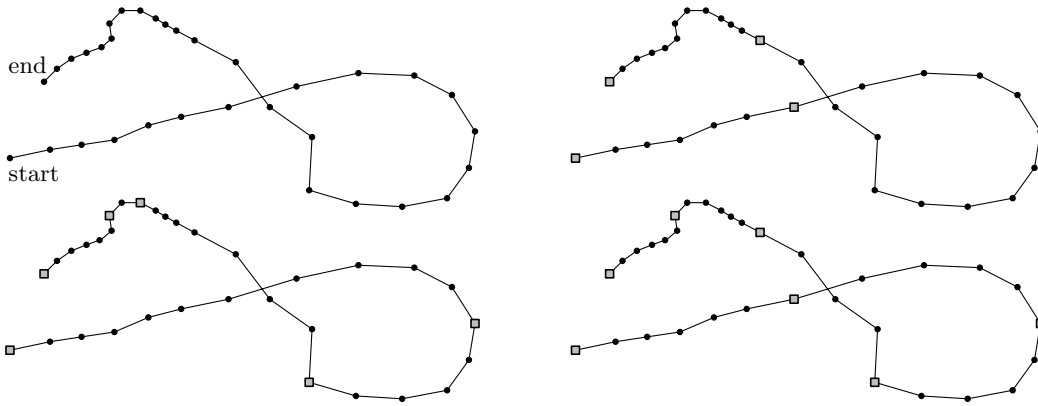


Figure 1: A trajectory and three segmentations of it, indicated by squares.

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).

Observe that using *relative values within a segment* is better than using *absolute* boundary values, like 10 km/h, 20 km/h, 50 km/h, and 100 km/h for speed (instead of a factor at most 2 between the highest and lowest speed within a segment). Using absolute values segments a trajectory that has speeds 51–51–98–98–103–103 km/h along its edges wrongly. Also, it would segment a trajectory with speeds 49–51–49–51 km/h along its edges (needlessly) into four segments.

Trajectories can be segmented either only at vertices or also in between vertices. We refer to the former as *discrete* and the latter as *continuous* segmentation. In this paper, we consider continuous segmentation because it can give fewer segments than discrete segmentation, see Figure 2. 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. Our results can be adapted (simplified) to discrete segmentation. Which form of segmentation is appropriate depends on the application: if no new vertices may be introduced in the segmentation process, then discrete segmentation is required; otherwise, continuous segmentation gives better results.

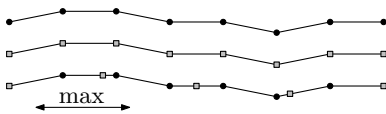


Figure 2: A trajectory and its segmentation by location. The arrow shows the maximum allowed separation of two points within one segment. Discrete segmentation requires seven segments and continuous segmentation requires four segments.

Several existing approaches segment moving object trajectories with the goal of efficient data handling and mining [1, 6, 14], rather than uniform movement properties. They consider trajectory segmentation using minimum bounding

boxes with the objective to realize efficient spatio-temporal range queries. Other approaches segment based on how well each segment fits with some polynomial [10], on spatio-temporal distances [16]. Computational techniques for trajectory segmentation include dynamic programming to find the minimum number of segments, and heuristics without performance guarantees as an alternative, due to the high computational cost of dynamic programming.¹

Our approach to the segmentation problem is the following. For each relevant movement characteristic we define an attribute function that specifies a value at every location on the trajectory. For instance, attributes can be speed, heading, and curvature. Then we define criteria that specify that within any single segment, the attribute values at any location 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 the similar attribute values, we minimize the number of segments used in the segmentation.

Overview. 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 many different criteria and combinations of these criteria. We prove that for monotone criteria we can use a greedy strategy for segmentation to obtain an optimal solution, and hence we can avoid dynamic programming. We show that for natural criteria relating to location, heading, speed, and velocity, efficient algorithms exist. In Section 5 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.

¹More related work exists in fields like video tracking, motion capture, and time-series analysis. Due to different objectives of the segmentation and lack of space in this paper, we will not review it here.

2. PRELIMINARIES

DEFINITION 1. A trajectory τ is a continuous mapping from a time interval $[t_0, t_n]$ to the plane (or a higher-dimensional space). For any time $t \in [t_0, t_n]$, we denote the location in the plane at time t by $\tau(t)$. 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 $\tau[t, t']$.

While a trajectory is the continuous motion path of a moving object, its representation usually follows from the way it is collected: a discrete sample of time-space positions.

DEFINITION 2. A piecewise-linear trajectory τ is a trajectory where for some sequence of time steps $t_0 < t_1 < \dots < t_n$, 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 $\frac{t_{i+1}-t}{t_{i+1}-t_i} \cdot \tau(t_i) + \frac{t-t_i}{t_{i+1}-t_i} \cdot \tau(t_{i+1})$.

Note that a subtrajectory of a trajectory is a trajectory itself, and the same holds true for piecewise-linear trajectories. In this paper we are only concerned with piecewise-linear trajectories, and from now on we use the term “trajectory” for any piecewise-linear trajectory.

Assume that a trajectory τ is defined by times t_0, t_1, \dots, t_n and their images $\tau(t_0), \tau(t_1), \dots, \tau(t_n)$. We define $v_i = \tau(t_i)$ for $0 \leq i \leq n$, and v_0, v_1, \dots, v_n are the *vertices* of τ . We define $e_i = \overline{v_{i-1}v_i}$ for $1 \leq i \leq n$ to be the *edges* of τ . Notice that if for some i , $v_{i-1} = v_i$, then e_i has zero length and the moving object was not moving in the time interval $[t_{i-1}, t_i]$. The speed of the moving object is constant on every edge, which follows from the definition of piecewise-linear trajectories.

An *optimal segmentation of a trajectory* is a segmentation into a minimum number of subtrajectories (segments) according to one or more criteria. Criteria typically concern attributes of the trajectory that are defined at any time t , and bounds on the relative values of these. 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. It is easy to see that a conjunction of monotone criteria is again monotone.

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. For instance, 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 (e.g., the difference in minimum and maximum speed is at most 20 km/h), or by bounding the ratio of the extreme values (e.g., at most a factor of 1.5 different). Our algorithmic framework to be described applies in case segmentation is done on one or more monotone criteria.

3. ATTRIBUTES AND CRITERIA

Next, we discuss some specific, basic attributes and corresponding criteria. These attributes are location, heading, and speed. We also discuss velocity, the combination of

heading and speed. For all, natural 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. Segmentation by the diameter criterion was shown in Figure 2.

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 East has angle 0 and purely North has angle $+\pi/2$. For trajectories represented by vertices and edges, the heading has a natural 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 natural criterion for heading is that in each segment, the heading lies within an angular range of some prespecified 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 2π ; a reasonable value might be $\pi/2$ 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 natural 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.

An example of a criterion that is not monotone is specifying that the standard deviation of the speed is below a given value in each segment. It is possible that the standard deviation decreases by extending a segment.

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, for which the shape is a sector of an annulus centered at O in the velocity vector plane. Figure 3 illustrates the two criteria.

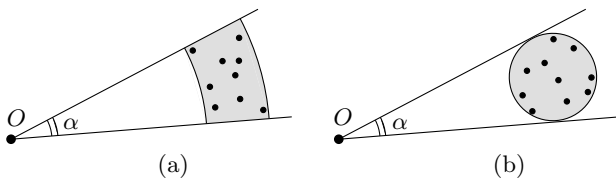


Figure 3: The points represent velocity vectors starting at O . (a) Speed and heading are bounded independently. (b) Speed and heading are bounded in conjunction with the disk criterion.

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. The segmentation we compute is continuous, that is, not necessarily at the vertices of τ .

- The criteria used are *monotone*.
- There is an $O(m)$ time test whether a subtrajectory $\tau[t_i, t_j]$ satisfies the criteria (where $m = j - i + 1$).
- If subtrajectory $\tau[t_i, t_{j-1}]$ satisfies the criteria 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 criteria.

4.1 Greedy strategy

A greedy strategy for segmentation starts at the start of the trajectory and makes the first segment as long as possible, until the segment would not satisfy some 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 criteria that are monotone, then a greedy method can be used to efficiently find an optimal solution. We show this next.

THEOREM 3. *Given a trajectory τ and a number of monotone criteria for segmentation that should all be satisfied, a greedy strategy yields an optimal solution to the segmentation problem.*

PROOF. Let $t_0 = a_0, \dots, a_k = t_n$ be the sequence of times where the greedy segmentation yields segment boundaries, and assume that an optimal segmentation has segment boundaries at times $t_0 = b_0, \dots, b_l = t_n$. 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 criteria 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. \square

On the one hand, it may be natural to have the ends of segments at the vertices of the trajectory only. On the other hand, this is a restriction on the segmentation that is not needed to obtain a simple and efficient algorithm. We will discuss both versions.

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 using one or more criteria has a rather simple structure. Starting at the end time s of the last completed segment (for the first segment $s = t_0$), we find the longest segment that satisfies the criteria by determining a vertex t_j so that $\tau[s, t_{j-1}]$ satisfies the criteria but $\tau[s, t_j]$ does not. If we restrict ourselves to segmenting at vertices only, then we end the current segment at t_{j-1} . Otherwise, we proceed by maximizing the part of the edge $[t_{j-1}, t_j]$ that still satisfies the criteria to determine the end time of the current segment.

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

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 $\text{TEST}(\tau[s, t_{i+1}])$, $\text{TEST}(\tau[s, t_{i+2}])$, \dots , until a test fails. If this happens for the test $\text{TEST}(\tau[s, t_j])$, then we run $\text{FURTHEST}(\tau[s, t_j])$.

The efficiency of this implementation depends on whether we can efficiently run $\text{TEST}(\tau[s, t_{i+a+1}])$ if we already know that $\text{TEST}(\tau[s, t_{i+a}])$ 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. In fact, using this method one can segment a trajectory with n edges optimally in time $O(n)$ with respect to range criteria of single-valued attributes, if FURTHEST takes $O(m)$ time on a subtrajectory of m edges.

For the disk and diameter criteria for location, this is not the case. We would need an algorithm to efficiently maintain 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 [8], 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 and combinations of 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.

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 $\text{TEST}(\tau[s, t_{i+a}])$. 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 $\text{TEST}(\tau[s, t_n])$. If successful, the whole remainder of the trajectory is the last segment and we stop.

If $a = 1$, we know $j = i + 1$. Else $a = 2^k$ for some $k \geq 1$ 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 $\text{FURTHEST}(\tau[s, t_j])$ to determine the latest time q on the edge $[t_{j-1}, t_j]$ such that the criteria for the segment up to q are still satisfied. The time q is then used as the new starting time s in the computation of the next segment.

```

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

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

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

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

```

Algorithm 1: Simple algorithmic framework for trajectory segmentation.

THEOREM 4. *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 at most linear in n).

PROOF. Suppose the optimal number of segments is k , let m_1, \dots, m_k be the numbers of edges (fully or partially) spanned by the segments which are computed by our algorithm. We can argue that $\sum_{i=1}^k m_i \leq 2n + (k - n) = n + k$ as follows. The number of segments spanned by a segment is at most one plus the number of vertices spanned. A vertex can be contained in at most two segments. There can be at most n segments that contain a vertex, the at most $k - n$ remaining segments each increase the sum by 1.

During the computation of one segment, TEST is called on values that are smaller than $2m$. The number of times it is called in the while loop is bounded by the maximal x such that $2^x \leq 2m$, i.e., $x \leq \log 2m$, 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 using values 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 in the end.

We conclude that the algorithm takes $\sum_{i=1}^k (T(2m_i) \log m_i + F(m_i))$ 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, $\sum_{i=1}^k (T(2m_i) \log m_i + F(m_i)) \leq T(n+k) \log(n+k) + F(n+k)$. Since $k = O(n)$ and $T(m)$ and $F(m)$ are at most polynomial, we have $T(n+k) \log(n+k) + F(n+k) = O(T(n) \log n + F(n))$. \square

The two routines TEST and FURTHEST depend on the actual criteria used to segment. For most of the monotone criteria mentioned above for location, heading, and speed, we can show that TEST and FURTHEST can be performed in linear time, and Theorem 4 yields $O(n \log n)$ time solutions for segmentation. Note that we do not need FURTHEST for attributes that are constant on edges.

Suppose that we require for each segment that the locations are within a disk of radius r , the headings are within an angular range with angle α , and the ratio of the maximum and minimum speed is at most s . Then $\text{TEST}(\tau[t_i, t_j])$ can easily be implemented in linear time using known, simple algorithms for smallest enclosing disk on the points v_i, \dots, v_j [4, 11, 15] and trivial algorithms for heading and speed. If the algorithm segments because of a too large difference in heading or a too large ratio in speed, then it segments at a vertex because heading and speed are constant over edges of τ . Otherwise, we need to use FURTHEST to determine the latest time that still satisfies the location criterion. First, assume that we use the disk criterion for location. Then a simple and efficient algorithm for FURTHEST 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 [3, 4, 5, 15]. Let P be the set of points p_1, \dots, p_m ; assume p_1, \dots, p_{m-1} fit inside some radius r disk but not p_1, \dots, p_m . Apply an affine 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 con-

tains 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 (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 , or it is realized by a disk that has two points on its boundary, see Figure 4. In the latter case, the two points on the boundary and the intersection point form a triangle that contains the center of the disk.

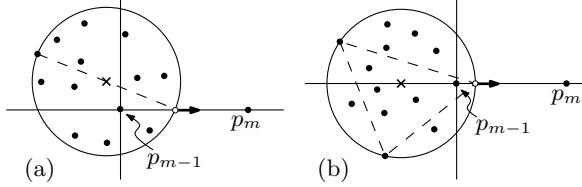


Figure 4: Optimal solution determined by (a) one point, or (b) two points. The arrow to p_m illustrates the optimization.

The code 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 [4].

```

// Input: A set  $P$  with  $m - 1$  points, and a radius  $r$ 
Let  $(p_1, \dots, p_{m-1})$  be the points of  $P$  in random order;
 $D_1 \leftarrow \text{OPTDISK}(r, p_1)$ ;
for  $h \leftarrow 2$  to  $m - 1$ 
{
  if  $D_{h-1}$  contains  $p_h$ 
    {  $D_h \leftarrow D_{h-1}$ ; }
  else  $D_h \leftarrow \text{OPTDISK}(r, p_1, \dots, p_{h-1})$  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: Randomized incremental construction for FURTHEST for the disk criterion for location.

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, \dots, 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 [4], Algorithm 2 takes linear expected time, where the expectation is only over the randomization performed within the algorithm (there are no difficult inputs).

LEMMA 5. *Given a subtrajectory $\tau[t_i, t_j]$ such that v_i, \dots, v_{j-1} fit in a disk of radius r but v_i, \dots, v_j do not fit in any disk of radius r , the problem of computing a disk of radius r*

that contains v_i, \dots, 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. We have observed that 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. \square

If we use the difference criterion for speed instead of the ratio criterion, only one simple test changes in the algorithm. If we use the diameter criterion for location instead of the disk criterion, then we can implement FURTHEST to run in $O(m)$ time, where $m = j - i$. We simply compute for each vertex (including $\tau(s)$) the furthest point on the edge e_j , which is still within distance d to this vertex and pick the closest over all of the computed points.

THEOREM 6. *An optimal segmentation using the disk or diameter criterion for location, 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.*

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 . The origin O represents $\vec{0}$. 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 5(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 exists 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 circles that contain a fixed point on their boundary while remaining twice tangent to some α -wedge, see Figure 5(c).

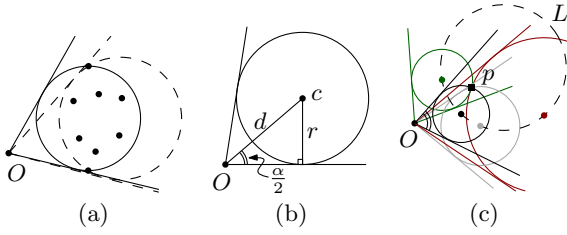


Figure 5: (a) Computing the tangents to the minimum enclosing circle is not guaranteed to give the wedge with smallest angle. (b) The radius r and the distance d of the circle center to O are directly related via $d \sin(\alpha/2) = r$. (c) The circles pivoting at a fixed point p while remaining tangent to an α -wedge have their center on the circle L .

LEMMA 7. *For any point p , the locus of the center points of circles that are twice tangent to an α -wedge that contains p is a circle.*

PROOF. For any circle 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 circles, see Figure 5(b). Hence, $p = (p_1, p_2)$ lies inside the circle 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 circle centered at the point $(\frac{p_1}{\cos^2(\alpha/2)}, \frac{p_2}{\cos^2(\alpha/2)})$ with radius $\sqrt{(p_1^2 + p_2^2) \frac{\sin^2(\alpha/2)}{\cos^4(\alpha/2)}}$. See Figure 5(c) for an illustration. \square

LEMMA 8. *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, \dots, 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 Lemma 7. We can compute these disks D_i, \dots, D_j in $O(m)$ time, where $m = j - i + 1$. The radius of a circle 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 5(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 $D_1 \cap \dots \cap D_n$ 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 they 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. \square

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

5. FURTHER ATTRIBUTES

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, see Remark 14 for details on how to deal with missing neighborhood information. There are three natural 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 gradually.

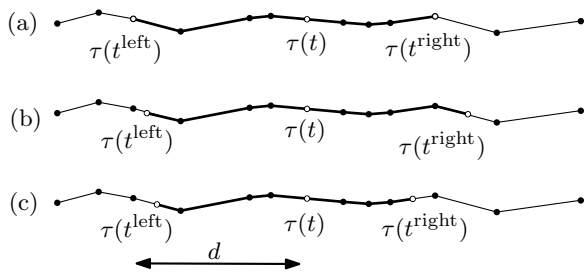


Figure 6: (a) Vertex, (b) space, and (c) time neighborhood of a point $\tau(t)$ on a trajectory which was sampled with regular time intervals.

An attribute which defines 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. 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 $\tau(t^{\text{right}})$ move forward along τ (unless the object was stationary at time t^{left} , t , or t^{right}). Since each of t^{left} , t , and t^{right} goes past the times t_0, \dots, 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 $3n$. We will see 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 .

When the segmentation uses a criterion based on an attribute that requires a neighborhood, we will preprocess the trajectory so that we can evaluate the attribute value at every time efficiently. We do this 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, \dots, 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, \dots, 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. Preprocessing is trivially done in linear time when the neighborhood of any point $\tau(t)$ involves only $O(1)$ vertices. When neighborhoods can involve many vertices, then it is still easy to perform the preprocessing in linear time by a scan over the trajectory with t^{left} , t , and t^{right} simultaneously, and maintaining suitable information about the analytic function on the way. We denote the analytic functions that define the attribute values along τ when $t \in [t_0, t_n]$ by $\phi_1(t), \dots, \phi_k(t)$. Here, $k \leq 3n$ is the number of edges of τ after adding the extra vertices. The time intervals over which the $\phi_0(t), \dots, \phi_k(t)$ 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 .

THEOREM 10. *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 $\phi_0(t), \dots, \phi_k(t)$, with $k = O(n)$. If for*

any $1 \leq i \leq k$, we can (i) evaluate $\phi_i(t)$ in time $O(1)$, (ii) compute the minima and maxima of $\phi_i(t)$ over the interval on which it is defined in time $O(1)$, and (iii) evaluate $\phi^{-1}(y)$ in time $O(1)$, then the optimal segmentation with respect to a range criterion of this attribute takes $O(n \log n)$ time after preprocessing, assuming there are at most $O(n)$ segments.

PROOF. Assumption (ii) implies that each $\phi_i(t)$ 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 extrema of ϕ causes ϕ to be monotone (increasing or decreasing) on each edge. The functions $\phi_1(t), \dots, \phi_k(t)$ can each be associated with $O(1)$ consecutive edges. With this adaptation, TEST can be used as before, and we need to evaluate ϕ at vertices of τ only 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}, \dots, t_{j-1} to obtain the maximum and minimum values realized. If function ϕ_j is increasing, then we use the minimum realized value on $\tau[s, t_{j-1}]$ to compute the maximum allowed value *Max* of ϕ_j . The correct result of FURTHEST is the time $\phi_j^{-1}(\text{Max})$. Symmetrically, if function ϕ_j is decreasing, then we use the maximum realized value on $\tau[s, t_{j-1}]$ to compute the minimum allowed value *Min* of ϕ_j . The correct result of FURTHEST is the time $\phi_j^{-1}(\text{Min})$. By assumption (iii), we can compute the inverse of ϕ_j in constant time.

Now, Theorem 4 implies that optimal segmentation takes $O(n \log n)$ time after preprocessing, using the same algorithm as before. \square

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.

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 [7, 9]. The commonly used methods 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 sample points from a constant-size neighborhood of this point, implicitly or explicitly. Three-point estimators define the curvature at a point p using the three points p_-, p, p_+ , where p_- is the starting point of the neighborhood of p , and p_+ is the location on τ where this neighborhood ends. We now discuss three definitions of this type, as described in [9], and how they can be used in our framework. Different definitions of curvature exist in differential geometry which are all equivalent. The corresponding definitions of discrete curvature are not equivalent, however.

Firstly, the curvature can be defined as the rate of change of the direction of the tangential vector. A first definition of discrete curvature therefore uses the turning angle of the

directed line segments connecting the three points:

$$\kappa(p) = \frac{\angle(p_-, p, p_+)}{\|p - p_-\| + \|p_+ - p\|}, \quad (1)$$

where $\angle(p_-, p, p_+) = \arccos \frac{\langle p - p_-, p_+ - p \rangle}{\|p - p_-\| \|p_+ - p\|}$ defines the turning angle.

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.

$$\kappa(p) = \frac{2|(p - p_-) \times (p_+ - p)|}{\|p_+ - p\| \cdot \|p - p_-\| \cdot \|p_- - p_+\|} \quad (2)$$

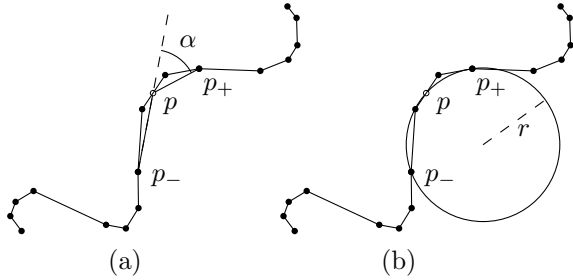


Figure 7: (a) Turning angle. (b) Osculating circle.

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 $\tau'(p_+) = p_+ - p$ and $\tau''(p_+) = (p_+ - p) - (p - p_-)$, plugged into the curvature formula:

$$\kappa(p) = \frac{\tau'(p_+) \times \tau''(p_+)}{\|\tau'(p_+)\|^3}. \quad (3)$$

The attribute functions $\phi_1(t), \dots, \phi_k(t)$ for discrete curvature according to all three definitions satisfy the three requirements stated in Theorem 10. When p_-, p , and p_+ are on some triple of edges, the time t defining p gives rise to a time t_- that defines p_- and linearly depends on t . The analogous statement is true for p_+ . 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 11. *An optimal segmentation using a range criterion for discrete curvature as defined in Equations (1), (2), or (3) 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 [12, 13], but also to describe trajectories of moving animals [2]. There appears to be no standard definition for sinuosity. If $\hat{\tau} = \tau[t^{\text{left}}, t^{\text{right}}]$ is the subtrajectory

that represents the neighborhood of $\tau(t)$, then some authors define the sinuosity at t as the arc length of $\hat{\tau}$ 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 $\hat{\tau}$. We will refer to this as *winding sinuosity*. Note that both definitions measure a slightly different winding behavior, as illustrated in Figure 8(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 $\hat{\tau}$ 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.

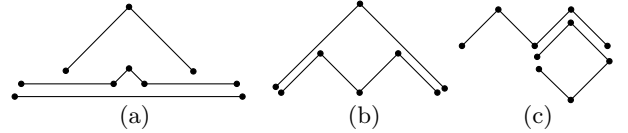


Figure 8: (a) Segments that are distinguished differently by winding and detour sinuosity. (b) Segments that are distinguished by curviness, but not by winding or detour sinuosity. (c) Segments that are distinguished by winding and detour sinuosity, but not by curviness.

COROLLARY 12. *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 10 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 $\hat{\tau}$ 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 only depends on the distance between the two endpoints of $\hat{\tau}$ 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 10. If we use time neighborhoods, the arc length of $\hat{\tau}$ is a linear function, since speed is constant on the edges of the trajectory. We can apply Theorem 10 since the composition of a linear function with the discussed functions still satisfies the conditions. \square

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{ov} and \vec{ow} . Again, this definition measures a different winding behavior than sinuosity, a fact illustrated in Figure 8 (b) and (c). The proof of the following corollary is analogous to the proof of Corollary 12.

COROLLARY 13. *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.*

REMARK 14. *At the beginning and end of the trajectory some of the neighborhood information will be missing. There are several ways to deal with this. One can extrapolate the curve based on fitting tangents or circles; alternatively, one could simply disregard the points with missing neighborhood information, which only occur near $\tau(t_0)$ and $\tau(t_n)$ if neighborhoods are small. In the limit case a point will still have a one-sided neighborhood, in which case we have to make some adaptations. The same is true if we take time or vertex neighborhoods and the object is stationary for some time.*

6. CONCLUSIONS

This paper presented a 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. Hence, no artificial structures like bounding boxes are needed, and the result is guaranteed to satisfy the criteria and be optimal.

The only algorithmic technique that is needed is randomized incremental construction [3, 4, 5], a very simple technique that can be used for all problems that are LP-type (among which smallest enclosing disk for a set of points). We identify two new LP-type problems that arise in optimal segmentation, which allows the randomized incremental construction to be used. It leads to $O(n \log n)$ time segmentation for a trajectory with n vertices in many cases. Our approach does not handle outliers, these must be handled in preprocessing.

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 [9].

The application of our framework (in which we presented abstract, spatio-temporal attributes) to real applications requires the conversion of semantically relevant characteristics of trajectories into attributes related the ones we considered, but specific for the application. How this can be done is a topic for further research.

7. REFERENCES

- [1] A. Anagnostopoulos, M. Vlachos, M. Hadjieleftheriou, E. Keogh, and P. Yu. Global distance-based segmentation of trajectories. In *Proc. 12th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining (KDD)*, pages 34–43, 2006.
- [2] P. Bovet and S. Benhamou. Optimal sinuosity in central place foraging movements. *Animal Behaviour*, 42(1):57–62, 1991.
- [3] K. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. *Comput. Geom.*, 3:185–212, 1993.
- [4] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, 3rd edition, 2008.
- [5] L. Guibas, D. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7(4):381–413, 1992.
- [6] M. Hadjieleftheriou, G. Kollios, V. Tsotras, and D. Gunopulos. Efficient indexing of spatiotemporal objects. In *Proc. 8th Int. Conference on Extending Database Technology EDBT*, volume 2287 of *Lecture Notes in Computer Science*, pages 251–268, 2002.
- [7] S. Hermann and R. Klette. A comparative study on 2D curvature estimators. In *Proc. IEEE Int. Conference on Computing: Theory and Applications (ICCTA)*, pages 584–589, 2007.
- [8] J. Hershberger and S. Suri. Off-line maintenance of planar configurations. *J. Algorithms*, 21:453–475, 1996.
- [9] T. Lewiner, J. Gomes Jr., H. Lopes, and M. Craizer. Curvature and torsion estimators based on parametric curve fitting. *Computers & Graphics*, 29:641–655, 2005.
- [10] R. Mann, A. Jepson, and T. El-Maraghi. Trajectory segmentation using dynamic programming. In *Proc. 16th Int. Conference on Pattern Recognition (ICPR)*, pages 331–334, 2002.
- [11] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. In *Proc. 8th annual ACM Symposium on Computational Geometry*, pages 1–8, 1992.
- [12] J. Mueller. An introduction to the hydraulic and topographic sinuosity indexes. *Annals of the Assoc. of American Geographers*, 58(2):371–385, 1968.
- [13] C. Plazanet. Measurement, characterization and classification for automated line feature generalization. In *Proc. of Auto-Carto 12*, pages 59–68, 1995.
- [14] S. Rasetic, J. Sander, J. Elding, and M. Nascimento. A trajectory splitting model for efficient spatio-temporal indexing. In *Proc. 31st Int. Conference on Very Large Data Bases (VLDB)*, pages 934–945, 2005.
- [15] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In *New Results and Trends in Computer Science*, volume 555 of *Lecture Notes in Computer Science*, pages 359–370, 1991.
- [16] H. Yoon and C. Shahabi. Robust time-referenced segmentation of moving object trajectories. In *Proc. 8th IEEE Int. Conference on Data Mining (ICDM)*, pages 1121–1126, 2008.