Approximate Hotspots of Orthogonal Trajectories

Ali Gholami Rudi

Abstract

In this paper we study the problem of finding hotspots of polygonal two-dimensional trajectories, i.e. regions in which a moving entity has spent a significant amount of time. The fastest optimal algorithm, due to Gudmundsson, van Kreveld, and Staals (2013), finds an axis-parallel square hotspot of fixed side length in $O(n^2)$. We present an approximation algorithm with the time complexity $O(n \log n)$ and approximation factor $1/4$ for orthogonal trajectories, in which the entity moves in a direction parallel either to the $x$ or to the $y$-axis. We also present a $1/4$-approximation algorithm for finding axis-parallel cube hotspots of fixed side length for orthogonal three-dimensional trajectories.

Keywords: Trajectory, Hotspot, Geometric algorithms

1 Introduction

Tracking technologies like GPS gather huge and growing collections of trajectory data, for instance for cars, mobile devices, and animals. The analysis of these collections poses many interesting problems, which has been the subject of much attention recently [1]. One of these problems is the identification of the region, in which an entity has spent a large amount of time. Such regions are usually called stay points, popular places, or hotspots in the literature.

We study polygonal trajectories, in which the trajectory is obtained by linearly interpolating the locations of the moving entity, recorded at specific points in time (this model, also called piecewise-linear trajectories [2], is very common in the literature [3]). For this model, Gudmundsson et al. define several problems about trajectory hotspots and present an $O(n^2)$ algorithm.

*Department of Electrical and Computer Engineering, Babol Noshirvani University of Technology, Babol, Iran. Email: gholamirudi@nit.ac.ir.
to solve the following [4]: defining a hotspot as an axis-aligned square of fixed side length, we wish to find the placement of such a square that maximizes the time the entity spends inside it (there are other models and assumptions about hotspots, for a brief survey of which, the reader may consult [4]; e.g. the assumption of pre-defined potential regions [5], counting only the number of visits or the number of visits from different entities [6], or based on the sampled locations only [7]). To solve this problem, they first show that the function that maps the location of the square to the duration the trajectory spends inside it, is piecewise linear and its break points happen when a side of the square lies on a vertex, or a corner of the square on an edge of the trajectory. Based on this observation, they subdivide the plane into $O(n^2)$ faces and test each face for the square with the maximum duration.

Limiting ourselves to orthogonal trajectories, in which each edge is parallel to an axis of the coordinate system, in this paper we present an $O(n \log n)$ time approximation algorithm. Compared to the time the entity spends in the optimal placement of the square, the entity spends no less than $1/4$ of that time in the square returned by our algorithm. It sweeps two parallel lines to find the best square, after decomposing the trajectory. Unlike Gudmundsson et al.’s algorithm, extending which to three-dimensional trajectories seems nontrivial, an extension of the algorithm presented in this paper can find approximate, axis-parallel, cube hotspots of trajectories in $\mathbb{R}^3$ with approximation factor $1/4$ and the time complexity $O(n^2 \log n)$.

The rest of this paper is organized as follows. In Section 2 we describe our model in more detail and introduce the notation used in this paper. We also prove results that we use in Section 3, in which we present our algorithm for finding approximate hotspots of orthogonal trajectories. In Section 4 we discuss three-dimensional trajectories, and finally, in Section 5 we conclude this paper.

2 Preliminaries and Basic Results

A trajectory specifies the location of a moving entity through time. Therefore, it can be described as a function that maps each point in a specific time interval to a location in the plane. Similar to Gudmundsson et al. [4], we assume that trajectories are continuous and piecewise linear. The location of the entity is recorded at discrete time intervals, which we call the vertices of a trajectory. We assume that the entity moves in a straight line and with constant speed from a vertex to the next (this simplifying assumption is
very common in the literature but there are other models for the movement of the entity between vertices \([8]\); we call the sub-trajectory connecting two contiguous vertices, an edge of the trajectory.

In this paper, we relax the requirement that a trajectory is continuous. We assume that a trajectory \(T\) is a set of edges \(\{e_1, e_2, ..., e_n\}\). For an edge \(e\), we associate a weight \(w_e\), which denotes the duration of the sub-trajectory through its end points (the difference between the time recorded for its end points). We also denote the left and the right vertex of an edge with \(e_s\) and \(e_e\) respectively. In orthogonal trajectories, all trajectory edges are parallel either to the \(x\)-axis or to the \(y\)-axis. In horizontal (similarly vertical) trajectories all edges are parallel to the \(x\)-axis (\(y\)-axis).

For any axis-parallel square \(r\) with some fixed side length, we define the weight of \(r\) as the total duration in which the entity has spent inside it and denote it with \(w_T(r)\), or if there is no confusion \(w(r)\). A hotspot is an axis-parallel square with side length \(s\) and with the maximum possible weight. We denote the weight of a hotspot of trajectory \(T\) with \(h_s(T)\).

**Theorem 2.1.** Let \(H\) and \(V\) be a partition of an orthogonal trajectory \(T\), in which \(H\) contains the horizontal edges and \(V\) contains the vertical edges of \(T\). Let \(w\) be the maximum of \(h_s(H)\) and \(h_s(V)\). Then, \(w\) is at least \(h_s(T)/2\).

**Proof.** Let \(r\) be a hotspot in \(T\). Every edge of \(T\) is either in \(H\) or in \(V\) and thus \(w_H(r) + w_V(r)\) equals \(h_s(T)\). Therefore, either \(w_H(r) \geq h_s(T)/2\) or \(w_V(r) \geq h_s(T)/2\). Since \(h_s(H) \geq w_H(r)\) and \(h_s(V) \geq w_V(r)\), we have \(\max(h_s(H), h_s(V)) \geq h_s(T)\) as required.

Let \(r\) be an axis-parallel square and \(T\) be a horizontal trajectory. The entering rate of an edge \(e\) of \(T\) with respect to \(r\) is the rate at which the contribution of the weight of the edge to the weight of \(r\) increases, if the right side of \(r\) is moved to the right. Similarly, the leaving rate of an edge \(e\) with respect to \(r\) is the rate at which the contribution of the weight of the edge to the weight of \(r\) decreases, if the left side of \(r\) is moved to the right. We denote the former as \(r_+(e)\) and the latter as \(r_-(e)\). It is not difficult to see that \(r_+(e)\) (and similarly \(r_-(e)\)) is either zero or the ratio of its duration to its length, which we denote as \(d(e)\). In Figure 1 except the entering rate of \(b\) and \(e\), and the leaving rate of \(b\) and \(c\), the entering and leaving rates of all edges are zero.

The entering rate of horizontal trajectory \(T\) with respect to square \(r\), denoted as \(r_+(T)\), is defined as the sum of the entering rate of all edges of \(T\). Similarly, the leaving rate of trajectory \(T\) with respect to square \(r\) is the sum of the leaving rate of all edges of \(T\); this is denoted as \(r_-(T)\).
Lemma 2.2. Let $T$ be a horizontal trajectory. There exists a square with side length $s$, whose weight equals $h_s(T)$ and one of its vertical sides contains a vertex of $T$.

Proof. Let $r$ be the region with weight $h_s(T)$ and suppose none of its vertical sides contains a vertex of $T$. If $r_+(T) \geq r_-(T)$, we move $r$ to the right and otherwise we move it to the left until one of the sides of $r$ is on a vertex of $T$. Clearly, the weight of the square could not have decreased and is at least $h_s(T)$, completing the proof. \[\square\]

Theorem 2.3. Let $T$ be a horizontal trajectory and let $h$ be the maximum weight of a square with side length $s$, one of whose corners coincides with one of the vertices of $T$. Then, $h \geq h_s(T)/2$.

Proof. Let $r$ be the square with weight $h_s(T)$, one of whose sides contains a vertex $v$ of $T$ (such a square surely exists, as shown in Lemma 2.2). Suppose $v$ is on the left side of $r$ (the argument for the right side is similar). This is demonstrated in Figure 2. Let $r'$ and $r''$ be the squares with side length $s$, whose lower left and upper left corners are on $v$ respectively. Given that the union of $r'$ and $r''$ covers $r$, $w(r') + w(r'')$ is at least $h_s(T)$ and therefore $\max(w(r'), w(r''))$ is at least $h_s(T)/2$. Since $h \geq \max(w(r'), w(r''))$, we have $h \geq h_s(T)/2$. \[\square\]
Figure 2: A hotspot, on whose left side there is a trajectory vertex

3 A 1/4 Approximation Algorithm

In this section we present an approximation algorithm that, given an orthogonal trajectory $T$, finds an axis-aligned square, whose weight is at least $h_s(T)/4$. We start with Algorithm 3.1 which finds the square with the maximum weight among those whose lower right or upper right corners are on one of the vertices of the given horizontal trajectory. The algorithm assumes that the $x$ coordinate of all vertices is at least 0; otherwise the trajectory may be shifted in the positive direction of the $x$-axis. Also, the algorithm uses the Fenwick tree data structure, supporting the computation of the prefix-sum of a sequence of $n$ numbers and updating any of them in $O(\log n)$ \cite{9}. For a Fenwick tree $f$ with $n$ elements, $\text{Add}(f, i, c)$ increases the value of the $i$-th element by $c$, $\text{Sum}(f, i, j)$ returns the sum of the elements $i$ through $j$, and $\text{Get}(f, i)$ returns the value of the $i$-th element. The algorithm maintains three Fenwick trees, each with size $n$: fixed for the fixed contributions of the edges to the weight of containing squares and entering (similarly leaving) for the entering (leaving) rate of the edges with respect to containing squares (the algorithm moves two parallel sweep lines and both vertical sides of the squares considered in the algorithm are on these lines).

One of the key ideas in Algorithm 3.1 is to maintain only the current
Algorithm 3.1: RightCornerHotspots(T, s)

**Input**: A horizontal trajectory T and length s; n is the number of the edges of T.

**Output**: An axis-parallel square with side length s and with one of its right corners on a vertex of T, with the maximum weight.

*fixed, entering, leaving*: n-element Fenwick trees, with all elements zero initially.

Sort the edges of T increasingly by the value of their y coordinate to obtain the sequence $e_1, e_2, ..., e_n$ (note that all edges are horizontal and the height of some of the edges may be equal).

Move two parallel vertical sweep lines horizontally to the right, one at $x = r_e$ and one at $x = r_s$, in which $r_s = r_e + s$.

**for each event**, i.e., a vertex v of T meeting any of these sweep lines (suppose v is an endpoint of $e_i$) **do**

if v is the left vertex of $e_i$ and is on $x = r_s$ then
  Add(entering, i, $d(e_i)$)
  Add(fixed, i, $-x \cdot d(e_i)$)

if v is the left vertex $e_i$ and is on $x = r_e$ then
  Add(entering, i, $-d(e_i)$)
  Add(fixed, i, $s \cdot d(e_i) - x \cdot d(e_i)$)

if v is the right vertex $e_i$ and is on $x = r_s$ then
  Add(leaving, i, $d(e_i)$)
  Add(fixed, i, $x \cdot d(e_i)$)

if v is the right vertex $e_i$ and is on $x = r_e$ then
  Add(leaving, i, $-d(e_i)$)
  Add(fixed, i, $-s \cdot d(e_i) - x \cdot d(e_i)$)

Find the maximum value of $j \geq i$ such that the difference in the height of $e_i$ and $e_j$ is no more than s (this can be done with a simple binary search). The weight of the square whose lower right corner is at v is:

$$\text{Sum(fixed, i, j)} + (\text{Sum(entering, i, j)} - \text{Sum(leaving, i, j)}) \cdot x$$

Record this as the best square, if this weight is the maximum so far.

Do likewise for the square whose upper right corner is at v (for this case, the minimum value of $j$ should be found such that $j \leq i$ and the difference between the height of $e_i$ and $e_j$ is at most s).

**return** the square with the maximum weight.
entering and leaving rates of the edges and assume that each entering or leaving interval has begun from \( x = 0 \). Therefore, to compute the total contribution of entering and leaving edges, the algorithm simply multiplies the sum of their rate with the current value of \( x \). To compensate for the intervals included in this computation before the actual entering or leaving intervals, the algorithm updates the fixed Fenwick tree. This frees the algorithm from storing a different starting position for the entering and leaving rates of each of the edges, which would make the computation of the weight of square \( r \) more complex. The correctness of this algorithm is shown in Theorem 3.1.

**Theorem 3.1.** Among all axis-parallel squares with side length \( s \) and with a right corner on a vertex of a horizontal trajectory \( T \), Algorithm 3.1 finds the square with the maximum weight with time complexity \( O(n \log n) \).

**Proof.** The algorithm maintains the entering and leaving rates of all edges in respect to any square \( r \) that contains them and its left and right sides are on \( r_s \) and \( r_e \) respectively: the entering rate of an edge \( e_i \) is increased by \( d(e) \) as its left vertex meets \( r_s \) and it is reset (decreased by \( d(e) \)) when it meets \( r_e \) (therefore, \( \text{Get(entering, } i \text{)} \) is always \( \text{r_e(e_i)} \)). The leaving rate of the edges is updated similarly. Let \( r \) be the square considered in the loop and \( e = e_k \), such that \( k \) is in the interval from \( i \) through \( j \). To show that the
algorithm finds the weight of \( r \) correctly, we show that the contribution of edge \( e \) to the weight of \( r \) is \( \text{contrib}(e) = (\text{Get}(\text{entering}, k) - \text{Get}(\text{leaving}, k)) \cdot x + \text{Get}(\text{fixed}, k) \).

There are five cases to consider regarding the relative position of an edge and the two sweep lines. These cases are demonstrated in Figure 3: an edge may be outside \( (e_5) \) or inside \( (e_2) \) the region bounded by the two sweep lines, or it may intersect the right sweep line \( (e_3) \), the left sweep line \( (e_1) \), or both \( (e_4) \).

It is not difficult to show that the contribution of an edge \( e_i \) to the weight of \( r \) is equal to \( \text{contrib}(e) \) in all five cases. If \( e_i \) is outside the region bounded by the two sweep lines, the entering and leaving rates and \( \text{Get}(\text{fixed}, k) \) are all zero. When \( e_i \) intersects only the right sweep line \( (r_e) \), as \( e_3 \) in Figure 3, \( \text{contrib}(e) = x \cdot d(e) - x(e_s) \cdot d(e) \), in which \( x(p) \) is the value of the \( x \) coordinate of point \( p \). This clearly is equal to the contribution of the edge \( e \) to the weight of any containing square \( r \) whose vertical sides are on \( r_s \) and \( r_e \). We omit other cases for brevity.

The following theorem shows how we can use Algorithm 3.1 and the theorems proved in the previous section to obtain an approximation algorithm for finding square hotspots of fixed side length for orthogonal trajectories.

**Theorem 3.2.** There is an approximation algorithm for finding hotspots (axis-parallel squares with side length \( s \)) of orthogonal trajectories, such that the weight of the square found by the algorithm is at least \( 1/4 \) of the optimal value \( h_s(T) \).

**Proof.** Let \( T \) be an orthogonal trajectory. \( T \) can be partitioned into sets \( V \) and \( H \) containing the vertical and horizontal edges of \( T \) respectively. Then, Algorithm 3.1 finds a square \( r_H \) with the maximum possible weight, in which one of its corners is on a vertex of \( H \) (Algorithm 3.1 can be performed twice, once after rotating the plane 180 degrees to find the maximum-weight squares with one of its left corners on a vertex of \( H \)). The same algorithm can obtain a square \( r_V \) with the maximum possible weight for \( V \), after rotating the plane 90 degrees. By Theorem 2.3, \( w(r_H) \geq h_s(H)/2 \) and \( w(r_V) \geq h_s(V)/2 \). Also, by Theorem 2.1, \( \max(h_s(H), h_s(V)) \geq h_s(T)/2 \) implying that \( \max(w(r_H), w(r_V)) \geq h_s(T)/4 \), as required.

### 4 Extension to Three Dimensions

The algorithm presented in Section 3 can be extended to find approximate, axis-parallel, cube hotspots of fixed side length for orthogonal trajectories.
in \( \mathbb{R}^3 \), as will be explained in this section. We first extend the definitions and notations presented in Section 2 to \( \mathbb{R}^3 \). The weight of a cube \( c \) with side length \( s \) in respect to trajectory \( T \) in \( \mathbb{R}^3 \) is the total duration in which the entity spends inside it; we represent it as \( w_T(c) \), as before. A hotspot of a trajectory \( T \) in \( \mathbb{R}^3 \) is an axis-parallel cube (i.e. a cube whose faces are parallel to the planes defined by any pair of the axes of the coordinate system) of fixed side length \( s \) and the maximum weight, \( h_s(T) \).

Let \( e \) be an edge parallel to the \( z \)-axis and let \( c \) be an axis-parallel cube. Exactly two faces of \( c \) are parallel to the \( xy \)-plane, \( Z_1(c) \) and \( Z_2(c) \), with \( Z_1(c) \) appearing first (in the positive direction of the \( z \)-axis). The entering rate of \( e \) with respect to \( c \), denoted as \( c_{z^+}(e) \), is the rate at which the contribution of the weight of \( e \) to the weight of \( c \) increases if \( Z_2(c) \) is moved to the right. Similarly, the leaving rate of \( e \) with respect to \( c \), denoted as \( c_{z^-}(e) \), is the rate at which the contribution of the weight of \( e \) to the weight of \( c \) decreases if \( Z_1(c) \) is moved to the right. As in the 2-dimensional case, \( c_{z^-}(e) \) or \( c_{z^+}(e) \) are either zero or the ratio of the duration of \( e \) to its length, which we denote as \( d(e) \). We define \( c_{z^+}(T) \) (similarly \( c_{z^-}(T) \)) for orthogonal trajectory \( T \) as the sum of the entering (leaving) rates of all edges of \( T \) that are parallel to the \( z \)-axis. The following theorem extends Theorem 2.2 to three dimensions.

**Theorem 4.1.** For a 3-dimensional orthogonal trajectory \( T \), there exists an axis-parallel hotspot of side length \( s \) such that a vertex of \( T \) is on one of the two planes formed by extending its \( xy \)-parallel faces.

**Proof.** Let \( c \) be an axis-parallel cube with side length \( s \) and weight \( h_s(T) \). If no vertex of \( T \) appears on the planes formed by extending faces \( Z_1(c) \) and \( Z_2(c) \) (the \( xy \)-parallel faces of \( c \)), we move \( c \) in a direction parallel to the \( z \)-axis: if \( c_{z^+}(T) \geq c_{z^-}(T) \), in the positive direction, and otherwise, in the negative, until a vertex of \( T \) is on one of these planes. Clearly the weight of \( c \) in its new position has not decreased (note that the weight contribution of the edges of \( T \) that are not parallel to the \( z \)-axis has not change). Therefore, to find a hotspot of \( T \), it suffices to search among the cubes with a vertex of \( T \) on one of the \( xy \)-parallel planes containing its \( xy \)-parallel faces. This observation suggests Algorithm 4.1.

Given that Algorithm 4.1 uses the algorithm presented in Theorem 3.2, it finds approximate hotspots with approximation ratio 1/4 and with the time complexity \( O(n^2 \log n) \).
Algorithm 4.1: ApproximateHotspots3D($T, s$)

**Input**: A horizontal trajectory $T$ and length $s$; $n$ is the number of the edges of $T$.

**Output**: An axis-parallel cube $c$ with side length $s$, with one vertex of $T$ on one of the planes formed by extending $Z_1(c)$ or $Z_2(c)$.

Sort the vertices of $T$ increasingly by the value of their $z$ coordinate to obtain $\sigma$.

for each vertex $v = \sigma_i$ of $\sigma$ do

Let $z(v)$ be the $z$ coordinate of $v$. Project all edges that are (maybe partially) between $z = z(v)$ and $z = z(v) + s$ to the plane $z = z(v)$ to obtain an orthogonal 2-dimensional trajectory $T'$. Edges parallel to the $z$-axis are projected to an edge with length zero, whose weight denotes the duration of the portion between $z = z(v)$ and $z = z(v) + s$.

Perform the algorithm explained in Theorem 3.2 on $T'$ to obtain a square $s$.

Let $c$ be the cube with $Z_1(c)$ on $s$. It is not difficult to see that $w_T(c)$ is equal to $w_{T'}(s)$. Record $c$, if it has the maximum weight so far.

Repeat the preceding steps after reversing the direction of the $z$-axis to find cubes like $c$, with $Z_2(c)$ on a vertex of $T$.

**return** the cube with the maximum weight.
5 Discussion

In this paper we presented a $1/4$-approximation algorithm for finding axis-parallel square hotspots of fixed side length for orthogonal trajectories in $\mathbb{R}^2$ and extended it to find axis-parallel cube hotspots of fixed side length for orthogonal trajectories in $\mathbb{R}^3$. It may be possible to improve the approximation factor, by combining, instead of partitioning the edges of a trajectory and choosing their maximum weight hotspot, in Theorem 2.1. Also, our algorithm may lead to an approximation algorithm for non-orthogonal trajectories by replacing non-orthogonal edges with few orthogonal edges.

References

[1] Y. Zheng. Trajectory data mining - an overview. *ACM Transactions on Intelligent Systems and Technology*, 6(3):29:1–29:41, 2015.

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

[3] B. Aronov, A. Driemel, M. J. van Kreveld, M. Löffler, and F. Staals. Segmentation of trajectories on nonmonotone criteria. *ACM Transactions on Algorithms*, 12(2):26:1–26:28, 2016.

[4] J. Gudmundsson, M. J. van Kreveld, and F. Staals. Algorithms for hotspot computation on trajectory data. In *SIGSPATIAL/GIS*, pages 134–143, 2013.

[5] L. O. Alvares, V. Bogorny, B. Kuijpers, J. A. F. de Macêdo, B. Moelans, and A. A. Vaisman. A model for enriching trajectories with semantic geographical information. In *ACM International Symposium on Geographic Information Systems*, page 22. ACM, 2007.

[6] M. Benkert, B. Djordjevic, J. Gudmundsson, and T. Wolle. Finding popular places. *International Journal of Computational Geometry and Applications*, 20(1):19–42, 2010.

[7] S. Tiwari and S. Kaushik. Mining popular places in a geo-spatial region based on gps data using semantic information. In *Workshop on Databases in Networked Information Systems*, pages 262–276. Springer, 2013.
[8] H. J. Miller. Modelling accessibility using space-time prism concepts within geographical information systems. *International Journal of Geographical Information Science*, 5(3):287–301, 1991.

[9] P. M. Fenwick. A new data structure for cumulative probability tables - an improved frequency-to-symbol algorithm. *Software, Practice and Experience*, 26(4):489–490, 1996.