New Results on Trajectory Grouping under Geodesic Distance*

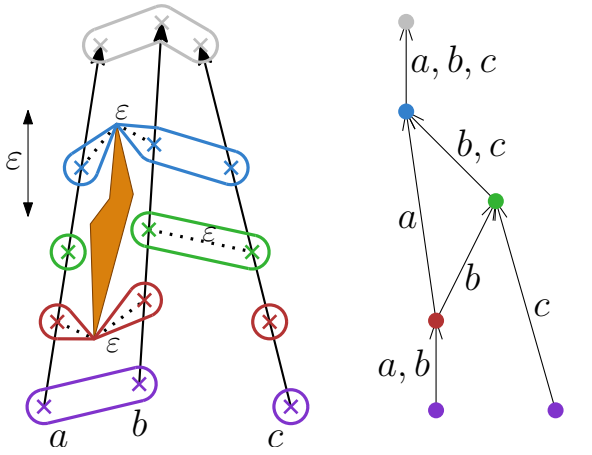
Maarten Löffler† Frank Staals‡ Jérôme Urhausen§

Abstract

We study grouping of entities moving amidst obstacles, extending the recent work of Kostitsyna et al. [5]. We present an alternative algorithm that can compute the Reeb-graph, a graph which captures when and how the partition of the entities into groups changes, when the entities move amidst arbitrary polygonal obstacles. Our new algorithm is significantly faster than the algorithm of Kostitsyna et al. when the number of entities is significantly larger than the total complexity of the obstacles. Furthermore, we consider a restricted setting in which the obstacles are big compared to ε: the parameter determining when entities are close enough together to be in the same group. We show that in this setting the Reeb-graph is much smaller, and we can compute it much faster, than in the case of general obstacles.

1 Introduction

In recent years, trajectory analysis has become a popular and well studied topic in computational geometry [1, 2, 3, 5]. We consider the problem of finding all (maximal) *groups* from the trajectory data. Intuitively, a group is a sufficiently large set of entities that travel together for a sufficiently long time. Buchin et al. [2] formalize this notion of groups, and show how to compute all *maximal* groups efficiently. A group is said to be maximal if the time interval on which the entities are together is maximal in length, and there is no group that contains it and stays together during the same time interval. Recently, Kostitsyna et al. [5] significantly extended the work of Buchin et al. by considering the environment in which the entities move. In particular, they study grouping when the entities move amidst various types of obstacles (see Table 1). So, when we decide if two entities are close enough together, we measure the distance using the geodesic distance (i.e. the length of the smallest obstacle-avoiding path) rather than the Euclidean distance. We continue the work of Kostitsyna et al. in two ways. First, we present an improved algorithm for the case in which the entities move amidst arbitrary obstacles, but their total complexity m is small compared to the number of entities n. Second, we consider a new environment setting, in which the obstacles are "large" (but may be arbitrarily complex and close together). This allows us to give a much faster algorithm than in the case of arbitrary obstacles. Next, we present the required notation and definitions following Buchin et al. [2] and Kostitsyna et al. [5], and formally define our problem, so that we can state our results more precisely.



(a) Entities a, b and c move along a linear trajectory around an obstacle. The colors of the entities indicate their positions at important moments. The circular forms indicate the groups at those moments.

(b) The Reeb graph for entities a, b, and c. The colors of the vertices correspond to the colors used in the figure on the left in order to indicate timestamps.

Figure 1: Example of entities moving in the two-dimensional space with obstacles and the corresponding Reed graph.

Notation and Problem Definition. We are given a set $\mathcal{X}$ of n entities, each moving along a piecewise linear trajectory with τ vertices, and a set of pairwise disjoint polygonal obstacles $\mathcal{O} = \{\mathcal{O}_1, .., \mathcal{O}_h\}$. Let m denote the total complexity of $\mathcal{O}$.

To determine if a set of entities may form a group, we have to decide if they are close together. We model this by a parameter ε. Two entities a and b are *directly connected* at time t if they are within geodesic distance ε from each other, that is, $\varsigma_{ab}(t) \leq \varepsilon$. A set of entities $\mathcal{X}'$ is *ε-connected* at time t if for any pair

*FS is supported by the Danish National Research Foundation under grant nr. DNRF84.

†Dept. of Information and Computing Sciences, Utrecht University, m.loffler@uu.nl

‡MADALGO, Aarhus University, f.staals@cs.au.dk

§Dept. of Informatics, KIT, jerome.urhausen@gmail.com

Simple polygon	$O(\tau n^2(\log^2 m + \log n) + m)$
Well-spaced obstacles	$O(\tau n^2 m \log n)$
General obstacles	$O(\tau n^2 m^2 \log n + m^2 \log m)$
2ε-big obstacles	$O(\tau(n^2(\log n + \log^2 m) + nm \log m))$
ε-big obstacles	$O(\tau n^2(\text{polylog}\, m + \log n) + m^2 \log m)$
General obstacles	$O(\tau(n^2 m + \lambda_4(n)m^3)(\log n + \log m))$

Table 1: The running time for computing the Reeb graph for various obstacle configurations. The top ones are from [5], the bottom ones are new. The $\lambda_s(n)$ term denotes the maximum length of a Davenport-Schinzel sequence of order s consisting of n symbols.

$a, b \in \mathcal{X}'$ there is a sequence $a = a_0, a_1, .., a_k = b$ such that a_i and a_{i+1} are directly connected. We refer to a time at which a and b become directly connected or disconnected as an *ε-event*. At such a time the distance between a and b is exactly ε. If an ε-event also connects or disconnects the maximal ε-connected set(s) containing a and b, it is a *critical event*. A (maximal) ε-connected set of entities $\mathcal{X}'$ is a *group* if it is ε-connected at any time t in a time interval of length at least δ, and it has at least a certain size.

The algorithm of Buchin et al. [2] proceeds in two phases. In the first phase, it computes the *Reeb-graph* $\mathcal{R}$ capturing the connectivity between the entities. In the second phase, it computes all maximal groups using only information in the Reeb-graph. So, once we compute $\mathcal{R}$, we can use the algorithm from Buchin et al. [2] to compute all maximal groups. An edge (u, v) in $\mathcal{R}$ corresponds to a maximal set of entities that is ε-connected during time interval $[t_u, t_v]$. The Reeb-graph has a vertex v at time t_v if (and only if) two maximal sets of ε-connected entities merge or split. A vertex corresponds uniquely to a critical event. See Fig. 1.

Results and Organisation. We start in Section 2 with the new algorithm for the case that the entities move amidst general obstacles. In Section 3 we formalize what it means for an obstacle to be ε-big, and show that if the obstacles are ε-big, the Reeb graph has low complexity. Furthermore, we show that we can compute the Reeb-graph efficiently in such a setting. Omitted proofs and details can be found in [8].

2 An Algorithm for General Obstacles

In this section we present an $O(\tau(n^2 m + \lambda_4(n)m^3)(\log n + \log m))$ time algorithm to compute the Reeb-graph when the entities move amidst arbitrary polygonal obstacles. This improves the algorithm of Kostitsyna et al. [5] if the total complexity m of the obstacles is $\omega(n^2/\lambda_4(n))$.

Most existing algorithms to compute the Reeb graph $\mathcal{R}$ first determine all ε-events, and use them to maintain the *entity-graph* while varying the time t. The entity-graph $G(t)$ at time t is the graph whose vertices are the entities, and whose edges connect two entities if and only if they are directly connected at time t. Clearly, $G(t)$ changes only at ε-events. The Reeb graph corresponds exactly to the changes in connected components in the entity-graph. That is, there is a critical event at time t if and only if the connected components in the entity graph change at time t. However, the number of critical events, and thus the size of $\mathcal{R}$, is much smaller than the number of ε-events [5]. Hence, we wish to reduce the number of ε-events that we have to consider.

The ER-graph. Our new algorithm will still use the idea of the entity-graph, but we add new vertices, corresponding to regions, that allow us to handle multiple ε-events at once. For our new graph, the *entity-region graph* (ER-graph), we still require that two entities are in the same connected component if and only if they are ε-connected.

The regions corresponding to the new vertices are built around the obstacles such that multiple ε-events involving entities in these regions only induce few critical events. To achieve that, we subdivide for each obstacle vertex v the area within geodesic ε-distance of v using the shortest path map originating at v [4].
Claim. We can further subdivide the shortest path maps into $O(m^2)$ regions such that

- each region has constant complexity,
- each region has (geodesic) diameter at most ε, and
- each entity enters and exits a region at most once per time step.

In the ER-graph we then only directly connect entities by an edge if they are closer than ε and can see each other, i.e. if the shortest path between them does not use an obstacle vertex. All other connections use region vertices. We connect a region to all the entities it contains and we connect two regions belonging to the same obstacle vertex v if and only if they contain entities for which the shortest path between them passing through v has length at most ε. See Fig. 2 for an example. The use of regions drastically reduces the number of events to handle.

There are $O(\tau n^2 m)$ events at which an edge between two entities appears or disappears in the ER-graph. For edges between an entity and a region there are $O(\tau n m^2)$ such events and for edges between two regions there are $O(\tau \lambda_4(n) m^3)$ such events.

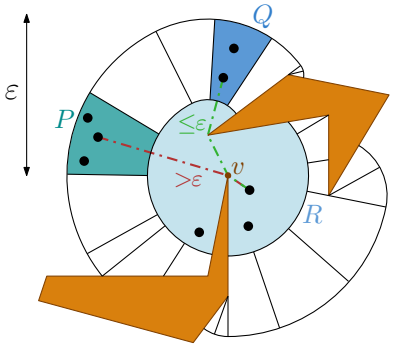


Figure 2: Entities inside regions corresponding to an obstacle vertex v at a fixed time t. Two shortest paths passing through v between entities in different regions are indicated in red and green. In the corresponding ER-graph the regions Q and R are connected by an edge, but P and R are not.

Algorithm The algorithm runs as follows. We first build the regions for each obstacle vertex. This gives us all the vertices of the ER-graph. Then we determine the events at which the edges of the ER-graph are added or removed and sort them. Using these events we keep an updated version of the ER-graph which allows us to build the Reeb graph. The regions can be built in $O(m^2 \log m)$ time. The determination and sorting of the events takes $O(\tau(n^2m + \lambda_4(n)m^3)\log(nm))$ time. In order to keep the connected components of the ER-graph updated we need $O(\tau(n^2m + \lambda_4(n)m^3)\log(n + m^2))$ time with the approach proposed by Parsa [7], because the ER-graph has at most $O(n + m^2)$ vertices.

Theorem 1 *Let $\mathcal{X}$ be a set of n entities, each moving amidst a set of obstacles $\mathcal{O}$ along a piecewise linear trajectory with τ vertices. The Reeb graph can be computed in $O(\tau(n^2m + \lambda_4(n)m^3)(\log n + \log m))$ time.*

3 Big Obstacles

In this section we investigate a new class of obstacles for which we can compute the Reeb graph efficiently. Namely, ε-big obstacles. An *ε-big obstacle* is an obstacle that does not fit into a strip of width ε of any orientation. We now show that if all obstacles are big there are only few ε-events, and thus the Reeb-graph is small, and we can compute it efficiently.

Two obstacle avoiding paths P_1 and P_2 have the same *homotopy type*, if and only if we can continuously deform P_1 into P_2 while remaining obstacle avoiding.

Lemma 2 *Let a and b be two entities moving amidst a set of ε-big obstacles. Let I be a time interval in which both a and b move linearly, and $t_1, t_2 \in I$ be two times at which their geodesic distance is at most ε. The geodesics $P_1 = \varsigma_{ab}(t_1)$ and $P_2 = \varsigma_{ab}(t_2)$ have the same homotopy type.*

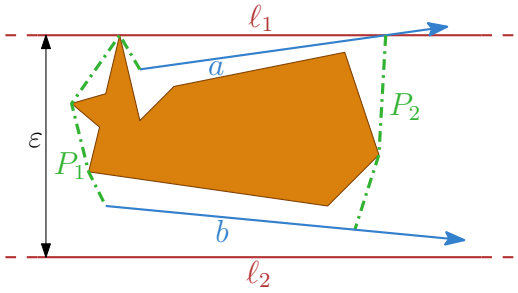


Figure 3: The construction showing that the obstacle can not be ε-big.

Proof. Assume, by contradiction, that P_1 and P_2 have different homotopy types. It follows that the region R bounded by $\overline{a(t_1)a(t_2)}$, P_2, $\overline{b(t_2)b(t_1)}$, and P_1 contains at least one obstacle. Let ℓ_1 be an outer tangent to P_1 and P_2 (See Fig. 3), and assume without loss of generality that ℓ_1 is horizontal, and that P_1 and P_2 lie below ℓ_1. Let ℓ_2 be the horizontal line at distance ε below ℓ_1.

We know that for $i = 1, 2$, every two points on P_i are at Euclidean distance at most ε, because the length of P_i is at most ε. Since both P_1 and P_2 have a common point with ℓ_1, all points on P_1 and P_2 lie on or above ℓ_2. It follows that $\overline{a(t_1)a(t_2)}$ and $\overline{b(t_2)b(t_1)}$ also lie on or above ℓ_2. This means that the strip between ℓ_1 and ℓ_2 contains the region R, and thus at least one ε-big obstacle. Contradiction. It follows that P_1 and P_2 have the same homotopy type. □

Theorem 3 *Let $\mathcal{X}$ be a set of n entities, each moving amidst a set of ε-big obstacles $\mathcal{O}$ along a piecewise linear trajectory with τ vertices. The number of ε-events, and thus the size of the Reeb-graph, is at most $O(\tau n^2)$.*

Proof. There are $O(n^2)$ pairs of entities and for each pair a, b there are $O(\tau)$ time intervals in which both of them move along a line with constant speed. Consider such an interval I, and let $t_1, t_2 \in I$ be two ε-events involving a and b. By Lemma 2, the geodesics $\varsigma_{ab}(t_1)$ and $\varsigma_{ab}(t_2)$ have the same homotopy type. Kostitsyna et al. [5] effectively show that if the homotopy type of the path between a and b is fixed, the length of such a path is a convex function in t, and thus, there are at most two ε-events involving a and b in interval I. The theorem then follows. □

3.1 Computing ε-events among 2ε-big obstacles

When all obstacles are 2ε-big we can efficiently compute the ε-events as follows. For each entity a and each interval I when a moves along an edge s, consider the geodesic *ε-surrounding* S of s, that is, all points

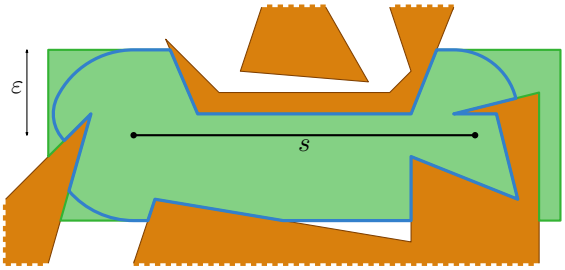


Figure 4: A simple polygon (green) containing the geodesic ε-surrounding (blue) of a trajectory edge s.

whose geodesic distance to s is at most ε. Clearly, all ε-events involving a in interval I are contained in S. Furthermore, since all obstacles are 2ε-big, S is simple; i.e. it contains no holes. See Fig. 4. We now construct a simple polygon P containing S, and compute ε-events involving a using the algorithm for entities moving in a simple polygon by Kostitsyna et al. [5].

We can construct P in $O(m \log m)$ time as follows. We first approximate the Euclidean ε-surrounding of s with the smallest rectangle possible. Then we calculate the intersections between the rectangle and the obstacle edges and sort them clockwise. Starting from the obstacle vertex that is the closest to s we can then walk along the boundary of the simple polygon P using these intersection points.

Since we have $O(\tau n)$ trajectory edges, building all these polygons takes $O(\tau n m \log m)$ time. Then, for each pair of entities and each time interval in which they travel at constant speed, we take the polygon of one of the entities and determine the interval during which the other entity is inside this polygon. Therefore computing each of the $O(\tau n^2)$ ε-events can then be made using parametric search in $O(\log^2 m)$ time per event [6]. Once we have determined and sorted all ε-events we can build the Reeb graph using $O(\log n)$ time per event. We conclude:

Theorem 4 *Let $\mathcal{X}$ be a set of n entities, each moving amidst a set of 2ε-big obstacles $\mathcal{O}$ along a piecewise linear trajectory with τ vertices. The Reeb graph can be computed in $O(\tau(n^2(\log n + \log^2 m) + nm \log m))$ time, where m is the total complexity of $\mathcal{O}$.*

3.2 Computing ε-events among ε-big obstacles

The main difference to the previous case is that an ε-big obstacle can be completely contained inside the Euclidean ε-surrounding of the segment. This means that the previously taken approach of building a polygon by approximating the Euclidean ε-surrounding yields a polygon with holes. Thus for a pair of entities we do not know the homotopy type of the shortest path yet. Therefore another approach is taken here.

The global idea of our approach is as follows. We compute all Euclidean ε-events, ignoring the obstacles. This gives us $O(\tau n^2)$ time intervals during which two entities, say a and b, are within Euclidean distance ε and move linearly. Any geodesic ε-event occurs within such intervals $[t_f, t_g]$. Furthermore, by Lemma 2 there are at most two such events per interval. Then the following claim holds.

Claim. For any time $t \in [t_f, t_g]$, there are only $O(1)$ choices for the first and last vertex on the geodesic between $a(t)$ and $b(t)$.

The algorithm tries all such pairs using the shortest path maps [4] to find the true geodesic at time t. Hence, for a given time t, we can test if the shortest path between $a(t)$ and $b(t)$ has length at most ε and if the derivative is positive or negative in $O(\text{polylog}\, m)$ time. This means that we can use parametric search to find the times at which the geodesic distance is exactly ε [5]. Overall, we conclude:

Theorem 5 *Let $\mathcal{X}$ be a set of n entities, each moving amidst a set of ε-big obstacles $\mathcal{O}$ along a piecewise linear trajectory with τ vertices. The Reeb graph can be computed in $O(\tau n^2(\text{polylog}\, m + \log n) + m^2 \log m)$ time, using $O(m^2)$ space, where m is the total complexity of $\mathcal{O}$.*

References

[1] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, 2008.

[2] K. Buchin, M. Buchin, M. van Kreveld, B. Speckmann, and F. Staals. Trajectory grouping structure. *Journal of Computational Geometry*, 6(1):75–98, 2015.

[3] J. Gudmundsson, M. van Kreveld, and B. Speckmann. Efficient detection of patterns in 2d trajectories of moving points. *Geoinformatica*, 11(2):195–215, 2007.

[4] J. Hershberger and S. Suri. An Optimal Algorithm for Euclidean Shortest Paths in the Plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

[5] I. Kostitsyna, M. van Kreveld, M. Löffler, B. Speckmann, and F. Staals. Trajectory grouping structure under geodesic distance. In *31st Int. Symp. on Comp. Geom.*, volume 34 of *LIPICS*, pages 674–688, 2015.

[6] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4(4):414–424, 1979.

[7] S. Parsa. A deterministic O(m log m) time algorithm for the reeb graph. *Discrete & Computational Geometry*, 49(4):864–878, 2013.

[8] J. Urhausen. New results on trajectory grouping under geodesic distance. `https://fstaals.net/research/bachelor_thesis_jerome.pdf`, 2015.