Trajectory Grouping Structure*

Kevin Buchin[1], Maike Buchin[1], Marc van Kreveld[2],
Bettina Speckmann[1], and Frank Staals[2]

[1] Dep. of Mathematics and Computer Science, TU Eindhoven
[2] Dep. of Information and Computing Sciences, Utrecht University

Abstract. The collective motion of a set of moving entities like people, birds, or other animals, is characterized by groups arising, merging, splitting, and ending. Given the trajectories of these entities, we define and model a structure that captures all of such changes using the Reeb graph, a concept from topology. The *trajectory grouping structure* has three natural parameters, namely group size, group duration, and entity inter-distance. These parameters allow us to obtain detailed or global views of the data. We prove complexity bounds on the maximum number of maximal groups that can be present, and give algorithms to compute the grouping structure efficiently. Furthermore, we showcase the results of experiments using data generated by the NetLogo flocking model and from the Starkey project. Although there is no ground truth for the groups in this data, the experiments show that the trajectory grouping structure is plausible and has the desired effects when changing the essential parameters. Our research provides the first complete study of trajectory group evolvement, including combinatorial, algorithmic, and experimental results.

1 Introduction

In recent years there has been an increase in location-aware devices and wireless communication networks. This has led to a large amount of trajectory data capturing the movement of animals, vehicles, and people. The increase in trajectory data goes hand in hand with an increasing demand for techniques and tools to analyze them, for example, in sports, ecology, transport, and social services.

An important task is the analysis of movement patterns. In particular, given a set of moving entities we wish to determine when and which subsets of entities travel together. When a sufficiently large set of entities travels together for a sufficiently long time, we call such a set a *group* (we give a more formal definition later). Groups may start, end, split and merge with other groups. Apart from the question what the current groups are, we also want to know which splits and merges led to the current groups, when they happened, and which groups they involved. We wish to capture this group change information in a model that we call the *trajectory grouping structure*.

* MB, BS & FS are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.106, 639.022.707 & 612.001.022, respectively.

The informal definition above suggests that three parameters are needed to define groups: (i) a spatial parameter for the distance between entities; (ii) a temporal parameter for the duration of a group; (iii) a count for the number of entities in a group. We will design our grouping structure definition to incorporate these parameters so that we can study grouping at different scales. We use the three parameters as follows: a small spatial parameter implies we are interested only in spatially close groups, a large temporal parameter implies we are interested only in long-lasting groups, and a large count implies we are interested only in large groups. By adjusting the parameters suitably, we can obtain more detailed or more generalized views of the trajectory grouping structure.

The use of scale parameters and the fact that the grouping structure changes at discrete events suggest the use of computational topology [6]. In particular, we use Reeb graphs to capture the grouping structure. Reeb graphs have been used extensively in shape analysis and the visualization of scientific data (see e.g. [2, 5, 8]). A Reeb graph captures the structure of a two- or higher-dimensional scalar function, by considering the evolution of the connected components of the level sets. The computation of Reeb graphs has received considerable attention in computational geometry and topology; an overview is given in [4]. Recently, a deterministic $O(n \log n)$ time algorithm was presented for constructing the Reeb graph of a 2-skeleton of size n [16]. Edelsbrunner et al. [5] discuss time-varying Reeb graphs for continuous space-time data. Although we also analyze continuous space-time data (2D-space in our case), our Reeb graphs are not time-varying, but time is the parameter that defines the Reeb graph.

Our research is motivated by and related to previous research on flocks [1, 9, 10, 19], herds [11], convoys [12], moving clusters [13], mobile groups [20] and swarms [14]. These concepts differ from each other in the way in which space and time are used to test if entities form a group: do the entities stay in a single disc or are they density-connected [7], should they stay together during consecutive time steps or not, can the group members change over time, etc. Only the herds concept [11] includes the splitting and merging of groups.

Contributions. We present the first complete study of trajectory group evolvement, including combinatorial, algorithmic, and experimental results. Our research differs from and improves on previous research in the following ways. Firstly, our model is simpler than herds and thus more intuitive. Secondly, we consider the grouping structure at continuous times instead of at discrete steps (which was done only for flocks). Thirdly, we analyze the algorithmic and combinatorial aspects of groups and their changes. Fourthly, we implemented our algorithms and provide evidence that our model captures the grouping structure well and can be computed efficiently. We created videos based on our implementation showing the maximal groups we found in simulated NetLogo flocking data [21] and in real-world data from the Starkey project [15], see www.staff.science.uu.nl/∼staal006/grouping.

A definition for a group. Let $\mathcal{X}$ be a set of entities of which we have locations over time. The *ε-disc* of an entity x (at time t) is a disc of radius ε centered at x at time t. Two entities are *directly connected* at time t if their ε-discs overlap. Two entities x and y are *ε-connected* at time t if there is a sequence $x = x_0, .., x_k = y$ of entities such that for all i, x_i and x_{i+1} are directly connected.

A subset $S \subseteq \mathcal{X}$ of entities is ε-connected at time t if all entities in S are pairwise ε-connected at time t. This means that the union of the ε-discs of entities in S forms a single connected region. The set S forms a *component* at time t if and only if S is ε-connected, and S is maximal with respect to this property. The set of components $\mathcal{C}(t)$ at time t forms a partition of the entities in $\mathcal{X}$ at time t.

Let the spatial parameter of a group be ε, the temporal parameter δ, and the size parameter m. A set G of k entities forms a *group* during time interval I if and only if the following three conditions hold: (i) G contains at least m entities, so $k \geq m$, (ii) the interval I has length at least δ, and (iii) at all times $t \in I$, there is a component $C \in \mathcal{C}(t)$ such that $G \subseteq C$.

We denote the interval $I = [t_s, t_e]$ of group G with I_G. Group H *covers* group G if $G \subseteq H$ and $I_G \subseteq I_H$. If there are no groups that cover G, we say G is *maximal* (on I_G). In Fig. 1, groups $\{x_1, x_2\}$, $\tilde{G} = \{x_3, x_4\}$, $\hat{G} = \{x_5, x_6\}$, and $G = \{x_1, .., x_4\}$ are maximal: $\tilde{G}$ and $\hat{G}$ on $[t_0, t_5]$, G on $[t_1, t_2]$. Group $\{x_1, x_3\}$ is covered by G and hence not maximal.

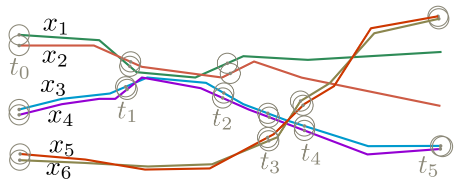

Fig. 1. For $m = 2$ and $\delta > t_4 - t_3$ there are four maximal groups: $\{x_1, x_2\}$, $\{x_3, x_4\}$, $\{x_5, x_6\}$, and $\{x_1, .., x_4\}$.

Note that entities can be in multiple maximal groups at the same time. For example, entities $\{y_1, y_2, y_3\}$ can travel together for a while, then y_4, y_5 may become ε-connected, and shortly thereafter y_1, y_4, y_5 separate and travel together for a while. Then y_1 may be in two otherwise disjoint maximal groups for a short time. An entity can also be in two maximal groups where one is a subset of the other. In that case the group with fewer entities must last longer. That an entity is in more groups simultaneously may seem counterintuitive at first, but it is necessary to capture all grouping information. We will show that the total number of maximal groups is $O(\tau n^3)$, where n is the number of entities in $\mathcal{X}$ and τ is the number of edges of each input trajectory. This bound is tight in the worst case.

Our maximal group definition uses three parameters, which all allow a more global view of the grouping structure. In particular, we observe that there is *monotonicity* in the group size and the duration: If G is a group during interval I, and we decrease the minimum required group size m or decrease the minimum required duration δ, then G is still a group on time interval I. Also, if G is a maximal group on I, then it is also a maximal group for a smaller m or smaller δ. For the spatial parameter ε we observe monotonicity in a slightly different manner: if G is a group for a given ε, then for a larger value of ε there exists a group $G' \supseteq G$. The monotonicity property is important when we want to have a more detailed view of the data: we do not lose maximal groups in a more detailed view. The group may however be extended in size and/or duration.

We capture the grouping structure using a Reeb graph of the ε-connected components together with the set of all maximal groups. Parts of the Reeb graph that do not support a maximal group can be omitted. The grouping structure can help us in answering various questions. For example:

- What is the largest/longest maximal group at time t?
- How many entities are currently (not) in any maximal group?
- What is the first maximal group that starts/ends after time t?
- What is the total time that an entity was part of any maximal group?
- Which entity has shared maximal groups with the most other entities?

Furthermore, the grouping structure can be used to partition the trajectories in independent data sets, to visualize grouping aspects of the trajectories, and to compare grouping across different data sets.

Results and Organization. We discuss how to represent the grouping structure in Section 2, and prove that there are always $O(\tau n^3)$ maximal groups, which is tight in the worst case. Here n is the number of trajectories (entities) and τ the number of edges in each trajectory. We present an algorithm to compute the trajectory grouping structure and all maximal groups in Section 3. This algorithm runs in $O(\tau n^3 + N)$ time, where N is the total output size. In Section 4 we discuss robustness briefly; all details can be found in the full version of the paper [3]. In Section 5 we evaluate our methods on synthetic and real-world data.

2 Representing the Grouping Structure

Let $\mathcal{X}$ be a set of n entities, where each entity travels along a path of τ edges. To compute the grouping structure we consider a manifold $\mathcal{M}$ in $\mathbb{R}^3$, where the z-axis corresponds to time. The manifold $\mathcal{M}$ is the union of n "tubes". Each tube consists of τ skewed cylinders with horizontal radius ε that we obtain by tracing the ε-disc of an entity x over its trajectory.

Let H_t denote the horizontal plane at height t, then the set $\mathcal{M} \cap H_t$ is the *level set* of t. The connected components in the level set of t correspond to the components (maximal sets of ε-connected entities) at time t. We will assume that all trajectories have their known positions at the same times $t_0, .., t_\tau$ and that no three entities become ε-(dis)connected at the same time. Our theory does not depend on these assumptions and we could remove them, but they make the descriptions considerably more clear.

2.1 The Reeb graph

We start out with a possibly disconnected solid that is the union of a collection of tube-like regions: a 3-manifold with boundary. Note that this manifold is not explicitly defined. We are interested in horizontal cross-sections, and the evolution of the connected components of these cross-sections defines the Reeb graph. Note that this is different from the usual Reeb graph that is obtained from the 2-manifold that is the boundary of our 3-manifold, using the level sets of the height function (the function whose level sets we follow is the height function above a horizontal plane below the manifold), see [6] for more on this topic.

To describe how the components change over time, we consider the Reeb graph $\mathcal{R}$ of $\mathcal{M}$. The Reeb graph has a vertex v at every time t_v where the components change. The vertex times are usually not at any of the given times $t_0, .., t_\tau$, but in between two consecutive time steps. The vertices of the Reeb graph can be classified in four groups. There is a *start vertex* for every component at t_0 and an *end vertex* at t_τ. A start vertex has in-degree zero and out-degree one, and an end vertex has in-degree one and out-degree zero. The remaining vertices are either *merge vertices* or *split vertices*. Since we assume that no three entities become ε-(dis)connected at exactly the same time there are no simultaneous splits and merges. This means merge vertices have in-degree two and out-degree one, and split vertices have in-degree one and out-degree two. A directed edge $e = (u, v)$ connecting vertices u and v, with $t_u < t_v$, corresponds to a set C_e of entities that form a component at any time $t \in I_e = [t_u, t_v]$. The Reeb graph is this directed graph. Note that the Reeb graph depends on the spatial parameter ε, but not on the other two parameters of maximal groups.

Theorem 1. *Given a set $\mathcal{X}$ of n entities, in which each entity travels along a trajectory of τ edges, the Reeb graph $\mathcal{R} = (V, E)$ has $O(\tau n^2)$ vertices and edges. These bounds are tight in the worst case.*

Proof. Lemma 1 in the full paper [3] gives a simple construction that shows that the Reeb graph may have $\Omega(\tau n^2)$ vertices and edges in the worst case. For the upper bound, consider a trajectory edge (v_i, v_{i+1}) of entity $x \in \mathcal{X}$. An other entity $y \in \mathcal{X}$ is directly connected to x during at most one interval $I \subseteq [t_i, t_{i+1}]$. This interval yields at most two vertices in $\mathcal{R}$. The trajectory of x consists of τ edges, hence a pair x, y produces $O(\tau)$ vertices in $\mathcal{R}$. This gives a total of $O(\tau n^2)$ vertices, each with constant degree, so there are $O(\tau n^2)$ edges. □

The trajectories of entities are associated with the edges of the Reeb graph in a natural way. Each entity follows a directed path in the Reeb graph from a start vertex to an end vertex. Similarly, (maximal) groups follow a directed path from a start or merge vertex to a split or end vertex. If $m > 0$ or $\delta > 0$, there may be edges in the Reeb graph with which no group is associated. These edges do not contribute to the grouping structure, so we can discard them. The remainder of the Reeb graph we call the *reduced Reeb graph*, which, together with all maximal groups associated with its edges, forms the *trajectory grouping structure*.

2.2 Bounding the Number of Maximal Groups

To bound the total number of maximal groups, we study the case where $m = 1$ and $\delta = 0$, because larger values can only reduce the number of maximal groups. It may seem as if each vertex in the Reeb graph simply creates as many maximal groups as it has outgoing edges. However, consider for example Fig. 2. Split vertex v creates not only the maximal groups $\{1, 3, 5, 7\}$ and $\{2, 4, 6, 8\}$, but also $\{1, 3\}$, $\{5, 7\}$, $\{2, 4\}$, and $\{6, 8\}$. These last four groups are all maximal on $[t_2, t]$, for $t > t_4$.

Notice that all six newly discovered groups start strictly before t_v, but only at t_v do we realize that these groups are maximal, which is the meaning that should be understood with "creating maximal groups". This example can be extended to arbitrary size. Hence a vertex v may create many new maximal groups, some of which start before t_v. We can show that each vertex creates at most n new maximal groups, which leads to a total of $O(\tau n^3)$ maximal groups. The proof of the following theorem is given in the full paper [3].

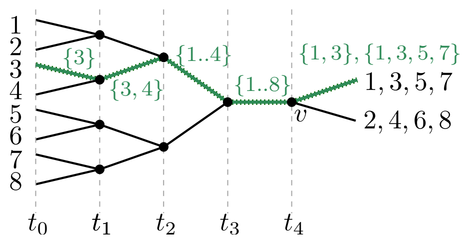

Fig. 2. The maximal groups containing entity 3 (green). Vertex v creates six new groups, including $\{1, 3\}$ and $\{1, 3, 5, 7\}$.

Theorem 2. *Let $\mathcal{X}$ be a set of n entities, in which each entity travels along a trajectory of τ edges. There are at most $O(\tau n^3)$ maximal groups, and this is tight in the worst case.*

3 Computing the Grouping Structure

To compute the grouping structure we need to compute the reduced Reeb graph and the maximal groups. We now show how to do this efficiently. Removing the edges of the Reeb graph that are not used is an easy post-processing step which we do not discuss further.

3.1 Computing the Reeb graph

We can compute the Reeb graph $\mathcal{R} = (V, E)$ as follows. We first compute all times where two entities x and y are at distance 2ε from each other. We distinguish two types of events, *connect events* at which x and y become directly connected, and *disconnect events* at which x and y stop being directly connected.

We now process the events on increasing time while maintaining the current components. We do this by maintaining a graph $G = (\mathcal{X}, Z)$ representing the directly-connected relation, and the connected components in this graph. The set of vertices in G is the set of entities. The graph G changes over time: at connect events we insert new edges into G, and at disconnect events we remove edges. At any given time t, G contains an edge (x, y) if and only if x and y are directly connected at time t. Hence the components at t (the maximal sets of ε-connected entities) correspond to the connected components in G at time t. Since we know all times at which G changes in advance, we can use the same approach as in [16] to maintain the connected components: we assign a weight to each edge in G and we represent the connected components using a maximum weight spanning forest. The weight of edge (x, y) is equal to the time at which we remove it from G, that is, the time at which x and y become directly disconnected. We store the maximum weight spanning forest F as an ST-tree [17], which allows connectivity queries, inserts, and deletes, in $O(\log n)$ time.

We spend $O(n^2)$ time to initialize the graph G at t_0 in a brute-force manner. For each component we create a start vertex in $\mathcal{R}$. We also initialize a one-to-one mapping M from the current components in G to the corresponding vertices in $\mathcal{R}$. When we handle a connect event of entities x and y at time t, we query F to get the components C_x and C_y containing x and y, respectively. Using M we locate the corresponding vertices v_x and v_y in $\mathcal{R}$. If $C_x \neq C_y$ we create a new merge vertex v in $\mathcal{R}$ with time $t_v = t$, add edges (v_x, v) and (v_y, v) to $\mathcal{R}$ labeled C_x and C_y, respectively. If $C_x = C_y$ we do not change $\mathcal{R}$. Finally, we add the edge (x, y) to G (which may cause an update to F), and update M.

At a disconnect event we first query F to find the component C currently containing x and y. Using M we locate the vertex u corresponding to C. Next, we delete the edge (x, y) from G, and again query F. Let C_x and C_y denote the components containing x and y, respectively. If $C_x = C_y$ we are done, meaning x and y are still ε-connected. Otherwise we add a new split vertex v to $\mathcal{R}$ with time $t_v = t$, and an edge $e = (u, v)$ with $C_e = C$ as its component. We update M accordingly.

Finally, we add an end vertex v for each component C in F with $t_v = t_\tau$. We connect the vertex $u = M(C)$ to v by an edge $e = (u, v)$ and let $C_e = C$ be its component.

Analysis. We need $O(\tau n^2 \log n)$ time to compute all $O(\tau n^2)$ events and sort them according to increasing time. To handle an event we query F a constant number of times, and we insert or delete an edge in F. These operations all take $O(\log n)$ time. So the total time required for building $\mathcal{R}$ is $O(\tau n^2 \log n)$.

Theorem 3. *Given a set $\mathcal{X}$ of n entities, in which each entity travels along a trajectory of τ edges, the Reeb graph $\mathcal{R} = (V, E)$ has $O(\tau n^2)$ vertices and edges, and can be computed in $O(\tau n^2 \log n)$ time.*

3.2 Computing the maximal groups

We now show how to compute all maximal groups using the Reeb graph $\mathcal{R} = (V, E)$. We will ignore the requirements that each maximal group should contain at least m entities and have a minimal duration of δ. That is, we assume $m = 1$ and $\delta = 0$. It is easy to adapt the algorithm for larger values.

Labeling the edges. Our algorithm labels each edge $e = (u, v)$ in the Reeb graph with a set of maximal groups $\mathcal{G}_e$. The groups $G \in \mathcal{G}_e$ are those groups for which we have discovered that G is a maximal group at a time $t \leq t_u$. Each maximal group G becomes maximal at a vertex, either because a merge vertex created G as a new group that is maximal, or because G is now a maximal set of entities that is still together after a split vertex. This means we can compute all maximal groups as follows.

We traverse the set of vertices of $\mathcal{R}$ in topological order. For every vertex v we compute the maximal groups on its outgoing edge(s) using the information on its incoming edge(s).

If v is a start vertex it has one outgoing edge $e = (v, u)$. We set $\mathcal{G}_e$ to $\{(C_e, t_v)\}$ where $t_v = t_0$. If v is a merge vertex it has two incoming edges, e_1 and e_2. We propagate the maximal groups from e_1 and e_2 on to the outgoing edge e, and we discover (C_e, t_v) as a new maximal group. Hence $\mathcal{G}_e = \mathcal{G}_{e_1} \cup \mathcal{G}_{e_2} \cup \{(C_e, t_v)\}$.

If v is a split vertex it has one incoming edge e, and two outgoing edges e_1 and e_2. A maximal group G on e may end at v, continue on e_1 or e_2, or spawn a new maximal group $G' \subset G$ on either e_1 or e_2. In particular, for any group G' in $\mathcal{G}_{e_i}$, there is a group G in $\mathcal{G}_e$ such that $G' = G \cap C_i \neq \emptyset$. The starting time of G' is $t' = \min\{t \mid (G, t) \in \mathcal{G}_e \wedge G' \subseteq G\}$. Thus, t' is the first time G' was part of a maximal group on e. Stated differently, t' is the first time G' was in a component on a path to v. Fig. 3 illustrates this case. If v is an end vertex it has no outgoing edges. So there is nothing to be done.

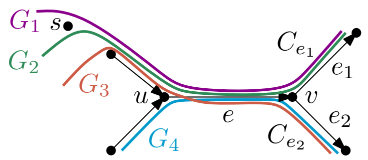

Fig. 3. After split vertex v, $\mathcal{G}_{e_1}$ contains the groups $C_{e_1} = G_1 \cup G_2$ (with starting time t_s), G_1, and G_2. Maximal groups $C_{e_2} = G_3 \cup G_4$ (with starting time t_u), G_3, and G_4 go to e_2. The maximal groups C_e and $G_1 \cup G_2 \cup G_3$ end at v.

Storing the maximal groups. We need a way to store the maximal groups $\mathcal{G}_e$ on an edge $e = (u, v)$ in such a way that we can efficiently compute the set(s) of maximal groups on the outgoing edge(s) of a vertex v. We now show that we can use a tree $\mathcal{T}_e$ to represent $\mathcal{G}_e$, with which we can handle a merge vertex in $O(1)$ time, and a split vertex in $O(k)$ time, where k is the number of entities involved. The tree uses $O(k)$ storage.

We say a group G is a *subgroup* of a group H if and only if $G \subseteq H$ and $I_H \subseteq I_G$. For example, in Fig. 1 $\{x_1, x_2\}$ is a subgroup of $\{x_1, .., x_4\}$. Note that both G and H could be maximal. The proof of the following lemma is given in the full paper [3].

Lemma 1. *Let e be an edge of $\mathcal{R}$, and let S and T be maximal groups in $\mathcal{G}_e$ with starting times t_S and t_T, respectively. There is also a maximal group $G \supseteq S \cup T$ on e with starting time $t_G \geq \max(t_S, t_T)$, and if $S \cap T \neq \emptyset$ then S is a subgroup of T or vice versa.*

We represent the groups $\mathcal{G}_e$ on an edge $e \in E$ by a tree $\mathcal{T}_e$. We call this the *grouping tree*. Each node $v \in \mathcal{T}_e$ represents a group $G_v \in \mathcal{G}_e$. The children of a node v are the largest subgroups of G_v. From Lemma 1 it follows that any two children of v are disjoint. Hence an entity $x \in G_v$ occurs in only one child of v. Furthermore, note that the starting times are monotonically decreasing on the path from the root to a leaf: smaller groups started earlier. A leaf corresponds to a smallest maximal group on e: a singleton set with an entity $x \in C_e$. It follows that $\mathcal{T}_e$ has $O(n)$ leaves, and therefore has size $O(n)$. Note, however, that the summed sizes of all maximal groups can be quadratic.

Analysis. We analyze the time required to label each edge e with a tree $\mathcal{T}_e$ for a given Reeb graph $\mathcal{R} = (V, E)$. Topologically sorting the vertices takes linear time. So the running time is determined by the processing time in each vertex, that is, computing the tree(s) $\mathcal{T}_e$ on the outgoing edge(s) e of each vertex. Start, end, and merge vertices can be handled in $O(1)$ time: start and end vertices are trivial, and at a merge vertex v the tree $\mathcal{T}_e$ is simply a new root node with time t_v and as children the (roots of the) trees of the incoming edges. At a split vertex we have to split the tree $\mathcal{T} = \mathcal{T}_{(u,v)}$ of the incoming edge (u, v) into two trees for the outgoing edges of v. For this, we traverse $\mathcal{T}$ in a bottom-up fashion, and for each node, check whether it induces a vertex in one or both of the trees after splitting. This algorithm runs in $O(|\mathcal{T}|)$ time. Since $|\mathcal{T}| = O(n)$ the total running time of our algorithm is $O(n|V|) = O(\tau n^3)$.

Reporting the groups. We can augment our algorithm to report all maximal groups at split and end vertices. The main observation is that a maximal group ending at a split vertex v, corresponds exactly to a node in the tree $\mathcal{T}_{(u,v)}$ (before the split) that has entities in leaves below it that separate at v. The procedures for handling split and end vertices can easily be extended to report the maximal groups of size at least m and duration at least δ by simply checking this for each maximal group. Although the number of maximal groups is $O(\tau n^3)$ (Theorem 2), the summed size of all maximal groups can be $\Theta(\tau n^4)$. The running time of our algorithm is $O(\tau n^3 + N)$, where N is the total output size.

Theorem 4. *Given a set $\mathcal{X}$ of n entities, in which each entity travels along a trajectory of τ edges, we can compute all maximal groups in $O(\tau n^3 + N)$ time, where N is the output size.*

4 Robustness

The grouping structure definition we have given and analyzed has a number of good properties. It fulfills monotonicity, and in the previous sections we showed that there are only polynomially many maximal groups, which can be computed in polynomial time as well. In this section we study the property of robustness, which our definition of grouping structure does not have yet. Intuitively, a robust grouping structure ignores short interruptions of groups, as these interruptions may be insignificant at the temporal scale at which we are studying the data. For example, if we are interested in groups that have a duration of one hour or more, we may want to consider interruptions of a minute or less insignificant.

We introduce a new temporal parameter α. Interruptions of duration at most α may be ignored, and the precise moment of events is not relevant beyond a value of α. We can incorporate α in our definition of the grouping structure and obtain (details and proofs are in the full paper [3]):

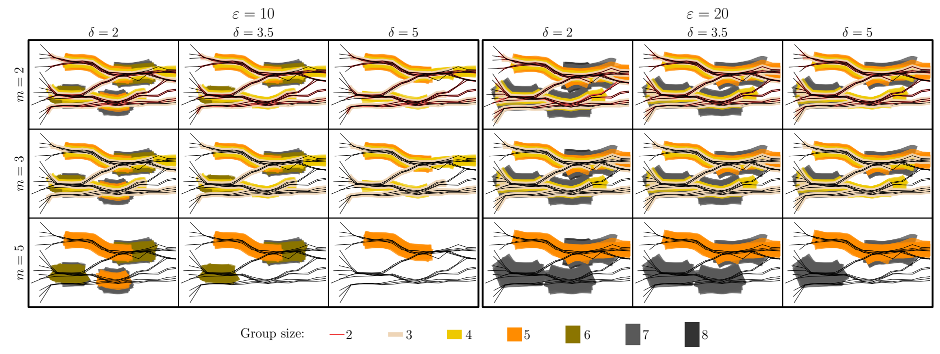

Fig. 4. The maximal groups for varying parameter values. The time associated with each trajectory vertex is proportional to its x-coordinate.

Theorem 5. *Given a set $\mathcal{X}$ of n entities, in which each entity travels along a trajectory of τ edges, we can compute all robust maximal groups in $O(\tau n^3 \log n + N)$ time, where N is the output size.*

5 Evaluation

To see if our model of the grouping structure is practical and indeed captures the grouping behavior of entities we implemented and evaluated our algorithms. We would like to visually inspect the maximal groups identified by our algorithm, and compare this to our intuition of groups. In restricted cases we can show this in a figure, see for example Fig. 4, but for a larger number of trajectories the resulting figures become too cluttered to analyze. So instead we generated short videos.[1]

We use two types of data sets to evaluate our method: a synthetic data set generated using a slightly modified version of the NetLogo Flocking model [21], and a real-world data set consisting of deer, elk, and cattle [15].

NetLogo. We generated several data sets using an adapted version of the NetLogo Flocking model [21]. In our adapted model the entities no longer wrap around the world border, but instead start to turn when they approach the border. Furthermore, we allow small random direction changes for the entities. The data set that we consider here contains 400 trajectories, each with 818 edges. Our videos show all maximal groups for varying parameter values.

The videos show that our model indeed captures the crucial properties of grouping behavior well. We observe that the choice of parameter values is important. In particular, if we make ε too large we see that the entities are loosely coupled, and too many groups are found. Similarly, for large values of m virtually no groups are found. However, for reasonable parameter settings, for example $\varepsilon = 5.25$, $m = 4$, and $\delta = 100$, we can clearly see that our algorithm identifies

[1] See www.staff.science.uu.nl/∼staal006/grouping.

virtually all sets of entities that travel together. Furthermore, if we see a set of entities traveling together that is not identified as group, we indeed see that they disperse quickly after they have come together. The coloring of the line-segments also nicely shows how smaller groups merge into larger ones, and how the larger groups break up into smaller subgroups. This is further evidence that our model captures the grouping behavior well.

Starkey. We also ran our algorithms on a real-world data set, namely on tracking data obtained in the Starkey project [15]. We chose a period of 30 days for which we have the locations of most of the animals. This yields a data set containing 126 trajectories with 1264 vertices each. In the Starkey video we can see that a large group of entities quickly forms in the center, and then slowly splits into multiple smaller groups. We notice that some entities (groups) move closely together, whereas others often stay stationary, or travel separately.

Running times. Since we are mainly interested in how well our model captures the grouping behavior, we do not extensively evaluate the running times of our algorithms. On our desktop system with a AMD Phenom II X2 CPU running at 3.2Ghz our algorithm, implemented in Haskell, computes the grouping structure for our data sets in a few seconds. Even for 160 trajectories with roughly 20 thousand vertices each we can compute and report all maximal groups in three minutes. Most of the time is spent on computing the Reeb graph, in particular on computing the connect/disconnect events.

6 Concluding Remarks

We introduced a trajectory grouping structure which uses Reeb graphs and a notion of persistence for robustness. We showed how to characterize and efficiently compute the maximal groups and group changes in a set of trajectories, and bounded their maximal number. Our paper demonstrates that computational topology provides a mathematically sound way to define grouping of moving entities. The complexity bounds, algorithms and implementation together form the first comprehensive study of grouping. Our videos show that our methods produce results that correspond to human intuition.

Further work includes more extensive experiments together with domain specialists, such as behavioral biologists, to ensure further that the grouping structure captures groups and events in a natural way, and changes in the parameters have the desired effect. Further, our research may be linked to behavioral models of collective motion [18] and provide a (quantifiable) comparison of these.

We expect that for realistic inputs the size of the grouping structure is much smaller than the worst-case bound that we proved. In almost all our initial experiments the number of maximal groups was less than τ. We plan to do further experiments to get a better estimate of this number, and to provide faster algorithms under realistic input models. We will also work on improving the visualization of the maximal groups and the grouping structure, based on the reduced Reeb graph.

References

1. M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, 2008.
2. S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theor. Comput. Sci.*, 392(1-3):5–22, 2008.
3. K. Buchin, M. Buchin, M. J. van Kreveld, B. Speckmann, and F. Staals. Trajectory grouping structures. *CoRR*, abs/1303.6127, 2013.
4. T. K. Dey and Y. Wang. Reeb graphs: approximation and persistence. In *Proc. 27th ACM Symp. on Computational Geometry*, pages 226–235, 2011.
5. H. Edelsbrunner, J. Harer, A. Mascarenhas, V. Pascucci, and J. Snoeyink. Time-varying Reeb graphs for continuous space-time data. *Computational Geometry*, 41(3):149–166, 2008.
6. H. Edelsbrunner and J. L. Harer. *Computational Topology – an introduction.* American Mathematical Society, 2010.
7. M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd International Conference Knowledge Discovery and Data mining*, volume 1996, pages 226–231. AAAI Press, 1996.
8. A. Fomenko and T. Kunii, editors. *Topological Methods for Visualization.* Springer, Tokyo, Japan, 1997.
9. J. Gudmundsson and M. van Kreveld. Computing longest duration flocks in trajectory data. In *Proc. 14th ACM International Symposium on Advances in Geographic Information Systems*, GIS '06, pages 35–42. ACM, 2006.
10. J. Gudmundsson, M. van Kreveld, and B. Speckmann. Efficient detection of patterns in 2D trajectories of moving points. *GeoInformatica*, 11:195–215, 2007.
11. Y. Huang, C. Chen, and P. Dong. Modeling herds and their evolvements from trajectory data. In *Geographic Information Science*, volume 5266 of *LNCS*, pages 90–105. Springer, 2008.
12. H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of convoys in trajectory databases. *PVLDB*, 1:1068–1080, 2008.
13. P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. In *Advances in Spatial and Temporal Databases*, volume 3633 of *LNCS*, pages 364–381. Springer, 2005.
14. Z. Li, B. Ding, J. Han, and R. Kays. Swarm: Mining relaxed temporal moving object clusters. *PVLDB*, 3(1):723–734, 2010.
15. Oregon Department of Fish and Wildlife and the USDA Forest Service. The Starkey project, 2004.
16. S. Parsa. A deterministic $O(m \log m)$ time algorithm for the Reeb graph. In *Proc. 28th ACM Symp. on Computational Geometry*, pages 269–276, 2012.
17. D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362 – 391, 1983.
18. D. Sumpter. *Collective Animal Behavior.* Princeton University Press, 2010.
19. M. R. Vieira, P. Bakalov, and V. J. Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *Proc. 17th ACM International Conference on Advances in Geographic Information Systems*, GIS '09, pages 286–295. ACM, 2009.
20. Y. Wang, E.-P. Lim, and S.-Y. Hwang. Efficient algorithms for mining maximal valid groups. *The VLDB Journal*, 17(3):515–535, May 2008.
21. U. Wilensky. NetLogo flocking model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1998.