Reporting Leadership Patterns Among Trajectories

This extended abstract also appeared in the proceedings of
ACM SAC ASIIS, 11-15 March 2007, Seoul, Korea

Mattias Andersson
Dept. of Comp. Science
Lund University
Sweden

Joachim Gudmundsson
NICTA*
Sydney, Australia

Patrick Laube
School of Geography and
Environmental Science
University of Auckland
New Zealand

Thomas Wolle†
NICTA*
Sydney, Australia

Abstract

Widespread availability of location aware devices (such as GPS receivers) promotes capture of detailed movement trajectories of people, animals, vehicles and other moving objects, opening new options for a better understanding of the processes involved. In this paper we investigate spatio-temporal movement patterns in large tracking data sets. We present a natural definition of the pattern 'one object is leading others', and discuss how such leadership patterns can be computed from a group of moving entities. The proposed definition is based on behavioural patterns discussed in the behavioural ecology literature. We also present several algorithms for computing the pattern, and they are analysed both theoretically and experimentally.

Categories and Subject Descriptors: G.4 [Mathematics of Computing]: Analysis of algorithms and problem complexity; J.4 [Computer Applications]: Social and behavioral sciences

Keywords: tracking data, movement patterns, leadership, computational geometry, agent-based simulation

1 Introduction

Movement is the spatio-temporal process par excellence. Technological advances of location-aware devices, surveillance systems and electronic transaction networks produce more and more opportunities to trace moving individuals. Consequently, an eclectic set of disciplines including geography, data base research, animal behaviour research, surveillance and security analysis, transport analysis and market research shows an increasing interest in movement patterns of entities moving in various spaces over various times scales [1, 4, 7].

Recently there has been considerable research in the area of analysing and modelling spatio-temporal data. In the database community there is ample research on moving object databases (MOD) [7] and generalisations from spatial data queries to spatio-temporal data queries. Whereas most database research on MOD focuses on data structures, indexing and efficient querying techniques for moving objects, only recently the potential of data mining for movement patterns has been acknowledged [9]. The focus within data mining research is to design techniques to discover new patterns in large repositories of spatio-temporal data.

Precursory to this research Laube et al. [10] proposed the REMO framework (RElative MOtion) which defines similar behaviour in groups of entities. They defined movement patterns such as 'flock', 'trend-setting', and 'leadership' based on similar movement properties such as speed, acceleration, or movement direction, and gave algorithms to compute them efficiently. Benkert et al. [2] and Gudmundsson and van Kreveld [5] recently revisited the flock pattern and gave a more generic definition that bases purely on the geometric arrangement of the moving entities and thus excludes the need for an analytical space as with the initial definition of the patterns. The present paper achieves a similar redefinition of the leadership pattern (LP) and will provide a generic, geometric definition as well as efficient algorithms for its detection.

The idea and the term of leadership has been used in several different contexts in the field of animal behaviour research [3]. In general, one can distinguish two different readings. The first describes the event or process of one animal actively initiating a group behaviour, such as leading a group of heifers to a new feeding place. The second reading of leadership includes a spatial configuration of the leader being in front of the others. Animals in front are considered to be more relevant to determine where the group will graze. In this paper we exclusively investigate the latter reading of leadership, since this pattern expresses an interesting geometrical arrangement of its contributers.

Here we use a modified NetLogo Flocking Model to generate trajectories as a testbed for our pattern detection al-

*National ICT Australia Ltd is funded through the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council
†DMiST-project, National ICT Australia Ltd, Locked Bay 9013, Alexandria NSW 1435, Australia. thomas.wolle@nicta.com.au

SAC'07 March 11-15, 2007, Seoul, Korea

gorithms. The use of the geometrical arrangement of moving entities has furthermore a long tradition for realistically modelling group behaviour, be it in animal behaviour science [6] or in the animation industry [11]. Most prominent is the flocking model implemented in NetLogo [12], which mimics the flocking of birds.

As can be seen from the pattern terminology, this research is largely inspired by movement patterns observed in gregarious animals, such as flocking sheep or schooling fish. It follows a strategy to link the proposed patterns as close as possible to observable patterns. The proposed pattern definitions are based on behavioural patterns discussed in the behavioural ecology literature and used for the modelling of realistic movement patterns of agent-based virtual life forms [8]. Even though the leadership pattern in this paper is motivated and investigated with respect to animal behaviour research, its definition is held generic and is thus applicable to arbitrary types of entities.

1.1 Preliminaries

We consider n entities moving in the two dimensional Euclidean plane during the time interval $[t_1, t_\tau]$, see Figure 1(a) for an example. The infinite set T_p of time-points is defined as $T_p = \{t \mid t \in [t_1, t_\tau]\}$, and the set $T_s = \{t_1, t_2, \ldots, t_\tau\}$ of time-steps is the set of discrete time-points given as input. We specify open and closed time intervals by (t_x, t_y) and $[t_x, t_y]$, respectively. A *unit-time-interval* is an open interval I between two consecutive time-steps. At time-point t_x, an entity e_j is located at a position with coordinates $xpos(e_j, t_x)$ and $ypos(e_j, t_x)$. As we do not have spatial information of an entity between two time-steps we make the following assumption for the remainder of this paper (see [2, 5] for a discussion).

Assumption 1. We assume that all entities move between two consecutive time-steps with constant direction and constant velocity.

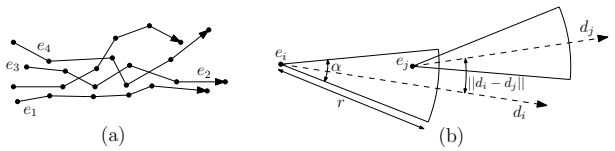

Figure 1: (a) Four entities moving from left to right. (b) Illustrating a *front region*.

Suppose we are given an entity e_j at time-point t with $t_{x-1} < t < t_x$ for $t_x \in \{t_2, ..., t_\tau\}$. We say e_j is heading into *direction* d (also denoted as $d(e_j)$) at time t, where d is an angle in $[0, 2\pi)$ that is specified by the line segment e_j is moving along between time-steps t_{x-1} and t_x. We declare the direction of an entity at a time-step t_x to be *undefined*, because at time-steps an entity might change its direction. However, the *direction* of an entity e_j at a time-step t_x *with respect to* (t_{x-1}, t_x) is the direction e_i is heading to at any time-point in (t_{x-1}, t_x). The difference between two directions d_1 and d_2 is denoted by $||d_1 - d_2||$, which is an angle in $[0, \pi]$.

Given an entity e and a time-point $t \notin T_s$, we define the *front-region* of e at time t in the following way, where r and $\alpha \leq 2\pi$ are assumed to be given non-negative real constants. Consider the disk C with radius r centred at $(xpos(e,t), ypos(e,t))$. Furthermore, consider three line segments s_1, s_2 and s of length r, all having one endpoint at $(xpos(e,t), ypos(e,t))$. Segment s points in the direction d that e is heading to at time t, and segments s_1 and s_2 are the well defined segments forming angles of $\frac{\alpha}{2}$ and $-\frac{\alpha}{2}$ with s, respectively. The part of the disk C that contains s and is bounded by the segments s_1 and s_2 is the *front-region*, see Figure 1(b). We denote this wedge-shaped region by $front(e)$ *at time* t. An entity e_j is said to be *in front* of an entity e_i at time $t \notin T_s$ if and only if $e_j \in front(e_i)$ at time t. If we have an additional requirement on their directions we can define when e_i is following e_j. Let β be constant with $0 \leq \beta \leq \pi$.

Definition 1. Let d_i and d_j be the directions of the entities e_i and e_j at time $t \notin T_s$, respectively. Entity e_i is said to *follow* e_j at time t if and only if $e_j \in front(e_i)$ at time t and $||d_i - d_j|| \leq \beta$.

An entity e_i is said to follow entity e_j at time $[t_x, t_y]$ for time-points t_x, t_y, if and only if e_i follows e_j at time t for all time-points $t \in [t_x, t_y] \setminus T_s$. Leadership patterns are characterised by the size (number of followers) and the duration (number of unit-time-intervals) of the pattern, specified by two constants m and k.

Definition 2. An entity e_j is said to be a *leader* at time $[t_x, t_y]$ for time-points t_x, t_y, if and only if e_j does not follow anyone at time $[t_x, t_y]$, and e_j is followed by sufficiently many entities at time $[t_x, t_y]$. We have a *leadership pattern* if there is an entity that is a leader of at least m entities for at least k unit-time-intervals.

Lemma 1. Let e_i and e_j be two entities, and let t_{x-1} and t_x be two consecutive time-steps. If e_i follows e_j at time-points t_y and t_z with $t_{x-1} < t_y \leq t_z < t_x$ then under Assumption 1, e_i follows e_j at any time-point $t \in [t_y, t_z]$.

Lemma 2. Given two entities e_i and e_j and two time-steps t_{x-1} and t_x, we can in constant time compute the subinterval of $[t_{x-1}, t_x]$ for which $e_j \in front(e_i)$ and for which e_i follows e_j, under Assumption 1.

As mentioned in related work [2, 5], specifying exactly which of the patterns should be reported is often a subject for discussion. Therefore, we consider the following problems where we assume that m and k are given constants.

LP-report-all: For each entity e, report all occurrences of e being a leader of at least m entities for at least k unit-time-intervals.

LP-max-length: Compute the length of a longest leadership pattern of size at least m, i.e. compute the largest value k^{max} such that there is an entity e that is a leader of at least m entities for k^{max} unit-time-intervals.

LP-max-size Compute the size of a largest leadership pattern of length at least k, i.e. compute the largest value m^{max} such that there is an entity e that is a leader of m^{max} entities for at least k unit-time-intervals.

These problems come in four flavours which are combinations of the granularity of the time axis (discrete vs. continuous) and the consistency of the set of followers (varying vs. non-varying). In the discrete case patterns (and follow behaviour) can only start and end at time-steps, while in the continuous case, patterns can start and end at any time-point. The other variation concerns the set of followers. If there is a subset S of entities such that for each time-point of the duration of the pattern all entities in S (and possibly others) follow the leader, then we call this a non-varying (subset) leadership pattern. In contrast to this, if we allow the subset of followers to change from one unit-time-interval to the next during the duration of the pattern, then we call such a pattern a varying (subset) leadership pattern, as long as always at least m entities are following at each unit-time interval of the pattern.

In this abstract we only consider algorithms for the discrete case. In the full paper we present, for each of the problems in the continuous case, algorithms that solve it in $O(n^2\tau \log n)$ time and $O(n\tau)$ space. We also prove an $\Omega(n^2)$ lower bound for the time complexity in the continuous case.

The paper is organised as follows. In the next section we present $O(n^2\tau)$ time algorithms for both the varying and non-varying case. In Section 3 we show a $(1+\varepsilon)$-approximation algorithm for the varying followers case with $O(\frac{\tau n}{\varepsilon}\log^2 n)$ running time. Then, in Section 4 we discuss the implementation and the experimental results, and in Section 5 we conclude the paper with future research and final remarks. Due to space constraints in this extended abstract we omit all proofs.

2 Exact Algorithms

We consider the discrete case, where patterns can only start and end at time-steps. We first describe arrays storing information about the follow behaviour of the entities with respect to a fixed entity e_i. These arrays will then be used to solve our leadership problems.

2.1 Non-varying Subset of Followers

For an entity e_i to determine whether it is a leader at the time (t_x, t_y), we need to know whether e_i is not following any other entity and whether e_i is followed by sufficiently many entities during (t_x, t_y). We consider e_i at this time as a potential leader, and we compute two arrays called '$notfollowing(t_x)$' and '$follows(t_x, e_j)$'. The array $notfollowing(t_x)$ is a one dimensional array storing nonnegative integers. Such an integer for time-step t_x (for $x \geq 2$; $notfollowing(t_1) = 0$) specifies for how many past consecutive unit-time-intervals (the last one ending at t_x) e_i is not following any other entity. The array $follows(t_x, e_j)$ is a $(\tau \times n-1)$ matrix storing nonnegative integers specifying for how many past consecutive unit-time-intervals (the last one ending at t_x) e_j is following e_i (for $e_j \neq e_i$ and $x \geq 2$; $follows(t_1, e_j) = 0$).

Computing the values of the $notfollowing$ and $follows$-array can be done a in straight forward way using two nested loops. Hence, we can conclude with the following lemma.

Lemma 3. The $notfollowing$ and $follows$-arrays for an entity e_i can be computed in $O(n\tau)$ time and space.

To determine whether an entity e_i is a leader of a non-varying-subset of followers we use the arrays $notfollowing$ and $follows$. We look for time-steps t_x such that $notfollowing(t_x) \geq k$. For each such time-step t_x, we inspect the array $follows(t_x, e_j)$ for $j = 1, ..., n$ and $j \neq i$, and we count the number of times that $follows(t_x, e_j) \geq k$. Let $m(k)$ denote this number. Now we can report e_i as a leader for every time-step t_x for which $m(k) \geq m$. As we only need to traverse our arrays once, this can be done on $O(n\tau)$ time.

So far, we have seen that we can compute in $O(n\tau)$ time and space at which time-steps an entity e_i is a leader. To find all leadership patterns amongst a set of entities we test any entity individually. Note that we only have to store one instance of each array at a time. Because algorithms for computing the maximum length or size of a pattern also build upon the arrays $notfollowing$ and $follows$; we can conclude with the following theorem.

Theorem 1. Suppose we are given a set of n trajectories over τ time-steps. In the discrete case, in $O(n^2\tau)$ time and $O(n\tau)$ space we can:

(i) report all non-varying-subset leadership patterns of size at least m and length at least k,

(ii) compute the longest duration leadership pattern for a non-varying-subset of followers of size at least m, and

(iii) compute largest non-varying-subset of entities that follow a leader for at least k time-steps

2.2 Varying Subset of Followers

For solving the leadership problems with a varying subset of followers, we describe two more arrays. The first one is the array $followers(t_x)$ which is a one-dimensional array storing integers specifying for how many consecutive past unit-time-intervals (the last one ending at t_x) there are at least m entities following entity e_i (for $x \geq 2$; $followers(t_1) = 0$). Another array is the array $followers'(t_x)$ which is a one-dimensional array (for $x \geq 2$; $followers'(t_1) = 0$) storing integers specifying how many entities are following entity e_i at time (t_{x-1}, t_x). These two arrays can be computed by counting appropriate values in the $follows$-array.

Lemma 4. The $followers$ and $followers'$-arrays for an entity e_i can be computed in $O(n\tau)$ time and space.

The variant of the problem where the set of followers can change during the leadership pattern can be solved similarly to the non-varying case. To determine if an entity e_i is a leader of a varying-subset of followers, we use the array $notfollowing(t_x)$ as described above, and furthermore, we utilise the array $followers(t_x)$. Also an algorithms for computing the maximum duration of a pattern is based on the arrays $notfollowing$ and $followers$. The complexity of finding all leadership patterns and the maximum length of a pattern is summarised by the following theorem.

Theorem 2. Suppose we are given n trajectories over τ time-steps. In the discrete case, in $O(n^2\tau)$ time and $O(n\tau)$ space we can: (i) report all varying-subset leadership patterns of size at least m and length at least k, and (ii) compute the length of a longest duration leadership pattern for a varying-subset of followers of size at least m

If we would like to compute the size of a largest varying set of followers that follow e_i for at least k unit-time-intervals, we cannot use the array $followers$ directly as this array contains information only for one specific m. One solution is to use binary search on m and recompute the $followers$ array for each value of m. This adds a $\log n$ factor to the running time. However, we propose a way to compute all minima of all substrings of length k of a sequence in linear time, and we apply this result to the array $followers'$.

Lemma 5. Let be given a sequence L of τ numbers and an integer $k \leq \tau$. Computing the minima of all substrings of L of length k can be done in $O(\tau)$ time.

Now we consider the array $followers'$ as a sequence and we look for at least k consecutive time-steps such that the minimum number of followers in the array $followers'$ during that time is as large as possible. According to Lemma 5, all minima can be computed in $O(\tau)$ time. Checking whether e_i can be a leader can be done using array $notfollowing$. We conclude with the following theorem.

Theorem 3. The size of a largest varying-subset of entities that follow a leader for at least k time-steps can be computed in $O(n^2\tau)$ time and $O(n\tau)$ space in the discrete case.

3 Approximate Varying Subset of Followers

In this section we consider an approximate version of the leadership pattern, where we approximate the front region. Due to the space constraint we only give a brief summary of the approach and the results.

The main idea is to approximate a front region by a constant number of triangles of fixed orientation. Since the triangles have a fixed orientation on can perform fast range counting queries using multi-level data structures, i.e., using range trees with priority search trees as an associated data structure. As a result a $(1+\varepsilon)$-approximation of the $notfollowing$-array and the $follows$-array for every entity in S can be computed in time $O(\frac{\tau n}{\varepsilon\alpha}\log^3 n)$ time using $O(n\log^2 n)$ space, for any constant $0 < \varepsilon < 1$.

In Section 2 it was shown how to extract all leadership patterns from the $notfollowing$-array and the $follows$-array. Thus, putting together the results we conclude this section (assuming α is a given constant):

Theorem 4. Reporting all varying-subset $(1+\varepsilon)$-approximate leadership patterns amongst n trajectories over τ time-steps can be done in $O(\frac{\tau n}{\varepsilon}\log^3 n)$ time and $O(\tau n+n\log^2 n)$ space.

4 Experiments

This section is devoted to reporting the experimental results. The algorithms were implemented in Java and all experiments were performed on a Linux operated PC with an Intel 3GHz processor and 2 GB of main memory.

All input files were generated artificially with a modified version of NetLogo's [13] Flocking Model. In this model there are many parameters to influence the behaviour of the entities and thus also to modify how many flocks and leadership-patterns are created. However, we have no direct control over the exact number or length or size of patterns.

We generated files with variable number of entities (128-4096), two different sizes of the underlying universe U (i.e. coordinate space 512×512 and 1024×1024) and two different characteristics CH (i.e. $CH = u$ and $CH = c$) of the entity distribution. $CH = u$ means that the parameters of the Flocking Model were chosen such that the entities are more uniformly distributed. $CH = c$ means that the parameters of the Flocking Model were chosen such that the entities form few but rather large clusters, and hence, the flocks tend to contain more entities and have a longer duration. The number of time-steps is $\tau = 1000$, thus the largest data sets contains roughly 4 million entries.

4.1 Methods and Results

We performed experiments with two variants of our algorithms. The first one is the method described in Section 2. This method contains (among others) two nested loops ranging over all entities. The disadvantage from a practical point of view is that when looking for entities that might be in a front-region, then also entities that are too far away will be considered. Therefore, our second method tries to overcome this drawback, by dividing the underlying plane into buckets (squares of side-length r). Now when looking for entities that might be in a front-region, only those entities will be considered that are in the nine neighbouring buckets (including the bucket at the centre).

Table 1 shows the results of our algorithms for $m = 10$, $k = 20$, $r = 20$, $\alpha = \pi$ and $\beta = \frac{\pi}{2}$. From our point of view the running times and their asymptotic behaviour are much more interesting than for example the exact number of patterns found as we deal with artificial data.

We observed that the vast majority of the running time is spent on computing the arrays $notfollowing$ and $follows$ (which can be done in $O(n^2\tau)$ time). Once these two arrays are computed, computing more arrays and/or extracting information to solve the leadership problem is very efficient (linear time). Therefore, our methods for the three different leadership problems result almost always in the same running times (they differ on average by less than three percent), as they compute all arrays from scratch. Hence, Table 1 depicts the running times of our methods only for the report-all leadership problem.

Non-Varying vs. Varying. As we could expect running times for the patterns with a varying subset of followers are often higher, as one more array is computed for the 'varying' problems. However, this increase is very marginal compared to other influencing factors. We can also observe that the values for the 'varying' patterns are at least as big as for the 'non-varying' patterns, since a non-varying pattern is also a varying pattern by definition.

Without Buckets vs. With Buckets. The approach to subdivide the space into buckets does not influence the reported values of our methods, however, it can have an impressive impact on the running times. Depending on the input characteristics, we can observe speed-up factors between 2 and 32. The running time of the methods without 'buckets' is clearly quadratic in the number of entities. An asymptotic behaviour of the methods with 'buckets' is more difficult to identify, but note that also this method has a quadratic worst case running time.

Uniform vs. Clustered Distribution. Almost always the input files with characteristic $CH = c$ contain more

n, U, CH	without buckets		with buckets	
	NV	V	NV	V
256, 512^2, c	41	42	13	15
1024, 512^2, c	664	684	192	221
4096, 512^2, c	14904	15046	5250	5651
256, 1024^2, c	33	34	3	4
1024, 1024^2, c	595	622	110	129
4096, 1024^2, c	11143	11300	1478	1705
256, 512^2, u	33	33	4	5
1024, 512^2, u	530	523	48	54
4096, 512^2, u	11126	10979	1024	1037
256, 1024^2, u	31	33	2	3
1024, 1024^2, u	513	516	24	28
4096, 1024^2, u	11202	11295	350	382

Table 1: Running times of our methods for the report-all problem. Reported times are in seconds. NV and V denotes non-varying and varying flocks.

patterns, longer patterns and larger patterns, which was expected. Interestingly these characteristics also indicate that the 'bucket' approach for speeding-up the computations has its limitations, because the speed-up factor of the 'buckets' method is strongly influenced by the characteristics. For $CH = u$, we observe speed-up factors between 5 and 11 for the instances with $U = 512^2$, and between 7 and 32 for the instances with $U = 1024^2$. On the other hand, for $CH = c$, the speed-up factors are between 2 and 4 for the instances with $U = 512^2$, and between 5 and 11 for the instances with $U = 1024^2$. This can be explained by noting that the files with characteristic $CH = c$ contain more and bigger flocks, and hence it is more likely that our algorithms encounter neighbouring buckets that are filled with more entities.

Small vs. Large Universe. The difference between the universe with $U = 512^2$ and $U = 1024^2$ is that the former is much denser when filled with the same number of entities. As a result, in the larger universe ($U = 1024^2$) less and smaller patterns exist. Also the running times are affected. The methods with buckets run faster on instances with a larger universe, because we have more buckets and therefore, buckets are likely to contain less entities on average.

5 Conclusion and Final Remarks

Movement patterns detect structure in large tracking data sets, and are thus key to a better understanding of the interactions amongst moving agents. In this paper we present the formal notion of a pattern called 'leadership', describing the event or process of one individual in front leading the movement of a group. Our approach is inspired by movement patterns documented in the animal behaviour and behavioural ecology literature. We provide a formal description of the pattern 'leadership' and subsequently algorithms for its efficient detection. The resulting running times match the theoretical bounds, however for improved methods (with buckets) the running times strongly depend on the characteristics of the instance. The methods in [2] are faster but only report patterns of a specified length with a specified start- and end-time. The methods in this paper, however, are more flexible. Once the arrays $notfollowing$ and $follows$ are computed we can very efficiently use them to report patterns of different lengths, and with different start- and end-times.

We also implemented the approximation algorithm and performed initial experiments. They show a better asymptotic behaviour of the approximation algorithm. However, the constant factors seem to be too large for practical purposes, because for our test-files the exact algorithms always outperformed the approximation algorithm.

One drawback of the given definition of leadership is that a leader has to be in the front region of all followers, as some entities at the end of the flock are too far away from the front-line to be able to see leading animals. Hence, one direction for future research could be the definition and analysis of cascading leaders or followers.

For the many fields interested in movement, the overall challenge lies in relating movement patterns with the surrounding environment, in order to understand *where*, *when* and ultimately *why* the agents move the way they do. Conceptualising detectable movement patterns and the development of algorithms for their detection is a first important step towards this ambitious long-term goal. With its traditional spatial awareness, computational geometry can make immense contributions to the theoretical framework underlying movement analysis in geographical information science, behavioural ecology or security analysis.

Acknowledgements. The authors wish to thank Bojan Djordjevic for implementing the algorithms.

References

[1] Wildlife tracking projects with GPS GSM collars, 2006. `www.environmental-studies.de/projects/projects.html`.

[2] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. Reporting flock patterns. In *Proc. of the 14th Annual European Symposium on Algorithms (ESA 2006)*, volume 4168 of *Lecture Notes in Computer Science*, pages 660–671. Springer-Verlag, 2006.

[3] B. Dumont, A. Boissy, C. Achard, A. M. Sibbald, and H. W. Erhard. Consistency of animal order in spontaneous group movements allows the measurement of leadership in a group of grazing heifers. *Applied Animal Behaviour Science*, 95(1-2):55–66, 2005.

[4] A. U. Frank. Socio-Economic Units: Their Life and Motion. In A. U. Frank, J. Raper, and J. P. Cheylan, editors, *Life and motion of socio-economic units*, volume 8 of *GISDATA*, pages 21–34. Taylor & Francis, 2001.

[5] J. Gudmundsson and M. van Kreveld. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th ACM Symp. on Advances in GIS*, 2006.

[6] S. Gueron and S. A. Levin. Self-Organization of Front Patterns in Large Wildebeest Herds. *Journal of theoretical Biology*, 165(4):541–552, 1993.

[7] R.H. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann Publishers, 2005.

[8] Y. Inada and K. Kawachi. Order and flexibility in the motion of fish schools. *Journal of theoretical Biology*, 214(3):371–387, 2002.

[9] G. Kollios, S. Sclaroff, and M. Betke. Motion mining: discovering spatio-temporal patterns in databases of human motion. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2001.

[10] P. Laube, S. Imfeld, and R. Weibel. Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science*, 19(6):639–668, 2005.

[11] C.W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Conference on Computer graphics and interactive techniques*, volume 21, pages 25–34. ACM Press, 1987.

[12] S. Tisue and U. Wilensky. NetLogo: A Simple Environment for Modeling Complexity. In *International Conference on Complex Systems*, Boston, 2004.

[13] U. Wilensky. NetLogo (and NetLogo User Manual), 1999. `http://ccl.northwestern.edu/netlogo`.