Transactions in GIS, 2011, ••(••): ••–••

•• 1

Discovering Chasing Behavior in Moving Object Trajectories

Fernando de Lucca
Siqueira and
Vania Bogorny
[fernandols, vania] @inf.ufsc.br
Departamento de Informatica e Estatistica
UFSC, Florianopolis, Brazil

Abstract

With the increasing use of mobile devices, a lot of tracks of movement of objects are being collected. The advanced trajectory data mining research has allowed the discovery of many types of patterns from these data, like flocks, leadership, avoidance, frequent sequences, and other types of patterns. In this paper we introduce a new kind of pattern: a chasing behavior between trajectories. We present the main characteristics of chasing and propose a new method that extracts these new kind of trajectory behavior pattern, considering time, distance, and speed as the main thresholds. Experimental results show that our method finds patterns that ~~not are~~ discovered by related approaches.

1 Introduction and Motivation

Modern tracking technology like GPS, cellphones and even sensor networks are being heavily used in many different ways. This use produces spatio-temporal data that are typically large and confused, and do not show/provide any visible information or knowledge. The spatio-temporal data generated by mobile devices, called *trajectories of moving objects*, provide characteristics of space and time, therefore making it possible to analyze where something happened and when it happened. Trajectory data can be interesting and useful in several application domains, like for instance, urban traffic, natural disasters, migration of birds and human mobility. For these applications, trajectory data can express different behaviors through space and time, e.g. move faster, change direction, stand still, repeat the same route, etc.

The identification of different types of behaviors can help the user of an application to understand why something happened or what was the cause of certain actions. For

doi: 10.1111/j.1467-9671.2011.01285.x

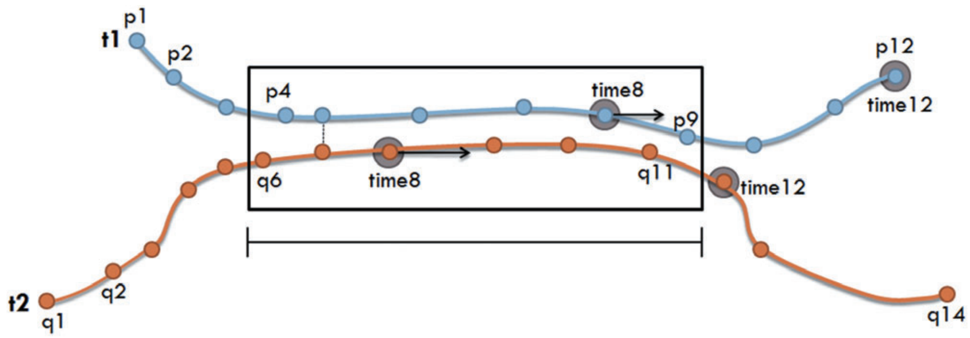


Figure 1 A chasing between trajectories

instance, if an object follows the same route everyday and one day changes it, something could have forced the change, such as a traffic jam. Or when in a large set of trajectories one object is avoiding a region, and this region is for instance a security camera, this object could be a thief. If in this case there has been a robbery in the same region, we could find a suspect.

There are many studies in the literature trying to extract different behavior patterns and more meaningful information from trajectories, as for instance, Baglioni et al. 2009, Cao et al. 2005, Loy et al. 2011, Palma et al. 2008, Rocha et al. 2010, Wachowicz et al. 2011. Although different types of patterns of movement have been identified like flocks, leadership, co-location episodes, convergence, and so on, we lacked a new kind of behavior pattern: *chasing*. In this article we introduce a new trajectory behavior pattern called chasing, where an object chases another one for a certain amount of time. Figure 1 shows an example of chasing where both trajectories move close to each other, for a certain amount time, and trajectory t_2 (points q_6 to ~~q_{11}~~) is chasing trajectory t_1 (points p_4 to p_9).

Several objects can present chasing behavior, and such a pattern can be interesting in a large number of applications. For instance, someone could be interested in monitoring if the vehicle of important people is being chased by the media, or terrorists, if boats are being chased by pirates, if animals are chasing a prey or even in the computer games domain, if an enemy is chasing a character. The automatic identification of chasing behavior can be useful to identify suspects of crimes like a killer or a thief that follows a victim once or several times until a crime event happens. It can also be interesting to understand the behavior of animals that chase animals of the same or different species. In the soccer game domain (Kang et al. 2006), chasing could be used to interpret the behavior of the players and to define new strategies for the next match. A chasing pattern can be found everywhere, having potential applications that justify this work.

The remaining of this article is organized as follows: In Section 2 we list a set of related works and show how they contribute or fail in identifying chasing behavior. In Section 3 we present the definitions of the problem that will help to understand the proposed method. Section 4 describes a new algorithm and how we use the definitions for finding patterns. In Section 5 we show some experiments over three sets of data. Section 6 presents the parameter analysis and finally, Section 7 presents the conclusion and future works.

2 Related Works

In this section we present some works that try to identify different types of patterns in trajectories. We can divide these works in three main groups.

The first group tries to identify patterns in groups of trajectories, where one trajectory is not aware of the other one, i.e., patterns are extracted among objects that follow the same path by coincidence, as in ~~Gianotti~~ et al. 2007, Lee et al. 2008, 2
Hornsby and Cole 2007, Cao et al. ~~2009~~. The objective is to extract patterns among 3
objects with similar movement, but without common intentional behavior. Giannotti proposed an algorithm to extract sequences of regions, frequently visited in a specified order and with similar transition times. Trajectory patterns are generated as sequences of regions visited by a minimal number of trajectories. Lee proposes a method to classify sub-trajectories with different behaviors and different goals. Trajectories with the same goal (discovered by the method) are added to the same group. For instance, ship trajectories that stop at a container port are classified into container ships, while trajectories that stop at a fishering area, for instance, are classified into fishing ships. Hornsby defines a model to represent groups of trajectories that have frequent sequences of events. Cao focuses on the frequent spatio-temporal sequential patterns problem. As an object can use several routes to get to the same place, both the beginning and the end of the trajectory must be the same to generate a sequential pattern. It uses the direction, length, and distance to find similarity between parts of trajectories.

The second group of works tries to identify patterns in single trajectories, trying to understand the behavior of the object by analyzing individual movements. ~~Thietbohl~~, for instance, uses the idea of stops and moves ~~[19]~~ to generate clusters (regions) with low 4
speed, considered as the interesting places in trajectories. Loy addresses a new behavior of trajectories, the *avoidance*. The objective is to find if a trajectory avoids a point, like a thief avoiding a surveillance camera. It also evaluates the confidence of the pattern, to ensure that it was an intentional avoidance. ~~Manso~~ proposes an algorithm to find places 5
in single trajectories where the direction change characterizes the behavior, as for instance a vessel in a fishing region. In ~~[1]~~, individual trajectories are enriched with semantic information obtained from ontologies to infer the goal of the trajectory. For example, a trajectory that always starts at the same place (e.g. home) and stops every day at the same place (e.g. work) is a worker trajectory. A trajectory that stops several times at touristic places is a trajectory of a tourist.

The works presented in the two ~~first~~ groups formally define the patterns and propose algorithms to extract the patterns from trajectories. There is a third group that defines a set of behaviors more conceptually. For instance, Laube et al. (2005) define five types of trajectory behavior patterns: Convergence, Encounter, Recurrence, Flock and Leadership. Two patterns are closer to our work: Flock and Leadership. The Flock pattern refers to a group of objects that move in the same direction at the same time. It traces a circle around a single object and searches for others inside this area that are moving in the same direction at the same time. The Leadership pattern makes a small addition to the previous one: the leader object of the pattern must be moving in a certain direction, and after a certain amount of time, other objects near to the first one start to move into the same direction as well. Both patterns use time, location, direction and distance to identify these behaviors, but neither the speed nor the length of the pattern is considered. The time is only used to assure a minimum duration of the behavior.

Dodge et al. (2008) proposes a conceptual framework of movement and a taxonomy of movement patterns with their definitions. This work uses existing approaches, like the works of Laube (2005) and Cao (2005), to define a set of measures to identify movement patterns, like for instance, distance, speed, direction, etc. He divided movement patterns in two groups: *generic patterns* and *behavioral patterns*. The main difference is that behavioral patterns are context-dependent, as for instance, the common movement of certain animal species. One of the defined behavioral patterns was pursuit/evasion, which refers to an animal trying to escape from a predator where both trajectories have high-speed movements and several turnings. Dodge only describes the movement patterns and the associated measures, without providing any formal definition or algorithm specific for finding chasing, which is the proposal of this article. Legendre et al. (2006) proposed a new modeling approach to mobility data. This work defines motions of objects with behavioral rules, where one object should have a certain behavior depending on a certain context. For example, the individual walk of an object in a certain context should follow some rules like avoid walls, avoid obstacles and avoid other objects. A chasing behavior is also defined, where one object moves in direction to another object or to a static point, but neither time nor distance is considered for really identifying if an object chases another one for a certain time. 6 7

In Hornsby and King (2008) a set of motion relations between vehicles on road network is presented. These relations, *isBehind*, *inFrontOf*, *driveBeside* and *passBy*, describe the relative position between two vehicles in a particular time, as for instance, if one vehicle is in front or behind the other. A chasing could be interpreted as the object that is behind a target for a certain amount of time. However, the relationship is defined only for a specific timestamp, not for a time interval, and only when one object is exactly behind the other, with the same timestamp. Indeed, no duration is taken into account to discover for how long the relation holds, and also if the object is not exactly behind the other, no chasing is characterized.

Reynolds (1999) addresses the problem of autonomous characters, movement in a virtual world. He defines a set of actions and movements, like seek, pursuit, flee and wander, that together model the steering behavior. The main objective is to set a path to be followed by the character given a certain goal as, for example, to follow a corridor avoiding obstacles. The pursuit behavior characterizes a chase, but differently from our proposal, the objective is to define a path to be followed by an object in real time. Our proposal is to find chasing behavior thorough the analysis of past trajectories, and not to define the route of an object.

Wachowicz (2011) extended the work of Laube, and proposed an algorithm to find flocks between objects that must be moving together during a certain amount of time. In this approach the objects may not stand without moving. In this approach the direction is not considered. Apart from this, the main problem is that the moving objects must remain in flocks at exactly the same timestamps, which is not exactly the case for chasing.

Cao et al. (2006) explore the collocation episodes in spatio-temporal data. For example, a puma hunting a dear. The main objective is to find objects that move together for a certain amount of time and make another object move with them. Therefore, the concept of time window is used, where trajectories are divided in time slices, and then each slice is evaluated to discover an episode. The relationship between objects is identified through the distance between points in each time window. The time is used to assure a minimum time duration and for the time window, but also to find a pattern a requirement is that trajectories must have the same timestamps inside the time window.

This restriction limits the method when trajectories were collected at time intervals which were similar but not identical, ~~but not~~ and also when the trajectories were collected with different time intervals (e.g. a trajectory collected every 1 second and another every 2 or more seconds).

Finding chasing patterns in trajectories is not the objective of Cao et al. Although distance and time are considered to find objects that are close in space, the way it uses these parameters is not enough to characterize chasing.

Previous works could somehow be used to identify chasing patterns, but none of them considered sufficient characteristics for really identifying it. While most previous works deal with groups of trajectories or simply identify patterns in the trajectory of one individual, here we work with pairs of trajectories, and discover the behavior of one object in relation to another one.

3 Basic Concepts and Definitions

A chasing pattern has some special characteristics that define its behavior. In this section we discuss some definitions that will help the reader to understand this new kind of pattern and our algorithm.

Definition 1. Trajectory. A *trajectory* T is a list of space-time points $\langle tid, p_0, p_1, \ldots, p_n \rangle$, where $p_i = (x_i, y_i, t_i)$ and $x_i, y_i, t_i \in \mathbf{R}$ for $i = 0, \ldots, n$ and $t_0 < t_1 < \ldots < t_n$. Every T is identified by a trajectory identifier called *tid*.

Because a trajectory chasing pattern may not exist in the whole trajectory, we partition a trajectory into sub-trajectories.

Definition 2. sub-trajectory. A *sub-trajectory* S of $T = \langle p0, p1, p2, \ldots, pn \rangle$ is a list of space-time points $\langle p_i, p_{i+1}, \ldots, p_{i+m}, \rangle$, where p_i (T and $0 \leq i \leq m + i \leq n$.

A chasing does not occur between trajectories of different days or with a large time interval. To avoid the comparison of trajectories collected with long time differences, we introduce the concept of time tolerance. A time tolerance Δt is a maximum time interval between two trajectories that ensures that they happened in a near/similar time period. If two trajectories are in the same time period we say that they are a *candidate chasing*.

Definition 3. Candidate Chasing. Let $S1 = \langle p0, p1, \ldots, pn \rangle$ and $S2 = \langle q0, q1, \ldots, qm \rangle$ be sub-trajectories of T1 and T2, respectively. S1 and S2 respect the *time tolerance* Δt if and only if $S2$ $|t_{p0} - t_{q0}| \leq \Delta t$ and $|t_{pn} - t_{qm}| \leq \Delta t$ and $t_{qm} > t_{pn}$.

Figure 2 shows an example of definition 3. Let us consider Δt as 0:05, the pair ($S1$, $S2$) is a candidate chasing because $p_{1t} = 1{:}00$ at S1 and $q_{1t} = 1{:}04$ at S2, so $|p_{1t} - q_{1t}| \leq \Delta t \equiv |1{:}00 - 1{:}04| \leq 0{:}05$ and $p_{2t} = 1{:}05$ at S1 and $q_{2t} = 1{:}08$ at S2, so $|p_{2t} - q_{2t}| \leq \Delta t \equiv |1{:}05 - 1{:}08| \leq 0{:}05$.

To reduce the number of points of a sub-trajectory, we build a line segment between the first and the last point of the sub-trajectory, that we call *representative line segment*.

Colour

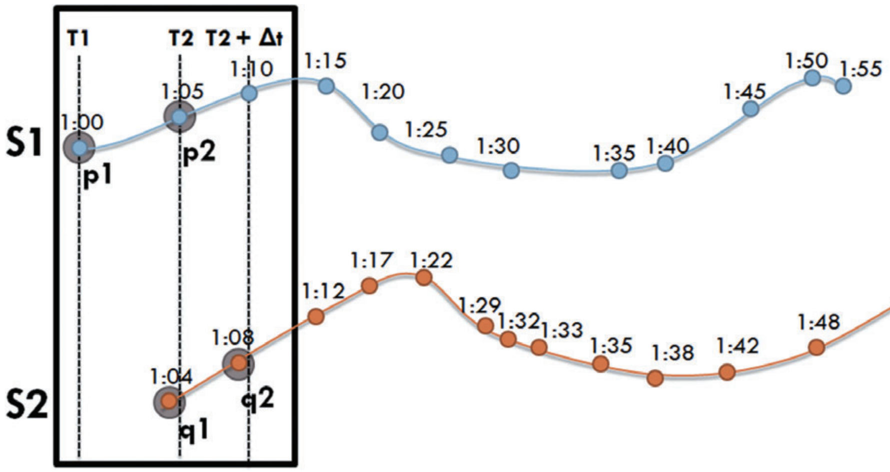


Figure 2 Example of candidate chasing for $\Delta t = 0{:}05$

Colour

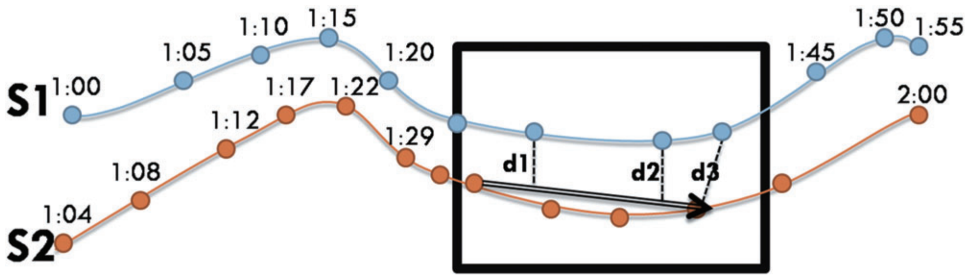


Figure 3 Exemple of potential chasing

Definition 4. Representative Line Segment. Let p be a point of a sub-trajectory $S = \langle p0, p1, \ldots, pn\rangle$, a *Representative Line Segment* L of S is the line segment (p_0, p_n).

Once the two sub-trajectories are a candidate chasing we go to the next step where we see if they are close to each other. Two trajectories being in the same period does not mean that there was a chasing. Two trajectories must be close to each other to characterize a potential chasing.

Definition 5. Potential Chasing. Let $S1 = \langle p0, p1, \ldots, pn\rangle$ and $S2 = \langle q0, q1, \ldots, qm\rangle$ be a candidate chasing, $L1$ be the representative line segment of $S1$, $L2$ be the representative line segment of $S2$. $S1$ and $S2$ are potential chasing with respect to the maximum average distance Δd if and only if $(\Sigma\ distance\ (p_i, L2)/n) \leq d$ where $0 \leq i \leq n$ and $(\Sigma\ distance\ (q_j, L1)/m) \leq d$ where $0 \leq j \leq m$.

In Figure 3 we have an example of a potential chasing. Note that in definition 5 we verify the closeness between both representative line segments. This way, we make sure that a pair of trajectories, as the example shown in Figure 4, are *not* a potential chasing. As shown in Figure 4 (a), the sub-trajectory S2 (q7, q8, q9, q10) is close to the

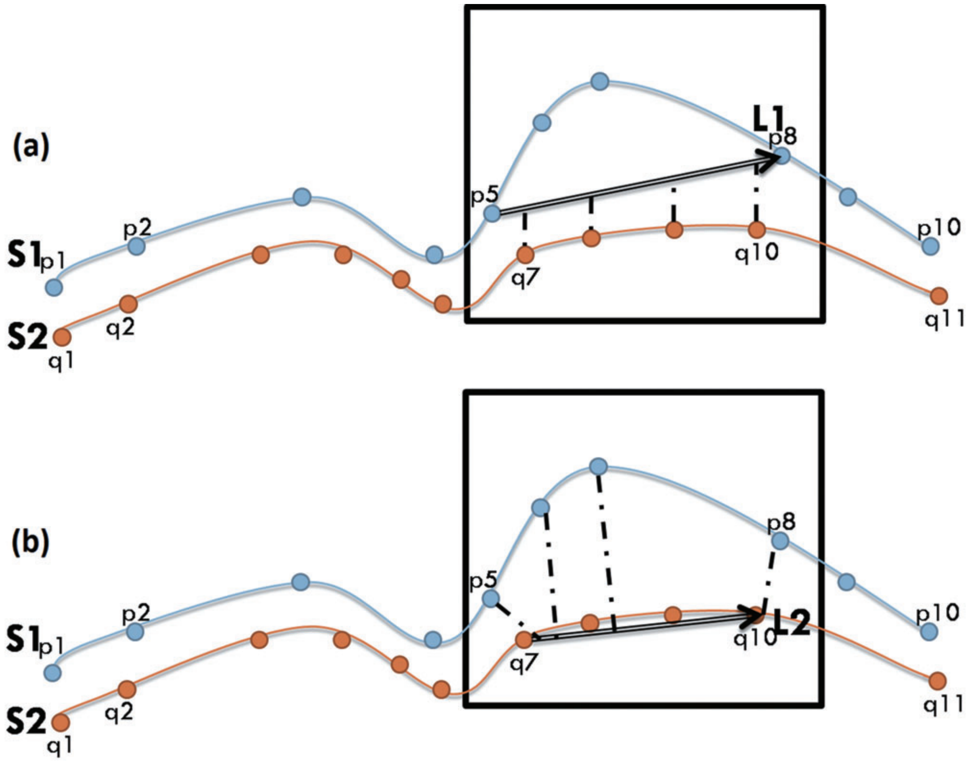


Figure 4 (a) potential chasing is detected and (b) potential chasing is not confirmed

representative line L1 (p5, p8) of S1. But when we compare the distance between sub-trajectory S1 (p5, p6, p7, and p8) in Figure 4 (b), with the line segment L2 (q7, q10), the minimal distance threshold is not satisfied, so not characterizing a potential chasing in this case.

In some applications the speed can also indicate a chasing. When an object is chasing another one, their average speed must be similar, or one object will move far away from the other. This can occur in some types of chasing like a thief chasing a victim or a police chasing a suspect. In this article we consider the speed as an optional factor, and evaluate chasing with and without using speed.

> **Definition 6.** Speed. Let $S1$ and $S2$ be a potential chasing, $\Delta a1$ and $\Delta a2$ be the average speed of $S1$ and $S2$, respectively, and α be the maximum percentage difference between speeds, both sub-trajectories will have the same *average speed* if $(1 - \alpha) \geq (\Delta a1/\Delta a2) \geq (1 + \alpha)$ with α ([0,1]

Notice that to find a chasing considering speed, the speed of both trajectories can be either high or low. What matters is that the speed must be similar.

With these definitions we can finally define a chasing behavior:

> **Definition 7.** Sub-Chasing. Let $S1$ and $S2$ be two candidate chasing trajectories with respect to a time tolerance Δt, if $S1$ and $S2$ are a potential chasing, we have a sub-chasing where $S2$ is chasing $S1$.

To have a chasing pattern we need two trajectories: One being chased and another chasing. We name the first one as target, because it is the target of the chasing, and the second as stalker, because it is chasing the target.

Definition 8. Pure-Chasing. A trajectory $T1$ named as target is being chased by a trajectory $T2$, named stalker, if Σ *duration* of the set of sub-chasing between $T1$ and $T2$ is greater then a minimum time duration Δc.

In others words, a chasing pattern is detected when two trajectories remain close to each other for a period of time and respecting a time tolerance.

Definition 9. Speed-Chasing. A trajectory $T1$ named as target is being chased by a trajectory $T2$, named stalker, if Σ *duration* of the set of sub-chasing between $T1$ and $T2$ is greater then a minimum time duration Δc and the average speed of $T1$ is the same as $T2$.

Based on the above definitions we can finally define an algorithm to find chasing patterns, which is presented in the following section.

4 TRA-CHASE: An Algorithm to Identify Trajectory Chasing Patterns

In this section we present an algorithm to identify trajectory chasing patterns, named TRA-CHASE. In general words, this algorithm, shown in listing 1.1, tries to identify sub-trajectories that contain a chasing pattern. The part that identifies sub-trajectories with a chasing pattern is presented in the SUB-CHASE procedure, shown in listing 1.3. The set of sub-chases identified between two trajectories will result in the TRA-CHASE pattern.

The TRA-CHASE algorithm takes as input a set of trajectories T of different objects, the minimum time duration of the pattern Δc, the time tolerance Δt and the maximum allowed distance Δd between trajectories that characterizes a chasing. The speed parameter is a flag that tells the algorithm if it should either consider speed or not for computing chasing patterns.

For each pair of different trajectories (lines 12, 13, 15), the algorithm analyses every sub-trajectory (lines 22 an 30) to find if there are two sub-trajectories having a chasing pattern. Then, the algorithm moves to the next step until it covers the complete trajectory.

The first step is to create the sub-trajectories $P1$ and $P2$ (lines 17 and 19), with the two initial points of trajectories $t1$ and $t2$, respectively. Having these two points $P1$ and $P2$, the algorithm generates sub-trajectories $S1$ and $S2$ (lines 20 and 21) with the method *GETsub-trajectory* (shown in listing 1.2). The objective of this step is to optimize the algorithm, avoinding point to point comparison of both trajectories. The next step is to identify chasing behavior between the sub-trajectories $S1$ and $S2$ (line 22). If a sub-chasing is found between the sub-trajectories, both $S1$ and $S2$ are added to the set of chasing patterns C (line 24), and the algorithm jumps to the timestamps of $t1$ and $t2$ (lines 25 and 27) that correspond to the final timestamps of $S1$ and $S2$, respectively.

In case no sub-chasing is found between $S1$ and $S2$, then we have to test point by point, and the algorithm searches for a pattern between $P1$ and $P2$ (line 30). If there is a pattern between $P1$ and $P2$ (line 32) it is added to C (line 32), and the algorithm jumps

Listing 1.1. TRA-CHASE pseudocode

```
1  INPUT:
2  T: group of trajectories
3  Δc: minimum duration
4  Δt: time tolerance
5  Δd: distance
6  speed:
7
8  OUTPUT:
9  chasingSet: a group of chasing pattern
10
11 METHOD:
12 FOR EACH(t1 in T)
13 FOR EACH(t2 in T)
14  C = new CHASINGPATTERN;
15  IF(t1.tid ≠ t2.tid) //to be sure is different trajectories
16   WHILE(t1 has next point)
17     P1 = (t1.currentPoint, t1.nextPoint);
18     WHILE(t2 has next poiny)
19     P2 = (t2.currentPoint, t2.nextPoint);
20     S1 = GETsub−trajectory(P1,t1,Δt);
21     S2 = GETsub−trajectory(P2,t2,Δt);
22      chase = SUB−CHASE(S1, S2, Δt, Δd, speed); //tests sub−trajectory chasing
23      IF (chase)
24       C.add(S1,S2);
25       t1.moveToTime(S1.endTime); //move to the point in t1 that corresponds
26       //to the last point of S1
27       t2.moveToTime(S2.endTime); //move to the point in t2 that corresponds
28       //to the last point of S2
29      ELSE
30       chase = SUB−CHASE(P1, P2, Δt, Δd, speed); //tests chasing between two points
31       IF (chase)
32        C.add(P1,P2);
33        t1.moveToNextPoint; //move the trajectory t1 to the next point
34        t2.moveToNextPoint; //move the trajectory t2 to the next point
35       ELSE
36        t2.moveToNextPoint;
37     END WHILE
38       t1.moveToNextPoint;
39    END WHILE
40    IF C.DURATION ≥ Δd
41     chasingSet.add(C);
42    END IF
43   END IF
44  END FOR
45 END FOR
46 RETURN chasingSet;
47 END METHOD
```

to the next point of $t1$ and $t2$ (lines 33 and 34) and starts the process again. In case no chasing behavior is found between $P1$ and $P2$ the algorithm moves to the next point of $t2$ (line 36), looking for a sub-trajectory of $t2$ that may have a chasing with the current sub-trajectory of $t1$. Finally, if no sub-trajectories of $t2$ have chasing behavior with the current sub-trajectory of $t1$, the algorithm moves to the next point of $t1$ (line 38), and the process starts again.

At the end, if the duration of the subchases is higher than the minimum duration Δd (line 40), the chasing pattern C is added to the set of chasing patterns *chasingSet* (line 41). The chasing pattern is two lists of points, one from the target that was being chased and the other from the stalker that chased the target.

Listing 1.2 shows the pseudo-code of the method that generates sub-trajectories, grouping them by time. This method has as input a pair of points P of the trajectory t and the time tolerance Δt. The output is a sub-trajectory S. First, the algorithm takes the last point of P (line 8) called q. Then the algorithm goes to the point after q (line 10) and checks if the timestamp of each consecutive point is lower than the timestamp of q plus

Listing 1.2. GETsub-trajectory pseudocode

```
1
2  INPUT:   P: sub-trajectory //a sub-trajectory of two consecutives points
3           t: trajectory //the original trajectory
4           Δt: time tolerance //the time tolerance
5
6  OUTPUT: S: sub-trajectory //a sub-trajectory group by time
7
8  q = P.getLastPoint;
9  S.add(P);
10 t.moveToPoint(q+1); //move to the point after q
11 FOR EACH(Point p of t)
12   IF(p.time <q.time + (Δt/2))
13     S.add(p)
14   ELSE
15     BREAK;
16   END IF
17 END FOR
18 RETURN S;
```

Listing 1.3. SUB-CHASE pseudocode

```
1  INPUT:
2  L1: sub-trajectory target
3  L2: sub-trajectory stalker
4  Δt: time tolerance
5  Δd: distance
6  speed: consider speed
7
8  OUTPUT:
9  chase: IF L2 is chasing L1
10
11 METHOD:
12 Pure-chase = FALSE, Speed-chase=FALSE
13 IF CANDIDATECHASING(L1, L2, Δt)
14  IF POTENTIALCHASING(L1,L2, Δd)
15   IF L1 ahead L2 (L1,L2)
16   IF speed
17    IF SAMEAVERAGESPEED(L1, L2)  //optional in the algorithm
18     chase = TRUE
19   END IF
20   ELSE
21    chase = TRUE
22   END ELSE
23 END IF
24 RETURN chase
25 END METHOD
```

$\Delta t/2$ (line 12).~~n.~~ We group the points by time to overcome the problem when different trajectories are generated with different time intervals. By grouping points to generate sub-trajectories using just the number of points instead of time, we could generate sub-trajectories without chasing characteristics.

To compute the sub-trajectory, first, we have to confirm that both sub-trajectories are in the same time period. So in the SUB-CHASE procedure, presented in listing 1.3, we test (line 13) the time tolerance and find the candidate chasing. In this step we test if the sub-trajectory $S2$ has its ending time before the ending time of $S1$ plus the time tolerance. It means that the trajectory of the stalker must have happened after the target. For example, if the target passes by a local X at time $t1$, the stalker has to pass around local X at a time $t2$ where $t2 \geq t1$. This step prevents comparing two trajectories from different days or with long time difference.

Only if $S2$ respects the time tolerance do we go to the next step, which analyzes the distance as in definition 5 (line 14), to check if the subchase is a potential chasing. The stalker cannot always act the same way as the target, but he always tries to be close to the target. The stalker can change its behavior over the sub-trajectory, being closer or farther from the target. Therefore, the algorithm uses the average distance to evaluate if both objects remained close. If the average distance between them is less than Δd, then both sub-trajectories are close to each other.

Another matter is the order of the objects. In a regular chasing, the target will always be in front of the stalker, because the stalker must see where the target is heading. This is checked in line 15 of the sub-chase procedure in 1.3.

An important remark is that, in case the roles of the chasing invert, i.e. the target becomes the stalker and the stalker becomes the target, the algorithm will find a new chasing pattern, different from the previous one.

The last analyzed property is the speed (line 17). Sometimes even if two trajectories move together for a certain amount of time, they should move around the same speed. If the target is moving faster then the stalker and the target is moving away, it means that the stalker did not intend to pursue the target, because he did not keep the target on track. On the other hand, if the stalker is moving faster than the target, he will pass the target and keep moving. Based on our definition 6 we check if both sub-trajectories have the same average speed during the same time period. We decide a maximum limit of 20% of difference between the trajectories speed. We considered that if an object is trying to adjust its speed with another one, ~~and~~ a 20% of difference should be a good value to control the distance between them.

Since the speed is optional, we evaluate the algorithm with and without using speed, calling the comparisons as $Pure - Chase$ and $Speed - Chase$, as in the previous definitions.

5 Experimental Results

To evaluate the proposed algorithm we considered three datasets. The first one, shown in Figure 5, was synthetically generated by ~~the Knowledge Discovery and Data Mining research unit~~ to simulate a flock pattern. In this dataset several objects move together at a certain time. The objective of using this dataset is to find chasing patterns among flocks, and to show that we find chasing patterns that are not discovered by the flock algorithm.

The second dataset is from a mobile learning game developed by the Waag Society, in the Netherlands. It is a city game with GPS and mobile phones with students aged 12 to 14. The game consists of various geo-referenced places associated with multimedia riddles and questions. The player receives a historical map with checkpoints and has the role to find these places in the real life. The 466 students were divided in six groups and the game took place in 2005 from 7 to 9 February. In this dataset we try to find chasing behavior between students of different groups. If someone did not ~~discover~~ the riddle and did not know where to go, he can try to follow someone of another group who decrypted the puzzle, therefore characterizing a chasing.

Figure 5 Trajectories of the first dataset

In the third dataset we captured data of a group of people walking with a GPS device at Jurere Internacional beach, located in Florianopolis, Brazil. Differently from the previous dataset, GPS devices produce data that can be diffuse and not linear. In this dataset, the target walked for around 1 hour and a group of 5 possible stalkers walked at different times simulating some chasing behaviors. In this dataset we know who is the target, who are the stalkers, when and where the chasing occurs. The points of the target were captured every 2 seconds and the points of the stalker each 1 second, so we show that our method works very well with trajectories collected at different time intervals.

We compare our algorithm with two others: the Moving Flock Finder (MFF) (Wachowicz et al. 2011~~, Knowledge Discovery and Data Mining Research unit 2011), [4]~~) and the Collocation Discovery Algorithm (CDA) (Cao et al. 2006). We run our algorithm PURE TRA-CHASE(PTC) and SPEED TRA-CHASE(STC).

The code was written in java, the data were stored in a postgres database extended with postgis, and we used the software Quantum GIS to visualize the results.

5.1 Experiments with Dataset 1

In this first experiment we used a subset of 17 trajectories. We run all four algorithms with the same parameters. All of them with the same distance $\Delta d = 80.0\ m$, duration $\Delta c = 10\ min$ and the time tolerance $\Delta t = 5\ min$. We defined 80 meters as the minimun distance because the average distance between the points of a trajectory in this synthetic dataset varies between 40 meters and 160 meters in a time interval of around one minute. The time window parameter for CDA should have the same value as our time tolerance.

Table 1 Comparison of the duration of the patterns found by the different algorithms considering $\Delta d = 80.0\ m$ for PTC, STC, CDA and MFF

Pattern	PTC	STC	CDA	MFF
C1	18:23–19:50	18:25–19:50	18:27–19:47	18:26–19:45
C2	18:16–18:55	18:16–18:49	–	–
C3	18:18–18:35	18:18–18:34	–	–
C4	20:11–20:21	–	–	–
C5	18:18–18:32	18:18–18:28	–	–
C6	18:23–19:50	18:23–19:50	–	–

In this experiment, the flock algorithm (MFF) found one pattern, the co-location (CDA) found two, PTC found six and STC found four patterns, as shown in Table 1. Both MFF and CDA found the same pattern, but since CDA does not differ which object is in front of the other, it found the same pattern twice. The PTC (considering speed) found more patterns then the others, once it does not consider the speed between trajectories. The patterns found by STC were also found by PTC, but they were shorter because the sub-trajectories did not ~~had~~ the same speed.

Figure 6 shows two sub-trajectories S1 and S2 to explain why CDA does not find a chasing that the Pure-chasing or the Speed-chasing find. CDA compares the points with same timestamp, so it compares the points q2 and q3 of sub-trajetory S2 with the points p5 and p6 of sub-trajectory S1, as shown in Figure 6 (a). These points are far from each other, so not respecting the minimal distance. On the other hand, our method PTC compares all points between q2 and q6 of sub-trajectory S2 with the points p5 and p6 of the sub-trajectory S1, as shown in Figure 6 (b). Notice that the areas that cover the points between the sub-trajectories intersect each other, therefore characterizing a chasing pattern.

What we can conclude in this experiment is that the way time is used by the method makes the difference in the discovered patterns.

5.2 Experiments with Dataset 2

In this dataset the objective was to find chasing patterns between individuals of different groups, assuming that if someone does not know where to go, he/she might want to chase someone who knows the next place.

Among the three datasets, this was the largest and more complex. Each trajectory has several points with different time intervals. In the same trajectory, two consecutives points could have from 1 to 60 seconds of difference.

Since the flock algorithm MFF cannot work with data captured at different time intervals, we ran a synchronizer software from Wachowicz et al. (2011) to try to find flock patterns. After analysing the data we ran the four algorithms with both the syncronized data and the original data, with the parameters Δt = [1 minute and 3 minutes] (time needed for one trajectory to catch the other), Δd = [15 meters and 30 meters] and Δc = 10 minutes. The results are shown in Tables 2 and 3.

Even with the synchronized data and considering two different values for both time tolerance and distance, the MFF did not find any pattern. The CDA found ~~many fewer~~ patterns then our method. Both PTC and STC found almost the same patterns, showing

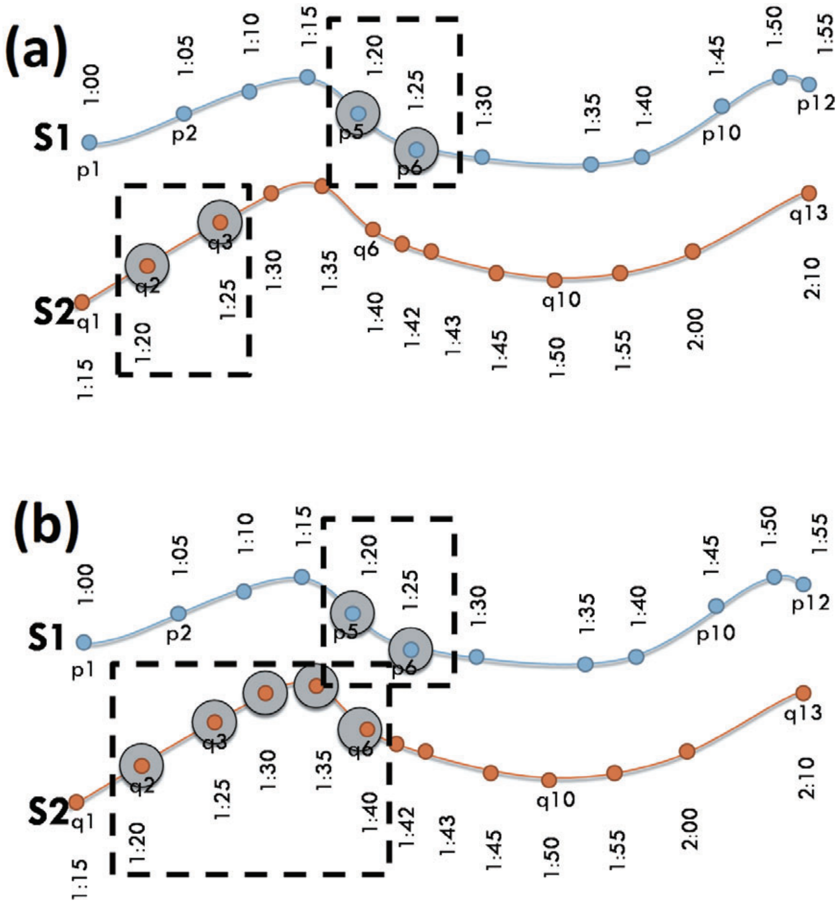


Figure 6 (a) Points compared in CDA and MFF and (b) points compared by PTC and STC

Table 2 Number of patterns found by PTC, STC, CDA and MFF with the syncronized data

~~Test~~	PTC	STC	CDA	MFF
$\Delta d = 15$ m $\Delta t = 1$ minute	105	101	64	0
$\Delta d = 15$ m $\Delta t = 3$ minutes	152	148	33	0
$\Delta d = 30$ m $\Delta t = 1$ minutes	244	240	180	0
$\Delta d = 30$ m $\Delta t = 3$ minutes	297	297	72	0

that the speed of the trajectories was very similar. As can be seen in Tables 2 and 3, and as was expected, by synchronizing the data, the co-location algorithm (CDA) found more patterns than with the original data. The same occurs with our methods, but our two algorithms always find more chasing patterns. In summary, all approaches found more patterns on the synchronized data.

Table 3 Number of patterns found by PTC, STC, CDA and MFF with the original data

~~Test~~	PTC	STC	CDA	MFF
$\Delta d = 15$ m $\Delta t = 1$ minute	93	90	11	0
$\Delta d = 15$ m $\Delta t = 3$ minutes	139	136	4	0
$\Delta d = 30$ m $\Delta t = 1$ minutes	169	167	36	0
$\Delta d = 30$ m $\Delta t = 3$ minutes	217	215	27	0

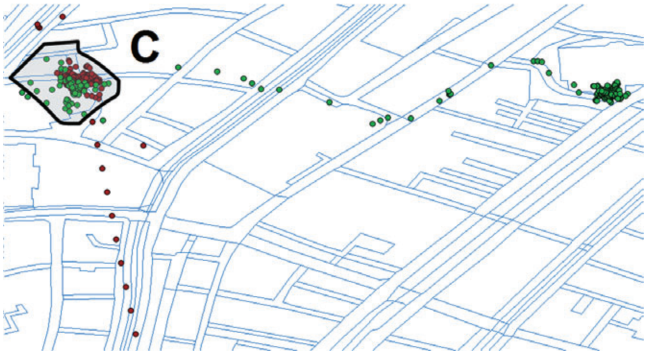


Figure 7 Pattern found by PTC, STC and CDA for $\Delta d = 15$ m $\Delta t = 1$ minute

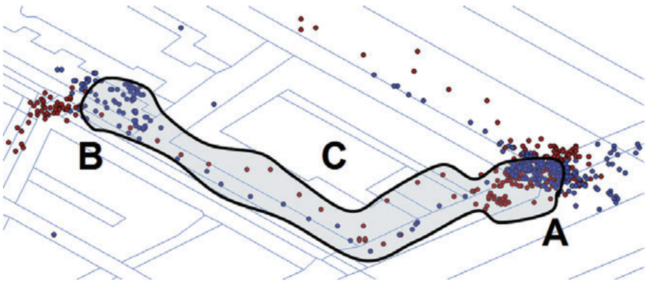


Figure 8 Pattern found by PTC and STC for $\Delta d = 15$ m $\Delta t = 1$ minute

Although there were many chasing patterns, this occurred because most patterns were found in regions where the students ~~stoped~~ moving, or moved slowly. These regions represent either the historical places (where the students were answering the riddles) or where the students were resting. As the points in these regions are very close, the algorithms CDA, PTC and STC found the same patterns. This kind of patterns located in dense regions of points, as can be seen in Figure 7, is the only kind of pattern found by CDA. CDA misses any pattern on the path between regions, while our method found chasing patterns on the path between the regions, as shown in Figure 8.

Figure 8 shows a more interesting kind of pattern that is only found by our method. There are two trajectories moving from region A to region B. The PTC identifies this as a chasing C. A student, after ~~answered~~ the riddles, walks to the next historical place followed by another student.

5.3 Experiments with Dataset 3

In this dataset we know where each chasing starts and finishes, so we can better evaluate the results. We ran the algorithms with distance $\Delta d = 10\ m$, duration $\Delta c = 3\ min$ and time tolerance $\Delta t = 30$ seconds. In pedestrian chasing, 10 meters is an acceptable distance for an object to keep another one under his/her eyes, avoiding being too close. As the time interval between the collected points was 1 second and 2 seconds, where pedestrians walked in low velocity, we considered that 30 seconds for time tolerance is a good measure for an object to be 10 meters behind the other.

This dataset has 5 chasing patterns, where object 0 was chased by objects 1 and 2 once, and by objects 3 and 5 twice. In this experiment, PTC found 6 patterns and STC found the 5 original patterns. Note that these are short trajectories, with duration around 20 to 30 minutes, so the chasing patterns are short too.

Both flock (MFF) and co-location (CDA) algorithms did not find any pattern. Then we ~~ran~~ the experiment with different parameters, and still did not get any ~~instance of~~ pattern from MFF. The algorithm CDA found 3 patterns only when we set $\Delta d = 25\ m$. A comparison of the results is shown in Table 4. By looking at the first row in the table, the pattern C1 represents the original pattern (real duration). Our two algorithms found almost the real pattern. On the other hand, CDA found a pattern before the original chasing.

Figure 9 shows a comparison of the pattern C1 generated by CDA and our algorithms (STC and PTC). Notice that the pattern found by CDA is before the real chasing. For all other patterns, represented in Table 4, our methods (STC and PTC) found patterns very similar to the original ones. The flock algorithm did not find patterns, and CDA found two patterns similar to the original ones (pattern C4a and C4b).

6 Parameter Analysis and Discurssion

An important matter is how to set the appropriate parameters. This is a general problem for most data mining algorithms. One can make several tests until finding the best parameters, and this has a cost and may change from one application to another.

The main contribution of our work is the way that we use the time dimension (time tolerance). The value of the time tolerance should be based on how much time the stalker takes to cross the same region as its target. For instance, in Figure 10 the target S1 entered the region X at time 1:35 and the stalker S2 entered at time 1:50. So the time tolerance in this case should be at least 0:15 in order to the sub-trajectory S1 at time 1:35 be comparable with the sub-trajectory S2 at time 1:50.

As the co-location and flock algorithms use the distance to define closeness at the same timestamp, it may occur that at the same time two sub-trajectories are far from each other. As we use the time tolerance to compare closeness between two sub-trajectories fixing the timestamp of the target and move ahead in time of the stalker, our method finds that two objects are close to each other at different timestamps, therefore characterizing a chasing. An example is given in Figure 10, where existing works would compare the distance between points p5 of S1 and q2 of S2 with a distance d1, while our method would compare point p5 of S1 with q5 of S2, therefore having a distance of d2. In summary, as our method compares one timestamp of the target with several timestamps of the stalker, respecting Δt, we find more realistic chasing.

Table 4 Duration comparison of the patterns found by the different algorithms considering $\Delta d = 10$ *m* for PTC and STC and $\Delta d = 25$ *m* for CDA and MFF

Pattern	Real Duration	PTC $\Delta d = 10$	STC $\Delta d = 10$	CDA $\Delta d = 25$	MFF $\Delta d = 25$
C1	17:12:00–17:25:40	17:09:12–17:23:27	17:10:41–17:23:19	17:07:33–17:10:14	–
C2	17:27:52–17:33:45	17:27:30–17:33:45	17:27:44–17:33:50	–	–
C3a	17:37:00–17:40:40	17:36:48–17:40:45	17:36:48–17:40:45	–	–
C3b	17:44:30–17:50:50	17:43:43–17:50:49	17:44:16–17:51:04	–	–
–	–	17:51:56–17:55:10	–	–	–
C4a	17:56:45–18:00:40	17:57:21–18:00:48	17:57:55–18:00:50	17:55:39–17:58:14	–
C4b	18:02:50–18:09:40	18:03:22–18:09:56	18:03:42–18:10:00	18:04:46–18:09:59	–

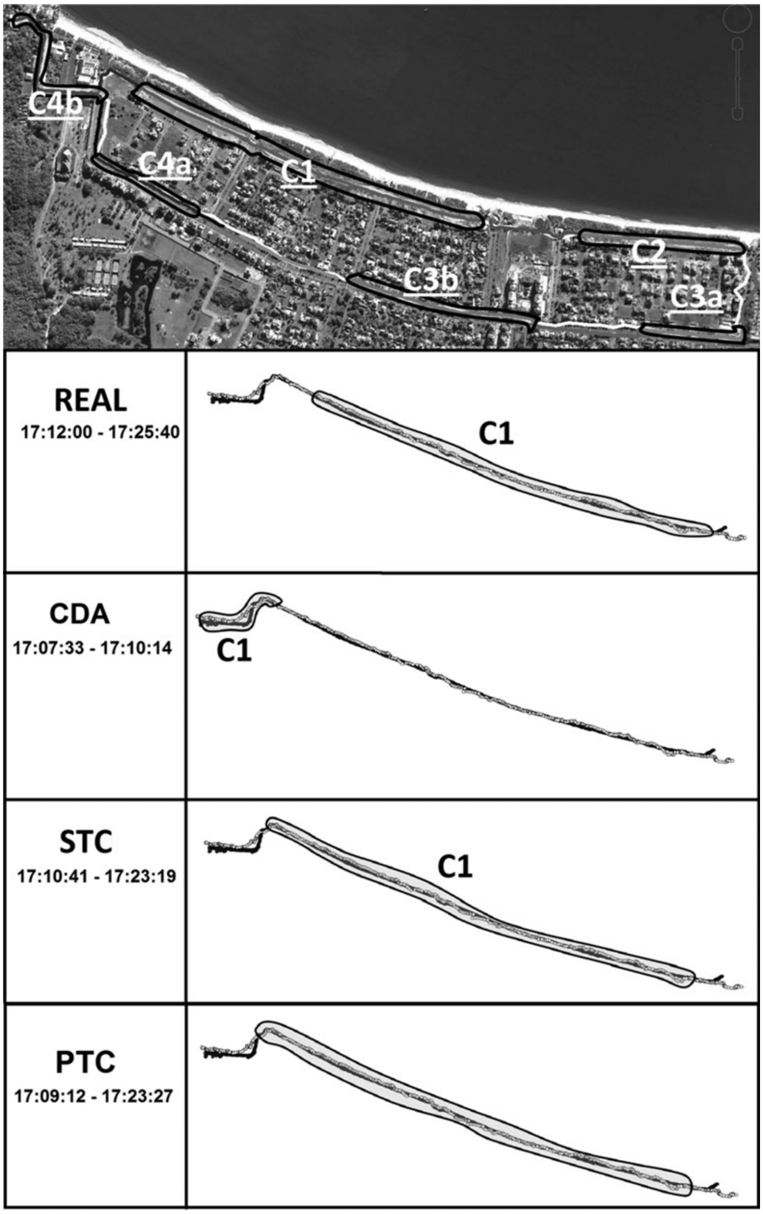


Figure 9 Comparison between the real chasing, the CDA and STC pattern for chasing C1

In case of using the co-location or flock methods to discover chasing patterns, a solution could be to increase the distance, but then we generate another problem: a large distance may lose the ~~semantic~~ of chasing, since two objects very far from each other do not characterize a chasing.

We evaluate the parameters considering the third dataset, where we know where the chasing patterns occur. The average distance between trajectories is between 15 and 40

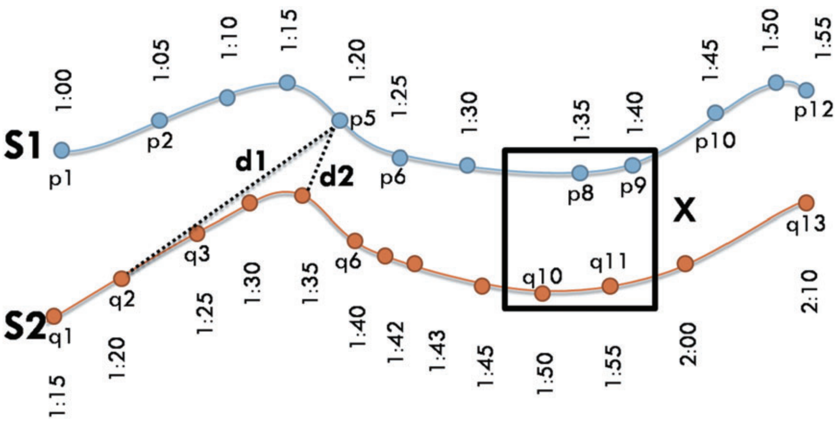


Figure 10 ~~In region X we have points p8,p9 of S1 and q10,q11 of S2. So we need a time tolerance Δt = 0:15 and distance d1 analyzed by Wchowicz et al., Hwang et al. 2005 and Cao et al. 2005 for point p5 and distance d2 analyzed by our algorithm for point p4~~

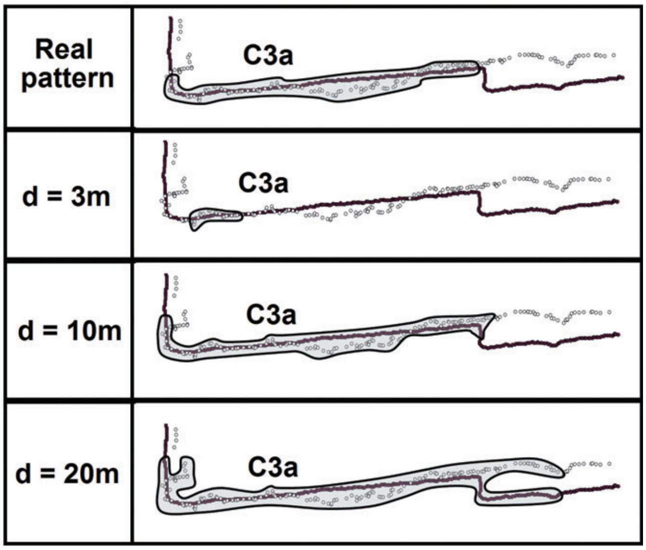


Figure 11 Results for chasing C3a with $\Delta d = 3$ m, $\Delta d = 10$ m and $\Delta d = 20$ m

meters at the same timestamp, while the average minor distance between trajectories with different timestamps is between 3 and 15 meters, with a difference of timestamp varying between 15 and 45 seconds.

As a first test we considered 30 seconds as the time tolerance, since it is the average difference of the timestamp between two points close in space (15 to 45). We considered as distance 3 and 10 meters, which is a coherent distance (between 3 and 15), and also considered 20 meters (between 15 and 45). The result of this test is show in Figure 11.

As may be seen in Figure 11, a very short distance (Δd = 3 meters) generates short patterns, since in a chasing pattern two trajectories may remain not so close. The same

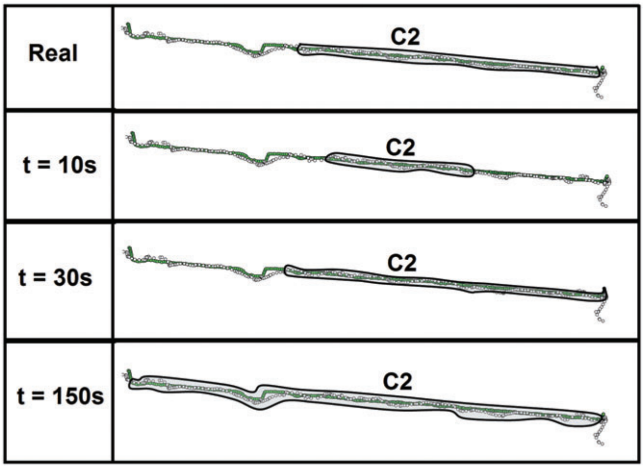


Figure 12 Results for chasing C2 with $\Delta t = 10$ s, $\Delta t = 30$ s and $\Delta t = 150$ s

occurs when considering a very long distance (Δd = 20 meters), that may also uncharacterize a chasing, because the sub-trajectories are to far from each other. Therefore, the best value should be Δd = 10 meters, which is not so close and not so far, generating the best results closer to the real pattern.

Considering the distance as 10 meters, we evaluate the time tolerance as 10, 30 and 150. The results are shown in Figure 12. With a low time tolerance (Δt = 10 seconds) the algorithm finds the chasing in the correct region, but with a size much smaller than the real pattern. With a very high time tolerance (Δt = 150 seconds) the algorithm finds the chasing in basically the whole trajectory. We test this very high value (150 seconds = 2.3 minutes) to show that a high time tolerance also discharacterizes a real chasing. A time tolerance (Δt = 30 seconds) as the interval of the trajectories between the closest points of the two trajectories resulted in a pattern close to the real chasing.

In summary, the distance parameter should not be higher than a distance that is impossible in a chasing. The time tolerance should be at least the time difference between the point collection interval, but the best value is the average time difference between two trajectories at the same place.

7 Conclusions and Future Work

The price reduction of mobile devices is increasing the generation of massive spatio-temporal datasets. These data, called trajectories of moving objects, provide characteristics of space and time, therefore it is possible to analyze where something happened and when it happened. Trajectory data can be interesting in several application domains, for instance weather conditions, urban traffic, natural disasters, migration of birds and human mobility. For these applications, trajectory data can express different behaviors through space and time.

The identification of different types of behavior in the trajectory domain can help the user of an application to understand why something happened or what was the cause of some actions. Although there are several types of trajectory patterns already identified in the literature, no works have focused on *chasing* patterns.

In this article we presented formal definitions to identify chasing patterns and an algorithm to find chasing behavior in moving object trajectories. The algorithm considers both space and time, where time is considered with different semantics in relation to other works. We evaluated the proposed approach with three different datasets, showing that our method finds patterns which are not discovered by other approaches.

It is important to emphasize that, as far as we know, there is no algorithm in the literature to find chasing patterns. We compare our work to some algorithms to show that they do not find chasing behavior.

The automatic discovery of chasing behavior among trajectories can be interesting in security applications, helping to identify the real behavior ~~of suspects. . .~~.

As future work we will be defining different types of chasing patterns and using semantic information to increase the confiability of the discovered patterns. For instance, we are planning to use domain knowledge such as road networks to differentiate intentional chasing from coincidental chasing as regular traffic in a highway.

8 Acknowledge

The authors would like to thank both CNPQ and FAPESC for supporting this research.

References

Baglioni M, Fernandes de Macédo J A, Renso C, Trasarti R, and Wachowicz M 2009 Towards semantic interpretation of movement behavior. In Sester M, Bernard L, and Paelke V (eds) *Proceedings of the 2009 AGILE Conference*. Berlin, Springer Lecture Notes in Geoinformation and Cartography: 271–88

Cao H, Mamoulis N, and Cheung D W 2005 Mining frequent spatiotemporal sequential patterns. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM '05)*, Houston, Texas: 82–9

Cao H, Mamoulis N, and Cheung D W 2006 Discovery of collocation episodes in spatiotemporal data. In *Proceedings of the Sixth International Conference on Data Mining (ICDM '06)*, Hong Kong: 823–27

Dodge S, Weibel R, and Lautenschutz A-K 2008 Towards a taxonomy of movement patterns. *Information Visualization* 7: 240–52

Giannotti F, Nanni M, Pinelli P, and Pedreschi D 2007 Trajectory pattern mining. In Berkhin P, Caruana R, and Wu X (eds) *KDD '07: Proceedings of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, ACM Press: 330–39

Hornsby K S and Cole S J 2007 Modeling moving geospatial objects from an event-based perspective. *Transactions in GIS* 11: 555–73

Hornsby K S and King K 2008 Modeling motion relations for moving objects on road networks. *GeoInformatica* 12: 477–95

Hwang S-Y, Liu Y-H, Chiu J-K, and Lim E-P 2005 Mining mobile group patterns: A trajectory-based approach. In *Proceedings of the Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, Hanoi, Vietnam: 713–18

Kang C-H, Hwang J-R, and Li K-J 2006 Trajectory analysis for soccer players. In *Proceedings of the Sixth IEEE International Conference on Data Mining (*ICDM '06*) Workshops*, Hong Kong: 377–81

~~Knowledge Discovery and Data Mining Research Unit 2010 Moving flock finder: Istituto di scienze e tecnologie dell'informazione, cnr dipartimento di informatica, universita di pisa. WWW document, http://www-kdd.isti.cnr.it/moving-flock~~

Laube P, Imfeld S, and Weibel R 2005 Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science* 19: 639–68

Lee J-G, Han J, Li X, and Gonzalez H 2008 raClass: Trajectory classification using hierarchical region-based and trajectory-based clustering. *PVLDB* 1: 1081–94

Legendre F, Borrel V, de Amorim M D, and Fdida S 2006 Modeling mobility with behavioral rules: The case of incident and emergency situations. In Cho K and Jacquet P (eds) *Proceedings of the 2006 Asian Internet Engineering Conference (AINTEC 2006)*. Berlin, Springer Lecture Notes in Computer Science Vol. 4311: 186–205

Loy A, Bogorny V, Renso C, and Alvares L O 2011 An algorithm to identify avoidance behavior in moving objects trajectories. *Journal of the Brazilian Computer Society* 17: in press

Palma A T, Bogorny V, and Alvares L O 2008 A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the Twenty-third Annual ACM Symposium on Applied Computing (ACMSAC '08)*, Fortaleza, Brazil: 863–68

Reynolds C W 1999 Steering behaviors for autonomous characters. In *Proceedings of the 1999 Game Developers Conference*, San Jose, California

Rocha J A M R, Times V C, Oliveira G, Alvares L O, and Bogorny V 2010 Db-smot: A direction-based spatio-temporal clustering method. In *Proceedings of the Fifth IEEE International Conference on Intelligent Systems*, London, United Kingdom: 114–19

Spaccapietra S, Parent C, Damiani M L, de Macedo J A, Porto F, and Vangenot C 2008 A conceptual view on trajectories. *Data and Knowledge Engineering* 65: 126–46 9

Waag Society 2005 Frequency 1550. WWW document, http://www.waag.org/project/frequentie

Wachowicz M, Ong R, Renso C, and Nannic M 2011 Finding moving flock patterns among pedestrian through collective coherence. *International Journal of Geographical Information Science* (Submitted, 2011)

Toppan Best-set Premedia Limited	
Journal Code: TGIS	Proofreader: Elsie
Article No: 1285	Delivery date: 15 August 2011
Page Extent: 22	

AUTHOR QUERY FORM

Dear Author,

During the preparation of your manuscript for publication, the questions listed below have arisen. Please attend to these matters and return this form with your proof.

Many thanks for your assistance.

Query References	Query	Remark
1	AUTHOR: Please provide the article type.	
2	AUTHOR: To match the reference list, should Gianotti et al. 2007 be changed to Giannotti et al. 2007? Please advise.	
3	AUTHOR: Cao et al. 2009 has not been included in the Reference List, please supply full publication details.	
4	AUTHOR: In order to match with rest of reference's citation, please provide the author name for the citation of [19] [1] and [4].	
5	AUTHOR: Manso is not mentioned in footnote 17 or in any other footnote. Author needs to add correct reference.	
6	AUTHOR: Laube, 2005 has not been included in the Reference List, please supply full publication details.	
7	AUTHOR: Cao, 2005 has not been included in the Reference List, please supply full publication details.	
8	AUTHOR: Cao et al. has not been included in the Reference List, please supply full publication details.	
9	AUTHOR: Spaccapietra, Parent, Damiani, de Macedo, Porto, Vangenot, 2008 has not been cited in the text. Please indicate where it should be cited; or delete from the Reference List.	

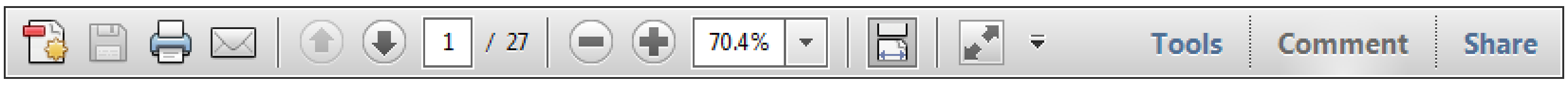
10	AUTHOR: To match the reference list, should Wchowicz et al. be changed to Wachowicz et al., 2011? Please advise.	

WILEY-BLACKWELL

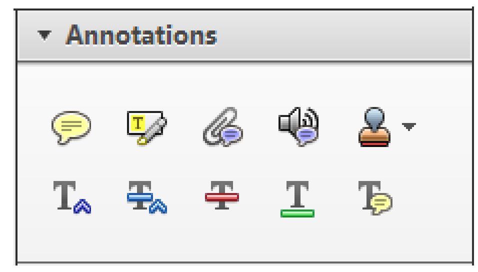
USING e-ANNOTATION TOOLS FOR ELECTRONIC PROOF CORRECTION

Required software to e-Annotate PDFs: Adobe Acrobat Professional or Adobe Reader (version 8.0 or above). (Note that this document uses screenshots from Adobe Reader X)
The latest version of Acrobat Reader can be downloaded for free at: http://get.adobe.com/reader/

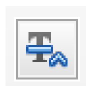
Once you have Acrobat Reader open on your computer, click on the Comment tab at the right of the toolbar:



This will open up a panel down the right side of the document. The majority of tools you will use for annotating your proof will be in the Annotations section, pictured opposite. We've picked out some of these tools below:



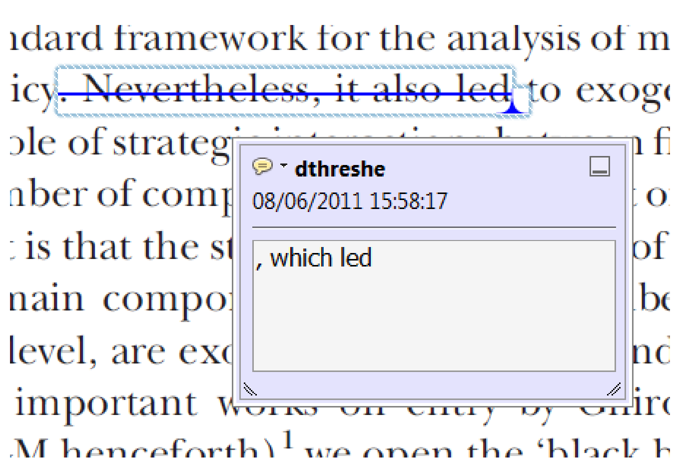
1. Replace (Ins) Tool – for replacing text.



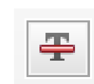
Strikes a line through text and opens up a text box where replacement text can be entered.

How to use it

- Highlight a word or sentence.
- Click on the Replace (Ins) icon in the Annotations section.
- Type the replacement text into the blue box that appears.



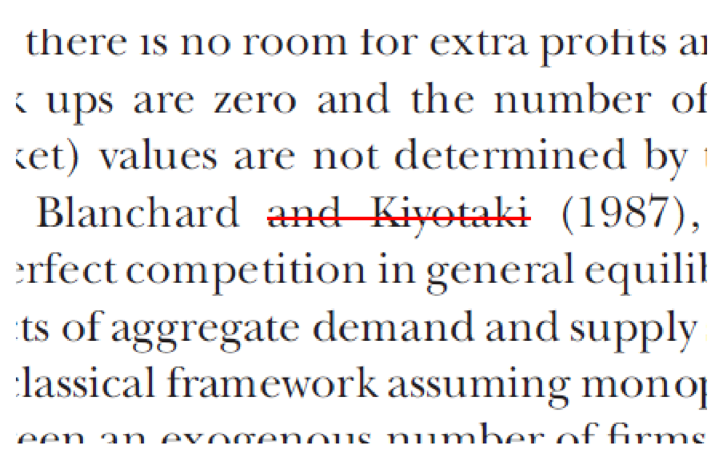
2. Strikethrough (Del) Tool – for deleting text.



Strikes a red line through text that is to be deleted.

How to use it

- Highlight a word or sentence.
- Click on the Strikethrough (Del) icon in the Annotations section.

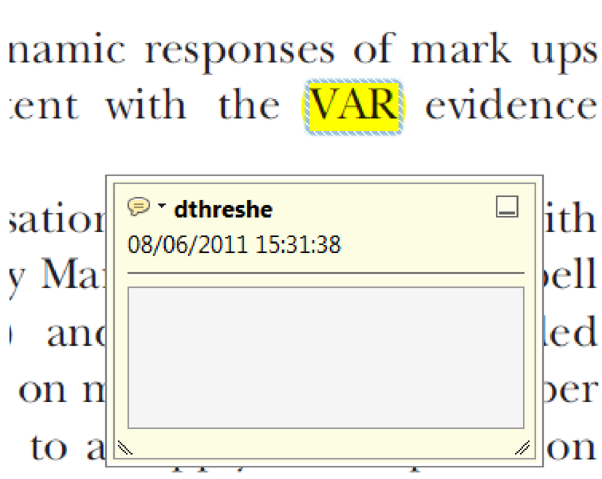


3. Add note to text Tool – for highlighting a section to be changed to bold or italic.

Highlights text in yellow and opens up a text box where comments can be entered.

How to use it

- Highlight the relevant section of text.
- Click on the Add note to text icon in the Annotations section.
- Type instruction on what should be changed regarding the text into the yellow box that appears.

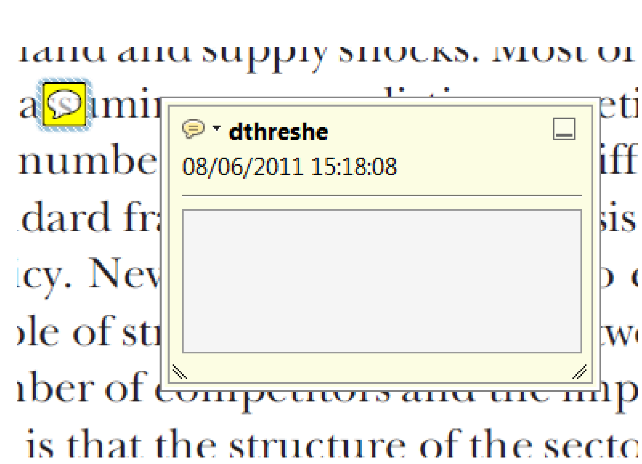


4. Add sticky note Tool – for making notes at specific points in the text.

Marks a point in the proof where a comment needs to be highlighted.

How to use it

- Click on the Add sticky note icon in the Annotations section.
- Click at the point in the proof where the comment should be inserted.
- Type the comment into the yellow box that appears.



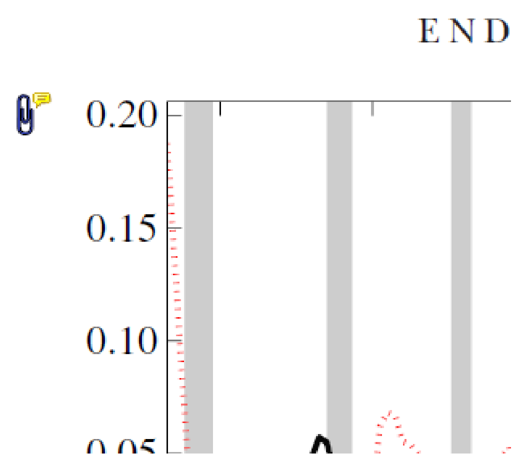
USING e-ANNOTATION TOOLS FOR ELECTRONIC PROOF CORRECTION

5. Attach File Tool – for inserting large amounts of text or replacement figures.

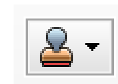
Inserts an icon linking to the attached file in the appropriate pace in the text.

How to use it

- Click on the Attach File icon in the Annotations section.
- Click on the proof to where you'd like the attached file to be linked.
- Select the file to be attached from your computer or network.
- Select the colour and type of icon that will appear in the proof. Click OK.



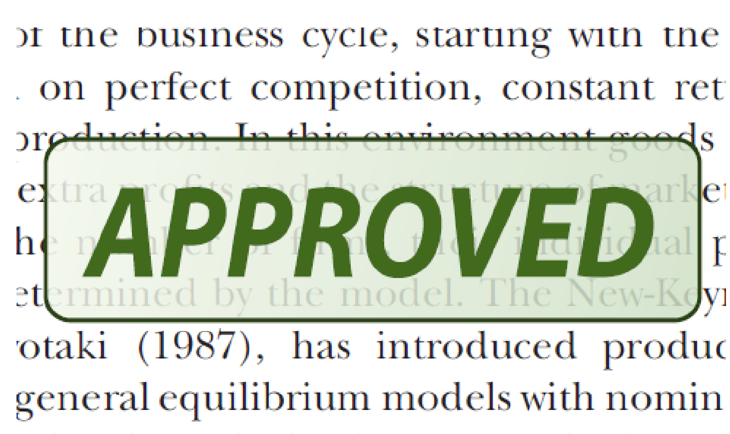
6. Add stamp Tool – for approving a proof if no corrections are required.

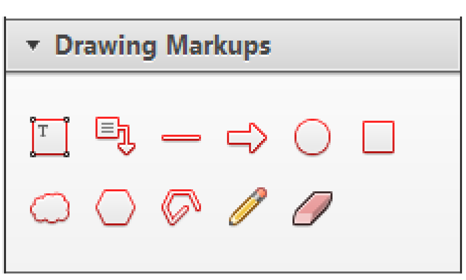


Inserts a selected stamp onto an appropriate place in the proof.

How to use it

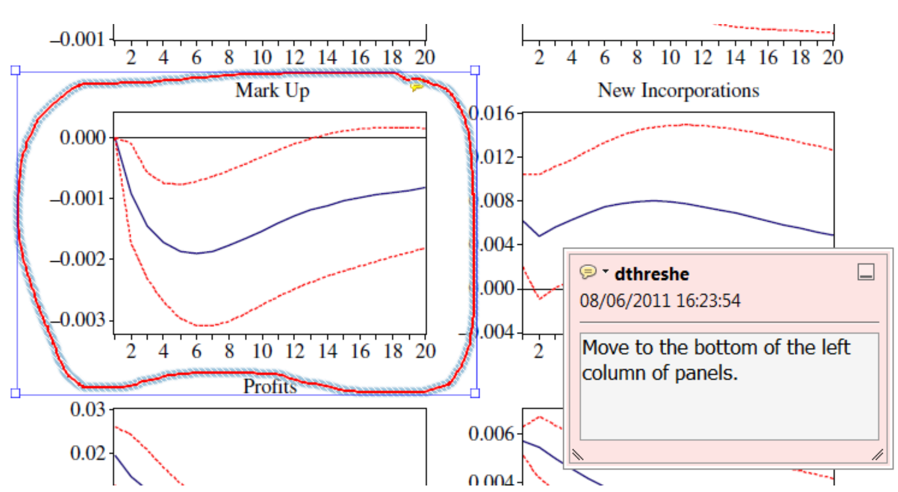
- Click on the Add stamp icon in the Annotations section.
- Select the stamp you want to use. (The Approved stamp is usually available directly in the menu that appears).
- Click on the proof where you'd like the stamp to appear. (Where a proof is to be approved as it is, this would normally be on the first page).





7. Drawing Markups Tools – for drawing shapes, lines and freeform annotations on proofs and commenting on these marks.

Allows shapes, lines and freeform annotations to be drawn on proofs and for comment to be made on these marks..



How to use it

- Click on one of the shapes in the Drawing Markups section.
- Click on the proof at the relevant point and draw the selected shape with the cursor.
- To add a comment to the drawn shape, move the cursor over the shape until an arrowhead appears.
- Double click on the shape and type any text in the red box that appears.

For further information on how to annotate proofs, click on the Help menu to reveal a list of further options: