Detecting Avoidance Behaviors Between Moving Object Trajectories

Francesco Lettich[a], Luis Otavio Alvares[b], Vania Bogorny[b,*], Salvatore Orlando[a], Alessandra Raffaetà[a], Claudio Silvestri[a]

[a] *Dipartimento di Scienze Ambientali, Informatica e Statistica, Università Ca' Foscari, Venezia, Italy*

[b] *Departamento de Informática e Estatística, Centro Tecnológico Universidade Federal de Santa Catarina, Florianopolis, Brazil*

Abstract

Several algorithms have been proposed in the last few years for mining different mobility patterns from trajectories, such as flocks, chasing, meeting, and convergence. An interesting behavior that has not been much explored in trajectory pattern mining is *avoidance*. In this paper we define the avoidance behavior between moving object trajectories, providing a set of theoretical definitions to precisely describe various kinds of avoidance, and propose an effective algorithm for detecting avoidances. The proposed method is quantitatively evaluated on a real-world dataset, and correctly detects with high precision the quasi totality of the trajectory pairs that exhibit avoidance behaviors (F-measure up to 95%).

The final authenticated version is available online at http://dx.doi.org/10.1016/j.datak.2015.12.003

1. Introduction

Current advances in mobile technology such as GPS and smartphones have increased the interest in mobility data analysis in several application domains such as security, smart cities, transportation systems, urban planning, and biological studies. As a consequence, several algorithms have been proposed for discovering various types of behaviors in trajectory data such as T-patterns [1], flocks [2, 3], meet [4], periodic movements [5, 6], anomalous traffic patterns [7] and chasing [8]. In [9] a taxonomy with different types of trajectory behaviors is proposed, while a summary of the most well known trajectory behaviors (also called patterns) is presented in [10].

---

*Corresponding author

Email addresses: `lettich@dais.unive.it` (Francesco Lettich), `luis.alvares@ufsc.br` (Luis Otavio Alvares), `vania.bogorny@ufsc.br` (Vania Bogorny), `orlando@unive.it` (Salvatore Orlando), `raffaeta@unive.it` (Alessandra Raffaetà), `silvestri@unive.it` (Claudio Silvestri)

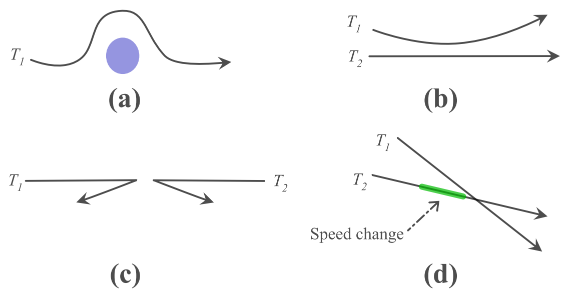


Figure 1 Different kinds of avoidance behaviors: avoidance with respect to a static object (a), and avoidance between moving objects: *individual* (b), *mutual* (c), and *individual* induced by a change in speed (d).

In this paper we focus on *avoidance detection* for trajectories, whose goal is to detect situations in which a moving object avoids a static object (area), as shown in Figure 1(a), or a moving object, as shown in Figure 1(b, c, and d). Avoidance detection can be interesting for discovering suspicious behaviors such as objects avoiding static objects, like surveillance cameras, police patrols, or speed controllers, or moving entities like criminals or terrorists that avoid policemen. In marine surveillance, ships with illicit products or illegal immigrants may avoid Coast Guard boats. In computer games, pinpointing avoidance behaviors can be useful to detect, for instance, the avoided enemies, while in soccer games it may be useful to analyze players avoiding markers. In zoological studies, avoidance detection may reveal how preys avoid predators (e.g., at which distance, by changing direction or changing speed).

The problem of avoidance detection has not received much attention in the literature. In [11] the authors introduce notions of attraction and avoidance degree between trajectories. Roughly, the attraction degree between two trajectories is higher when the corresponding objects meet (get close) more frequently than they would by moving randomly along the same points. Avoidance degree is defined as the dual relation. This approach thus takes a global view on the trajectories, highlighting the frequent behaviors and hiding the occasional episodes. For example, consider a boat that generally passes close to a Coast Guard boat and changes just once its route for a specific part of its trip to avoid the encounter. The approach in [11] would reveal a relevant degree of attraction between the trajectories of the boat and the Coast Guard due to their most frequent behavior, despite the occasional avoidance of the Coast Guard, which is essentially ignored.

The technique we propose in this paper, instead, is aimed at detecting all specific instances of avoidance between trajectories, including occasional episodes as those described above. A first treatment of this problem in [12] is limited to the simpler case of avoidance of static objects. Here we aim at treating avoidance in the general case in which both objects are moving. The discovery of this type of avoidance is clearly challenging. Some natural new questions that arise are: what are the main features that characterize an avoidance between two trajectories? Who is avoiding who? At what distance two objects initiate an avoidance?

As a first step we introduce a notion of *avoidance behavior*. An avoidance between two trajectories occurs when both objects are moving towards the same area at the same time, but either one or both change their behavior when they come close enough to be aware of each other. Avoidance is thus characterized in terms of a change of behavior which prevents the two objects to meet, i.e., a discordance between a forecast of the movement of the two objects and their actual behavior. Figure 1 shows three examples of avoidance behavior of interest: Figure 1(b) illustrates an example of trajectory avoidance where T_1 avoids trajectory T_2 by changing its direction. In Figure 1(c) both T_1 and T_2 avoid each other by changing their direction whereas in Figure 1(d), although the trajectories have a spatial intersection relationship, T_2 avoids T_1 by slowing down the speed in order to not spatio-temporally intersect T_1.

We also introduce a classification of avoidance based on the evidence of changes in the behavior of the trajectories. Avoidance is called *mutual* when both trajectories alter significantly their movement in order to cause a missed meet, as shown in Figure 1(c). On the other hand, we classify an avoidance as *individual* when only one trajectory presents a relevant change while the other one behaves as expected. Figures 1(b) and (d) provide examples of this kind of avoidance. Finally, in some cases an avoidance is determined by minor changes in the movements of one or both trajectories: this is referred as a *weak* avoidance.

In order to detect avoidances we propose an algorithm that returns, for each pair of trajectories, all the occurrences of avoidance labeled by the corresponding type. The algorithm requires some parameters, like the spatial threshold for considering two trajectories in a *meet* relationship and the temporal look-ahead used for forecasting the future behavior of trajectories, and is able to process real-world trajectories collected at different sampling rates. The algorithm is also used to define two different kinds of detectors: the *single* detector consists of a single run of the algorithm with fixed input parameters, while the *fused* detector considers multiple runs of the algorithm with different parameters, lastly fusing the results in a unique output to improve the quality of results.

The proposed methods are evaluated under different points of view. First, we quantitatively test the *effectiveness* of the methods by analyzing the ability to correctly detect expected avoidance behaviors. To this end we use an ad-hoc annotated dataset, our ground truth, created for the purposes of this work. Second, we assess the ability of the algorithm to highlight interesting and previously unknown patterns emerging from avoidance behaviors when using datasets for which no prior knowledge related to avoidance behaviors is available.

In summary, we make the following contributions in this paper: (i) we propose a framework which defines the avoidance between pairs of trajectories considering changes of behavior and a criteria to classify any avoidance as *weak*, *mutual* or *individual*; (ii) we present an algorithm which is able to automatically detect every avoidance between two trajectories; (iii) we define a detector for analyzing the avoidance with different sets of parameters; finally, (iv) we show the effectiveness of our methods when considering real-world datasets.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 introduces some basic definitions. Section 4 illustrates the new definitions for avoidance detection while Section 5 proposes an algorithm to detect avoidance behaviors. Section 6 describes experiments on real-world trajectories, while Section 7 concludes the paper and suggests directions of future research.

2. Related Work

The goal of *avoidance detection* is to determine those situations in which objects avoid spatial regions or other moving objects. To the best of our knowledge, there are basically two works on this topic. The first [12] concentrates on moving objects that avoid static objects, and does not treat avoidances between trajectories of moving objects. The latter [11] introduces the notions of attraction and avoidance relationship between trajectories, but this work has different focus and objectives with respect to our approach. The goal of the approach in [11], called ApproxCount, is to check whether there is a statistical evidence that an object A is avoiding/attracting an object B. To this aim the authors define attraction and avoidance significance values by performing a permutation test over the known trajectory observations. They evaluate the *expected* number of meeting events of trajectory A with a random traversal of the points belonging to trajectory B. Actual meeting frequency then is compared to expected meeting frequency to compute an attraction/avoidance score in the range $[0, 1]$.

We note that this method focuses on the global behavior of a pair of trajectories, without identifying where avoidance episodes actually occur. On the other hand, our work aims to determine each single occurrence of avoidance behavior between moving objects. This difference of objectives has also consequences at technical level. In [11] the authors only consider the object observations and how close they are, without considering whether objects change their predicted trajectories to avoid meeting other objects. Additionally, they suppose that trajectory data are manipulated, so that tracking times of all the trajectories are synchronized and each trajectory has the same number of points. Despite such differences, we perform some experiments to compare the avoidance detection capabilities between ApproxCount and our approach; results are reported in Section 6.2.

A topic which has some relations with avoidance detection is *collision avoidance*, whose goal is to determine the actions needed to avoid collision between objects. Collision avoidance deals with models, systems, and practices designed to prevent vehicles such as cars, ships, and airplanes from colliding with other vehicles. Consequently, the focus is on detecting a future collision and changing (or suggesting a change of) the current route for one or more of the involved vehicles to avoid a collision. All these operations must be carried on in *real-time*.

There is a vast literature on collision avoidance for various kinds of vehicles, such as cars [13, 14, 15, 16], ships [17, 18], and aircrafts [19, 20, 21].

In [13] the proposal is to minimize the safety distance error and to regulate the relative speed between two cars, so to avoid rear-end collision, using hierarchical longitudinal control. In [14] the authors propose a real-time method for computing a car trajectory towards a safe final state, as soon as an endangering obstacle is detected by a sensor (e.g. radar or lidar). Two scenarios are considered: a car which is overtaking another (slower) car in the same lane and an overtaking car which faces another car coming from the opposite direction. The solution is obtained from a simplified car model based on two control variables (steering velocity and braking force), state variables (speed, yaw angle, yaw angle rate, the center of gravity and direction) and a state dynamics defined by a system of differential equations. In [15] experimental results for an active control intersection collision avoidance system are presented. The system is implemented on modified Lexus test vehicles and it utilizes vehicle-to-vehicle dedicated short-range communications to share safety critical state information. Safety is achieved in potential collision scenarios by controlling the speeds of both vehicles with automatic brake and throttle commands. Another approach for car collision avoidance considers pedestrians [16], where the use of stereo cameras on board of vehicles supports the detection of pedestrians with the aim of avoiding them. In the domain of ships, [17] proposes a fuzzy-neural inference network that learns a set of examples from a set of rules defined by the International Regulations for Preventing Collisions at Sea. Based on the learned examples, the method suggests direction changes of the ship to avoid a possible collision. The main input data are the ships' direction and speed, the distance between them, and the type of water area (sea, coast, limit water). The model only considers cases where an encounter situation is already detected. The output is the set of actions to avoid the collision. In the aircraft domain, [20] considers a set of aircrafts where each one has a known destination and the related trajectory is represented as a straight line going from its current location to the destination or to the next waypoint. The speed of the aircrafts is known and constant, and it is assumed that aircrafts fly in layers. The contribution of the paper is a linear model to modify aircraft routes in order to avoid a collision when two or more aircrafts become sufficiently close to each other. The model considers real dynamics constraints to be more realistic.

Another domain in which collision avoidance is well studied is robotics. In this domain, when a robot is planning its route (its possible trajectory) it should consider known obstacles and avoid them. Some representative works are [22, 23, 24, 25]. For instance, in [24] a dynamical system-based approach is used to deviate the robot from the obstacles. In [25] a behavior-based multi-robot collision avoidance system is proposed to efficiently coordinate the simultaneous navigation of large robot teams.

It is worth noticing that collision avoidance and avoidance detection are remarkably different. Indeed, while in collision avoidance the main aim is to change the route of a moving object considering some of its physical properties (e.g. mass, center of gravity, steering angle), in avoidance detection the goal is to detect situations in which objects alter their movements in presence of other objects.

3. Preliminaries

Moving objects are entities having a time variant position, uniquely determined at each time instant. A trajectory is a continuous part of the movement of an object [26]. For the sake of simplicity, in the following we will restrict to the 2D Euclidean space, but note that the generalization to higher dimensional spaces is straightforward. For the reader's convenience, the main symbols used throughout the paper are reported in Table A.3, Appendix A.

Definition 3.1 (Movement and trajectory). *The* movement *of an object o is a continuous function $M_o : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^2$ from the real positive numbers, representing time instants, to 2D space. Given an object o and a time interval* [tBegin,tEnd]*, a trajectory T is the restriction of the movement M_o of the object to the given time interval. The spatio-temporal position of the object at* tBegin *(resp.* tEnd*) is called the* Begin *(resp.* End*) of the trajectory.*

Commonly, in mobility applications, the continuous movement of an object is not completely known. In fact, trajectories are often given by means of a finite set of time-stamped positions, called trajectory points or samples.

Definition 3.2 (Trajectory point). *A trajectory point, or trajectory sample, of a trajectory T is a tuple (x, y, t), where $T(t) = (x, y)$ is the object position at time t (called the* timestamp *of the trajectory point).*

The set of known positions of a moving object during the definition interval of a trajectory is named *trajectory track*, or trajectory *sampling*.

Definition 3.3 (Trajectory track). *Given a temporally ordered sequence $\langle t_1, \ldots, t_n \rangle$ of timestamps, the track of a trajectory T for the given timestamps is the temporally ordered sequence of trajectory points $\langle p_1, \ldots, p_n \rangle$, where $p_i = (x_i, y_i, t_i)$ and $(x_i, y_i) = T(t_i)$.*

In many situations an (approximate) reconstruction of each trajectory from its track is needed. To accomplish this task, different interpolation functions can be used which describe, in an analytic and continuous way, the movement of the object between pairs of consecutive trajectory points.

4. Avoidance

An *avoidance* between two moving objects occurs when both are moving towards the same area at the same time, but either one or both change their behavior when they come close enough to be aware of each other. In the following, with a slight abuse of terms, we will indistinctly use the terms trajectory and object when referring to the avoidance.

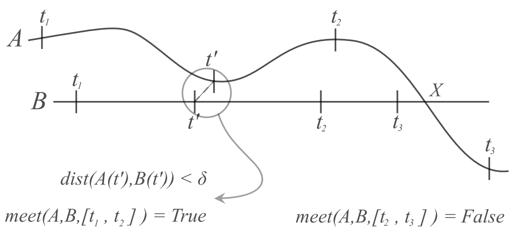

A
B
t_1
t_1
t'
t'
t_2
t_2
t_3
t_3
X
dist(A(t'),B(t')) < δ
meet(A,B,[t_1 , t_2]) = True
meet(A,B,[t_2 , t_3]) = False

Figure 2 *A meets B* (the distance in t' is less than δ) during $[t_1, t_2]$ but not during $[t_2, t_3]$.

In order to define the avoidance concept, we first introduce the predicates *meet* and *will-meet*. The first one expresses the fact that in a certain interval two trajectories become sufficiently close to be considered in contact, whereas the second one states that the *forecast* of these trajectories, determined by some technique on the basis of some observed behavior, will lead to a contact.

Definition 4.1 (Meet). *Given two trajectories T_a and T_b, a time interval $[t_1, t_2]$ and a distance threshold δ, we define the predicate* meet_δ *as*

$$\mathrm{meet}_\delta(T_a, T_b, [t_1, t_2]) \equiv \exists t \in [t_1, t_2].\, dist(T_a(t), T_b(t)) < \delta$$

where $dist(p_a, p_b)$ is the Euclidean distance of the points p_a and p_b.

When the predicate is satisfied we also say that T_a *meets* T_b during $[t_1, t_2]$ with threshold δ. In case the threshold δ is evident from the context it will be omitted in the predicate notation, writing *meet* instead of *meet*$_\delta$.

Figure 2 illustrates a pair of trajectories, A and B, during the time interval $[t_1, t_3]$. The predicate *meet* is true for $[t_1, t_2]$, since at time t' the distance of the two trajectories is less than δ. Note that for the interval $[t_2, t_3]$, even if the two trajectories have a spatial intersection X the *meet* predicate is false since the distance between the two moving objects at any instant $t \in [t_2, t_3]$ is always greater than δ. In fact, A and B cross X at different time instants.

Having a way of predicting the movement of the trajectories on the basis of what happened in the past, we can establish if two trajectories will meet each other or not. We next introduce an abstract notion of *movement predictor* clarifying which are the expected properties.

Definition 4.2 (Movement predictor). *Given the temporal domain $\mathbb{R}_{\geq 0}$ and a movement domain $\mathcal{M} = \{M \mid M : \mathbb{R}_{\geq 0} \to \mathbb{R}^2\}$, consisting of all possible movement functions, a* movement predictor *is a functional forecast* : $\mathcal{M} \times (\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}) \to \mathcal{M}$ *such that for all $M, M' \in \mathcal{M}$*

$$M \mid_{[0,t1]} = M' \mid_{[0,t1]} \implies \textit{forecast}(M, [t1, t2]) = \textit{forecast}(M', [t1, t2])$$

The functional *forecast* maps a movement function M to its forecast movement into the interval $[t_1, t_2]$ by exploiting only the behavior of M in the past interval

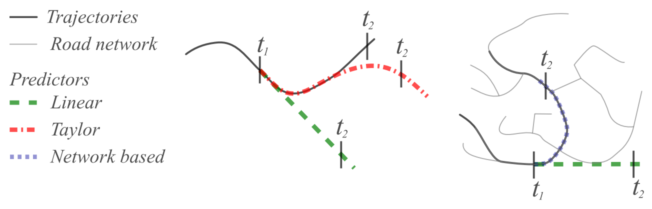


Figure 3 Different kinds of predictors based on several interpolations and on movement constraints (road network).

$[0, t_1]$. It can be defined in different ways depending on the contexts. Commonly, it is based on the assumption that the behavior does not change with respect to the recent past of the trajectory. For instance, in case of objects that move freely in space, we can use the Taylor series in order to estimate the next positions of the trajectory, and the more terms of these series we compute, the more precise we obtain the approximation, e.g., the first term preserves the direction, the second one the curvature. On the other hand, if the object movement is constrained by a network, we can exploit the network to predict where the object is going.

Figure 3 shows several different forecast functionals for the interval $[t_1, t_2]$. The solid line represents the actual trajectory. The dashed, green line models a linear prediction, preserving the direction at time t_1; the dash-dot, red line shows a forecast based on a higher order Taylor series, preserving several derivatives of the trajectory; and the dotted, blue line in the right figure illustrates a prediction based on the knowledge of the road network.

The linear predictor is based on the assumption that objects move, in general, with constant speed and direction. Thus a turn is considered as a deviation from the expected movement. In Figure 3 on the left, for example, according to the prediction based on movement at time t_1 the object is expected to continue along the dashed straight line and to be at some position at time t_2. The predictor based on higher order Taylor series assumes that the object moves smoothly, by preserving the rate of turn (second order), the rate at which the rate of turn changes (third order) or higher invariants. Those are captured by Taylor series based on the derivatives (up to second, third, or higher order) of the trajectory function at time t_1. In this case the forecast might be not straight. In Figure 3, on the left, the dash-dot red line shows a prediction based on the fact that the object is involved in a sequence of curves. Thus the object acts unexpectedly only when it stops curving. Finally, the road based predictor assumes that the object follows a road network at constant speed, maintaining its direction at crossings (staying on the current road) or taking the highest flow road when it is not possible to continue in the current direction (dotted, blue line in Figure 3 on the right).

In the following examples, and in the experimental section, we will use a linear movement predictor and we will employ the same notation as in Figure 3: solid lines for actual trajectories and dashed lines for the linear forecast. We re-

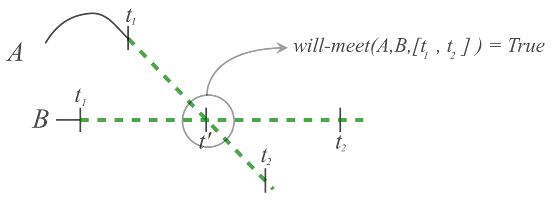


Figure 4 According to the forecasts, A *will meet* B (the distance will be less than δ) at some time t' in the time interval $[t_1, t_2]$.

mark, however, that any predictor could be used without any significant change to the proposal.

Definition 4.3 (Will-meet). *Given two trajectories T_a and T_b, a time interval $[t_1, t_2]$, a movement predictor* forecast *and a threshold δ, we define the predicate* will-meet$_\delta$ *as*

$$\text{will-meet}_\delta(T_a, T_b, [t_1, t_2]) \equiv \\ \text{meet}_\delta(\text{forecast}(T_a, [t_1, t_2]), \text{forecast}(T_b, [t_1, t_2]), [t_1, t_2])$$

When the predicate is satisfied we also say that T_a is *expected to meet* T_b during $[t_1, t_2]$ with threshold δ. In case the threshold δ is evident from the context it will be omitted in the predicate notation, writing *will-meet* instead of *will-meet$_\delta$*.

Figure 4 illustrates an example where the predicate *will-meet* holds: A and B are supposed to meet at time t', provided that they maintain the same speed and direction they had at time t_1 (as already mentioned we use a linear movement predictor).

With these definitions we define an avoidance between trajectories as an event that happens in a time interval where the prediction is a *meet* between two trajectories but this meet does not occur.

Definition 4.4 (Avoid). *Given two trajectories T_a and T_b, a time interval $[t_1, t_2]$, and a threshold δ, we define the predicate* avoid$_\delta$ *as*

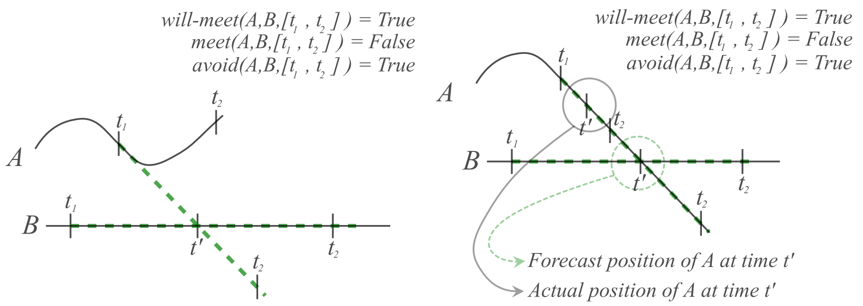
$$\text{avoid}_\delta(T_a, T_b, [t_1, t_2]) \equiv \text{will-meet}_\delta(T_a, T_b, [t_1, t_2]) \land \neg\text{meet}_\delta(T_a, T_b, [t_1, t_2])$$

When the predicate is satisfied we say that T_a and T_b *avoid* to meet, with threshold δ, during $[t_1, t_2]$. In case the threshold δ is evident from the context it will be omitted in the predicate notation, writing *avoid* instead of *avoid$_\delta$*.

The duration of the time interval $[t_1, t_2]$ is a measure of the future awareness of the moving objects (e.g., a person or an animal), that is the amount of time they are able to reliably forecast all of the involved trajectories to avoid collisions or encounters.

It is worth mentioning that the avoidance cases shown in Figure 1(b, c, and d), in which the avoidance is due to changes of speed and/or direction, are covered by Definition 4.4.

Figure 5 shows two different cases of avoidance in which trajectory B exactly fulfills the prediction (dashed green), whereas trajectory A behaves differently



will-meet(A,B,[t1 , t2]) = True
meet(A,B,[t1 , t2]) = False
avoid(A,B,[t1 , t2]) = True
A
B
t1
t2
t'
will-meet(A,B,[t1 , t2]) = True
meet(A,B,[t1 , t2]) = False
avoid(A,B,[t1 , t2]) = True
Forecast position of A at time t'
Actual position of A at time t'

Figure 5 A *avoids* B during the time interval $[t_1, t_2]$: meet is expected according to the forecast computed at time t_1 but no actual meet happened during the given time interval.

with respect to the forecast (dashed green): on the left, A changes direction after time t_1, and on the right it reduces its speed. As a result, the forecasts of the two trajectories (A and B) are expected to meet at time $t' \in [t_1, t_2]$, but the two trajectories do not actually meet since their distance is always larger than δ (omitted for simplicity in Figure 5) during $[t_1, t_2]$. Thus, in both cases the *avoid* predicate is true.

4.1. Avoidance Classification

The concept of avoidance is related to a change of behavior of one or both the involved objects such that a predicted meet does not occur. Thus, it is important to understand if the avoidance is caused by one of the objects, as in Figure 1(b) and Figure 1(d), or by both as in Figure 1(c). To this end, we need to formalize the concept of change of behavior in a way that covers any alteration of the expected movement with respect to the forecast.

We define a new predicate, *change-behavior*, to better characterize different kinds of avoidance. Informally, we regard the behavior of an object as *changed* during a time interval when there is a difference between the actual trajectory and its forecast that is sufficient to cause missed meets without any change in the other trajectory. Intuitively such a difference should have at least the same magnitude as the meet threshold δ.

Definition 4.5 (Change-behavior). *Given a trajectory T, a time interval $[t_1, t_2]$, and a meet distance threshold δ, we define the predicate* $\text{change-behavior}_\delta$

$$\text{change-behavior}_\delta(T, [t_1, t_2]) \equiv \exists t \in [t_1, t_2],\ \text{dist}(\text{forecast}(T, [t_1, t_2])(t), T(t)) > \delta$$

where $\text{forecast}(T, [t_1, t_2])(t)$ *and* $T(t)$ *are respectively the forecast and the actual position for trajectory T at time t.*

When the predicate is satisfied we say that T *changes* its behavior, with threshold δ, during $[t_1, t_2]$. In case the threshold δ is evident from the context it will be omitted in the predicate notation, writing *change-behavior* instead of *change-behavior*$_\delta$.

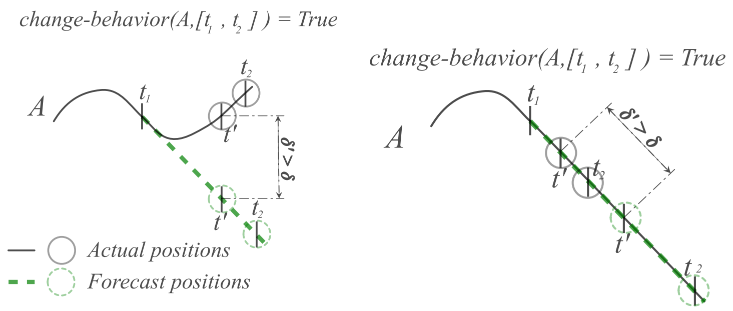

change-behavior(A,[t1 , t2]) = True
A
t1
t2
t'
δ' < δ
t'
t2
Actual positions
Forecast positions
change-behavior(A,[t1 , t2]) = True
A
t1
δ' > δ
t'
t2
t'
t2

Figure 6 Trajectory A *changes behavior*: at some time t' during $[t_1, t_2]$ the distance of the actual position from the forecast position is greater than δ.

Figure 6 focuses on trajectory A of Figure 5, showing how it changes its behavior. On the left it changes direction whereas on the right it reduces the speed with respect to the forecast (green dashed line). In both cases the distance at t' between the forecast and the actual trajectory is greater than δ, hence the predicate *change-behavior* holds.

Based on the above definition, we distinguish among *weak*, *individual* and *mutual* avoidance.

Definition 4.6 (Avoidance classification). *Given two trajectories T_a and T_b, and a time interval $[t_1, t_2]$ such that* $\text{avoid}_\delta(T_a, T_b, [t_1, t_2])$ *is true, we classify that avoidance in the following way:*

$$type\_avoid_\delta(T_a, T_b, [t_1, t_2]) = \begin{cases} mutual & \text{if } \text{change-behavior}_\delta(T_a, [t_1, t_2]) \wedge \\ & \quad \text{change-behavior}_\delta(T_b, [t_1, t_2]) \\ weak & \text{if } \neg\text{change-behavior}_\delta(T_a, [t_1, t_2]) \wedge \\ & \quad \neg\text{change-behavior}_\delta(T_b, [t_1, t_2]) \\ individual & \text{otherwise} \end{cases}$$

In other words, we have a *mutual avoidance* when there is an evident change of behavior for both trajectories, an *individual avoidance* when only one trajectory significantly changes its behavior, and a *weak avoidance* when there is an avoidance despite the fact that the behavior changes are minimal for both trajectories.

Thus, in Figure 1(b) there is an individual avoidance since T2 exactly follows the forecast, whereas T1 changes direction. In Figure 1(c), instead both T1 and T2 change direction, determining a mutual avoidance. Figure 7 shows, with several temporal and spatial details, two examples of *mutual* and *weak* avoidance, whereas avoidances in Figure 5 are both *individual*. We recall that trajectories are represented in solid black, forecasts in dashed green, and their specific positions at a given time are circled, respectively in solid gray and dashed green. In Figure 7 (left), A changes its behavior by reducing its speed and this is detected since there exists a time $t' \in [t_1, t_2]$ such that the distance of the

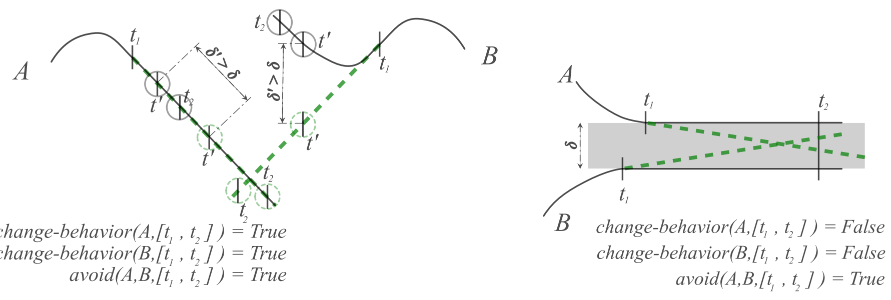

A
B
t_1
t_2
t'
$\delta' > \delta$
$\delta' > \delta$
change-behavior(A,[t_1 , t_2]) = True
change-behavior(B,[t_1 , t_2]) = True
avoid(A,B,[t_1 , t_2]) = True
A
B
t_1
t_2
δ
change-behavior(A,[t_1 , t_2]) = False
change-behavior(B,[t_1 , t_2]) = False
avoid(A,B,[t_1 , t_2]) = True

Figure 7 On the left, both A and B change behavior to avoid each other (*mutual* avoidance). On the right, according to the forecast A and B should meet but this does not happen even if both did not significantly change behavior (*weak* avoidance).

forecast position from the actual position is greater than δ. The same happens for B that changes its direction. Since both trajectories changed behavior during interval $[t_1, t_2]$, this is a *mutual* avoidance. In Figure 7 (right) at any time in $[t_1, t_2]$ for both trajectories the distance of the forecast position from the actual position is less than δ (the height of the gray rectangle). Nevertheless, trajectories were expected to meet but did not actually meet (their distance is larger than δ) and thus the *avoid* predicate is true. Since the *change-behavior* predicate is false for both trajectories, this is a *weak* avoidance.

4.2. Problem Statement

In this paper we address two problems of increasing complexity regarding the detection of avoidance behaviors between moving object trajectories: the *avoidance decision problem* (i.e., decide whether a pair of trajectories in a given temporal interval satisfies the predicate *avoid*) and the *avoidance search problem* (for every possible pair of trajectories find those time intervals such that the predicate *avoid* is true).

We define both problems on trajectory tracks since real trajectory datasets usually consist of sets of samplings. To obtain the continuous representation of a trajectory we use an interpolation function, namely *interp*, and we denote as $interp(\mathcal{T})$ the application of *interp* to the trajectory track $\mathcal{T}$.

We check avoidances in intervals starting at the samples of a trajectory and the duration of the interval is based on a *look-ahead* time Δt, which can be interpreted as a measure of the future awareness of moving objects. Consequently, Δt must be chosen according to their characteristics. For instance, pedestrians have a consistently smaller look-ahead time than big ships (e.g., cargos), since in the latter case, due to the tonnage and the size of the objects, changes of heading or speed are necessarily much slower. Also the choice of the meet threshold δ is related to the characteristics of the moving objects. It varies for different types of moving objects and for different conditions, as remarked in [11] too.

The first problem we address is the decision problem.

Definition 4.7 (Avoidance decision problem). *Given two trajectory tracks $\mathcal{T}_a$,$\mathcal{T}_b$, the set of their timestamps $TS = \{timestamp(p) \mid p \in \mathcal{T}_a \vee p \in \mathcal{T}_b\}$, a time instant t, such that $t \in TS$, a look-ahead interval Δt, and a meet threshold δ, the* avoidance decision problem *consists in determining whether*

$$\text{avoid}_\delta(interp(\mathcal{T}_a), interp(\mathcal{T}_b), [t, t + \Delta t]) \textit{ holds.}$$

The second problem we address is a *search* problem: for each pair of trajectories we look for all the timestamps belonging to one of the two trajectory tracks such that the predicate *avoid* holds and we determine the associated type of avoidance. As a first step we formulate this problem for a pair of trajectories in order to find the set of timestamps and the relative avoidance type where an avoidance between such trajectories is detected. Then the notion will be generalized to a set of trajectories.

Definition 4.8 (Avoidances set). *Given two trajectory tracks $\mathcal{T}_a$,$\mathcal{T}_b$, the set of their timestamps $TS = \{timestamp(p) \mid p \in \mathcal{T}_a \vee p \in \mathcal{T}_b\}$, a look-ahead time Δt and a meet threshold δ, we define*

$$\begin{aligned} avoidances_\delta(\mathcal{T}_a, \mathcal{T}_b) = \{(t, type) \mid\ & t \in TS\ \wedge \\ & avoid_\delta(interp(\mathcal{T}_a), interp(\mathcal{T}_b), [t, t + \Delta t])\ \wedge \\ & type\_avoid_\delta(interp(\mathcal{T}_a), interp(\mathcal{T}_b), [t, t + \Delta t]) = type\} \end{aligned}$$

In words, a pair $(t, type) \in avoidances_\delta$ when t is a timestamp in one of the trajectory tracks and the predicate *avoid* holds for the two trajectories in the interval $[t, t + \Delta t]$. *type* is the kind of the detected avoidance.

Definition 4.9 (Avoidance search problem). *Given a set of trajectory tracks $\mathcal{D}$, a look-ahead time Δt, and a meet threshold δ, the* avoidance search problem *consists in finding the set of tuples*

$$\{(\mathcal{T}_a, \mathcal{T}_b, (t, type)) \mid \mathcal{T}_a \in \mathcal{D} \wedge \mathcal{T}_b \in \mathcal{D} \wedge (t, type) \in avoidances_\delta(\mathcal{T}_a, \mathcal{T}_b)\}$$

each consisting of a pair of trajectory tracks $\mathcal{T}_a, \mathcal{T}_b$, a time instant t and a type type such that $\mathcal{T}_a$ and $\mathcal{T}_b$ avoid to meet, with threshold δ, after t for Δt time and the kind of avoidance is specified by type.

In order to get a more compact representation of the result set, we next replace the set of timestamps for a couple of trajectories ($avoidances(\mathcal{T}_a, \mathcal{T}_b)$) with a disjoint set of intervals. The idea is to join two avoidances if the intervals where they are detected overlap. To this aim, we introduce the notion of *repeated avoidance* in an interval.

Definition 4.10 (Repeated avoidance). *Given two trajectory tracks $\mathcal{T}_a$,$\mathcal{T}_b$, the set of their timestamps $TS = \{timestamp(p) \mid p \in \mathcal{T}_a \vee p \in \mathcal{T}_b\}$, a look-ahead time Δt, and $t_1, t_2 \in TS$ we say that the interval $[t_1, t_2]$ contains a* repeated avoidance*, written $repeated\_avoid_\delta(\mathcal{T}_a, \mathcal{T}_b, [t_1, t_2], type)$, when for any $t \in [t_1, t_2]$ there exists $t' \in TS$ such that $t \in [t', t' + \Delta t]$ and*

$(t', \_) \in avoidances_\delta(\mathcal{T}_a, \mathcal{T}_b)$. *Moreover,* $type = sup\{type' \mid t' \in [t_1, t_2] \cap TS \wedge (t', type') \in avoidances_\delta(\mathcal{T}_a, \mathcal{T}_b)\}$ *where*

$$sup(X) = \begin{cases} weak & if \ \forall x \in X \ x = weak \\ mutual & if \ \exists x \in X \ x = mutual \\ individual & otherwise \end{cases}$$

Hence, the compression of the set of avoidances consists of a set of *maximal* intervals satisfying the *repeated_avoid* predicate. The type of a repeated avoidance is the upper bound of the types of the avoidances occurring in the associated maximal interval.

Definition 4.11 (Compression of the set of avoidances). *Given two trajectory tracks* $\mathcal{T}_a$,$\mathcal{T}_b$*, the set of their timestamps* $TS = \{timestamp(p) \mid p \in \mathcal{T}_a \vee p \in \mathcal{T}_b\}$*, a look-ahead time* Δt*, the compression of the set* $avoidances_\delta(\mathcal{T}_a, \mathcal{T}_b)$ *is defined as follows:*

$$\begin{aligned} compressed\_avoid_\delta(\mathcal{T}_a, \mathcal{T}_b) = \{([t_1, t_2], type) \mid\ & t_1, t_2 \in TS \wedge \\ & repeated\_avoid_\delta(\mathcal{T}_a, \mathcal{T}_b, [t_1, t_2], type) \wedge \\ & [t_1, t_2] \ \ maximal\} \end{aligned}$$

5. Algorithmic Framework

In this section we first present an algorithm for solving the avoidance search problem (Section 5.1) and then make some considerations on the choice of the right parameters to detect avoidances. As a consequence, we propose two different strategies for using our algorithm: the *simple detector* and the *fused detector*. The simple detector consists of a single execution of the algorithm with a fixed set of parameters. The fused detector consists of multiple executions of the algorithm with various parameter sets; result sets produced by individual executions are finally merged into a single result set. Both detectors are formalized in Section 5.2.

5.1. An Algorithm for Avoidance Detection

In order to solve the problems posed in Section 4.2 we propose Algorithm 1 which, given a finite set of trajectory tracks $\mathcal{D}$, a meet threshold δ and a look-ahead time Δt, returns a set of avoidance behaviors A, consisting of tuples specifying a pair of trajectories, a compressed interval in which the avoidance is detected between such trajectories and the relative type of avoidance.

The algorithm starts by considering every possible pair of trajectory tracks in $\mathcal{D}$ and, for each pair, it determines the first and last useful sample timestamps of each track with respect to the look-ahead time Δt, (functions *getTimeFrame*, lines 4-5). More precisely, if t_m^{start} and t_m^{end} denote, respectively, the timestamps of the very first and last samples of a trajectory track $\mathcal{T}_m$, then the timestamps of the first and last *useful* samples with respect to Δt are determined as *getTimeFrame*$(\mathcal{T}_m, \Delta t) = (min\{t \mid p \in \mathcal{T}_m \ \wedge \ p = (x, y, t) \ \wedge \ t \geq t_m^{start} + \Delta t\}, max\{t \mid p \in \mathcal{T}_m \ \wedge \ p = (x, y, t) \ \wedge \ t \leq t_m^{end} - \Delta t\})$.

Algorithm 1: AVOIDANCE BEHAVIOR DETECTION

```
Input: Finite set of trajectory tracks $\mathcal{D}$, meet threshold $\delta$, look-ahead time $\Delta t$.
Output: Set of avoidance behaviors $A$.

1  begin
2      $A = \emptyset$
3      foreach $\mathcal{T}_m, \mathcal{T}_n \in \mathcal{D}$ with $m < n$ do
4          $(t_m, t_m^{last}) = getTimeFrame(\mathcal{T}_m, \Delta t)$
5          $(t_n, t_n^{last}) = getTimeFrame(\mathcal{T}_n, \Delta t)$
6          while $(t_m \leq t_m^{last} \wedge t_n \leq t_n^{last})$ do
7              $t = min(t_m, t_n)$
8              if $(avoid_\delta(interp(\mathcal{T}_m), interp(\mathcal{T}_n), [t, t + \Delta t]))$ then
9                  $type = type\_avoid_\delta(interp(\mathcal{T}_m), interp(\mathcal{T}_n), [t, t + \Delta t])$
10                 $([t_1, t_2], typelast) = getLastResult(A, \mathcal{T}_m, \mathcal{T}_n)$
11                 if $(t_2 + \Delta t < t)$ then
12                     $appendResult(A, \mathcal{T}_m, \mathcal{T}_n, [t, t], type)$
13                 else
14                     $removeLastResult(A, \mathcal{T}_m, \mathcal{T}_n)$
15                     $appendResult(A, \mathcal{T}_m, \mathcal{T}_n, [t_1, t], sup\{typelast, type\})$
16             if $(t = t_m)$ then
17                 $t_m = nextSampleTime(\mathcal{T}_m, t_m)$
18             if $(t = t_n)$ then
19                 $t_n = nextSampleTime(\mathcal{T}_n, t_n)$
20     return $A$
```

Algorithm 2: THE FUNCTION *type_avoid*

```
Input: Trajectories $T_m, T_n$, time interval $[t_1, t_2]$, meet threshold $\delta$ such that $avoid_\delta(T_m, T_n, [t_1, t_2])$ is true.
Output: Type of the avoidance.

1  begin
2      $type = weak$
3      if $(change\text{-}behavior_\delta(T_m, [t_1, t_2]) \wedge change\text{-}behavior_\delta(T_n, [t_1, t_2]))$ then
4          $type = mutual$
5      else if $(change\text{-}behavior_\delta(T_m, [t_1, t_2]) \vee change\text{-}behavior_\delta(T_n, [t_1, t_2]))$
         then
6          $type = individual$
7      return $type$
```

Once the first and last useful samples of both trajectory tracks are determined (if any), the algorithm starts scanning the trajectories from the first useful samples (line 6). The sample with the smaller timestamp, t, is chosen (line 7) and the algorithm verifies if there is an avoidance in the interval $[t, t + \Delta t]$ (line 8). In case the *avoid* predicate is true, the type of the avoidance (Definition 4.6) is determined using Algorithm 2 (line 9).

Avoidance classification (Algorithm 2) first determines whether the avoidance is mutual (line 3) or individual (line 5) by testing the *change-behavior* predicate. It could happen that there is not enough evidence to further specify the avoidance, hence it remains labeled as a *weak* avoidance.

Once the avoidance has been detected and classified, the algorithm (Algorithm 1) checks the previous avoidance, identified by $[t_1, t_2]$ with type *typelast*, for the two trajectory tracks under investigation $\mathcal{T}_m$, $\mathcal{T}_n$ (line 10). It verifies whether the interval $[t_1, t_2 + \Delta t]$ and the interval $[t, t + \Delta t]$ are disjoint. In this case the algorithm adds the new avoidance $[t, t]$ with *type* (line 12). Otherwise, it merges the two avoidances, by removing the subsumed one from the result set and inserting the interval $[t_1, t]$ associated with the least upper bound between *typelast* and *type* (lines 13-15). The definition of *sup* is given in Definition 4.10.

After having processed a sample, the function *nextSampleTime* returns the timestamp of the next sample(s) in the trajectory track(s) (lines 16-19).

The algorithm, for each pair of trajectories, linearly scans the union of their samples. Given N trajectories having, on average, M samples, the outer *foreach* loop (line 3) is executed $N * (N - 1)/2$ times since each trajectory has to be compared with the others. The inner *while* loop (line 6) is executed at most $2 * M$ times, since we scan each point of the trajectories only once. This leads to an overall complexity of $O(N^2 * M)$. We remark that the algorithm is just intended to show that the detection of avoidance patterns is feasible and, when considering real-world datasets, is able to provide meaningful results. Clearly, the algorithm can be improved in many aspects; in particular, a sensible gain in efficiency would derive from the introduction of a preprocessing phase able to filter out unnecessary information prior to the avoidance behavior detection phase, hence reducing the overall amount of data to be analyzed. Additionally, the algorithm can be trivially parallelized with respect to the set of trajectory pairs under investigation, since each pair can be processed independently, thus entailing linear speedups with respect to the number of cores used. A thorough discussion of these improvements is out of the scope of this paper and is deferred to future work.

5.2. Avoidance Detectors

In this section we present two different ways of using the algorithm proposed in the previous section for avoidance detection: *single detector* and *fused detector*. *Single detector* represents a single run of the algorithm using a fixed pair of $(\Delta t, \delta)$ values.

Given a pair of trajectory tracks $(\mathcal{T}_a$,$\mathcal{T}_b)$, a meet threshold δ and a look-ahead time Δt, the **single detector** returns the (possibly empty) set:

$$\mathcal{RS}^{\delta}_{\Delta t} = \{I_i\}_{i=1,\dots,m} = \{[t^i_s, t^i_e]\}_{i=1,\dots,m} \tag{1}$$

where t^i_s and t^i_e (with $t^i_s \leq t^i_e$) denote the *starting* and *ending* timestamps of the i–th avoidance, $I_i = [t^i_s, t^i_e]$ denotes a temporal interval during which an avoidance occurs, while the set $\{I_i\}_{i=1,\dots,m}$ consists of *disjoint intervals*, i.e., $\forall i, j,\ I_i \cap I_j = \emptyset$.

Depending on the set of parameter values, an avoidance can be detected or missed. To minimize this problem, we propose a fusion detector that will detect the avoidance with different sets of parameters, and fuse the results in a unique output. We call this the **fused detector**.

Given a fixed look-ahead time Δt, and a monotonically increasing sequence of meet thresholds $\langle \delta_1, \dots, \delta_h \rangle$, where $\delta_1 < \dots \delta_i < \dots < \delta_h$, we can *fuse* the result sets obtained for each δ_i, still obtaining a disjoint set of temporal intervals. Specifically,

$$\mathcal{RS}^{\langle \delta_1,\dots,\delta_h \rangle}_{\Delta t} = \biguplus_{j=1,\dots,h} \mathcal{RS}^{\delta_j}_{\Delta t} \tag{2}$$

where the result set $\mathcal{RS}^{\langle \delta_1,\dots,\delta_h \rangle}_{\Delta t}$ is obtained by *fusing* the interval sets obtained by all the single detectors with parameters $\delta \in \{\delta_1, \dots, \delta_h\}$. The operation $\biguplus$ is a simple set-union of the various intervals, except that the groups of *overlapping intervals* – such that, for each interval I in a group, there is at least another interval I' belonging to the same group, where $I \cap I' \neq \emptyset$ – are fused and replaced by a single larger interval, that spans all the overlapping ones. Note that this guarantees that the final fused result set is still composed of disjoint intervals, each one representing a distinct avoidance.

6. Experimental Evaluation

The goal of this section is to evaluate the proposed algorithm under different points of view. First, we quantitatively test the *effectiveness* of the algorithm by analyzing the ability to correctly detect expected avoidance behaviors. To this end we use an ad-hoc annotated dataset, our ground truth, created for the purposes of this work (Section 6.1). Second, we compare our approach with the one presented in [11] since, as pointed out in Section 2, this is the only work we are aware of that deals with avoidance detection between moving objects (Section 6.2). Third, we assess the ability of the algorithm in highlighting interesting and previously unknown patterns emerging from avoidance behaviors when using datasets for which no prior knowledge related to avoidance behaviors is available (Section 6.3).

6.1. Experiments with the Ground-Truth Dataset

We exploit a *ground truth* of annotated trajectories, explicitly created for this work, whose data derive from real GPS observations of moving objects collected in Florianopolis and Venice. This public dataset[1] contains an overall amount of 86 trajectories representing pedestrian movements, for a total of 7,834 samples and an average sampling rate of one second. Although this dataset is not large, it is sufficient to validate the proposed method.

In this trajectory dataset, 32 pairs of trajectories are labeled as *positive*, since they exhibit at least one avoidance. Among the positive pairs, eight exhibit two distinct avoidances while the remaining ones exhibits a single avoidance.

For each positive pair, the set of intervals during which the avoidance(s) occur(s) is reported as well. The positive/negative labels and the temporal intervals referring to single avoidances were given by human assessors. This may result in imprecise annotation of temporal intervals, since their span may depend on the perception of assessors.

In order to evaluate the effectiveness of the (simple/fused) detector that solves the *decision problem* (Definition 4.7), for each pair of trajectories we check the correctness of the detector by verifying if the yes/no answer matches the positive/negative label in the ground truth. The detector returns *yes* if a non-empty set of avoidances is detected, *no* otherwise.

On the other hand, in order to evaluate the quality of the detector that solves the *search problem* (Definitions 4.9 and 4.11), for each positive pair correctly detected we also inspect the temporal intervals returned by the detector, by comparing them with the ones associated by the human assessor in the ground truth.

In the following we define the metrics used to evaluate the proposed method.

Decision problem. For this study we recur to well-established metrics commonly used in data mining to evaluate the quality of a classifier [27]. Given a set of N pairs of trajectories, we evaluate the results of the detector algorithm by constructing an integer *confusion matrix* (see Table 1), a 2×2 table where the number of *true positives* (tp) and *true negatives* (tn) are given in the main diagonal, while the anti-diagonal contains the number of *false positives* (fp) and *false negatives* (fn) detected. Clearly, $N = tp + tn + fp + fn$.

We call *positive* any trajectory pair in the ground truth that is labeled as *avoidance behavior* $=$ *yes*, while we use the term *negative* otherwise. Hence, tp and tn correspond to the pairs which the detector labels correctly *yes* or *no*, respectively, while fp and fn correspond to mislabeled pairs. Specifically, fp (fn) are pairs that the detector labels as positive (negative), but in fact appear as negative (positive) in the ground truth.

Recall and *Precision* are two widely used metrics employed in applications where successful detection of positive cases, i.e., in our case trajectory pairs for which an avoidance behavior is observed, is considered more significant than detection of other behavior. A formal definition of these metrics is given below:

$$\text{Precision,} \quad p = \frac{tp}{tp + fp} \qquad\qquad \text{Recall,} \quad r = \frac{tp}{tp + fn}$$

[1]anonymous link: https://www.dropbox.com/s/y7ch5v8yvyj6ivs/dataset.tar.gz?dl=0

		Detected	
		positive	*negative*
Actual	*positive*	tp	fn
	negative	fp	tn

Table 1 Confusion matrix.

Precision p and recall r can be summarized into another metric known as *F-Measure*, defined as follows:

$$\text{F-Measure}, \ \ F = \frac{2 \cdot r \cdot p}{r + p}$$

where $0 \le F \le 1$.

Search problem. As previously stated, for positive pairs correctly detected by the simple/fused detector we also inspect the temporal intervals returned by comparing them with the ones associated by human assessor in the ground truth.

Let TP be the set of positive pairs in the ground truth that were correctly identified by the detector, i.e., pairs for which the result set returned by the algorithm is not empty. Assuming that we are using a given combination of Δt and δ, for clarity purposes in this context we denote such result set as $\mathcal{RS}^p$ for a pair of trajectories p, omitting the parameters symbols in the notation. For each pair $p \in TP$, we know the set of actual disjoint temporal intervals $\mathcal{G}^p = \{\overline{I}_1, \ldots, \overline{I}_k\}$ in the ground truth, associated with $k > 0$ avoidances. The detector, either single or fused, also returns a set of m intervals $\mathcal{RS}^p = \{I_1, \ldots, I_m\}$.

Let $\widehat{\mathcal{G}}^p = \{\overline{I}_j \in \mathcal{G}^p \mid \exists! I_h \in \mathcal{RS}^p \ s.t. \ (\overline{I}_j \cap I_h \neq \emptyset) \ \wedge \ (\nexists \overline{I}_k \in \mathcal{G}^p, k \neq j \ s.t. \ \overline{I}_k \cap I_h \neq \emptyset)\}$, $\widehat{\mathcal{G}}^p \subseteq \mathcal{G}^p$, be the set of intervals in $\mathcal{G}^P$ such that each interval overlaps with *only one* interval in $\mathcal{RS}^p$, and the latter does not overlap with any other intervals in $\mathcal{G}^p$.

We can quantitatively evaluate the quality of the results for all the pairs in TP by *Q-Measure*:

$$\textit{Q-Measure} = \frac{\sum_{p \in TP} \frac{|\widehat{\mathcal{G}}^p|}{|\mathcal{G}^p|}}{|TP|}$$

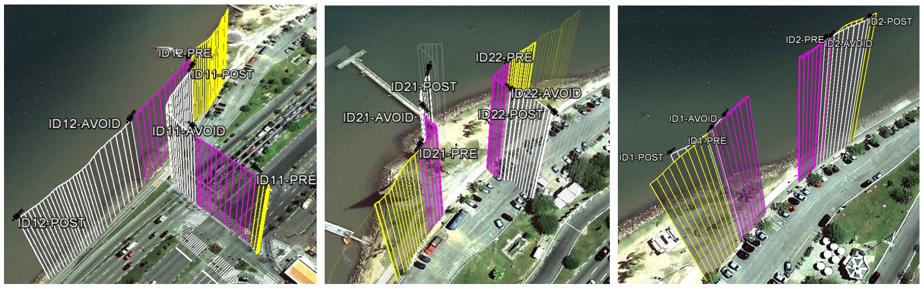
where $0 \le \textit{Q-Measure} \le 1$. Ideally *Q-Measure* should be equal or close to 1.

In the following we present some visual examples of the output of the algorithm (Section 6.1.1), then in Sections 6.1.2 and 6.1.3 we show quantitatively the results obtained by exploiting the (simple/fused) detector that solves the decision or the search problem.

In all the experiments described below, we discuss the various results obtained by changing the two main parameters of our avoidance detection algorithm, namely Δt and δ, whose values are reported in seconds and meters, respectively.

6.1.1. Visual inspection of avoidances

In order to visualize some avoidances detected by our algorithm on the ground truth dataset, we use Google Earth. Specifically, the avoidances shown in Figure 8 refer to trajectory pairs collected in Florianopolis. The subset of segments highlighted in *purple* represents the set of samples over which an avoidance is detected. The segments highlighted in *yellow* and *white* represent, respectively, a fixed sequence of samples occurring *before* and *after* the avoidance detected by the algorithm. Figure 8(a) represents an individual avoidance by entity ID11 (which moves initially from bottom-right to top-left), slowing down at some point in order to avoid ID12 (moving from top-right to bottom-left). Figure 8(b) depicts a mutual avoidance where ID21 (moving from bottom-left to top-right) and ID22 (moving in the opposite direction) change their direction as soon as they get close. Finally, Figure 8(c) depicts a mutual avoidance where the two entities invert their direction as soon as they get too close.



(a) (b) (c)

Figure 8 Examples of three visual inspections performed on three different avoidances returned by the algorithm.

6.1.2. Results for the Decision Problem

In this section we evaluate the ability of the detectors (simple detector and fused detector) of correctly identifying the trajectory pairs of the ground truth that are positively/negatively labeled (decision problem).

Simple detector. For each pair of parameters Δt and δ, we build the confusion matrix by considering all trajectories in the ground truth, then determine precision/recall, and finally compute the F-Measure scores. Figure 9 reports the scores obtained by the algorithm for all combinations of parameters. Specifically, each curve in the plot refers to a distinct Δt, and shows the F-Measure score as a function of δ.

For almost all values of Δt, a common optimal value for the meet threshold δ that maximizes the F-measure score falls in the interval $[3, 6]$; for larger values of δ, the score degrades. It is worth noting that these optimal values for δ approximately reflect the average avoidance distance used to physically produce the avoidances for (positive) trajectory pairs included in the ground truth.

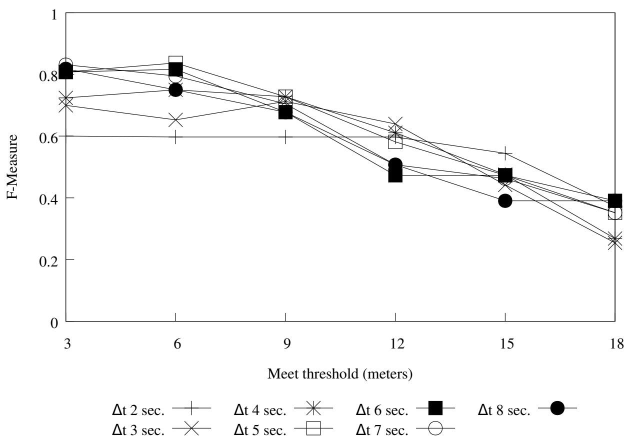


Figure 9 Decision problem with simple detector: F-Measure analysis.

Fused detector. In the following we show how the overall performance of the algorithm can substantially be improved by *fusing* different result sets of distinct simple detectors, according to the fusion operator defined by Equation (2).

Given a Δt, and a sequence of values for δ, e.g., $\langle 3, 6, 9 \rangle$, a pair of trajectories is identified as a positive case, i.e., *avoidance behavior* = *yes*, if at least a simple detector for some δ in the sequence identifies one or more avoidance behaviors. Conversely, if no simple detector is able to recognize any avoidance, the pair is identified as a negative case, i.e., *avoidance behavior* = *no*. Still, for the fused detector we can build the confusion matrix, by considering all trajectory pairs in the ground truth, and finally compute the F-Measure scores.

Considering the results shown in Figure 9, we choose $\Delta t \in \{4, 5, 6, 7\}$ to evaluate the fused detector. For each Δt, we compute the F-Measure related to the fused result sets $\mathcal{RS}_{\Delta t}^{\langle \delta=3 \rangle}, \mathcal{RS}_{\Delta t}^{\langle \delta=3, \delta=6 \rangle}, ..., \mathcal{RS}_{\Delta t}^{\langle \delta=3, \delta=6, ..., \delta=18 \rangle}$ (each one defined as per Equation 2). Results are reported in Figure 10, where each fused result set is represented by its upper δ threshold in the X-axis. The figure shows how the fused detector entails substantial improvements in terms of F-Measure (up to 95%), provided that Δt is properly chosen according to the dataset features.

In general, we argue that the opportunity of fusing different result sets depends on the kind of analysis we want to perform. Specifically, it depends on the classes of avoidance behaviors we want to discover (e.g., only values for δ that are relevant for our purposes should be used for the fusion operation), and on the amount of useful information an analyst is interested in extracting at the expense of possible losses in precision (due to the detection of false avoidances).

6.1.3. Results for the Search Problem

In this section we assess the quality of the temporal intervals reported by both detectors (simple and fused), for the positive pairs correctly detected, and

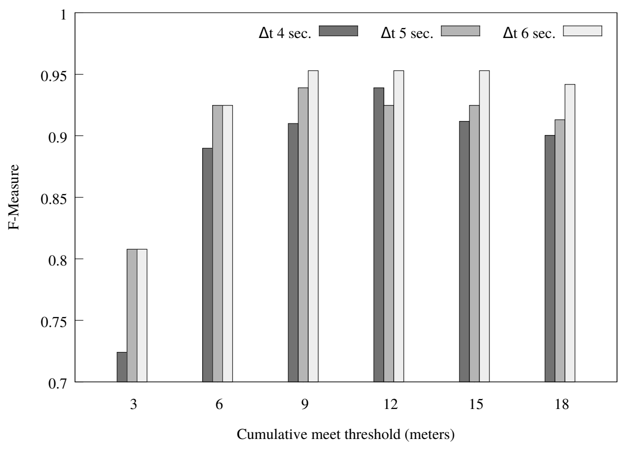


Figure 10 Decision problem with fused detector: F-Measure analysis. In X-axis the values are the maximums of the thresholds δ used during the fusion operation, e.g., 9 is the maximum for the set $\{3, 6, 9\}$.

thus included in the set of trajectory pairs TP, where $tp = |TP|$.

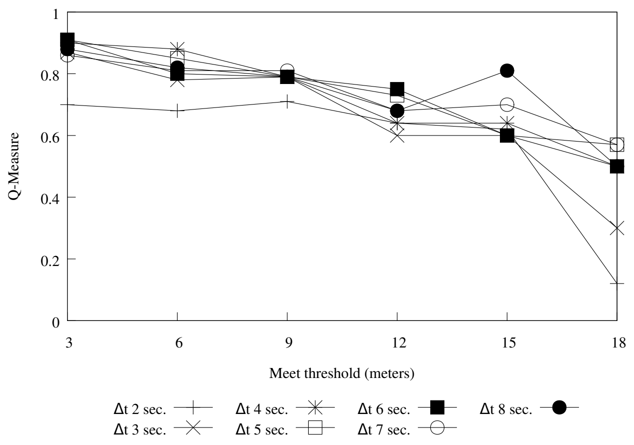


Figure 11 Search problem with simple detector: Q-Measure analysis.

Simple detector. Figure 11 reports the Q-Measure scores obtained in the experiments. The experimental findings confirm all the remarks done in Section 6.1.2 concerning Δt and δ. In general, we obtain the best Q-Measure score for the same parameters Δt and δ for which we obtained the best F-Measure scores. We also point out that using high values of Δt (where *high* is relative to the dataset features) may erroneously induce the fusion of distinct avoidance behaviors, due to compression, thus potentially mapping multiple avoidances occurring between two trajectories in the ground truth to a single detected avoidance. This in turn induces losses in terms of Q-Measure scores.

Fused detector. Also for the fused detector we aim at analyzing the quality of the temporal intervals for the trajectory pairs in TP. To this end, we consider again the fused result sets belonging to $\{\mathcal{RS}_{\Delta t}^{\langle\delta=3\rangle}, \mathcal{RS}_{\Delta t}^{\langle\delta=3,\delta=6\rangle}, ..., \mathcal{RS}_{\Delta t}^{\langle\delta=3,\delta=6,...,\delta=18\rangle}\}$.

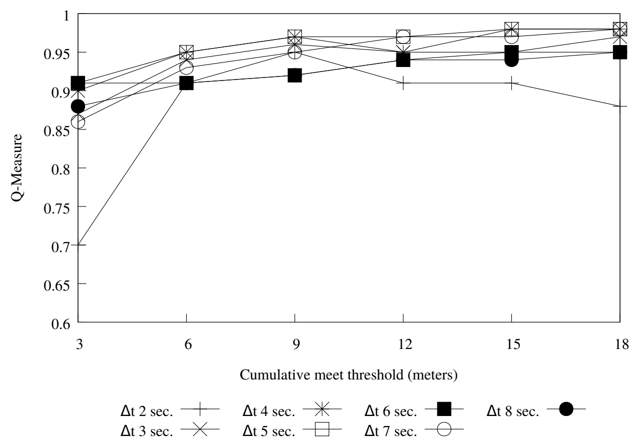


Figure 12 Search problem with fused detector: Q-Measure analysis. In X-axis the values are the maximums of the thresholds δ used during the fusion operation, e.g., 9 is the maximum for the set $\{3, 6, 9\}$.

Figure 12 reports the performance of the algorithm in terms of Q-Measure score. We can observe how fusing different result sets entails substantial improvements. These improvements are particularly evident whenever the fusion is performed for δ values close to the average distances used for physically producing avoidances ($\delta \in [3, 6]$).

6.2. *Comparison with* ApproxCount

In this section we report the results of some experiments aimed at comparing our technique with ApproxCount, the approach proposed in [11].

As already explained in Section 2, the goal of ApproxCount is to find *statistical evidence* of general avoidance/attraction relationships between moving objects. By means of permutation tests, ApproxCount tries to infer whether pairs of moving objects, sharing a common territory, are consistently attracted to each other (e.g., animals sharing common resources, therefore tending to meet in the same locations) or consistently avoiding each other (e.g., preys avoiding predators). Nonetheless, in the context of our work it is interesting to check whether the statistical model underlying ApproxCount is able to "sense" avoidance episodes according to the definitions provided in our work.

In order to test ApproxCount we used the application kindly provided by the authors[2]. The parameters fed into ApproxCount were (i) the *gap threshold* (0.5 seconds), which represents the minimum amount of time below which ApproxCount interpolates between the original trajectory samples, (ii) the *number of rounds* (1000) during which a single permutation test is performed and (iii) the *distance threshold* δ below which we have a meet between moving objects; in our experimental setting we used again $\delta \in \{3, 6, 9, 12, 15, 18\}$ meters. For each pair of moving objects ApproxCount outputs a real value, *avoid*, ranging over the interval $[0, 1]$: the closer *avoid* is to zero, the stronger is the avoidance relationship between the moving objects.

The comparison focuses on the decision problem which can be solved by both approaches. We note, in particular, that ApproxCount is not suited when dealing with the research problem. In fact, as already mentioned previously, ApproxCount analyzes the global behavior of a pair of trajectories without identifying when and where individual avoidance episodes occur. For the decision problem, we say that ApproxCount answers *yes* whenever the output *avoid* is *less than* 0.3, which represents a conservative value with respect to the suggestions given in [11]. ApproxCount is compared with our fused detector, which in the following we denote by DetectAvoid.

The results of the experimental comparison are reported in Figure 13. We observe that, as far as F-Measure is concerned, ApproxCount lags noticeably behind DetectAvoid. A separate analysis of the recall and precision measures reveals that the discrepancy is mainly due to recall; indeed, DetectAvoid is able to find almost all the trajectory pairs having an avoidance behavior. The recall values are particularly good for $\Delta t \in \{5, 6\}$ when δ increases. Instead, ApproxCount yields 56% of recall as its best result. Concerning precision, both algorithms achieve similar results. We believe that these results are justified by the fact that ApproxCount does not detect trajectory pairs showing occasional avoidance episodes, since it is designed to detect more general avoidance relationships.

6.3. Analysis of a Real World Unannotated Dataset

In this section we consider the AIS Brest dataset, a real world unannotated dataset containing 824 trajectories related to the movements of 824 ships[3] nearby Brest's harbor [28]. Basic statistics reveal that the dataset contains 5.756.438 points, the trajectories move at an average speed of 7.77 km/h and most of the trajectories have an average sampling rate between 1 and 20 seconds. These characteristics make the dataset quite interesting in terms of the precision with which the trajectories are described.

By considering the aforementioned statistics, and after a scrutinization of different meet thresholds and look-ahead times, according to entities' features we chose a meet threshold δ of 30 meters and a look-ahead time Δt of 50 seconds.

[2]The application can be retrieved at `https://faculty.ist.psu.edu/jessieli/MoveMine`

[3]Each trajectory is uniquely associated with a ship.

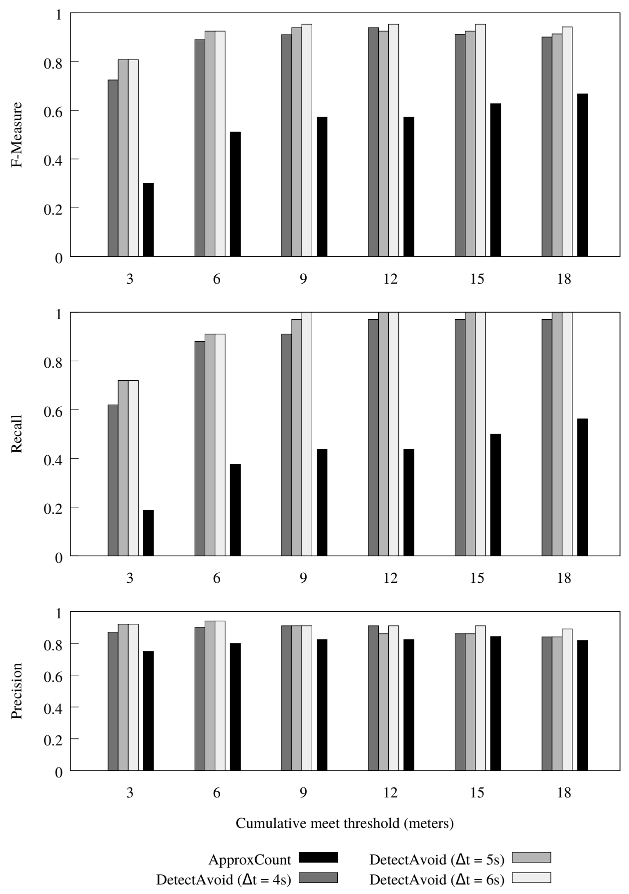


Figure 13 Comparison between DetectAvoid and ApproxCount.

Indeed, we argue that for this case study this combination of parameters allows us to capture interesting patterns, as we will show further on.

The algorithm detects a total of 1480 avoidances, among which 321 are mutual, 970 individual and 189 weak. Given this considerable amount of information it is necessary to perform a deeper analysis in order to infer meaningful patterns.

Among the 824 ships, 229 are involved in at least one avoidance. We call this set as the *set of active ships.* If we further look at the number of avoidances in which each active ship is involved, we notice that 8 ships are involved in more than 100 avoidances, while the vast majority - more precisely 196 ships (which constitutes the 85,5% of the active ships set) - are involved in a number of avoidances between 1 and 10. We call the former set as the *set of frequent ships* while the rest of the ships ends up in the set of *infrequent ships.*

The information above suggests that the frequent ships play a very important role in the dataset. If we decompose the total amount of avoidances detected by the algorithm, we find out that the overall amount of avoidances between frequent and infrequent ships are 973 (65,7% of the result set), while the avoidances between frequent ships are 386 (26,1%) and between infrequent ones are 121 (8,2%).

If we look at the MMSI codes of the frequent ships in order to find out their type (Table 2), we have that the top-2 frequent ships are *pilot ships*, while the remaining ones are *passenger ships* and *tugboats*.

MMSI Code	*Type*	Amount of avoidances
227730220	Pilot ship	414
227005550	Pilot ship	364
227635210	Passenger ship	194
227592820	Passenger ship	175
227574020	Passenger ship	174
227612860	Passenger ship	158
227574030	Passenger ship	147
228051000	Tugboat	119

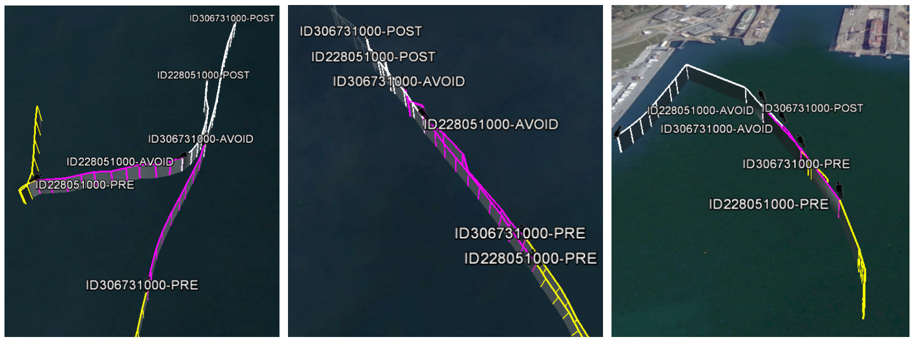
Table 2 *Frequent ships* details.

Given these data we want to answer the following questions:

1. Which are the events producing so many avoidances between frequent and infrequent ships?
2. Which are the events producing a considerable amount of avoidances between frequent ships?
3. Is there any kind of recurring pattern causing avoidances between infrequent ships?

When answering Question (1) we notice a dominant pattern (Figure 14) that we call *paired movement event*. Through a graphical inspection we observe that almost all these events can be decomposed in 3 phases: during the first phase the two ships approach each other, mostly when they are entering or exiting the harbor (*approach* phase, Figure 14(a)). Then, the ships proceed paired (*paired movement* phase, Figure 14(b)) until they approach the docks or they exit the harbor area. During this intermediate phase some avoidances may emerge or not, depending on the continuous adjustment performed by both ships in order to maintain the relative distance. Finally, the ships separate (*detach* phase), as shown in Figure 14(c).

Given the types of the frequent ships reported in Table 2 we argue that the pilot ships and the tugboats produce these events when they have to pilot (or tow, respectively) an incoming (or outgoing) ship. For what concerns the passenger ships, we argue that they adjust their trajectories in order to avoid other infrequent ships nearby; moreover, the amount of avoidances in which they are involved is justified by the fact that they are servicing, and thus repeatedly going through, a fixed route.



(a) Approach (b) Paired movement (c) Detach

Figure 14 Example of a paired movement event involving the frequent ship 228051000. The ships are moving from bottom to top.

In general we expect that these avoidances are mostly distributed in predetermined areas. Indeed, if we plot the avoidances occurring during a two-month window we see that they are approximately distributed on a fixed path going from the harbor's docks to the strait exit (Figure 15). This also gives an idea about the most dangerous or trafficked areas (especially near the docks).

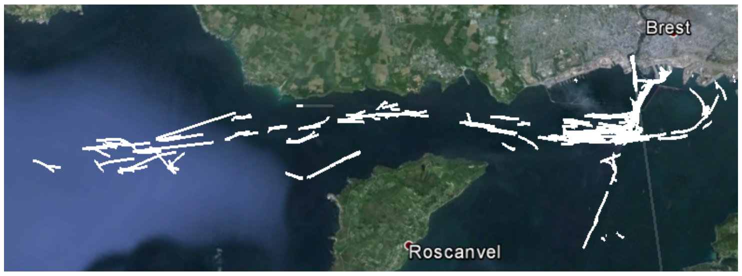


Figure 15 Subtrajectories related to avoidance behaviors detected in a time interval spanning two months ([20/04/2009, 20/06/2009]).

Concerning Question (2), we found that many avoidances are produced according to the same pattern observed for Question (1), or when a frequent ship is docking (and therefore slowing down, hence the avoidance) in the harbor nearby already docked ships. The latter pattern is observed between frequent and infrequent ships as well, although with a lesser extent.

Finally, as far as Question (3) is concerned, we found out that the second pattern observed when explaining the avoidances related to Question (2) also occurs, i.e., almost all the avoidances between infrequent ships happen near the docks when one or more ships are docked while one ship is docking nearby.

7. Conclusion

Several algorithms have been proposed for mining different types of trajectory patterns. However, an interesting behavior that has not been much explored in trajectories of moving objects is *avoidance*.

In this paper we have introduced a new type of trajectory pattern: avoidance between moving objects. We presented a set of theoretical definitions and an algorithm which is able to detect such a pattern. The discovery of avoidances between moving objects is challenging, since the intent of any moving object may not be immediately apparent from its trajectories. To determine an avoidance, two objects should move towards the same area at the same time, but either one or both should change their behavior when they come close enough to be aware of each other. To identify a behavior change we forecast the movements of both moving objects and compare them with the actual movements. If the forecasts predict a meet but the actual movements do not meet, an avoidance is detected. Each detected avoidance is in turn classified, whenever possible, as individual when only one moving object changes significantly its behavior, or as mutual when both objects change their behavior significantly.

It is worth mentioning that a behavior change is measured through the distance between the forecast movement and the actual movement. Such generalization allows us to detect changes of behavior without using specific conditions on speed or direction.

In this paper, besides analyzing the parameters of the algorithm, we went one step further by introducing the idea of fused detector, which merges the result sets of several simple detectors (with different meet thresholds) in order to allow the detection of a possibly broad range of avoidance behaviors. The proposed approach for avoidance detection makes use of only two parameters and is able to deal with trajectories collected at different sampling rates and/or having different temporal lengths.

The algorithm has been evaluated with two real-world datasets. The first dataset is annotated and it contains pedestrian movements; the purpose of analyzing such dataset is to verify that the algorithm is able to correctly detect avoidances which actually occur and, in the experiments, we demonstrate that this actually happens (F-measure up to 95%). The second real-world dataset, unannotated, contains ship movements nearby the Brest's harbor. Since no prior avoidance information is available, the purpose of analyzing such dataset is to check whether the algorithm is able to extract interesting evidence from the data. Indeed, by characterizing the avoidances between ships on the basis of their frequency, their spatial distribution and by means of visual inspections on the behavior of frequent ships, we were able to highlight the most trafficked areas, as well as a frequently recurring event, i.e., the paired movement event.

Future work includes an analysis on the effect of using different forecast functions and the definition of a confidence measure to evaluate the avoidance.

Acknowledgments

This work has been supported by the EU's IRSES-SEEK (grant 295179) and partially by Brazil Agency CNPq, the Italian Ministry of University and Research (MIUR) Projects PON ADAPT (no. SCN00447), and by the Italian Ministry of Economic Development Project MOTUS (no. MS01 00015 – Industria2015).

Appendix A. Main symbols

In the following we report the main symbols used throughout the paper.

Symbol	***Description***
T	Trajectory.
$\mathcal{T}$	Trajectory track.
$\mathcal{D}$	Set of trajectory tracks.
t	Time instant.
TS	Set of timestamps associated with trajectory samples.
δ	Meet threshold.
Δt	Look-ahead time.
M_o	Movement of an object o which can be split into different trajectories.
$forecast(T, [t_1, t_2])$	Movement predictor mapping a trajectory T to its forecast in the interval $[t_1, t_2]$ by exploiting the behavior of T in the interval $[0, t_1]$.
$interp(\mathcal{T})$	Interpolation function mapping a trajectory track $\mathcal{T}$ to a trajectory T.
$meet_\delta(T_a, T_b, [t_1, t_2])$	Predicate indicating whether the trajectories T_a and T_b meet in the time interval $[t_1, t_2]$ according to the meet threshold δ.
$will\_meet_\delta(T_a, T_b, [t_1, t_2])$	Predicate indicating whether the trajectories T_a and T_b are *expected* to meet in the time interval $[t_1, t_2]$ according to the meet threshold δ.
$avoid_\delta(T_a, T_b, [t_1, t_2])$	Predicate indicating whether the trajectories T_a and T_b exhibit an avoidance behavior in the time interval $[t_1, t_2]$ according to the meet threshold δ.
$change\_behavior_\delta(T, [t_1, t_2])$	Predicate indicating whether the trajectory T exhibits a significant change in movement behavior in the time interval $[t_1, t_2]$ according to the meet threshold δ.
$type\_avoid_\delta(T_a, T_b, [t_1, t_2])$	Function ranging over $\{individual, mutual, weak\}$ indicating the avoidance type detected between trajectories T_a and T_b in the time interval $[t_1, t_2]$ and according to the meet threshold δ.
$\mathcal{RS}^{\delta}_{\Delta t}$	Result set produced by Algorithm 1 according to the meet threshold δ and the look-ahead time Δt.
$\mathcal{RS}^{\langle \delta_1, \ldots, \delta_h \rangle}_{\Delta t}$	Union of different result sets produced by different runs of Algorithm 1 with different δs and the look-ahead time Δt.

Table A.3 List of main symbols used throughout the paper.

References

[1] F. Giannotti, M. Nanni, F. Pinelli, D. Pedreschi, Trajectory pattern mining, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 330–339.

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

[3] M. Wachowicz, R. Ong, C. Renso, M. Nanni, Finding moving flock patterns among pedestrians through collective coherence, International Journal of Geographical Information Science 25 (11) (2011) 1849–1864.

[4] J. Gudmundsson, M. J. van Kreveld, Computing longest duration flocks in trajectory data, in: Proceedings of the 14th ACM International Symposium on Geographic Information Systems, ACM, 2006, pp. 35–42.

[5] Z. Li, B. Ding, J. Han, R. Kays, P. Nye, Mining periodic behaviors for moving objects, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining, ACM, 2010, pp. 1099–1108.

[6] R. Trasarti, F. Pinelli, M. Nanni, F. Giannotti, Mining mobility user profiles for car pooling, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 1190–1198.

[7] L. X. Pang, S. Chawla, W. Liu, Y. Zheng, On detection of emerging anomalous traffic patterns using GPS data, Data Knowl. Eng. 87 (2013) 357–373.

[8] F. de Lucca Siqueira, V. Bogorny, Discovering chasing behavior in moving object trajectories, Transactions in GIS 15 (5) (2011) 667–688.

[9] S. Dodge, R. Weibel, A.-K. Lautenschütz, Towards a taxonomy of movement patterns, Information Visualization 7 (3-4) (2008) 240–252.

[10] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, Y. Theodoridis, Z. Yan, Semantic trajectories modeling and analysis, ACM Computing Surveys 40.

[11] Z. Li, B. Ding, F. Wu, T. K. H. Lei, R. Kays, M. Crofoot, Attraction and Avoidance Detection from Movements, Proceedings of VLDB Endowment 7 (3) (2013) 157–168.

[12] L. O. Alvares, A. M. Loy, C. Renso, V. Bogorny, An algorithm to identify avoidance behavior in moving object trajectories, Journal of the Brazilian Computer Society 17 (3) (2011) 193–203.

[13] D.-J. Kim, K.-H. Park, Z. Bien, Hierarchical longitudinal controller for rear-end collision avoidance, IEEE Transactions on Industrial Electronics 54 (2) (2007) 805–817.

[14] I. Xausa, R. Baier, M. Gerdts, M. Gonter, C. Wegwerth, Avoidance Trajectories for Driver Assistance Systems via Solvers for Optimal Control Problems, in: Proceedings of the International Symposium on Mathematical Theory of Networks and Systems, Springer, 2012, pp. 1–8.

[15] M. R. Hafner, D. Cunningham, L. Caminiti, D. D. Vecchio, Cooperative collision avoidance at intersections: Algorithms and experiments, IEEE Transactions on Intelligent Transportation Systems 14 (3) (2013) 1162–1175.

[16] S. Nedevschi, S. Bota, C. Tomiuc, Stereo-based pedestrian detection for collision-avoidance applications, Transactions on Intelligent Transportation Systems 10 (3) (2009) 380–391.

[17] Y.-H. Liu, C.-J. Shi, A fuzzy-neural inference network for ship collision avoidance, in: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, IEEE Computer Society, 2005, pp. 4754–4754.

[18] J. M. Mou, C. van der Tak, H. Ligteringen, Study on collision avoidance in busy waterways by using AIS data, Ocean Engineering 37 (5) (2010) 483–490.

[19] S. Shandy, J. Valasek, Intelligent agent for aircraft collision avoidance, in: Proceedings of AIAA Guidance, Navigation, and Control Conference, American Institute of Aeronautics and Astronautics, 2001, pp. 1–11.

[20] A. Richards, J. P. How, Aircraft trajectory planning with collision avoidance using mixed integer linear programming, in: American Control Conference, 2002. Proceedings of the 2002, Vol. 3, 2002, pp. 1936–1941.

[21] M. Pechoucek, D. Sislak, Agent-based approach to free-flight planning, control, and simulation, IEEE Intelligent Systems 24 (1) (2009) 14–17.

[22] O. Khatib, Real-time obstacle avoidance for robot manipulator and mobile robots, International Journal of Robotics Research 5 (1) (1986) 90–98.

[23] J. Borenstein, Y. Koren, The vector field histogram-fast obstacle avoidance for mobile robots, IEEE Transactions on Robotics and Automation 7 (3) (1991) 278–288.

[24] S. M. Khansari-Zadeh, A. Billard, A dynamical system approach to realtime obstacle avoidance, Autonomous Robots 32 (4) (2012) 433–454.

[25] D. Sun, A. Kleiner, B. Nebel, Behavior-based multi-robot collision avoidance, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2014, pp. 1668–1673.

[26] C. Renso, S. Spaccapietra, E. Zimanyi (Eds.), Mobility Data: Modeling, Management, and Understanding, Cambridge University Press, Cambridge, UK, 2013.

[27] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, (First Edition), Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[28] L. Etienne, T. Devogele, A. Bouju, Spatio-temporal trajectory analysis of mobile objects following the same itinerary, Advances in Geo-Spatial Information Science 10 (2012) 47.