Semantic Web 0 (0) 1
IOS Press

A Stream Reasoning System for Maritime Monitoring

Georgios M. Santipantakis [a,*], Akrivi Vlachou [a], Christos Doulkeridis [a], Alexander Artikis [b], Ioannis Kontopoulos [b], and George A. Vouros [a]
[a] *Department of Digital Systems, University of Piraeus, Greece*
E-mails: gsant@unipi.gr, avlachou@aueb.gr, cdoulk@unipi.gr, georgev@unipi.gr
[b] *Institute of Informatics and Telecommunications, N.C.S.R. "Demokritos", Greece*
E-mails: a.artikis@iit.demokritos.gr, ikon@iit.demokritos.gr

Abstract. We present a stream reasoning system for monitoring vessel activity in large geographical areas. The system ingests a compressed vessel position stream, and performs, in real-time, spatio-temporal link discovery to calculate proximity relations between vessels and topological relations between vessel and static areas. Capitalizing on the discovered relations, a complex activity recognition engine, based on the Event Calculus, performs continuous pattern matching to detect various types of dangerous, suspicious and potentially illegal vessel activity. We evaluate the performance of the system by means of real datasets including kinematic messages from vessels, and demonstrate the effects of the highly efficient spatio-temporal link discovery on performance.

Keywords: Spatio-temporal link discovery, Complex activity recognition, Complex event processing, Event Calculus

1. Introduction

Nowadays, maritime surveillance systems that keep track of the daily operation of vessel fleets constitute an essential tool for large shipping companies, coast guards, as well as governmental agencies, due to their immense effect on economy and environment [27]. Apart from the typical application of trajectory management and optimization, such systems provide more advanced functionality including optimized port operations, monitoring emissions, improved maritime situation awareness, monitoring regulated markets (e.g., fishing), and increased maritime safety, opening new research challenges for the maritime domain [14].

Advances in navigation technology enable the real-time provision of vessel positional information for the benefit of surveillance systems. The Automatic Identification System (AIS)[1], for example, is a tracking system for identifying and locating vessels at sea through data exchange. This technology integrates a VHF transceiver with a positioning device (e.g., GPS), and other electronic navigation sensors, such as a gyrocompass or rate of turn indicator, thus producing a wealth of valuable data regarding the vessel and its current status. The acquisition of positional data is achieved either by AIS base stations along coastlines, or even by satellites when out of range of terrestrial networks. AIS messages typically contain noise and may be delayed. As this data is streamed in a maritime surveillance system, several operations need to be performed in real-time, including data cleaning, data integration, as well as complex maritime activity recognition.

We focus on complex activity recognition, which is fundamental for maritime surveillance, since it allows for the timely detection of various types of dangerous, suspicious and potentially illegal vessel behavior. We present a system that exploits spatio-temporal relations between vessels, or ves-

*Corresponding author. E-mail: gsant@unipi.gr.
[1] http://www.imo.org/en/OurWork/Safety/Navigation/Pages/AIS.aspx

sels and static areas of interest (e.g., protected areas), which are calculated in real-time by a dedicated component for spatio-temporal link discovery (stLD). These relations are provided as input to a complex activity recognition component, which is based on the 'Event Calculus for Run-Time reasoning' (RTEC) [6]. This is an Event Calculus [29] implementation with various optimization techniques for continuous narrative assimilation on data streams.

We address the problem of real-time spatio-temporal link discovery over streaming and archival data. This link discovery (LD) problem is challenging because of (a) the streaming nature of data, and (b) the complexity of evaluating spatio-temporal similarity functions to determine the links between spatio-temporal entities. There exists only limited work on spatio-temporal link discovery, especially in a streaming setting. Typically, LD tasks are solved by *filter-and-refine* algorithms that identify a set of candidate entities (for linking) in the *filtering step*, which need to be verified in the *refinement step*. The refinement step is the principal cost factor that determines the overall performance of LD, since it requires the evaluation of distance/similarity functions on complex geometrical entities. For the case of stream to static LD, we propose a filtering technique that improves the efficiency of the filtering step by eliminating entities that cannot be linked, thereby reducing the number of entities that have to be considered during refinement. For the case of streaming data only, we provide an efficient method for streaming LD, identifying proximity relations between moving vessels.

In earlier work, we presented a maritime monitoring system which employed RTEC for complex activity recognition [45]. In that work, RTEC performed spatial calculations to determine whether a vessel is close to a port, or within an area of interest. Furthermore, it approximated crudely the *nearby* relation between vessels by checking whether a pair of vessels is located within the same cell of a grid. In this work, we placed emphasis on the detection of spatial relations and developed a separate component for highly efficient spatio-temporal link discovery. Moreover, RTEC has at its disposal additional spatial relations—whether a vessel is *nearby* some area—and a much more accurate account of proximity between vessels.

To summarise, this paper makes the following contributions:

- We present a stream reasoning system integrating a component for spatio-temporal link discovery (stLD), and a component for recognizing complex activities by means of temporal pattern matching. This way, we extend earlier work [45] by optimizing the computation of spatial relations, leading to more efficient and accurate activity recognition.
- We propose a technique for LD between streaming positions of vessels and static areas, which improves the efficiency of the filtering step of LD, by eliminating entities that cannot be linked, thereby reducing the number of entities that have to be considered during refinement.
- We tackle the problem of spatio-temporal LD in a streaming environment, by discovering links in real-time between moving entities based on their proximity, an issue that has not been addressed in the literature so far.
- We evaluate the performance of the system by means of real datasets including AIS kinematic messages from vessels sailing in the Atlantic Ocean around the port of Brest (Brittany, France), spanning between 1 October 2015 to 31 March 2016.

The rest of this paper is organized as follows: Section 2 presents the system architecture. The key components of our prototype system are presented in Sections 3 and 4. Then Section 5 presents the empirical analysis. Finally, Section 6 provides an overview of related work, while Section 7 concludes the paper.

2. System Architecture

Our work is performed in the context of the datAcron (EU-funded H2020 Big Data) research project[2], which aims at advancing the management and integrated exploitation of voluminous and heterogeneous data-at-rest (archival data) and data-in-motion (streaming data) sources, so as to promote the safety and effectiveness of critical operations of moving objects in large geographical areas. The system architecture of the datAcron

[2] http://datacron-project.eu/

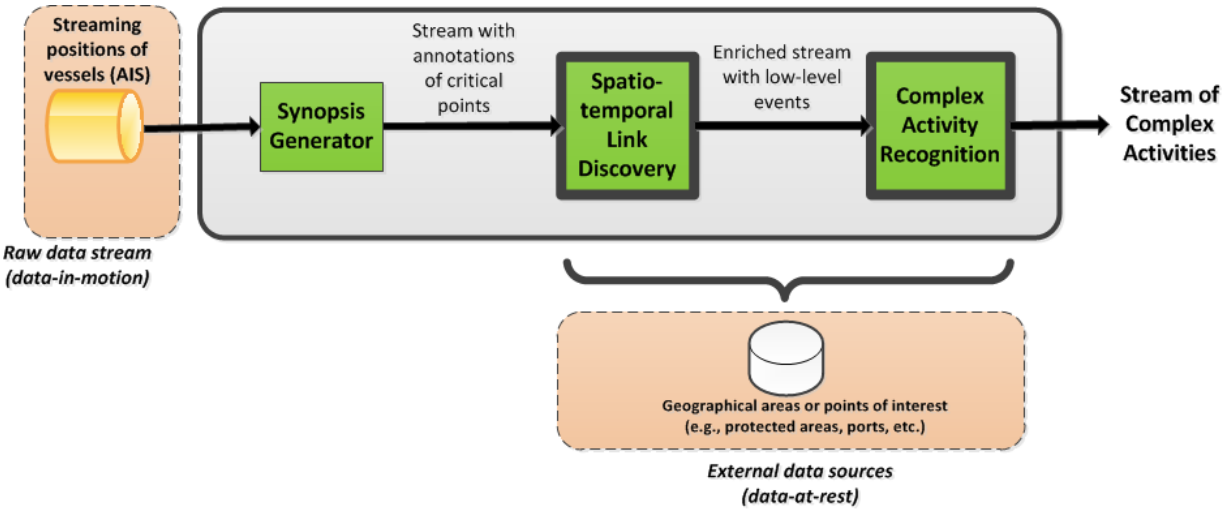


Fig. 1. The datAcron prototype architecture.

prototype for complex activity recognition is depicted in Figure 1. The main input is streaming positions of vessels, in the form of AIS messages. As this streaming data flows in the system, trajectory compression takes place, by annotating a subset of the original vessel positions as *critical points/events*. Then, critical points are linked with archival data in real-time, namely marine areas or points of interest, such as protected areas (Natura2000) or ports (World Port Index). Link discovery is performed at the spatio-temporal level, thus identifying those areas (or points) that are related to a given position of a vessel. Relations may be topological (e.g., a vessel is located *within* and area) or proximity-based (e.g., a vessel is *nearby* a port). In addition, vessel positions are linked to each other, by processing the streaming data only; this results in discovering proximity relations between moving vessels (e.g., a vessel is *nearby* another vessel) in an online fashion. Subsequently, a complex activity recognition module consumes the stream of spatial relations and critical points to recognize, in real-time, various types of suspicious, dangerous or illegeal vessel activity.

The *Synopses Generator* component (see Figure 1) provides algorithms for trajectory reconstruction and compression, by cleaning erroneous data and eliminating vessel positions that do not significantly affect the quality of trajectory representation. Thus, its main role is to compress a stream of positions of vessels, to a stream of 'critical points/events' i.e. 'speed change', 'heading change', 'communication gap', 'slow motion' and 'stopped', expressing trajectory synopsis. It has been shown that compressing trajectories this way, i.e. keeping only the critical events, can significantly reduce the computational cost of stream reasoning, and at the same time allow for accurate trajectory reconstruction [45]. The output of this component is thus a cleansed stream of vessel positions, with a subset of them being annotated as critical.

The *Spatio-temporal Link Discovery* component operates on the stream of vessel positions and derives in real-time a stream of spatial and spatio-temporal relations between vessels and static areas (or points) of interest, but also between vessels. To accomplish this online task, it capitalizes on efficient spatial index structures and applies optimized algorithms designed specifically for the purpose of link discovery. In the case of linking streaming with archival data, the spatial index is built on static data describing areas of interest or static points, and each vessel position from the stream is linked to areas or points by efficient index-based search. In the case of purely streaming data, the spatial index is built and maintained online on the vessel positions, and greatly improves the performance of spatio-temporal link discovery, by enabling effective filtering of vessel positions based on spatial distance. In this way, a stream of spatial relations is derived and provided as input for the recognition of complex activities.

The *Complex Activity Recognition* component is responsible for matching patterns of suspicious and potentially illegal activity on the stream of spatial relations and critical points. This component is based on the 'Event Calculus for Run-Time reasoning' (RTEC) [6]. RTEC includes various optimization techniques for efficient pattern matching. A form of caching stores the results of sub-computations in the computer memory to avoid unnecessary re-computations. A set of interval manipulation constructs simplify patterns and improve reasoning efficiency. Moreover, a 'windowing' mechanism makes RTEC independent of the data stream size.

The innovative feature of the proposed architecture is the integration of an optimized component for real-time discovery of spatial relations with an activity recognition engine. In the following, we delve into the technical details of spatio-temporal link discovery (Section 3) and complex activity recognition (Section 4) for maritime surveillance.

3. Spatio-temporal Link Discovery

We begin this section by providing the necessary definitions and problem setting for spatio-temporal link discovery. Then, we focus on: (a) topological and proximity relations between the positions of moving entities and static geographical areas of interest (Section 3.2), and (b) proximity relations between the positions of moving entities (Section 3.3). The former is a case of streaming to static link discovery, while the latter is a case of streaming to streaming link discovery.

3.1. Notation and Definitions

Let $p = (p.x, p.y, p.t)$ denote a spatio-temporal point corresponding to a vessel's V position $(p.x, p.y)$ at a given time $p.t$, and $\mathcal{A}$ a dataset that consists of geographical areas represented as polygons. A polygon $A \in \mathcal{A}$ is represented as a set of points a_i, i.e., $A = \{a_1, a_2, \ldots, a_n\}$, and we write $a_i \in A$ to denote that a_i is included in the representation of A.

Further, let $d(p, p')$ denote the distance between the spatial positions $(p.x, p.y)$ and $(p'.x, p'.y)$ of two vessels, whereas $t(p, p')$ stands for their temporal difference. Without loss of generality, we employ the Euclidean distance function $d(p, p') = \sqrt{(p.x - p'.x)^2 + (p.y - p'.y)^2}$ to quantify the spatial distance of two points, whereas $t(p, p') = |p.t - p'.t|$. Other distance functions, such as the Haversine distance, are applicable. Also, note that for small distances the Euclidean distance serves as an approximation of the Haversine distance, and the relative error is small. We abuse notation slightly by denoting $d(p, A)$ the distance between a point p and an area A (polygon), which is defined as: $d(p, A) = \min_{a_i \in A} d(p, a_i)$.

Below are the formal descriptions of the spatial relations discovered by the stLD component.

Definition 1. *withinArea*(V, A)*: Given a spatio-temporal position p of a vessel V and an area $A \in \mathcal{A}$,* *withinArea*(V, A) *is true, if p is enclosed in A.*

Definition 2. *nearbyArea*(V, A, θ)*: Given a spatio-temporal point p of a vessel V, an area $A \in \mathcal{A}$, and a distance threshold θ,* *nearbyArea*(V, A, θ) *is true, if $d(p, A) \leqslant \theta$.*

Definition 3. *nearby*$(V_1.p, V_2.p', \theta, \tau)$*: Given two spatio-temporal points p and p' of vessels V_1 and V_2 respectively, a distance threshold θ, and a temporal threshold τ,* *nearby*$(V_1.p, V_2.p', \theta, \tau)$ *is true, if $d(p, p') \leqslant \theta$ and $t(p, p') \leqslant \tau$.*

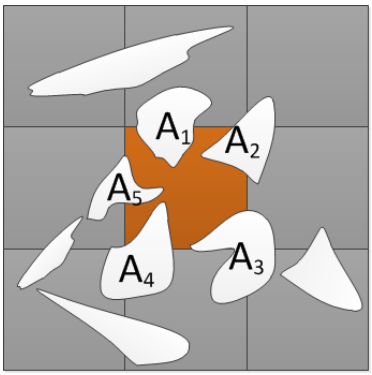


Fig. 2. Illustration of the mask of a grid cell that overlaps five areas $\{A_1, \ldots, A_5\}$.

3.2. Stream to Static LD

3.2.1. Discovery of Topological Relations: Within

Given a *target* dataset T, a *source* dataset S, and a relation r, the goal of link discovery is to detect the pairs $(\sigma, \tau) \subseteq S \times T$, where $\sigma \in S$ and $\tau \in T$, s.t. (σ, τ) satisfies r. Consider the case of

relation 'within' between a moving entity p, whose current spatio-temporal position is streamed into our system, and a set of static geographical areas of interest $\mathcal{A} = \{A_1, \ldots, A_n\}$. The result of link discovery is to identify all areas $A_i \in \mathcal{A}$ which enclose the spatial position $(p.x, p.y)$ of the moving entity p.

A brute force link discovery algorithm would have to perform the geometrical test between p and all areas in $\mathcal{A}$, thereby producing the result using $O(n)$ comparisons, where $n = |\mathcal{A}|$. To avoid this prohibitively expensive cost, the state-of-the-art LD methods (e.g., [40, 48]) employ a *space tiling* approach, which essentially partitions the space in cells and assigns each area to its overlapping cells. Then, to compute the relation "within", the position of a vessel is compared only against those (say c) areas overlapping the cell, thus resulting in $O(c)$ comparisons, and typically $c \ll n$. Practically, this method belongs to the filter-and-refine paradigm, where in the filtering step only a small set of c (out of n) candidate areas are identified, and in the refinement step the candidates are examined one-by-one to discover the subset of areas actually enclosing the vessel. Since the refinement step is the most costly processing part, any efficient link discovery algorithm should minimize the number of candidates.

In several application domains of spatial link discovery , such as the maritime domain, quite often grid cells contain a significant amount of "empty space", namely the space of the cell that overlaps with no areas. Our observation is that positions of moving vessels located in the empty space of a cell induce high processing cost, as they must be compared to all areas in the cell in vain, since no "within" links can be produced. Motivated by this observation, we propose a technique to explicitly represent the empty space within cells as yet another area. Thus, for each grid cell, we construct an artificial area called *mask*, which is defined as the difference between the cell and the union of areas overlapping with the cell, i.e. $mask = c - (c \cap \bigcup(A)_i)$. Figure 2 shows an example of the mask of a cell; the middle cell overlaps with areas $\{A_1, \ldots, A_5\}$, and the mask of the cell is the area represented in orange color.

Having the mask of a cell as yet another area, we can devise an efficient algorithm for link discovery that eagerly avoids comparisons to areas for positions located in the empty space. In practice, after we identify the enclosing cell of a position of a vessel, we first compare it to the mask of the cell, to check if it is enclosed in the empty space. If this single comparison returns true, we stop processing this position, thereby saving c comparisons (c denotes the average number of areas in a cell). For the typical case where a cell contains several areas, this technique can save significant computational cost, as will be demonstrated in the empirical analysis presented in Section 5.

Algorithm 1 Spatio-temporal LD algorithm for relation *"within"* using mask.

```
1:  Input: Grid cells C = {c_1, ..., c_m}, Areas A = {A_1, ..., A_n}, vessel position p (p.x, p.y)
2:  Output: Subset of areas A^w ⊆ A that enclose the vessel position (p.x, p.y) of p
3:  Requires: Grid has been constructed and areas have been assigned to overlapping cells
4:  A^w ← ∅
5:  locate cell c_i that encloses p
6:  if within(p, mask(c_i)) then
7:      return A^w
8:  else
9:      for each A_j ∈ c_i do
10:         if within(p, A_j) then
11:             A^w ← A^w ∪ A_j
12: return A^w
```

Algorithm 1 presents the pseudo-code for discovering a 'within' link between the position p of a vessel and the areas that enclose it. As a prerequisite, the grid has already been constructed and the static areas in the dataset R have been assigned to cells. This is essentially a pre-processing step. In the first step, the cell c_i that encloses p is determined (line 5). This operation is performed in constant time $O(1)$ in the case of equi-grids. Then, we check if p is contained in the mask $mask(c_i)$ of cell c_i (line 6). If it is contained, no further processing is required, and the algorithm terminates returning the empty set. If it is not contained, then we check for containment against all areas A_j in cell c_i (line 9). For those areas A_j that contain p, we add them to the result set $\mathcal{A}^w$ (line 11) and eventually return them as result.

The lines 9–11 of Algorithm 1 are processed in parallel, i.e., each iteration in the for loop is carried out by a different thread ('worker'). The number of concurrent workers is usually a predefined constant w.r.t. system configuration, to allow uninter-

Algorithm 2 Spatio-temporal LD algorithm for relation *'nearby'* using mask.

1: **Input:** Grid cells $C = \{c_1, \ldots, c_m\}$, Areas $\mathcal{A} = \{A_1, \ldots, A_n\}$, vessel position p $(p.x, p.y)$, threshold θ
2: **Output:** Subset of areas $\mathcal{A}^n \subseteq \mathcal{A}$ within distance θ from the vessel position $(p.x, p.y)$ of p
3: **Requires:** Grid has been constructed and areas have been assigned to overlapping cells
4: $\mathcal{A}^n \leftarrow \emptyset$
5: locate cell c_i that encloses p
6: **if** within(p, $mask^{\theta}(c_i)$) **then**
7: **return** $\mathcal{A}^n$
8: $\mathcal{P} \leftarrow \emptyset$
9: locate cells C' that overlap with circle at p with radius θ
10: **for** each $c_i \in C'$ **do**
11: **for** each $A_j \in c_i$ **do**
12: **if** $A_j \in \mathcal{P}$ **then**
13: **continue**
14: **if** $d(p, A_j) \leqslant \theta$ **then**
15: $\mathcal{A}^n \leftarrow \mathcal{A}^n \cup A_j$
16: $\mathcal{P} \leftarrow \mathcal{P} \cup A_j$
17: **return** $\mathcal{A}^n$

ruptible system operation (in our experiments we employ 8 workers). We have enabled multi-thread processing using a *pool of tasks*, populated with the refinement tasks of $within(p, A_j)$. As soon as a worker is available and the pool contains tasks, the next task is selected and assigned to the worker for processing.

Also, the algorithm is amenable to parallelization beyond the scope of a single machine for high velocity streams, by simply partitioning the stream of positions of vessels to the available processing nodes, each of which runs an instance of Algorithm 1 on an off-line constructed grid.

3.2.2. Discovery of Proximity Relations: Nearby

Interestingly, the above technique is applicable also for link discovery of a proximity relation, such as the 'nearby' relation, between vessel position p and a set of static areas $\mathcal{A}$. The 'nearby' relation is defined using a spatial threshold θ, and retrieves the subset of areas in $\mathcal{A}$ that are located at most at distance θ from p.

The technique of using the mask needs to be slightly adjusted to work in the case of relation 'nearby'. The main adjustment concerns the way the mask of each cell is computed. We expand each area A_i by θ and then the cell's mask is computed as previously, only using the expanded areas (A_i^{θ}) instead of the actual areas A_i. To differentiate this mask of a cell c_i from the one used in the previous algorithm, we denote it by $mask^{\theta}(c_i)$.

Algorithm 2 presents the pseudo-code for LD of relation 'nearby'. Notice that the grid is constructed exactly as before, using the original areas $\{A_i\}$. First, the cell c_i that encloses p is located (line 5). If $mask^{\theta}(c_i)$ contains p (line 6), then we can safely stop processing, since no area is nearby p. This is because $mask^{\theta}(c_i)$ has been constructed based on expanded areas. If this pruning is not successful, we need to examine all cells $c_i \in C'$ that overlap with a circle centered at p with radius θ, since they may contain results. We examine each area A_j in c_i (line 11), and if the distance of p to A_j is lower or equal to the threshold θ, then A_j is added to the result $\mathcal{A}^n$ (line 15). To avoid computing the distance of p to an area A_j multiple times (due to A_j assignment to multiple cells), we maintain the already examined areas in a set $(\mathcal{P})$. In summary, the algorithm avoids processing points that would safely produce no results, due to the use of the mask technique.

3.3. Stream to Stream LD

Streaming link discovery of 'nearby' relations between positions of moving vessels requires meticulous use of the available memory, since it is not feasible to store the complete data stream in memory. Our solution relies on the use of a grid data structure on the spatial domain, similarly to the case of Section 3.2. The grid is used to maintain the positions arriving in the stream. When a new vessel position p arrives in the stream, in the filtering step, we examine the cells that intersect with a circle centered at $(p.x, p.y)$ with radius θ, and retrieve all vessel positions p_i' that satisfy the spatial constraint, i.e., $d(p, p_i') \leqslant \theta$. Then, in the refinement step, we identify the subset of vessel positions $\{p_i'\}$ that in addition satisfy the temporal constraint, i.e., $t(p, p_i') \leqslant \tau$. This solution avoids the exhaustive comparison of p to all other positions that have arrived before p. Algorithm 3 is invoked for each incoming p and describes this solution.

However, in order to manage the available memory effectively, we need to find a suitable way to

Algorithm 3 Spatio-temporal LD algorithm for relation *'nearby'* between vessels.

1: **Input:** Grid cells $C = \{c_1, \ldots, c_m\}$, vessel position p $(p.x, p.y)$, thresholds θ, τ
2: **Output:** Set R of pairs of vessel positions (p, p') satisfying Definition 3
3: $R \leftarrow \emptyset$
4: locate cells C that overlap with circle at p with radius θ
5: **for** each $c_j \in C$ **do**
6: **for** each $p'_i \in c_j$ **do**
7: **if** $\mathrm{d}(p, p'_i) \leqslant \theta$ **then**
8: **if** $\mathrm{t}(p, p'_i) \leqslant \tau$ **then**
9: $R \leftarrow R \cup (p, p'_i)$
10: add p to grid C
11: **return** R

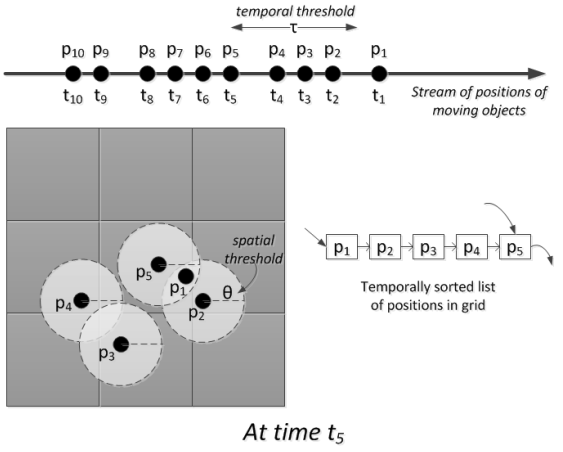


Fig. 3. Example of grid and bookkeeping structure for "nearby" LD between positions of moving vessels.

clean up the grid, since vessel positions that have arrived before more than τ time units will never produce a 'nearby' link to a new vessel position. A second reason to perform this cleaning operation is efficiency. If no cleaning were performed, then too many (old) vessel positions would be retrieved that satisfy the spatial constraint, but would be eliminated due to the temporal constraint, leading to wasteful processing.

A naive approach for cleaning would be to scan all grid cells (set at time t_{now}) and delete vessel positions p whose timestamp $p.t$ is more than τ units ago, i.e., $t_{now} - p.t > \tau$. However, this operation has linear complexity wrt. the number of positions in the grid, and incurs non-negligible overhead, as it must be performed during stream processing. To address this problem, we introduce an auxiliary, bookkeeping data structure that efficiently detects the vessel positions that need to be deleted. For this purpose, we maintain a list of pointers to the vessel positions in the grid. The list is in temporal order, since vessel positions are inserted in the list in the order in which they arrive in the stream. Then, cleaning can be performed efficiently, by traversing the list and deleting vessel positions (from the grid and list) until a position p is found with timestamp $t_{now} - p.t \leqslant \tau$. The maintenance cost of this list is small; insertions have $O(1)$ cost, whereas deletions have linear cost to the number of vessel positions that need to be deleted.

Figure 3 shows an example of the bookkeeping structure. Assuming that $t_{now} = t_5$, then the grid and the list contain the five vessel positions $p_1, \ldots, p_5$, as depicted. In case no cleaning is performed, then p_1 would be identified as candidate for 'nearby' to p_5, which would then be rejected due to the temporal threshold, since $p_5.t - p_1.t > \tau$. In case cleaning has already been performed, p_1 would have already been deleted from the grid, thus we would have avoided the cost of examining p_1.

An interesting issue that deserves further discussion is the frequency of performing the cleaning operation. An *eager* strategy could invoke the cleaning operation prior to processing every new vessel position that arrives in the stream. This would guarantee that all vessel positions satisfying the spatial distance threshold would be results, without the need to check the temporal threshold. However, the eager strategy would result in paying the overhead of grid cleanup for every new position. Another approach is to follow a *lazy* strategy, where the cleaning operation is invoked periodically, thus limiting the induced overhead at the expense of retrieving some candidate vessel positions that may be rejected by the temporal threshold. We evaluate the frequency of performing the grid cleanup in Section 5 presenting the empirical evaluation.

4. Complex Activity Recognition

The complex activity recognition (CAR) component of the datAcron prototype consumes the

Table 1

Complex Activity Recognition: Input events are presented above the two horizontal lines, while the output stream is presented below these lines. The input events above the single horizontal line are detected by stLD, while the remaining input events are emitted by the trajectory compression component. All input events, except $nearBy(V_1, V_2)$, are instantaneous, while all output activities are durative.

Event/Activity	**Description**
$withinArea(V, A)$	A vessel V is within some area A of interest
$nearbyArea(V, A)$	A vessel V is close to some area A of interest
$nearPorts(V)$	A vessel V is close to one or more ports
$nearby(V_1, V_2)$	Two vessels V_1 and V_2 are close to each other
$gapStart(V)$	A vessel V stopped sending position signals
$gapEnd(V)$	A vessel V resumed sending position signals
$slowMotionStart(V)$	A vessel started moving at a low speed
$slowMotionEnd(V)$	A vessel stopped moving at a low speed
$stopStart(V)$	A vessel started being idle
$stopEnd(V)$	A vessel stopped being idle
$changeInSpeedStart(V)$	A vessel started changing its speed
$changeInSpeedEnd(V)$	A vessel stopped changing its speed
$changeInHeading(V)$	A vessel changed its heading
$gap(V)$	A vessel V has a communication gap in the open sea
$stopped(V)$	A vessel V is stopped in the open sea
$slowMotion(V)$	A vessel V moves slowly in the open sea
$illegalFishing(V)$	A vessel V is potentially engaged in illegal fishing
$loitering(V)$	A vessel V is suspiciously idle
$rendezVous(V_1, V_2)$	Two vessels V_1 and V_2 are having a rendez-vous
$highSpeedIn(V, AreaType)$	A vessel V exceeds the speed limit of an area of $AreaType$

stream of spatial relations detected by stLD, as well as the critical points expressing compressed trajectories, to detect various types of suspicious, dangerous and illegal vessel activity (see Section 2). The upper part (above the two horizontal lines) of Table 1 presents the input to the CAR component. The output of this component, i.e. the list of recognized activities, specified in collaboration with the domain experts of datAcron, is presented in the lower part of Table 1.

4.1. Run-Time Event Calculus

The CAR component of datAcron is based on the 'Event Calculus for Run-Time reasoning' (RTEC) [6]. The Event Calculus is a logic programming language for representing and reasoning about events and their effects [29]. RTEC is a Prolog implementation of the Event Calculus, designed to compute continuous narrative assimilation queries for pattern matching on data streams. RTEC has a formal, declarative semantics—complex (vessel) activity formalisations are (locally) stratified logic programs [46]. Moreover, RTEC includes various optimisation techniques for efficient pattern matching, such as 'windowing', whereby all input events that took place prior to the current window are discarded/'forgotten'. Details about the reasoning algorithms of RTEC, including a complexity analysis, may be found in [6]. In what follows, we illustrate the use of RTEC for specifying maritime activities, focusing on the effects of stLD on CAR.

The time model in RTEC is linear and includes integer time-points. Variables start with an uppercase letter, while predicates and constants start with a lower-case letter. Where F is a *fluent*—a property that is allowed to have different values at different points in time—the term $F = V$ denotes that fluent F has value V. An *event description* in RTEC includes rules that define the event instances with the use of the happensAt predicate, the effects of events on fluents with the use of the initiatedAt and terminatedAt predicates, and the values of the fluents with the use of the holdsAt and holdsFor predicates. Table 2 summarizes the main predicates of RTEC. Complex activities are typically durative (see the lower part of Table 2), thus in

Table 2
Main predicates of RTEC.

Predicate	Meaning
happensAt(E, T)	Event E occurs at time T
holdsAt$(F = V, T)$	The value of fluent F is V at time T
holdsFor$(F = V, I)$	I is the list of the maximal intervals for which $F = V$ holds continuously
initiatedAt$(F = V, T)$	At time T a period of time for which $F = V$ is initiated
terminatedAt$(F = V, T)$	At time T a period of time for which $F = V$ is terminated
union_all(L, I)	I is the list of maximal intervals produced by the union of the lists of maximal intervals of list L
intersect_all(L, I)	I is the list of maximal intervals produced by the intersection of the lists of maximal intervals of list L
relative_complement_all(I', L, I)	I is the list of maximal intervals produced by the relative complement of the list of maximal intervals I' with respect to every list of maximal intervals of list L

CAR the task generally is to compute the maximal intervals for which a fluent expressing a complex activity has a particular value continuously. Below, we discuss the representation of fluents/complex maritime activities, and briefly present the way we compute their maximal intervals.

4.2. Maritime Pattern Representation

For a fluent F, $F = V$ holds at a particular time-point T if $F = V$ has been initiated by an event at some time-point earlier than T, and has not been terminated at some other time-point in the meantime. This is an implementation of the law of *inertia*. The time-points at which $F = V$ is initiated are computed with the use of initiatedAt rules, which have the following form (terminatedAt rules, expressing the ending time-points of a fluent-value pair, have a similar form):

$$\begin{aligned}&\mathsf{initiatedAt}(F = V,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(E,\ T),\\ &\quad \mathit{conditions}[T]\end{aligned}$$

The $\mathit{conditions}[T]$ set includes further constraints on time-point T, expressed as follows:

- a possibly empty set of happensAt predicates representing constraints on the occurrence of events;
- a possibly empty set of holdsAt predicates expressing constraints on fluents; and
- a possibly empty set of atemporal constraints.

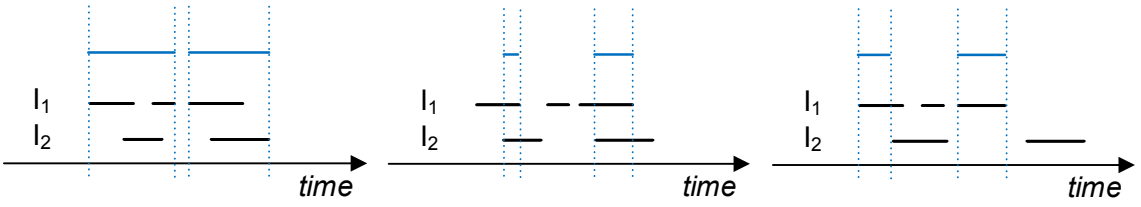
Consider the following example:

$$\begin{aligned}&\mathsf{initiatedAt}(\mathit{gap}(V) = \mathsf{true},\ T) \leftarrow\\ &\quad \mathsf{happensAt}(\mathit{gapStart}(V),\ T),\\ &\quad \mathsf{not\ happensAt}(\mathit{nearPorts}(V),\ T)\\ &\mathsf{terminatedAt}(\mathit{gap}(V) = \mathsf{true},\ T) \leftarrow\\ &\quad \mathsf{happensAt}(\mathit{gapEnd}(V),\ T)\end{aligned} \tag{1}$$

$\mathit{gap}(V)$ is a Boolean fluent denoting a communication gap for some vessel V, i.e. V stops transmitting AIS messages in the open sea. In some cases, the absence of AIS messages is suspicious and thus we need to record it. $\mathit{gapStart}(V)$ and $\mathit{gapEnd}(V)$ are instantaneous critical events indicating, respectively, the time-points in which a vessel V stops and resumes sending AIS messages (see Table 2). 'not' is negation by failure. $\mathit{nearPorts}(V)$ is an event emitted by stLD when vessel V is close to a port. Thus, rule-set (1) states that $\mathit{gap}(V) = \mathsf{true}$ is initiated if the trajectory compression component reports a *gapStart* event for V, and stLD does not report that V is close to a port. Furthermore, $\mathit{gap}(V) = \mathsf{true}$ is terminated when V resumes communications.

Given rule-set (1), RTEC computes the maximal intervals I for which $\mathit{gap}(V) = \mathsf{true}$ holds continuously, i.e. $\mathsf{holdsFor}(\mathit{gap}(V) = \mathsf{true},\ I)$, by finding all time-points T_s at which $\mathit{gap}(V) = \mathsf{true}$ is initiated, and then, for each T_s, computing the first time-point T_e after T_s at which $\mathit{gap}(V) = \mathsf{true}$ is terminated.

Most of the maritime patterns (defined in collaboration with domain experts) concern vessel activity close to, or within, some area of interest, such a Natura area. To simplify the structure



(a) Union. (b) Intersection. (c) Relative Complement.

Fig. 4. A visual illustration of the three interval manipulation constructs of RTEC. In this example, there are two input fluent streams, I_1 and I_2. The output of each interval manipulation construct is colored light blue.

of such patterns, we defined the auxiliary fluent *inArea* as follows:

$$\begin{aligned}
&\mathsf{initiatedAt}(\mathit{inArea}(V, \mathit{protected}) = \mathsf{true},\ T) \leftarrow \\
&\quad \mathsf{happensAt}(\mathit{withinArea}(V, A),\ T), \\
&\quad \mathit{protected}(A) \\
&\mathsf{terminatedAt}(\mathit{inArea}(V, \mathit{protected}) = \mathsf{true},\ T) \leftarrow \\
&\quad \mathsf{happensAt}(\mathit{nearbyArea}(V, A),\ T), \\
&\quad \mathit{protected}(A) \\
&\mathsf{terminatedAt}(\mathit{inArea}(V, \mathit{protected}) = \mathsf{true},\ T) \leftarrow \\
&\quad \mathsf{happensAt}(\mathsf{start}(\mathit{gap}(V) = \mathsf{true}),\ T)
\end{aligned} \tag{2}$$

$\mathit{inArea}(V, \mathit{AreaType})$ records the maximal intervals in which a vessel V is in an area of some type. $\mathit{withinArea}(V, A)$ and $\mathit{nearbyArea}(V, A)$ are spatial events emitted by stLD, stating, respectively, that V is within, or close to area A. $\mathit{protected}(A)$ is an atemporal predicate that stores protected areas. $\mathsf{start}(F = V)$ (respectively $\mathsf{end}(F = V)$) is a built-in RTEC event taking place at each starting (ending) point of each maximal interval for which $F = V$ holds continuously. According to rule-set (2), $\mathit{inArea}(V, \mathit{protected}) = \mathsf{true}$ is initiated when stLD detects a $\mathit{withinArea}(V, A)$ event for vessel V and protected area A. Furthermore, $\mathit{inArea}(V, \mathit{protected}) = \mathsf{true}$ is terminated when there is information that V has exited the area, i.e. when a $\mathit{nearbyArea}(V, A)$ event is emitted for the protected area A, or when V stops transmitting position signals (in this case we do not know the location of V).

Similar rule-sets define *inArea* for other types of area. Moreover, *inArea* is represented as a Boolean fluent since areas of different type may overlap, and thus a vessel may be within various types of area at the same time.

In previous work, RTEC performed both temporal and atemporal reasoning for CAR [45]. For example, RTEC performed spatial calculations to determine whether a vessel is within some area of interest or close to a port. In this work, all spatial relations necessary for CAR are computed by stLD. Thus, the evaluation of the initiatedAt rule of rule-set (2), for example, is reduced to checking for (*withinArea*) facts (in the input stream). This way, RTEC is used only for temporal reasoning for which it is optimized.

The absence of the *nearbyArea* relation in earlier work [45] did not allow us to compute the *intervals* during which a vessel is in some area. These intervals allow us to develop more accurate patterns of maritime activity—consider e.g. illegal fishing:

$$\begin{aligned}
&\mathsf{holdsFor}(\mathit{illegalFishing}(V) = \mathsf{true},\ I) \leftarrow \\
&\quad \mathit{fishing}(V), \\
&\quad \mathsf{holdsFor}(\mathit{slowMotion}(V) = \mathsf{true},\ I_1), \\
&\quad \mathsf{holdsFor}(\mathit{inArea}(V, \mathit{protected}) = \mathsf{true},\ I_2), \\
&\quad \mathsf{intersect_all}([I_1, I_2],\ I)
\end{aligned} \tag{3}$$

In addition to the domain-independent definition of holdsFor, an event description may include domain-specific holdsFor rules, such as rule (3), used to define the values of a fluent F, e.g. *illegalFishing*, in terms of a Boolean function on the values of other fluents. Domain-specific holdsFor rules make use of three interval manipulation constructs: union_all, intersect_all, and relative_complement_all. These are presented in Table 2 and illustrated in Figure 4. *fishing* is an atemporal predicate recording fishing vessels. According to the rule (3), the list I of maximal intervals during which a fishing vessel V is said to be engaged in illegal fishing is computed by determining the list I_1 of maximal intervals during which V is moving in slow motion in the open sea (these intervals are computed given the instantaneous *slowMotionStart* and *slowMotionEnd* criti-

cal events), the list I_2 of maximal intervals during which V is in a protected area, and then calculating the list I representing the intersections of the maximal intervals in I_1 and I_2.

The interval manipulation constructs of RTEC support the following type of definition: for all time-points T, $F=V$ holds at T if and only if some Boolean combination of fluent-value pairs holds at T. For a wide range of fluents, this is a much more concise definition than the traditional style of Event Calculus representation, i.e. identifying the various conditions under which the fluent is initiated and terminated so that maximal intervals can then be computed using the domain-independent holdsFor. For instance, the representation of *illegalFishing* by means of initiatedAt and terminatedAt predicates requires four rules.

Rule (3) is but one of the possible examples of suspicious or abnormal activity. For instance, we may want to flag communication gaps that start within, or even close to, an area of interest (say, a protected area). Similarly, we may want to flag vessels that are idle in some designated area (loitering). The list of activities recognized by the CAR component of the datAcron prototype is presented in the lower part of Table 1.

In addition to patterns for individual vessels, the knowledge base of the CAR component of datAcron includes formalizations of activities for pairs of vessels. Consider *rendezVous*:

$$\begin{aligned}&\mathsf{holdsFor}(\mathit{rendezVous}(V_1, V_2) = \mathsf{true},\ I) \leftarrow\\ &\quad \mathsf{holdsFor}(\mathit{nearby}(V_1, V_2) = \mathsf{true},\ I_1),\\ &\quad \mathsf{holdsFor}(\mathit{slowMotion}(V_1) = \mathsf{true},\ I_2),\\ &\quad \mathsf{holdsFor}(\mathit{stopped}(V_1) = \mathsf{true},\ I_3),\\ &\quad \mathsf{union\_all}([I_2, I_3],\ I_4),\\ &\quad \mathsf{holdsFor}(\mathit{slowMotion}(V_2) = \mathsf{true},\ I_5),\\ &\quad \mathsf{holdsFor}(\mathit{stopped}(V_2) = \mathsf{true},\ I_6),\\ &\quad \mathsf{union\_all}([I_5, I_6],\ I_7),\\ &\quad \mathsf{intersect\_all}([I_1, I_4, I_7],\ I)\end{aligned} \tag{4}$$

$\mathit{nearby}(V_1, V_2) = \mathsf{true}$ is a fluent denoting whether two vessels V_1 and V_2 are close to each other. This relation is computed by stLD. $\mathit{stopped}(V) = \mathsf{true}$ expresses that vessel V has stopped in the open sea. The intervals of this fluent-value pair are computed with the use of the instantaneous *stopStart* and *stopEnd* critical events (see Table 1). According to rule (4), vessels V_1 and V_2 are said to have a rendez-vous when they are close to each other, and each of them is stopped or moving in slow motion in the open sea.

In previous work, we approximated very crudely the $\mathit{nearby}(V_1, V_2)$ relation by checking whether V_1 and V_2 are located within the same cell of a grid. It is not surprising that this approximation produced several false negatives (vessels located close to each other but in different cells) as well as false positives (vessels located within the same cell but not close to each other). In this work, we can avoid these issues, due to the efficient spatial relation detection of stLD.

5. Experimental Evaluation

We present the results of our experimental study using real-life datasets from the maritime domain.

5.1. Experimental Setup

Platform. All experiments presented below were performed on a computer with 8 cores (Intel(R) Core(TM) i7-7700 CPU @ 3.6GHz) and 16 GB of RAM, running Ubuntu 16.04 LTS 64-bit with Linux Kernel 4.8.0-53-generic, Java 8, and SWI Prolog 7.2.3 for RTEC.

Datasets. Our main dataset is provided by our partners in datAcron and contains AIS kinematic messages from vessels sailing in the Atlantic Ocean around the port of Brest (Brittany, France) spanning between 1 October 2015 to 31 March 2016. AIS messages are compressed by the trajectory synopsis module (see Figure 1), keeping only critical events, regarding the start/end of low speed of a vessel, changes in speed/heading as well as notifications about gaps (a vessel has not emitted a message over a given time period), concerning $5,050$ vessels. Each critical point is accompanied by mobility information such as speed and heading. The dataset contains $4,765,647$ critical points. This dataset is consumed by stLD, which further produces $3,204,206$ spatial events that are additionally provided to CAR as input. Spatial events determine whether a vessel is in or near an area of interest, and if two vessels are close to each other in space and time.

In addition, we employ a set of spatial datasets describing areas and points of interest, as follows:

– Natura2000 regions: Natura 2000 is an index of protected regions for biodiversity in the European Union. It is set up to ensure the survival of Europe's most valuable species and habitats, indexing 3,522 polygons.
– Fishing areas: This dataset includes 5,076 polygons generated from raster images depicting the fishing intensity in European waters (as reported by European Union).
– The World Port Index (WPI) including a listing of 3,685 ports throughout the world, describing their location, characteristics, known facilities, and available services. Ports in this dataset are represented as points, and WPI is used for discovering proximity relations between vessel positions and ports.

Metrics. Our main metric is the achieved throughput of the stream reasoning system, assessed by delving into the individual performance of its two constituent components, stLD and CER. Other auxiliary metrics are additionally presented, in order to explain performance, such as processing time and recognition time for stLD and complex activity recognition respectively. We also measure the number of *unrewarding comparisons* of each LD configuration, i.e., the number of comparisons that did not result in a relation.

Parameters. In the experiments evaluating the performance of link discovery, we vary the input size and granularity of the grid (i.e., cell size). In the case of stream to static LD (i.e., 'within' and 'nearby' relations between moving objects and regions), we setup multiple equi-grid configurations varying in granularity {0.5°, 1°, 2°, 3°} (degrees in latitude/longitude), using the datasets of Natura2000 and fishing regions, totaling 8,599 regions. An equi-grid of 0.5° granularity means that the size of a cell is $0.5° \times 0.5°$. Notice that we construct the grid over the complete static datasets, as typically done in link discovery tasks. We evaluate the performance and scalability of stLD, using subsets of critical points datasets, i.e., {10K, 200K, 400K, 600K} entries. Also, we evaluate the gain obtained by using the mask technique (for the stream to static link discovery method of Section 3.2), and using the bookkeeping technique (for the stream to stream link discovery method of Section 3.3). Finally, we evaluate the effect of varying the number of cores on CAR, as well as the effect of increasing the window size.

Granularity	Average Number of Areas in Cell	Cells Constructed
0.5°	5.74	2,852
1°	13.54	858
2°	36.09	272
3°	64.14	147

Table 3
Distribution of areas from target dataset for each grid configuration.

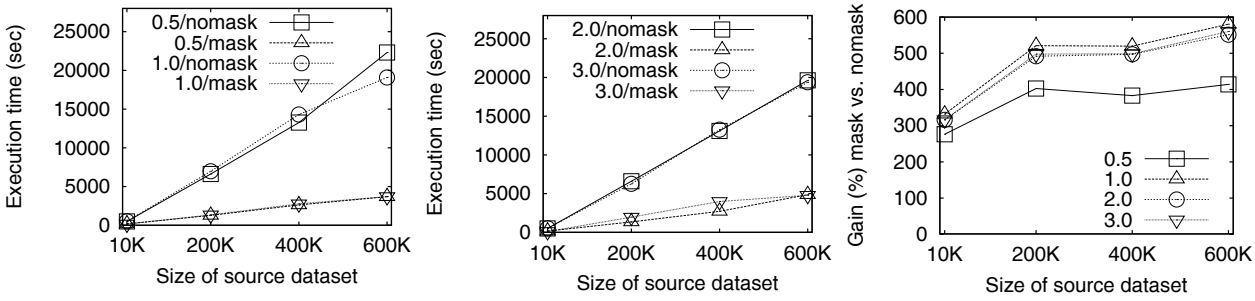
5.2. Link Discovery

The first objective of the experimental study is to evaluate the case of stream to static link discovery (i.e., relations within and nearby), in terms of achieved throughput, while also quantifying the gain offered by the mask technique. Then, we study the performance for the case of stream to stream link discovery (i.e., vessel 'nearby' vessel), focusing on measuring the throughput and the effect of bookkeeping.

5.2.1. Stream to Static LD

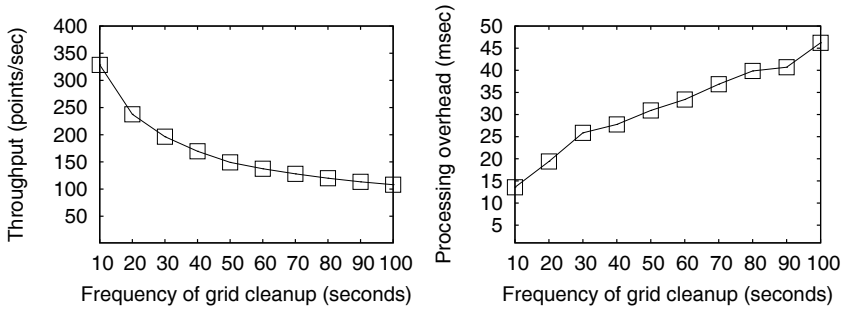
In this experiment, we refer to the areas organized in a grid as *target* dataset, whereas *source* dataset refers to the dataset of streaming vessel positions. We use as *target* dataset the regions of Natura2000 and fishing areas, totaling 8,599 areas. This static data is organized off-line in main-memory using grid configurations of varying granularity for the complete geographical space covered by the areas. For the streaming *source* dataset, we employ subsets of the critical points dataset of different sizes, in order to evaluate its effect. The distribution of areas for each grid configuration is shown in Table 3. The second column shows the average number of areas in a cell, while the third column indicates the number of non-empty cells that were constructed (i.e., hold at least one of the given areas).

Figure 5 shows the benefits of using the proposed technique with mask. Figure 5a depicts the total execution time required for processing a subset of critical points, whose size is indicated in the x-axis, using two grids with granularity 0.5° and 1.0° respectively. We observe that the mask technique achieves better execution time for all grid configurations and input sizes. Figure 5b shows both grid configurations of granularity 2.0° and 3.0°. In terms of throughput, the use of the mask



(a) Total execution time. (b) Total execution time. (c) Gain (%) of using mask.

Fig. 5. Spatio-temporal LD: Stream to static.



(a) Throughput. (b) Overhead of grid cleanup.

Fig. 6. Spatio-temporal LD: Stream to stream.

achieves to increase the throughput up to a factor of 5, compared to not using the mask. Figure 5c quantifies this gain, as the ratio of unrewarding comparisons, i.e., the number of unrewarding comparisons without mask, divided by the number of unrewarding comparisons with mask. For instance, the ratio of unrewarding comparisons for input size 200K and granularity 0.5° is 402.7%. This result clearly demonstrates the advantage of using the mask technique.

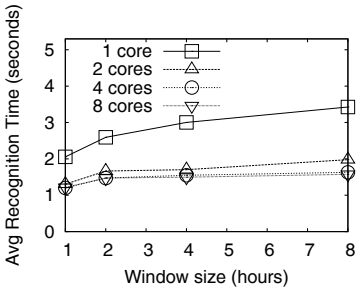
5.2.2. Stream to Stream LD

In the case of stream to stream link discovery (Figure 6), the main metric is the throughput (number of processed positions per second). We evaluate the effect of the grid cleaning technique on throughput, using a temporal threshold of 30 seconds. Figure 6a shows how throughput is affected when varying the frequency of grid cleanup from 10–100 sec. The chart shows that the more frequent cleanup is performed, the higher the achieved throughput. When the cleanup is delayed, the processing time spent on comparisons is increased, affecting the total execution time, and decreasing the average throughput.

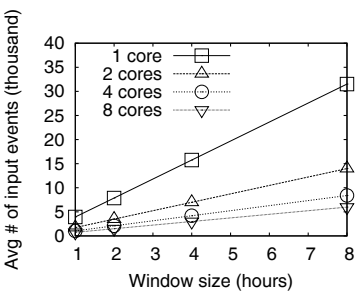
The next question to be answered concerns the overhead imposed by frequent grid cleanup. Figure 6b shows the average time spent for grid cleanup in milliseconds. We observe that the overhead is very small, and increases when the cleaning is not performed regularly. Recall that the bookkeeping structure is a list of spatio-temporal positions of vessels ordered by their temporal values, and the disposal of part of the list on each call allows us to avoid searching every cell of the grid for 'expired' entries. In summary, the cleanup operation has minimum overhead (in the order of milliseconds), which makes it applicable with high frequency, thereby increasing the achieved throughput.

5.3. Complex Activity Recognition

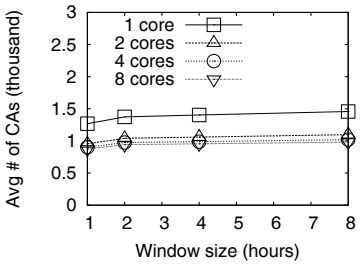
RTEC performs continuous query processing, computing the maximal intervals of fluents repre-



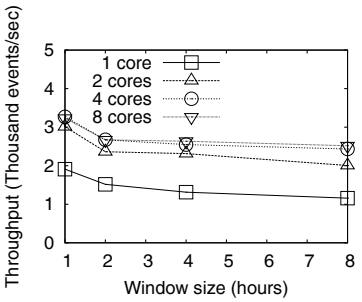
(a) Average recognition time.



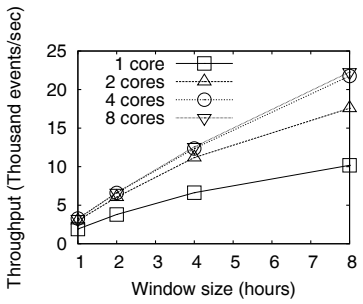
(b) Average number of input events.



(c) Average number of recognised activities.



(d) Throughput for slide step $\beta = 1$ hour.



(e) Throughput for slide step $\beta = \omega$.

Fig. 7. Complex activity recognition.

senting complex maritime activities. At each query time Q_i, the input events that fall within a specified sliding window ω are taken into consideration. All input events that took place before or at $Q_i - \omega$ are discarded/'forgotten'. This way, the cost of reasoning is independent of the data stream size. At Q_i, the complex activity intervals computed by RTEC are those that can be derived from input events that occurred in the interval $(Q_i - \omega, Q_i]$, as recorded at time Q_i. When the range ω is longer than the slide step β, it is possible that an input event occurs in the interval $(Q_i - \omega, Q_{i-1}]$ but arrives at RTEC only after Q_{i-1}; its effects are taken into account at query time Q_i. Window parameters for ω and slide step β are defined by the domain experts along with the maritime patterns, since they reflect the time horizon over which interesting phenomena may be detected. Usually, ω spans many minutes or even hours for capturing meaningful events across a vessel's route.

Figure 7 presents the experimental results for increasing window sizes ω: 1 hour to 8 hours. Figure 7a presents the average recognition time per window ω, while Figures 7b and 7c show, respectively, the average number of input events and recognized activities per window. Recall that Table 1 lists the types of input event and recognized activity. The slide step β is set to 1 hour. The empirical analysis shows that RTEC is capable of real-time maritime activity recognition—e.g. RTEC recognizes ($\approx$ 1,500) maritime activities given a window of 8 hours ($\approx$ 31,000 input events) in less than 3.5 sec, even when operating on a single processing core of the desktop computer. (The use of multiple cores will be discussed shortly.)

Figure 7d presents the throughput. As the window size ω increases, throughput decreases since the average recognition time per window increases (see Figure 7a). Figure 7e presents the throughput when the slide step is the same as the window size ($\beta = \omega$), i.e. windows are non-overlapping. In this case, the average recognition time per window increases, but only very slightly (due to the higher cost of 'forgetting' more past events), and thus not presented here. (The average numbers of input events and recognized activities per window are also similar to the case of $\beta = 1$ hour). Figure 7e shows that, as the window size ω in-

creases, throughput increases. When ω increases, the number of windows/query-times decreases (we have significantly less windows than the case of $\beta = 1$ hour), while the average recognition time per window increases only slightly.

We also run RTEC in parallel, by launching different instances of the engine, each operating on a different processing core. Each RTEC instance was responsible for activity recognition in a separate sub-area of the dataset, receiving input events only from vessels in that sub-area. To avoid false negatives on the borders of the sub-areas, we allowed for some overlap. Figure 7 presents the results when 2, 4 and 8 processing cores are used. Figure 7a presents the average recognition of the worst-performing RTEC instance, while Figures 7b and 7c show the average input events and recognised activities for that instance. Figures 7d and 7e present the throughput. As shown in Figure 7, the benefits of parallelization can be significant. A more refined segmentation of the dataset into subareas, optimizing load allocation, would have had more profound effects on performance.

6. Related Work

In earlier work, we presented a system for maritime monitoring, which employed RTEC for complex activity recognition [45]. In that work, RTEC performed spatial calculations to determine whether a vessel is close to a port, represented as a static point, or within an area of interest. Furthermore, it approximated crudely the *nearby* relation between vessels by checking whether a pair of vessels is located within the same cell of a grid. In this work, we placed emphasis on the detection of spatial relations and developed a separate component for highly efficient spatio-temporal link discovery. Moreover, RTEC had at its disposal additional spatial relations—whether a vessel is *nearby* some area—and a much more accurate account of proximity between vessels.

6.1. Spatio-temporal Link Discovery

Even though the topic of link discovery has attracted much interest and attention lately (see [37] for a recent survey), there is not much work on the challenging topic of spatio-temporal link discovery nor on link discovery over streaming datasets. Our work tackles explicitly these topics. In addition, our work in the context of datAcron is related to semantic integration and stream reasoning [17, 26], as we also employ an ontology [47] to express the discovered relations in RDF, and to spatio-temporal stream reasoning [25], but our focus is on the efficiency of spatio-temporal link discovery. Below, we provide an overview of link discovery frameworks and techniques.

Generic LD frameworks include LIMES [41] and SILK [28]. LIMES [41] is an LD framework for metric spaces that uses the triangular inequality in order to avoid processing all possible pairs of objects. For this purpose, it employs the concept of exemplars, which are used to represent areas in the multidimensional space, and tries to prune entire areas (and the respective enclosed entities) from consideration during link discovery. SILK [28] is an LD framework proposing a novel blocking method called MultiBlock, which uses a multidimensional index in which similar objects are located near each other. In each dimension the entities are indexed by a different property or different similarity measure. Then, the indices are combined together to form a multidimensional index, which is able to prune more entities by taking into account the combination of dimensions.

HR^3 [39] and HYPPO [38] address LD tasks when the property values that are to be compared are expressed in an affine space with a Minkowski distance. Both approaches are designed with main objectives to be efficient and lossless. In addition, HR^3 [39] comes with theoretical guarantees on reduction ratio, a metric that corresponds to the percentage of the Cartesian product of two datasets that was not explored before reporting the link discovery results. However, all aforementioned approaches for LD do not explicitly focus on spatio-temporal link discovery, nor do they tackle the streaming nature of data sources.

The spatial LD methods [40, 48] apply grid partitioning (a.k.a. space tiling) on the two sources A and B, in order to perform efficiently the *filtering step*, i.e., avoid comparing all entities in A to all entities in B. Then, in the *refinement step*, different optimizations are employed in order to minimize the number of computations necessary to produce the correct result set. RADON [48] is the most recent approach for discovering topological relations between datasets of areas, and can discover efficiently multiple relations using space tiling. One of

its main techniques for efficiency relies on the use of caching to avoid recomputing distances. However this imposes non-negligible requirements for main memory, especially for large datasets. ORCHID [40] studies the problem of discovering all pairs of polygons, such that their Hausdorff distance (practically Max-Min distance) is below a given threshold. It also employs space tiling to improve the filtering step. Also, it employs bounding circles as approximations of polygons together with applying the triangular inequality and already computed distances to avoid computing new distances, thus pruning regions without distance computations. Smeros et al. [50] study link discovery on spatio-temporal RDF data. The authors study several topological relations that are defined on polygons. The topological relations do not take into account proximity nor distance of the polygon, and several of those are meaningful only when both datasets store polygons. The algorithm provided creates an equi-grid, and filters out cells that contain polygons that cannot satisfy the relation.

In summary, most of the above papers target spatial data, rather than spatio-temporal data, which is the goal of our work. Moreover, the type of relations supported is very restricted, mainly focusing on topological relations. Instead, in this paper, proximity relations (e.g., 'nearby') are targeted, and (perhaps most importantly) in a streaming context.

6.2. Complex Activity Recognition

Various languages and systems have been proposed in the literature for complex event/activity recognition—see [4, 7, 16] for three surveys. RTEC has a formal, declarative semantics—activity patterns in RTEC are (locally) stratified logic programs. In contrast, most complex event processing languages, including [1], several event query languages and data stream processing languages, such as ESL [8] which extends CQL [3], and most commercial production rule systems, rely on an informal and/or procedural semantics [15, 22, 44]. The lack of theoretical underpinnings from the viewpoint of stream reasoning [18, 34] has also been observed [10, 11, 51].

RTEC explicitly represents complex activity intervals (unlike e.g. [11, 15, 21, 32]) and thus avoids the related logical problems (see [42] for a discussion of these problems). Furthermore, RTEC supports out-of-order event streams—in contrast to e.g. [15, 19, 20, 24, 31]—whereby the intervals of previously recognised activities may be (partially) retracted, or extended, due to delayed input events [6]. RTEC also supports reasoning over background knowledge, which is important for maritime surveillance [23, 52]. On the other hand, several approaches to activity recognition, such as [3, 9, 15, 21, 30, 33], lack the ability of combining pattern matching with background knowledge reasoning [2].

Maritime activities form hierarchies, in the sense that the formulation of one activity is used to define other, higher-level activities. We defined potentially illegal fishing, for example, in terms of slow motion (recall rule (3)). In contrast to many state-of-the-art recognition systems, such as the widely used Esper engine[3] and SASE[4], RTEC can naturally express hierarchical knowledge by means of well-structured specifications, and consequently employ caching techniques to avoid unnecessary re-computations [6].

As an Event Calculus implementation, RTEC has built-in axioms for representing complex temporal phenomena, including the formalization of inertia, which facilitate considerably the development of succinct activity patterns, and therefore code maintenance. This is a key difference to related approaches [2, 10, 11, 49]. Concerning the Event Calculus literature, an important feature of RTEC is that it includes a windowing technique. No other Event Calculus system (including [5, 12, 13, 35, 36, 43]) system 'forgets' or represents concisely the data stream history.

7. Summary & Future Work

We presented a stream reasoning system for maritime monitoring, which supports complex activity recognition assisted by real-time spatio-temporal discovery of relations between moving vessels and areas of interest. Compared to earlier work, the proposed system optimizes the computation of spatial relations, leading to improved system performance. Our experimental evaluation on real maritime datasets demonstrates the efficiency of the proposed prototype.

[3] http://www.espertech.com/esper/
[4] http://sase.cs.umass.edu/

There are several directions for further work. With respect to link discovery, we intend to identify more complex spatio-temporal relations in real-time. Moreover, refined data partitioning schemes that provide load balancing and fair work allocation to workers deserve more attention, aiming at improving performance and scalability even further. Concerning activity recognition, we are implementing RTEC in the Scala programming language to pave the way for the use of frameworks with built-in support for distributed reasoning, such as the Akka actors toolkit[5].

References

[1] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Efficient pattern matching over event streams. In *Proceedings of SIGMOD*, 2008.

[2] D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic. Real-time complex event recognition and reasoning. *Applied Artificial Intelligence*, 26(1–2):6–57, 2012.

[3] A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: semantic foundations and query execution. *The VLDB Journal*, 15(2):121–142, 2006.

[4] A. Artikis, A. Margara, M. Ugarte, S. Vansummeren, and M. Weidlich. Complex event recognition languages: Tutorial. In *Proceedings of DEBS*, pages 7–10, 2017.

[5] A. Artikis and M. J. Sergot. Executable specification of open multi-agent systems. *Logic Journal of the IGPL*, 18(1):31–65, 2010.

[6] A. Artikis, M. J. Sergot, and G. Paliouras. An event calculus for event recognition. *IEEE Trans. Knowl. Data Eng.*, 27(4):895–908, 2015.

[7] A. Artikis, A. Skarlatidis, F. Portet, and G. Paliouras. Logic-based event recognition. *Knowledge Engineering Review*, 27(4):469–506, 2012.

[8] Y. Bai, H. Thakkar, H. Wang, C. Luo, and C. Zaniolo. A data stream language and system designed for power and extensibility. In *Proceedings of CIKM*, pages 337–346, 2006.

[9] R. S. Barga, J. Goldstein, M. H. Ali, and M. Hong. Consistent streaming through time: A vision for event stream processing. In *Proceedings of CIDR*, pages 363–374, 2007.

[10] H. R. Bazoobandi, H. Beck, and J. Urbani. Expressive stream reasoning with laser. *CoRR*, abs/1707.08876, 2017.

[11] H. Beck, M. Dao-Tran, and T. Eiter. LARS: A Logic-Based Framework for Analytic Reasoning over Streams. Technical Report INFSYS RR-1843-17-03, Institute of Information Systems, TU Vienna, October 2017.

[12] I. Cervesato and A. Montanari. A calculus of macro-events: Progress report. In *Proceedings of TIME*, pages 47–58, 2000.

[13] L. Chittaro and A. Montanari. Efficient temporal reasoning in the cached event calculus. *Computational Intelligence*, 12(3):359–382, 1996.

[14] C. Claramunt, C. Ray, E. Camossi, A. Jousselme, M. Hadzagic, G. L. Andrienko, N. V. Andrienko, Y. Theodoridis, G. A. Vouros, and L. Salmon. Maritime data integration and analysis: recent progress and research challenges. In *Proceedings of EDBT*, pages 192–197, 2017.

[15] G. Cugola and A. Margara. TESLA: a formally defined event specification language. In *Proceedings of DEBS*, pages 50–61, 2010.

[16] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3):15, 2012.

[17] D. de Leng and F. Heintz. Ontology-based introspection in support of stream reasoning. In *Proceedings of JOWO*, pages 1–8, 2015.

[18] D. Dell'Aglio, E. D. Valle, F. van Harmelen, and A. Bernstein. Stream reasoning: A survey and outlook. *Data Science*, 2017.

[19] N. Dindar, P. M. Fischer, M. Soner, and N. Tatbul. Efficiently correlating complex events over live and archived data streams. In *Proceedings of DEBS*, pages 243–254, 2011.

[20] L. Ding, S. Chen, E. A. Rundensteiner, J. Tatemura, W.-P. Hsiung, and K. Candan. Runtime semantic query optimization for event stream processing. In *Proceedings of ICDE*, pages 676–685, 2008.

[21] C. Dousson and P. L. Maigat. Chronicle recognition improvement using temporal focusing and hierarchisation. In *Proceedings of IJCAI*, pages 324–329, 2007.

[22] M. Eckert and F. Bry. Rule-based composite event queries: the language xchange^{eq} and its semantics. *Knowledge Information Systems*, 25(3):551–573, 2010.

[23] J. Garcia, J. Gomez-Romero, M. Patricio, J. Molina, and G. Rogova. On the representation and exploitation of context knowledge in a harbor surveillance scenario. In *Proceedings of FUSION*, pages 1–8, 2011.

[24] D. Gyllstrom, E. Wu, H.-J. Chae, Y. Diao, P. Stahlberg, and G. Anderson. SASE: Complex event processing over streams. In *Proceedings of CIDR*, 2007.

[25] F. Heintz and D. de Leng. Spatio-temporal stream reasoning with incomplete spatial information. In *Proceedings of ECAI*, pages 429–434, 2014.

[26] F. Heintz and Z. Dragisic. Semantic information integration for stream reasoning. In *Proceedings of FUSION*, pages 1454–1461, 2012.

[27] B. Idiri and A. Napoli. The automatic identification system of maritime accident risk using rule-based reasoning. In *Proceedings of SoSE*, pages 125–130, 2012.

[28] R. Isele, A. Jentzsch, and C. Bizer. Efficient multidimensional blocking for link discovery without losing recall. In *Proceedings of WebDB*, 2011.

[29] R. A. Kowalski and M. J. Sergot. A logic-based calculus of events. *New Generation Comput.*, 4(1):67–95, 1986.

[5] https://akka.io/

[30] J. Krämer and B. Seeger. Semantics and implementation of continuous sliding window queries over data streams. *ACM Transactions on Database Systems*, 34(1):1–49, 2009.

[31] M. Li, M. Mani, E. A. Rundensteiner, and T. Lin. Complex event pattern detection over streams with interval-based temporal semantics. In *Proceedings of DEBS*, pages 291–302, 2011.

[32] K. Mahbub, G. Spanoudakis, and A. Zisman. A monitoring approach for runtime service discovery. *Automated Software Engineering*, 18(2):117–161, 2011.

[33] Y. Mei and S. Madden. Zstream: a cost-based query processor for adaptively detecting composite events. In *Proceedings of SIGMOD*, 2009.

[34] A. Mileo, M. Dao-Tran, T. Eiter, and M. Fink. Stream reasoning. In *Encyclopedia of Database Systems*. Springer Science+Business Media, 2017.

[35] R. Miller and M. Shanahan. Some alternative formulations of the event calculus. In *Computational Logic: Logic Programming and Beyond*, LNAI 2408, pages 452–490. 2002.

[36] M. Montali, F. M. Maggi, F. Chesani, P. Mello, and W. M. P. van der Aalst. Monitoring business constraints with the event calculus. *ACM TIST*, 5(1):17:1–17:30, 2013.

[37] M. Nentwig, M. Hartung, A. N. Ngomo, and E. Rahm. A survey of current link discovery frameworks. *Semantic Web*, 8(3):419–436, 2017.

[38] A. N. Ngomo. A time-efficient hybrid approach to link discovery. In *Proceedings of OM*, 2011.

[39] A. N. Ngomo. Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures. In *Proceedings of ISWC*, pages 378–393, 2012.

[40] A. N. Ngomo. ORCHID - reduction-ratio-optimal computation of geo-spatial distances for link discovery. In *Proceedings of ISWC*, pages 395–410, 2013.

[41] A. N. Ngomo and S. Auer. LIMES - A time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of IJCAI*, pages 2312–2317, 2011.

[42] A. Paschke. ECA-RuleML: An approach combining ECA rules with temporal interval-based KR event/action logics and transactional update logics. Technical Report 11, Technische Universität München, 2005.

[43] A. Paschke and M. Bichler. Knowledge representation concepts for automated SLA management. *Decision Support Systems*, 46(1), 2008.

[44] A. Paschke and A. Kozlenkov. Rule-based event processing and reaction rules. In *Proceedings of RuleML*, LNCS 5858. 2009.

[45] K. Patroumpas, E. Alevizos, A. Artikis, M. Vodas, N. Pelekis, and Y. Theodoridis. Online event recognition from moving vessel trajectories. *GeoInformatica*, 21(2):389–427, 2017.

[46] T. Przymusinski. On the declarative semantics of stratified deductive databases and logic programs. In *Foundations of Deductive Databases and Logic Programming*. Morgan, 1987.

[47] G. Santipantakis, G. Vouros, C. Doulkeridis, A. Vlachou, G. Andrienko, N. Andrienko, G. Fuchs, J. M. C. Garcia, and M. G. Martinez. Specification of semantic trajectories supporting data transformations for analytics: The datAcron ontology. In *Proceedings of Semantics*, 2017.

[48] M. A. Sherif, K. Dreßler, P. Smeros, and A. N. Ngomo. Radon - rapid discovery of topological relations. In *Proceedings of AAAI*, pages 175–181, 2017.

[49] V. Shet, J. Neumann, V. Ramesh, and L. Davis. Bilattice-based logical reasoning for human detection. In *Proceedings of CVPR*, 2007.

[50] P. Smeros and M. Koubarakis. Discovering spatial and temporal links among RDF data. In *Proceedings of LDOW*, 2016.

[51] E. D. Valle, S. Ceri, F. van Harmelen, and D. Fensel. It's a streaming world! reasoning upon rapidly changing information. *IEEE Intelligent Systems*, 24(6):83–89, 2009.

[52] J. van Laere and M. Nilsson. Evaluation of a workshop to capture knowledge from subject matter experts in maritime surveillance. In *Proceedings of FUSION*, pages 171–178, 2009.