Geoinformatica (2017) 21:389–427
DOI 10.1007/s10707-016-0266-x

Online event recognition from moving vessel trajectories

Kostas Patroumpas[1,2] · Elias Alevizos[3] · Alexander Artikis[3,4] · Marios Vodas[2] · Nikos Pelekis[5] · Yannis Theodoridis[2]

Received: 27 January 2016 / Revised: 13 June 2016 /
Accepted 3 August 2016 / Published online: 16 August 2016

Abstract We present a system for online monitoring of maritime activity over streaming positions from numerous vessels sailing at sea. The system employs an online tracking module for detecting important changes in the evolving trajectory of each vessel across time, and thus can incrementally retain concise, yet reliable summaries of its recent movement. In addition, thanks to its complex event recognition module, this system can also offer instant notification to marine authorities regarding emergency situations, such as suspicious moves in protected zones, or package picking at open sea. Not only did our extensive tests validate

✉ Elias Alevizos
alevizos.elias@iit.demokritos.gr

Kostas Patroumpas
kpatro@dblab.ece.ntua.gr

Alexander Artikis
a.artikis@unipi.gr

Marios Vodas
mvodas@unipi.gr

Nikos Pelekis
npelekis@unipi.gr

Yannis Theodoridis
ytheod@unipi.gr

1 School of Electrical, Computer Engineering, National Technical University of Athens, Athens, Greece

2 Department of Informatics, University of Piraeus, Piraeus, Greece

3 Institute of Informatics, Telecommunications, NCSR Demokritos, Athens, Greece

4 Department of Maritime Studies, University of Piraeus, Piraeus, Greece

5 Department of Statistics, Insurance Science, University of Piraeus, Piraeus, Greece

the performance, efficiency, and robustness of the system against scalable volumes of real-world and synthetically enlarged datasets, but its deployment against online feeds from vessels has also confirmed its capabilities for effective, real-time maritime surveillance.

Keywords AIS · Event recognition · Geostreaming · Moving objects · Trajectorys

1 Introduction

Maritime surveillance systems have been attracting considerable attention for economic as well as environmental reasons [18, 39, 40]. For instance, accidents at sea may cause ecological disasters (e.g., oil spill) and shipping companies may be fined to pay billions of dollars. In the past decade, monitoring vessel activity has emerged as a precious tool for preventing such risks, thanks to the Automatic Identification System (AIS)[1]. By integrating a VHF transceiver with positioning and navigational devices (e.g., GPS, gyrocompass), AIS can be used to track vessels at sea in real-time through data exchange with other ships nearby, coastal stations, or even satellites. The initial purpose of AIS was to prevent collisions; yet, the amount and precision of the collected data and its real-time availability can be used by a broader spectrum of maritime monitoring applications. International regulations require AIS to be aboard cargo ships of at least 300 gross tonnage, as well as all passenger ships, regardless of size. Considering that AIS data is continuously emitted from over 580,000 vessels worldwide[2], maritime surveillance systems certainly demand capabilities of highly scalable, continuous, spatiotemporal processing over transient data streams.

To address this requirement, we have been developing a maritime surveillance system that consists of two main components. A *trajectory detection* component accepts a positional stream of AIS messages and tracks major changes along each vessel's movement. Given that vessels normally follow planned routes (except for accidents, storms, etc.), this process can instantly identify *"critical points"* along each trajectory, such as a stop, a turn, or slow motion. Therefore, we may discard redundant locations along a "normal" course, and approximately reconstruct each vessel's trajectory from such a synopsis consisting of critical points only. But, apart from archiving or displaying it on maps, this derived stream of critical points is mostly useful in recognizing complex maritime phenomena that involve interaction among vessels or spatiotemporal relationships between vessels and geographical areas of interest. This is handled by our *complex event recognition* component, which can efficiently detect suspicious or potentially dangerous situations, such as fast approaching vessels or package picking at open sea, and accordingly issue alert notifications to marine authorities. In this system, we have set the following major objectives:

– *Timeliness.* Event detection must be carried out in real-time. Critical points concerning trajectories of vessels and alerts regarding suspicious maritime situations must be issued within seconds in order to enable immediate action if necessary.
– *Compression.* We wish to extract trajectory synopses per vessel from the incoming AIS positions retaining salient movement features only. This online data reduction can yield huge space savings (empirically, up to 98 % compared with the raw data), but it is also advantageous for effective complex event recognition.

[1] http://www.imo.org/OurWork/Safety/Navigation/Pages/AIS.aspx
[2] https://www.vesselfinder.com

- *Quality.* Such compressed representations should be reliable enough in reconstructing trajectories with very small deviations (i.e., tolerable approximation error) from original traces, also coping with inherent imperfections in AIS streams.
- *Scalability.* The system must be able to manage frequently updated, streaming AIS positions from large fleets of vessels moving over a large area. Not only do we verify that a centralized system can handle scalable volumes of incoming data at varying arrival rates, but we also demonstrate that both components are parallelizable, thus offering even higher gains in efficiency.

Several platforms and monitoring applications have been proposed for managing and analyzing data streams. For instance, the system in [11] focuses on recency-probing pattern queries against both live and archived streams. The UpStream platform [29] offers low latency response to continuous queries over massively updated data, but it lacks support for the specific demands of trajectory detection. Besides, automating ingestion of streaming data feeds from various sources into a data warehouse is also important [17], but does not pay attention to complex event recognition. Our particular interest is on *geostreaming* data [21] from sailing vessels acquired continuously over time, which must be processed on-the-fly in order to recognize important phenomena regarding their movement and their interaction with the maritime environment. In [37] an approach for anomaly detection and classification of vessel interactions is presented. Patterns of interest are expressed as left-to-right Hidden Markov Models and classified using Support Vector Machines, also taking into account contextual information via first-order logic rules. However, this work focuses more on predictive accuracy rather than real-time performance in a streaming scenario. To the best of our knowledge, no streaming framework has been specifically tailored for maritime surveillance over fluctuating, noisy, intermittent, geostreaming AIS messages from large fleets, as the one we present in this work.

This paper is an extended and revised version of previous works presented in [2, 33], and developed in the context of the AMINESS project[3]. It now offers heuristics for coping with noisy situations, improved algorithms for better capturing important events related to vessel mobility and interaction, as well as a more thorough empirical validation. In particular, our contributions are:

- We introduce single-pass heuristics to drastically reduce noisy AIS positions, much to the benefit of the resulting trajectory synopses (in size and quality).
- We provide a detailed account of online spatiotemporal filters that can detect important changes in each vessel's mobility and also incrementally maintain succinct, reliable representations of their evolving trajectories.
- We analyze in depth several rules and conditions for efficiently recognizing complex maritime events, which may also involve topological relationships between vessels and geographical zones of interest.
- We empirically validate our methodology in terms of performance and quality of results against a large AIS dataset of real vessel traces. Moreover, we extensively investigate the robustness, efficiency, and timeliness of the system against scalable volumes of synthetically enlarged datasets, as well as its capacity to recognize complex events related to sensitive environmental zones.
- Finally, we outline a deployment of the system in a real-world scenario, against streaming AIS data feeds that are being collected across the Aegean Sea.

[3]http://www.aminess.eu/

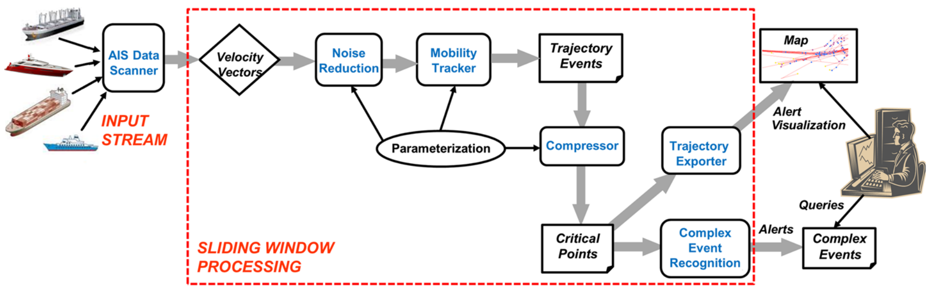


Fig. 1 Processing flow of streaming AIS data for online maritime surveillance

The remainder of the paper is organized as follows. In Section 2, we present the architecture of the proposed maritime surveillance system. Sections 3 and 4 respectively present the two main components for online trajectory detection and complex event recognition, respectively. Experimental results are reported in Section 5. In Section 6, we discuss a deployment of this system against real-time AIS data. Finally, in Section 7 we summarize our approach and outline directions for future work.

2 System architecture

Next, we outline the processing flow of the proposed maritime surveillance system. As illustrated in Fig. 1, this system consumes a geospatial stream of AIS tracking messages from vessels, it continuously detects important features that characterize their movement and recognizes complex events such as suspicious vessel activity. Table 1 presents the types of events that can be monitored by the proposed system. This list covers important events that

Table 1 Taxonomy of monitored events over moving vessels at sea

Trajectory Movement Events (ME)	*Instantaneous*	Pause	
		Speed change	
		Turn	
	Long – lasting	Gap	
		Stop	
		Slow motion	
		Smooth turn	
Complex Maritime Events (CE)	*Instantaneous*	*Single – vessel*	Illegal s hipping
			Fast ap proach
		Dual – vessel	Package picking
		Area – based	Suspicious area
	Long – lasting	*Single – vessel*	Suspicious vessel delay
		Dual – vessel	Vessel rendezvous

characterize vessel movement according to domain experts (our partners in the AMINESS project).

In order to meet the real-time requirements of the geostreaming paradigm [21], this online process necessitates the use of a *sliding window* [23, 34]. Typically, such a window abstracts the time period of interest by focusing only on phenomena that occurred in a recent *range* ω (e.g., positions received during past 10 minutes). This window slides forward to keep in pace with newly arrived stream tuples, so it gets refreshed at a specific *slide step* every β units (e.g., each minute). For instance, an aggregate query could report at every minute the distance traveled by a ship over the past 10 minutes. Typically, $\beta < \omega$, hence successive window instantiations may share positional tuples over their partially overlapping ranges across time.

As input, we consider particular AIS messages (specifically, of types 1, 2, 3, 18, or 19 according to AIS regulations) and extract position updates. Each message specifies the *MMSI* (Maritime Mobile Service Identity) of the reporting vessel. For a given *MMSI*, each of its successive positional samples p consists of geographical coordinates (Lon, Lat) observed at discrete, totally ordered timestamps τ (e.g., at the granularity of seconds). Without loss of generality, we abstract vessels as 2-dimensional point entities moving across time, because our primary concern is to capture their motion features. By monitoring the timestamped locations from a large fleet of N vessels, the system must deal with a *positional stream* of tuples $\langle MMSI, Lon, Lat, \tau \rangle$. A *Data Scanner* decodes each AIS message, identifies those four attributes (the rest are ignored in our analysis), and cleans them from distortions caused during transmission (e.g., discard corrupt messages with bad checksum). This constitutes an *append-only* data stream, as no deletions or updates are allowed to already received locations.

But it is the sequential nature of each vessel's trace that mostly matters for capturing movement patterns *en route* (e.g., a slow turn), as well as spatiotemporal interactions (e.g., ships traveling together). Such a *trajectory* is approximated as an evolving sequence of successive point samples that locate this vessel at distinct timestamps (e.g., every few seconds). Our system accounts for stream imperfections, i.e., the noise inherent in vessel positions due to sea drift, delayed arrival of messages, or discrepancies in GPS signals. Indeed, prior to any processing, all incoming AIS positions are filtered through the *Noise Reduction* module by applying heuristics against a velocity vector maintained per vessel[4]. Afterwards, the *Mobility Tracker* module accepts clean data and checks when and how velocity changes with time. Working entirely in main memory and without any index support, it can detect two kinds of *trajectory movement events* (ME) as shown in Table 1:

- *Instantaneous trajectory events* involve individual time points per route, by simply checking potentially important changes with respect to the previously reported location (e.g., a sharp change in heading).
- *Long-lasting trajectory events* are deduced after examining a sequence of instantaneous events over a longer (yet bounded) time period in order to identify evolving motion changes. For example, a few consecutive changes in heading may be very small if each is examined in isolation from the rest, but cumulatively they could signify a notable change in the overall direction.

[4]Typically for trajectories [7], linear interpolation is applied between each pair of successive measurements (p_i, τ_i) and (p_{i+1}, τ_{i+1}). For simplicity, we assume that this also holds in the case of vessels. With the exception of intermittent signals, their course between any two consecutive positions practically evolves in a very small area, which can be locally approximated with a Euclidean plane using Haversine distances.

At each window slide, those events are compiled by a *Compressor* and a sequence of *"critical"* points (such as a stop) are emitted, which are much fewer compared to the originally relayed positions. Accordingly, the current vessel motion can be characterized in real time with particular *annotations* (e.g., stop, turn). Once new trajectory events are detected per vessel upon each window slide, the annotated critical points can be readily emitted and visualized on maps through a *Trajectory Exporter*, e.g., as KML polylines (for trajectories) and placemarks (for vessel locations).

Not surprisingly, detecting trajectory events from positional streams essentially performs a kind of path simplification. In the literature, some strategies like [7, 25, 27] specify an error tolerance for the resulting approximation. The memory footprint occupied by the compressed trajectory may also be a constraint in a single-pass evaluation [35]. Mainly focusing on savings in communication cost, dead-reckoning policies like [41] may be employed on board of the moving objects to relay positional updates only upon significant deviation from the course already known to a centralized server. However, this does not hold for AIS data, as maritime control centers wish to locate ships as frequently as possible. Most importantly, a major advantage of our proposed scheme is that it annotates the simplified representations according to particular trajectory events (turn, stop, etc.), thus adding rich semantic information all along each compressed trace.

Moreover, the derived critical points are propagated to the *Complex Event Recognition* module, which combines this event stream with static geographical data, such as protected areas. The objective of this process is to detect potentially suspicious or dangerous situations, such as illegal shipping and vessel rendezvous, as detailed in Table 1. This list contains a subset of events that could be of interest; for a more complete list, see [10, 24]. We have chosen to implement only those that were deemed most important, after communicating with the domain experts of the AMINESS project. As with the trajectory movement events, complex events are also categorized either as instantaneous or long-lasting (durative). Moreover, complex events can be also classified according to the following criteria:

- *Single-vessel events* keep track of a single vessel only (e.g., *illegal shipping*).
- *Dual-vessel events* must take into account all possible combinations of two vessels when we need to detect some form of interaction between them.
- *Area-based events*, like *suspicious area*, do not aim to report the specific vessels involved, but only the geographic region in which such an activity takes place.

All recognized complex events are eventually pushed in real-time to the marine authorities for decision-making.

3 Detecting trajectory events

As illustrated in Fig. 1, the system accepts fresh AIS messages from ships and extracts positional tuples $\langle MMSI, Lon, Lat, \tau \rangle$. With the possible exception of local manoeuvres near ports, marine regulations, or harsh weather conditions, vessels are normally expected to follow almost straight, predictable routes. In terms of vessel mobility, what matters most is to detect when and how the general course has changed, e.g., identify a stop, a turn, or slow motion. Such instantaneous and long-lasting *trajectory movement events* (ME) can indicate "critical points" along the trace of each vessel and thus offer a concise, yet quite reliable representation of its course.

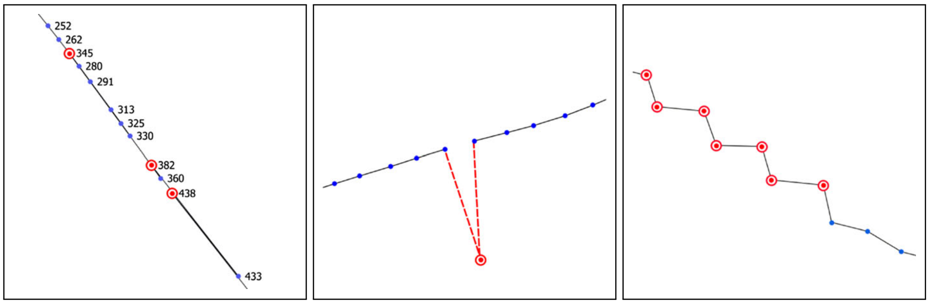
In order to identify significant motion changes, we employ an instantaneous velocity vector $\overrightarrow{v}_{now}$ over the two most recent positions reported by each vessel *MMSI*. In addition, we maintain a mean velocity $\overrightarrow{v_m}$ per ship over its previous m positions (m is a small integer) so as to abstract its short-term course. With our heuristics, it turns out that a large portion of the raw positional reports can be suppressed with minimal loss in accuracy, as they hardly contribute any additional knowledge.

In this Section, we first present simple, yet quite effective filters as a means of eliminating noise inherent in the streaming AIS data. Next, we describe how the sequence of vessel positions can be processed *online* in order to detect trajectory movement events and thus maintain a lightweight synopsis of each vessel's course.

3.1 Online noise reduction

Despite its high value in maritime surveillance, AIS data is not error-free. In fact, there are several sources of error that render a portion of this data noisy and inadequate for monitoring. First, no precise timestamp value is present in AIS messages relayed by the transponders on board; instead, they only report a lag value (in seconds) from the previously transmitted message. Obviously, this value cannot be used for establishing a temporal order, since positional updates from a single vessel may come from a series of base stations (those within range of its antenna along the route).

Therefore, a *transaction timestamp* marking the arrival of each AIS message at a station has to be used instead. Inevitably, transmission delays may frequently occur between the original message and its arrival. Successive positional messages from a single vessel may often arrive intermingled at a distorted order. Figure 2a illustrates such *out-of-sequence* messages, where numbers signify timestamp values since the beginning of this trajectory. Had those positions been retained according to their order of arrival, the vessel would occasionally appear in a state of sudden *agility*, moving back and forth very rapidly at a quite unusual speed. To make matters worse, AIS networks do not have synchronized clocks. Hence, if a vessel is within range of two stations, then a broadcasted message may be received by each one and possibly assigned with a different transaction timestamp. It may also happen that the same timestamp is assigned to different locations (maybe of considerable distance) of the same vessel. Therefore, *duplicate* or *contradicting positions* of a vessel may be present in the collected data. Noise might not always be caused by technical issues or the inherent errors and discrepancies in the GPS positions. It may be also due to deliberate, *suspicious actions*, e.g., switching off the transponder or emitting "spoofed" coordinates in order to avoid surveillance in a sensitive area. To the extent possible, such intermittent or falsified positions should be detected and cleared.



(a) Out-of-sequence (b) Off-course position (c) Zig-zag movement

Fig. 2 Noise-related situations along a vessel's course

Coping with noisy situations over AIS data is particularly challenging and has attracted significant research interest. As argued in [31], noise reduction should not be confused with anomaly detection, because data must be cleaned in advance, before performing any analysis. In that particular work, the well-known DBSCAN clustering algorithm [15] was used to identify outlier positions, which could then be removed from the dataset. With respect to time delays, an adaptive filtering strategy was suggested in [28], which employed Kalman filters and Monte-Carlo simulations to sequentially detect such delays and probabilistically rearrange a "correct" timestamp order. Besides, a stochastic method can be used to cope with position spoofing [19] by exploiting auxiliary data from radars or comparing it with previously tracked information. A common characteristic of all such methods is that they work in offline fashion employing expensive, iterative filters over archived AIS datasets.

In contrast, in this work we wish to apply *online*, single-pass filters over the incoming stream of AIS positions. Besides, given that trajectory compression is one of our principal objectives, we can afford to lose garbled, out-of-sequence positions and not consider correcting their timestamps. After all, except for cases of malicious activity, a fresh noise-free location will be soon received from a vessel, effectively compensating for the removal of any erroneous preceding one(s). In order to effectively and efficiently eliminate noise in timestamped AIS positions, we resort to applying a series of simple heuristics that examine the instantaneous velocity vector $\overrightarrow{v}_{now}$ of each vessel as computed by its two most recent observations. A noisy situation is identified if at least one of the following conditions apply:

– *Off-course positions* incur an abrupt change both in speed and heading of velocity $\overrightarrow{v}_{now}$. Such an outlier can be easily detected since it signifies an abnormal, yet only temporary, deviation from the known course as abstracted by mean velocity $\overrightarrow{v_m}$ of the ship over its previous m positions. Figure 2b illustrates such a case with a vessel that is unexpectedly located far away from its anticipated route.
– When vessels are on the move, they normally take their turns very smoothly (especially larger ships), so a series of AIS locations are transmitted, each marking a small change in heading as in Fig. 3f. However, if the latest position update indicates that a vessel has suddenly made a very abrupt turn (e.g., over 60^o) with respect to its known course (even though its speed may not be altered significantly), then this message should better be ignored altogether. Note that in case of adverse weather conditions (e.g., a storm) a vessel's route may appear as a *'zig-zag'* polyline with a series of such abrupt turns as shown in Fig. 2c. Dropping those consecutive points as noise is not typically correct; yet, in terms of data reduction this is quite desirable, as the vessel does not make any intentional turn and generally follows its planned course.
– When a vessel appears to accelerate too much, i.e., at a rate that it is not usual for a ship, this is another indication of noise. This is typical for *out-of-sequence* messages with twisted timestamps, as the three red spots in Fig. 2a. Each of these three locations is along the course of the ship, but due to their late arrival to the base station, the vessel is seen as suddenly retracting backwards. The location at timestamp $t = 438$ seconds is 270 meters back from the position at $t = 433$ seconds, resulting in a speed of 105 knots, quite unrealistic for any vessel.
– If an identical location from the same vessel has been already recorded before, then this might be a sign of error. Note that even if a vessel remains anchored, its successive GPS measurements usually differ by a few meters. In that case, instantaneous velocity $\overrightarrow{v}_{now}$ is infinitesimal, but not exactly zero. Instead, coincidental coordinates in succession should be deemed as almost certain duplicates, possibly concerning position reports

that arrived at slightly different times at more than one AIS stations within range of the vessel's antenna.

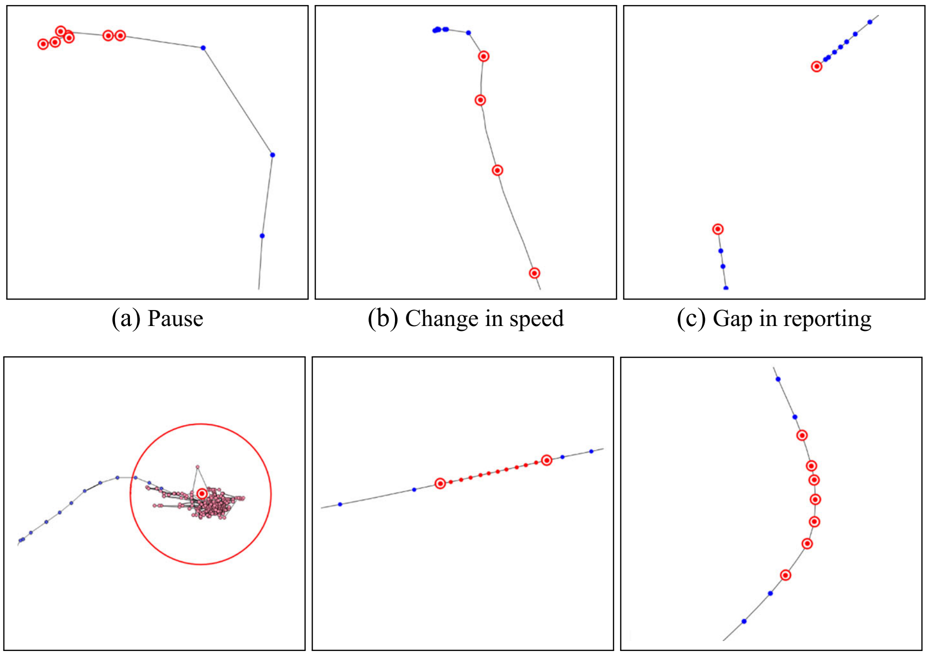
– A similar problem occurs with conflicts in timestamping, when the same timestamp is assigned to two distinct messages from a given vessel, even though they may be probably reporting different coordinates. In this case, instantaneous velocity $\overrightarrow{v}_{now}$ cannot be computed, signifying that these messages are contradictory (if not violating previous rules, we arbitrarily retain the latest one).

As we experimentally verified (cf. Section 5), as much as 20 % of the raw AIS positions may be qualifying as noise, falling in one of the aforementioned cases. Most importantly, accepting noisy positions would drastically distort the resulting trajectory synopsis, as the red dashed line in Fig. 2b illustrates. Even worse, noise may affect proper detection of movement events, as we will discuss next. Hence, although based in empirical heuristics, such noise reduction has proven certainly beneficial in terms of performance without sacrificing accuracy in the resulting approximation.

3.2 Online tracking of moving vessels

Once potentially noisy positions are cleared, the *mobility tracker* can promptly deduce *instantaneous* events by examining the trace of each vessel alone. In particular:

i) *Pause* indicates that a vessel is temporarily halted, once its instantaneous speed v_{now} is below a suitable threshold v_{min}. For example, if v_{now} is currently less than $v_{min} = 1$ knot, then the ship seems idle. For the vessel shown in Fig. 3a, the red bullets indicate



(a) Pause (b) Change in speed (c) Gap in reporting

(d) Long-term stop (e) Slow motion (f) Smooth turn

Fig. 3 Instantaneous and long-lasting trajectory events

several pause events; apparently, the ship is anchored at the port and such small displacements may be due to GPS errors or sea drift.

ii) *Speed change* occurs once current v_{now} deviates by more than $\alpha\%$ from the previously observed speed v_{prev}. Given a threshold α, the formula $|\frac{v_{now}-v_{prev}}{v_{now}}| > \frac{\alpha}{100}$ indicates whether the vessel has just decelerated or accelerated. This normally happens when approaching to or departing from a port, as depicted in Fig. 3b.

iii) *Turn* is spotted when heading in $\overrightarrow{v}_{now}$ has just changed by more than a given angle $\Delta\theta$; e.g., if there is a difference of $\geq 15^o$ from the previous direction.

No critical point gets immediately issued upon detection of any such simple events. An instantaneous pause or turn may be haphazard only; these are not meaningful out of context, because a series of such events may signify that the ship is stopped for some time. To avoid iteratively probing these locations later, we simply attach a *bitmap* to each stream tuple, using one bit for each particular instantaneous event. Note that multiple bits may be set at each location; e.g., the vessel may have just taken a sudden turn and also changed its speed above the respective thresholds, hence two distinct bits must be set to 1.

By buffering these instantaneous events within the window, we then can detect spatiotemporal phenomena of some duration. Examination of such *long-lasting trajectory events* is carried out in the following order. Note that if a certain long-lasting event has just been detected at a location, then checking if it also qualifies for another event is skipped altogether.

1. *Gap* in reporting is examined first. This event is spotted when a vessel has not emitted a message for a time period ΔT, e.g., over the past 10 minutes. This may occur when the vessel sails in an area with no AIS receiving station nearby, or because the transmission power of its transponder allows broadcasting in a shorter range. Then, its course is unknown during this period, as it occurs between the two red bullets in Fig. 3c. Reporting that contact was lost is important not only for online monitoring, but also for safety reasons, e.g., a suspicious move near maritime boundaries, or a potential intrusion of a tanker into a marine park. A pair of critical points signify when contact was lost (*gapStart* annotates the previously reported location) and when it was restored (*gapEnd* for current location).
2. Checking for a *long-term stop* is only fired if the vessel is noticed to move ($v_{now} > v_{min}$) just after a pause. If current location is preceded by at least m consecutive instantaneous pause or turn events in the buffer, and they are all within a predefined radius r (e.g., 250 meters), then a long-term stop is identified. In Fig. 3d, the red points inside the circle succeed one another and indicate such immobility, so they are collectively approximated by a single critical point (their centroid) annotated as *stopped* with their total duration.
3. *Slow motion* means that a vessel consistently moves at very low speed ($\leq v_{min}$) over its m most recent messages, as in Fig. 3e. If those buffered positions have not already qualified as a long-term stop by the previous rule (because they did not fall inside a small circle), then they probably succeed each other slowly along a path. The first and the last of these positions are both reported as critical, respectively annotated as *lowSpeedStart* and *lowSpeedEnd*.
4. *Smooth turns* are examined last. Due to their large size and maritime regulations, vessels normally report a series of locations when they change course. By checking whether the cumulative change in heading over buffered previous positions exceeds a given angle $\Delta\theta$, a series of such critical *turning* points may be emitted, as illustrated with the red points in Fig. 3f.

Thus, critical points are emitted from each detected long-lasting trajectory event, and this relies heavily on efficient noise reduction (cf. Section 3.1). For instance, an outlier breaking the subsequence of instantaneous pause events could prevent characterization of a long-term stop, and instead yield two successive such stops very close to each other. Moreover, in case that no long-lasting trajectory event was identified at this location, we check its associated bitmap with these additional rules:

5. If the bit for 'turn' is set, we check whether the current heading in $\overrightarrow{v}_{now}$ also deviates more than $\Delta\theta$ from mean velocity $\overrightarrow{v}_m$ of the vessel. If true, we emit a critical *turning* point. Note that this could possibly affect only raw AIS locations that are not qualified as erroneous. In fact, noisy positions as those in Fig. 2c have already been discarded by the Noise Reduction module.
6. If the bit for 'speed change' is set (Fig. 3b) and current speed v_{now} also deviates by more than $\alpha\%$ from the mean speed v_m of this vessel, then a critical point must be emitted and annotated as *speedChange*. It signifies that this instantaneous event was not caused by fluctuations in the measured speed due to delayed messages, but that such change in speed is probably valid.

Clearly, this detection process can only lead to a *single annotation* for each critical point. For instance, if a vessel disappeared for long and is suddenly found anchored somewhere, this event will be spotted either as a gap or a stop, but not both. We have deliberately chosen such a 'crisp' classification allowing a single characterization per detected point, as our goal is to achieve a concise trajectory representation, by dropping superfluous locations. In future work, we plan to introduce a fuzzy, probabilistic scheme of multiple annotations per critical point at diverse confidence margins.

Each critical point is issued along with a *velocity vector* (comprising instantaneous speed and heading), as an indicator of the short-term course of that particular vessel. This measurement may be useful for further analysis, e.g., in order to identify complex maritime events as explained in Section 4.

The example trajectory in Fig. 4 illustrates the data compression gains achieved when retaining critical points only. Obviously, such filtering greatly depends on proper choice of parameter values, which is a trade-off between reduction efficiency and approximation accuracy. For a suitable calibration of these parameters, apart from consulting maritime domain experts (our partners in the AMINESS project), we have also conducted several exploratory tests on randomly chosen vessels from AIS data in the Aegean Sea. For instance, setting $\Delta\theta = 5^o$ instead of $\Delta\theta = 15^o$ may even double the amount of critical points, because more raw AIS locations would qualify as turning points due to sea drift and discrepancies in GPS signals. Since our analysis is mostly geared towards data reduction, for our empirical study (Section 5.1) we have chosen an aggressive parametrization (values listed in bold in Table 4), which yields quite tolerable accuracy. With more relaxed parameter values, additional movement events can be detected, capturing slighter changes along each trajectory.

The complexity for detecting instantaneous events and communication gaps is $O(1)$ per incoming positional tuple, since only the two latest locations are examined per vessel. The cost for the remaining long-lasting events is $O(b)$, where b is the number of buffered positions that need inspection by each such rule. This may involve just a few points in case of smooth turns, but b may be higher (sometimes, a few hundred) if a series of positions qualify for a stop or slow motion. However, we stress that checks for stop of slow motion are fired rather infrequently, only once a vessel starts moving after a certain period of idleness. Thus, the overall cost is more than affordable and detection is near real-time, as we empirically verify in Section 5.1.

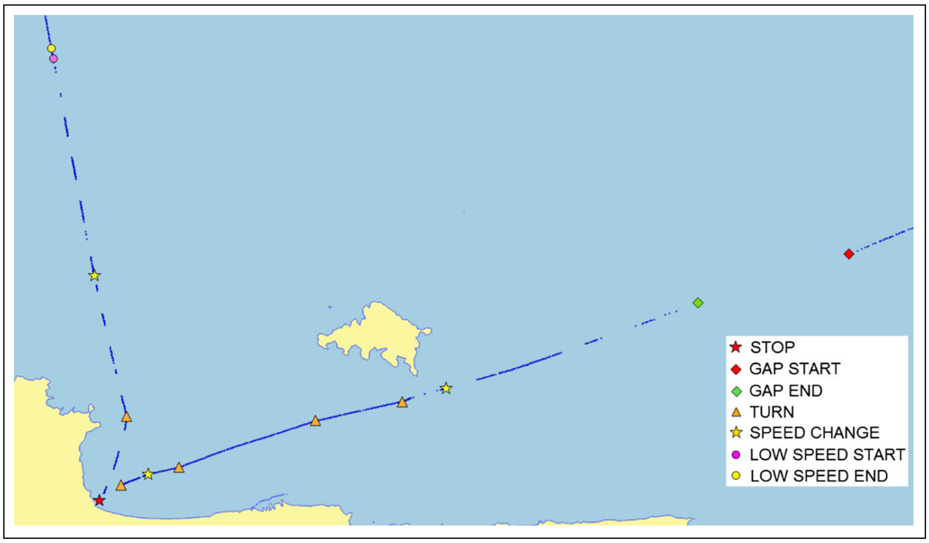


Fig. 4 Critical points identified along a vessel trajectory (raw AIS positions are shown as blue dots)

By taking advantage of those online annotations at critical points along trajectories, lightweight, succinct *synopses* can be retained per vessel over the recent past. Then, the *compressor* module simply evicts point locations that have not been detected as critical. Instead of resorting to a costly simplification algorithm, we opt to reconstruct vessel traces approximately from already available critical points. Such summarization depends on the annotation of detected trajectory events (stop, turn, gap, etc.), so as to refresh each trajectory accordingly. This main-memory process affects trajectory portions currently within the sliding window. Of course, resulting synopses may be also archived via the *Trajectory Exporter* module for offline use as files (e.g., KML, CSV) or in a database by incrementally emitting *"delta" batches* of critical points as they get identified at each slide of the window.

The aforementioned rules for detecting trajectory events are suitably defined in the mobility tracker, which allows fast, in-memory maintenance of movement features. Note that additional events can be detected by simply enhancing the mobility tracker with extra conditions. In future work, we plan to complement this methodology so as to capture more features, such as traveled distance from a given origin (e.g., a port). Nonetheless, even with this set of filters, we can figure out the mutability in each trajectory and distinctly characterize its course across time. Most importantly, these spatiotemporal features can serve as a basis to recognize more complex maritime events, as we discuss next.

4 Complex event recognition

The trajectory detection module compresses a vessel position stream to a stream of critical events, including the instantaneous events *gapStart* and *gapEnd*, indicating communication gaps, *lowSpeedStart*, *lowSpeedEnd*, *speedChange* and *turn*, and the durative event *stopped*.

Each such event is accompanied by the coordinates and velocity (speed and heading) of the corresponding vessel. This data stream, hereafter Movement Event (ME) stream, is transmitted to the complex event (CE) recognition module, which combines it with the locations of ports and protected areas, in order to recognize potentially suspicious or dangerous maritime situations, for the benefit of marine authorities.

The CE recognition module is based on the 'Event Calculus for Run-Time reasoning' (RTEC) [4]. The Event Calculus [22] is a logic programming action language. RTEC has a formal, declarative semantics—CE patterns in RTEC are (locally) stratified logic programs [36]. In contrast, almost all complex event processing languages, including [1, 6, 26], and several data stream processing languages, such as ESL [5] that extends CQL [3], lack a rigorous, formal semantics [9]. Reliance on informal semantics constitutes a serious limitation for maritime monitoring, where validation and traceability of the effects of events are crucial. Moreover, the semantics of event query languages and production rule languages often have an algebraic and less declarative flavor [14, 32]. In the following sections we present RTEC and illustrate its use for maritime monitoring.

4.1 Event calculus for run-time reasoning

RTEC has a linear temporal model including integer time-points. Following Prolog's convention, variables start with an upper-case letter, while predicates and constants start with a lower-case letter. For a *fluent* F—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. $\mathsf{holdsFor}(F = V, I)$ denotes that I is the list of the maximal intervals for which $F = V$ holds continuously. $\mathsf{holdsAt}(F = V, T)$ represents that fluent F has value V at some time-point T. holdsAt and holdsFor are defined in such a way that, for any fluent F, $\mathsf{holdsAt}(F = V, T)$ if and only if T belongs to one of the maximal intervals of I for which $\mathsf{holdsFor}(F = V, I)$.

An *event description* in RTEC includes rules that define the *event instances* with the use of the happensAt predicate, the *effects of events* with the use of the initiatedAt and terminatedAt predicates, and the *fluent values* with the use of the holdsAt and holdsFor predicates, as well as other, possibly atemporal, constraints. Table 2 presents a fragment of the predicates available to the event description developer.

Fluents in RTEC are of two kinds: *simple* and *statically determined*. For a simple fluent F, $F = V$ holds at a particular time-point T if $F = V$ has been *initiated* by an event that has occurred 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*. To compute the

Table 2 RTEC Predicates

Predicate	Meaning
$\textsc{holdsAt}(F = V,\ T)$	The value of fluent F is V at time T
$\textsc{holdsFor}(F = V, I)$	I is the list of the maximal intervals for which $F = V$ holds continuously
$\textsc{happensAt}(E, T)$	Event E occurs at time T
$\textsc{initiatedAt}(F = V, T)$	At time T a period of time for which $F = V$ is initiated
$\textsc{terminatedAt}(F = V, T)$	At time T a period of time for which $F = V$ is terminated
$\textsc{intersectAll}(L, I)$	I is the list of maximal intervals produced by the intersection of the lists of maximal intervals of list L

intervals I for which $F = V$, i.e. $\mathsf{holdsFor}(F = V, I)$, we find all time-points T_s at which $F = V$ is initiated, and then, for each T_s, we compute the first time-point T_f after T_s at which $F = V$ is terminated. As an example, consider the formulation below:

$$\begin{aligned}
&\mathsf{initiatedAt}(gap(Vessel) = \mathsf{true},\ T) \leftarrow \\
&\quad \mathsf{happensAt}(gapStart(Vessel),\ T), \\
&\quad \mathsf{holdsAt}(coord(Vessel) = (Lon, Lat),\ T), \\
&\quad \mathsf{not}\, nearPorts(Lon, Lat) \\
&\mathsf{terminatedAt}(gap(Vessel) = \mathsf{true},\ T) \leftarrow \\
&\quad \mathsf{happensAt}(gapEnd(Vessel),\ T)
\end{aligned} \tag{1}$$

gap(*Vessel*) is a Boolean simple fluent denoting a communication gap for some *Vessel*, i.e. the *Vessel* stops transmitting AIS messages. In some cases, the absence of AIS messages is suspicious and thus we need to record it. *gapStart*(*Vessel*) and *gapEnd*(*Vessel*) are instantaneous MEs indicating, respectively, the time-points in which a *Vessel* stops and resumes sending AIS messages. *coord* is a fluent reporting the coordinates of a vessel. Like MEs, this type of information is provided by the trajectory detection module. *nearPorts*(*Lon*, *Lat*) is an atemporal predicate that becomes true when the point (*Lon*, *Lat*) is close to a port. 'not' is negation-by-failure [8]. Rule-set (1) states that *gap*(*Vessel*) = true is initiated if the trajectory detection module reports a *gapStart* ME for the *Vessel*, and the *Vessel* is far from the ports of the area under surveillance. Furthermore, *gap*(*Vessel*) = true is terminated when the *Vessel* resumes communications. Given rule-set (1), RTEC computes the list of maximal intervals during which *gap*(*Vessel*) = true holds continuously.

4.2 Spatial indexing

CE recognition for maritime surveillance requires various types of spatial operation [16, 24]. For instance, we need to determine whether a point—a vessel's location—lies inside a polygon indicating an area of interest, such as a protected area, or whether it is near another point, such as a port. Moreover, we need to detect the vessels that are in close proximity (heading towards each other). In our approach to CE recognition, the availability of the full power of logic programming is one of the main attractions of employing RTEC as the temporal formalism. It allows CE patterns to include not only temporal constraints but also (complex) atemporal constraints. Recall e.g. the use of the atemporal predicate *nearPorts* in the specification of communication *gap* in rule-set (1). This is in contrast to various state-of-the-art CE recognition approaches, such as [9, 12, 23, 42], which support very limited atemporal reasoning, thus being unsuitable for maritime monitoring.

For efficient spatial reasoning, we adopt a grid partitioning scheme which divides the surveillance area into equally sized cells (see Fig. 5). Each area of interest and port is assigned only to those cells with which it overlaps. This assignment is performed off-line and provided as background knowledge to RTEC. The use of a grid enables us to quickly determine, through a simple calculation on the coordinates, the cell inside which a vessel is located. The task of determining each vessel's cell is performed before each CE recognition query. This way, we can efficiently compute the number of vessels in close proximity and check whether a vessel is inside an area of interest, by performing calculations (e.g. using the ray crossings algorithm [30] for determining whether a point lies inside a polygon) *only* for those vessels/areas in the same or adjacent cells.

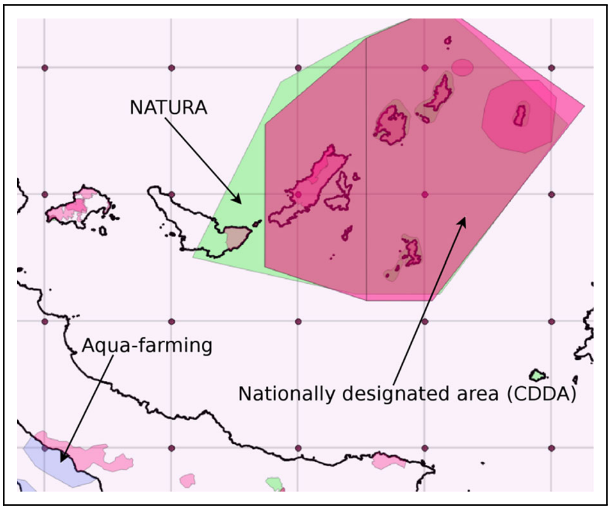


Fig. 5 Grid partitioning: different polygon types indicate different types of (overlapping) area of interest

4.3 Specifying complex maritime events

Given the critical ME stream produced by the trajectory detection module, and a set of protected areas, RTEC recognizes a set of CEs for the benefit of maritime authorities. The choice of CEs and their patterns were specified in collaboration with the domain experts of the AMINESS project. Below we present a fragment of our CE patterns. The complete list may be found in [2].

Suspicious vessel delay Some vessels, such as those passing through protected areas in order to minimize trip length and fuel consumption, switch off their transmitters and stop sending position signals. But sailing through a protected area is not the only reason for switching off an AIS transmitter. To investigate the behavior of vessels during a communication gap, we formulated the CE below:

$$\begin{aligned}&\mathsf{holdsFor}(\mathit{suspiciousDelay}(\mathit{Vessel}) = \mathsf{true},\ I) \leftarrow\\&\quad \mathsf{holdsFor}(\mathit{gap}(\mathit{Vessel}) = \mathsf{true},\ I_{gap}),\\&\quad \mathit{extendedDelays}(\mathit{Vessel},\ I_{gap},\ I)\end{aligned} \tag{2}$$

Recall that I in $\mathsf{holdsFor}(F = V, I)$ is the list of the maximal intervals for which $F = V$ holds continuously (see Table 2). I_{gap} in $\mathsf{holdsFor}(\mathit{gap}(\mathit{Vessel}) = \mathsf{true},\ I_{gap})$, therefore, is the list of maximal intervals during which a *Vessel* stops transmitting AIS signals while at open sea (see rule-set (1) for the *gap* fluent). $\mathit{extendedDelays}(\mathit{Vessel}, I', I)$ selects the maximal intervals I of the list I' for which the highest possible speed of the *Vessel* is below a threshold. We estimate the highest possible speed of a vessel in a simplified way: we assume that the vessel moved along a straight line from the point of *gapStart* to that of *gapEnd*. Under this assumption, its speed cannot have been greater than the one determined by dividing this shortest path by the time spent to travel it. Rule (2) thus states that a very low vessel speed combined with a communication gap occurring at open sea is to be treated as a suspicious delay.

The *suspiciousDelay* fluent is defined by a domain-specific holdsFor rule. We call fluents defined by such rules *statically determined*. For some fluents, it is much more concise to use domain-specific holdsFor rules defining the value of a fluent in terms of the values of other fluents, as opposed to 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.

Vessel rendezvous ‘Suspicious delay’ allows us to define additional types of suspicious activity; consider the rule below:

$$\begin{aligned}
&\mathsf{holdsFor}(\mathit{possibleRendezvous}(\mathit{Vessel_1}, \mathit{Vessel_2}) = \mathsf{true},\ I) \leftarrow \\
&\quad \mathsf{holdsFor}(\mathit{in}(\mathit{Vessel_1}, \mathit{Cell}) = \mathsf{true},\ I_1), \\
&\quad \mathsf{holdsFor}(\mathit{in}(\mathit{Vessel_2}, \mathit{Cell}) = \mathsf{true},\ I_2), \\
&\quad \mathsf{holdsFor}(\mathit{suspiciousDelay}(\mathit{Vessel_1}) = \mathsf{true},\ I_3), \\
&\quad \mathsf{holdsFor}(\mathit{suspiciousDelay}(\mathit{Vessel_2}) = \mathsf{true},\ I_4), \\
&\quad \mathsf{intersectall}([I_1, I_2, I_3, I_4],\ I)
\end{aligned} \tag{3}$$

in(*Vessel*, *Cell*) is a statically determined fluent indicating the *Cell* of the grid in which the *Vessel* is located. The value of this fluent is calculated according to the reported vessel coordinates and is set prior to each CE recognition query, as described in Section 4.2. intersectall is a built-in RTEC predicate which calculates the intersection of a list of lists of maximal intervals (see Table 2). According to rule (3), if two vessels simultaneously exhibit a *suspiciousDelay* and are located in the same area, then this could indicate that they had arranged for a rendezvous. Note that, since we do not have information about the vessels' locations during communication gaps, the above rule cannot capture the precise place and time of the rendezvous, if any.

Figure 6 illustrates the definition of *possibleRendezvous*. During the rendezvous, there would be no *in*(*Vessel*, *Cell*) fluents available. These fluents would become available only when a vessel re-appears, i.e. at the end of a communication *gap*. To detect instances of *possibleRendezvous*, we make the assumption that a *Vessel* remains in the same *Cell* for an interval extending 60 seconds before and after the timestamp of the reported vessel coordinates (see Fig. 6). Therefore, the intersection of the two *in* and the two *suspiciousDelay* intervals is not necessarily empty. We reserve for future work a more refined specification of this CE.

Maritime activities form hierarchies, in the sense that the formulation of one activity is also used to define other, higher-level activities. For example, vessel rendezvous is specified in terms of suspicious vessel delay. In contrast to many state-of-the-art CE recognition

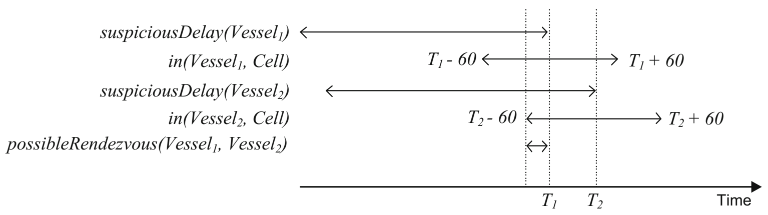


Fig. 6 Example of vessel rendezvous. *Vessel*$_1$ re-appears at time T_1 while *Vessel*$_2$ re-appears at time T_2. The interval of *possibleRendezvous* is the intersection of the intervals displayed above it

systems, such as Esper[5] and SASE[6], RTEC can naturally express hierarchical knowledge by means of well-structured specifications.

Fast approach Another dangerous situation may arise when a vessel is rapidly moving towards some other vessel(s). Such a behavior could indicate a vessel pursuit or even imminent collision. Consider the formalization below:

$$\begin{aligned}
&\mathsf{happensAt}(\mathit{fastApproach}(\mathit{Vessel}),\ T) \leftarrow \\
&\quad \mathsf{happensAt}(\mathit{speedChange}(\mathit{Vessel}),\ T), \\
&\quad \mathsf{holdsAt}(\mathit{velocity}(\mathit{Vessel}) = \mathit{Speed},\ T), \\
&\quad \mathit{Speed} > \mathit{20\ knots}, \\
&\quad \mathsf{holdsAt}(\mathit{coord}(\mathit{Vessel}) = (\mathit{Lon}, \mathit{Lat}),\ T), \\
&\quad \mathsf{not}\,\mathit{nearPorts}(\mathit{Lon}, \mathit{Lat}), \\
&\quad \mathsf{holdsAt}(\mathit{headingToVessels}(\mathit{Vessel}) = \mathsf{true},\ T)
\end{aligned} \tag{4}$$

fastApproach(*Vessel*) and *speedChange*(*Vessel*) are instantaneous CE and ME respectively. *velocity* is a fluent indicating the speed of a vessel. This information, as well as a vessel's heading, is provided by the trajectory detection module and accompanies every detected ME. *headingToVessels*(*Vessel*) is a fluent that becomes true whenever a *Vessel*'s direction of movement is towards at least one nearby vessel. According to rule (4), a 'fast approach' is recognized when a *Vessel* changes its speed at open sea, the new speed is above 20 knots, and there is at least one other nearby vessel towards which it is heading. The value of 20 knots was chosen by domain experts.

Package picking Another possible interaction between two vessels is when one of them drops a package at some area and another vessel appears later in order to pick it up. One way of formulating this type of interaction is the following:

$$\begin{aligned}
&\mathsf{happensAt}(\mathit{possiblePicking}(\mathit{Vessel_1}, \mathit{Vessel_2}),\ T_{pick}) \leftarrow \\
&\quad \mathsf{happensAt}(\mathsf{end}(\mathit{stopped}(\mathit{Vessel_1}) = \mathsf{true}), T_{drop}), \\
&\quad \mathsf{holdsAt}(\mathit{in}(\mathit{Vessel_1}) = \mathit{Cell},\ T_{drop}), \\
&\quad \mathsf{happensAt}(\mathsf{start}(\mathit{stopped}(\mathit{Vessel_2}) = \mathsf{true}), T_{pick}), \\
&\quad \mathsf{holdsAt}(\mathit{in}(\mathit{Vessel_2}) = \mathit{Cell},\ T_{pick}), \\
&\quad T_{pick} - T_{drop} < \mathit{1\ hour}, \\
&\quad \mathsf{holdsAt}(\mathit{coord}(\mathit{Vessel_1}) = (\mathit{Lon_1}, \mathit{Lat_1}),\ T_{drop}), \\
&\quad \mathsf{holdsAt}(\mathit{coord}(\mathit{Vessel_2}) = (\mathit{Lon_2}, \mathit{Lat_2}),\ T_{pick}), \\
&\quad \mathit{distance}((\mathit{Lon_1}, \mathit{Lat_1}), (\mathit{Lon_2}, \mathit{Lat_2}),\ \mathit{Dist}), \\
&\quad \mathit{Dist} < \mathit{0.5\ km}
\end{aligned} \tag{5}$$

stopped(*Vessel*) is a Boolean simple fluent indicating that a *Vessel* has stopped at open sea. The definition of this fluent is based on the information provided by the trajectory detection module, which reports the list of maximal intervals during which a vessel has stopped. From this list, we keep only those intervals where the vessel is not in any port. start($F = V$) (respectively end($F = V$)) is a built-in RTEC event taking place at the starting (ending) point of each maximal interval for which fluent F has value V continuously. For instance, start(*stopped*(*Vessel*) = true) takes place at the starting point of each maximal interval for which the *Vessel* has stopped at at open sea. Rule (5) describes a scenario where a vessel

[5]http://www.espertech.com/esper/

[6]http://sase.cs.umass.edu/

had stopped at some area and started moving at time T_{drop}, then, after no more than an hour, another vessel arrived and stopped at the same area, and the Haversine distance between the two stop locations, as calculated by the *distance* predicate, was no more than half a kilometer.

4.4 Recognizing complex maritime events

Listing 1 shows the pseudo-code of the main loop of RTEC. CE recognition is performed by means of continuous query processing, and concerns the computation of the maximal intervals of simple fluents (e.g. *gap* as defined by rule-set (1)) and statically determined fluents (such as *possibleRendezvous* defined by rule (3)), as well as the time-points in which events occur (e.g. *fastApproach* as defined by rule (4)). At each query time Q_i, the MEs that fall within a specified sliding window ω are taken into consideration. All MEs that took place before or at $Q_i - \omega$ are discarded/'forgotten' (see line 1 of Listing 1). At Q_i, the CE intervals computed by RTEC are those that can be derived from MEs 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 ME 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.

Listing 1 rtec(Q_i, ω)

Input: k: Depth of CE hierarchy; $SimpleFluents_n$: set of simple fluents of level n; $SDFluents_n$: set of statically determined fluents of level n; $Events_n$: set of events of level n

```
 1: forget($Q_i - \omega$)
 2: spatialProcessing($Q_i - \omega$)
 3: for $n \leftarrow 1$; $n \leq k$; $n{+}{+}$ do
 4:     for all $SDF \in SDFluents_n$ do
 5:         recognizeSDFluent($SDF$, $Q_i - \omega$)
 6:     end for
 7:     for all $SF \in SimpleFluents_n$ do
 8:         recognizeSimpleFluent ($SF$, $Q_i - \omega$)
 9:     end for
10:     for all $Ev \in Events_n$ do
11:         recognizeEvent ($Ev$, $Q_i - \omega$)
12:     end for
13: end for
```

Listing 2 recognizeSDFluent(SDF, $Q_i - \omega$)

```
 1: indexOf($SDF$, $Index$)
 2: retract(sdFList($Index$, $SDF$, $OldI$, $OldPE$))
 3: amalgamate($OldPE$, $OldI$, $OldList$)
 4: if $Start, End$ : $[Start, End) \in OldList \wedge End > Q_i - \omega \wedge Start \leq Q_i - \omega$ then
 5:     $PE := [(Start, Q_i - \omega + 1)]$
 6: else
 7:     $PE := [\,]$
 8: end if
 9: holdsFor($SDF$, $I$)
10: assert(sdFList($Index$, $SDF$, $I$, $PE$))
```

After ‘forgetting’ MEs, RTEC determines the cell inside which a vessel is located (see line 2 of Listing 1 and Section 4.2). Then, RTEC computes and stores the intervals of each CE of interest. RTEC restricts attention to *hierarchical* CE patterns, those where it is possible to define a function *level* that maps all fluents and all events to the non-negative integers as follows. Events and statically determined fluents of level 0 are those whose happensAt and holdsFor definitions do not depend on any other events or fluents. These are the input MEs. There are no simple fluents in level 0. Events and simple fluents of level n are defined in terms of at least one event or fluent of level $n-1$ and a possibly empty set of events and fluents from levels lower than $n-1$. Statically determined fluents of level n are defined in terms of at least one fluent of level $n-1$ and a possibly empty set of fluents from levels lower than $n-1$.

RTEC adopts a caching technique where the fluents and events of the CE hierarchy are processed in a bottom-up manner; this way, the intervals (resp. time-points) of the fluents (events) that are required for the processing of a fluent (event) of level n will simply be fetched from the cache without the need for re-computation. This technique is illustrated in the outer for-loop of Listing 1 (see lines 3–13) where fluent (event) processing starts at level 1 of the CE hierarchy and proceeds towards the top (k) level. The fluents (events) of the same level may be processed in any order. For illustration purposes, in Listing 1 the following order is adopted: statically determined fluents (lines 4–6), simple fluents (lines 7–9) and events (lines 10–12).

Listing 2 shows the pseudo-code of recogniseSDFluent, the procedure for computing and storing the intervals of statically determined fluents. First, RTEC determines the index of the given fluent *SDF*. The index is set by the user and allows for the fast retrieval of stored intervals for a given fluent even in the presence of very large numbers of fluents. Then, RTEC retrieves from the sdFList predicate the maximal intervals of *SDF* computed at the previous query time Q_{i-1} and checks if there is such an interval that overlaps $Q_i-\omega$ (lines 2–8). *OldI* represents the intervals of *SDF* computed at Q_{i-1}. These intervals are temporally sorted and start in $(Q_{i-1}-\omega, Q_{i-1}]$. *OldPE* stores the interval, if any, ending at $Q_{i-1}-\omega$. RTEC amalgamates *OldPE* with the intervals in *OldI*, producing *OldList* (line 3). If there is an interval [*Start*, *End*) in *OldList* that overlaps $Q_i-\omega$, then the sub-interval [*Start*, $Q_i-\omega+1$) is retained. See *PE* in Listing 2. All *SDF* intervals in *OldList* after $Q_i-\omega$ are discarded.

Subsequently, RTEC evaluates holdsFor rules to compute the *SDF* intervals from MEs recorded as occurring in $(Q_i-\omega, Q_i]$ (line 9). The computed list of intervals I of *SDF*, along with *PE*, are stored in sdFList (line 10), replacing the intervals computed at Q_{i-1}. When the user queries the maximal intervals of a fluent, RTEC amalgamates *PE* with the intervals in I.

Details about simple fluent and event processing, as well as a complexity analysis of RTEC, may be found at [4].

5 Empirical evaluation

Our maritime surveillance system has a modular design with loosely coupled components. The mobility tracker for online trajectory detection[7] is developed in GNU C++ and runs entirely on main memory for efficiently coping with volatile, asynchronously

[7]Source code is publicly available at http://www.dblab.ece.ntua.gr/~kpatro/tools/streamAIS/.

Table 3 Experimental settings

Parameter	Value
Vessel count N	6,425; 128,000; 1,280,000
Window range ω	10min; 1h; 2h; 6h; 9h; 24h
Window slide β	1min; 5min; 10min; 15min; 20min; 30min; 1h; 90min; 2h; 4h
Position stream rate ρ (positions/sec)	original; 1K; 2K; 5K; 10K
Protected areas	3,966 polygons with 78,418 edges

updated, streaming locations. RTEC[8], the complex event (CE) recognition component, is implemented in Prolog[9].

We conducted experiments against a real AIS dataset containing 23GB of AIS messages spanning from 1 June 2009 to 31 August 2009 for $N = 6{,}425$ vessels in the Aegean, the Ionian, and part of the Mediterranean Sea. Not all vessels were actually on the move at all times, since a considerable part (chiefly cargo ships) were just passing by, and thus tracked for a limited period (days or even hours). But most vessels were frequently sailing, e.g., passenger ships or ferries to the islands. When decoded and cleaned from corrupt messages, the dataset yielded 168,240,595 raw timestamped positions[10].

We simulated a streaming behavior by consuming this positional data little by little, i.e., reading small chunks periodically according to window specifications. We examine sliding windows with varying ranges ω and slide steps β based on timestamps from the original AIS messages. Thus, we replay this stream and the window keeps in pace with the reported timestamps and not the actual time of each simulation. The arrival rate of positions is fluctuating throughout this 3-month period and varies widely among vessels; none of them reports at a fixed frequency, whereas there are ships inactive for large intervals. If we only consider the activity period of each vessel (i.e., when it actually relays positions, either moving or not), then it reports every two minutes on average, which translates into a mean arrival rate $\rho \approx 50$ positions/sec from the entire fleet. For consistency with the real-world scenario, we consume the original stream "as is" in some simulations, even though this is a very low rate for a streaming application. We performed additional experiments at artificially increased rates so as to stress test our system and verify its efficiency and robustness. For the CE recognition component, artificially enlarged datasets include 1,2M vessels and 3,2B MEs. The simulation settings are listed in Table 3, whereas the calibrated settings for online mobility tracking are given in Table 4; default values are shown in bold.

In addition to this centralized approach, we have also examined the case of *parallelizing* the monitoring process for advanced efficiency against scalable data volumes. As a proof-of-concept, due to the different tasks of the two components, we employed different data partitioning schemes. First, concerning the trajectory detection component, the mobility tracker updates and maintains each trajectory in isolation from the rest. This process has been parallelized by using a varying number of *concurrent POSIX threads* in C++: each one

[8] https://github.com/aartikis/RTEC.

[9]The patterns of the complex maritime events are available at http://users.iit.demokritos.gr/~a.artikis/aminess.tar.gz.

[10]This anonymized dataset (for privacy, each original MMSI has been replaced by a sequence number) is publicly available at http://chorochronos.datastories.org/?q=content/imis-3months

Table 4 Mobility tracking parameters

Parameter	Value
Minimum speed v_{min} for asserting movement	1 knot ($\cong$1.852 km/h)
Maximum rate α of speed change between successive locations	25%
Minimum gap period ΔT (minutes)	5; 10; 15; 30 ; 60
Turn threshold $\Delta\theta$ (degrees)	2^o; 3^o; 5^o; 10^o; 15^o; 20^o
Radius r to determine long-term stops	250 meters
Minimal number m of inspected positions	10

is responsible to monitor a distinct subset of vessels. The resulting synopses get merged into a single derived stream of critical points (MEs) that is subsequently consumed by RTEC. With respect to the complex event recognition component, we employed multiple processors on which RTEC operated in parallel. We divided the grid covering the surveillance area into *multiple sub-grids* (groups of adjacent cells) whose number was equal to that of the processors used in parallel. Each processor was responsible for the areas and ports located in, and the vessels passing through its assigned sub-grid. We used three distributed settings: performing CE recognition on two, four and twelve processors. We made an attempt to evenly distribute the load of MEs among the different processors, by exhaustively searching for the best configuration. The different sub-grids had to be compact rectangles without dispersed cells. As a result, we did not take into account solutions with sub-grids of arbitrary shapes and the load distribution was thus not the best possible. Note that this partitioning is an off-line process and takes place only once.

Next, we report indicative results from these experiments. The trajectory event detection component operated on a server running Debian Linux “Wheezy” 7.5 amd64 with 48GB of RAM and two Intel Xeon X5675 processors at 3.07GHz. The CE recognition component RTEC operated on a computer with Intel Xeon CPU E5-2630 v2@2.60GHz$\times$12 processors and 256GB RAM, running Ubuntu Linux 14.04 and SWI Prolog 7.2.1.

5.1 Assessment of trajectory detection

5.1.1 Timeliness of online mobility tracking

First, we examine performance of online detection concerning trajectory movement events using simulations at the original arrival rate. These experiments have been performed employing window specifications with varying ranges ω and slide steps β, and we measure the total time it takes to update a window with a fresh batch of raw AIS locations, evict expired ones, detect trajectory events, and report critical points. Then, we calculate averages of these time values over the total count of window instantiations, hence obtaining the per slide cost for window maintenance and identification of any trajectory events therein. Figure 7 plots this average execution cost per window for monitoring the entire fleet.

From Fig. 7a it turns out that our mobility tracker provides results instantly for smaller ω up to 2 hours. In the worst case, it takes less than 150ms to track down any critical points per incoming batch of raw positional tuples. Quite expectedly, the cost grows linearly with an increasing slide β, as the window slides forward less often and thus each batch contains more input locations (illustrated with bars in this plot) directly proportional to the sliding step β.

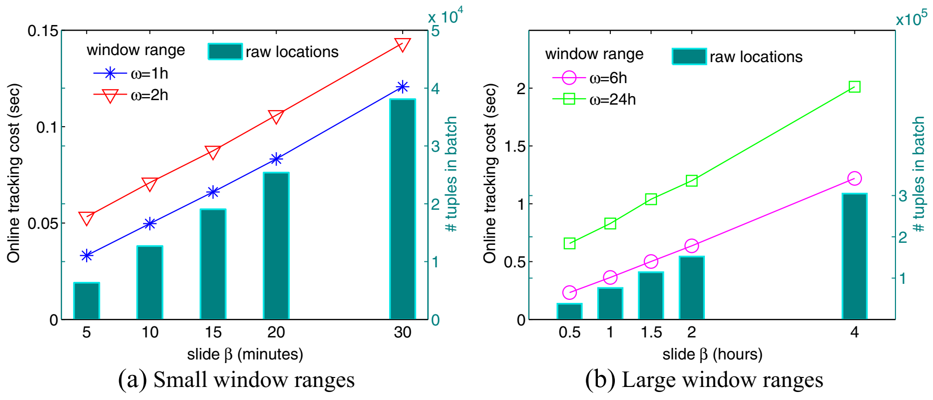


Fig. 7 Online mobility tracking cost per window slide

For larger windows with range ω up to 24 hours shown in Fig. 7b, the cost is greater. Increased by almost an order of magnitude compared to the execution times in Fig. 7a, the cost still remains linear with the size of wider sliding steps. Again, this is due to the larger amount of accumulated raw locations per batch (depicted with the bar plots). For the larger window tested ($\omega = 24$ hours and $\beta = 4$ hours), critical points can be reported in less than 2 seconds per batch, even though the mobility tracker has to validate almost 30,000 fresh raw positions each time. This clearly testifies the robustness and timeliness of the online tracking process.

We should also stress that these execution costs are drastically reduced compared to our previous performance study in [33]. Apart from better memory management in the implementation of the method, this improvement should be also attributed to the extra module for noise reduction. Since each position qualified as noise is filtered out without further processing, this incurs no more checks against the rest of the positions reported per vessel. Moreover, it reduces the size of the trajectory synopsis (i.e., critical points retained out of the raw positions per vessel), since such erroneous deviations from the known course are suppressed. Figure 8 plots the average amount of critical points retained per window state for several window ranges ω. It is no wonder that the number of critical points in window are proportional to its range, as this is actually the memory footprint of the maintained trajectory synopses. Space consumption is provably lightweight, since the trajectories of all vessels within the latest $\omega = 24$ hours can be approximately reconstructed from the almost 52,000 critical point locations maintained (on average) in the respective window state.

5.1.2 Scalability under varying arrival rates

Admittedly, such swift processing of raw positions is largely due to the low arrival rate of the original AIS stream (on average $\rho \approx$50 positions/sec). Hence, for a more stringent assessment of the online mobility tracking module, we performed some extra simulations, by admitting bigger chunks of data for processing at considerably increased arrival rates up to $\rho =$10,000 positions/sec. Then, given the fleet size N, every ship appears as reporting almost twice per second; although quite improbable in practice, this makes sense as a stress test for scalability.

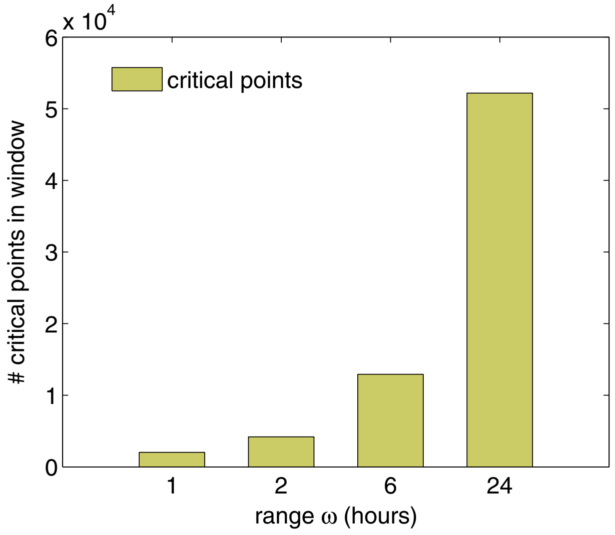


Fig. 8 Window states for varying ranges

As timeliness is also an objective, the window for the simulations in Fig. 9 was set with range $\omega = 10$ minutes and slide $\beta = 1$ minute. We first discuss performance when employing a *single thread* to tackle the entire trajectory detection process. Observe that critical points are still issued promptly for $\rho = 1,000$ positions/sec, but the latency grows with increasing rates. Note that this cost includes reporting time for the resulting critical points (i.e., after detection), and this adds a significant overhead at higher arrival rates, as greater chunks of AIS updates inevitably generate more critical points. In the worst case tested with $\rho = 10,000$ positions/sec, the online mobility tracker accepts 600,000 fresh raw positions

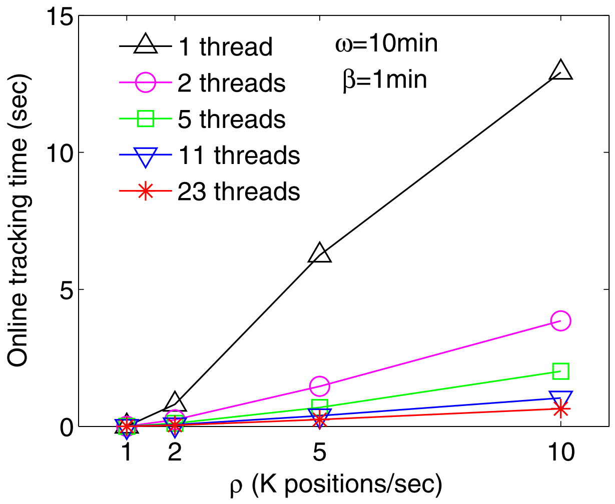


Fig. 9 Trajectory detection at various stream rates

every minute; yet, it can output results in less than 13 seconds, well before the next window slide.

The fact that the mobility tracker updates and maintains each trajectory in isolation from the rest, offers great opportunities for more advanced scalability. For the trajectory detection process, we also employed a varying number of *concurrent threads*, each one monitoring a distinct subset of vessels. Each thread consumes a substream of the incoming raw positions that correspond to the particular vessels it has been assigned to monitor. For simplicity, this subdivision is based on simple hashing over the $MMSI$ identifier of the vessel, such that its positions are always propagated to the same thread to establish consistency in trajectory maintenance. The system load may not be evenly balanced among the threads and cannot account to fluctuations in the arrival rate, but still the burden of processing can be shared and thus boost performance. In this case, the overall tracking cost per window is considered the maximum of costs incurred among the concurrent threads in order to emit results. Obviously, this cost differs depending on the size of the incoming batch consumed by a thread in each window instantiation. Figure 9 plots the average tracking cost per window slide when multiple threads are used (due to hashing, a prime number of threads was specified). There are significant savings even when employing two threads only; the original stream is halved into two substreams, but the cost drops by almost two thirds as each thread exploits better the available system resources. With more threads the cost still drops, although at a lower pace due to the overhead from context switching and contention for system resources. Overall, even this simplified approach confirms that the trajectory detection process is capable of handling scalable volumes of streaming vessel positions and has great potential for parallelization and advanced load balancing, as we plan to study in future work.

5.1.3 Compression efficiency

In this experiment, we examine the efficiency of our prototype in keeping only major trajectory characteristics as critical points and discard the rest. In order to measure the *compression ratio* accomplished by online trajectory tracking, we compared the amount of discarded points against the originally relayed locations per vessel. A compression ratio close to 1 signifies stronger data reduction, as the vast majority of original locations are dropped. The red line plot in Fig. 10 depicts measurements of this ratio with varying tolerance angles for detecting changes in heading. With a lower $\Delta\theta$, even slight deviations in vessel direction can be spotted, and thus extra critical points get reported. Bar charts in Fig. 10 illustrate the amount of critical points in each class (gap, low speed, speed change, stop, turn) retained from the entire dataset. Clearly, every further increase in threshold $\Delta\theta$ suppresses more and more turning points and only marginally affects the share of other classes, incurring extra reduction in the total amount of emitted critical points. Hence, relaxing this parameter value leads to a more intense compression. Most importantly, compression ratio always remains above 92 %, and with a more relaxed $\Delta\theta$ it reaches as much as 98 %. In this latter case, only 2 % of the original locations survive as critical, mostly by eliminating local manoeuvres of little impact on vessel's course. Eliminating noise also plays an important role in data reduction, as erroneous deviations are dropped and no points need be retained.

A similar pattern regarding reduction efficiency can be observed in Fig. 11 with respect to varying periods ΔT for detecting gaps in communication. Not surprisingly, it is the amount of critical points marking those gap periods that gets reduced with increasing thresholds ΔT. This time, reduction ratio is never below 96%, even though many more points are required to keep track that contact was lost even for 5 minutes. In a streaming context,

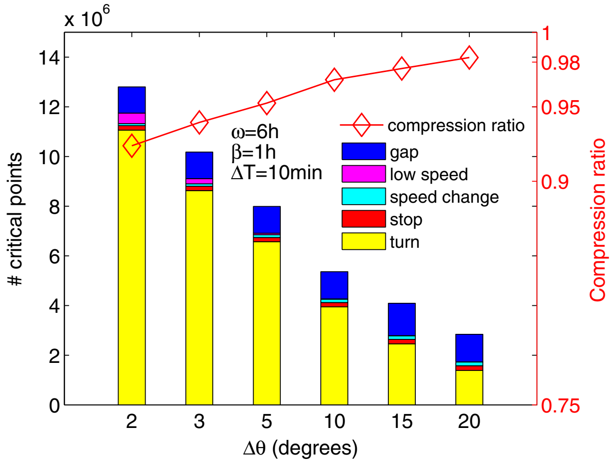


Fig. 10 Compression for varying $\Delta\theta$

such high compression ratios may lead to reduced system load in subsequent stages of the analysis, without sacrificing quality, as discussed next.

5.1.4 Approximation error

Preserving only critical points incurs a lossy approximation in trajectory representations. To assess the quality of those compressed trajectories, we estimated their deviation from the original ones (i.e., without discarding any raw positions except for those qualified as noise). Deviation can be computed from the pairwise distance between *synchronized* locations from the original and the compressed trajectory. If an original AIS point p_i at time t_i has been evicted as non-critical, then its corresponding time-aligned p'_i in the compressed trace can be estimated using linear interpolation along the path that connects the two critical

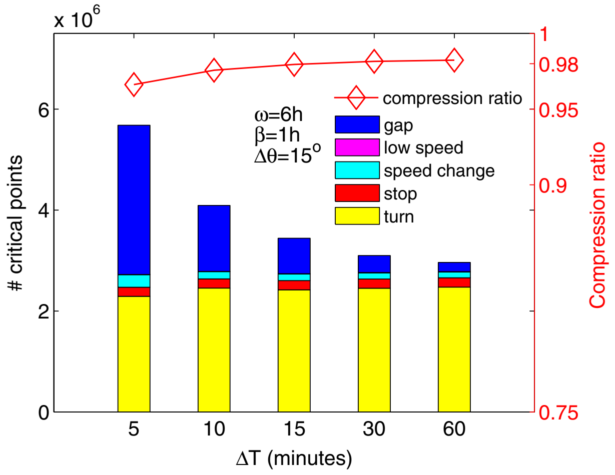


Fig. 11 Compression for varying period ΔT

points before and after t_i. For each vessel that has reported M raw positions, we estimated the root mean square error (*RMSE*) between the original and synchronized sequences of its locations as:

$$RMSE = \sqrt{\frac{1}{M} \cdot \sum_{i=1}^{M} (H(p_i, p_i'))^2}$$

which returns one RMSE estimate (in meters) per vessel trajectory and employs Haversine distance H between geographic coordinates. Figure 12 illustrates the number of vessel trajectories for certain intervals of these RMSE estimates. For example, RMSE is between 10 and 25 meters for trajectories of 3093 vessels (almost half of the fleet), whereas RMSE less than 10 meters occurs for another 1428 vessels. In contrast, only 3 vessels were found with RMSE above 100 meters. Although parametrization of the mobility tracker is common for all vessels, this result proves that it can offer a correct (or at least fairly trustful) approximation in almost all cases.

Figure 13 plots the *average* and *maximum RMSE* over the entire fleet for several values of turn threshold $\Delta\theta$, which is used to recognize significant changes in heading. As discussed in Section 3.2, the degree of trajectory approximation is mostly sensitive to parameter $\Delta\theta$ compared to the rest in Table 4 and this is reflected on the plot. Both error estimates escalate as this angle tolerance gets more relaxed. In the worst case for $\Delta\theta = 20^o$, average RMSE is only 22 meters and the maximum *RMSE* ever observed is 133 meters, which are negligible compared to the much larger size of open-sea vessels, and also considering the discrepancies inherent in GPS positioning and AIS transmissions. In practice, a moderate threshold of 10^o or 15^o may be adequate for balancing compression efficiency without losing important details in vessel mobility. Therefore, the suggested method can provide quite acceptable accuracy and can capture most, if not all, critical changes along each vessel's course.

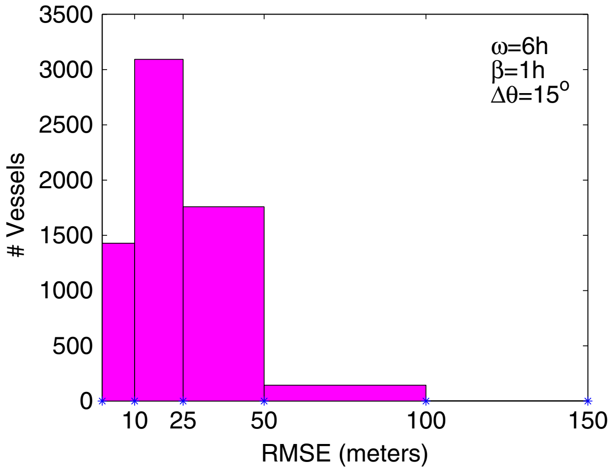


Fig. 12 Breakdown of RMSE for $\Delta\theta = 15^o$

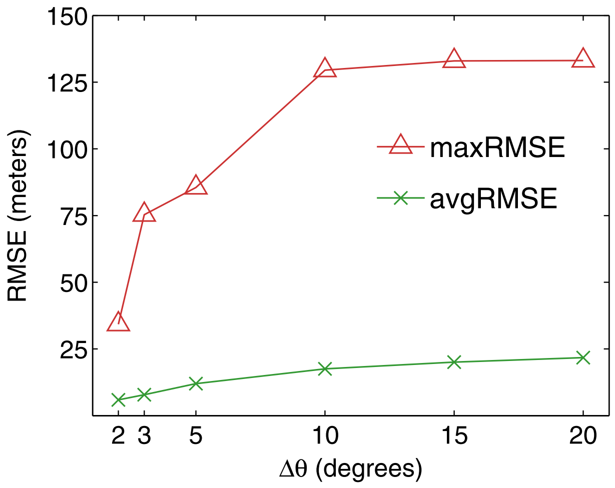


Fig. 13 Trajectory approximation quality

5.1.5 *Quality of synopses*

As raw AIS locations pass through the trajectory detection module in successive window instantiations, they get characterized according to their significance on vessel mobility. Figure 14 illustrates a breakdown of the resulting classifications after the input stream was exhausted and all critical points were detected for the entire 3-month period. More than half of the relayed raw positions indicate a "normal" course, i.e., a vessel moves according to its known velocity vector with a steady speed and heading. Thus, apart from notifying on the current position of each vessel, such points practically do not alter its trajectory and can be safely discarded without any further consideration. In addition, almost one out of five original positions is classified as noise for reasons explained in Section 3.1. It must be stressed that the vast majority of such points are not really "off-course" positions from the reporting vessel, but they actually fall along its route. However, due to their delayed arrival, these locations falsely indicate the vessel as moving back and forth in an agitating manner with no obvious reason, hence they should be purged altogether.

But it is the remaining vessel locations, after eliminating redundancy and noise, which get actually classified as critical points and can be used in trajectory summarization. As Fig. 14 testifies, there are relatively very few cases (about 0.1 % of the raw data) in which

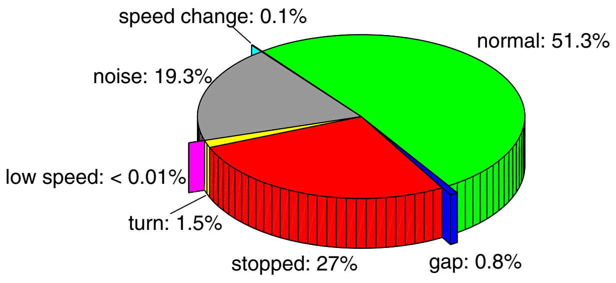


Fig. 14 Classified raw locations of all vessels

vessels either move at low speed or they change their speed considerably. This is to be expected, since vessels usually follow their planned course and they typically manoeuvre when arriving to or departing from ports. Gaps in communication account for about 0.8 % of the raw data. Although we observed that this phenomenon is rather frequent in practice, we note that at most two points are used to delimit this period and thus indicate the loss of contact with the respective vessel. Locations indicating turns are very important and must be certainly considered in trajectory representation. In fact, these points are roughly 1.5 % of the total, since only significant deviations (over 15^o) from the known course qualify for a turn, even though possibly emitting a series of such critical points in case of smooth turns as depicted in Fig. 3b. Finally, about 27 % of the raw positions are emitted when vessels are idle, most usually when anchored in a port. As discussed in Section 3, instead of keeping all these points, we collect successive "pause" events occurring within a small distance and merge them into a collective "stop" event located at their centroid. This incurs huge savings in the resulting synopses, leading into much more concise trajectory representations that can be highly usable in subsequent query processing and offline analytics.

This dramatic effect on summarization is much more evident in Fig. 15, which plots a breakdown of the accumulated critical points after processing the entire dataset. The resulting trajectories mainly consist of turning points between stops with some occasional changes in speed, but rather frequent communication gaps. Indeed, 60 % of critical points are turning points, which are only 1.5 % of the original AIS locations (Fig. 14). Points indicating gap periods are almost 32 % of the critical points, testifying the frequent loss of contact with vessels on the move. In contrast, long-term stops comprise a meagre 4.5 % in the resulting trajectory synopses, since locations that belong to the same stop event are compressed into a single centroid that suffices to designate that the vessel is idle during this period. Apart from redundant "normal" points and eliminating the inherent noise, condensing these stops really adds much to the reduction efficacy of the trajectory detection module, offering a valuable semantic interpretation of the motion features.

As an offline, post-processing step, we have reconstructed trajectories from the entire sequence of critical points accumulated per vessel. In effect, the long motion history of a ship can be broken up into shorter "trips" between identified stop points, which usually indicate anchorage at ports. Table 5 lists representative statistics from these approximate trajectories, and offers insight on a possible offline usage of the results from trajectory summarization. It turns out that a typical trip spanning several hours over a long distance (more than 131km) can be approximated with 48 points only; once more, this confirms the strong reduction effect of the method. Note that almost 5 % of the detected critical

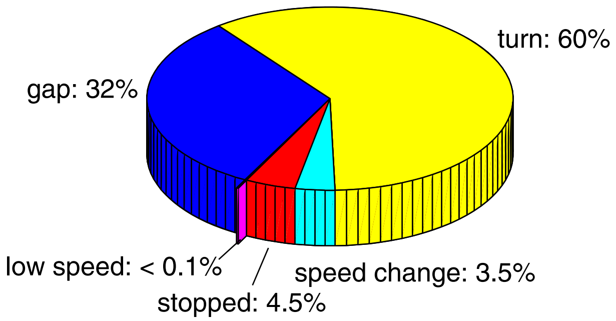


Fig. 15 Characterization of critical points

Table 5 Statistics from post-processing of compressed trajectories

Critical points in reconstructed trips	3,895,112
Critical points in open-ended trips	196,131
Average trips per vessel	20
Average number of critical points per trip	48
Average travel time per trip	07:24:48
Average traveled distance per trip	131.513km

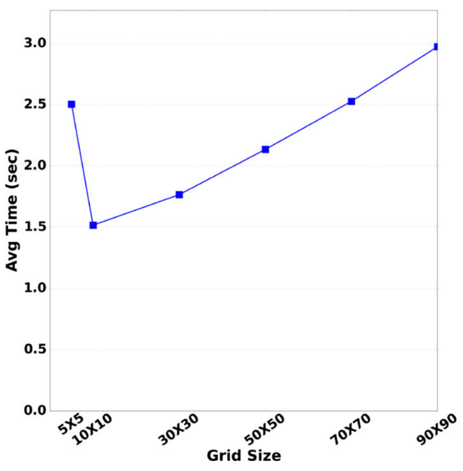
points belong to "open-ended" trips, as certain vessels were only spotted while traversing the Aegean without anchoring there.

5.2 Assessment of complex event recognition

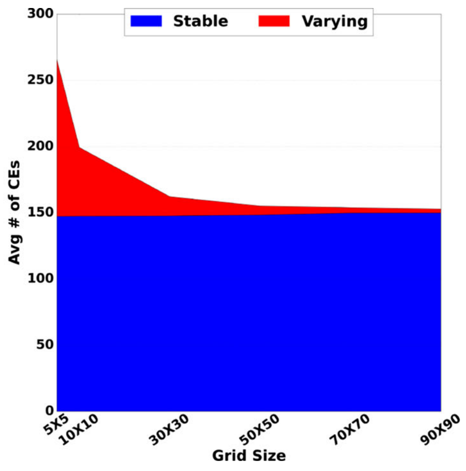
The trajectory detection module compresses a vessel position stream to a stream of critical movement events (ME)s. Each such event is represented by predicates expressing the activity of the vessel, its coordinates and its velocity (see Section 4). This way, the ME stream given to RTEC includes 15,884,253 predicates spanning over the 3-month period. In addition to this stream, RTEC makes use of real data consisting of protected areas, such as NATURA areas, represented as polygons, and ports, represented as points, across the Greek seas. The dataset has 3,966 protected areas with a total of 78,418 edges, and 64 ports. The size of the grid is 720×900 km^2. Given this combination of event stream and static geographical information, RTEC recognizes the following CEs: illegal shipping, suspicious vessel delay, vessel rendezvous, suspicious areas, vessel pursuit, and package picking (see Table 1).

5.2.1 Grid partitioning

Figure 16 shows the results from a first set of experiments in which we attempted to determine the optimal grid granularity/cell size. Starting with a grid having $5 \times 5 = 25$ cells (with a cell size of 138×170 km^2), we increased the number of cells along each dimension, up to $90 \times 90 = 8{,}100$ cells (each cell being 9×11 km^2 wide). Figure 16a shows the average CE recognition times in CPU seconds for each different grid. Both the window ω and the slide β are set to 1 hour. The worst grid configuration (90×90) is almost two times slower than the best (10×10), but in all cases the average time is within the same order of magnitude, and less than 3 seconds. Figure 16b shows the average number of recognized CEs for each different grid as a stack plot. The number of recognized CEs shows a decreasing trend initially, with a tendency to stabilize after grid configuration 30×30. The reason for the difference in the number of detected CEs lies in the way the respective patterns are defined. Most of the CE patterns are grid-independent. Therefore, the number of recognized CEs remains stable across all grid configurations. On the other hand, *possibleRendezvous*, as defined by rule (3), depends on the size of the grid cells. Recall that the place of vessel rendezvous cannot be determined precisely, since vessels stop transmitting AIS signals during the time of the meeting. Therefore, we can only define *possibleRendezvous* in terms of the cells in which the vessels in question stopped (respectively resumed) transmitting AIS signals. As a result, when the size of the cells increases, more *possibleRendezvous* CEs are being recognized. For this reason, instead of choosing



(a) AverageCE recognition times.



(b) Average number of CEs.

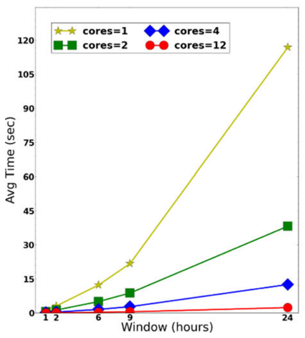
Fig. 16 CE recognition for different grid cell sizes

the 10×10 grid for the remaining experiments, we opted for the 30×30 one, which has comparable time performance and a stable number of detected CEs.

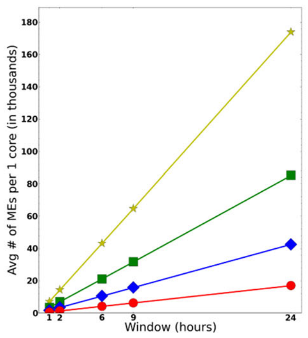
5.2.2 *Timeliness of complex event recognition*

Next, we proceed with a more thorough analysis of the performance of RTEC. Figure 17 shows the results of experiments under various window sizes and distributed configurations. First, we used a single processor to perform CE recognition for all 6,425 vessels, 3,966 areas and 64 ports. We subsequently employed multiple processors in three different settings, with two, four and twelve processors.

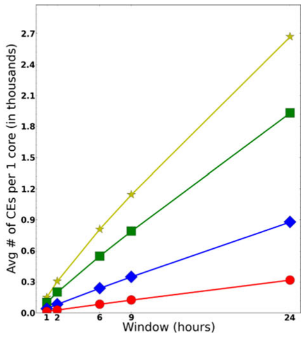
Figure 17a shows the average CE recognition times, including the time taken for spatial indexing (see Section 4.2). The slide β is 1 hour while the window ω ranges from 1 hour to 24 hours. Figure 17b shows the average number of MEs for each setting. In the case of a single processor, the window ranges from ≈7,200 MEs (1 hour) to 175,000 MEs (24 hours).



(a) Average CE recognition times.



(b) Average number of MEs.



(c) Average number of CEs.

Fig. 17 CE recognition under varying window sizes and distributed configurations

In the distributed settings—two, four and twelve processors for CE recognition— the input MEs are forwarded to the appropriate processor according to vessel location. For instance, when twelve processors are used in parallel, each one of them processes ≈700 MEs for the 1 hour window, and 17,000 MEs for the 24 hour window. Figure 17c shows the average number of CEs for each setting. This number also depends on the window size. In the case of a single processor, e.g., for 1 hour windows approximately 150 CEs are recognized, while for 24 hour windows RTEC recognizes around 2,900 CEs. We do not show memory consumption figures because memory usage is negligible and stable.

Figure 17a shows that we can achieve a significant performance gain by running RTEC in parallel. As the window size increases, the gain becomes more pronounced. Furthermore, Figure 17a shows that RTEC supports real-time CE recognition. E.g. for a window of 6 hours, RTEC recognizes all CEs in 14 sec when a single processor is used, and in 0.4 sec when twelve processors are used in parallel.

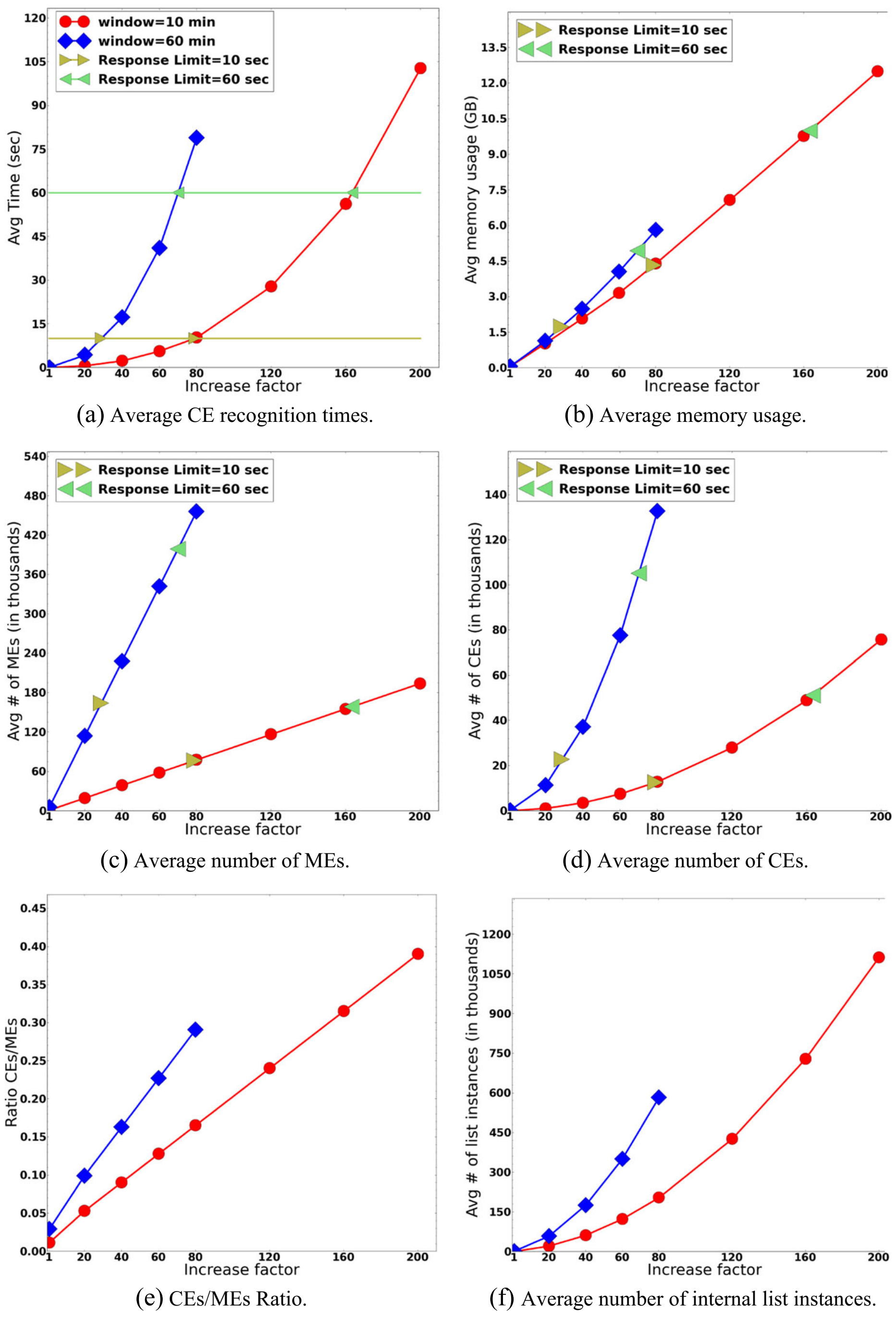
5.2.3 Scalability of complex event recognition

Since RTEC can comfortably handle the initial dataset, we tested its performance with artificially enlarged datasets in order to determine its limits in terms of scalability. We created enlarged data streams by inserting extra critical MEs to the real dataset. For each vessel, we replicated its trajectories by a specified number of times (increase factor). The replicated trajectories were assigned to new vessels which do not exist in the original dataset, thus adding more vessels as well. An increase factor X implies that we have exactly X times as many critical MEs and X times as many vessels, compared to the original dataset of ≈16,000,000 MEs and ≈6,500 vessels. We varied the increase factor from 20, producing datasets of ≈128,000 vessels and 320,000,000 MEs, to 200, creating data of ≈1,280,000 vessels and 3,200,000,000 MEs. To the best of our knowledge, this is the most comprehensive evaluation of CE recognition techniques in the maritime domain. Compared to e.g. [40], our surveillance area, number of vessels and data volume are substantially larger. Moreover, CE recognition needs to consider (a large number of) protected areas.

Figure 18 displays the experimental results. Twelve processors are used in parallel, the slide β is set to 1 min, and the window sizes ω are 60 min (1 hour) and 10 min. We set two 'response limits' to 10 sec and 60 sec—we stopped performing tests once the response limit of 60 sec was exceeded.

Figure 18a shows the average CE recognition times while Fig. 18b the average memory usage. As expected, a smaller window produces both lower recognition times and lower memory usage. Figures 18c and d display respectively the average number of MEs and CEs. We also show how RTEC performs with respect to the two response limits of 10 and 60 sec. Assume, for instance, that we want to guarantee that RTEC responds within 60 sec. In this case, for a window of 60 min, RTEC can handle data streams that are ≈70 times larger (≈450,000 vessels) than the original one (see Fig. 18a). This means that a window may include up to approximately 400,000 MEs, while recognizing around 105,000 CEs (see the 60 sec response limit markers on the 60 min window lines in Figs. 18c and d respectively). Memory consumption is 4,9 GB (see the 60 sec response limit marker on the 60 min window line in Fig. 18b). Similarly, for a window of 10 min, RTEC is guaranteed to respond within 60 sec even in data streams that are 160 times larger than the original (≈1,000,000 vessels).

Comparing the 60 min window/80 increase factor setting against the 10 min window/200 increase factor setting, we see that the latter has fewer MEs and CEs (see the right-most marks of the window lines in Figs. 18c and d). However, both the CE recognition times and memory usage are higher (see the right-most marks of the window lines



(a) Average CE recognition times.

(b) Average memory usage.

(c) Average number of MEs.

(d) Average number of CEs.

(e) CEs/MEs Ratio.

(f) Average number of internal list instances.

Fig. 18 CE recognition for approximately 128K–1,28M vessels and 320M–3,2B MEs. Response limits are indicated with lines in Fig. 18a and with points in Figs. 18b–d

in Figs. 18a and 18b). One reason for this behavior is that the complexity of the recognition task is higher in the 10 min window/200 increase factor setting. A simple measure of complexity in CE recognition is the ratio of CEs to MEs. This ratio is depicted in Fig. 18e.

As a further step towards understanding RTEC's behavior, we investigated how it stores some of its internal structures. During CE recognition, RTEC maintains several lists of time intervals for different types of event and fluent. In Fig. 18f, we plot the average number of instances of these lists. The results show that these lists are mostly responsible for the increased recognition times and memory usage, as the increase factor grows. More specifically, the lists concerning *possibleRendezvous* and *possiblePicking* event (see rules (3) and (5) respectively), require that all combinations of vessels within some area are checked. Therefore, when the number of vessels increases, as is the case with replicated trajectories, the number of possible combinations is increased by the square of the number of vessels, hence the parabolic lines in Fig. 18f. On the other hand, a window increase does not result in many more vessels being considered. Many of the added MEs may refer to extra messages transmitted by the same vessel over the longer duration of a larger window. Note that this is a different kind of complexity than the one reflected in the ratio of CEs to MEs. In the case where we have more combinations to check, this does not necessarily imply that proportionally more CEs will be recognized, since their definitions might not be satisfied. RTEC has to check all possible combinations though, thus incurring a higher latency and memory consumption.

6 System deployment

In order to verify its operational capabilities, we have set up our system to monitor *live* AIS feeds from vessels across the Aegean Sea. Fresh positions are periodically fetched from a

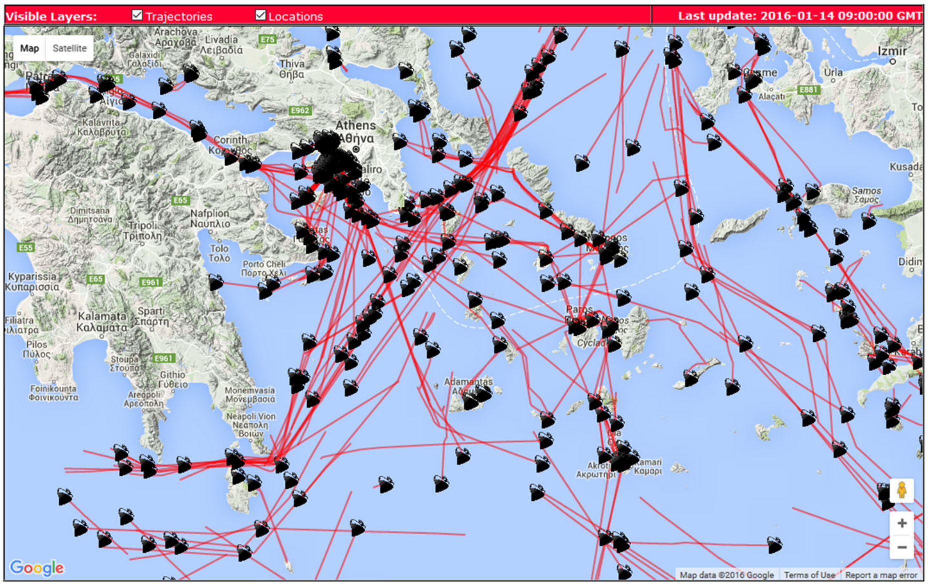


Fig. 19 Monitoring real-time AIS positions and vessel trajectories in the Aegean

database (maintained by the University of the Aegean) that receives any transmitted AIS messages. Each retrieved record only includes vessel identification, geographic position, and a timestamp. Currently, fresh input is fetched every hour (the sliding step β of the window employed in this instance), although a different period may be chosen. Typically, we observed that up to 40,000 fresh records can be fetched per hour, which is a rather moderate amount of geostreaming data compared with the arrival rates simulated in our empirical validation (cf. Section 5).

Once each batch of fresh data arrives, it is being consumed by the online mobility tracker in order to identify any critical points in each vessel's course. The resulting points with their annotations (turn, stop, etc.) are then archived into a PostgreSQL database. But the Trajectory Exporter module also converts them into KML files for map visualization in a web application, as illustrated in Fig. 19. Critical points may be exported upon detection, but in order to avoid duplication of data across the modules, we opt to export them once they get evicted from the sliding window (currently set to a range of ω =6 hours). Hence, exported results currently have a lag of 6 hours from the current time, but this is only a configuration parameter.

This deployment against real-time AIS messages collected across the Aegean has been activated since April 2015. This confirms that our system can integrate with a precious source of online data, offering to marine experts and authorities the means to instantly locate, recognize, and correlate events from real-time vessel traces.

7 Summary & future work

In this paper, we introduced a system that monitors activity of thousands of vessels and can instantly recognize events with a potentially serious impact on the environment and on safe navigation at sea. The system can sustain streaming messages from vessels and can filter out noise and redundant positions along their course. Hence, it can retain only succinct synopses of vessel trajectories, drastically reducing the original path into few critical points that convey major motion features. As empirically validated, with a proper parametrization our suggested trajectory summarization may incur a compression ratio of almost 98 %, with tolerable error in the resulting approximation. Moreover, trajectory detection is highly scalable and can be easily parallelized in order to sustain very high arrival rates in the input stream. Furthermore, this reduced information may be readily analyzed online for Complex Event (CE) recognition. Equipped with efficient pattern matching algorithms, this module correlates critical trajectory positions with static geographical data, and detects suspicious or dangerous situations, such as illegal shipping, suspicious vessel delay, vessel rendezvous, suspicious area, vessel pursuit and package picking. We showed that the CE recognition module performs in real-time using real data as well as synthetically enlarged datasets that include up to 1,28M vessels and 3,2B critical positions.

We plan further extensions and improvements in the existing implementation. First, since trajectory detection may be sensitive to parameters, we intend to study advanced methods for adaptive, auto-calibrated parameterization depending on the size, the type, and the motion patterns of vessels. Also, creating CE patterns manually is painstaking and error-prone. To facilitate the process of CE pattern construction, we plan to employ a recent framework for incremental structure learning that takes advantage of Big Data in order to construct Event Calculus programs [20]. Besides, maritime surveillance exhibits various types of uncertainty [39]. AIS messages are often corrupt, with incorrect or missing fields. Furthermore, maritime CE patterns do not account for all possible situations. To deal with these issues, we have been developing an Event Calculus dialect for Markov Logic Networks [38]. This way, we can use weight learning techniques [13] for estimating the confidence values of CE patterns, and subsequently perform probabilistic inference. Maritime surveillance may also benefit from combining multiple data sources. For example, the use of heterogeneous data sources, as in [40], can help in constructing more refined CE patterns. We aim to increase our data sources in order to improve monitoring quality and possibly recognize additional events. Last, but not least, it would be challenging to apply ideas from our methodology against other sources of big geostreaming mobility data, e.g., traces of aircrafts or vehicles. Although the particular definitions of events may differ, as well as their configurations, we expect that our methodology may still be valid.

Acknowledgments This work was funded partly by the *"AMINESS: Analysis of Marine INformation for Environmentally Safe Shipping"* project, which was co-financed by the European Fund for Regional Development and from Greek National funds, and partly by the EU-funded H2020 datACRON project (H2020-ICT-2015 687591). We wish to thank IMIS Hellas, our partner in AMINESS, for providing the AIS dataset used in the experiments.

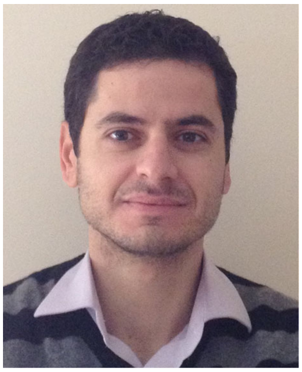
References

1. Agrawal J, Diao Y, Gyllstrom D, Immerman N (2008) Efficient pattern matching over event streams. In: SIGMOD
2. Alevizos E, Artikis A, Patroumpas K, Vodas M, Theodoridis Y, Pelekis N (2015) How not to drown in a sea of information: an event recognition approach. In: IEEE International conference on big data
3. Arasu A, Babu S, Widom J (2006) The CQL continuous query language: semantic foundations and query execution. VLDB J 15(2):121–142
4. Artikis A, Sergot MJ, Paliouras G (2015) An event calculus for event recognition. IEEE Trans Knowl Data Eng 27(4):895–908
5. Bai Y, Thakkar H, Wang H, Luo C, Zaniolo C (2006) A data stream language and system designed for power and extensibility. In: CIKM, pp 337–346
6. Brenna L, Demers AJ, Gehrke J, Hong M, Ossher J, Panda B, Riedewald M, Thatte M, White WM (2007) Cayuga: a high-performance event processing engine. In: SIGMOD, pp 1100–1102
7. Cao H, Wolfson O, Trajcevski G (2006) Spatio-temporal data reduction with deterministic error bounds. VLDB J 15(3):211–228
8. Clark K (1978) Negation as failure. In: gallaire H., Minker J. (eds) Logic and Databases, pp. 293–322. Plenum Press
9. Cugola G, Margara A (2010) TESLA: a formally defined event specification language. In: DEBS, pp 50–61
10. Project datACRON Deliverable D5.1: Maritime use case and scenarios. http://ai-group.ds.unipi.gr/datacron/system/files/datACRON_D5.1.Maritime_Use_Case.pdf
11. Dindar N, Fischer PM, Soner M, Tatbul N (2011) Efficiently correlating complex events over live and archived data streams. In: DEBS, pp 243–254
12. Dousson C, Maigat PL (2007) Chronicle recognition improvement using temporal focusing and hierarchisation. In: IJCAI, pp 324–329
13. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12:2121–2159
14. Eckert M, Bry F (2010) Rule-based composite event queries: the language xchange^{eq} and its semantics. Knowl Inf Syst 25(3):551–573
15. Ester M, Kriegel H, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp 226–231
16. Garcia J, Gomez-Romero J, Patricio M, Molina J, Rogova G (2011) On the representation and exploitation of context knowledge in a harbor surveillance scenario. In: FUSION, pp 1–8
17. Golab L, Johnson T (2013) Data stream warehousing (tutorial). In: ACM SIGMOD, pp 949–952
18. Idiri B, Napoli A (2012) The automatic identification system of maritime accident risk using rule-based reasoning. In: SoSE, pp 125–130

19. Katsilieris F, Braca P, Coraluppi S (2013) Detection of Malicious AIS position spoofing by exploiting radar information. In: FUSION, pp 1196–1203
20. Katzouris N, Artikis A, Paliouras G (2015) Incremental learning of event definitions with inductive logic programming. Mach Learn 100(2–3):555–585
21. Kazemitabar SJ, Demiryurek U, Ali MH, Akdogan A, Shahabi C (2010) Geospatial stream query processing using Microsoft SQL Server Streaminsight. PVLDB 3(2):1537–1540
22. Kowalski R, Sergot M (1986) A logic-based calculus of events New Generation Computing 4(1)
23. Krämer J., Seeger B (2009) Semantics and implementation of continuous sliding window queries over data streams ACM Transactions on Database Systems 34(1)
24. van Laere J, Nilsson M (2009) Evaluation of a workshop to capture knowledge from subject matter experts in maritime surveillance. In: FUSION, pp 171–178
25. Lange R, Dürr F., Rothermel K (2011) Efficient real-time trajectory tracking. VLDB J 20(5):671–694
26. Li G, Jacobsen HA (2005) Composite subscriptions in content-based publish/subscribe systems. In: Middleware
27. Meratnia N, de By R (2004) Spatiotemporal compression techniques for moving point objects. In: EDBT, pp 765–782
28. Millefiori LM, Braca P, Bryan K, Willett P (2015) Adaptive filtering of imprecisely time-stamped measurements with application to AIS networks. In: FUSION, pp 359–365
29. Moga A, Tatbul N (2011) UpStream: A storage-centric load management system for real-time update streams. PVLDB 4(12):1442–1445
30. O'Rourke J (1998) Computational Geometry in C cambridge university press
31. Pallotta G, Vespe M, Bryan K (2013) Vessel pattern knowledge discovery from AIS data: A framework for anomaly detection and route prediction. Entropy 15(6):2218–2245
32. Paschke A, Kozlenkov A (2009) Rule-based event processing and reaction rules. In: RuleML, LNCS 5858
33. Patroumpas K, Artikis A, Katzouris N, Vodas M, Theodoridis Y, Pelekis N (2015) Event recognition for maritime surveillance. In: EDBT, pp 629–640
34. Patroumpas K, Sellis T (2011) Maintaining consistent results of continuous queries under diverse window specifications. Inf Syst 36(1):42–61
35. Potamias M, Patroumpas K, Sellis T (2007) Online amnesic summarization of streaming locations. In: SSTD, pp 148–165
36. Przymusinski T (1987) On the declarative semantics of stratified deductive databases and logic programs. In: Found. of deductive databases and logic programming. Morgan
37. Shahir HY, Glasser U, Shahir AY, Wehn H (2015) Maritime situation analysis framework: Vessel interaction classification and anomaly detection. In: Big Data, pp 1279–1289
38. Skarlatidis A, Paliouras G, Artikis A, Vouros G (2015) Probabilistic event calculus for event recognition ACM Transactions on Computational Logic 16(2)
39. Snidaro L, Visentini I, Bryan K (2015) Fusing uncertain knowledge and evidence for maritime situational awareness via markov logic networks. Inf Fusion 21:159–172
40. Terroso-Saenz F, Valdes-Vela M, Skarmeta-Gomez AF (2015) A complex event processing approach to detect abnormal behaviours in the marine environment. Information Systems Frontiers 1–16
41. Wolfson O, Sistla A, Chamberlain S, Yesha Y (1999) Updating and querying databases that track mobile units. Distributed & Parallel Databases 7(3):257–287
42. Zhang H, Diao Y, Immerman N (2014) On complexity and optimization of expensive queries in complex event processing. In: SIGMOD, pp 217–228

Kostas Patroumpas is a Research Associate at IMIS/Athena Research Centre, a member of Knowledge & Data Base Systems Laboratory at the National Technical University of Athens, and also collaborates with the Information Management Lab at the University of Piraeus. He holds a Diploma in Computer Engineering from the University of Patras and a M.Sc. in Geoinformatics from the National Technical University of Athens. His research interests relate to data stream processing, trajectory data management, linked data, and geospatial visualisation. He has been involved in several national and EU/FP7 research projects, and he has published over 30 articles in international conferences and refereed journals.

Elias Alevizos is a Research Associate in the Institute of Informatics & Telecommunications at NCSR "Demokritos". He obtained a diploma in Electrical and Computer Engineering from the National Technical University of Athens and a M.Phil. in neuro-robotics from the University of Edinburgh.



Alexander Artikis is a Lecturer in the University of Piraeus and a Research Associate in NCSR "Demokritos". He holds a PhD from Imperial College London on the topic of norm-governed multi-agent systems. His research interests lie in the areas of artificial intelligence and distributed systems. He has been working in many international research projects and has the role of scientific coordinator in the SPEEDD project.

Marios Vodas is a PhD candidate at the Department of Informatics of the University of Piraeus in Greece under the supervision of Prof. Yannis Theodoridis. Born in 1988, he received his BSc degree (2011) and his MSc in Advanced Information Systems (2013) from the Department of Informatics of the University of Piraeus. His research interests include spatiotemporal data management and data mining.

Nikos Pelekis is Assistant Professor at the Department of Statistics and Insurance Science, University of Piraeus, Greece. Born in 1975, he received his B.Sc. degree from the Computer Science Department of the University of Crete (1998). He has subsequently joined the Department of Computation in the University of Manchester Institute of Science and Technology (UMIST) to pursue his M.Sc. in Information Systems Engineering (1999) and his Ph.D. in Moving Object Databases (2002). His research interests include data mining, spatiotemporal data management and machine learning. He has been particularly working for almost ten years in the field of Mobility Data Management and Mining, being the architect of the widely cited ?Hermes? Moving Object Database (MOD) engine. Nikos has coauthored one monograph and more than 50 refereed articles in scientific journals and conferences, receiving more than 500 citations, while he has received 3 best paper awards. He has offered several invited lectures in Greece and abroad (including PhD/MSc/summer courses at Rhodes, Milano, KAUST, Aalborg, Trento, Ghent, JRC Ispra) on Mobility Data Management and Data Mining topics. He is a reviewer in many international journals and conferences. He has been member of the Organizing Committee for ECML/PKDD 2011 and member of the Editorial Board of ECML PKDD 2014-, Data Mining and Knowledge Discovery journal. He has been actively involved in more than 10 European and National R&D projects. Among them he is or was principal researcher in GeoPKDD (FP6/IST, 2005-09), MODAP (FP7/ICT, 2009-12), MOVE (COST, 2009-13; substitute member of the mgmt committee), DATASIM (FP7/ICT, 2011-14) and SEEK (FP7/PEOPLE, 2012-15). For more information: http://www.unipi.gr/faculty/npelekis/

Yannis Theodoridis (Ph.D., National Technical University of Athens, Greece, 1996) is a Professor at the Dept. of Informatics, University of Piraeus. His research interests include database management, in particular mobility data management and exploration.