Event Recognition for Maritime Surveillance

Kostas Patroumpas[1,2], Alexander Artikis[3,4], Nikos Katzouris[4],
Marios Vodas[1], Yannis Theodoridis[1], Nikos Pelekis[5]
[1]Department of Informatics, University of Piraeus, Greece
[2]School of Electrical & Computer Engineering, National Technical University of Athens, Greece
[3]Department of Maritime Studies, University of Piraeus, Greece
[4]Institute of Informatics & Telecommunications, NCSR Demokritos, Athens, Greece
[5]Department of Statistics & Insurance Science, University of Piraeus, Greece
{kpatro, a.artikis, mvodas, ytheod, npelekis}@unipi.gr, nkatz@iit.demokritos.gr

ABSTRACT

We present a system that combines intelligent online tracking with complex event recognition against streaming positions relayed from numerous vessels. Given the vital importance of maritime safety to the environment, the economy, and in national security, our system leverages the real-time acquisition of vessel activity with geographical and other static information. Thus, it can offer timely notification in emergency situations, such as intrusion into marine preservation areas, loitering, and unsafe sailing. Thanks to a mobility tracking module, evolving trajectories generated by massive positional updates can be compressed online into concise, but reliable synopses per ship, retaining only salient motion features within a sliding window. These features are exploited by a complex event recognition module that detects suspicious situations of interest to maritime authorities. We conducted a comprehensive empirical validation against a real dataset of traces collected from thousands of vessels. Our results confirm the scalability and approximation accuracy of the proposed system, and thus demonstrate its potential for effective, real-time maritime monitoring.

1. INTRODUCTION

Maritime surveillance systems have been attracting attention both for economic and environmental reasons [1, 3, 16]. For instance, preventing ship accidents by monitoring vessel activity represents substantial savings in financial cost for shipping companies (e.g., oil spill cleanup) and averts irrevocable damages to maritime ecosystems (e.g., fishery closure). Nowadays, maritime navigation technology can automatically provide real-time information from sailing vessels. The Automatic Identification System (AIS) [24] is a tracking system for identifying and locating vessels at sea through data exchange: either with other ships nearby, or AIS base stations along coastlines, or even satellites when out of range of terrestrial networks. AIS is intended to assist vessel crews in collision avoidance and allows maritime authorities to monitor vessel movements. This technology integrates a VHF transceiver with a positioning device (e.g., GPS), and other electronic navigation sensors, such as a gyrocompass or rate of turn indicator.

AIS raw tracking data offers a wealth of information including unique identification of vessels, their position, course, and speed. Such information can be displayed on screen aboard of the ship or in maritime control centers. AIS-equipped vessels may be of diverse type, size, or tonnage. Not all of them relay their position simultaneously or at a fixed frequency, but depending on the transponder configuration aboard the ship, proximity to base stations, and the type of their motion. Vessels anchored or slowly moving transmit less frequently than those cruising fast in the open sea or manoeuvering near the docks. Note that this data is not noise-free; AIS messages may be delayed, intermittent, or conflicting.

Considering that AIS information is continuously emitted from over 400 thousand ships worldwide [2], it evidently fulfills all four '*V*' challenges (*Volume, Velocity, Variety, lack of Veracity*), as well as the '*D*' challenge (*Distribution of data sources*) in big data management. Therefore, for effective vessel identification and tracking, maritime surveillance systems need to scale to the increasing traffic activity witnessed in the past few years[1]. Such systems should detect threats and abnormal activity over voluminous, fluctuating, and noisy data streams from thousands of vessels, and also correlate them with static data expressing vessel characteristics (type, tonnage, cargo, etc.) and geographical information (such as bathymetric data and protected areas).

To address these requirements, we introduce a maritime surveillance system that consists of two main components. A *trajectory detection* component consumes a positional stream of AIS messages from a large fleet and tracks major changes along each vessel's movement. Thus, it can instantly identify "critical points" en route, indicating important changes like a stop, a sudden turn, or slow motion of a ship. Except for harsh weather conditions, traffic regulations, local manoeuvres in ports, etc., ships are expected to move along almost straight, predictable paths. Therefore, most of the frequently relayed positional messages are not really required for representing the actual trace of a vessel. Instead, by discarding superfluous locations along a "normal" course with a known velocity, we can approximately reconstruct each vessel's *trajectory* from the sequence of its critical points only. This online summarization achieves data compression close to 95%, incurring negligible loss in approximation accuracy. With such dramatic reduction in system load, execution of continuous and historical queries can be greatly improved, e.g., reducing latency of online collision detection or similarity search among recent vessel paths.

The detected critical points may be used for map display, but they are mostly valuable for recognition of complex phenomena and thus issuing alert notifications to marine authorities. To this end, a *complex event recognition* component combines the derived

[1] http://www.bbc.com/news/science-environment-28372461

10.5441/002/edbt.2015.63

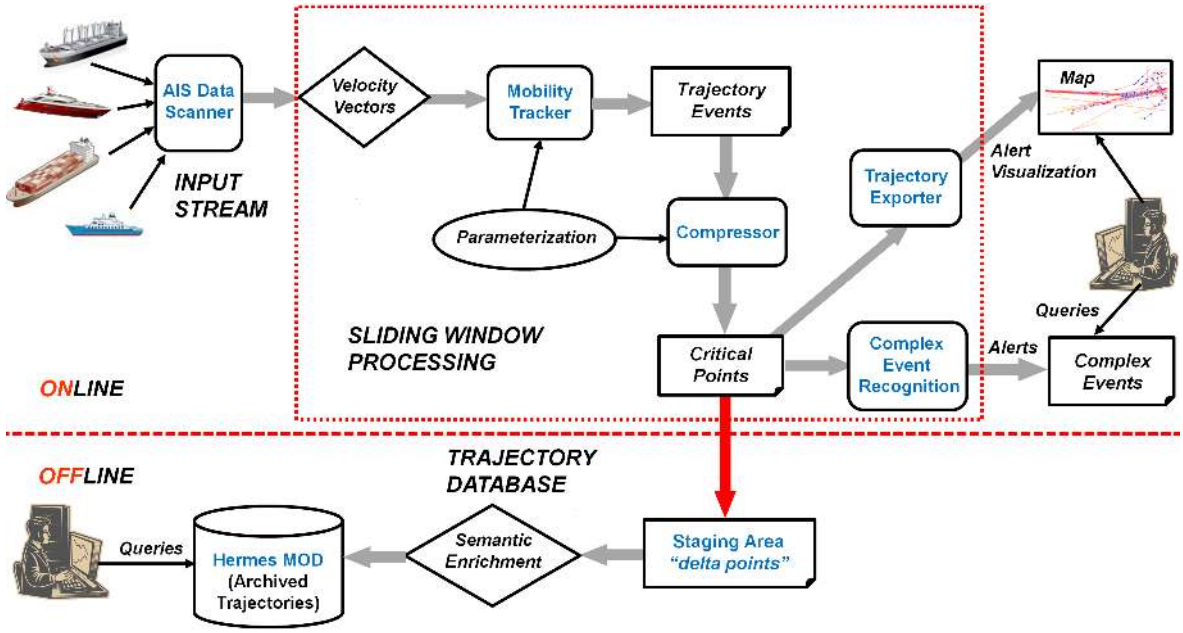


Figure 1: Processing scheme of the maritime surveillance system.

stream of critical points expressing vessel activity, with static geographical and vessel data, and can detect suspicious or potentially dangerous situations, such as loitering, vessels passing through protected areas, and unsafe shipping. In contrast to map display of current locations or information about specific vessels of interest [2], this module can be used to spot emergency situations in real-time.

The recognized complex events and lightweight trajectory synopses may be physically archived in a database for extracting *offline analytics*, including travel statistics, motion trends, origin–destination matrices, frequent routes, area and vessel classification (suspicious areas, illegal or dangerous shipping), and much more.

In this paper, we emphasize the real-time features of our maritime surveillance system, developed in the context of the AMINESS project [1]. This interdisciplinary project aims to reinforce safety and assist in the management of sea environments, particularly in the Aegean Sea. We conducted an extensive empirical validation of the proposed system on a large dataset of real vessel traces collected in the summer of 2009 from AIS base stations along the coastline of Greece. Our study confirms that the high compression ratio achieved by the trajectory detection component, along with the efficient pattern matching algorithms of the complex event recognition component, allow this system to scale to high velocity data streams expressing the current activity of large fleets.

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: the trajectory detection and complex event recognition modules. Section 5 reports performance results from a comprehensive evaluation of the implemented system. In Section 6, we compare our approach against related systems and methodologies. Finally, in Section 7 we summarize our work and outline directions for further research and implementation.

2. SYSTEM ARCHITECTURE

In this Section, we outline the processing flow of the proposed maritime surveillance system. As illustrated in Figure 1, the system consumes a stream of AIS tracking messages from vessels, detects important features that characterize their movement and recognizes complex events such as suspicious vessel activity. These results can be used to evaluate *continuous location-aware queries*, e.g., to detect whether a ship is approaching a port or if a vessel has just entered into an environmentally protected area.

In order to meet the real-time requirements of data stream processing, this online process necessitates the use of a *sliding window* [18, 29], which abstracts the time period of interest and keeps up with the evolving movement. Typically, a window looks for phenomena that occurred in a recent *range* ω (e.g., positions received during past 60 minutes). This window moves 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 (β) the distance traveled by a ship over the past hour (ω). Typically, it holds that $\beta < \omega$; so, as time goes by, successive window instantiations may share positional tuples over their partially overlapping ranges.

As input, we consider AIS messages of certain types (1, 2, 3, 18, 19) and extract position reports. 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 longitude/latitude coordinates (Lon, Lat) measured 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 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). In order to detect motion changes, the *Mobility Tracker* module maintains one velocity vector per vessel based on its two most recent positions[2]. Working entirely in main memory and without any index support, the Mobility Tracker checks when and how velocity changes with time. Thus, it can detect trajectory events, either instantaneous (e.g., a sudden turn) or of longer duration (e.g., a smooth turn). At each window slide, those events are properly filtered (e.g., from possible outliers) via a *Compressor*. Then, a sequence of *"critical"* points (such as a stop) is emitted, which are much fewer compared to the originally relayed positions. So, 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).

Moreover, the derived critical points are transmitted to the *Complex Event Recognition* module, which combines this event stream with static geographical and vessel data, such as bathymetric data and protected areas. The objective of this process is to detect potentially suspicious or dangerous situations, such as loitering, vessels passing through protected areas, and unsafe shipping. The recognized complex events are pushed in real-time to the end user (marine authorities) for real-time decision-making.

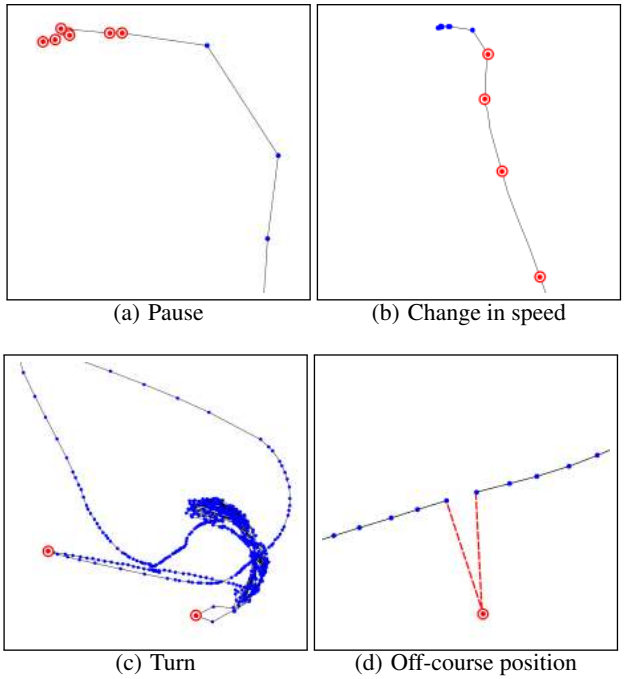
Finally, historical information can be progressively compiled from these detected features. "Delta" critical points (issued once the window slides forward) are periodically sent from main memory into a staging area on disk. An *offline* module accepts these lightweight, digested traces and reconstructs trajectory segments for archiving in Hermes Moving Objects Database (MOD) [30], instead of naïvely storing enormous quantities of raw AIS positions. This reconstruction process also identifies ships docked at ports, so that trajectory semantics can be enriched accordingly so as to extract further knowledge through motion mining or computation of statistics.

3. DETECTING TRAJECTORY EVENTS

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 turning point, or slow motion. Such *trajectory movement events* (ME) suffice to indicate "critical points" along the trace of each vessel and thus offer a concise, yet quite reliable representation of its course. 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. We distinguish two kinds of trajectory events:

- *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 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.

[2]Typically for trajectories [8], 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.



(a) Pause (b) Change in speed

(c) Turn (d) Off-course position

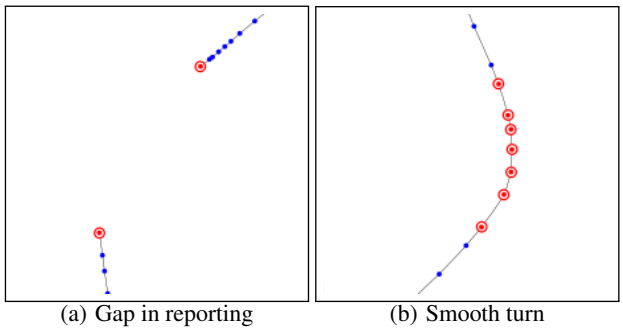
Figure 2: Instantaneous events and outliers in a vessel's course.

In this Section, we first describe how the sequence of vessel positions can be processed *online* in order to detect such trajectory events. An early version of the online tracking module was introduced in [28]. This has been substantially enhanced to identify additional events and much more robust in coping with increased data volumes. We also explain how the resulting critical points per vessel can provide a lightweight summary of its trajectory, which then offers many opportunities for affordable *offline* analysis.

3.1 Online Tracking of Moving Vessels

As illustrated in Figure 1, the system accepts fresh AIS messages from ships and extracts positional tuples $\langle MMSI, Lon, Lat, \tau \rangle$. In order to identify significant changes in movement, it first computes the instantaneous velocity vector $\overrightarrow{v}_{now}$ from the two most recent positions reported by each vessel $MMSI$. Then, the *mobility tracker* can instantly deduce a variety of *instantaneous* events by examining the trace of each vessel alone:

- *Pause* indicates whether a vessel is currently halted, once its instantaneous speed $\overrightarrow{v}_{now}$ does not exceed a suitable threshold v_{min}. For example, if $\overrightarrow{v}_{now}$ is currently less than $v_{min} = 1$ knot, then the ship rests practically immobile. For the vessel shown in Figure 2(a), the red bullets indicate several pause events; apparently, the ship is anchored at the port and such small displacements may be caused by GPS errors or sea drift.
- *Speed change* is issued once current v_{now} deviates by more than $\alpha\%$ from the previously observed speed v_{prev}. For a given threshold α, the formula $|\frac{v_{now} - v_{prev}}{v_{now}}| > \frac{\alpha}{100}$ indicates whether the vessel has just decelerated or accelerated. This is normally the case when a ship is approaching to or departing from a port, as depicted in Figure 2(b).



(a) Gap in reporting (b) Smooth turn

(c) Long-term stop (d) Slow motion

Figure 3: Long-lasting trajectory events.

- *Turn*: It occurs when the direction has changed by more than a given angle $\Delta\theta$, e.g., there is a difference of more than 15^o from its previous heading. Red bullets in Figure 2(c) illustrate two such sudden turns.

- *Off-course positions* incur a very abrupt change in vessel's velocity $\overrightarrow{v}_{now}$ (both in speed and heading). Yet, such an outlier can be easily detected as 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 2(d) illustrates such a case.

No critical point gets immediately issued upon detection of any such simple events. An instantaneous pause or turn may be coincidental and is not meaningful out of context, because a series of such events may signify that the ship is stopped for some time, as we will explain shortly. Besides, accepting an outlier would drastically distort the resulting trajectory representation, as the red dashed line in Figure 2(d) illustrates. Even worse, such noisy positions may affect detection of important events. 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. Thus, any off-course positions should be discarded as noise.

Those instantaneous events are used to detect spatiotemporal phenomena of some duration, i.e., *long-lasting trajectory events* like:

- *Gap* in reporting is issued when a vessel has not emitted a message for a time period ΔT, e.g., over the past 10 minutes. Therefore, its course is unknown during this period, as it occurs between the two red bullets in Figure 3(a). Reporting such a critical point (i.e., when the gap started) is important, not only for properly monitoring vessels, but also for safety reasons, e.g., a suspicious move near maritime boundaries, or a potential intrusion of a tanker into a marine park.

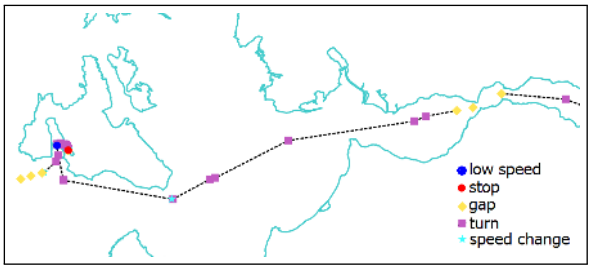


Figure 4: Critical points identified along a vessel trajectory.

- *Smooth turn* can be identified by checking whether the cumulative change in heading across a series of previous positions exceeds a given angle $\Delta\theta$, as illustrated with the red points in Figure 3(b). Then, the latest of them is emitted as a critical (turning) point.

- *Long-term stop* occurs when at least m consecutive instantaneous pause or turn events are found within a predefined radius r (e.g., 200 meters). In Figure 3(c), the red points inside the circle succeed one another and indicate such immobility, so they could be collectively approximated by a single critical point (their centroid) with their total duration.

- *Slow motion* means that the vessel consistently moves at low speed ($\leq v_{min}$) over its m last messages, as in Figure 3(d). In contrast to a stop event, these successive locations usually occur along a path and not all of them fall in a small circle. The median of these m positions is reported as a representative critical point.

Thus, critical points are emitted from each long-lasting event. Provided that they do not qualify for outliers, instantaneous events for speed change (Figure 2(b)) or isolated turns (Figure 2(c)) also contribute to critical points. The example trajectory in Figure 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 [1]), 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 $\Delta\theta = 15^o$ incurs a 10% increase in the amount of critical points, because more original 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 the aggressive parametrization listed in Table 3, which yields quite tolerable accuracy. With relaxed parameter values, additional events can be detected, capturing slighter changes in 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 $O(m)$, where m is the number of latest positions that need inspection. As m is usually a small integer (we set $m = 10$ in our experiments), this cost is affordable.

Rules for such trajectory events are suitably defined in the mobility tracker, which is equipped with robust data structures for in-memory maintenance of movement features. Note that more 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 additional features, such as traveled distance

from a given origin (e.g., a port). Better coping with noisy situations is also a challenge, e.g., ignoring *delayed positions* that erroneously mark a ship moving back and forth along its course. Nonetheless, even with this set of events, 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 later in Section 4.

3.2 Trajectory Reconstruction

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 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. This summarization depends on the type of detected movement events (i.e., stop, low speed, turn, gap, speed change), so as to refresh each trajectory accordingly. This main-memory process affects trajectory portions currently within the sliding window.

But we also incrementally update trajectories in the underlying database with *"delta" points*. Once the window slides forward, expiring critical points are transferred in an intermediate *staging table* on disk. So, this table temporarily records all recent "delta" changes, i.e., critical points evicted from the window, but not yet admitted in disk-based trajectories. Obviously, information archived in the database is always lagged behind the current vessel reports by ω. This is a deliberate decision dictated for data consistency reasons, so as to avoid having any trajectory portions duplicated in memory (online) and on disk (offline).

Offline trajectory reconstruction in Hermes MOD [30] periodically converts a sequence of critical points per ship into disjoint, consecutive trajectory segments. In particular, a long journey breaks up into smaller trips between ports. This segmentation reduces the latency for offline queries in the database, and also decreases the cost of trajectory updates on every batch of critical points. Eventually, instead of representing the entire motion of a vessel with one long trajectory that gets repetitively updated, Hermes MOD deals with multiple, but much smaller segments; only the last segment per vessel may receive any updates.

AIS messages sometimes include information regarding the destination of sailing vessels. Unfortunately, after scrutinizing AIS data samples, we concluded that this voyage-related information is often missing or error-prone, mainly because it is updated manually by the crew. So, we employ an automated procedure for performing *semantic enrichment* of trajectories with added value information about trips between ports. This method takes as input the critical points identified as long-term stops and a set of known port areas (polygons). Once a stop is located inside such a polygon, the name of the respective port becomes an attribute of that point. It is reasonable to assume that between two such distinct stops O and D, the ship sailed from origin port O and reached destination port D. Then, this is identified as a new trip with a known destination D. Note that origin port O may remain unknown, because the ship might have been on the move when the AIS base stations started receiving its signals. Of course, as each vessel continues sailing, more and more critical points will be detected. However, as long as a specific destination port is not yet identified, these points will be piling up in the staging table awaiting assignment to a trajectory.

3.3 Offline Trajectory Analysis

Once reconstructed trajectories get stored in Hermes MOD, useful statistics and patterns can be extracted from them in an *offline* fashion. Since the focus of this paper is on online processing, we only give a brief overview of such analytics. First, a series of *derived tables* can offer historical information about traveled distances and travel times per ship, idle periods at dock, visited ports, etc. Such aggregates may be obtained at various time granularities (e.g., per week, month, or year) and may be computed by other dimensions as well (e.g., flag, cargo, vessel type, etc.). By maintaining *Origin-Destination matrices*, we may identify connections between ports and compute aggregated statistics (duration, speed, frequency, etc.) for such itineraries, often varying by ship type, period of the year, etc. In addition, *motion patterns* can be identified, such as frequently traveled paths ("corridors"), periodicity in movement (e.g., ferries between two specific ports), etc. Hermes MOD incorporates an algorithm for *spatiotemporal clustering*, which can help exploring periodicity of trips. Indeed, two (or more) trajectory clusters may be almost identical spatially, but they are distinct because the temporal dimension is taken into consideration when calculating distances between pairs of trajectory segments.

4. COMPLEX EVENT RECOGNITION

The critical Movement Events (ME) computed by the trajectory detection component are transmitted to the *Complex Event Recognition* module. This module correlates the derived stream of MEs with static geographical and vessel information, such as bathymetric data and locations of protected areas, to detect potentially suspicious and dangerous situations, such as forbidden fishing or unsafe shipping. When recognized, such Complex Events (CE) are forwarded to the marine authorities for real-time decision making.

Our CE recognition component is based on the Event Calculus for Run-Time reasoning (RTEC) [5, 6]. The Event Calculus [17] is a logic programming language for representing and reasoning about events and their effects. The benefits of a logic programming approach to CE recognition are well-documented [26]: such an approach has a formal, declarative semantics, and direct routes to machine learning for constructing CE definitions in an automated way. The use of the Event Calculus has additional advantages: the process of CE definition development is considerably facilitated, as the Event Calculus includes built-in rules for complex temporal representation and reasoning, including the formalization of *inertia*. With the use of the Event Calculus, one may develop intuitive, succinct CE definitions, facilitating the interaction between CE definition developer and domain expert (marine authorities), and allowing for code maintenance. In this Section, we present RTEC following [5, 6], and illustrate its use for maritime surveillance.

4.1 Representing Maritime Activities

The time model of RTEC is linear and includes integer time-points (such as the timestamps of the MEs computed by the trajectory event detection component). Variables start with an upper-case letter, while predicates and constants start with a lower-case letter. Where F is a *fluent*—a property that is allowed to have different values at different points in time—the term $F = V$ denotes that fluent F has value V. Boolean fluents are a special case in which the possible values are true and false. holdsAt$(F = V, T)$ represents that fluent F has value V at a particular time-point T. holdsFor$(F = V, I)$ represents that I is the list of the maximal intervals for which $F = V$ holds continuously. holdsAt and holdsFor are defined in such a way that, for any fluent F, holdsAt$(F = V, T)$ if and only if T belongs to one of the maximal intervals of I for which holdsFor$(F = V, I)$.

The happensAt predicate represents an instance of an event type. E.g., happensAt$(turn(vessel_1), 5)$ represents the occurrence of event type $turn(vessel_1)$ at time 5. When it is clear from the con-

text, we do not distinguish between an event and its type. 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 values of the fluents with the use of the holdsAt and holdsFor predicates, as well as other, possibly atemporal, constraints. Table 1 presents the predicates available to the event description developer.

We represent instantaneous MEs and CEs by means of happensAt, while durative MEs and CEs are represented as fluents. In the maritime surveillance setting, the stream of critical MEs, which constitutes the input of RTEC, consists of both instantaneous MEs, such as $speedChange(Vessel)$, and durative ones, such as $stopped(Vessel) = \mathsf{true}$ indicating the maximal intervals during which a $Vessel$ is considered stopped. The majority of CEs are durative and, therefore, in CE recognition the task generally is to compute the maximal intervals for which a fluent representing a CE has a particular value continuously.

For a 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 *intervals* I for which $F = V$, that is, $\mathsf{holdsFor}(F = V, I)$, 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 'broken'. The time-points at which $F = V$ is initiated are computed by means of domain-specific initiatedAt rules. The time-points at which $F = V$ is 'broken' are computed as follows:

$$\begin{aligned}&\mathsf{broken}(F = V,\ T_s,\ T) \leftarrow \\ &\quad \mathsf{terminatedAt}(F = V,\ T_f),\ \ T_s < T_f \leq T\end{aligned} \tag{1}$$

$$\begin{aligned}&\mathsf{broken}(F = V_1,\ T_s,\ T) \leftarrow \\ &\quad \mathsf{initiatedAt}(F = V_2,\ T_f),\ \ T_s < T_f \leq T,\ \ V_1 \neq V_2\end{aligned} \tag{2}$$

$\mathsf{broken}(F = V,\ T_s,\ T)$ represents that $F = V$ is terminated at some time T_f such that $T_s < T_f \leq T$. Similar to initiatedAt, terminatedAt rules are domain-specific (examples are presented below). According to rule (2), if $F = V_2$ is initiated at T_f then effectively $F = V_1$ is terminated at time T_f, for all other possible values V_1 of F. Thus, rule (2) ensures that a fluent cannot have more than one value at any time. We do not insist that a fluent must have a value at every time-point. There is a difference between initiating a Boolean fluent $F = \mathsf{false}$ and terminating $F = \mathsf{true}$: the former implies, but is not implied by, the latter.

In what follows, we illustrate the use of RTEC for CE representation in the maritime domain.

Scenario 1. In maritime surveillance, it is necessary to detect areas in which vessel activity is suspicious. Below is the formalization of one type of suspicious activity:

$$\begin{aligned}&\mathsf{initiatedAt}(suspicious(Area) = \mathsf{true},\ T) \leftarrow \\ &\quad \mathsf{happensAt}(\mathsf{start}(stopped(Vessel) = \mathsf{true}),\ T), \\ &\quad \mathsf{holdsAt}(coord(Vessel) = (Lon, Lat),\ T), \\ &\quad close(Lon, Lat, Area), \\ &\quad \mathsf{holdsAt}(vesselsStoppedIn(Area) = N,\ T),\ N > 3 \\ &\mathsf{terminatedAt}(suspicious(Area) = \mathsf{true},\ T) \leftarrow \\ &\quad \mathsf{happensAt}(\mathsf{end}(stopped(Vessel) = \mathsf{true}),\ T), \\ &\quad \mathsf{holdsAt}(coord(Vessel) = (Lon, Lat),\ T), \\ &\quad close(Lon, Lat, Area), \\ &\quad \mathsf{holdsAt}(vesselsStoppedIn(Area) = N,\ T),\ N \leq 3\end{aligned} \tag{3}$$

$stopped(Vessel) = \mathsf{true}$ is a durative ME stating that $Vessel$ has stopped. $\mathsf{start}(F = V)$ (respectively $\mathsf{end}(F = V)$) is a built-in RTEC event taking place at each starting (ending) point of each maximal interval for which $F = V$ holds continuously. Along with each vessel critical ME, the trajectory event detection system provides the coordinates (Lon, Lat) of the vessel. These are represented in RTEC by the $coord$ fluent. $close(Lon, Lat, Area)$ is an atemporal predicate calculating whether the Haversine distance between a point (Lon, Lat) and an $Area$ is less than some predefined threshold. Fluent $vesselsStoppedIn(Area)$ records the number of vessels that have stopped in this $Area$ at some point in time.

Table 1: Main predicates of RTEC.

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

According to rule-set (3), an $Area$ is said to be suspicious as long as at least four vessels have stopped close to, or in it. The value of four vessels was set by domain experts. Usually, officials monitoring vessel activity are familiar with potentially suspicious areas, such as areas where loitering takes place, and thus restrict[3] computation of the maximal intervals of the $suspicious$ fluent to these areas. The maximal intervals during which $suspicious(Area) = \mathsf{true}$ holds continuously are computed using the built-in RTEC predicate holdsFor from rule-set (3).

$\mathsf{initiatedAt}(F = V, T)$ does not necessarily imply that $F \neq V$ at T. Similarly, $\mathsf{terminatedAt}(F = V, T)$ does not necessarily imply that $F = V$ at T. Suppose that $F = V$ is initiated at time-points 10 and 20 and terminated at time-points 25 and 30 (and at no other time-points). In that case $F = V$ holds at all T such that $10 < T \leq 25$. Note that, in this case, the event $\mathsf{start}(F = V)$ takes place at 10 and at no other time-point, while the event $\mathsf{end}(F = V)$ takes place at 25 and at no other time-point.

CE recognition for maritime surveillance requires reasoning over streaming data, such as the MEs reported by the trajectory event detection system, as well as atemporal reasoning [14]. In rule-set (3), for instance, we had to compute the Haversine distance between a point and an area. Unlike various other CE recognition approaches, such as [12, 18, 9], which lack the ability of (complex) reasoning over static/domain knowledge, RTEC combines event pattern matching over event streams with atemporal reasoning.

Scenario 2. Marine authorities are often interested in detecting illegal fishing. Below is a formalization of two of the conditions in which illegal fishing starts being recognized:

$$\begin{aligned}&\mathsf{initiatedAt}(illegalFishing(Area) = \mathsf{true},\ T) \leftarrow \\ &\quad \mathsf{happensAt}(\mathsf{start}(stopped(Vessel) = \mathsf{true}),\ T), \\ &\quad fishing(Vessel), \\ &\quad \mathsf{holdsAt}(coord(Vessel) = (Lon, Lat),\ T), \\ &\quad close(Lon, Lat, Area) \\ &\mathsf{initiatedAt}(illegalFishing(Area) = \mathsf{true},\ T) \leftarrow \\ &\quad \mathsf{happensAt}(slowMotion(Vessel),\ T), \\ &\quad fishing(Vessel), \\ &\quad \mathsf{holdsAt}(coord(Vessel) = (Lon, Lat),\ T), \\ &\quad close(Lon, Lat, Area)\end{aligned} \tag{4}$$

[3] Such a restriction is achieved through the 'declarations' facility of RTEC.

fishing is an atemporal predicate indicating fishing vessels. In addition to having a database of *fishing* facts, this predicate may compute whether a vessel (not recorded in the database) is a fishing one given its characteristics. $slowMotion(Vessel)$ is an ME indicating that $Vessel$ was moving 'too' slowly at some point in time (see Section 3). The computation of the maximal intervals during which $illegalFishing(Area) = \text{true}$ holds continuously is restricted to areas in which fishing is forbidden. According to rule-set (4), illegal fishing starts being recognized when a fishing vessel stops or moves 'too' slowly close to, or in an area in which fishing is forbidden.

Illegal fishing stops being recognized when there are no fishing vessels in the forbidden fishing area, or when their movement does not allow for fishing. The termination rules are formalized similar to the rules already shown and are therefore omitted to save space.

Scenario 3. A common feature of illegal shipping is communication gap; vessels with illegal activity, such as those passing through protected areas in order to minimize the length of a trip and therefore fuel consumption, switch off their transmitters and stop sending position signals. In such cases, it is often claimed that the transmitter temporarily broke down. To capture this type of activity, we defined the rule below:

$$\begin{aligned} &\text{happensAt}(illegalShipping(Area),\ T) \leftarrow \\ &\quad \text{happensAt}(gap(Vessel),\ T), \\ &\quad \text{holdsAt}(coord(Vessel) = (Lon, Lat),\ T), \\ &\quad close(Lon, Lat, Area) \end{aligned} \tag{5}$$

$gap(Vessel)$ is an ME—the trajectory detection component reports such an ME when the $Vessel$ stops sending signals, i.e. when the communication gap starts. The computation of the time-points of the $illegalShipping(Area)$ events is restricted to protected areas, such as the National Marine Park of Alonnisos in the Aegean sea. According to rule (5), $illegalShipping(Area)$ is recognized when a vessel stops reporting position signals close to a protected $Area$.

Scenario 4. Ships sometimes approach inadvertently or intentionally (to reduce the length of a trip and fuel consumption) 'too' shallow waters. To alert marine authorities and prevent accidents resulting from such type of shipping, we formalized the rule below:

$$\begin{aligned} &\text{happensAt}(dangerousShipping(Area),\ T) \leftarrow \\ &\quad \text{happensAt}(slowMotion(Vessel),\ T), \\ &\quad shallow(Area, Vessel), \\ &\quad \text{holdsAt}(coord(Vessel) = (Lon, Lat),\ T), \\ &\quad close(Lon, Lat, Area) \end{aligned} \tag{6}$$

$shallow$ is an atemporal predicate indicating whether some waters are 'too' shallow for a vessel. Similar to the *fishing* predicate, in addition to having a database of $shallow$ facts, we may compute whether an area is 'too' shallow for a vessel (not recorded in the database) given the vessel's characteristics. In rule (6), we chose the $slowMotion$ ME as a condition for the recognition of dangerous shipping. Similarly, we may add rules with other types of vessel movement for recognizing this CE.

4.2 Recognizing Maritime Activities

CE recognition may be performed retrospectively—e.g., at the end of each day in order to evaluate the activity of a particular fleet of vessels. Typically, though, CE recognition has to be efficient enough to support real-time decision-making, and scale to very large numbers of MEs and CEs. MEs may not necessarily arrive at the CE recognition system in a timely manner, i.e. there may be a (variable) delay between the time at which MEs take place and the time at which they arrive at the CE recognition system.

RTEC performs CE recognition by computing and storing the maximal intervals of fluents and the time-points in which events occur. CE recognition takes place at specified query times $Q_1, Q_2, \ldots$. At each Q_i the MEs that fall within a specified sliding window ω ("working memory" in the terminology of RTEC) are taken into consideration. All MEs that took place before or at $Q_i - \omega$ are discarded. This is to make the cost of CE recognition dependent only on range ω and not on the complete ME history.

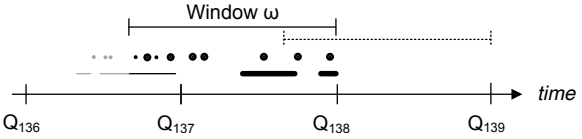


Figure 5: Complex event recognition in RTEC.

Window parameters for range ω and slide step β are specified by the users along with the definition of events, since they actually reflect the time horizon over which interesting phenomena shall be detected. Usually, range ω could span many minutes or even several hours for capturing meaningful events across a vessel's route.

At Q_i, the maximal 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. This is illustrated in Figure 5. Occurrences of MEs are displayed as dots and a Boolean fluent as line segments. For CE recognition at Q_{138}, only the events marked in black are considered, whereas the greyed out events are neglected. Assume that all events marked in bold arrived only after Q_{137}. Then, we observe that two MEs were delayed, i.e., they occurred before Q_{137}, but arrived only after Q_{137}. In our setting, the window range ω is larger than the slide step. Hence, these events are not lost but considered as part of the recognition process at Q_{138}.

In the common case that MEs arrive at RTEC with delays, it is thus preferable to make the range of ω longer than the slide step. Note that information may still be lost. Any MEs arriving between Q_{i-1} and Q_i are discarded at Q_i if they took place before or at $Q_i - \omega$. To reduce the possibility of losing information, one may increase the window range ω. But doing so, decreases recognition efficiency. This issue is illustrated in the following section.

5. EMPIRICAL EVALUATION

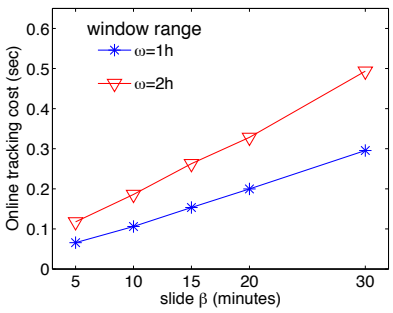
Our maritime surveillance system has a modular design with loosely coupled components. The online component for trajectory detection is developed in GNU C++ and runs entirely on main memory for efficiently coping with massive, volatile, streaming locations. RTEC, the CE recognition component, is implemented in YAP Prolog[4]. Built on top of PostgreSQL, Hermes MOD[5] accepts feeds of critical points from a Java wrapper and performs offline processing through SQL queries and stored procedures.

We conducted experiments against a real AIS dataset obtained from IMIS Hellas[6], our partner in the AMINESS project. Raw data is 23GB in size and spans from 1 June 2009 to 31 August 2009 for $N = 6425$ 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).

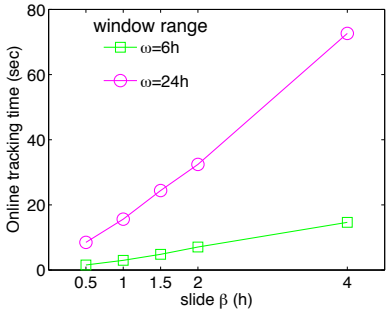
[4]The source code of RTEC, along with several sample CE definitions, is available at `http://users.iit.demokritos.gr/~a.artikis/EC.html`.

[5]Available at `https://hermes-mod.java.net/`

[6]`http://www.imishellas.gr/`



(a) Small window ranges



(b) Large window ranges

Figure 6: Online mobility tracking cost per window.

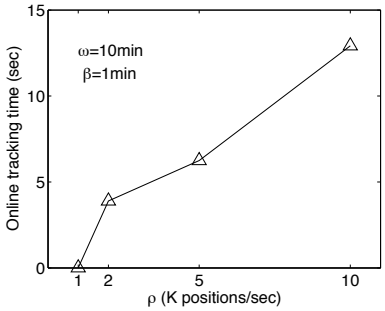


Figure 7: Varying arrival rates.

Table 2: Experimental settings.

Parameter	Value
Vessel count N	6425
Window range ω	10min, 1h, 2h, **6h**, 9h, 24h
Window slide β	1min, 5min, 10min, 15min, 20min, 30min, **1h**, 90min, 2h, 4h
Stream arrival rate ρ (positions/sec)	**original**, 1K, 2K, 5K, 10K

Table 3: Mobility tracking parameters.

Parameter	Value
Minimum speed v_{min} for asserting movement	1 knot (≅1.852 km/h)
Rate of speed change α	25%
Minimum gap period ΔT	10 minutes
Turn threshold $\Delta\theta$	5^o, 10^o, **15^o**, 20^o
Radius r for long-term stops	200 meters
Minimal number m of inspected positions	10

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 timestamped positions[7].

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. On average, each vessel reports its position once every 2 minutes, by considering only its activity period (i.e., when it actually relays positions, either moving or not). This translates into a mean arrival rate ρ of about 50 positions/sec. For consistency with the real-world scenario, we consume the original stream "as is" in all our simulations, with the exception of one test conducted at artificially increased rates. Simulation settings are listed in Table 2, whereas calibrated settings for online mobility tracking are given in Table 3; default values are shown in bold.

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 two Intel Xeon X5675 processors at 3.07GHz. This machine has 48GB of RAM, and five hard disks at 15K RPM with RAID 0 and a total capacity of 3TB.

[7]This anonymized data (with MMSIs replaced by sequence numbers) is publicly available at `http://www.chorochronos.org/?q=content/imis-3months`

The complex event recognition component operated on a computer with Intel i7-4770@3.40GHz×8 processors and 16GiB RAM, running Ubuntu Linux 14.04 and YAP Prolog 6.2.2.

5.1 Assessment of Trajectory Detection

Performance of online tracking. The first set of experiments examines performance of online mobility tracking for window specifications with varying ranges ω and slide steps β. Figure 6 illustrates the execution cost for the entire fleet per window, i.e., how much time it takes to update the window with fresh locations, evict expired ones, detect trajectory events, and report critical points. All values are averages over the total count of window instantiations, so they represent the per slide cost for window maintenance and identification of any trajectory events therein. Simulations with the original arrival rate reveal that critical points are issued almost instantly for small ranges up to two hours (Figure 6(a)). Not surprisingly, this cost escalates linearly when the window slides forward less often (larger β), because the mobility tracker must check many more fresh positions spanning a wider period β. Yet, processing positional batches arrived over the past $\beta = 30$ minutes never takes more than 500ms for small window ranges. The same linear pattern in online tracking cost repeats with wider ranges (Figure 6(b)), but it takes more time to complete upon each slide. In the worst case of a window spanning 24 hours, critical points are reported in only 72 seconds based on the bulk of data accumulated over each 4-hour period, which clearly demonstrates the robustness of this method.

One might argue that such performance results should be expected, given the low rate of the original stream. For a more stringent assessment of the online mobility tracking module, we performed an extra simulation, by admitting bigger chunks of data for processing at considerably increased arrival rates up to ρ =10,000 positions/sec. Given the fleet size N, every ship appears as reporting almost twice per second. This is quite improbable in practice, but makes sense as a stress test. As our objective is timeliness, the window was set to span $\omega = 10$ minutes and to slide each minute. In Figure 7, observe that critical points are still issued promptly for $\rho = 1,000$ positions/sec, but the latency grows with increasing rates. Reporting cost for critical points (i.e., cost after detection) is included in these times, and this becomes a significant overhead when massive AIS updates inevitably generate more critical points. For $\rho = 10,000$ positions/sec, the online tracker has to deal with 600,000 fresh positions every $\beta = 1$ minute, which is undoubtedly a demanding task. Nonetheless, it never takes more than a few seconds to respond, well before the next window slide. This behavior confirms that the trajectory detection process is capable of handling scalable volumes of streaming vessel positions.

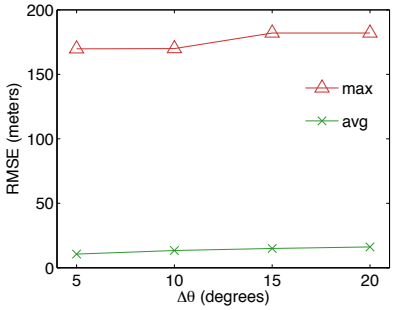


Figure 8: Trajectory approximation error.

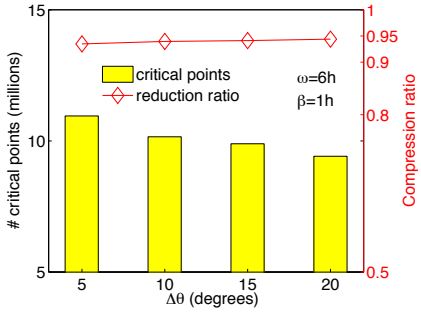


Figure 9: Compression for varying $\Delta\theta$.

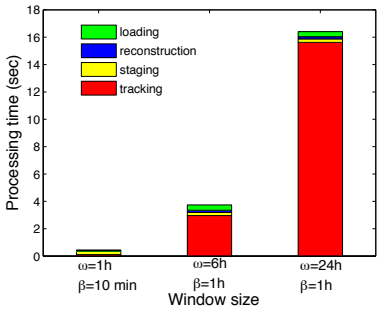


Figure 10: Trajectory maintenance cost.

Approximation error. Apart from performance, we also assessed the quality of trajectories approximately reconstructed from critical points only. For the entire motion history of each vessel, we estimated deviation between the original trajectory and its approximate representation (i.e., after compression). Deviation between two polylines can be computed from the pairwise distance of their corresponding points; in our case, between each pair of *synchronized* locations. Suppose that an original AIS point p_i did not qualify as critical and was discarded at timestamp τ_i. To estimate the resulting deviation for a particular vessel, we interpolated between the pair of adjacent critical points retained immediately before and after each such p_i. Assuming a constant velocity between these two critical points, we obtained its time-aligned point trace p'_i along the approximate path at timestamp τ_i. Thus, the compressed trajectory retained its approximate shape, but with additional (synchronized) vertices that account for the evicted positions. Assuming that a vessel originally reported M positions, we estimated the root mean square error ($RMSE$) between original and synchronized sequences of its locations using this formula:

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

where H denotes the Haversine distance between geographic coordinates, and returns RMSE estimates in meters. One error value was computed per vessel trajectory; Figure 8 plots the *average* and *maximum* $RMSE$ of them for several values of turn threshold $\Delta\theta$, which is used to recognize significant changes in heading. As discussed in Section 3.1, the degree of trajectory approximation is sensitive mostly to angle $\Delta\theta$ compared to other parameters in Table 3. In our tests, average error along complete trajectories (i.e., the entire motion history of each vessel) never exceeds 16 meters. This is negligible compared to the much larger size of most ships, and also considering the discrepancies inherent in GPS positioning. The maximum $RMSE$ ever observed is 182 meters, under a relaxed sensitivity of $\Delta\theta = 20^o$ for capturing important turns only. Although such a threshold is rather wide for actual monitoring of maritime activity, the worst error among all trajectories is comparable to the length of large ships. So, it turns out that the online tracking component provides quite acceptable accuracy and can capture most, if not all, critical changes along each vessel's course.

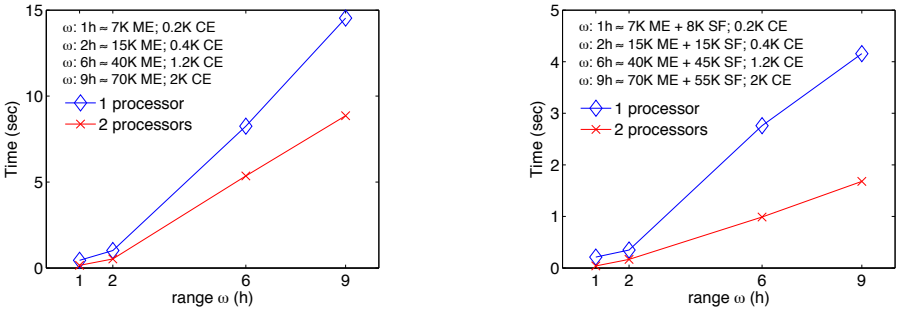
Compression efficiency. In this experiment, we examine the efficiency of our prototype in keeping only major trajectory characteristics as critical points and discard the rest. In order to measure the *compression ratio* accomplished by online trajectory tracking, we compared the amount of discarded 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 line plot in Figure 9 depicts measurements of this ratio with varying tolerance angles for detecting heading changes. With a lower $\Delta\theta$, even slight deviations in vessel direction can be spotted, and thus extra critical points get reported. From the bar plot in Figure 9, we notice that every further increase by 5^o in turn threshold $\Delta\theta$ results in about 5% drop in the total amount of critical points. So, relaxing the parameter values leads to a slightly less intense compression. However, compression ratio remains close to 94%, which means that about 6% of the original locations only survive as critical. In a streaming context, such high compression may lead to reduced system load in subsequent stages of the analysis, and we stress that it comes without significant loss in quality, as discussed earlier.

Table 4: Statistics from compressed trajectories.

Critical points in reconstructed trajectories	7,776,947
Critical points remaining in staging area	2,524,925
Number of trips between ports	68,501
Average trips per vessel	30
Average number of critical points per trip	113
Average travel time per trip	1 day 07:20:58
Average traveled distance per trip	221.976km

Trajectory maintenance. Since the system accepts streaming positions from vessels and manages to maintain their historical trajectories, we now provide some evidence of its overhead. Figure 10 plots the average processing cost per window slide for all four phases. Evidently, online mobility *tracking* undertakes the hardest task by filtering the huge volume of incoming positions, hence it dominates the trajectory maintenance cost especially for greater window sizes. As argued before, tracking time escalates with the window range, and also increases linearly when the window slides less often. A batch of "delta" critical points evicted from the sliding window is transferred into the *staging* area (on disk) in less than 260ms. This cost does not fluctuate a lot, as it mainly depends on database connection tunnelling via the Java wrapper. Trajectory *reconstruction* into trips between ports is also quite efficient and takes at most 163ms per batch of critical points. This offline module has to consider a drastically reduced amount of critical points per vessel instead of the voluminous dataset of raw positions, hence it incurs little overhead. In the last stage of *loading*, trajectory segments are inserted or updated in Hermes MOD, and this is also fast (390ms in the worst case), thanks to the reduced data volumes involved.

In Table 4, we list representative statistics from trajectories reconstructed and archived in the database. This computation took place after the input stream was exhausted and all critical points



(a) Input: critical movement events (ME).

(b) Input: critical movement events (ME) and spatial facts (SF).

Figure 11: Complex event recognition for 6,425 vessels and 35 areas.

were detected for the entire 3-month period. An outstanding benefit is that, with only a moderate amount of points, we can approximately describe trips spanning more than a day and covering long distances. Note that about 25% of critical points have not been assigned into any trajectory. These come from vessels still sailing and having not yet reached their destination port ("open-ended" trips). Besides, the number of trips is an order of magnitude greater than fleet size N. This certainly increases the amount of records that must be stored in the trajectory database, but thanks to their semantic enrichment with port information, such shorter geometry representations are more preferable at query time than protracted sequences of featureless locations.

5.2 Complex Event Recognition Performance

The task of the complex event (CE) recognition module is to detect activities of special significance for maritime surveillance, given the critical movement events (ME) produced by the trajectory event detection component. The input of RTEC—our CE recognition module—consists of the MEs (communication) gap, $lowSpeed$, $stopped$, $speedChange$ and $turn$, as well as the coordinates of each vessel at the time of ME detection. Given such an event stream, RTEC recognizes suspicious vessel activity (several vessels stopped in some area), illegal fishing, illegal shipping (passing through protected areas) and dangerous shipping (sailing through shallow waters). The choice of CEs and their definitions (see Section 4) were specified in collaboration with the domain experts of the AMINESS project. To allow for the recognition of the aforementioned CEs, we enhanced the input of RTEC with static synthetic data. For each vessel we added information about its draft, while a number of vessels were designated as fishing vessels. Moreover, we generated 35 polygons representing protected areas, forbidden fishing areas, and areas with shallow waters.

Figure 11(a) shows the results of two sets of experiments. First, we used a single processor to perform CE recognition for all 6,425 vessels and 35 areas. Second, we used two processors of the computer on which RTEC operated in parallel. One processor performed CE recognition for the areas located in, and the vessels passing through the west part of the area under surveillance. Similarly, the other processor performed CE recognition for the areas located in, and the vessels passing through the east part of the monitored area. Figure 11(a) shows average CE recognition times in CPU seconds. The slide β is 1 hour, including approximately 7,000 MEs. The window range ω varies from 1 hour ($\approx$ 7,000 MEs) to 9 hours ($\approx$ 70,000 MEs). In the distributed setting—two processors for CE recognition—the input MEs are forwarded to the appropriate processor (according to vessel location). The number of CEs also depends on the window range. For $\omega = 1$ hour, approximately 200 CEs are recognized, while for $\omega = 9$ hours RTEC recognizes approximately 2,000 CEs.

Figure 11(a) shows that we can achieve a significant performance gain by running RTEC in parallel. The input to each processor is restricted to the MEs of the vessels for which it performs CE recognition. Furthermore, each processor has to compute and store the maximal intervals of a smaller number of CEs. One may further distribute CE recognition by dividing further the monitored area, thus reducing CE recognition times. Figure 11(a) also shows that RTEC is capable of supporting real-time CE recognition. E.g., for a window ω of 6 hours, RTEC recognizes all CEs requested by end users in 8 sec when a single processor is used, and in 5 sec when two processors are used in parallel.

Note that, since the input stream consists of *critical* MEs, most of them fire the CE definition rules. This is in contrast to other experiments [5, 6] where the input stream includes several events that do not affect CE recognition. CE recognition performance does not depend entirely on the size of the input data streams. The complexity of the CE definitions affects recognition times significantly. In this work, we make use of quite complex CE definitions including various constraints on vessels and areas. This is in contrast to the majority of the event processing literature where quite simple CE definitions are used for empirical analysis.

Figure 11(b) shows average CE recognition times without spatial reasoning. More precisely, the ME stream is augmented by timestamped facts indicating the spatial relations between vessels and (protected, forbidden fishing, shallow) areas. Each ME expressing the movement of a vessel is accompanied by facts stating whether the vessel is 'close' to some area of interest—the timestamp of these facts is the same as the timestamp of the ME. For these experiments, the CE definitions were updated in order to make use of spatial facts (as opposed to RTEC computing on-demand spatial relations in the CE recognition process). Figure 11(b) shows results concerning CE recognition by single processor, as well as CE recognition performed by two processors in parallel. The slide β is 1 hour. In this setting, however, 1 hour of data includes approximately 15,000 input facts: 7,000 MEs and 8,000 spatial facts. Moreover, the window range ω varies from 15,000 MEs and spatial facts (1 hour) to 125,000 MEs and spatial facts (9 hours). As expected, the number of recognized CEs does not change with respect to the experiments including spatial reasoning.

Figure 11(b) shows that even though the stream used as input for CE recognition increases significantly when RTEC does not per-

form spatial reasoning, the average CE recognition times decrease substantially. Moreover, this figure shows that RTEC scales to large data streams—e.g. CE recognition for all 6,425 vessels and 35 areas using a window of 125,000 input facts takes on average 1.5 sec when two processors are used in parallel.

5.3 Discussion

Empirical results confirm that the proposed system is capable of recognizing, in real-time, suspicious and potentially dangerous situations that require immediate attention from marine authorities. Even when it must cope with an increased amount of incoming positions (expressed with larger window ranges sliding less often), it can recognize complex events in a few seconds at most. Such performance is noteworthy, given the large number of monitored vessels in the simulated scenario. This comes thanks to the scalability and robustness of the trajectory event detection mechanism, which offers reliable incremental response (often in less than a second) and can filter out noisy or intermittent input signals. In addition, the complex event recognition module can take advantage of the evolving trajectory features in order to recognize complex spatiotemporal relationships between vessels and areas of interest.

6. RELATED WORK

Detecting events from massive, streaming data has attracted a lot of research interest, but also opens up great perspectives for building powerful monitoring applications in several domains. Event detection from both live and archived streams has been proposed in [10], introducing optimizations specifically for recency-probing pattern queries. The goal of UpStream platform [23] is to answer continuous queries with the lowest staleness possible, when each data item represents an update to a previous one; this certainly applies to GPS positions from moving vessels, but UpStream lacks support for the specific demands of trajectory monitoring. Automating ingestion of streaming data feeds from various sources into data warehouses is also a challenging issue, as outlined in a recent tutorial [15]. To the best of our knowledge, no streaming framework is specifically tailored for maritime surveillance over fluctuating, noisy, intermittent AIS messages from large fleets.

Detecting trajectory events from positional streams essentially performs path simplification, a topic explored in several previous works. Some strategies opt for acceptable approximations in terms of a given error margin, such as those in [8, 19, 22]. Besides, the memory space available for retaining a compressed sequence may also be crucial in a single-pass evaluation [31]. Dead-reckoning policies like [33] are employed in moving sources to relay positional updates upon significant deviation from the course already known to a centralized server, so they aim to reduce communication cost. This is not the case with AIS data, as maritime control centers wish to locate ships as frequently as possible. The advantage of our proposed scheme is that it 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. Most importantly, we annotate reduced representations according to particular movement events along each vessel trace.

For archiving trajectories, we make use of Hermes [30], a prototype Moving Object Database (MOD) equipped with a powerful query language. Hermes MOD supports modeling and querying of moving objects, and enables support of aggregative Location-Based Services. It defines a trajectory data type as well as a collection of spatiotemporal operations (range, nearest neighbor, similarity, etc.), which take advantage of a robust indexing mechanism. Semantic-aware trajectory construction [34] applies cleaning, compression and segmentation over positional data, in order to define "stop" and "move" episodes along each trace in online fashion. Besides, a formal model for OLAP operations at different granularities was recently proposed in [20], but it is mostly geared towards visual analytics over offline trajectory data.

In terms of Complex Event Recognition, RTEC has a formal, declarative semantics in contrast to other related systems that usually rely on an informal and/or procedural semantics. Cugola and Margara [9] point out that almost all "complex event processing languages", including [4], and several "data stream processing languages", such as ESL [7], lack a rigorous, formal semantics. Eckert and Bry [13] note that the semantics of "event query languages" often are somewhat ad hoc, unintuitive and generally have an algebraic and less declarative flavor. Paschke and Kozlenkov [27] state that commercial "production rule languages" lack a declarative semantics. Unlike [12, 18, 9], RTEC supports complex atemporal reasoning and reasoning over background knowledge (e.g., identifying the type of a vessel given its characteristics), which are quintessential for maritime surveillance [14]. This way, it is possible to express the complex phenomena required by the maritime experts [32]. Furthermore, RTEC explicitly represents complex event intervals and thus avoids the related logical problems (see [25] for a discussion of these problems), and supports out-of-order event streams (in contrast to e.g. [11, 9, 10, 21]). Concerning the Event Calculus literature, a key feature of RTEC is that it includes a windowing semantics. In contrast, no Event Calculus system "forgets" or represents concisely the event history.

7. SUMMARY & FUTURE WORK

In this paper, we introduced a system that monitors the activity of thousands of vessels and can instantly detect and recognize events with a potentially serious impact on the environment and on safe navigation at sea. The system can sustain large amounts of 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 characteristics. Furthermore, this reduced information may be readily analyzed online for complex event recognition. Equipped with efficient pattern matching algorithms, this module correlates critical trajectory positions with static geographical and vessel data, and detects suspicious or dangerous situations, such as loitering, vessels passing through protected areas, and unsafe shipping. Our platform has been empirically validated against a large real dataset and met expectations for timeliness, scalability, and robustness.

We plan further extensions and improvements in the existing implementation. First, we soon expect to be given access to *live* AIS feeds from all vessels across the Aegean Sea. This will integrate our system 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.

Existing definitions of complex events were manually developed in collaboration with vessel traffic service staff. However, creating CE definitions manually is painstaking and error-prone. We have begun exploring techniques based on abductive-inductive logic programming, for automated generation and refinement of definitions from very large datasets.

Besides, maritime surveillance exhibits various types of uncertainty, exactly like most event processing applications. So, we are porting RTEC into probabilistic logic programming frameworks, in order to deal with imperfect complex event definitions, incomplete and erroneous data streams. A probabilistic treatment would be also challenging for addressing the gradual ageing and transmission delays in AIS data. Traffic forecasts at short-term horizons

(e.g., 5, 15, or 30 minutes ahead) could also be issued, gracefully weighing online events with offline trajectory analytics.

Last, but not least, maritime surveillance may benefit from combining multiple data sources. As RTEC readily supports heterogeneous stream processing [6], we aim to experiment with additional data, such as weather forecasts, for improved monitoring.

Acknowledgements

This work was carried out in the framework of the project *"AMINESS: Analysis of Marine INformation for Environmentally Safe Shipping"* co-financed by the European Fund for Regional Development and from Greek National funds through the operational programs "Competitiveness and Enterpreneurship" and "Regions in Transition" of the National Strategic Reference Framework - Action: "COOPERATION 2011 – Partnerships of Production and Research Institutions in Focused Research and Technology Sectors". An initial version of the online tracking module was partially developed during a Short Term Scientific Mission (STSM) of the first author at Naval Academy Research Institute (Brest, France) with the support of EU COST Action IC0903 on Knowledge Discovery from Moving Objects (MOVE). Helpful discussions with Cyril Ray and Christophe Claramunt in this STSM are kindly acknowledged.

8. REFERENCES

[1] Aminess project. `http://www.aminess.eu/`.

[2] Marine traffic. `http://www.marinetraffic.com/`.

[3] Seabilla project. `http://www.seabilla.eu/`.

[4] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Efficient pattern matching over event streams. In *SIGMOD*, pages 147–160, 2008.

[5] A. Artikis, M. Sergot, and G. Paliouras. An event calculus for event recognition. *IEEE Transactions on Knowledge and Data Engineering*, 2014.

[6] A. Artikis, M. Weidlich, F. Schnitzler, I. Boutsis, T. Liebig, N. Piatkowski, C. Bockermann, K. Morik, V. Kalogeraki, J. Marecek, A. Gal, S. Mannor, D. Gunopulos, and D. Kinane. Heterogeneous stream processing and crowdsourcing for urban traffic management. In *EDBT*, pages 712–723, 2014.

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

[8] H. Cao, O. Wolfson, and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *VLDB Journal*, 15(3):211–228, 2006.

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

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

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

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

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

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

[15] L. Golab and T. Johnson. Data stream warehousing *(tutorial)*. In *ACM SIGMOD*, pages 949–952, 2013.

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

[17] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–96, 1986.

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

[19] R. Lange, F. Dürr, and K. Rothermel. Efficient real-time trajectory tracking. *VLDB Journal*, 20(5):671–694, 2011.

[20] L. Leonardi, S. Orlando, A. Raffaetà, A. Roncato, C. Silvestri, G.L. Andrienko, and N.V. Andrienko. A general framework for trajectory data warehousing and visual OLAP. *GeoInformatica*, 18(2):273–312, 2014.

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

[22] N. Meratnia and R.A. de By. Spatiotemporal compression techniques for moving point objects. In *EDBT*, pages 765–782, 2004.

[23] A. Moga and N. Tatbul. UpStream: A storage-centric load management system for real-time update streams. *PVLDB*, 4(12):1442–1445, 2011.

[24] International Maritime Organization. Automatic identification systems. `http://www.imo.org/OurWork/Safety/Navigation/Pages/AIS.aspx`.

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

[26] A. Paschke and M. Bichler. Knowledge representation concepts for automated SLA management. *Decision Support Systems*, 46(1):187–205, 2008.

[27] A. Paschke and A. Kozlenkov. Rule-based event processing and reaction rules. In *RuleML*, pages 53–66, 2009.

[28] K. Patroumpas. Online tracking and summarization over streaming maritime trajectories. In *MOVE Workshop on Moving Objects at Sea*, 2013.

[29] K. Patroumpas and T. Sellis. Maintaining consistent results of continuous queries under diverse window specifications. *Information Systems*, 36(1):42–61, 2011.

[30] N. Pelekis, E. Frentzos, N. Giatrakos, and Y. Theodoridis. Hermes: Aggregative LBS via a trajectory DB engine. In *ACM SIGMOD*, pages 1255–1258, 2008.

[31] M. Potamias, K. Patroumpas, and T. Sellis. Online amnesic summarization of streaming locations. In *SSTD*, pages 148–165, 2007.

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

[33] O. Wolfson, A.P. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distributed & Parallel Databases*, 7(3):257–287, 1999.

[34] Z. Yan, N. Giatrakos, V. Katsikaros, N. Pelekis, and Y. Theodoridis. SeTraStream: Semantic-aware trajectory construction over streaming movement data. In *SSTD*, pages 367–385, 2011.