A Geofencing Algorithm Fit for Supply Chain Management

Paolo Walter Modica, Mark Phillip Loria and Marco Toja
See Your Box
2 Cormont Road
London, England
SE5 9RA (UK)
Email: {pmodica, mloria, mtoja}@seeyourbox.com

Vincenza Carchiolo and Michele Malgeri
Dip. Ingegneria Elettrica Elettronica e Informatica,
Universitá di Catania,
Viale Andrea Doria 6,
95125 Catania, Italy
Email: {vincenza.carchiolo, michele.malgeri}@dieei.unict.it

***Index Terms*—IIoT, Geofencing, Supply chain, algorithm**

***Abstract*— Location Based Services play an important role in decision-making processes, company activities or in any control and policy system in modern computer organizations. Usually LBS applications provide location-specific information only when user requests them. However, Supply Chain Management applications require to push geolocalized information directly to the user. The most discussed and requested application is *Geofencing*, which allows to determine the topological relation between a moving object and a set of delimited geographical areas. This paper describes the design of an innovative solution for implementing proactive location-based services suitable for application scenarios with strong time constraints, such as real-time systems, called Proactive Fast and Low Resource Geofencing Algorithm.**

I. Introduction

Location-based services (LBS), software-level functionalities that use location data to control features, have recently become a hot topic for both consumer and industrial applications, evolving from simple synchronization-based service models to authenticated and complex tools thanks to advancements in telecommunication technologies and localization services.

Nowadays, Location Based Services are *crucial for many businesses*, as well as for government organizations, as they could play an important role in decision-making processes, company activities or in any control and policy system in modern computer organizations.

The majority of applications exploiting LBS are based on the idea to present location-specific information in case the user asks for it. A relatively small amount of new applications act *proactively*, delivering *enter*, *exit* and *cross* geo-notifications directly to the end user. The most discussed and requested proactive LBS nowadays is *Geofencing*, which allows to determine the topological relation between a moving object and a set of delimited geographical areas.

This paper describes the design of a new *geofencing algorithm*, Proactive Fast and Low Resource Geofencing Algorithm (PFLGA) proposed as an innovative solution for implementing proactive location-based services suitable for application scenarios with strong time constraints, such as real-time systems. PFLGA is then exploited in a real application scenario to implement a geofencing service within the See Your Box system, an IT company offering Business-to-Business services which allow early detection of logistic issues happening in Supply Chain Management across several industry verticals.

In particular, PFLGA aims to resolve trajectory-based topological join queries to catch the occurrence of topological events, related to the movement of mobile IoT/M2M tracking devices, within strict time costraints, in the context of real-time supporting services for Supply Chain Management.

PFLGA proposes a centralized, thin-client solution to the geofencing problem, exploiting the GeoJSON format and tree-based index structures for the representation, collection and indexing of geospatial geometric shapes. These features allow the proposed solution to face two classical challenges of Geofencing: reducing the energy consumption at the mobile device, and allowing the matching process within the centralized solution to scale [1].

Classic geofencing solutions require the use of GPS locators continuously connected and therefore in need of continuous power supply.

Several different algorithms have been proposed in the technical literature for the implementation of geofencing functionalities and this section analyzes some of the latest and most interesting designs of the recent years. One of the main problem is the *point-in-polygon problem (PIP)*, which, in computational geometry, solves the question about the position of a point with respect to the boundary of a polygon in the same plane. The paper compares PFLGA with *Parallel In-memory Spatio-temporal TOpological joiN* (PISTON) [2], *Scan-Line Algorithm and Grid Compression (SLGC-1)* [3] and *Geofencing via Hybrid Hashing* [4]. The features and performance of PFLGA are far better than that of the algorithms mentioned above, making it a good solution for the Supply Chain Management context.

Section II examines location-based services, addressing their components, their technical characteristics, focusing on geofencing and route-matching services. Section III describes features and requirements of the application scenario considered in this work, which relates to *Machine-to-Machine (M2M)* data processing systems based on infrastructure-assisted mobile devices, potentially exploitable in various areas such

as Supply Chain Management, then considers many existing geofencing solutions, taken from the technical literature. Section IV deeply discuss PFLGA and one of its possible implementation and provides a comparison between the algorithm mentioned above and those existing geofencing designs which better fit with the service requirements of the considered application scenario. Section IV-B provides comparison both in terms of features and performance for achieving the geofencing result. Finally, conclusions and ideas for future developments are discussed in Section V.

II. Related Works

A. Location Based Service and Geofencing

A location-based service is a software-level service that uses location data to control features [5]. LBS is a part of virtually all control and policy systems which work in computers today [6] and can be used in a variety of contexts, such as health-care [7], entertainment, indoor object location and work. LBS has rapidly evolved from simple synchronization-based service models to authenticated and complex tools for implementing virtually any location based service model or facility, becoming crucial for many businesses, as well as for government organizations, as they could play an important role in any control and policy system in modern computer organizations.

Together with the Internet of Things (IoT) paradigm, it constitutes an enabling technology for advanced Machine-to-Machine services, useful for companies across diverse industries where often the efficiency plays an important role [8].

According to the definition given by the international Open Geospatial Consortium [5] LBS is an intersection of three technologies:

1) wireless and mobile telecommunication technologies, which play an important role for the development of many new location-based services for both business and consumer applications, providing applications with an infrastructure able to manage the communications between mobile terminals and Providers of the service [5];
2) Geographic Information Systems (GIS) [9] that provides a strong framework to build database management systems and object extensions, to store and maintain geographical records to monitor the status and changing of the worlds geography, and application software such as interactive maps and analysis instruments;
3) New Information and Communication Technologies (NICT), which encompass all those technologies and smart assets, with embedded processing and communication capabilities, which enable human actors to access, store, manipulate, transmit and share information wherever and whenever they want.

LBSs differ from the common Internet services because they must be aware of the context in which they are being used and they must adapt their contents actions accordingly [5]. LBSs must be aware of any information that could be used to describe the context, such as a place, its features, the objects and people standing in it and anything that is relevant to the interaction between an user and the LBS application. Systems that can dynamically change their behavior according to the context are defined *context-sensitive or responsive*. Service adaption can take place at different levels, from the information level, where the information provided by the service is adapted according to the context, to the user interface and presentation level.

According to literature [10] LBS can be classified in the following two categories:

- *Pull LBSs*, which deliver information directly requested from the user/customer. Pull LBSs can be further classified [5] in functional services, which facilitate the user in the acquisition of goods and services related with his/her position, and information services, which retrieve information about a specific subject depending on the user's position;
- *Push LBSs*, which deliver information not requested or indirectly requested from the user, although the user may have originally subscribed to the service at an earlier time. Push services are activated by events and are usually more complex to establish

In the last years the attention moved from Pull LBSs to a more advanced, proactive type of location-based services, where environmental information is pushed to the user depending on the geographical position of a mobile device [11]. The most discussed and requested Push LBS in recent time is geofencing.

In this paper we focus on *Geofencing* that is is a location-based service which enables to detect and monitor when a mobile IoT/M2M device enters, leaves, crosses or bypasses a precise geographical area delimited by a virtual perimeter, called geofence [10], [12] providing alerts or notifications, usually referred to as *geo-notifications*. A geofence can be dynamically generated, like a circular area surrounding the current position of a mobile device, or can be made of a predefined set of boundaries, which may be arbitrarily drawn by the user or specific for a place or a building. Geofencing services can be classified, depending on the geographical references used to check device's position, in [13] *static* that checks the geographical position of a mobile device with respect to a fixed area, *dynamic* that operates according to the position of a mobile device with respect to a changing area and *peer-to-peer* that uses the geographical position of a mobile device with respect to other mobile devices.

A geofencing can be characterized according to the following features [13]:

- *location accuracy*: geofencing accuracy is strictly related to the accuracy of the geographical position provided by the service used to track the location of the mobile device, either satellite/GPS or GSM-based.
- *Tracking Rate*: expresses the frequency by which the device provides a location update to the server of the proactive LBS.
- *Device Speed*: the speed of a device determines the time

period within which the device must provide a location update to be evaluated against eligible events.
- *Device Route*: the path a device takes across a geofenced area which affects the time period within which location update must occur.
- *Geo-notification delivery*: geo-notifications can be delivered to the user only once or every time the mobile device successfully enters, leaves, crosses or bypasses a geofenced area.

The spread of location-based services applied to IoT technologies, especially for mobile-based solutions, makes it necessary to add to the features listed above the *power efficiency*, since the signal is triggered by small, battery powered mobile devices.

The behavior of a moving mobile IoT/M2M device relative to a set of one or more geofences can be easily defined using the spatial predicates enter, leave, bypass and cross, proposed for the first time by Erwig, Schneider et al. in [14]. Whether a moving object enters, leaves, crosses or bypasses a given geofenced area can be determined by examining one or more segments of its trajectory, checking if they intersect with the before mentioned geofence and evaluating the intersections found. The whole process is often referred to as trajectory-based topological join query.

Trajectory-based topological join queries are really powerful instruments for spatiotemporal analysis, but they are also rather compute-intensive. For their resolution they require a description of the route traveled by the moving object, which could be expressed as a set of segments or as a polyline. This requirement raise the issue of how the route should be determined, which will be discussed later on when it comes to route-matching service.

B. Geofencing implementation

One of the most important component in a geofencing system is the Location Monitoring Unit (LMU), which is the component inside the geofencing system infrastructure which is responsible for location processing of the positions of a mobile device and for keeping the geofence scenarios secret. Technically, a geofencing system can be implemented in two different solutions [1]: mobile-based and centralized system.

In a mobile-based geofencing system, the device positioning, determined with satellite/GPS technology, together with the matching of the position with a set of geofences is executed at the mobile device. This type of geofencing systems represents a thick client solution which is mainly used in case a trustful position of nodes is needed, although it requires high battery consumption due to the geospatial processing executed at the mobile node.

In a centralized geofencing system, a mobile device is being tracked by the surrounding network infrastructure, while the matching of the retrieved position with a set of geofences is executed by the servers which make up the geofencing system infrastructure. Centralized geofencing systems represent thin client solutions [15] and have several advantages over mobile-based counterparts, such as:

- mobile devices get rid of the CPU-intensive geospatial processing necessary to determine the current state of a mobile client regarding the geofence scenarios.
- Centralized geofencing systems use network-based positioning methods within the infrastructure, such as satellite-GPS positioning or GSM Cell ID positioning, which relieve the mobile clients from the energy-draining positioning process.
- Since the LMU monitors all the mobile clients of the system, centralized systems allow for collaborative geofencing and monitoring of the current number of clients within a particular geofence.

On the other side, in centralized geofencing systems the communication between the LMU and mobile clients increases, accompanied by all the weaknesses of mobile communications like loss of connection, unpredictable latency and an energy consumption tightly depending on the location update frequency.

C. Geofencing challenges

Geofencing is associated with two main technical challenges: reducing the energy consumption at the mobile device, in particular within the mobile-based solution, and allowing the matching process within the centralized solution to scale [1]. The high energy consumption of the mobile device is mainly caused by the positioning modules (satellite-GPS or GSM-based) nodes are equipped with and, in case of mobile-based geofencing systems, by the geospatial processing necessary to determine node status.

As for the high energy consumption caused by the positioning modules inside the device, this is tackled by selecting the positioning method based on:

- accuracy need: satellite-GPS technologies allow for a more precise localization than GSM Cell ID technology;
- current environment: in case GSM signal doesn't reach the area the mobile device is currently in, positioning technique is switched to satelliteGPS;
- current position/speed towards a geofence: in case the mobile device is far from a geofence boundary, at a distance which is greater than a specific safety radius, then no location update is needed. Otherwise, if the mobile device is close to a geofence boundary, it will transmit periodic location updates to make the system check its position relative to the geofence.

In the context of geofencing systems, scalability is required in two dimensions [16]:

1) amount of geofences set by the user;
2) number of location updates processed per time unit (throughput).

To allow the customer using the geofencing system to store as many geofences as he wants, without compromising the efficiency of the service and aiming at the maximum scalability, geospatial objects need to be indexed using spatial indexes.

The use of spatial indexes allows to reduce greatly the time needed to resolve geospatial queries, which results in

more location updates processed per time unit, which increases system scalability.

D. Route-matching

Intelligent Transport Systems (ITS) and Location-Based Services (LBS) require location information about mobile IoT/M2M telemetry devices. In the last few years Global Positioning System (GPS) has established itself as the major positioning technology for providing location data. This information can be used with spatial road network data to determine the spatial reference of device location via a process known as map-matching or route-matching [17].

Route-matching techniques integrate positioning data, coming from satellite/GPS or GSM positioning technologies, with spatial road network data to provide enhanced positioning capabilities, with the aim of identifying the most plausible route segment traveled by a mobile object between two or more location points [18] and determining the device location inside the calculated road segment [17].

The quality of the result returned by a map-matching service depends on:

- the quality of the spatial road map used by the algorithm, which must always been up to date and checked in-depth in order to identify and correct flaws in the available road network data;
- devices sampling frequency, which in turn depends on the precision requirements of the localization service and on the performance of the transceiver the mobile device is equipped with. Since the positioning technology chosen for tracking the mobile device is characterized by a known measurement error the desired accuracy for the route-matching algorithm can be achieved by adjusting the sampling rate of the tracking service, according to the lens-shaped probability distribution function describing the sampling error, depicted in [19];
- the result of the initial map matching process, which selects a set of road segments falling within an error ellipse, representing the area in which the current position of the mobile device may be, according to the error of the localization service. In case the vehicle initial position is further from roads junctions, the ellipse produced by the initial matching process won't contain any junction point nor shape point assuming the vehicle is outside of the known road network;
- the implementation of the route-matching service. The mobile-based approach requires high battery consumption, due to the route-matching algorithm executed at the mobile node whilst it reduces communications between the device and the central server. The centralized implementation relieves mobile devices of the CPU-intensive map-matching processing, although communications between devices and the server increase, accompanied by all the weaknesses of mobile communications, such as loss of connection, unpredictable latency and energy consumption.
- The route detail level required by the route-matching service application. Simplified routes are less accurate than fully detailed routes, but their computation time is lower .

E. Geofencing algorithms in literature

Custom virtual fences surrounding specific areas of interest have been used for more than a decade for on-line mapping applications, proximity-based digital coupon distribution and many other application software. Since its first appearance in research and technical literature, geofencing has evolved into a powerful geospatial analysis tool, becoming one of the most cutting-edge feature in application software and systems used in different fields.

Several different algorithms have been proposed in the technical literature for the implementation of geofencing functionalities and this section analyzes some of the latest and most interesting designs of the recent years.

One of the main problem is the *point-in-polygon problem (PIP)*, which, in computational geometry, solves the question about the position of a point with respect to the boundary of a polygon in the same plane. The PIP test finds application in areas dealing with geometrical data processing, such as computer graphics, computer vision, geographical information systems and many more. One of the first approach is the ray-casting algorithm, proposed in the early description of the point-in-polygon problem [20], but this method doesn't work in case the point is on the edge of the polygon.

Many of the discussed examples define simplified geofencing features by solving, with different approaches, the point-in-polygon test.

PISTON: Parallel In-memory Spatio-temporal TOpological joiN (PISTON) is a geofencing algorithm, designed by the research team of the Department of Computer Science of the University of Toronto, which implements a parallel, main memory, query execution infrastructure designed specifically to address spatiotemporal join [2]. PISTON, which was initially designed as an optimization of the INLJ2I geofencing algorithm, introduces a novel parallel, in-memory trajectory index I_R, designed to handle a high rate of location data updates, and a novel in-memory spatial index I_S, organized with a two level grid approach and specifically optimized for point-in-polygon test. PISTON delivers low query response times acceptable for real-time use-cases, even with large geofence datasets.

SLGC-1: Scan-Line Algorithm and Grid Compression (SLGC-1) is a geofencing algorithm, designed by the development team of the Software School of the Xiamen University of China to solve regional limited problems in Internet of Vehicles (IoV) systems with restricted time and storage requirements [3]. It works in two separate steps. In the preprocessing step the algorithm imposes a spherical grid on the geofence area in input, matching the shape of the real region, then a scan conversion algorithm is used to determine the location attribute of each cell of the grid and. Finally the grid is compressed using a QuadTree compression algorithm,

which provides a memory-efficient index structure (storage requirements is less than O(n)) for the geofence area to analyze and calculates the Morton Code (MD code) to identify each node inside the QuadTree structure.

Geofencing via Hybrid Hashing: Geofencing via Hybrid Hashing was selected as one of the three best geofencing algorithms, out of the 29 submitted ones, proposed as a solution for the task posed by the ACM SIGSPATIAL GIS CUP 2013 contest [4]. It builds and updates the in-memory hash tables used to index polygons during system spare time, shifting some computation cost from the point-in-polygon test stage to non-time-critical processing stage [4]. taking advantage that, in typical geofencing applications, points position are changed much more frequently than those of polygons.

On the basis of the results obtained by testing the algorithm on the dataset provided for the ACM GIS CUP 2013 contest, the algorithm provides low response times with respect to many other algorithms.

III. Application Scenario

This section focuses on the application of geofencing algorithms in services related to Supply Chain Management (SCM), which involve the movement and storage of raw materials and unfinished products from the point of origin to the point of destination and/or consumption. SCM, was traditionally driven by Enterprise Resource Planning systems, which provided plans and estimation regarding the different aspects of the business activity. In recent times, a quiet revolution has been taking place thanks to the use of Location-based technologies and innovative solutions to track and trace transportation equipment, materials and drivers across all the step of the supply chain [21]. LBS allow the enterprise to dynamically tender and dispatch shipments in real time, divert a route because of weather conditions or a severe accident that is causing major delays in the transportation route, or transmit notification messages to the stakeholders whenever a shipment arrives at a warehouse [21] making the whole process more efficient and less expensive.

For all these reasons, location-based services can be considered a disruptive technology for the supply chain that will bring great opportunity for logistics innovation. The evolution in mobile telecommunication technology, together with the advances in electronics and the introduction of the IoT paradigm, has enabled the networking of portable wireless devices and wearable computers that can provide new types of usable knowledge to all the members and stakeholders of a globally dispersed supply chain [22].

These devices, equipped with sensors and actuators, exploit their connection capability to transmit to the Service Provider's servers, in a Machine-to-Machine communication, all the data regarding the status of the shipment which are important for the service's functioning.

The installation of M2M nodes in pallets, containers, vehicles and warehouses, along with new types of inference algorithms and techniques, will enable seamless, efficient, and transparent movement of raw materials and products through the global supply chain [22], allowing the business customers to look at all the critical points of the chain.

M2M communication, together with geofencing, map-matching and localization services, represent the enabling technologies for developing and deploying a location-based service.

Developing a Location-Based Service, founded on M2M Communication, to support the Supply Chain Management raises several technical challenges, the most important are: geographical diversity and telecommunication coverage, location awareness, response time, accuracy of the result, power conservation, security and privacy, meet customer expectations.

Geographical diversity and telecommunication coverage: along their trip, from the moment goods are packaged for shipping to the moment they arrive at destination, containers and cargoes go to many places where GSM coverage is poor to non-existent. Thus, the use of dual mode GSM-satellite M2M devices is crucial to provide uninterrupted service to customers thus satellite communication is always available as a back-up technology to transmit the device's position. Furthermore, mathematical statistical interpolation may be used to fill the missing data.

Location awareness: in some application scenarios, such as air transport, the ability of the LBS to switch operative mode depending on devices' position could be an important feature both to meet customer needs and legislative restrictions (e.g. IATA restrictions on network-enabled electronic devices [23]).

Response time: the algorithm behind the LBS services should return the result of the computation within a specific deadline from the moment the packet, transmitted by the device, is received. This is important in order to guarantee the responsiveness of the application which uses the service, and is crucial for time-critical applications.

Accuracy of the result: the precision of devices location depends on the hardware and software used in the mobile communication system, as well as on the positioning service [24]. The accuracy level requested to the Location-Based Service, both for position tracking or route-matching, influence the service's response time and varies depending on the application scenario in which the service will be used.

Power management: energy efficiency and power consumption are critical aspect when developing a LBS using battery-powered M2M devices. Containers and cargoes trips from source to destination may last 75 days in average, so the device attached to them should work properly for long period of time, often without the possibility to recharge the battery.

Security and privacy: customer concerns about security and privacy are another challenge for location-based technologies applied to Supply Chain Management. With regards to shipment security, it is desirable that the LBS integrates a priority function which immediately alerts the customer in case of illicit manipulation of the container holding the goods. On the other hand, the M2M devices should transmit the shipment status and location data using data security instruments, such as cryptography, to keep them confidential and avoid interception of sensible information.

Finally, the developed services have to satisfy customers expectations, in terms of expected results, perceived Quality of Service, reliability, availability and more importantly cost.

Among all available tools which contribute to the establishment of applications and systems for Supply Chain Management, geofencing plays an important role in the context detection of proactive applications, which can automatically adapt business and industrial operations to the geospatial context a user, or a mobile device, is currently in.

A. Geofencing service for SCM systems

Geofencing allows to detect and monitor the changing in the topological relation between a mobile device and a bounded geographical area (the geofence). The aforementioned topological relation can be expressed in terms of the spatio-temporal predicates enter, leave, cross and bypass, which are obtained as the result of trajectory-based topological join queries. These queries test the intersection between the whole or part of the trajectory of a moving object and a geofence, returning the spatio-temporal predicate describing their relationship based on the intersections found. These queries are really powerful instruments for geospatial analysis, but they are also computationally intensive.

This represents a challenge, as it requires to identify or construct the most efficient algorithm or method which, under the operating conditions of the specific system and application scenario, resolves geofencing problems, returning the result of trajectory-based topological join queries in a period of time that is acceptable for interactive, ad hoc geospatial analysis services.

The application scenario of the geofencing service proposed in this paper is that of IoT-based industrial services supporting Supply Chain Management and logistics for remote monitoring of goods and assets using mobile devices, smart cards, tags or similar technologies. This kind of services are placed in the context of Industry 4.0. In particular, the application scenario presented in this paper provides for a system using uniquely identifiable mobile objects, from here onwards called trackers, which transmit real-time location data with a precise, configurable frequency, which may change over time. The aforementioned system is centralized and thin-client, meaning that the trackers have limited resources and processing capabilities, in order to save battery power to provide a long-lasting monitoring service.

The geofencing service to be implemented should make it possible to detect whether one of the aforementioned mobile devices, capable of transmitting real-time location data to the system, enters, leaves or crosses one or more specific areas of interest, the geofences, and, whenever this occurs, it should notify the system of the event, depending on the specific service configuration assigned to the specific device. Each geofence should be statically defined by a geometric shape or by indicating a location identifier, such as an address. In order to reduce the processing within the tracker, all computing related to geofencing should be performed within server computers. Furthermore, to meet the real-time requirements, the geofencing routine must be non-blocking and the service must return a correct result within specific time constraints, often referred to as deadlines, failing which the result should be invalid.

In order to determine the topological relation between the moving object and the set of assigned geofences, the service should be coupled with a utility capable of reconstructing the path traveled by the device. In addition, if it is not possible to estimate all or part of the route traveled by the tracker, or in case a low level geospatial analysis is requested, the service should be able to work with the simpler geospatial data available at the time of the request. Following the reception of a notification from the geofencing service, the system should notify the user about the event and/or switch the device configuration depending on the event occurred.

According to the above features, the requirements of a good geofencing for SCM are:

- static, meaning that the spatio-temporal predicates are verified by checking the trajectory of the moving object with respect to fixed, bounded areas;
- geometric and symbolic addressed, so that the geofences could be defined with both geometric shapes or symbols, such as words and alphanumeric codes, which identifies precise locations;
- centralized, so that the matching between the trajectory of the moving object with the set of associated geofences is executed by the servers, which are the main part of the system;
- capable of operating effectively with different and variable location accuracy, tracking rate and device speed.

IV. Proactive Fast and Low Resource Geofencing Algorithm

In order to satisfy all the requirements discussed in the previous section, we propose a new geofencing design inspired by the ray-casting algorithm called Proactive Fast and Low Resource Geofencing Algorithm (PFLGA). The developed solution exploits geofences drawn over the WGS84 (or EPSG:4326) world geodetic coordinate system [25], which is a mathematical model of the Earth from a geometric, geodetic and gravitational point of view and is used by GPS navigation system and for aviation as a mandatory standard.

The proposed design provides every tracker for which geofencing service is enabled with a set of one or more geofences, i.e. geospatial objects, such as polygons and circles, whose boundaries are drawn over a specific geodetic coordinate system.

The data periodically transmitted by the tracker carries its geographical position, expressed in terms of latitude and longitude coordinates, enabling the location update for each tracker. These geographical points are used to determine the most plausible *path traveled* by the device between location updates, exploiting route-matching services with different levels of detail. The *traveled path* is then used to determine the topological relation between the moving object and the set of geofences assigned to the shipment the device is attached to.

Although the algorithm is designed to determine the topological relation between a moving object and a set of geofences using its trajectory, it can also perform the geofence inclusion test using other geospatial objects, from a geographical location point to a single segment of the whole complex trajectory traveled by the tracker, depending on the available geospatial information regarding the moving object and the complexity required for the geofencing analysis. This allows the algorithm to be potentially applied to different use cases, from those which require an examination with low level of detail, in favor of a low query response time, to those that require a detailed geospatial analysis regardless of the query response time.

Since the application scenario in which the geofencing service will be used provides that each tracker can be assigned a set of one or more geofences, and the trajectory-based topological join queries are rather compute-intensive, it is important that data structures containing geospatial data support the retrieval of elements of an arbitrarily large size in an efficient way, therefore the proposed geofencing algorithm uses an in-memory, tree-based index structure for indexing the set of geofences assigned to each tracker for which the geofencing service is required. The insertion strategy for these structures has a computational complexity of $O(n)$, while the search operation has a computational complexity of $O(\log n$, which permits a fast object retrieving in time critical applications.

The use of this kind of spatial index enables the application of an efficient filtering strategy on the set of geofences on which the intersection test with the trajectory will be performed.

PFLGA searches for any intersection between the route traveled by the tracker and progressively smaller bounding areas, called *Minimum Bounding Rectangles* (MBRs), which contain one or more geofences within them. In case the trajectory doesn't have any intersection with those bounding areas, the test ends without checking the set of geofences, otherwise the test continues with smaller bounding areas, until a precise geofence is found and tested.

PFLGA is based on the theory behind trajectory-based topological join queries. Given a set of geofences, bounded geographical areas represented as polygons or circular shapes, and the whole or a part of a trajectory defining the path traveled by the moving object, represented as a polyline geometric object, the algorithm verifies the existence of intersections between this polyline and the set of MBRs containing the geofences to be analyzed. If the polyline defining the trajectory intersects one or more MBRs, the algorithm performs the following steps on each geofence contained within the MBRs of interest:

- gets the previous position of the device and checks whether it was inside or outside the current geofence;
- calculates the intersections between the geofence and the trajectory traveled by the device, if there is any;
- analyzes the result obtained above and returns a a composite topological join predicate [19], which tells whether the object entered, crossed or left that precise delimited area.

This algorithm implies the a-priori construction of the in-memory, tree-based index, which will use a time interval proportional to the dimension of the set of geofences in exam (since the insertion algorithm for this tree-based index structure has a computational complexity of $O(n)$). Since the set of geofences is quite static and it is updated rarely, compared to the location of the moving object and its trajectory, the additional processing required for the index is bearable, especially if it is compared to the query processing speed up offered by the use of this index structure. The algorithm, whose possible implementation in pseudo-code is shown below, has a computational complexity of $O(\log n)$.

input : $\mathcal{F}$: set of geospatial objs making geo-fences
idx : index of the set of geofences
$route$: trajectory traveled by the object
output: $predicate$, position with respect to the fence

```
// list of FeatureIDs of the MBR in
// the index intersecting the route
I = getIntersection(idx, route)
foreach pos ∈ I do
    prev_pos = getPrevPosition(route);
    // checks if the prev_pos was
    // inside the current MBR
    wasInside =
     getIntersection(position, prev_pos)! = null);
    Intersections = getIntersections(pos, route)
    if (wasInside) then
        if Intersections number is odd then
            the object leaved the fence
        else
            the object is still inside
        end
    else
        if Intersections is empty then
            the object is still outside
        else
            if Intersections number is odd then
                the object entered the fence
            else
                the object crossed the fence and it is
                 outside
            end
        end
    end
end
```

Algorithm 1: PFLGA description

A. Comparison between geofencing algorithms

In order to verify the performance of the proposed geofencing algorithm, the developed solution is compared with other geospatial analysis algorithms which resolve the same geofencing problem.

First, the algorithms are compared on the basis of the set of features requested by the application scenario the proposed solution has been developed for. Subsequently, the developed design is tested using the same dataset and setup utilized for testing the performance of the other solutions considered in this comparison, which vary depending on the algorithm in exam.

All the features of PFLGA are used to study the similarities between the proposed design and the other geofencing solutions considered in the previous paragraph. Table I shows the comparison among the following features:

1) In-Memory Spatial Index (IMSP);
2) Spatio-Temporal Topological Join Predicates (STTJP);
3) Use of Trajectory-Filtering / Filtering Strategy (TFFS);
4) Geofencing via evaluation of point-in-polygon (PIP);
5) Geofencing via evaluation of spatial intersection with trajectory segments (SEGS);
6) Geofencing via evaluation of spatial intersection with complex trajectories (TRAJ).

	IMSP	STTJP	TFFS	PIP	SEGS	TRAJ
Hybrid Hashing	Yes	Yes	Yes	Yes		
SLGC-1	Yes		Yes	Yes		
PISTON	Yes	Yes	Yes		Yes	Yes
PFLGA	Yes	Yes	Yes	Yes	Yes	Yes

TABLE I
COMPARISON OF FEATURES

Next we compare the algorithms with respect to the benefits and/or their problems in the context of the application scenario the algorithm has been developed for.

Hybrid hashing: The Hybrid Hashing adopts a very efficient filtering strategy, based on the use of two in-memory hash tables, to reduce the time spent for the point-in-polygon test. Moreover, it builds or updates the in-memory hash tables for the geofences during system spare time, shifting some computation from the point-in-polygon test stage (which is often time critical) to non-time-critical processing stage. This algorithms shows a low response time. CONs. The main drawbacks are the high storage requirements, the time required for the construction and update of index and data structures for the geofences. Finally Hybrid Hashing is not able to return trajectory-based spatio-temporal topological join predicates.

SLGC-1: This algorithms uses QuadTree compression algorithm to store data regarding geofences, which reduces the amount of memory requested for the index structure, the complexity of the storage is less then $O(n)$ and the approach to geofencing is a simple and straightforward point-in-polygon based solution. Lastly the point-in-polygon test time does not increase with the number of edges of the analyzed geofence. As previous algorithm it is not able to return spatio-temporal topological join predicates, the geospatial analysis is not deep.

PISTON: PISTON adopts a parallel in-memory indexing for trajectories and spatial geofences, which is a very scalable approach. Moreover, it evaluates the spatio-temporal topological join predicates with a sequence of topological relations that may hold between the trajectory of the moving object and the geofences at different time units, The trajectory index IR is optimized for high rate of location updates and can handle both coordinate-based and trajectory-based queries. PISTON adopts an efficient trajectory-filtering strategy and it is scalable due to its native multi-threaded setup. Main problems deals with the time required for the construction and update of the R-Tree index for the geofences is high and the high memory requirements.

PFLGA: PFLGA adopts an in-memory, tree-based indexing for the spatial polygons representing the geofences of interest, the time required for the construction and update of the in-memory index for the geofences is low. Moreover, it adopts an efficient trajectory-filtering strategy which checks for intersections between the MBR of the analyzed trajectory and the MBRs inside the in-memory index and is able to returns the trajectory-based spatio-temporal topological join predicates describing the relation between the moving object and each area in the set of geofences. Lastly, the algorithm can also evaluate the simple point-in-polygon test for the current position of the moving object if a simple geospatial analysis is requested. The main drawbacks is the response time of the trajectory-based topological join.

B. Performance Comparison

The performance of PFLGA is compared against the above discussed algorithms. In order to allow a fair comparison and avoid data dependencies, every test uses the same dataset used by the algorithm's author to determine the performance. The test has been executed on a laptop computer loaded with Intel Core i7 4720HQ CPU (Quad Core, 2.60 GHz up to 3.60 GHz) and 8 GB DDR3 RAM, running Linux Ubuntu 14.04 LTS using Apache Bench (also indicated as ab), a tool designed for benchmarking Apache installations and any HTTP server in general [26].

This section reports the dataset characteristics for each algorithms and the results in term of performance.

PISTON vs PFLGA: We use the following datasets to compare these algorithms:

1) Geofences Dataset: US TIGER Texas Arealm, a real-world spatial-objects dataset which contains 6694 geofences drawn in the Texas area [52];
2) Trajectories Dataset: 10000 trajectories between couples of random location point inside the Texas area, generated according to the specifications in the PISTON paper [48].

The test requires the execution of a geospatial query for each of the trajectories contained in the dataset.

The following table compares the performance of PFLGA against PISTON [2].

Hybrid Hashing vs. PFLGA: Also for this performance test we used two datasets , according to the paper [4]. The first dataset is provided by the ACM Open GIS Cup 2013 [4], which includes two location point files and two polygons files, for this comparison we selected two of these files, one for each category:

	PISTON	PFLGA
Average Spatial Index Creation Time (ms)	537000	860
Average Query Execution Time (ms)	200-550	330

TABLE II
PISTON VS. PFLGA

1) Geofences Dataset: Poly10 file, which contains 32 instances of 10 different polygons;
2) Locations Dataset: Point500, which contains 39289 instances of 500 different points.

The test requires the execution of 10000 geospatial queries for each of the trajectories contained in the dataset, according to the experiment specification in the paper. The table III contains the result obtained at the end of the test. According with

	Hybrid Hashing	Proposed
Average Spatial Index Creation Time (ms)	5	15
Average Query Execution Time (ms)	7	10

TABLE III
HYBRID VS PFLGA

Hybrid Hashing's Authors we used another dataset constructed as follows:

1) Geofences Dataset: Poly-OSM1, which contains 200 instances of 20 different polygons, selected from the Land Polygon dataset.
2) Locations Dataset: Point-OSM1, which contains 80000 instances of location points randomly selected from the MBR area of each polygon.

The test requires the execution of 10000 geospatial queries for each of the trajectories contained in the dataset. Table IV shows the result obtained.

	Hybrid Hashing	PFLGA
Average Spatial Index Creation Time (ms)	40.9	45.0
Average Query Execution Time (ms)	76.0	46.0

TABLE IV
HYBRID HASHING VS PFLGA

SLGC-1 vs. PFLGA: The two datasets used for this performance comparison are constructed according to the SLGC-1 paper [3]. This geofences dataset contains 5 different polygons, respectively having 5, 10, 50, 100 and 223 vertices, and a circular area. Each of these geospatial objects covers a geographical area almost equal to 4000 km^2, and Locations Dataset, 125000 different GPS location points limited in the geofences' areas. For each couple of GPS points a trajectory connectiong them is calculated. The test requires the execution of a geospatial query for each of the trajectories contained in the dataset. Table V shows the results of the comparison.

The comparison between PFLGA and the other solutions highlights the proposed algorithm has a wider set of features compared to the other solutions. In particular PFLGA is designed to perform the geofencing test with different geospatial objects, depending on the available information regarding the moving object, which offers a great flexibility depending on the use cases it is applied to. Moreover, in the inclusion test stage, PFLGA performances are comparable to, and in some case better than, those of the other designs, both in terms of average spatial index creation time, in the preprocessing stage, and average query execution time.

	SLGC-1	PFLGA
Average Spatial Index Creation Time (ms)	19.0	1.5
Average Query Execution Time (ms)	3255	3

TABLE V
SLGC-1 VS. PFLGA

V. CONCLUSIONS

This work has described the design process of a geofencing algorithm to be used to implement proactive, context aware functionalities in Supply Chain Management systems with real-time requirements.

PFLGA has been extensively set side by side to other existing geofencing designs, presented in the last few years in technical literature. PFLGA shows more features and comparable performance to the other geofencing solutions, making it more flexible to the different service requirements which may occur depending on available data or service requests.

Subsequently, PFLGA has been implemented in a back end software module within See Your Box test environment. The algorithm has been tested using data collected by a set of trackers along their journeys on different routes in Europe, proving its correctness and its ability to provide the result of the requested analysis within precise time constraints. Then the algorithm has been implemented as a service within See Your Box system. As for future developments, there is still room for optimization of the search strategy inside the set of geofences assigned to a specific mobile device, referred to as the tracker. Upcoming implementations may adopt a new indexing structure for the set of geofences, in order to further reduce the computational complexity and the response time of the algorithm. Another great opportunity for improvement concerns the route-matching service. Route-matching is used to determine the most plausible route traveled by the tracker, given as input a set of geographical points, each of which is coupled with a timestamp, which is then exploited in the proposed algorithm to perform a topological analysis to catch the occurrence of precise events. The introduction or improvement of route selection strategies based on the timestamps of the collected geographical points, or on the estimation of the cruising speed of the device, would allow to increase the accuracy of the result offered by the route-matching service, which would bring great benefit to the geofencing service in all those application scenarios within Supply Chain Management where the estimated trajectory traveled by the device represents a critical parameter (e.g. alert service in case of crossing a specific area not allowed by company policies).

REFERENCES

[1] S. R. Garzon and B. Deva, "Infrastructure-assisted geofencing: Proactive location-based services with thin mobile clients and smart servers," in *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, March 2015, pp. 61–70.

[2] S. Ray, A. D. Brown, N. Koudas, R. Blanco, and A. K. Goel, "Parallel in-memory trajectory-based spatiotemporal topological join," in *2015 IEEE International Conference on Big Data (Big Data)*, Oct 2015, pp. 361–370.

[3] K. Lin, Y. Chen, M. Qiu, M. Zeng, and W. Huang, "Slgc: A fast point-in-area algorithm based on scan-line algorithm and grid compression," in *2016 11th International Conference on Computer Science Education (ICCSE)*, Aug 2016, pp. 352–356.

[4] S. Tang, Y. Yu, R. Zimmermann, and S. Obana, "Efficient geo-fencing via hybrid hashing: A combination of bucket selection and in-bucket binary search," *ACM Trans. Spatial Algorithms Syst.*, vol. 1, no. 2, pp. 5:1–5:22, Jul. 2015. [Online]. Available: http://doi.acm.org/10.1145/2774219

[5] S. Steiniger, M. Neun, and A. Edwardes, "Foundations of location based services," 01 2006.

[6] I. Junglas and R. Watson, "Location-based services," vol. 51, pp. 65–69, 03 2008.

[7] V. Carchiolo, L. Compagno, M. Malgeri, N. Trapani, M. L. Previti, M. P. Loria, and M. Toja, "An efficient real-time monitoring to manage home-based oxygen therapy," in *Trends and Advances in Information Systems and Technologies*, Á. Rocha, H. Adeli, L. P. Reis, and S. Costanzo, Eds. Cham: Springer International Publishing, 2018, pp. 741–749.

[8] M. Loria, M. Toja, V. Carchiolo, and M. Malgeri, "An efficient real-time architecture for collecting iot data," 2017, pp. 1157–1166, cited By 1. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85039924461&doi=10.15439%2f2017F381&partnerID=40&md5=8de00d20f17d82a86236f43f33a716f8

[9] ESRI, "What is gis?" http://www.esri.com/what-is-gis, accessed: 2018.

[10] D. M. S. Sachin W. Rahate, "Geo-fencing infrastructure: Location based service," *International Research Journal of Engineering and Technology*, vol. 3, 11 2016.

[11] ITU, "Y.2060 : Overview of the internet of things," https://www.itu.int/rec/T-REC-Y.2060-201206-I, 06 2012, accessed: 2018.

[12] M. Rouse, "geo-fencing (geofencing)," http://whatis.techtarget.com/definition/geofencing, accessed; 2016.

[13] G. Allen, "Internet of things, industrial internet of things, industry 4.0its all connected! (no pun intended)," https://redshift.autodesk.com/industrial-internet-of-things-iot-terms/, accessed: 2015.

[14] M. Erwig and M. Schneider, "Developments in spatio-temporal query languages," in *Proceedings. Tenth International Workshop on Database and Expert Systems Applications. DEXA 99*, 1999, pp. 441–449.

[15] T. SearchNetworking, "Thin client (lean client) definition," accessed: 2016.

[16] M. Bauer, D. Dobre, N. Santos, and M. Schmidt, "Scalable processing of geo-tagged data in the cloud," *Nec Technical Journal*, vol. 7, no. 2, 2012.

[17] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions," 2007.

[18] A. L. B. Nagendra R. Velaga, Mohammed A. Quddus, "Developing an enhanced weight-based topological map-matchingalgorithm for intelligent transport systems," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 6, pp. 672–683, 12 2009.

[19] D. Pfoser and C. S. Jensen, "Capturing the uncertainty of moving-object representations," in *Proceedings of the 6th International Symposium on Advances in Spatial Databases*, ser. SSD '99. London, UK, UK: Springer-Verlag, 1999, pp. 111–132. [Online]. Available: http://dl.acm.org/citation.cfm?id=647226.719082

[20] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, "A characterization of ten hidden-surface algorithms," *ACM Comput. Surv.*, vol. 6, no. 1, pp. 1–55, Mar. 1974. [Online]. Available: http://doi.acm.org/10.1145/356625.356626

[21] G. Allen, "Harnessing the power of location based services," http://blogs.dcvelocity.com/supply_chain_innovation/2016/03/harnessing-the-power-of-location-based-services.html, accessed: 2016.

[22] B. Rao and L. Minakakis, "Evolution of mobile location-based services," *Commun. ACM*, vol. 46, no. 12, pp. 61–65, Dec. 2003. [Online]. Available: http://doi.acm.org/10.1145/953460.953490

[23] I. A. T. A. (IATA), "Guidance on the expanded use of passenger portable electronic devices (peds)," 2014.

[24] R. Ahas and lar Mark, "Location based servicesnew challenges for planning and public administration?" *Futures*, vol. 37, no. 6, pp. 547 – 561, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0016328704001521

[25] *DOC 9674/AN 946 - WGS84 Manual*, ICAO, 2002.

[26] T. A. S. Foundation, "ab - apache http server benchmarking tool," https://httpd.apache.org/docs/2.4/programs/ab.html, accessed: 2018.