Real-Time Analysis of Localization Data Streams for Ambient Intelligence Environments

Dimokritos Stamatakis, Dimitris Grammenos, and Kostas Magoutis[1]
Institute of Computer Science (ICS)
Foundation for Research and Technology Hellas (FORTH)
Heraklion GR-70013, Greece
{dstamat,gramenos,magoutis}@ics.forth.gr

Abstract. In this paper we describe a novel methodology for performing real-time analysis of localization data streams produced by sensors embedded in ambient intelligence (AmI) environments. The methodology aims to handle different types of real-time events, detect interesting behavior in sequences of such events, and calculate statistical information using a scalable stream-processing engine (SPE) that executes continuous queries expressed in a stream-oriented query language. Key contributions of our approach are the integration of the Borealis SPE into a large-scale interactive museum exhibit system that tracks visitor positions through a number of cameras; the extension and customization of Borealis to support the types of real-time analysis useful in the context of the museum exhibit as well as in other AmI applications; and the integration with a visualization component responsible for rendering events received by the SPE in a variety of human readable forms.

Keywords: Scalable stream processing, location-tracking via cameras.

1 Introduction

Ambient Intelligence (AmI) environments require the ability to sense physical locations in space (e.g., human visitors in a museum exhibit) and to interact with them in a context-specific manner. It is important in such environments to have the ability to detect interesting events (e.g., a visitor approaching a specific spot), interesting behavior within sequences of events (e.g., visitor is following a specific path), and provide easy access to statistical information (e.g., popularity of certain exhibit over a specific time window in the past) in real-time. There are several ways to sense presence and location (using cameras, ultrasound, or magnetic field sensors) and transmit it to an AmI application in the form of localization data streams. However, performing complex processing of those streams is currently done in an ad-hoc manner, constraining the ability of AmI deployments to evolve and scale over time. In this paper we propose a novel methodology to process localization data streams using a new class of data processing systems called continuous stream-processing engines [2]. The advantages of our approach include the ability to express interesting events and behavior in terms of continuous-queries expressed in a structured stream-oriented

[1] We thankfully acknowledge the support of the European FP7-ICT program through the SCALEWORKS (MC IEF 237677) and CUMULONIMBO (STREP 257993) projects.

query language. Besides the extensibility that this allows, our approach offers the ability to scale the stream-processing engine as the volume of events increases [3] and to tolerate failures in the underlying infrastructure [4]. In this paper we improve on the state of the art by presenting a proof-of-concept prototype that integrates an open-source SPE (Borealis [2]) with an interactive museum exhibit within which a computer vision system [1] uses cameras to track the position of visitors; describing our extensions to the Borealis SPE for expressing the different types of events, behavior, and statistics that are useful in the museum exhibit and more generally in AmI applications; and finally, describing our novel visualization component in terms of both its interaction with the SPE as well as its feature-rich user interface.

2 The stream-processing engine

Stream processing engines (SPEs) support the execution of continuous queries expressed in stream processing languages. The data operated on (often referred to as *tuples*) are associated with a monotonically increasing timestamp, forming a time series. An example of a tuple typical in localization data streams consists of the fields <visitor ID, x, y, epoch>, meaning a specific visitor was observed in "box" (x, y) within a specific epoch. A continuous stream processing query is composed of one or more interconnected operators, each computing a function (such as sum, average, max – in general any aggregation function) over sets (also known as *windows*) of tuples. A window selects all tuples that satisfy a certain predicate on one or more of the tuple fields, also referred to as the *group-by fields* (e.g., group-by (x, y) selects all tuples carrying the same (x, y) values). Standard SPEs such as Borealis close a window either when a certain number of tuples have entered that window (count-based) or a timeout has been reached (time-based) and produce an output tuple at the closing of a window. We have extended Borealis to support the production of an output tuple either at the opening of a window, or periodically while the window is open, or (the default) at closing time. Additionally, we allow closing of a window w_i either when a *correlated* window w_j opens (where correlated is defined by w_i, w_j sharing a value at a certain field or satisfying some other predicate on their field values) or when the window is not updated during one or more tuple epochs (where epoch is defined based on a specific tuple field). Our SPE currently supports a range of event-processing queries used in the interactive museum exhibit described in Section 3:

Popularity of a box (x, y): At the core of this query is an aggregate operator grouping tuples by "box" (x, y) and counting visitor ID observations within each box. The operator produces output tuples periodically and windows remain open indefinitely.
Time visitor spent in the room: This query is similar to the previous one except that the aggregate operator groups tuples by visitor ID.
Trajectory of a visitor: This query uses an aggregate operator grouping tuples by (x, y, visitor ID) and produces an output tuple only when a window opens. A window (x, y, visitor ID) closes when a correlated window (x′, y′, visitor ID) opens.
Reduce flicker/noise in the localization stream: This query uses an aggregate operator grouping tuples by (x, y, visitor ID) and producing an output tuple when a window opens. In this query we use a different definition of correlation: A window w_i = (x, y, visitor ID) closes when a window (x′, y′, visitor ID) opens whose distance to

w_i exceeds a certain radius R. Since a number of windows can be within R, a window opening can cause the closure of several windows. This query avoids producing unnecessary outputs when a visitor flickers (goes back and forth) between neighboring boxes. This query can also detect the case when a visitor travels over long distances rapidly by checking for window distance exceeding a large constant R′.
Visitor appears or disappears: This query uses an aggregate operator grouping tuples by visitor ID and producing an output tuple when a window opens, identifying a new visitor and its location (x, y). The window state is always the last position seen of a given visitor ID and its associated epoch. Windows must be updated at each epoch unless cameras can no longer locate the visitor. This query identifies internally whether a window misses an epoch update and if so, closes the window, producing a tuple with the location of the lost visitor. A subsequent operator checks whether the event took place at a valid (e.g., a door) or invalid (e.g., middle of the room) location.

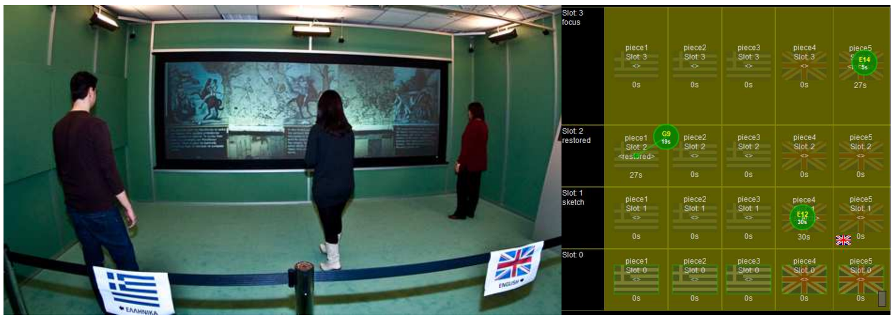

Fig. 1. (left) Overview of the museum exhibit; (right) zone map

3 Application example: Interactive museum room

The extended SPE system described in Section 2 was integrated into a large-scale interactive museum exhibit that takes up a whole room (6x6x2.5m^3). In this room a computer vision subsystem (for more details see [1]) with 8 cameras tracks the position of visitors. On one wall a dual-projector back-projection screen is installed. Localization of persons is performed at 10Hz and has an accuracy of ~2cm. In our demo installation (Fig. 1 - left), the projection screen presents a wall painting. Visitors enter the room from an entrance opposite the display. The vision system assigns a unique ID number to each person entering the room. The room is conceptually split in 5 zones of interest, delimited by different themes presented on the wall painting. These zones cut the room in 5 vertical slices. The room is also split in 4 horizontal zones that run parallel to the wall painting, which are delimited by their distance from it. Thus, a 5x4 grid is created, comprising 20 interaction slots (Fig. 1 - right). When a visitor is located over a slot, the respective wall painting part changes and, depending on the slot's distance from the wall, visitors can see a sketch, a restored version or a detail of the wall part, accompanied by related information. All information is presented in the user's preferred language.

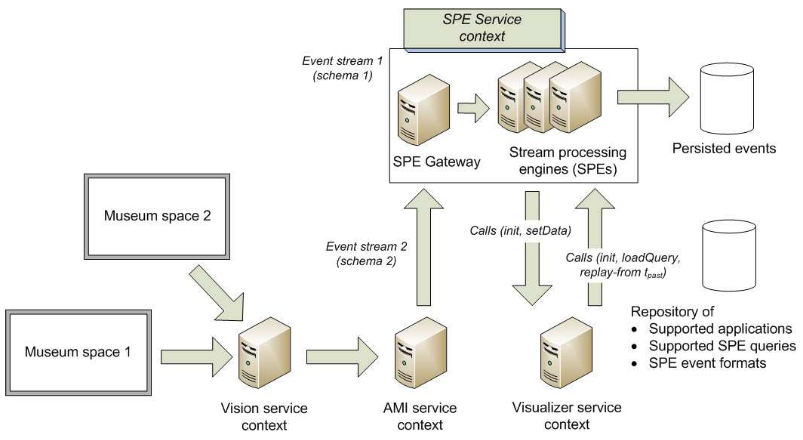

Fig. 2. System setup for the museum exhibit application

A schematic representation of the full system setup is illustrated in Fig. 2: The computer vision subsystem emits localization events (comprising user IDs and their position in space) which are picked up by the interactive application that is responsible for: (a) updating the visual information presented on the projection screen; and (b) providing the SPE subsystem with both low-level (e.g., position) and semantic (e.g., interaction slot that the user is on, user language) localization data. The SPE analyzes and stores this data on-the-fly and feeds a visualization component which presents quantitative and qualitative information about the exhibit's visitors.

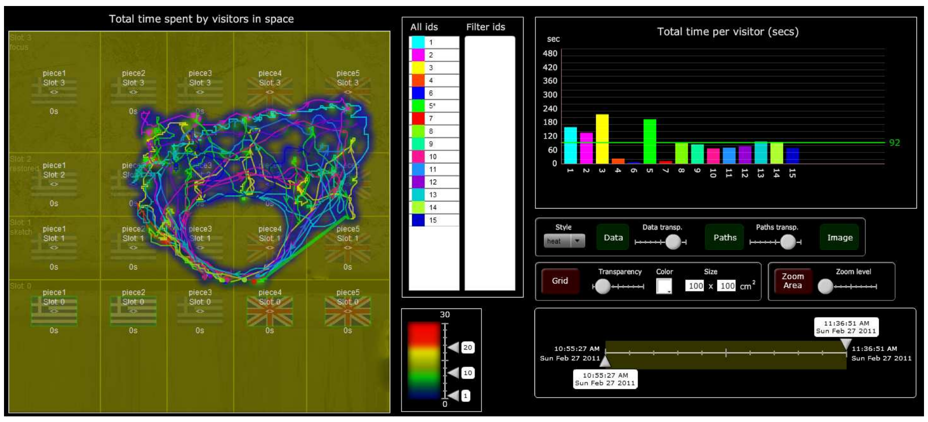
Total time spent by visitors in space
All ids
Filter ids
Total time per visitor (secs)
sec
Style
Data
Data transp.
Paths
Paths transp.
Image
Grid
Transparency
Color
Size
Zoom Area
Zoom level
10:55:27 AM Sun Feb 27 2011
11:36:51 AM Sun Feb 27 2011

Fig. 3. User interface of the visualization component

More specifically, the SPE propagates: User position; Accumulated time spent by visitors in specific positions; Total time spent by each visitor in the exhibit, average time for all users; "interesting" or "abnormal" patterns, such as: (i) too long distance between consecutive user positions; (ii) too short / long user stays in the room; (iii) incomplete user visits (where a user appears or disappears in locations other than entries/exits). The visualization component interacts with the SPE to set parameters such as type of data it is “interested” in and how to analyze and interpret them (e.g., how long should the distance between user observations be to be considered “abnormal”).

The main part of the visualization component’s user interface (Fig. 3) comprises a top-down view of the room and its interactive slots (as in Fig. 1 right). On this view user paths are presented (using a different color for each distinct user). “Abnormal” (i.e., too long) path sections are presented using thicker lines. Additionally, accumulated user time spent on a specific area is illustrated in the form of a heat map. Users can get quantitative information about each point of the heat map by positioning the mouse pointer over it. Additionally, they can modify several parameters of this view, for example changing the transparency level of the paths or the heat map, hiding the background image, overlaying a grid and changing the coloring scheme of the heat map. Furthermore, they can zoom in any part of the view at different levels.

The interface also hosts a list which contains all detected user IDs. This list has a multifunctional role, since it: (i) works as a legend for correlating path colors to user IDs; (ii) highlights user IDs related to identified patterns (e.g., an asterisk ‘*’ is appended to IDs related to long path sections, a minus ‘-‘ to IDs with incomplete paths); and (iii) can be used to filter out data by selecting a subset of IDs (simply by clicking on them). Next to the IDs list, there is a graph presenting the total time spent by each visitor in the room and the average time for all visitors. Finally, there is an interactive control for setting a time period of interest that can be used for filtering data according to the time of their creation. The visualization component can be used to work with both real-time and stored data.

4 Conclusions & Future Work

In this paper we presented a novel approach to extensive and scalable real-time analysis of sensor data in AmI Environments. Currently, the presented solution has been installed and tested in a laboratory space at ICS-FORTH containing a fully working version of the aforementioned museum exhibit. Since this exhibit is also installed in a permanent exhibition of a major museum in Greece, our next steps include testing our proposed solution under real conditions.

References

1. X. Zabulis, D. Grammenos, T. Sarmis, K. Tzevanidis, A. A. Argyros. Exploration of Large-scale Museum Artifacts through Non-instrumented, Location-based, Multi-user Interaction. In Proc. of VAST 2010, Paris, France, 21-24 September 2010.
2. D. Carney et al. Monitoring Streams: A New Class of Data Management Applications. In Proc. of the 28th VLDB, Hong Kong, China, August 2002.
3. Y. Ahmad et al. Distributed Operation in the Borealis Stream Processing Engine. In Proc. of the 2005 SIGMOD, Baltimore, MD, June 2005.
4. Z. Sebepou and K. Magoutis. CEC: Continuous Eventual Checkpointing for Data Stream Processing Operators. In Proc. of 41st IEEE/IFIP DSN, Hong Kong, China, June 2011.