RACER: Real-Time Automated Complex Event Recognition in Smart Environments

Rohit Verma, Justas Brazauskas, Vadim Safronov, Matthew Danish, Ian Lewis, Richard Mortier

University of Cambridge
Cambridge, United Kingdom

{rv355,jb2328,vs451,mrd45,ijl20,rmm1002}@cam.ac.uk

ABSTRACT

As smart environments become laden with more and more sensors, there has been a need to develop systems that could derive useful information from these sensors and make the smart environments smarter. Complex Event Processing (CEP) has emerged as a popular strategy to identify crucial events from sensor data. However, the existing CEP strategies overlook the relationship with other sensors in the spatial vicinity and understate the temporal variation of sensor data. In this paper, we develop *RACER*, which is an end-to-end complex event processing system that takes into consideration both the spatial location of the sensor in observation and the varying impact of temporal changes in the sensor data. Experiments performed for a duration of five months over both collected and live streaming data shows that *RACER* fares well compared to the other state-of-the-art approaches.

CCS CONCEPTS

• **Human-centered computing** → Ubiquitous and mobile computing; • **Information systems** → Spatial-temporal systems.

KEYWORDS

real-time spatio-temporal data, complex event processing, smart environments

ACM Reference Format:

Rohit Verma, Justas Brazauskas, Vadim Safronov, Matthew Danish, Ian Lewis, Richard Mortier. 2021. *RACER: Real-Time Automated Complex Event Recognition in Smart Environments*. In *29th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '21)*, November 2–5, 2021, Beijing, China. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3474717.3484270>

1 INTRODUCTION

The Smart Environment – a physical world where connected devices like sensors, actuators, and computational elements work together to make users’ lives more comfortable – has seen several advances in recent years. Several applications are designed to achieve the goal of user comfort like remote health monitoring, air quality monitoring, power-grid management, building automation, etc. The key components of these aforementioned applications are the

events that either bring in opportunities (lowering the power usage in vacant rooms) or threats (gas leaks or fires). Effective and fast identification of these events is crucial for any such application. These events could be simple (atomic) events like a patient missing an appointment or complex like power failure in a complete building unit.

Recently, Complex Event Processing (CEP) [5] has been of interest in the field of smart environment applications [4, 7, 17]. CEP utilizes the understanding that multiple atomic/simple events are an indication of a much important complex event.

There have been several works that utilize a CEP system to design alerting applications [11, 16]. The approach is to assign area experts who identify a set of atomic events and design rules which could lead to a complex event. These rules are then fed into a CEP engine to be matched later from the incoming event streams.

These state-of-the-art approaches overlook important points. First, the atomic event detection methods only consider a single sensor’s data for an atomic event. However, other sensors in the spatial vicinity would also have a relationship with the sensor’s reading [12]. For example, in a smart wearable, simply using the accelerometer to detect walking could be erroneous as a person swaying their hand could give similar signatures of walking. Similarly, a high-temperature reading on a sensor in a room would have a relationship with other temperature or humidity sensors in the room. Second, the time of the year/day when the data is being observed is also important. High temperature during summers is normal but a similar high temperature during winters could mean a broken heating system.

In light of this, we develop an end-to-end system, *Real-time Automated Complex Event Recognizer (RACER)*, which provides a robust complex event detection mechanism ensuring that the spatial positioning of the sensor being observed and the temporal variation in data is taken into account when doing any computation. We first extract two sets of features (§ 3) from incoming sensor data; relationship with other sensors in the spatial vicinity of the sensor being observed, and the temporal features linked to the sensor data. Following this, we design a model (§ 3.2) which takes these features as input and detects if an atomic event has occurred. Next, we design a CEP engine, which takes as input atomic events and a set of user-generated rules to detect complex events in the incoming data stream (§ 4). We evaluate the end-to-end system over our deployment of a National Digital Twin in the university campus, consisting of five buildings (§ 5).

2 RELATED WORK

Complex Event Processing (CEP) [5] is an approach that observes real-time events in data streams and the causal relationship between

This work is licensed under a Creative Commons Attribution International 4.0 License.
SIGSPATIAL '21, November 2–5, 2021, Beijing, China
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8664-7/21/11.
<https://doi.org/10.1145/3474717.3484270>

Figure 1: System Architecture

these events to identify opportunities or threats as they happen. The usual approach is to assign area experts the task to define such rules in a query language [2] and deploy these as operator graphs [8] used to match event patterns in the incoming event streams.

CEP systems have seen wide usage in numerous use-case scenarios like stock market trading [14], predictive maintenance of sensors/equipments in large facilities [15], and real-time marketing [1]. Recently, there has been increasing interest towards applying CEP to autonomous vehicles [6]. CEP is also being widely used in smart environments which we discuss next.

In a smart environment, it is critical to react to any opportunities or threats with low latency. These could be accidents like fire or gas leaks, sudden change in weather conditions, power-grid faults, sudden load variations, etc. A CEP system continuously monitoring event streams from the deployed sensors would serve well to address these scenarios. Many works have employed CEP to such scenarios in the context of smart-grids [4, 10], smart-hospitals [16, 17] or smart-buildings [7, 11].

Although there has been a lot of research in the field of CEP systems, in these systems, the relationship with the sensors in the spatial vicinity is not taken into consideration [7, 10, 11]. However, several works have pointed out that the readings in the same spatial vicinity, be it a room, building or just a sensor box, are correlated to each other [12]. Essentially, the need is to identify atomic events based on the temporal distribution of sensor data over a period and not the instantaneous values or a minor divergence from the data distribution.

3 ATOMIC EVENT DETECTION MODULE

As shown in Fig 1, the Atomic Event Detection Module comprises of the Feature Extractor and Atomic Event Detector. We describe these two sub-modules in this section.

3.1 Feature Extractor

Any atomic event linked to a single sensor could be defined as a function of the data it generates (X), its location in the smart

environment ($\mathcal{L}$) and at what time is the data collected (T). Hence, the system needs to extract features that cover all three aspects of the sensor data. The *Feature Extractor* thus works with three types of features; (a) correlation with nearby sensors, (b) raw sensor data, and (c) temporal parameters.

3.1.1 Sensor Correlation Score. The Sensor Correlation Score (C), extracts the impact of the sensor's location. The primary idea being that any other sensor in the spatial vicinity of the sensor being observed would in one way or another impact the data being transmitted.

We define two parameters before computing C . First is the *spatial vicinity* (S)— which is the spatial area inside which a sensor (s_c) to be correlated with the sensor being observed (s_o) lies. S is a configurable parameter based on application requirements. Second, the time window ω over which the correlation is to be measured. This is important both for storage purposes (larger ω would require higher memory consumption)

Once S and ω are defined, the Sensor Correlation Score for the sensor s_o with respect to sensor s_c , represented as C_i , is computed as the Pearson Correlation Coefficient [9] between the readings of s_o and s_c in the time window ω . For n sensors in the vicinity, we obtain a vector $C_{i=1}^n$

3.1.2 Temporal Parameters. The timestamp of sensor data collection would impact the predictions linked to any event. For example, low readings on a temperature sensor in winter is obvious, but similar reading in summer could be an HVAC issue. Hence, to capture the temporal characteristics of the sensor readings we extract three temporal parameters, *month* (m), *day* (d), and *hour* (h).

These set of features, along with the sensor readings in ω time window, and the type of sensor compose the input vector at time t , represented as,

$$\mathcal{I}_t = \{(\mathcal{X} = \{x_i | i \in [t - \omega, t]\}), C_{i=1}^n, m, d, h, s\}$$

3.2 Atomic Event Detector

The *Atomic Event Detector* takes the input vector I_t , obtained from the feature extractor, and predicts if it signifies any atomic event in ω time window. We design the *Atomic Event Detector* using a Convolution Neural Network (CNN), and train for each event in isolation. The model is trained using sparse categorical cross-entropy loss function with 60 – 20 – 20% data split for training, validation, and testing.

The identified atomic event, if any, are then fed into the CEP Module for further processing. It should be noted that at the same time instance multiple atomic events could be detected which are fed into the CEP module.

4 cep-LAYER: COMPLEX EVENT PROCESSING

The *cep-layer* takes care of identifying a complex event from the incoming atomic, based on human-generated rules. In this section, we will first describe how the rules are generated and their structure and following that describe how the *cep-Engine* works.

4.1 CEP Rules

Given a set of n atomic events, $\{E_a^i\}_{i \in (1,n)}$, a complex event can be defined as a function of one or more of these atomic events, represented as;

$$E_c^i = f(E_a^1(e^1, t^1, v^1, s^1), E_a^2(e^2, t^2, v^2, s^2), \dots) \quad (1)$$

where, e , t , v , and s are the atomic event id, timestamp, the sensor data reading, and the sensor associated with the complex event.

As we observe, from the above example, a complex event rule depends on the type of the event, when the event occurred, what values were linked to the atomic event, and the location of the sensor which provided those readings. Hence, we define a template for any complex event as;

$$\begin{aligned} E_c^i(e^i, t^i, [sensor_ids]^i) \\ \leq E_a^x(e^x, t^x, v^x, s^x) \cap E_a^y(e^y, t^y, v^y, s^y) \\ \cap t^x < t^y \\ \cap v^x \leq \mathcal{V}_1 \\ \cap v^y \geq \mathcal{V}_2 \\ \cap distance(s^x, s^y) < \Delta \end{aligned} \quad (2)$$

The above rule states that, the complex event E_c^i is implied when both atomic events E_a^x and E_a^y have been detected and E_a^x occurred before E_a^y . Additionally, the sensor data values at the time of detection were related to the thresholds $\mathcal{V}_1$ for E_a^x and $\mathcal{V}_2$ for E_a^y and both sensors were within Δ distance of each other. The rule outputs the detected complex event id, the detection time and the sensors which were used to identify the complex event.

4.2 CEP Engine

Owing to its roots in formal logic, we use Prolog to design the CEP engine. The incoming atomic events and the fed back complex events are stored as a set of Prolog *facts*. However, this set of *facts* is kept of a finite size $\mathcal{K}$, and the arrival of a new atomic/complex event removes the oldest entry. This is done to keep only the events that

are relevant at the current time. The parameter $\mathcal{K}$ is configurable and depending on the requirements could be increased or decreased. The CEP rules are loaded as Prolog *rules* in the environment.

Once a pattern match is obtained by the CEP engine, it outputs the detected complex event, the timestamp when it is detected and the sensor's responsible for the event.

5 EVALUATION

We performed the experiments on *RACER* with 185 LoRaWAN sensors, the data of which arrived from five different buildings that included two offices and three residential buildings. The data used in this section was collected in experiments run from January 2021 to May 2021, for a period of five months.

Total	None	CO ₂ Rise	CO ₂ Fall	CO ₂ DC	Temp DC
3553027	3494477	1032	3864	114	59
Lux Drop	Lux Rise	Water Low	Occ Mod.	Occ High	Motion
793	9764	350	685	5	41884

Table 1: Number of training data points per atomic event. (DC: At Discomfort level, Mod.: Moderate)

Corridor Walk (C1)	Group Meet (C2)	Cook (C3)	Office Vacated (C4)	Window Opened (C5)
50	10	100	140	80

Table 2: Number of data points per complex event labelled in the data. The label in brackets are what these complex events will be referred to as henceforth in the text.

Table 1 lists the number of atomic events used in training the CNN model. The number of complex events identified in the live data stream and the existing training data is listed in Table 2.

5.1 Comparison with Competing Systems

We compare *RACER* with three existing systems which also perform complex event processing for smart environments.

5.1.1 Raj et al. [11]. This work develops a system that performs threshold-based atomic event detection and then feed these detected atomic events to the CEP rule engine to identify complex events in smart buildings.

5.1.2 Sun et al. [13]. This work relies solely on anomaly detection to identify any complex events in the building. The system uses an Isolation Forest (IFO) [3] algorithm for anomaly detection in the incoming data, however, only detects complex events that could be observed from one sensor at a time.

5.1.3 Mongiello et al. [7]. This system is built to alert for fire and danger management but is easily extensible to other complex event detection. Instead of a single threshold based atomic event detection, the system uses a threshold range.

We ran the four systems along with *RACER* for detection of the five complex events as listed in Table 2. In our implementation, we used sensors that were in the same room or the same sensor box as being in the vicinity of each other. We set the configurable

Figure 2: Comparing F1-Score of Complex Event Detection with competing systems for all five complex events.

parameter $\mathcal{K}$, which limited the number of facts with which the CEP engine works as 100 based on empirical results.

Figure 2 shows the F1-Score for the complex events and the average F1-score for all four systems. *RACER* fairs better compared to all other systems, and there are few important points to be highlighted regarding this result. As Sun et al. [13] only works with single sensor data, hence, it wasn't able to detect C1. Moreover, for other events too, we had to select one sensor which gave the best result for this system. Although the result for other systems seems similar for C1, it should be noted that the CEP rule had to be different for Mongiello et al. [7] and Raj et al. [11].

5.2 Real-time Analysis of *RACER*

	Feature Extraction	Atomic Event Detection	cep-layer
Latency (ms)	1.21	35.57	37.35

Table 3: Mean latency at key points of the work flow in *RACER* from message arrival time.

As a real-time system, *RACER* should complete computations with minimal latency. As we don't have any control over the latency of the sensor data originating at the sensor and reaching the system, we compute the latency from the time when the sensor data arrives at the system. We consider three key points in the *RACER* workflow: (i) feature generation, (ii) atomic event detection, (iii) cep-layer.

Table 3 shows that at all three points the latency is minimal. Owing to the neural networks processing, the atomic event detector takes 34.36ms. However, the other feature extractor and cep-layer show negligible latency of 1.21ms and 1.78ms respectively. For all practical purposes, an average latency of 37.35ms is negligible validating that *RACER* could operate as a real-time system.

6 CONCLUSION

As smart environments turn smarter, detection of complex events becomes a crucial requirement. *RACER* provides a robust end-to-end system that detects such complex events using sensor data received from these smart environments in real-time. *RACER* achieves this goal by considering several key aspects: (i) the impact of other

sensors in the spatial vicinity of the sensor being observed, (ii) the temporal variations in the data that arrives from the sensors.

Experiments run for five months show that *RACER* can detect complex events with high accuracy. There do exist key aspects of automatic CEP rule generation, and timeliness of data, which require further analysis to improve *RACER*.

ACKNOWLEDGMENTS

This research forms part of Centre for Digital Built Britain's work within the Construction Innovation Hub. The funding was provided through the Government's modern industrial strategy by Innovate UK, part of UK Research and Innovation.

REFERENCES

- [1] Mohamed H Ali, Ciprian Gerea, Balan Sethu Raman, Beysim Sezgin, Tiho Tarnavski, Tomer Verona, Ping Wang, Peter Zabback, A Ananthanarayan, A Kirilov, et al. 2009. Microsoft CEP server and online behavioral targeting. *Proceedings of the VLDB Endowment* 2, 2 (2009), 1558–1561.
- [2] Gerald G Koch, Boris Koldehofe, and Kurt Rothermel. 2010. Cordies: Expressive event correlation in distributed systems. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*. 26–37.
- [3] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth IEEE international conference on data mining*. IEEE, 413–422.
- [4] Guangyi Liu, Wendong Zhu, Chris Saunders, Feng Gao, and Yang Yu. 2015. Real-time complex event processing and analytics for smart grid. *Procedia Computer Science* 61 (2015), 113–119.
- [5] David C Luckham and Brian Frasca. 1998. Complex event processing in distributed systems. *Computer Systems Laboratory Technical Report CSL-TR-98-754*. Stanford University, Stanford 28 (1998), 16.
- [6] István Majzik, Oszkár Semeráth, Csaba Hajdu, Kristóf Marussy, Zoltán Szatmári, Zoltán Micskei, András Vörös, Aren A Babikian, and Dániel Varró. 2019. Towards system-level testing with coverage guarantees for autonomous vehicles. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 89–94.
- [7] Marina Mongiello, Luigi Patrono, Tommaso Di Noia, Francesco Nocera, Angelo Parchitelli, Ilaria Sergi, and Piercosimo Rametta. 2017. A complex event processing based smart aid system for fire and danger management. In *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IASI)*. IEEE, 44–49.
- [8] Beate Ottenwälder, Ruben Mayer, and Boris Koldehofe. 2014. Distributed complex event processing for mobile large-scale video applications. In *Proceedings of the Posters & Demos Session*. 5–6.
- [9] K Pearson. 1895. Notes on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London*, 58, 240–242.
- [10] Manuel Peralta, Supratik Mukhopadhyay, and Ramesh Bharadwaj. 2009. A Formal Framework for Developing High Assurance Event Driven Service-Oriented Systems. In *High Assurance Services Computing*. Springer, 145–161.
- [11] Rohit Raj, Rajat Kumar Sahu, Biki Chaudhary, Bakshi Rohit Prasad, and Sonali Agarwal. 2017. Real time complex event processing and analytics for smart building. In *2017 conference on information and communication technology (CICT)*. IEEE, 1–6.
- [12] Huizhong Shen, Weiying Hou, Yaqi Zhu, Shuxiu Zheng, Subinuer Ainiwaer, Guofeng Shen, Yilin Chen, Hefa Cheng, Jianying Hu, Yi Wan, et al. 2021. Temporal and spatial variation of PM_{2.5} in indoor air monitored by low-cost sensors. *Science of The Total Environment* 770 (2021), 145304.
- [13] Alexander Y Sun, Zhi Zhong, Hoonyoung Jeong, and Qian Yang. 2019. Building complex event processing capability for intelligent environmental monitoring. *Environmental modelling & software* 116 (2019), 1–6.
- [14] Kia Teymourian, Malte Rohde, and Adrian Paschke. 2012. Knowledge-based processing of complex stock market events. In *Proceedings of the 15th International Conference on Extending Database Technology*. 594–597.
- [15] Klaus-Dieter Thoben, Abderrahim Ait-Alla, Marco Franke, Karl Hribernik, Michael Lütjen, and Michael Freitag. 2018. Real-time Predictive Maintenance Based on Complex Event Processing. *Enterprise Interoperability: Smart Services and Business Impact of Enterprise Interoperability* (2018), 291–296.
- [16] V Vaidehi, R Bhargavi, Kirupa Ganapathy, and C Sweetlin Hemalatha. 2012. Multi-sensor based in-home health monitoring using complex event processing. In *2012 International Conference on Recent Trends in Information Technology*. IEEE, 570–575.
- [17] Wen Yao, Chao-Hsien Chu, and Zang Li. 2011. Leveraging complex event processing for smart hospitals using RFID. *Journal of Network and Computer Applications* 34, 3 (2011), 799–810.