Noname manuscript No.
(will be inserted by the editor)

A Probabilistic Interval-based Event Calculus for Activity Recognition

Evangelos Makris · Alexander Artikis · Georgios Paliouras

Received: date / Accepted: date

Abstract Activity recognition refers to the detection of temporal combinations of 'low-level' or 'short-term' activities on sensor data. Various types of uncertainty exist in activity recognition systems and this often leads to erroneous detection. Typically, the frameworks aiming to handle uncertainty compute the probability of the occurrence of activities at each time-point. We extend this approach by defining the probability of a maximal interval and the credibility rate for such intervals. We then propose a linear-time algorithm for computing all probabilistic temporal intervals of a given dataset. We evaluate the proposed approach using a benchmark activity recognition dataset, and outline the conditions in which our approach outperforms time-point-based recognition.

1 Introduction

Complex event recognition concerns temporal pattern matching on sensor data [4, 19]. Event recognition systems have been used in various applications, ranging from the recognition of attacks in computer network nodes [22], to human activities [11], and suspicious and dangerous behaviour in the maritime and aviation domains [50, 65]. In activity recognition, multiple sources provide spatial and temporal data that can be used to detect various types of human behaviour. The input data are *short-term* activities (STA), such as 'walking', 'running', 'active' and 'inactive', indicating that a person is walking, running, moving his arms while in the same position, and so on. The output is a set of *long-term* activities (LTA), which are

E. Makris
Institute of Informatics & Telecommunications, NCSR 'Demokritos', Athens, Greece
E-mail: vmakris@iit.demokritos.gr

A. Artikis
Institute of Informatics & Telecommunications, NCSR 'Demokritos', Athens, Greece
Department of Maritime Studies, University of Piraeus, Greece
E-mail: a.artikis@unipi.gr

G. Paliouras
Institute of Informatics & Telecommunications, NCSR 'Demokritos', Athens, Greece
E-mail: paliourg@iit.demokritos.gr

temporal combinations of STA. Examples are 'fighting', 'meeting', 'moving', etc. When a rule that consists of temporal constraints on a set of STA is satisfied, it leads to the recognition of LTA.

Uncertainty is inherent in activity recognition. For example, STA, typically detected by visual information processing tools operating on video feeds, often have probabilities attached to them, serving as a confidence estimates. In earlier work, we presented an activity recognition system based on a probabilistic version of the Event Calculus [35], hereafter Prob-EC, that computes the probability of an LTA at each time-point [60]. We extend this approach by defining the probability of a maximal interval and the credibility rate for such intervals. We then propose a linear-time algorithm for computing all maximal temporal intervals that satisfy a given probability threshold. Our contributions may thus be summarised as follows:

- We define the notion of probabilistic maximal interval, and a credibility rate for such intervals, that allow us to perform interval-based activity recognition under uncertainty.
- We propose a linear-time algorithm for computing all credible, probabilistic maximal intervals of a given dataset.
- We evaluate our algorithm on a benchmark activity recognition dataset, and outline the conditions in which it outperforms time-point-based recognition. Our approach is robust to noisy LTA probability fluctuations, and performs better in the common case of non-abrupt probability change.

The rest of the paper is organized as follows. In Section 2 we present the background of our work. In Section 3 we define the problem that we study, and present a linear-time algorithm for interval-based activity recognition. Then, Section 4 presents the empirical analysis. Finally, in Section 5 we put the work in context, and in Section 6 we summarise our findings and present further work directions.

2 Background

We use Prob-EC [60], a probabilistic version of the Event Calculus, to compute the probability of an LTA of interest at each time-point. We will first present the Event Calculus, and then its probabilistic, point-based implementation.

2.1 Event Calculus

The Event Calculus is a logic programming language for representing and reasoning about events and their effects [35]. We restrict attention to a simple version of the Event Calculus where the time model is linear and includes integer time-points. 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. Table 1 presents the main predicates of this simple Event Calculus.

Predicate	Meaning
$\mathsf{happensAt}(E,\ T)$	Event E is occurs at time T
$\mathsf{holdsAt}(F=V,\ T)$	The value of fluent F is V at time T
$\mathsf{initiatedAt}(F=V,\ T)$	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

Table 1: Main Predicates of the Event Calculus.

The domain-independent axioms are presented below:

$$\begin{aligned}&\mathsf{holdsAt}(F\ =V,\ T)\leftarrow\\&\quad\mathsf{initiatedAt}(F=V,\ T_s),\ T_s<T,\\&\quad\mathsf{not\ broken}(F=V,\ T_s,\ T).\end{aligned}\tag{1}$$

$$\begin{aligned}&\mathsf{broken}(F=V,\ T_s,\ T)\leftarrow\\&\quad\mathsf{terminatedAt}(F=V,\ T_f),\ T_s<T_f<T.\end{aligned}\tag{2}$$

$$\begin{aligned}&\mathsf{broken}(F=V,\ T_s,\ T)\leftarrow\\&\quad\mathsf{initiatedAt}(F=V',\ T_f),\ V\neq V',\ T_s<T_f<T.\end{aligned}\tag{3}$$

According to axiom (1), $F=V$ holds at some time-point T if it has been initiated by an event previously and has not been 'broken' in the meantime. This expresses the law of inertia. $F=V$ is 'broken' in (T_s,T) if it is terminated (see axiom (2)) or $F=V'$ is initiated, for some $V'\neq V$ (see axiom (3)). The definitions of $\mathsf{initiatedAt}$ and $\mathsf{terminatedAt}$ are domain-specific. Consider, for example, the (partial) definition of *moving* from the domain of activity recognition:

$$\begin{aligned}&\mathsf{initiatedAt}(\mathit{moving}(P_1,P_2)=\mathsf{true},\ T)\leftarrow\\&\quad\mathsf{happensAt}(\mathit{walking}(P_1),\ T),\\&\quad\mathsf{happensAt}(\mathit{walking}(P_2),\ T),\\&\quad\mathsf{holdsAt}(\mathit{close}(P_1,P_2)=\mathsf{true},\ T),\\&\quad\mathsf{holdsAt}(\mathit{similarOrientation}(P_1,P_2)=\mathsf{true},\ T).\end{aligned}\tag{4}$$

$$\begin{aligned}&\mathsf{terminatedAt}(\mathit{moving}(P_1,P_2)=\mathsf{true},\ T)\leftarrow\\&\quad\mathsf{happensAt}(\mathit{walking}(P_1),\ T),\\&\quad\mathsf{holdsAt}(\mathit{close}(P_1,P_2)=\mathsf{false},\ T).\end{aligned}\tag{5}$$

$$\begin{aligned}&\mathsf{terminatedAt}(\mathit{moving}(P_1,P_2)=\mathsf{true},\ T)\leftarrow\\&\quad\mathsf{happensAt}(\mathit{active}(P_1),\ T),\\&\quad\mathsf{happensAt}(\mathit{active}(P_2),\ T).\end{aligned}\tag{6}$$

$$\begin{aligned}&\mathsf{terminatedAt}(\mathit{moving}(P_1,P_2)=\mathsf{true},\ T)\leftarrow\\&\quad\mathsf{happensAt}(\mathit{running}(P_1),\ T).\end{aligned}\tag{7}$$

moving is a long-term activity (LTA) expressed as a Boolean fluent, and defined in terms of a set of short-term activities (STA) expressed as instantaneous events, and contextual information detected on video content. *walking*, *running*, *active* and *inactive* are mutually exclusive STA detected on video frames. Each such STA is accompanied by the coordinates and orientation of the tracked entity in question. These are the input of the activity recognition system. $\mathit{close}(P_1,P_2)$ is true when the distance between the tracked entities P_1 and P_2 is smaller than some pre-defined threshold of pixel positions. Similarly, $\mathit{similarOrientation}(P_1,P_2)$ is true when the difference in orientation of P_1 and P_2 is less than 45 degrees.

According to rule (4), $moving(P_1, P_2) = \mathsf{true}$ is said to be initiated when both P_1 and P_2 are walking, they are close to each other and have a similar orientation. Furthermore, $moving(P_1, P_2) = \mathsf{true}$ is said to be terminated when the two tracked persons walk away from each other (see rule (5)), stop walking (see rule (6)), or one of them starts running ((7)). The remaining terminating conditions are defined in a similar manner [60].

Note that $\mathsf{initiatedAt}(F = V, T)$ 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 [7]. 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 $10 < T \leq 25$.

2.2 Point-based Probabilistic Event Calculus

Prob-EC is a probabilistic version of the Event Calculus implemented in ProbLog [33, 23]. The aim of Prob-EC is to compute the probabilities of $\mathsf{holdsAt}(F = V, T)$, i.e. the truth value of $F = V$ at time-point T. ProbLog allows for probabilistic facts: $p :: f$ indicating that fact f, which may not be ground, has probability p in all its groundings. All these facts represent independent random variables. Thus, a rule that is defined as a conjunction of k probabilistic facts, has a probability equal to the product of the probabilities of these facts. Moreover, for predicates that appear in the head of more than one rule, the probability is equal to the probability of the disjunction of the rules. ProbLog computes the probability that a query q holds in a ProbLog program — the 'success probability' — as follows:

$$P_s(q) = P\Big(\bigvee_{e \in Proofs(q)} \bigwedge_{f_i \in e} f_i\Big) \tag{8}$$

In other words, the task of computing the success probability of a query q is transformed into the task of computing the probability of the Disjunctive Normal Form (DNF) formula of eq. (8). ProbLog uses Binary Decision Diagrams (BDDs) [12] to compactly represent the DNF of eq. (8). This way, ProbLog's inference is able to scale to queries containing thousands of different proofs.

The bodies of Prob-EC rules consist of probabilistic facts. Consequently, the head of a rule holds with a certain probability, calculated as described above. This way, Prob-EC deals with uncertainty in the input data. The probability of $\mathsf{holdsAt}(LTA = \mathsf{true}, T)$ is equal to the probability of the disjunction of the initiation conditions of $LTA = \mathsf{true}$ before T, assuming that $LTA = \mathsf{true}$ has not been 'broken' in the meantime. Hence, multiple initiations of $LTA = \mathsf{true}$ increase its probability. Moreover, if $LTA = \mathsf{true}$ is 'broken' with probability p_1, then the probability of $LTA = \mathsf{true}$ becomes equal to the product of the probability of the disjunction of initiations and $1 - p_1$ (see axiom (1)). Therefore, the highest the probability p_1 the more significant the decrease of the probability of $LTA = \mathsf{true}$. Furthermore, consecutive terminations decrease further the probability of $LTA = \mathsf{true}$.

The following example illustrates probabilistic activity recognition with the use of Prob-EC. Suppose that two people, Mike and Sarah, are moving together for a number of video frames as shown in Figure 1. An initiation rule for *moving* is fired at video frame *1* when both start walking. At frame *2*, Sarah stops walking and exhibits active body movement, while Mike continues walking, but does not move

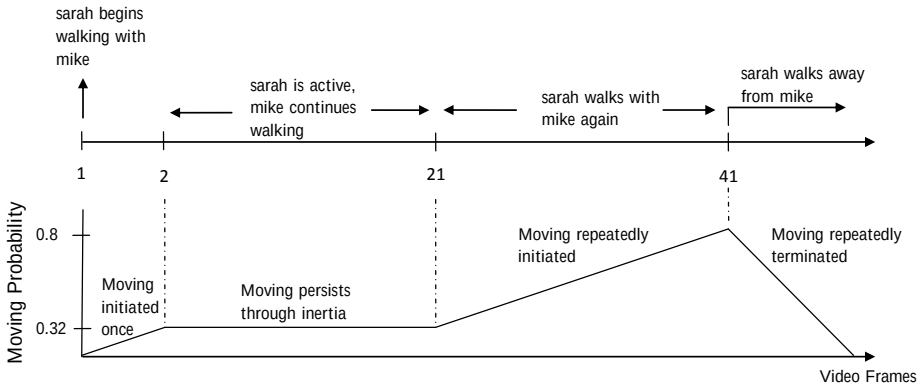

Fig. 1: Probabilistic activity recognition with Prob-EC (after [60]).

far enough to trigger a termination rule of *moving*. At frame *21*, Sarah resumes walking, once again initiating *moving*. They continue moving together until frame *40* firing multiple initiations. At frame *41*, Mike is inactive and Sarah continues walking. Their distance passes the pre-defined distance threshold for *moving* and thus fires multiple terminations.

To simplify the illustration, we assume that the probabilities of the orientation and coordinates predicates are equal to 1. Furthermore, the STA probabilities are shown below:

$$
\begin{aligned}
&0.70 :: \mathsf{happensAt}(walking(mike),\ 1).\\
&0.46 :: \mathsf{happensAt}(walking(sarah),\ 1).\\
&\ldots\\
&0.69 :: \mathsf{happensAt}(walking(mike),\ 21).\\
&0.58 :: \mathsf{happensAt}(walking(sarah),\ 21).\\
&\ldots\\
&0.18 :: \mathsf{happensAt}(inactive(mike),\ 41).\\
&0.32 :: \mathsf{happensAt}(walking(sarah),\ 41).
\end{aligned}
$$

At video frame *2*, the query $\mathsf{holdsAt}(moving(mike, sarah) = \mathsf{true},\ 2)$ has a probability equal to the probability of the initiation condition of frame *1*, which, according to rule (4), and given that all coordinate and orientation-related information are crisply recognised, is the product of the probabilities that both Mike and Sarah are walking, that is, $0.70 \times 0.46 = 0.322$. This is visualised at the far left of Figure 1, where the LTA's probability jumps from 0 to 0.322. From frame *2* to frame *20* no initiation or termination conditions are fired and the probability of *moving* remains unchanged. This occurs due to the law of inertia and is depicted by the horizontal line between frames *2* and *20*. At frame *21*, Sarah starts walking alongside Mike again. Consequently, at frame *22*, $\mathsf{holdsAt}(moving(mike, sarah) = \mathsf{true},\ 22)$ has two initiation conditions to consider, one fired at frame *1* and one at frame *21*. The

probability of $\mathsf{holdsAt}(moving(sarah, mike) = \mathsf{true},\ 22)$ is thus computed as follows:

$$\begin{aligned} P(\mathsf{holdsAt}_{22}) &= P(\mathsf{initiatedAt}_1) \vee P(\mathsf{initiatedAt}_{21}) \\ &= P(\mathsf{initiatedAt}_1) + P(\mathsf{initiatedAt}_{21}) - P(\mathsf{initiatedAt}_1) \times P(\mathsf{initiatedAt}_{21}) \\ &= 0.7 \times 0.46 + 0.69 \times 0.58 - 0.7 \times 0.46 \times 0.69 \times 0.58 = 0.593 \end{aligned} \quad (9)$$

To avoid clutter, we omit $moving(sarah, mike) = \mathsf{true}$ from the above equation, while the time-point is written as a subscript. Since $moving(sarah, mike) = \mathsf{true}$ is not 'broken' until time-point *22*, the probability of $\mathsf{holdsAt}(moving(sarah, mike) = \mathsf{true}, 22)$ depends solely on the probabilities of the two initiating conditions at time-points *1* and *21*.

The probability that Mike and Sarah are moving at frame *22* has increased, due to the additional initiation condition of frame *21*. This is one of the characteristics of Prob-EC: the continuous presence of initiation conditions of a particular LTA causes an increase of the probability of the LTA. Given continuous indication that an activity has possibly occurred, we are more inclined to agree that it has indeed taken place, even if the confidence of every individual indication is low. For this reason, from frame *22* up to and including frame *41*, the probability of *moving* increases, as is visible in Figure 1. In this example, at frame *41* the activity's probability has escalated to around 0.8.

At frame *42*, Prob-EC has to take into consideration the termination condition that was fired at frame *41*. This termination condition, corresponding to rule (5), is also probabilistic: it bears the probability that Sarah walked away from Mike, which, according to rule (5) and the fact that *close* is crisply detected, is equal to the probability of the *walking* STA itself, which is 0.32. Consequently, the probability that Mike and Sarah are still moving together at frame 42 is $0.8 \times (1 - 0.32) = 0.544$. Similar to the steady probability increase given continuous initiations, when faced with subsequent terminations, the probability of the LTA will steadily decrease. The slope of the descent (ascent) is defined by the probability of the termination (initiation) conditions involved. In this example, we assume that Sarah keeps walking away from Mike until the end of the video, causing the LTA's probability to approximate 0, as shown at the far right of Figure 1.

3 Probabilistic Maximal Interval Estimation

An instantaneous indication of an activity, by means of holdsAt for example, may lead to erroneous detection, which may be due to the unreliability of the sensors or due to the inaccuracy of the recognition patterns, or a variety of other external factors that introduce noise [4]. In terms of the monitoring process, such an erroneous activity recognition often causes unreasonable delays and holds back the operations. Hence, there is a need for a more robust recognition that identifies the temporal intervals within which a LTA may be taking place. Towards this, we propose a *Probabilistic Interval−based Event Calculus* (*PIEC*). Figure 2 shows a high-level description of the inference procedure. First, we use Prob-EC, as described in the previous section, to compute the probabilities of LTA at each time-point given the probabilistic 'short-term' activities (STA). The recognition is based on domain-specific rules of initiation and termination, such as rules (4)–(7). The next phase consists of the interval-based activity recognition. With respect to

a user-defined probability threshold, PIEC computes all maximal temporal intervals, within which an activity is likely to hold.

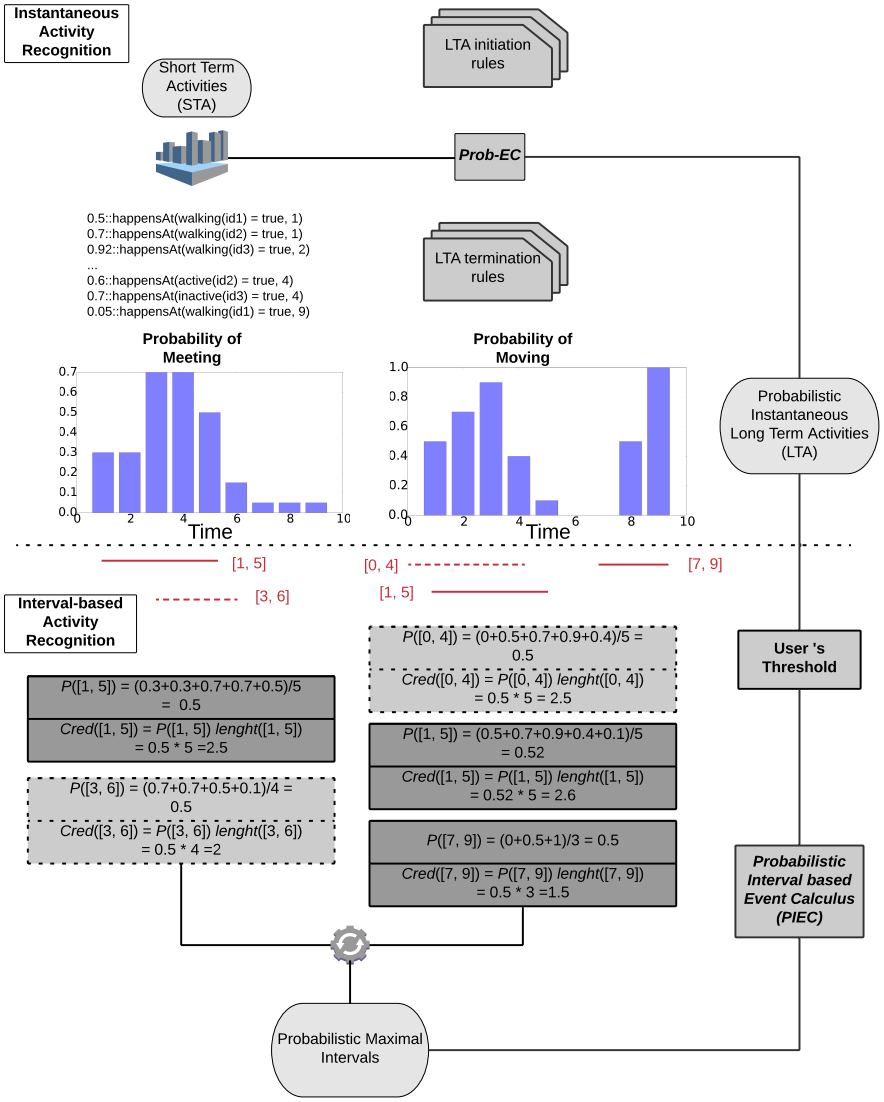

Fig. 2: Probabilistic, interval-based activity recognition. First (top figure, above the dashed line), Prob-EC computes the instantaneous probabilities of LTA, such as *meeting* and *moving*, given the probabilistic STA, such as 'walking', 'active' and 'inactive'. Second (bottom figure, below the dashed line), PIEC computes the 'probabilistic maximal intervals' of LTA, assuming a user-defined probability threshold (0.5 in this example). These intervals are depicted by the red (dashed) lines below the instantaneous probability evolution diagrams. The computation of the probability and 'credibility' of each such interval is presented in the boxes below the red lines.

Before proceeding with the presentation of our approach, we provide a set of definitions that aid understanding.

Definition 1 The probability of interval $I_{LTA}=[i,j]$ of LTA with $length(I_{LTA})=j-i+1$ time-points is defined as

$$P(I_{LTA})=\frac{\sum_{k=i}^{j} P(\mathsf{holdsAt}(LTA=\mathsf{true},\ k))}{length(I_{LTA})}\ .$$

In other words, the probability of an interval is equal to the average of the probabilities at the time-points that it contains.

A key concept of PIEC is that of probabilistic maximal interval:

Definition 2 A probabilistic maximal interval $I_{LTA}=[i,j]$ of LTA is an interval such that, given some threshold $\mathcal{T}\in[0,1]$, $P(I_{LTA})\geq\mathcal{T}$, and there is no other interval I'_{LTA} such that $P(I'_{LTA})\geq\mathcal{T}$ and I_{LTA} is a sub-interval of I'_{LTA}.

A consequence of the definition of a probabilistic maximal interval is that such intervals may be overlapping. Two examples are shown in Figure 2—see the overlapping red lines under the instantaneous probability evolution diagrams of *meeting* and *moving*. From a set of overlapping probabilistic maximal intervals, we keep only one, using interval 'credibility', defined as the product of interval length and probability:

$$Cred(I_{LTA})=length(I_{LTA})\cdot P(I_{LTA})=\sum_{k} P(\mathsf{holdsAt}(LTA=\mathsf{true},\ k))\ , \qquad (10)$$

where k are the time-points of the interval I_{LTA}. Hence, for each set of overlapping probabilistic maximal intervals $S=\{I_1,I_2,\ldots,I_n\}$, we select the one with the highest credibility value, i.e. we select I_{LTA} with $Cred(I_{LTA})=\max_i(Cred(I_i))$ for $i=1,\ldots,n$. In Figure 2 the credible intervals are depicted by the solid red lines. Equation (10) ensures that we keep an interval which is as likely and long as possible. Nevertheless, this is just one of the many ways of picking between overlapping probabilistic maximal intervals.

Figure 3 shows another example of (credible) probabilistic maximal interval estimation in PIEC. Moreover, it compares these intervals against the interval that may be derived just by the use of Prob-EC: the set of all consecutive time-points with LTA probability above a given threshold.

3.1 An Efficient Estimation Algorithm

PIEC computes probabilistic maximal intervals following [6]. First, we construct a set of lists that are subsequently used in the interval estimation procedure. Table 2 summarises these lists, while Table 3 illustrates them with the use of a simple example. The list $A[0..n]$ includes our input data, i.e. each element $A[i]$ is equal to the probability of $\mathsf{holdsAt}(LTA=\mathsf{true},i)$, computed by Prob-EC. In the example presented in Table 3, list A consists of the probabilities at 10 time-points. List L contains each element of A subtracted by the given (user-specified) probability threshold $\mathcal{T}$. The *prefix* list contains the cumulative or prefix sums of list L. PIEC

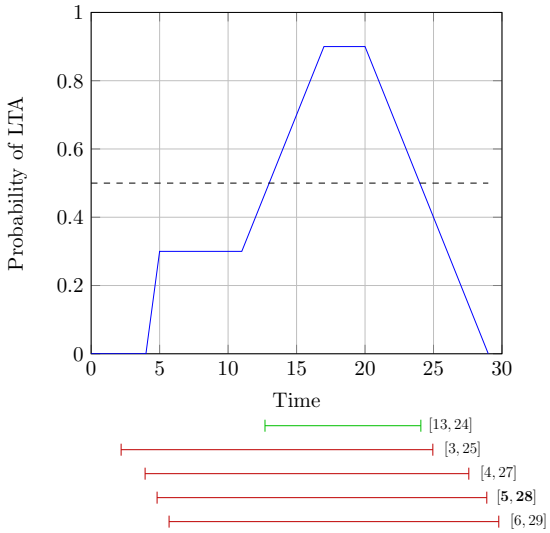

Fig. 3: Prob-EC and PIEC intervals with a 0.5 threshold. The blue line represents the instantaneous probability evolution computed by Prob-EC. The red lines represent the probabilistic maximal intervals computed by PIEC—the credible one is [5,28], denoted in bold. The green line represents the interval that may be derived by Prob-EC.

traverses the *prefix* list in reverse order, calculating, for each element i, the maximum prefix sum from i to n, and stores it in the dp list. In other words, $dp[0]$ holds the maximum prefix sum from 0 to n, $dp[1]$ holds the maximum prefix sum from 1 to n, etc, meaning that dp list is recorded in decreasing order. See Table 2 for the list specification and Table 3 for an illustration.

An interval $I_{LTA} = [i, j]$ satisfies the condition of a probabilistic maximal interval, i.e. $P(I_{LTA}) \geq \mathcal{T}$, if and only if the sum of the corresponding elements of list L is non-negative:

$$P(I_{LTA}) \geq \mathcal{T} \Leftrightarrow P([i,j]) \geq \mathcal{T} \Leftrightarrow \frac{\sum_{k=i}^{j} A[k]}{j-i+1} \geq \mathcal{T} \Leftrightarrow \sum_{k=i}^{j} A[k] \geq \mathcal{T}(j-i+1) \Leftrightarrow$$
$$\sum_{k=i}^{j} A[k] - \mathcal{T}(j-i+1) \geq 0 \Leftrightarrow (A[i]-\mathcal{T}) + \ldots + (A[j]-\mathcal{T}) \geq 0 \Leftrightarrow \sum_{k=i}^{j} L[k] \geq 0 \quad (11)$$

Hence, probabilistic maximal interval estimation may be based on L. For instance, in the example shown in Table 3, the interval $[0, 4]$ has probability $P([0,4]) = 0.5 \geq \mathcal{T}$, while $\sum_{k=0}^{4} L[k] = 0 \geq 0$.

For efficient interval estimation, however, PIEC relies solely on the following construct:

$$dprange[i,j] = \begin{cases} dp[j] - prefix[i-1], & \text{if } i > 0 \\ dp[j], & \text{if } i = 0 \end{cases} \quad (12)$$

$dprange[i, j]$ expresses the maximum sum that may be computed by adding all elements of L starting from i and ending in some $j^* \geq j$, i.e. $\max_{j^*}(L[i] + \ldots + L[j^*])$. For instance, in the example shown in Table 3, $dprange[0, 1] = dp[1] = 0.1$, while

Notation	Meaning
$\mathcal{T}$	Probability threshold
$A[0..n]$	The list of input instantaneous LTA probabilities
$L[i]$	$= A[i] - \mathcal{T}$, i.e. each element of A is subtracted by $\mathcal{T}$
$prefix[i]$	$= \sum_{j=0}^{i} L[j]$, i.e. the cumulative sum over L
$dp[i]$	$= \max_{j}(prefix[j]),\ j \in [i, n]$, i.e. the maximum prefix sum that can be reached from element i to n

Table 2: The lists of PIEC.

Time	0	1	2	3	4	5	6	7	8	9
A	0	0.5	0.7	0.9	0.4	0.1	0	0	0.5	1
L	-0.5	0	0.2	0.4	-0.1	-0.4	-0.5	-0.5	0	0.5
prefix	-0.5	-0.5	-0.3	0.1	0	-0.4	-0.9	-1.4	-1.4	-0.9
dp	0.1	0.1	0.1	0.1	0	-0.4	-0.9	-0.9	-0.9	-0.9

Table 3: PIEC in action. In this example, the threshold $\mathcal{T}$ is 0.5. The input instantaneous probabilities, computed by Prob-EC, are the elements of list A.

the maximum sum that may be computed by adding all elements of L starting from 0 and ending in some $j^* \geq 1$ is $L[0] + L[1] + L[2] + L[3] = 0.1$.

PIEC goes through the data sequentially, using two pointers, *start* and *end*, to indicate the starting and ending points of a probabilistic maximal interval. Initially, $start = end = 0$. Briefly, if $dprange[start, end] \geq 0$ then the interval $[start, end]$ is flagged, since there is an $end^* \geq end$ such that $\sum_{k=start}^{end^*} L[k] \geq 0$. In other words, $[start, end]$ is a probabilistic maximal interval or *start* is the beginning of such an interval ending after *end*. When $dprange[start, end]$ becomes negative, we know that there is no element $end^* \geq end$ that will make $dprange[start, end^*] \geq 0$. See eq. (12) for the definition of *dprange* and recall that *dp* is in decreasing order. Therefore, when we compute a negative value for $dprange[start, end]$, we check whether the interval examined in the previous step, $[start, end-1]$, was flagged, and if that is the case, we classify $[start, end-1]$ as a probabilistic maximal interval.

The pseudo-code of the interval estimation process is presented in Algorithm 1. Table 4 illustrates this process using the *moving* LTA and considering the *10* time-points displayed in Table 3. In the first step of this example, $start = end = 0$ and $dprange[0, 0] \geq 0$, thus PIEC flags the interval $[0, 0]$ by setting the value of a Boolean *flag* to true. See step 1 in Table 4 and line 14 in Algorithm 1. Subsequently, PIEC increments the *end* pointer until it reaches time-point *5*, when $dprange[start, end]$, i.e. $dprange[0, 5]$ becomes negative (see step 6 in Table 4). Since the previously computed interval, $[0, 4]$, was flagged, then $[0, 4]$ is considered a probabilistic maximal interval (see line 18 in Algorithm 1).

Next, PIEC sets the *flag* to false, increments the starting pointer, i.e. *start* becomes *1*, and searches for a probabilistic maximal interval starting from *1*. For interval $[1, 5]$, $dprange[start, end]$ is positive and thus the *flag* is set to true. Then, PIEC increments the ending pointer, *end*, in order to look for a longer interval (see line 15 of Algorithm 1). In step 8, PIEC computes again a negative

Algorithm 1 Probabilistic Maximal Interval Estimation

Input: List A **with instantaneous LTA probabilities and threshold** $\mathcal{T}$.
Output: List *output* **of credible probabilistic maximal intervals.**

```
1: function PIEC(List A[0..n], threshold T)
2:     L[i] ← A[i]−T for each 0 ≤ i ≤ n
3:     prefix ← computePrefix(L)
4:     dp ← computeDp(prefix)
5:     start ← 0, end ← 0, output ← ∅, dprange[start, end] ← 0
6:     while (start ≤ n and end ≤ n) do
7:         if start > 0 then
8:             dprange[start, end] ← dp[end]−prefix[start−1]
9:         else
10:            dprange[start, end] ← dp[end]
11:        if dprange[start, end] ≥ 0 then
12:            if (end = n and start < end) or (end = start = n and A[start] ≥ T) then
13:                add(start, end) to output
14:            flag ← true
15:            end++
16:        else
17:            if (start < end and flag = true) then
18:                add(start, end−1) to output
19:            if (start = end and A[start] ≥ T) then
20:                add(start, end) to output
21:            flag ← false
22:            start++
23:    return getCredible(output)
```

dprange$[start, end]$, confirms that *flag* = true, and thus outputs $[1, 5]$ as a maximal interval. Subsequently, PIEC sets the *flag* to false to indicate that $[2, 5]$ cannot be a probabilistic maximal interval, and sets $start = 2$. The value of *dprange*$[start, end]$ remains negative until step 13, where *start* is continuously incremented, while *end* remains the same.

In steps 14–17, *dprange*$[start, end]$ is non-negative and thus PIEC increments *end* until it reaches the bounds of the input list—recall that the last time-point in this example is 9. At step 17, PIEC outputs $[7, 9]$ as a probabilistic maximal interval (see line 13 of Algorithm 1). This way, PIEC does not discard a probabilistic maximal interval when the ending pointer reaches the bounds of the input list (even if *dprange*$[start, end] < 0$ does not hold).

For each set of overlapping maximal intervals, PIEC keeps the one that maximises the credibility value (see eq. (10) and line 23 in Algorithm 1). In this example, the overlapping intervals are $[0, 4]$ and $[1, 5]$. PIEC discards $[0, 4]$ since $Cred([0, 4]) = \sum_{k=0}^{4} A[k] = 2.5$ while $Cred([1, 5]) = \sum_{k=1}^{5} A[k] = 2.6$.

3.2 Complexity

Assuming a temporally sorted dataset, the computation of lists L, *prefix* and *dp* takes linear time $\Theta(n)$ where n is the number of the input time-points. Concerning the computation of probabilistic maximal intervals, we observe that, in the worst case, both pointers *start* and *end* traverse the input list of size n. Therefore, this step requires $O(2n)$. The selection of the most credible intervals requires $\Theta(n)$ time

since maximal intervals are sorted by their starting point. The use of memory is also linear with respect to the size n of the input and auxiliary lists.

Step	Pointers	Calculations
1	$start = 0$ $end = 0$	$dprange[0,0] \leftarrow dp[0] = 0.1 \geq 0$ $flag \leftarrow$ true
2	$start = 0$ $end = 1$	$dprange[0,1] \leftarrow dp[1] = 0.1 \geq 0$ $flag \leftarrow$ true
3	$start = 0$ $end = 2$	$dprange[0,2] \leftarrow dp[2] = 0.1 \geq 0$ $flag \leftarrow$ true
4	$start = 0$ $end = 3$	$dprange[0,3] \leftarrow dp[3] = 0.1 \geq 0$ $flag \leftarrow$ true
5	$start = 0$ $end = 4$	$dprange[0,4] \leftarrow dp[1] = 0.1 \geq 0$ $flag \leftarrow$ true
6	$start = 0$ $end = 5$	$dprange[0,5] \leftarrow dp[5] = -0.4 < 0$ **add interval** $[0,4]$ **to** *output* $flag \leftarrow$ false
7	$start = 1$ $end = 5$	$dprange[1,5] \leftarrow dp[5] - prefix[0] = 0.1 \geq 0$ $flag \leftarrow$ true
8	$start = 1$ $end = 6$	$dprange[1,6] \leftarrow dp[6] - prefix[0] = -0.4 < 0$ **add interval** $[1,5]$ **to** *output* $flag \leftarrow$ false
9	$start = 2$ $end = 6$	$dprange[2,6] \leftarrow dp[6] - prefix[1] = -0.4 < 0$ $flag \leftarrow$ false
10	$start = 3$ $end = 6$	$dprange[3,6] \leftarrow dp[6] - prefix[2] = -0.6 < 0$ $flag \leftarrow$ false
11	$start = 4$ $end = 6$	$dprange[4,6] \leftarrow dp[6] - prefix[3] = -1 < 0$ $flag \leftarrow$ false
12	$start = 5$ $end = 6$	$dprange[5,6] \leftarrow dp[6] - prefix[4] = -0.9 < 0$ $flag \leftarrow$ false
13	$start = 6$ $end = 6$	$dprange[6,6] \leftarrow dp[6] - prefix[5] = -0.5 < 0$ $flag \leftarrow$ false
14	$start = 7$ $end = 6$	$dprange[7,6] \leftarrow dp[6] - prefix[6] = 0 \geq 0$ $flag \leftarrow$ true
15	$start = 7$ $end = 7$	$dprange[7,7] \leftarrow dp[7] - prefix[6] = 0 \geq 0$ $flag \leftarrow$ true
16	$start = 7$ $end = 8$	$dprange[7,8] \leftarrow dp[8] - prefix[6] = 0 \geq 0$ $flag \leftarrow$ true
17	$start = 7$ $end = 9$	$dprange[7,9] \leftarrow dp[9] - prefix[6] = 0 \geq 0$ **add interval** $[7,9]$ **to** *output*

Table 4: Interval Estimation Example.

4 Experimental Evaluation

4.1 Experimental Setup

We evaluated PIEC using three variations of CAVIAR[1], a benchmark activity recognition dataset. This dataset includes 28 surveillance videos of a public space. The videos are staged; actors walk around, sit down, meet one another, leave objects behind, fight, and so on. Each video has been manually annotated by the CAVIAR team in order to provide the ground truth for both STA and LTA. The

[1] `https://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/`

input to the recognition system consists of the STA 'walking', 'running', 'active' (non-abrupt body movement in the same position) and 'inactive' (standing still), together with their time-stamps, i.e. the video frame in which the STA took place. Furthermore, the dataset includes the coordinates of the tracked people and objects as pixel positions at each time-point, as well as their orientation. Given such input, the task is to recognise LTA such as two people moving together, having a meeting or fighting.

The variations of CAVIAR include three levels of noise and were generated in order to compare the accuracy of Prob-EC and a deterministic Event Calculus [60]. The three versions of the dataset are as follows:

- In the first version of the dataset—*smooth noise*—a subset of the STA have probabilities attached, generated by a Gamma distribution with a varying mean. All other STA have no probabilities attached, as in the original dataset.
- In the second version—*intermediate noise*—probabilities have been attached also to coordinate and orientation predicates using the same Gamma distribution.
- In the last version—*strong noise*—spurious STA that do not belong to the original dataset were added using a uniform distribution.

These versions of CAVIAR, along with the LTA definitions in the Event Calculus, are publicly available[2]. We feed these data to Prob-EC in order to perform point-based activity recognition, i.e. calculate a series of positives of the form $Prob :: \mathsf{holdsAt}(LTA = \mathsf{true},\ T)$. Subsequently, we use PIEC to compute the credible probabilistic maximal intervals for each LTA. In the analysis that follows, we compare the predictive accuracy of PIEC against that of Prob-EC, by filtering the output of Prob-EC to keep only the positives with probability *Prob* above a chosen threshold, indicating that we trust these positives to be accurate.

4.2 Experimental Results

Figure 4 presents common cases of the experimental results. The top-left diagram of Figure 4 shows a series of initiations, leading to a continuous LTA probability increase, followed by a series of terminations, resulting in a continuous probability decrease. In the middle-right and the bottom-right diagrams, there is a single termination, that causes a dramatic drop in probability. In the middle-left and the right diagrams, the LTA is subject to the law of inertia between the initiations and the termination(s)—this is denoted by a horizontal line in the probability evolution plot.

By definition, given a threshold value $\mathcal{T}$, the intervals computed by PIEC are super-intervals of the intervals that may be derived from the instantaneous probabilities calculated by Prob-EC (recall Definition 2). In most cases, when the probability increase or the probability decrease is not abrupt, that is, in the cases of continuous 'soft' indications that an activity has started (respectively ended), the additional time-points of the PIEC intervals (if any) have relatively high-probability. See, for instance, the top and the middle-right diagrams of Figure 4. However, when both the probability increase and decrease are abrupt, that is, when

[2] `https://anskarl.github.io/publications/TPLP15/`

there is instantaneous strong evidence about the initiation and the termination of an LTA, the intervals of PIEC may include time-points with low, even 0 probability (see e.g. the middle-left diagram of Figure 4). In these cases, including some time-points with low probability in a PIEC interval may not drop the interval probability below the given threshold value.

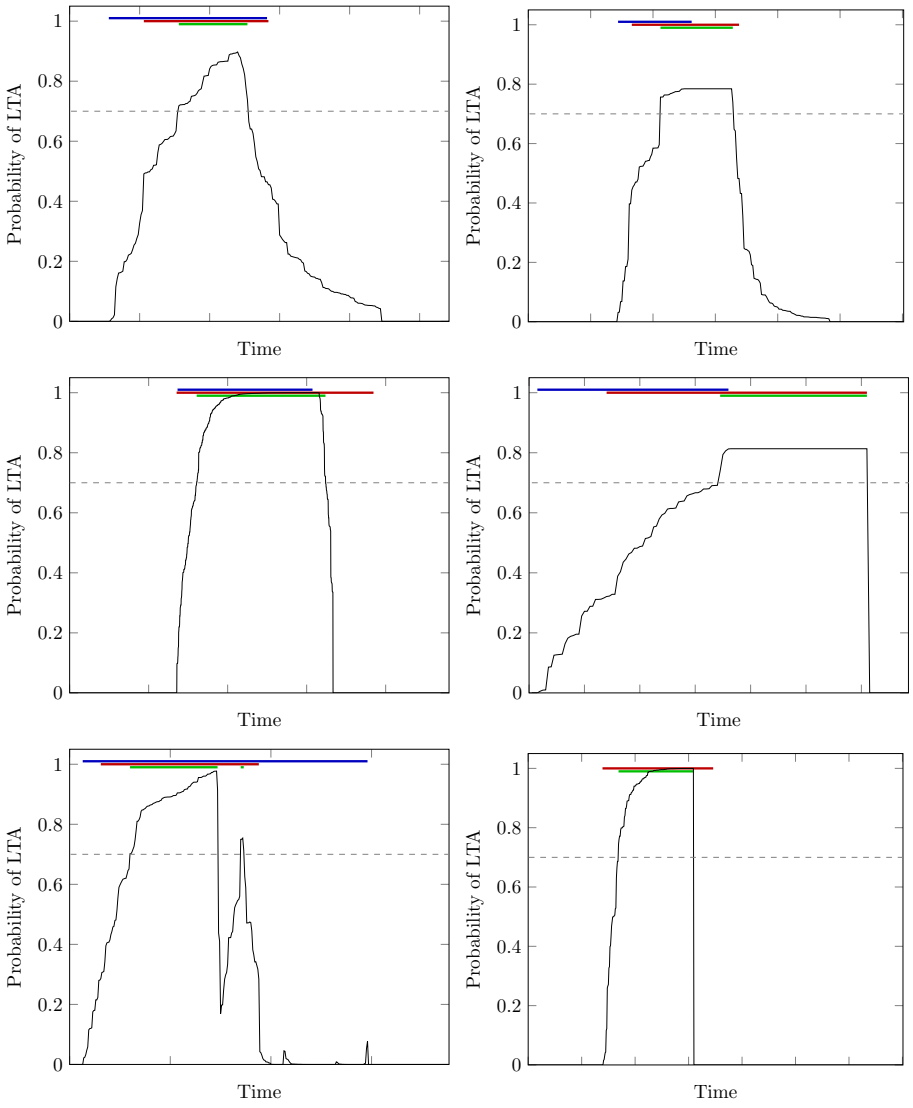

Fig. 4: Probabilistic activity recognition in CAVIAR. The black line represents the instantaneous probability evolution computed by Prob-EC. The horizontal lines denote the maximal intervals that may be derived by Prob-EC using a 0.7 threshold (green), the credible probabilistic maximal intervals computed by PIEC using the same threshold value (red), and the ground truth (blue).

In the common cases of non-abrupt probability increase and decrease, the intervals of Prob-EC may approximate those of PIEC by lowering the threshold value. In the top and middle-right diagrams, the intervals of Prob-EC would be the same, more or less, as the intervals of PIEC by setting the threshold value for Prob-EC to 0.5. However, lowering the threshold value may create false positives, as in the cases when the probability of an LTA raises above the threshold only momentarily, due to a few noisy observations.

Figure 4 shows that PIEC and Prob-EC do not always agree with the ground truth given by the CAVIAR team. Typically, the manually annotated intervals start at the first initiation of the LTA (although there are cases, not shown here to save space, where they start earlier). There is no fixed relation between the annotation and the terminations, however. In some cases, the annotated intervals end after a series of terminations (see the top-left diagram of Figure 4), while in other cases the end of an annotated interval coincides with the first termination (see the middle-left diagram of Figure 4). In the middle-right diagram of Figure 4, the annotated interval ends with the last initiation, while in the top right-diagram, it ends while the LTA is subject to inertia, that is, when there are no initiations and terminations.

There are also some extreme cases, such as, for example, that displayed in the bottom-right diagram, where there are numerous time-points with high-probability, including 1, but there are no positive examples at all according to the ground truth. In such cases, PIEC is penalised more than Prob-EC, since it typically produces longer intervals for a given threshold value. These issues arise because there is noise even in the original CAVIAR dataset [60, 37], and hence the (manually constructed) LTA definitions cannot match perfectly the ground truth. Recall that PIEC builds upon the instantaneous probabilities computed by Prob-EC, which in turn are based on the given LTA definitions.

The bottom-left diagram of Figure 4 shows a case of probability fluctuation—a steady increase in probability is followed by a noisy observation that reduces dramatically the LTA probability. Subsequently, the probability increases again. PIEC is able to compute a single maximal interval, mitigating the effects of the noisy observation reducing temporarily the LTA probability. In contrast, Prob-EC is directly affected by the noisy observation, creating a series of false negatives between its two maximal intervals. To approximate the interval of PIEC, we would have to lower significantly the threshold value for Prob-EC, creating numerous false positives in other cases. This example, therefore, illustrates one of the key benefits of using PIEC for activity recognition.

Figure 5 displays the F1-score for the *fighting*, *moving* and *meeting* LTA. We repeated the experiments 5×16 times, i.e. 5 times for each Gamma distribution mean value in the range 0.5 to 8.0, with a step of 0.5. The higher the mean, the lower the probabilities attached to the CAVIAR input facts, indicating a higher amount of noise. The results for 'strong noise', i.e. adding spurious STA, are omitted since they are very similar to the those of 'intermediate noise'. The insertion of spurious facts affects very rarely the performance of PIEC and Prob-EC, since the initiation and termination rules require the satisfaction of a combination of predicates that appear in their body.

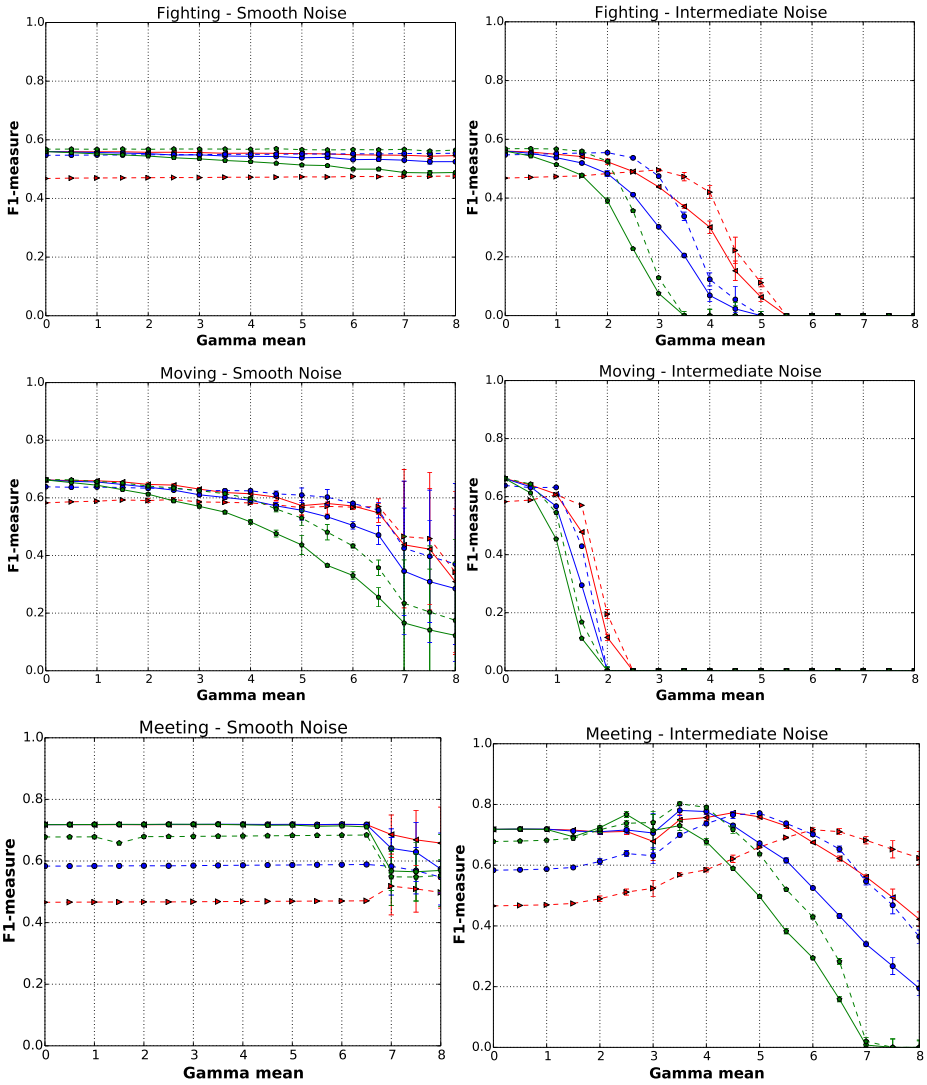

Fig. 5: The F1-score of PIEC and Prob-EC per Gamma mean value for the *fighting*, *moving* and *meeting* LTA, under smooth noise (left) and intermediate noise (right). The dashed lines display the performance of PIEC while the solid lines display the performance of Prob-EC. Red lines correspond to a 0.5 threshold value, blue lines to 0.7, while green lines correspond to a 0.9 threshold value.

As shown in Figure 5, the performance of PIEC is mostly similar to that Prob-EC. Cases such as that depicted in the bottom-right diagram of Figure 4 do not allow PIEC to outperform (significantly) Prob-EC. In the experiments with the 'smooth noise' dataset, the F1-score of PIEC is very close with that of Prob-EC, with PIEC performing slightly better in *fighting*, and *moving* for high Gamma mean values (that is, high noise levels). In the 'intermediate noise' dataset, neither PIEC nor Prob-EC can cope with the noise of high Gamma mean values in *fighting*

and *moving*. PIEC remains the best option in *fighting*, while the difference in the F1-score is bigger in many cases. In *moving*, PIEC performs at least as well as Prob-EC, while in *meeting* PIEC manages to outperform Prob-EC in high Gamma mean values, being more tolerant to high noise levels. Interestingly, the best choice in these noise levels, PIEC with a 0.5 threshold, has increased performance as the Gamma mean value increases. This is due to the increasing precision (see the Appendix for the precision and recall figures of all LTA).

5 Related Work

Activity recognition is a type of composite (often called 'complex') event recognition [8, 19]. Systems for composite event recognition accept as input a stream of time-stamped, 'simple, derived events', such as such as events coming from sensors, and identify composite events of interest—collections of events that satisfy some pattern. The definition of a composite event imposes temporal and, possibly, atemporal constraints on its sub-events (simple, derived events or other composite events).

The event streams that provide the input data to a recognition system exhibit various types of uncertainty [4]. One such type is that of incomplete or missing evidence [13]. A sensor, such as a camera, may fail to report some events due to a hardware malfunction. Even if the hardware works as expected, certain characteristics of the monitored environment could prevent events from being reported—consider, for instance, an occluded object in activity recognition, or a voice being drowned by stronger acoustic signals. Additionally, the events of the input stream typically have a noise component added to them. Consequently, events are often accompanied by a probability value. Several factors contribute to the corruption of the input stream, such as the limited accuracy of sensors and distortion along a communication channel. When noise corrupts the input event stream, a recognition system may receive events asserting contradictory statements. In an activity recognition application which needs to track objects, if there are multiple software classifiers, one of them may assert the presence of an object whereas another may indicate that no such object has been detected.

A recent survey identified the following classes of methods for handling uncertainty in composite event recognition: automata-based methods, probabilistic graphical models, probabilistic/stochastic Petri Nets and approaches based on stochastic (context-free) grammars [4]. Automata-based methods, such as [68, 1, 57, 69], focus on detecting sequences of events, in which some of those events may be related, via their attributes, to other events of the sequence. In contrast to the Event Calculus, time representation is implicit. Moreover, hierarchical knowledge, such as defining an LTA in terms of some other LTA, is not supported ([67] is an exception). The approaches that use Petri Nets lack a mechanism for modelling a domain in a truly relational manner, i.e. by allowing relations to be defined between attributes of events. Similarly, in approaches based on stochastic grammars, events are not relational.

The closest line of work to our approach, therefore, concerns the use of probabilistic graphical models, such as Markov Networks. When used for activity recognition, Markov Networks are combined with first-order logic, in which case they are called Markov Logic Networks (MLN) [53]. MLN are undirected probabilistic

graphical models which encode a set of weighted first-order logic formulas. The combination of a general but formal representation language, together with a well-defined probability space, constitutes MLN suitable for activity recognition. In [44], for example, Allen's Interval Algebra [5] is employed and the most consistent sequence of LTA is determined, based on the observations of low-level classifiers. In order to avoid the combinatorial explosion of all possible intervals, and eliminate the existential quantifiers in LTA definitions, a bottom-up process discards the unlikely event hypotheses. This elimination process can only be applied to domain-dependent axioms, as it is guided by the observations. Sadilek and Kautz [54] employ hybrid-MLN [66] in order to distinguish between successful and failed interactions between humans, using location data from GPS devices. 'Hybrid formulas' are used to de-noise the location data. These are formulas with weights associated with a real-valued function, such as the distance of two persons.

The work of Skarlatidis et al. [61] is one of the first attempts to provide a *general* probabilistic framework for activity recognition via MLN. In contrast, most MLN-based approaches to activity recognition are domain-dependent. In order to establish such a framework, Skarlatidis and colleagues employed the Event Calculus. They aimed to tackle LTA definition uncertainty, i.e. model imperfect rules defining LTA. Instead, we built upon a probabilistic Event Calculus handling data uncertainty. Although probabilistic STA can be incorporated into graphical models, correctly encoding their dependencies can be far from obvious, especially with MLN [4].

There are also logic-based approaches to activity recognition that do not (directly) employ graphical models. The Probabilistic Event Logic [11, 56], for instance, has been used to define a log-linear model from a set of weighted formulas, expressing LTA and represented in Event Logic, a formalism for durative events [59]. Each formula defines a 'soft' constraint over some events, in a manner similar to MLN. However, instead of building a ground network with all the time variables, reasoning is performed using 'spanning intervals' that allow for a compact representation of event occurrences satisfying a formula. Albanese et al. [3] proposed an activity description language merging logic-based formulation with probabilistic modelling. LTA are defined in a first-order logic. The input STA can be either deterministic or probabilistic, while their dependencies are modelled by triangular norms [25]. Shet and colleagues [58] proposed a method employing logic programming and handling uncertainty using the Bilattice framework [28]. Each LTA or STA is associated with two uncertainty values, indicating a degree of information and confidence respectively. The more confident information is provided, the stronger the belief about the corresponding LTA becomes.

Moldovan et al. proposed a ProbLog-based method for robotic action recognition [42]. The method employs a relational extension of the *affordance models* [27] in order to represent multi-object interactions. Affordances can model the relations between objects, pre-programmed robotic arm movements and their effects. In contrast to a standard propositional Bayesian Network implementation of an affordance model, the method can scale to multiple object interactions in a scene without the need of retraining.

A key difference between our work and these methods lies in the use of the Event Calculus, which allows us to develop an expressive activity recognition framework, specifying succinctly complex LTA by taking advantage of the built-in representation of inertia. With the use of the Event Calculus, one may develop in-

tuitive activity definitions, facilitating the interaction between activity definition developer and domain expert, and allowing for code maintenance.

The Event Calculus is related to other formalisms for commonsense reasoning, such as the Situation Calculus [40, 52], the Fluent Calculus [62, 63], the action language $\mathcal{A}$ [26], the action language $\mathcal{C}+$ [29, 2] and Temporal Action Logics [21, 36]. Comparisons and proofs of equivalence may be found in [34, 64, 17, 41, 55, 46, 45, 18, 49]. Probabilistic extensions of the Situation Calculus support noisy input as well as stochastic events, and model Markov Decision Processes—see, for example, [9, 51, 39, 30, 52]. Furthermore, a probabilistic extension of the Fluent Calculus has been proposed [31]. In contrast to the (probabilistic variants of) the Situation Calculus and the Fluent Calculus, we use a single time-line on which events occur. This is a more suitable model for activity recognition, where the task is to recognise LTA in a time-stamped sequence of STA. A probabilistic extension of the action language $\mathcal{A}$ is proposed by Baral et al. [10], which aims at an elaboration-tolerant representation of Markov Decision Processes, and at formulating observation assimilation and counter-factual reasoning. $\mathcal{PC}+$ is an extension of $\mathcal{C}+$ supporting, among others, non-deterministic and probabilistic effects of events [24]. Similar to most of the aforementioned probabilistic variants of commonsense reasoning languages and [32], $\mathcal{PC}+$ focuses on planning under uncertainty.

Various Event Calculus dialects have been proposed in the literature for narrative assimilation/temporal projection, the task supporting activity recognition. Several of these dialects allow for durative LTA by means of durative events or by explicitly representing fluent intervals. Some examples are [16, 14, 47, 48, 15, 43, 7]. However, these dialects cannot handle the inherent uncertainty of activity recognition. A recently proposed probabilistic Event Calculus is presented in [20]. Our work is complementary to the Event Calculus of [20], as well as that of Skarlatidis et al. [61]. We proposed a method, PIEC, for efficient computation of probabilistic maximal intervals, that may operate on top of any dialect for point-based probability calculation.

PIEC computes probabilistic maximal intervals following [6], a method for calculating the longest interval having a non-negative sum in a sequence of real numbers. The work presented in [6] has been applied to biological sequences. We translate the problem of probabilistic maximal interval estimation to the problem of maximal non-negative sum interval estimation (see Section 3.1). Moreover, we extend the work presented in [6] in order to compute *all* maximal non-negative sum intervals, as opposed to the longest one.

A very short version of this paper, giving a high-level view of the proposed approach, has been submitted to the Probabilistic Logic Programming workshop [38].

6 Summary and Further Work

We proposed an efficient algorithm computing maximal intervals for activity recognition under uncertainty. We demonstrated the use of the algorithm by employing Prob-EC, an Event Calculus dialect handling data uncertainty. We evaluated our approach on a benchmark activity recognition dataset, and outlined the conditions in which it outperforms time-point-based recognition. Our approach is robust to

noisy LTA probability fluctuations, and performs better in the common case of non-abrupt probability change.

There are several directions for further work. We aim to evaluate the proposed algorithm in different datasets. We also aim to incorporate probabilistic dialects of the Event Calculus handling LTA definition uncertainty. Furthermore, we plan to extend our approach by handling streaming data, possibly by means of a window-based approach.

Acknowledgements This work is funded by the EU H2020 datAcron project (687591). We would also like to thank Elias Alevizos for his feedback at the early stages of this work, and Evangelos Michelioudakis for his support in the experiments.

References

1. Agrawal, J., Diao, Y., Gyllstrom, D., Immerman, N.: Efficient pattern matching over event streams. In: SIGMOD, pp. 147–160 (2008)
2. Akman, V., Selim T. Erdoğan, J.L., Lifschitz, V., Turner, H.: Representing the Zoo World and the Traffic World in the language of the Causal Calculator. Artificial Intelligence **153**(1), 105–140 (2004)
3. Albanese, M., Chellappa, R., Cuntoor, N., Moscato, V., Picariello, A., Subrahmanian, V.S., Udrea, O.: PADS: A Probabilistic Activity Detection Framework for Video Data. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(12), 2246–2261 (2010)
4. Alevizos, E., Skarlatidis, A., Artikis, A., Paliouras, G.: Probabilistic complex event recognition: A survey. ACM Comput. Surv. **50**(5), 71:1–71:31 (2017)
5. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11), 832–843 (1983)
6. Allison, L.: Longest biased interval and longest non-negative sum interval. Bioinformatics **19**(10), 1294–1295 (2003)
7. Artikis, A., Sergot, M.J., Paliouras, G.: An event calculus for event recognition. IEEE Trans. Knowl. Data Eng. **27**(4), 895–908 (2015)
8. Artikis, A., Skarlatidis, A., Portet, F., Paliouras, G.: Logic-based event recognition. Knowledge Eng. Review **27**(4), 469–506 (2012)
9. Bacchus, F., Halpern, J.Y., Levesque, H.J.: Reasoning about Noisy Sensors in the Situation Calculus. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI), pp. 1933–1940. Morgan Kaufmann (1995)
10. Baral, C., Nam, T.H., Tuan, L.: Reasoning about actions in a probabilistic setting. In: Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, pp. 507–512 (2002)
11. Brendel, W., Fern, A., Todorovic, S.: Probabilistic event logic for interval-based event recognition. In: The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011, pp. 3329–3336 (2011)
12. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Computers **35**(8), 677–691 (1986)

13. Busany, N., Gal, A., Senderovich, A., Weidlich, M.: Interval-based queries over multiple streams with missing timestamps (2017)
14. Cervesato, I., Montanari, A.: A calculus of macro-events: Progress report. In: TIME, pp. 47–58 (2000)
15. Chesani, F., Mello, P., Montali, M., Torroni, P.: A logic-based, reactive calculus of events. Fundamenta Informaticae **105**(1-2), 135–161 (2010)
16. Chittaro, L., Montanari, A.: Efficient temporal reasoning in the cached event calculus. Computational Intelligence **12**(3), 359–382 (1996)
17. Chittaro, L., Montanari, A.: Temporal Representation and Reasoning in Artificial Intelligence: Issues and Approaches. Annals of Mathematics and Artificial Intelligence **28**(1), 47–106 (2000)
18. Craven, R.: Execution mechanisms for the action language C+. Ph.D. thesis, University of London (2006)
19. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. **44**(3), 15:1–15:62 (2012)
20. D'Asaro, F.A., Bikakis, A., Dickens, L., Miller, R.: Foundations for a probabilistic event calculus. In: M. Balduccini, T. Janhunen (eds.) Logic Programming and Nonmonotonic Reasoning - 14th International Conference, LPNMR 2017, Espoo, Finland, July 3-6, 2017, Proceedings, *Lecture Notes in Computer Science*, vol. 10377, pp. 57–63. Springer (2017)
21. Doherty, P., Gustafsson, J., Karlsson, L., Kvarnström, J.: TAL: Temporal Action Logics Language Specification and Tutorial. Electronic Transactions on Artificial Intelligence **2**(3–4), 273–306 (1998)
22. Dousson, C., Maigat, P.L.: Chronicle recognition improvement using temporal focusing and hierarchization. In: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007, pp. 324–329 (2007)
23. Dries, A., Kimmig, A., Meert, W., Renkens, J., den Broeck, G.V., Vlasselaer, J., Raedt, L.D.: Problog2: Probabilistic logic programming. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part III, pp. 312–315 (2015)
24. Eiter, T., Lukasiewicz, T.: Probabilistic Reasoning about Actions in Nonmonotonic Causal Theories. In: C. Meek, U. Kjærulff (eds.) UAI, pp. 192–199. Morgan Kaufmann (2003)
25. Fagin, R.: Combining fuzzy information from multiple systems. In: In Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 216–226. ACM Press (1996)
26. Gelfond, M., Lifschitz, V.: Representing action and change by logic programs. J. Log. Program. **17**(2/3&4), 301–321 (1993)
27. Gibson, J.: The ecological approach to visual perception (1979)
28. Ginsberg, M.L.: Multivalued logics: a uniform approach to reasoning in artificial intelligence. Computational Intelligence **4**, 265–316 (1988)
29. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic Causal Theories. Artificial Intelligence **153**(1), 49–104 (2004)
30. Hajishirzi, H., Amir, E.: Sampling First Order Logical Particles. In: Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI), Helsinki, Finland, pp. 248–255. AUAI Press (2008)

31. Hölldobler, S., Karabaev, E., Skvortsova, O.: FLUCAP: a heuristic search planner for first-order MDPs. Journal of Artificial Intelligence Research (JAIR) **27**(1), 419–439 (2006)
32. Iocchi, L., Lukasiewicz, T., Nardi, D., Rosati, R.: Reasoning about actions with sensing under qualitative and probabilistic uncertainty. ACM Trans. Comput. Log. **10**(1), 5:1–5:41 (2009)
33. Kimmig, A., Demoen, B., Raedt, L.D., Costa, V.S., Rocha, R.: On the Implementation of the Probabilistic Logic Programming Language ProbLog. Theory and Practice of Logic Programming **11**, 235–262 (2011)
34. Kowalski, R., Sadri, F.: Reconciling the Event Calculus with the Situation Calculus. The Journal of Logic Programming **31**(1), 39–58 (1997)
35. Kowalski, R.A., Sergot, M.J.: A logic-based calculus of events. New Generation Comput. **4**(1), 67–95 (1986)
36. Kvarnström, J.: TALplanner and Other Extensions to Temporal Action Logic. Ph.D. thesis, Linköping (2005)
37. List, T., Bins, J., Vazquez, J., Fisher, R.B.: Performance evaluating the evaluator. In: Proceedings 2nd Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp. 129–136 (2005)
38. Makris, E., Artikis, A., Paliouras, G.: Interval-based activity recognition. In: The 5th Workshop on Probabilistic Logic Programming (2018). Under Review. `http://users.iit.demokritos.gr/~a.artikis/PIEC-PLP.pdf`
39. Mateus, P., Pacheco, A., Pinto, J., Sernadas, A., Sernadas, C.: Probabilistic Situation Calculus. Annals of Mathematics and Artificial Intelligence **32**(1), 393–431 (2001)
40. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. Stanford University (1968)
41. Miller, R., Shanahan, M.: Some Alternative Formulations of the Event Calculus. In: Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II, Lecture Notes in Computer Science, pp. 452–490. Springer (2002)
42. Moldovan, B., Moreno, P., van Otterlo, M., Santos-Victor, J., De Raedt, L.: Learning relational affordance models for robots in multi-object manipulation tasks. In: Proceedings of International Conference on Robotics and Automation (ICRA), pp. 4373–4378 (2012)
43. Montali, M., Maggi, F.M., Chesani, F., Mello, P., van der Aalst, W.M.P.: Monitoring business constraints with the Event Calculus. ACM TIST **5**(1) (2014)
44. Morariu, V.I., Davis, L.S.: Multi-agent event recognition in structured scenarios. In: The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011, pp. 3289–3296 (2011)
45. Mueller, E.T.: Commonsense Reasoning. Morgan Kaufmann (2006)
46. Mueller, E.T.: Event calculus and temporal action logics compared. Artif. Intell. **170**(11), 1017–1029 (2006)
47. Paschke, A.: ECA-RuleML: An approach combining ECA rules with temporal interval-based KR event/action logics and transactional update logics. Tech. Rep. 11, Technische Universität München (2005)
48. Paschke, A., Bichler, M.: Knowledge representation concepts for automated SLA management. Decision Support Systems **46**(1), 187–205 (2008)

49. Paschke, A., Kozlenkov, A.: Rule-Based Event Processing and Reaction Rules. In: Proceedings of the 3rd International Symposium on Rules (RuleML), *Lecture Notes in Computer Science*, vol. 5858, pp. 53–66. Springer (2009)
50. Patroumpas, K., Alevizos, E., Artikis, A., Vodas, M., Pelekis, N., Theodoridis, Y.: Online event recognition from moving vessel trajectories. GeoInformatica **21**(2), 389–427 (2017)
51. Pinto, J., Sernadas, A., Sernadas, C., Mateus, P.: Non-determinism and uncertainty in the Situation Calculus. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **8**(2), 127–149 (2000)
52. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press (2001)
53. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**(1–2), 107–136 (2006)
54. Sadilek, A., Kautz, H.A.: Location-Based Reasoning about Complex Multi-Agent Behavior. Journal of Artificial Intelligence Research (JAIR) **43**, 87–133 (2012)
55. Schiffel, S., Thielscher, M.: Reconciling Situation Calculus and Fluent Calculus. In: Proceedings of the National Conference on Artificial Intelligence, vol. 21, p. 287. AAAI Press (2006)
56. Selman, J., Amer, M.R., Fern, A., Todorovic, S.: PEL-CNF: Probabilistic event logic conjunctive normal form for video interpretation. In: ICCVW, pp. 680–687. IEEE (2011)
57. Shen, Z., Kawashima, H., Kitagawa, H.: Probabilistic event stream processing with lineage. In: Proc. of Data Engineering Workshop (2008)
58. Shet, V.D., Neumann, J., Ramesh, V., Davis, L.S.: Bilattice-based Logical Reasoning for Human Detection. In: (CVPR), pp. 1–8. IEEE Computer Society (2007)
59. Siskind, J.M.: Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. JAIR **15**, 31–90 (2001)
60. Skarlatidis, A., Artikis, A., Filipou, J., Paliouras, G.: A probabilistic logic programming event calculus. TPLP **15**(2), 213–245 (2015)
61. Skarlatidis, A., Paliouras, G., Artikis, A., Vouros, G.A.: Probabilistic event calculus for event recognition. ACM Trans. Comput. Logic **16**(2) (2015)
62. Thielscher, M.: From Situation Calculus to Fluent Calculus: State update axioms as a solution to the inferential frame problem. Artificial intelligence **111**(1), 277–299 (1999)
63. Thielscher, M.: The qualification problem: A solution to the problem of anomalous models. Artificial Intelligence **131**(1), 1–37 (2001)
64. Van Belleghem, K., Denecker, M., De Schreye, D.: On the relation between Situation Calculus and Event Calculus. The Journal of Logic Programming **31**(1), 3–37 (1997)
65. Vouros, G.A., Vlachou, A., Santipantakis, G.M., Doulkeridis, C., Pelekis, N., Georgiou, H.V., Theodoridis, Y., Patroumpas, K., Alevizos, E., Artikis, A., Claramunt, C., Ray, C., Scarlatti, D., Fuchs, G., Andrienko, G.L., Andrienko, N.V., Mock, M., Camossi, E., Jousselme, A., Garcia, J.M.C.: Big data analytics for time critical mobility forecasting: Recent progress and research challenges. In: Proceedings of the 21th International Conference on Extending Database Technology, EDBT, pp. 612–623 (2018)

66. Wang, J., Domingos, P.: Hybrid Markov Logic Networks. In: Proceedings of the 23rd AAAI Conference on Artificial Intelligence, pp. 1106–1111. AAAI Press (2008)
67. Wang, Y.H., Cao, K., Zhang, X.M.: Complex event processing over distributed probabilistic event streams. Computers & Mathematics with Applications **66**(10), 1808–1821 (2013)
68. Wu, E., Diao, Y., Rizvi, S.: High-performance Complex Event Processing over Streams. In: ACM SIGMOD, pp. 407–418 (2006)
69. Zhang, H., Diao, Y., Immerman, N.: Recognizing patterns in streams with imprecise timestamps. VLDB **3**(1-2), 244–255 (2010)

Appendix A

The figures below present the precision and recall of the *fighting*, *meeting* and *moving* LTA under smooth and intermediate noise.

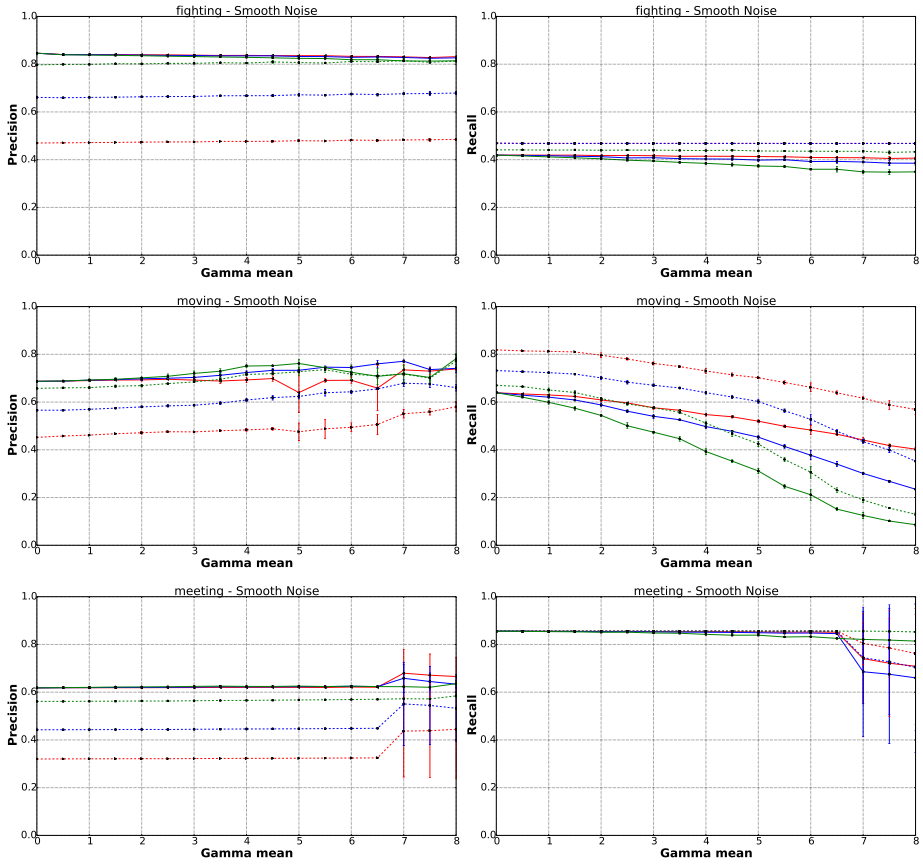

Fig. 6: The precision (left) and recall (right) of PIEC and Prob-EC per Gamma mean value for the *fighting*, *moving* and *meeting* LTA under smooth noise. The dashed lines display the performance of PIEC while the solid lines display the performance of Prob-EC. Red lines correspond to a 0.5 threshold value, blue lines to 0.7, while green lines correspond to a 0.9 threshold value.

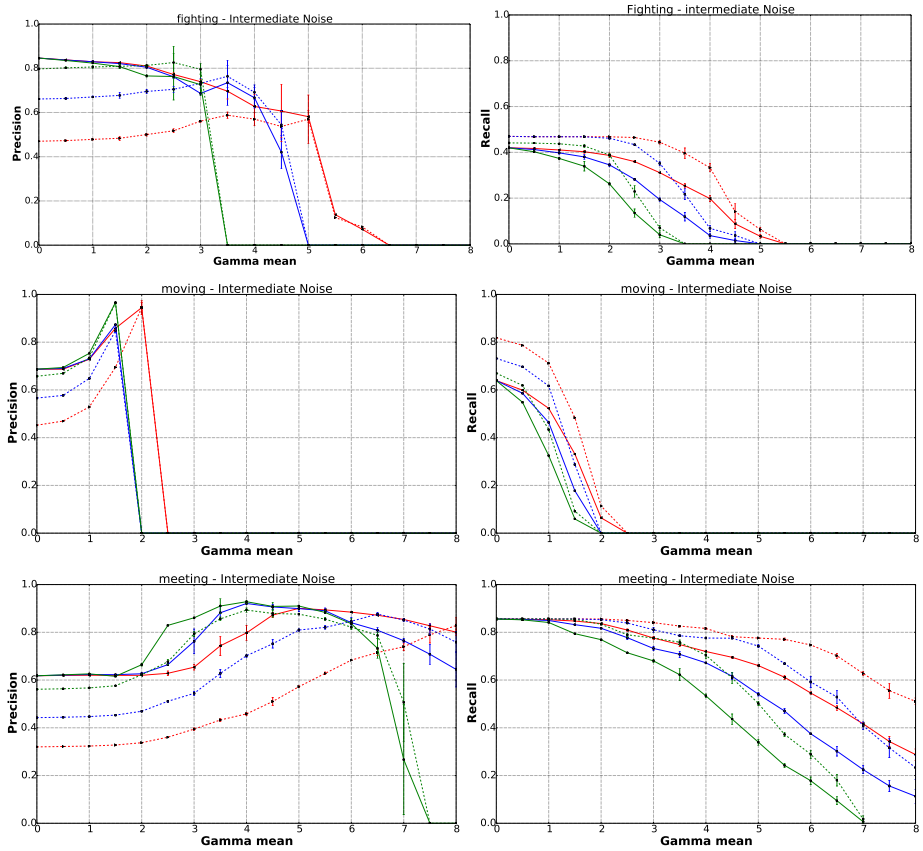

Fig. 7: The precision (left) and recall (right) of PIEC and Prob-EC per Gamma mean value for the *fighting*, *moving* and *meeting* LTA under intermediate noise.

Appendix B

For brevity, in all proofs below, we assume intervals $[i, j]$ with $0 < i < j < n$, where n is the last time-point of the dataset.

Proposition 1 *Every interval computed by PIEC has probability greater or equal to the given threshold $\mathcal{T}$.*

Proof Assume that PIEC computes the interval $[i, j]$ and that $P([i, j]) < \mathcal{T}$.

Since PIEC computes $[i, j]$, then, by Algorithm 1, we have that:

$$\begin{aligned} &dprange[i,j] \geq 0 \Leftrightarrow dp[j] - prefix[i-1] \geq 0 \Leftrightarrow \\ &\max_{k\in[j,n]}(prefix[k]) - prefix[i-1] \geq 0 \Leftrightarrow \max_{k\in[j,n]}(prefix[k]) \geq prefix[i-1] \end{aligned} \tag{13}$$

Moreover, since PIEC computes $[i, j]$, then:

$$\begin{aligned} &dprange[i, j+1] < 0 \Leftrightarrow dp[j+1] - prefix[i-1] < 0 \Leftrightarrow \\ &\max_{l \in [j+1,n]}(prefix[l]) - prefix[i-1] < 0 \Leftrightarrow \max_{l \in [j+1,n]}(prefix[l]) < prefix[i-1] \end{aligned} \tag{14}$$

From inequalities (13) and (14), we have that:

$$\max_{l \in [j+1,n]}(prefix[l]) < prefix[i-1] \leq \max_{k \in [j,n]}(prefix[k]) \tag{15}$$

Consequently:

$$\max_{l \in [j+1,n]}(prefix[l]) < \max_{k \in [j,n]}(prefix[k]) \Leftrightarrow \max_{k \in [j,n]}(prefix[k]) = prefix[j] \tag{16}$$

From formulas (15) and (16), we have:

$$prefix[i-1] \leq prefix[j] \tag{17}$$

However, according to our assumptions, we have $P([i, j]) < \mathcal{T}$, and according to formula (11), we have that:

$$\sum_{k=i}^{j} L[k] < \mathcal{T} \Leftrightarrow prefix[j] - prefix[i-1] < 0 \Leftrightarrow prefix[j] < prefix[i-1] \tag{18}$$

Formulas (17) and (18) are contradicting and thus our initial assuption that $P([i, j]) < \mathcal{T}$ does not hold.

Proposition 2 *For every interval $[i, j]$ computed by PIEC, there is no interval $[k, l]$ with*

- *$k = i$ and $j < l$, or*
- *$k < i$ and $j = l$, or*
- *$k < i$ and $j < l$,*

and $P([k, l]) \geq \mathcal{T}$.

Proof FIRST CASE: Assume that PIEC computes the interval $[i, j]$ and that there is an interval $[i, k]$ with $k > j$ and $P([i, k]) \geq \mathcal{T}$.

Since PIEC computes $[i, j]$, then we have from formula (15) that:

$$\max_{l \in [j+1,n]}(prefix[l]) < prefix[i-1] \tag{15'}$$

However, according to our assumptions, $P([i, k]) \geq \mathcal{T}$, which, according to formula (18), implies that:

$$prefix[i-1] \leq prefix[k] \tag{19}$$

Formulas (15') and (19) are contradicting, since $\max_{l \in [j+1,n]}(prefix[l]) < prefix[k]$ does not hold, and thus our initial assumption that there is an interval $[i, k]$ with $k > j$ and $P([i, k]) \geq \mathcal{T}$ does not hold.

FOR THE SECOND CASE, WE MAY NEED TO TAKE THE FOLLOWING CASES: (a) PIEC computes $[k, j]$ (in addition to computing $[i, j]$) and (b) PIEC does not compute $[k, j]$.