Event Detection in Continuous Video: An Inference in Point Process Approach

Zhen Qin and Christian R. Shelton

***Abstract*—We propose a novel approach towards event detection in real-world continuous video sequences. The method 1) is able to model arbitrary-order non-Markovian dependencies in videos to mitigate local visual ambiguities, 2) conducts simultaneous event segmentation and labeling, and 3) is time-window free. The idea is to represent a video as an event stream of both high-level semantic events and low-level video observations. In training, we learn a point process model called piecewise-constant conditional intensity model (PCIM) that is able to capture complex non-Markovian dependencies in the event streams. In testing, event detection can be modeled as the inference of high-level semantic events, given low-level image observations. We develop the first inference algorithm for PCIM and show it samples exactly from the posterior distribution. We then evaluate the video event detection task on real-world video sequences. Our model not only provides competitive results on the video event segmentation and labeling task, but also provides benefits including being interpretable and efficient.**

***Index Terms*—video event detection, event segmentation and labeling, video understanding, dependency modeling, video grammar, point process.**

I. Introduction

EVENT detection systems aim at identifying and localizing the classes of the events present in a video, such as a person sitting down, independently of the background. It is a key step towards real-world video understanding and has applications such as video indexing, video retrieval, and anomaly detection [33]. The large corpus of literature [34] usually model video event detection as a classification or labeling problem. Given coherent constituent parts from video segmentation in the temporal domain, a feature vector can be generated for each segment and serves as input for a discriminative classifier [18]. Another popular approach involves generating segmentation candidates via sliding windows and perform analysis at multiple temporal scales [20] [2].

However, video segmentation is an unsolved computer vision problem, and sliding windows approaches can be expensive. Also, visual ambiguity is unavoidable in real-world videos. By only looking at local visual features (either from a segmented clip or a time window), events of the same label might look quite different (when performed by different characters, or if the event has intrinsic intra-class variance), and events of different labels might look similar (for example, "punching" and "shaking hands" both consist of putting one's arm forward, see Fig. 1).

Z. Qin and C. Shelton are with the Department of Computer Science and Engineering, University Of California, Riverside, Riverside, CA, 92521 USA e-mail: {zqin001, cshelton}@cs.ucr.edu.

This work was supported by the Defense Advanced Research Projects Agency (FA8750-14-2-0010) and the National Science Foundation (IIS 1510741).

Fig. 1: (Left & Middle) Punching, (Right) Shaking Hand. Based only on visual features, the same action can look different, while different actions can possess similar appearances.

Contextual information could help to disambiguate, and we focus on modeling temporal context in this work. For example, if followed by the "person running" or "person falling down" events, we should be more certain that the event before is "punching", instead of "shaking hand".

We propose a new approach to explore temporal dependencies among events and visual observations in video: In training, given both observed low-level events (local visual features) and annotated high-level semantic events, we can build a point process model to learn complex dependencies in video event streams. In testing, the detection of high-level events can be modeled as an inference problem, given the observed low-level events. See Fig. 2 for an illustration of an event stream representation of video. This modeling approach allows us to leverage the machine learning and statistics literature on dependency modeling in point process.

We use a state-of-art point process model, called a piecewise-constant conditional intensity model (PCIM) [17]. PCIM captures the dependencies among the types of events through a set of piecewise-constant conditional intensity functions. A PCIM is represented as a set of decision trees (see Fig. 3 for an example), which provide model interpretability and allow for efficient model selection. In training, by applying PCIM learning on annotated videos, PCIM is able to learn the complex dependencies in the (annotated) video event streams, with the extra benefit of providing a meaningful video grammar.

In testing, the video event detection task we are interested in requires an inference algorithm for PCIM. An inference algorithm allows localizing and labeling high-level semantic events given only low-level visual observations in unannotated testing videos. An exact inference algorithm is able to take advantage of the rich dependencies learned in training, thus mitigating local visual ambiguities in video event detection. Also, inference in point process provides both time and label of inferred events, allowing automatic simultaneous event segmentation and labeling, without the need of sliding windows.

However, no inference algorithm has been proposed for

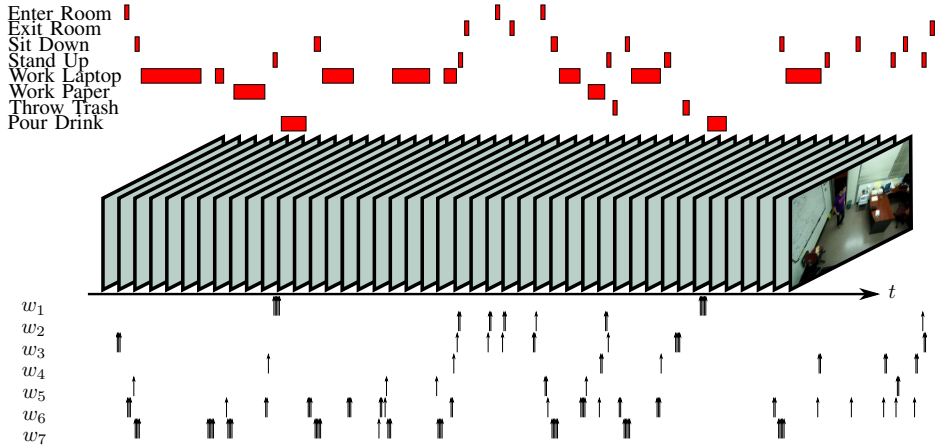

Fig. 2: A sample event stream representation of video. The events above the video clip are the semantic high-level events. ws are the low-level visual observations. Note for high-level events we use bars for a clearer illustration, in practice each of them is represented as two points (starting time and ending time with event labels). See text for more details.

PCIM that can condition on evidence (visual observations). Correctly filling in incomplete event streams from a PCIM is challenging, due to the complex non-Markovian dependencies between past and future evidence.

In this paper, we propose the first general inference algorithm for PCIMs, based on *thinning* for inhomogeneous Poisson process [26]. This inference algorithm can be used in the video event detection task, as well as any other tasks using PCIMs. Our formulation is an auxiliary Gibbs sampler that alternates between sampling a finite set of virtual event times given the current trajectory, and then sampling a new trajectory given the set of evidences and event times (virtual and actual). Our method is convergent, does not involve approximations like fixed time-discretization, and the samples generated can answer any type of query. We propose an efficient state-vector representation to maintain only the necessary information for diverging trajectories, reducing the exponentially increasing sampling complexity to linear in most cases. We show empirically our inference algorithm converges to the true distribution and permits effective query answering for PCIM models with both Markovian and complex non-Markovian dynamics.

We apply the PCIM inference algorithm to the video event detection task and show competitive results over state-of-arts on real-world long continuous videos. Our modeling approach is able to learn complex temporal dependencies in video, and exploit these dependencies to mitigate local visual ambiguities in event detection. The major contributions of our work are:

- A novel approach for video event segmentation and labeling via inference in point process, which does not rely on video pre-segmentation or sliding windows;
- The use of a PCIM to learn complex dependencies in video and provide meaningful video grammars; and
- The first exact inference algorithm for PCIMs that can be used for event detection and other tasks. [35] described a preliminary piece of this inference algorithm.

A. Related Work

We first describe related work in machine learning that models temporal dependencies. A dynamic Bayesian network (DBN) [10] models temporal dependencies between variables in discrete time. Continuous-time models have drawn attention recently in applications ranging from social networks [11][13] to genetics [8] to biochemical networks [15]. Continuous Time Bayesian Networks (CTBN) [29] are homogeneous *Markovian* models of the joint trajectories of discrete finite variables, analogous to DBNs. Non-Markovian continuous models allow the rate of an event to be a function of the process's history. Poisson Networks [36] constrain this function to depend only on the counts of the number of events during a finite time window. Hawkes processes [19] define the rate to be the sum of a kernel applied to each historic event, requiring the modeler to choose the form for the kernel.

A PCIM defines the intensity function as a decision tree, with internal nodes' tests mapping time and history to leaves. Each leaf is associated with a constant rate. A PCIM is able to model non-Markovian temporal dependencies, and is an order of magnitude faster to learn than Poisson networks. Successful applications include modeling supercomputer event logs and forecasting future interests of web search users [17]. While PCIMs have drawn attention recently [31] [44] and have potential usage in a wide variety applications, there is no general inference algorithm.

Inference algorithms developed for continuous systems are mainly for Markovian models (or specifically designed for a particular application). For CTBNs, there are variational approaches such as expectation propagation [12] and mean field [8], which do not converge to the true value as computation time increases. Sampling based approaches include importance sampling [14] and Gibbs sampling [37] [38] that converge to the true value. The latter is the current state-of-the-art method designed for general Markov Jump Processes (MJPs) and its extensions (including CTBNs). It uses the idea of uniformization [16] for Markov models, similar to thinning [26] for inhomogeneous Poisson processes. We note that our inference method for PCIM generalizes theirs to non-Markovian models.

Modeling temporal dependencies in general event streams

has wide applications. For example, users' behaviors in online shopping [45] and web searches [27], as well as electronic health records [44] can all be viewed as a stream of events over time. Event streams in video is a specific case of general event streams.

To identify and recognize events or actions in video, computer vision researchers mainly focused on the classification or labeling problem given pre-segmented video clips [34]. Feature representations used in the literature are generated using local features such as Space Time Interest Points (STIP) [24], Dense Trajectories [41], Fisher Vectors [32], and more recently, deep learned representations [18] [21] [42] [47], among many others. Some work explore contextual information and temporal structure within complex video events [1] [40] [48] [23], largely motivated by the popular TRECVID MED challenge [30]. Our model learns dependencies both within and among high-level events, and can also be applied to the complex event detection task.

However, real-world videos are continuous, and the task of video segmentation is an unsolved problem [9]. Also, most existing methods entail shot boundary detection, i.e., the segmentation of a video into continuously imaged (usually in terms of motion) temporal segments, which rarely maps to individual high-level events [5]. Recently, some work tries to address the problem of simultaneous video segmentation and labeling [20] [7] [6]. These methods mostly use the time-consuming sliding window approaches, which process each segment independently at multiple time scales. Post-processing such as duration priors and non-max suppression is required [6]. It is very difficult for these approaches to handle local visual ambiguities in real-world videos, since each event/video segment is processed independently from the others. High-level contexts at video level, such as temporal dependencies, have rarely been explored.

Some work explores Markovian dependencies among events [22], which is limited for real-world videos with complex dependencies. [50] models dependencies among events in a video based on Conditional Random Fields (CRF). However, it can only explore dependencies up to some fixed order (chosen manually), and the computation becomes infeasible when the order of dependency specified increases. Also, it assumes a long video has been segmented before doing dependency modeling. Recently, recurrent Neural Network (RNN) based methods are drawing more attention [46]. They focus more on sequence dependency modeling, instead of temporal modeling. Temporal modeling using PCIM has the advantage of being able to easily model cases such as "someone enters the office around 8am everyday", regardless of (potentially unbounded number of) events happened in between. Explicitly modeling time can also be beneficial in real-world videos with timestamps (such as videos from surveillance cameras). It is also easier for humans to interpret and generate rules from the decision tree representation of PCIM. Furthermore, RNN based approaches tend to be training data hungry. Our idea of using event stream models explicitly address temporal dependencies in the continuous-time domain and is the first to do so, to the best of our knowledge.

II. Background on PCIM

We first briefly review the background on Piecewise-constant conditional intensity model (PCIM).

Assume events are drawn from a finite label set L. An event then can be represented by a time-stamp t and a label l. An event sequence $x = \{(t_i, l_i)\}_{i=1}^n$, where $0 < t_1 < \ldots < t_n$. We use $h_i = \{(t_j, l_j) \mid (t_j, l_j) \in x, t_j < t_i)\}$ for the history of event i, when it is clear from context which x is meant. We define the ending time $t(y)$ of an event sequence y as the time of the last event in y, so that $t(h_i) = t_{i-1}$. A conditional intensity model (CIM) is a set of non-negative conditional intensity functions indexed by label $\{\lambda_l(t|x;\theta)\}_{l=1}^{|L|}$. The data likelihood is

$$p(x|\theta) = \prod_{l \in L} \prod_{i=1}^{n} \lambda_l(t_i|h_i;\theta)^{\mathbf{1}_l(l_i)} e^{-\Lambda_l(t_i|h_i;\theta)} \tag{1}$$

where $\Lambda_l(t|h;\theta) = \int_{t(h)}^{t} \lambda_l(\tau|h;\theta)d\tau$. The indicator function $\mathbf{1}_l(l^{'})$ is one if $l^{'} = l$ and zero otherwise. $\lambda_l(t|h;\theta)$ is the expected rate of event l at time t given history h and model parameters θ. Conditioning on the entire history causes the process to be non-Markovian. The modeling assumptions for a CIM are quite weak, as any distribution for x in which the timestamps are continuous random variables can be written in this form. Despite the weak assumptions, the per-label conditional factorization allows the modeling of label-specific dependence on past events.

A PCIM is a particular class of CIM that restricts $\lambda(h)$ to be piecewise constant (as a function of time) for any history, so the integral for Λ breaks down into a finite number of components and forward sampling becomes feasible. A PCIM represents the conditional intensity functions as decision trees. Each internal node in a tree is a binary test of the history, and each leaf contains an intensity. If the tests are piecewise-constant functions of time for any event history, the resulting function $\lambda(t|h)$ is piecewise-constant. Examples of admissible tests include

- Was the most recent event of label l?
- Is the time of the day between 6am and 9am?
- Did an event with label l happen at least n times between 5 seconds ago and 2 seconds ago?
- Were the last two events of the same label?

Note some tests are non-Markovian in that they require knowledge of more than just which event was most recent. See Fig. 3 for an example of a PCIM model.

The decision tree for label l maps the time and history to a leaf $s \in \Sigma_l$, where Σ_l is the set of leaves for l. The resulting data likelihood can be simplified:

$$p(x|S,\theta) = \prod_{l \in L} \prod_{s \in \Sigma_l} \lambda_{ls}^{c_{ls}(x)} e^{-\lambda_{ls} d_{ls}(x)}. \tag{2}$$

S is the PCIM structure represented by the decision trees; the model parameters θ are rates at the leaves. $c_{ls}(x)$ is the number of times label l occurs in x and is mapped to leaf s. $d_{ls}(x)$ is the total duration when the event trajectory for l is mapped to s. Together, c and d are the sufficient statistics for calculating the likelihood.

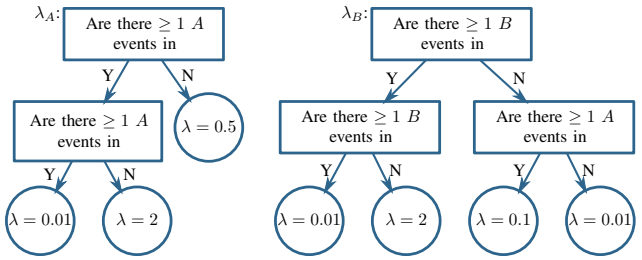

Fig. 3: Decision tree representing S and θ for events of labels A and B. Note the dependency among event labels (the rate of B depends on A). [17]

[17] showed that given the structure S, by using a product of Gamma distributions as a conjugate prior for θ, the marginal likelihood of the data can be given in closed form, and thus parameter estimation can be done in closed form. The prior density is given by

$$p(\lambda_{ls}|\alpha_{ls},\beta_{ls}) = \frac{\beta_{ls}^{\alpha_{ls}}}{\Gamma(\alpha_{ls})}\lambda_{ls}^{\alpha_{ls}-1}e^{-\beta_{ls}\lambda_{ls}}, \tag{3}$$

and the posterior density is given by

$$p(\lambda_{ls}|\alpha_{ls},\beta_{ls},x) = p(\lambda_{ls}|\alpha_{ls}+c_{ls}(x),\beta_{ls}+d_{ls}(x)). \tag{4}$$

Assuming the prior over the model parameters θ is a product of such priors, the marginal likelihood of data is

$$p(x|S) = \prod_{l\in L}\prod_{s\in\Sigma_l}\gamma_{ls}(x), \tag{5}$$

with

$$\gamma_{ls}(x) = \frac{\beta_{ls}^{\alpha_{ls}}}{\Gamma(\alpha_{ls})}\frac{\Gamma(\alpha_{ls}+c_{ls}(x))}{(\beta_{ls}+d_{ls}(x))^{\alpha_{ls}+c_{ls}(x)}}. \tag{6}$$

Then the authors choose to use a simple point estimate $E[\lambda_{ls}|x]$ for the rate, which is $\frac{\alpha_{ls}+c_{ls}(x)}{\beta_{ls}+d_{ls}(x)}$.

Furthermore, imposing a structural prior allows a closed form Bayesian score to be used for greedy tree learning. The local structure S_l can be chosen independently for each l by using a factored structural prior

$$p(S) \propto \prod_{l\in L}\prod_{s\in\Sigma_l}\kappa_{ls} \tag{7}$$

and the prior and the marginal likelihood that also factor over l. Given the current structure S_l (initialized as a single root), a new structure $S_l^{'}$ is considered by choosing a leaf s and expand it with a test to get a set of new leaves $\{s_1,\cdots,s_m\}$. The gain in the posterior of the structure is

$$\frac{p(S_l^{'}|x)}{p(S_l|x)} = \frac{\kappa_{ls_1}\gamma_{ls_1}(x)\cdots\kappa_{ls_m}\gamma_{ls_m}(x)}{\kappa_{ls}\gamma_{ls}(x)}. \tag{8}$$

The new structure with the largest gain is chosen if the gain is larger than 1.

III. Event Detection in Video as Inference in Point Process

We apply PCIM to event localization and labeling in video. We first show how to represent videos as event streams. A PCIM can be learned from training videos, represented by event streams, to encode nonlinear temporal dependencies between high-level events and low-level observations. In testing, an inference algorithm can be used to infer high-level events given low-level visual observations. This framework is among the first to encode temporal context for the event detection in long continuous video task.

A. Representation

Assume there are M high-level events in a video dataset, each of which is an event with high-level semantics, such as "a person sitting down" and "a person working on a laptop". We generate $2M$ event labels to be used in PCIM: $\{s_1,\ldots,s_M\}$ and $\{e_1,\ldots,e_M\}$. s_i indicates the starting of a high-level event type i and e_i indicates the ending of a high-level event type i. These are the event types that are labeled in training and to be inferred in testing.

Given a video, we divide it into segments of fixed length, and a feature vector is generated for each of the segments. This step is flexible, any existing video descriptors might be applied here. Then we learn a dictionary (e.g. using K-means clustering), so that each segment can be assigned to one visual word in $\{w_1,\ldots,w_K\}$ and a time (we use the middle time of each segment in the video). Together with the starting and ending of high-level events, we have an event stream representation of a video. See Fig. 4 for an example.

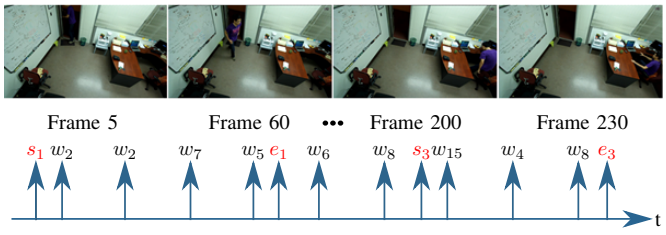

Fig. 4: An illustration of an event stream representation for video. Low-level observations are represented as regularly spaced events from a dictionary (the ws). s_1 and e_1 indicate the starting and ending of "entering room". s_3 and e_3 indicate the starting and ending of "sitting down".

There are several benefits for this representation: First, by using fixed-length segment, we do not assume semantic video pre-segmentation. Second, the usage of starting and ending of high-level events enables automatic temporal localization and labeling. Note that even though the low-level visual words are regularly sampled, each word is sparse across the timeline, which is suitable for a continuous-time model.

B. Training

Given an event stream representation of video, training can be done by directly feeding the training data into the PCIM learning algorithm. The resulted PCIM encodes temporal dependencies between both high-level events and low-level visual observations. There are three types of dependencies a PCIM can learn:

Dependency between high-level events. Global dependencies between high-level events can be modeled, which helps to

mitigate visual ambiguities by utilizing temporal context. For example, after a person working on laptop, the probability of a person standing up should be higher than the person sitting down. PCIM is also able to learn the dependency between s and e for each high-level events, which encodes the temporal range distribution of each event type.

Dependency between high-level and low-level events. This type of dependency can be treated as local dependency. Certain low-level observations indicate the appearance of high-level events, or as a generative model, the high-level events "cause" low-level features. In testing, low-level visual words are observed, and are responsible for proposing high-level events.

Dependency between low-level events. This kind of dependency provides interesting information about how low-level observations can be correlated from a video grammar perspective. But for the event detection problem, it is not very useful as in testing all low-level observations are observed.

C. Testing as Inference

Given a PCIM learned from training data, we can model the problem of event detection in video as the inference of high-level events, given low-level observations. In other words, all the ws are observed, while the ss and es are completely unobserved. We can then apply our new inference algorithm, ThinnedGibbs, detailed in Sec. IV, to infer the starting and ending times of the high-level events. See Fig. 5 and Fig. 6 for an illustration.

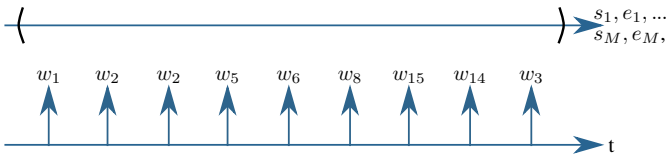

Fig. 5: In testing, the low-level events are fully observed, while the high-level events are not observed (in parentheses).

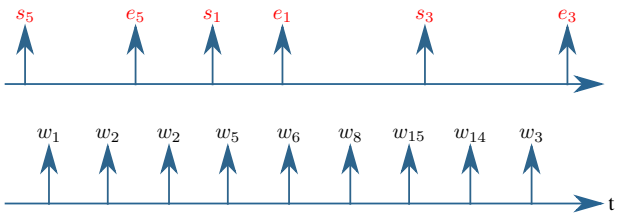

Fig. 6: After running inference, each sample would fill in the unobserved intervals for high-level events, which indicate their starting and ending times.

IV. Auxiliary Gibbs Sampling for PCIM

In this section we introduce our new inference algorithm for PCIM, called ThinnedGibbs, based on the idea of *thinning* for inhomogeneous Poisson processes. We handle incomplete data in which there are intervals of time during which events for particular label(s) are not observed.

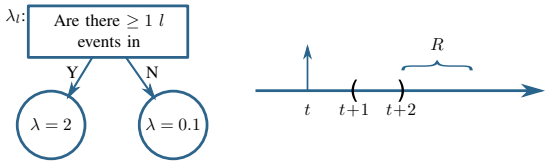

Fig. 7: A simple PCIM with a partially observed trajectory. The vertical solid arrow indicates an evidence event. Areas between parentheses are unobserved. History alone indicates there should be events filled in, while the future (no events in R) provides contradictory evidence.

A. Why Inference in PCIM is Difficult

Filling in partially observed trajectories for PCIM is hard due to the complex dependencies between unobserved events and both past and future events. See Fig. 7 for an example. While the history (the event at t) says it is likely that there should be events in the unobserved area (with an expected rate of 2), future evidence (no events in R) is contradictory: If there were indeed events in the unobserved area, those events should stimulate events happening in R.

Such a phenomenon might suggest existing algorithms such as the forward-filtering-backward-sampling (FFBS) algorithm for discrete-time Markov chains. However, there are two subtleties here: First, we are dealing with non-Markovian models. Second, we are dealing with continuous-time systems, so the number of time steps over which to propagate is infinite.

B. Thinning

Thinning [26] can be used to turn a continuous-time process into a discrete-time one, without using a fixed time-slice granularity. We select a rate λ^* greater than any in the inhomogeneous Poisson process and sample from a *homogeneous* process with this rate. To get a sample from the original inhomogeneous process, an event at time t is thinned (dropped) with probability $1 - \frac{\lambda(t)}{\lambda^*}$.

This process can also be reversed. Given the set of thinned event times (from the inhomogeneous process), the extra events can be added by sampling from a Poisson process with rate $\lambda^* - \lambda(t)$. The cycle can then repeat by thinning the new total set of times. At each cycle, the times (after thinning) are drawn from the original inhomogeneous process. We will use this type of cycle in our sampler.

The difficulty is that a PCIM is not an inhomogeneous Poisson process. The intensity depends on the entire history of events, not just the current time. For thinning, this means that we cannot independently sample whether each event is to be thinned. Furthermore, we wish to sample from the posterior, conditioned on evidence. All evidence (both past and future) affect the probability of a specific thinning configuration.

C. Overview of Our Inference Method

To overcome both of these problems, we extend thinning to an auxiliary Gibbs sampler in the same way that [37], [38] extended Markovian-model uniformization [16] (a specific example of thinning in a Markov process) to a Gibbs sampler.

To do this we introduce auxiliary variables representing the events that were dropped. We call these events *virtual* events.

As a standard Gibbs sampler, our method cycles through each variable in turn. In our case, a variable corresponds to an event label. For event label l, let x_l be the sampled event sequence for this label. Let Y_l be all evidence (for l and other labels) and all (currently fixed) samples for other labels. Our goal is to sample from $p(x_l \mid Y_l)$.

Let v_l be the virtual events (the auxiliary variable) associated with l and $z_l = x_l \cup v_l$ (all event times virtual and non-virtual). Our method first samples from $p(v_l \mid x_l, Y_l)$ and then samples from $p(x_l \mid z_l, Y_l)$. The first step adds virtual events given the non-virtual events are "correct." The second step treats all events as potential events and drops or keeps events. The dropped events are removed completely. The kept events, x_l, remain as the new sampled trajectory for label l.

The proof of correctness follows analogously to that of [38] for Markovian systems. But, the details for sampling from $p(v_l \mid x_l, Y_l)$ and $p(x_l \mid z_l, Y_l)$ differ. We describe them next.

D. Sampling Auxiliary Virtual Events with Adaptive Rates

Sampling from $p(v_l \mid x_l, Y_l)$ amounts to adding just the virtual (dropped) events. As the full trajectory (x_l for all l) is known, the rate at any time step for a virtual event is independent of any other virtual events. Therefore, the process is an inhomogeneous Poisson process for which the rate at t is equal to $\lambda^* - \lambda_l(t|h)$ where h is fully determined by x_l and Y_l. Recall that $\lambda_l(t|h)$ is piecewise-constant in time, so sampling from such an inhomogeneous Poisson process is simple.

The auxiliary rate, λ^*, must be strictly greater than the maximum rate possible for irreducibility. We use an auxiliary rate of $\lambda^* = 2\max(\lambda(t|h))$ to sample virtual events in the unobserved intervals. This choice trades off well between mixing time and computational complexity in the experiments.

A naïve way to pick λ^* is to find λ_{max}: the maximum rate in the leaves of PCIM, and use $2\lambda_{max}$. However, there could be unobserved time intervals with a possible maximum rate much smaller than λ_{max}. Using λ_{max} in those regions would generate too many virtual events, most of which will be dropped in the next step leading to computational inefficiency. We therefore use an adaptive strategy.

Our adaptive $\lambda^*(t|h)$ cannot depend on x_l (this would break the simplicity of sampling mentioned above). Therefore, we determine $\lambda^*(t|h)$ by passing (t, h) down the PCIM tree for λ_l. At each internal node, if the branch does not depend on x_l, we can directly take one branch. Otherwise, the test is related to the sampled events, and we take the maximum rate of taking both branches. This method results in $\lambda^*(t|h)$ as a piecewise-constant function of time (for the same reasons that $\lambda_l(t|h)$ is piecewise-constant).

Consider Fig. 8 as an example. When sampling event $l = A$ on the interval $[1, 5)$, we would not take the left branch at the root (no matter what events for A have been sampled), but must maximize over the other two leaves (as different x_l values would result in different leaves). This results in a $\lambda^* = 4$ over this interval, which is smaller than 6.

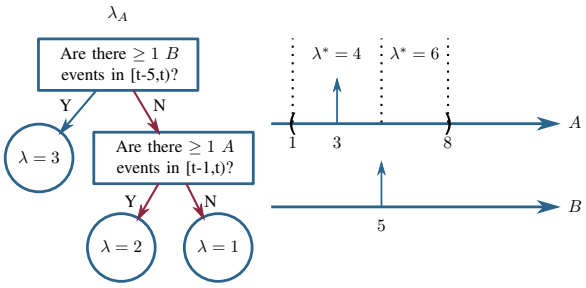

Fig. 8: Adaptive auxiliary rate example. When sampling A, the branch to take at the root does not depend on unobserved events for A. If the test is related to the sampled event, we take the maximum rate from both branches. The red arrows indicate the branches to take between time $[1, 5]$, and $\lambda^* = 2 \times 2$ in that interval, instead of 6.

E. The Naïve FFBS Algorithm

Once these virtual events are added back in, we take z_l (the union of virtual and "real" sampled events) as a sample from the Poisson process with rate λ^*, ignoring which were originally virtual and which were originally "real." We thin this set to get a sample from the conditional marginal over l.

The restriction to consider events only at times in z_l transforms the continuous-time problem into a discrete one. Given z_l with m possible event times $(z_{l,1}, z_{l,2}, \ldots, z_{l,m})$, let $b = \{b_i\}_{i=1}^m$ be a set of binary variables, one per event, where $b_i = 1$ if event i is included in x_l (otherwise $b_i = 0$ and the event is not included in x_l). Thus sampling b is equivalent to sampling x_l (z_l is known) as it specifies which events in z_l are in x_l. Let $Y_l^{i:j}$ be the portion of Y between times $z_{l,i}$ and $z_{l,j}$, and $b^{i:j} = \{b_k | i \leq k \leq j\}$ We wish to sample b (and thereby x_l) from

$$p(b \mid Y) \propto \left(\prod_i p(Y_l^{i-1:i}, b_i \mid b^{1:i-1}, Y_l^{1:i-1}) \right) p(Y_l^{m:\infty} \mid b) \tag{9}$$

where the final $Y_l^{m:\infty}$ signifies all of the evidence after the last virtual event time $z_{l,m}$ and can be handled similarly to the other terms.

The most straight-forward method for such sampling considers each possible assignment to b (of which there are 2^m). For each interval, we multiply terms from Eq. 9 of the form $p(Y_l^{i-1:i}, b_i \mid b^{1:i-1}, Y_l^{1:i-1}) =$

$$p(Y_l^{i-1:i} \mid b^{1:i-1}, Y_l^{1:i-1}) p(b_i \mid b^{1:i-1}, Y_l^{1:i}) \tag{10}$$

where the first term is the likelihood of the trajectory interval from $z_{l,i-1}$ to $z_{l,i}$ and the second term is the probability of the event being thinned, given the past history. The first can be computed by tallying the sufficient statistics (counts and durations) and applying Eq. 2. Note that these sufficient statistics take into account $b^{1:i-1}$ which specifies events for l during the unobserved region(s), and the likelihood must also be calculated for labels $l' \neq l$ for which $\lambda_{l'}(t|h)$ depends on events from l. The second term is equal to $\frac{\lambda_l(t|h)}{\lambda^*(t)}$ if $b_i = 1$ (and $1 - \frac{\lambda_l(t|h)}{\lambda^*(t)}$ if $b_i = 0$). The numerator's dependence on the full history similarly dictates a dependence on $b^{1:i-1}$.

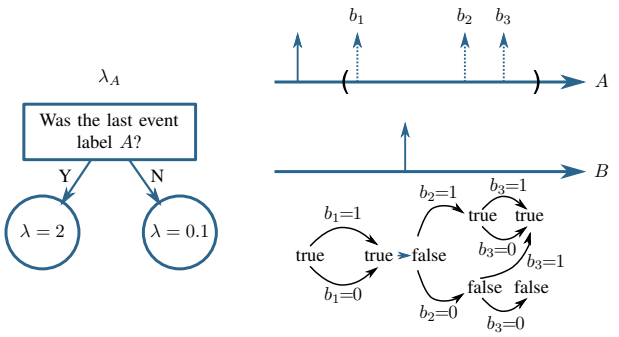

Fig. 9: Dotted events are the virtual events that we sample as binary variables (b_i is 1 if event i is kept). The state diagram below the trajectory indicates the state of the test as we diverge (keep or drop a virtual event). Though there are 2^3 possible configurations, state merges can reduce the exponentially increasing complexity to linear in this case.

This might be formulated as a naïve FFBS algorithm: To generate one sample, we propagate possible trajectories forward in time, multiplying in Eq. 10 at each inter-event interval to account for the evidence. Every time we see a virtual event, each possible trajectory diverges into two (depending on whether the virtual event is to be thinned or not). By the end, we have all 2^m possible trajectories, each with its probability (Eq. 9). We sample one trajectory as the output, in proportion of the calculated likelihoods. As we explicitly keep all possible trajectories, the sampled trajectory immediately tells us which virtual events are kept, so no actual backward pass is needed.

F. An Efficient State-Vector Representation

The naïve FFBS algorithm is not practical, as the number of possible trajectories grows exponentially with the number of auxiliary virtual events, m. We propose a more efficient state-vector representation to only keep the necessary information for each possible trajectory. The idea takes advantage of the structure of the PCIM and leads to state merges, similar to what happens in FFBS for hidden Markov models (HMMs).

The terms in Eq. 10 depend on $b^{1:i-1}$ only through the tests in the internal nodes of the PCIM trees. Therefore, we do not have to keep track of all of $b^{1:i-1}$ to calculate these likelihoods, but only the current state of such tests that depend on events with label l. For example, a test that asks "Is the last event of label l?" only needs to maintain a bit as the indicator. The test "Are there more than 3 events of label q in the last 5 seconds?" for $q \neq l$ has no state, as $b^{1:i-1}$ does not affect its choice. By contrast, a test such as "Is the last event of label q?" does depend on b, even if $q \neq l$.

As we propagate forward, we merge $b^{1:i}$ sequences that result in the same set of states for all internal tests inside the PCIM. See Fig. 9 as a simple example. Though there are 8 possible trajectories, they merge to only 2 states that we can sample from. Similar to the FFBS for an HMM, we need to maintain the transition probabilities in the forward pass and use them in a backward sampling pass to recover the full trajectory, but such information is also linear.

Note that this conversion to a Markov system for sampling is *not* possible in the original continuous-time system. Thinning allowed this by randomly selecting a few discrete time points and thereby restricting the possible state space to be finite.

The state space depends on the actual tests in the PCIM model. See Tbl. I for the tests we currently support and their state representations. The LastStateTest and StateTest are used to support discrete finite variable systems, such as a CTBN. We can see that for tests that only depend on the *current* time (i.e. TimeTest), the diverging history does not affect them, so no state is needed. For Markovian tests (LastEventTest and LastStateTest), we only need a Boolean variable. For the non-Markovian test (EventCountTest), the number of possible states does grow exponentially with the number of virtual events maintained in the queue. This is the best we can do and still be exact. It is much better than growing with the number of all virtual events. However, note that commonly $lag2 = 0$ and n is small. In this case, the state space size at any point is bounded as $\binom{m'}{n}$, where m' is the maximum number of sampled events in any time interval of duration $lag1$ (which is upper bounded by m). If n is 1, this is linear in the number of samples generated in during $lag1$ time units.

As noted above, if the test is not related to the sampled event (for example, we are sampling event $l = A$ and the test is "are there ≥ 3 B events in the last 5 seconds?"), the state of the test is set to null. This is because the evidence and sampled values for B (which is not the current variable for Gibbs sampling) can answer this test without reference to samples for l.

See Alg. 1 for the algorithm description for resampling event l. The complete algorithm iterates this procedure for each event label to get a new sample. The helper function UpdateState(s,b,t) returns the new state given the old state (s), the new time (t), and whether an event occurs at t (b). SampProbMap(M) takes a mapping from objects to positive values (M) and randomly returns one of the objects with probability proportional to the associate value. AddtoProbMap(M,o,p) checks to see if o is in M. If so, it adds p to the associated probability. Otherwise, it adds the mapping $o \to p$ to M.

G. Extended Example

Fig. 10 shows an example of resampling the events for label A on the unobserved interval $[0.8, 3.5)$. On the far left is the PCIM rate tree for event A. Box (a) shows the sample from previous iteration (single event at 2.3). Dashed lines and λ show the piecewise-constant intensity function given the sample. Box (b) shows the sampling of virtual events. For this case $\lambda^* = 3$ for all time. $\lambda^* - \lambda$ is the rate for virtual events. The algorithm samples from this process, resulting in two virtual events (dashed). In box (c) all events become potential events. The state of the root test is a queue of recent events. The state of the other test is Boolean (whether A is more recent). On the bottom is the lattice of joint states over time. Solid arrows indicate $b_i = 1$ (the event is kept). Dash arrows indicate $b_i = 0$ (the event is dropped). Each arrow's weight is as per Eq. 10. The probability of a node is the sum over all paths to the node of the product of the weights on the path (calculated by dynamic programming). In box (d) a single

TABLE I: Tests and their corresponding state representations.

Test	Example	State Representation	Property
TimeTest	Is the time between 6am and 9am?	Null	independent of b
LastEventTest	Is the last event of type A?	Boolean	Markovian
EventCountTest	Are there $\geq n$ events of type A in $[t-lag1, t-lag2]$?	A queue maintaining all the times of A between $[t-lag2, t]$, and the most recent n events between $[t-lag1, t-lag2]$.	Non-Markovian
LastStateTest	Is the last sublabel of var $A=0$?	Boolean	Markovian
StateTest	Is the current sublabel of var $A=0$?	Null	independent of b

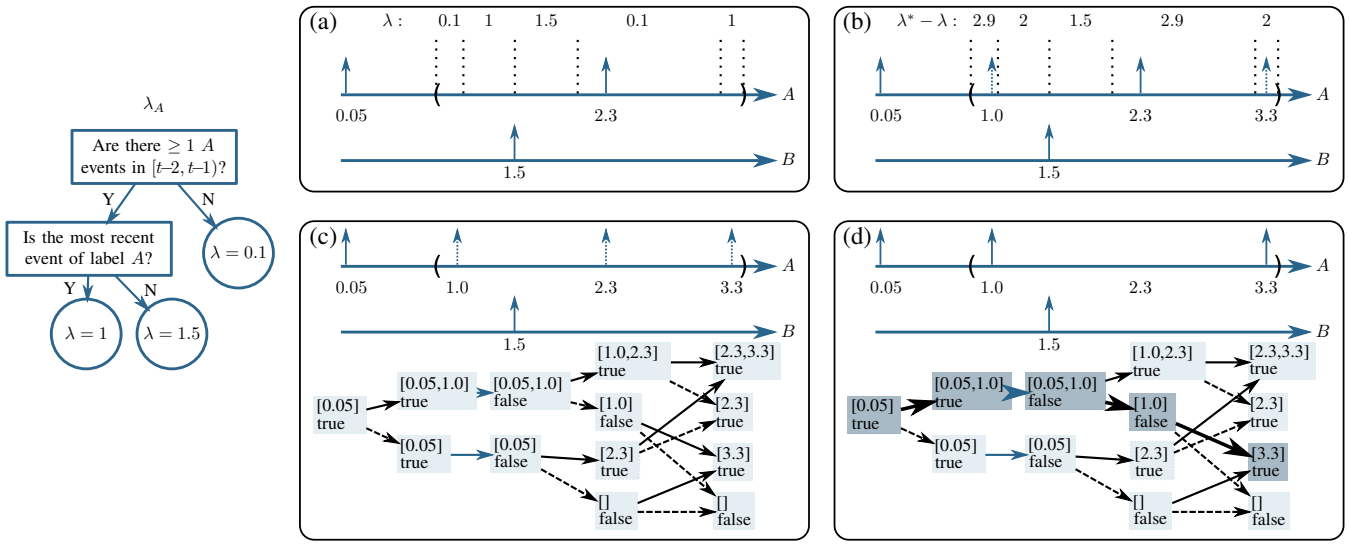

Fig. 10: Extended Example, see Section IV-G

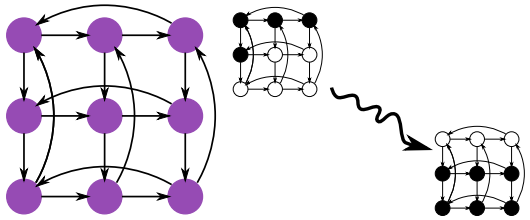

Fig. 11: The toroid network and observed patterns [12].

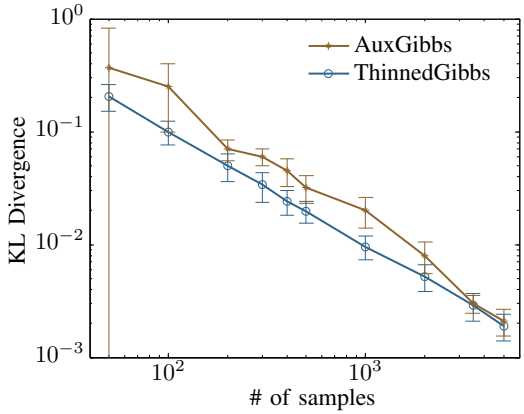

Fig. 12: Number of samples versus KL divergence for the toroid network. Both axes are on a log scale.

path is sampled with backward sampling, shown in bold. This path corresponds to keeping the first and last virtual events and dropping the middle one.

V. Experiments

A. ThinnedGibbs Validation

We perform inference with our method on both Markovian and non-Markovian models, and compare the result with the ground-truth statistics. For both we show our result converges to the correct result. Ours is the first that can successfully perform inference tasks for non-Markovian PCIMs.

1) Verification on the Ising Model: We first evaluate our method, ThinnedGibbs, on a network with Ising model dynamics. The Ising model is a well-known interaction model with applications in many fields including statistical mechanics, genetics, and neuroscience [8]. The model is Markovian and has been tested by several prior inference methods for CTBNs.

Using this model, we generate a directed toroid network structure with cycles following [12]. Nodes can take values -1 and 1, and follow their parents' states according to a coupling strength parameter (β). A rate parameter (τ) determines how fast nodes toggle between states. We test with $\beta = 0.5$ and $\tau = 2$. The network and the evidence patterns are shown in Fig. 11. The network starts from a deterministic state: at $t = 0$ variables $1-5$ are $+1$ and $6-9$ are -1. At $t = 1$, variable $1-3$ have switched to -1, $4-5$ remain $+1$, and $6-9$ have switched to $+1$. The nodes are not observed between $t = 0$ and $t = 1$. We query the marginal distribution of nodes at $t = 0.5$ and measure the sum of the KL-divergences of all marginals against the ground truth. We compare with the state-of-the-art CTBN Auxiliary Gibbs method [38]. Other existing methods

Algorithm 1: Resampling event l

input: The previous trajectory (x_l, Y_l)
output: The newly sampled $x_l^{'}$
for *each unobserved interval for* l **do**
  Find piecewise constant $\lambda^*(t|h)$ using Y_l
  Find piecewise constant $\lambda(t|h)$ using x_l, Y_l
  Sample virtual events v_l with rate $\lambda^*(t|h) - \lambda(t|h)$
Let $z_l = x_l \cup v_l$, $m = |z_l|$, and s_0 be the initial state
AddtoProbMap(S_0,s_0,1.0)
for $i \leftarrow 1$ *to* m **do**
  for *each* $\{(s_{i-1}, \cdot) \rightarrow p\}$ *in* S_{i-1} **do**
    $p_{keep} = p(E_{i-1:i}, b_i = 1 \mid s_{i-1}, E_{1:i-1})$
    $p_{drop} = p(E_{i-1:i}, b_i = 0 \mid s_{i-1}, E_{1:i-1})$
    $s_i^{keep} \leftarrow$ UpdateState(s_{i-1}, true, $z_{l,i}$)
    $s_i^{drop} \leftarrow$ UpdateState(s_{i-1}, false, $z_{l,i}$)
    AddtoProbMap(S_i,(s_i^{keep}, $z_{l,i}$), $p \times p_{keep}$)
    AddtoProbMap(S_i,(s_i^{drop}, $\emptyset$), $p \times p_{drop}$)
    AddtoProbMap($T_i(s_i^{keep})$, $(s_{i-1}, z_{l,i})$, $p \times p_{keep}$)
    AddtoProbMap($T_i(s_i^{drop})$, $(s_{i-1}, \emptyset)$, $p \times p_{drop}$)
Update S_m by propagating until ending time
$x_l^{'} \leftarrow \emptyset$ and $(s_m^{'}, t) \leftarrow$ SampProbMap(S_m)
if $t \neq \emptyset$ **then**
  $x_l^{'} \leftarrow x_l^{'} \cup \{t\}$
for $i \leftarrow m - 1$ *to* 1 **do**
  $(s_i^{'}, t) \leftarrow$ SampProbMap($T_{i+1}(s_{i+1}^{'})$)
  if $t \neq \emptyset$ **then**
    $x_l^{'} \leftarrow x_l^{'} \cup \{t\}$
return x_l

either produce similar or worse results [3]. For example, the mean field variational approach [8] produce error that is above the error range of the methods we use. We vary the sample size between 50 and 5000, and set the burn-in period to be 10% of this value. We run the experiments 100 times, and plot the means and standard deviations.

Results in Fig. 12 verify that our inference method indeed produces results that converge to the true distribution. Our method reduces to that of [38] in this Markovian model. Differences between the two lines are due to slightly different initializations of the Gibbs Markov chain and not significant.

2) Verification on a Non-Markovian Model: We further verify our method on a much more challenging non-Markovian PCIM (Fig. 13). This model contains several non-Markovian EventCountTests. We have observations for event A at $t = 0.4, 0.6, 1.8, 4.7$ and for event B at $t = 0.1, 0.2, 3.4, 3.6, 3.7$. Event A is not observed on $[2.0, 4.0)$ and event B is not observed on $[1.0, 3.0)$.

In produce ground truth, we discretized time and converted the system to a Markovian system. Note that because the time since the last A event is part of the state, as the discretization becomes finer, the state space increases. For this small example, this approach is just barely feasible. We continued to refine the discretization until the answer stabilized. The ground-truth expected total number of A events between $[0, 5]$ is 22.3206 and the expected total number of B events is 11.6161. That is, there are about 18.32 A events and 6.62

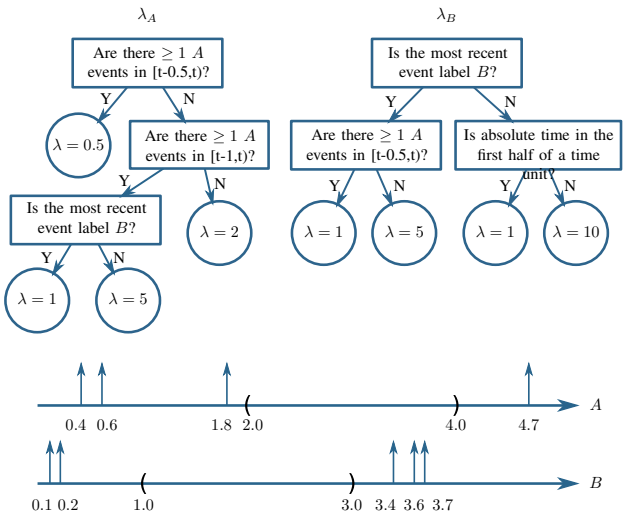

Fig. 13: Non-Markovian PCIM and evidence. End time is 5.

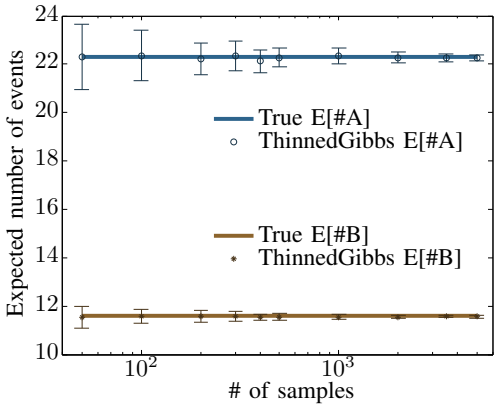

Fig. 14: Number of samples versus the inferred expected number of events. The horizontal axis is on a log scale.

B events in the unobserved areas. Note that if the evidence is changed to have no events these numbers drop to 1.6089 and 8.6866 respectively and if the evidence after the unobserved intervals is ignored the expectations are 22.7183 and 8.6344 respectively. Therefore the evidence (both before and after the unobserved intervals) is important to incorporate in inference.

We compare our inference method to the exact values, again varying the sample size between 50 and 5000 and setting the burn-in period to be 10% of this value. We ran the experiments 100 times and report the mean and standard deviation of the two expectations. Our sampler has very small bias and therefore the average values match the true value almost exactly. The variance decreases as expected, demonstrating the consistent nature of our method. See Fig. 14. We are not aware of existing methods that can perform inference on this type of model to which we could compare.

B. Experiments on Event Detection

We tested our proposed event detection approach on three challenging datasets: Hollywood [25], UCLA office [39], and PKU-MMD [28]. The Hollywood dataset is a human action dataset from movies, the UCLA dataset is from a video camera

TABLE II: Comparison of the event detection result on the Hollywood dataset. In each cell, the first number is precision and the second number is recall. Seg means the method requires video pre-segmentation.

Events	SVM(seg)	Hoai[20](seg)	SSM[7]	Ours
AnswerPhone	0.62/0.30	0.64/0.32	0.64/0.43	0.62/0.50
HugPerson	0.46/0.29	0.46/0.29	0.51/0.33	0.45/0.35
Kiss	0.39/0.46	0.40/0.48	0.44/0.59	0.45/0.55
SitDown	0.36/0.35	0.36/0.36	0.39/0.38	0.41/0.40
Overall	0.45/0.34	0.47/0.36	0.50/0.42	0.48/0.45

recording students' daily activities in an office, and the PKU-MMD dataset is a recent large-scale benchmark dataset for continuous multi-modal human action understanding.

1) Evaluation on Hollywood Dataset: The Hollywood dataset focuses on realistic human actions including AnswerPhone, Kiss, SitDown, HugPerson, StandUp, HandShake, SitUp, and GetOutCar. This dataset has two disjoint subsets with 219 video sample in the training set and 211 in the test set. We follow [7] for the experimental setting: we only focus on the first four classes to be recognized. Since the dataset contains only pre-segmented clips, new video clips of longer durations are created with enforced temporal relationships. 1-order dependency (such as SitDown-AnswerPhone) and 2-order dependency (such as HugPerson-Kiss-SitDown) are inserted. 40 such video samples were formed (see more details in [7]). To generate the visual sequences, we use mean pooling of STIP features to generate a feature vector for each 20-frame video segment, then use k-means (k is fixed to 200 for this dataset) to assign a label to each of the segment. For PCIM structure learning, we fix the bank of possible PCIM tests as EventCountTests (see Tbl. I) with $(n, lag1, lag2) \in \{1,2,3\} \times \{2,3,4,5,6\} \times \{0,1,2\}$ (omitting tests for which $lag1 \leq lag2$) and LastEventTest for all high-level and low-level event labels. We show some meaningful learned dependencies in Tbl. IV. For each run, we set the burn-in period to be 100 samples, and report the average performance of the next 20 samples.

We compare with methods that require pre-segmentation (linear SVM and [20]), as well as state-of-art method that does simultaneous segmentation and labeling ([7]). Note that results reported with pre-segmentation is biased since video segmentation is very challenging itself (for this dataset, we directly use the segments before concatenation). We perform 5-fold cross validation and the results averaged across 5 runs are reported in Tbl. II.

Our approach shows competitive performance when comparing with state-of-art methods. Though SVM and Hoai[20] are already reported on segmented videos, they still have worse performance due to the local visual ambiguities in this challenging dataset. Methods considering temporal context (SSM[7] and ours) are able to produce better results. Our approach is able to learn from the enforced temporal constraints and generate more reliable event candidates, leading to a better overall recall. Also, SSM[7] relies on training classifiers and doing inference at multiple scales and is more computationally expensive. During inference, for each video, our method generates 20 samples in less than a second, while the running time of our SSM implementation based on LibSVM[4] is in the order of minutes.

TABLE III: High-level Event Types in the UCLA Office Datasets.

ID	Event Type	ID	Event Type
1	Enter Room	2	Exit Room
3	Sit Down	4	Stand up
5	Work on Laptop	6	Work on Paper
7	Throw Trash	8	Pour Drink
9	Pick Phone	10	Place Phone Down

TABLE IV: Examples of meaningful structures learned by PCIM on the Hollywood and UCLA dataset. Time unit used is 20 frames.

Structure Learned	Semantics
if e_3 in $[t-1, t)$ rate of $s_1 = 0.67$	The ending of "SitDown" stimulates "AnswerPhone". (Hollywood)
if w_{57} is last event and if w_{12} in $[t-1, t)$ rate of $s_2 = 0.33$	Visual words in specific order stimulate "Kiss". (Hollywood)
if s_5 in $[t-2, t)$ rate of $w_3 = 0.68$	The starting of "work on laptop" tends to generate w_3. (UCLA)
if s_3 in $[t-5, t-2)$ rate of $e_3 = 0.7$	This encodes the duration distribution of "sit down". (UCLA)
if e_5 in $[t-3, t-1)$ rate of $s_4 = 0.22$	The ending of "work on laptop" stimulates "stand up". (UCLA)

2) Evaluation on UCLA Office Dataset: The UCLA office dataset consists of 3 videos of a total length of 32 minutes, in which actors perform 10 kinds of actions in an office setting. See Tbl. III for the high-level events types and their ID numbers. (We also use mean pooling of STIP features [24] generated for each segment, but other methods are applicable.)

We perform 3-fold cross validation, using 2 videos for training and 1 video for testing. We use 20 frames as the segment length and 30 for the dictionary size (K). For PCIM structural learning and sample size, we use the same setting as in the Hollywood dataset. PCIM is able to learn 3 major kinds of meaningful dependencies that are useful for the event localization and labeling task. We show some examples in Tbl. IV. It is the combination of these rules that encodes complex dependencies in video.

We compare with state-of-art method that require pre-segmentation [50]. This method also considers temporal dependencies in a Conditional Random Field framework. Due to the complexity of inference, this method can only encode dependencies up to a fixed order. Pre-segmentation of dataset is highly nontrivial and we notice it requires manual intervention. Thus we used the segmented data from the authors of [50]. We also compare with an SVM-based approach that classifies the video segments into one of the ten high-level events plus the null event. After classification, consecutive segments of the same labels are merged to one event. This method also relaxes the necessity of video pre-segmentation, but only considers local visual evidences.

We report the results in Tbl. V. Comparing with the Hollywood dataset, this dataset contains real-world continuous videos with natural high-order temporal dependencies. SVM-based approaches can produce reasonable overall result for this dataset. For this dataset, certain dimensions in the feature

vector are salient. For example, the extracted features contain location information that is salient for this dataset because the camera is static and certain events only happen at certain locations (for example, "pour drink" always happen close to the drinking machine.) SVM is able to get good result for most event classes, and tends to confuse between certain pairs of event types (i.e. "enter room" and "exit room", "stand up" and "sit down", "work on laptop" and "work on paper".) So it is a strong baseline for the event detection task on this dataset. [50] considers temporal dependencies and generate better results. We observe that, since [50] considers fixed order event dependencies, when there are null events happening between actual events (which is common in real-world videos), the learned dependencies may not be useful. Also, for datasets with high order dependencies, this model can be limited. On the other hand, PCIM explicitly models temporal dependencies of arbitrary order and is able to skip null events.

We can see that, by taking advantage of the complex temporal contexts, PCIM can produce the best precision by correcting wrong labels. Note that the discretization of image features and using uniform weights of different features (in K-Means clustering based dictionary learning) tend to lose some salient visual information. On dataset with larger intra-class variance and without dominating salient visual features, we expect PCIM to perform even better as visual evidences alone are less useful.

For recall, PCIM tends to omit event types with few training instances. PCIM can be treated as a data-driven approach and needs sufficient data to learn meaningful structures for each event types. For this particular dataset, events such as "pick up phone" are very scarce in training data, so PCIM is not able to learn meaningful structures for such event types. Then in testing, these events tend to be missing. However, since these are scarce events (also in testing data), missing them does not significantly affect the overall performance. When there is more data (such as videos from streaming surveillance that could be hundreds of hours long), we expect PCIM to mitigate such drawbacks as more instances are observed.

We show one example in Fig. 15 in which global temporal context among high-level events help to produce better result. In this example, SVM based approach tends to confuse the events "sit down" and "stand up" by only looking at local appearance information, because both events involve the actor performing actions close to the chair. PCIM, on the other hand, is able to get the correct result, by learning the temporal context that, a person should not sit down again after he had already sat down and been working on the laptop. Other typical cases that PCIM performs better involves differentiating between events such as "enter room" and "exit room."

3) Evaluation on PKU-MMD Dataset: The PKU-MMD dataset is a recent large-scale dataset for human action understanding in long continuous video sequences. It contains over 1000 long video sequences in 51 action categories, performed by 66 subjects in three camera views. In this work we focus on cross-subject evaluation. For fair comparison and evaluation, the same dataset partition setting as that in [28] was used, where the dataset is split into training and testing groups which consists of 57 and 9 subjects respectively (944 and 132 video samples respectively).

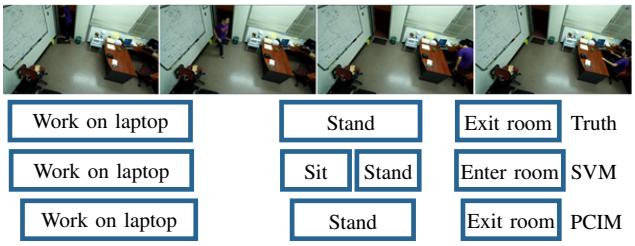

Fig. 15: Temporal context information helps to recover wrong detection by local discriminative methods. The slight time shift for PCIM is due to its continuous-time nature.

We use 20 frames as the segment length and 200 for the dictionary size (K). We compare with methods using state-of-the-art representations and temporal detection methods. Deep RGB (DR) [43] represents videos using features derived from the Temporal Segment Network (TSN). Raw skeleton (RS) is also tested in the multi-modality benchmark, but note our method only uses the videos as input. For temporal detection, we use a three stacked bidirectional LSTM (BLSTM) [49].

We report the same metrics as in [28]: $\frac{|I \cap I^*|}{|I \cup I^*|} > \theta$, where $I \cap I^*$ denotes the intersection of the predicted and ground truth intervals and $I \cup I^*$ denotes their union. Mean Average Precision (mAP) of different actions at different θ is evaluated. We directly use the evaluation tool provided in the dataset.

We report the results in Tbl. VI. Comparing with state-of-art methods that rely on more complex representations and temporal detection method, our model can produce competitive results by using RGB video input alone. Our method can produce better results at all θ levels comparing to the deep model using only RGB video input, it can also outperform the multi-modality approach (DR + RS) at higher θ levels, showing that our approach can more accurately localize events in the temporal domain. It would be interesting to extend the inference in point process for video event detection idea to multi-modality input.

VI. Conclusion

We review PCIM and how it models nonlinear dependencies in general event streams. We propose the first effective inference algorithm, ThinnedGibbs, for PCIM. Our auxiliary Gibbs sampling method effectively transforms a continuous-time problem into a discrete one. Our state-vector representation of diverging trajectories takes advantage of state merges and reduces complexity from exponential to linear for most cases. Then we show how PCIM can be used to model temporal context for event detection in video, and how event detection can be modeled as an high-level event inference problem given low-level observations. The formulation provides a generic way to model temporal context in event streams and relaxes the assumption of video pre-segmentation. It is also flexible in terms of feature extraction and dictionary learning methods.

We then validate ThinnedGibbs on several tasks, including sampling from complicated distributions with known statistics and its effectiveness in video event detection. We show our method generalizes the state-of-art inference method for

TABLE V: Comparison of the event detection result on the UCLA office dataset. Seg means the method requires video pre-segmentation. In each cell, the first number is precision and the second number is recall. See event types in Tbl. III.

Event ID	1	2	3	4	5	6	7	8	9	10	Overall
SVM	0.70/0.72	0.65/0.68	0.72/0.79	0.82/0.75	0.84/0.82	0.70/0.75	0.85/0.85	0.83/0.80	0.75/0.75	0.74/0.80	0.75/0.76
CRF[50] (seg)	0.90/0.85	0.90/0.82	0.76/0.76	0.74/0.78	0.84/0.81	0.66/0.70	0.65/0.80	0.72/0.75	0.82/0.81	0.84/0.86	0.79/0.79
Ours	0.95/0.87	0.92/0.84	0.83/0.78	0.79/0.80	0.83/0.80	0.70/0.68	0.65/0.76	0.70/0.72	0.74/0.61	0.75/0.58	0.81/0.77

TABLE VI: Mean Average Precision (mAP) comparison of the event detection result on the PKU-MMD dataset.

Method	$\theta = 0.1$	$\theta = 0.3$	$\theta = 0.5$	$\theta = 0.7$
DR + BLSTM	0.617	0.439	0.221	0.051
DR + RS + BLSTM	0.675	0.498	0.255	0.050
Ours	0.650	0.510	0.294	0.065

CTBN models. We also validate our inference idea on non-Markovian PCIMs, which is the first to do so. Then we show that the modeling of temporal context with PCIM can improve event detection performance in real-world continuous video.

REFERENCES

[1] M. R. Amer and S. Todorovic. Sum-product networks for modeling activities with stochastic structure. In *CVPR*, 2012.

[2] S. Bhattacharya, M. M. Kalayeh, R. Sukthankar, and M. Shah. Recognition of complex events: Exploiting temporal dynamic between underlying concepts. In *CVPR*, 2014.

[3] E. B. Celikkaya and C. R. Shelton. Deterministic anytime inference for stochastic continuous-time Markov processes. In *ICML*, 2014.

[4] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *CVPR*, 2009.

[6] Q. Chen, Y. Cai, L. Brown, A. Datta, Q. Fan, R. Feris, S. Yan, and S. Pankanti. Spatio-temporal Fisher vector coding for surveillance event detection. In *ACM Multimedia*, 2013.

[7] Y. Cheng, Q. Fan, S. Pankanti, and A. Choudhary. Temporal sequence modeling for video event detection. In *CVPR*, 2014.

[8] I. Cohn, T. El-Hay, R. Kupferman, and N. Friedman. Mean field variational approximation for continuous-time Bayesian networks. In *UAI*, 2009.

[9] C. Cotsaces, N. Nikolaidis, and I.Pitas. Video shot detection and condensed representation: a review. In *IEEE Signal Processing Magazine*, 2006.

[10] T. Dean and K. Kanazawa. Probabilistic temporal reasoning. In *AAAI*, 1988.

[11] N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha. Scalable influence estimation in continuous-time diffusion networks. In *NIPS*, 2013.

[12] T. El-Hay, I. Cohn, N. Friedman, and R. Kupferman. Continuous-time belief propagation. In *ICML*, 2010.

[13] Y. Fan and C. R. Shelton. Learning continuous-time social network dynamics. In *UAI*, 2009.

[14] Y. Fan, J. Xu, and C. R. Shelton. Importance sampling for continuous time Bayesian networks. *Journal of Machine Learning Research*, 11(Aug):2115–2140, 2010.

[15] A. Golightly and D. J. Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface Focus*, 2011.

[16] W. Grassmann. Transient solutions in Markovian queueing systems. *Computers & Operations Research*, 4(1):47–53, 1977.

[17] A. Gunawardana, C. Meek, and P. Xu. A model for temporal dependencies in event streams. In *NIPS*, 2011.

[18] M. Hasan and A. K. Roy-Chowdhury. Continuous learning of human activity models using deep nets. In *ECCV*, 2014.

[19] A. G. Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

[20] M. Hoai, Z.-Z. Lan, and F. D. la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011.

[21] C. Hong, J. Yu, J. Wan, D. Tao, and M. Wang. Multimodal deep autoencoder for human pose recovery. In *IEEE Transactions on Image Processing*, 2015.

[22] Y. Kim, J. Chen, M.-C. Chang, X. Wang, E. M. Provost, and S. Lyu. Modeling transition patterns between events for temporal human action segmentations and classification. In *FG*, 2015.

[23] T. Lan, Y. Zhu, A. R. Zamir, and S. Savarese. Action recognition by hierarchical mid-level action elements. In *ICCV*, 2015.

[24] I. Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005.

[25] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[26] P. Lewis and G. Shedler. Simulation of nonhomogeneous Poisson processes by thinning. *Naval Research Logistics Quarterly*, 26(3):403–413, 1979.

[27] L. Li, H. Deng, A. Dong, Y. Chang, and H. Zha. Identifying and labeling search tasks via query-based hawkes processes. In *KDD*, 2014.

[28] C. Liu, Y. Hu, Y. Li, S. Song, and J. Liu. PKU-MMD: A large scale benchmark for continuous multi-modal human action understanding. *arXiv preprint arXiv:1703.07475*, 2017.

[29] U. Nodelman, C. R. Shelton, and D. Koller. Continuous time Bayesian networks. In *UAI*, 2002.

[30] P. Over, J. Fiscus, G. Sanders, D. Joy, M. Michel, G. Awad, A. Smeaton, W. Kraaij, and G. Quénot. Trecvid 2014–an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID*, 2014.

[31] A. Parikh, A. Gunawardana, and C. Meek. Cojoint modeling of temporal dependencies in event streams. In *UAI Workshops*, 2012.

[32] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked Fisher vectors. In *ECCV*, 2014.

[33] O. P. Popoola and K. Wang. Video-based abnormal human behavior recognition : A review. In *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2012.

[34] R. Poppe. A survey on vision-based human action recognition. In *Image and Vision Computing*, 2010.

[35] Z. Qin and C. R. Shelton. Auxiliary Gibbs sampling for inference in piecewise-constant conditional intensity models. In *UAI*, 2015.

[36] S. Rajaram, T. Graepel, and R. Herbrich. Poisson-networks: A model for structured point process. In *AIStats*, 2005.

[37] V. Rao and Y. W. Teh. Fast MCMC sampling for Markov jump processes and continuous time Bayesian networks. In *UAI*, 2011.

[38] V. Rao and Y. W. Teh. Fast MCMC sampling for Markov jump processes and extensions. *Journal of Machine Learning Research*, 14:3207–3232, 2013. arXiv:1208.4818.

[39] Z. Si, M. Pei, B. Yao, and S.-C. Zhu. Unsupervised learning of event and-or grammar and semantics from video. In *ICCV*, 2011.

[40] K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012.

[41] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, 2011.

[42] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015.

[43] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Val Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[44] J. Weiss and D. Page. Forest-based point processes for event prediction from electronic health records. In *ECML-PKDD*, 2013.

[45] L. Xu, J. A. Duan, and A. Whinston. Path to purchase: A mutually exciting point process model for online advertising and conversion. In *Management Science*, 2014.

[46] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016.

[47] J. Yu, X. Yang, F. Gao, and D. Tao. Deep multimodal distance metric learning using click constraints for image ranking. In *IEEE Transactions on Cybernetics*, 2017.

[48] Y. Zhang, X. Liu, M.-C. Chang, W. Ge, and T. Chen. Spatio-temporal phrases for action recognition. In *ECCV*, 2012.

[49] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In *AAAI*, 2016.

[50] Y. Zhu, N. M. Nayak, and A. K. Roy-Chowdhury. Context-aware activity modeling using hierarchical conditional random fields. In *PAMI*, 2014.