Sequential Interval Network for Parsing Complex Structured Activity

Nam N. Vo[a], Aaron F. Bobick[a]

[a]*Georgia Tech*

Abstract

We propose a new graphical model, called Sequential Interval Network (SIN), for parsing complex structured activity whose composition can be represented as a string-length limited stochastic context-free grammar. By exploiting the grammar, the generated network captures the activity's global temporal structure while avoiding time-sliced manner model. In this network, the hidden variables are the timings of the component actions (i.e. when each action starts and ends), thus allows reasoning about duration and observation on interval/segmental level. Exact inference can be achieved and yield the posterior probabilities of the timings and of the frame's label. We demonstrate this framework on vision tasks such as recognition and temporally segmentation of action sequence, or parsing and making future prediction online when running in streaming mode.

Keywords: Activity parsing, Activity prediction, Action recognition, Stochastic context-free grammar, Sequential Interval Network

1. Introduction

For a variety of activity monitoring tasks ranging from surveillance to work-flow monitoring to quality control inspection, the challenge is to observe some complex activity being performed and to be able to label which action has been performed and often to parse or segment the input sequence into its component actions. Additionally, if a system is intended to respond appropriately and at the correct time with respect to an activity, it is necessary to perform such parsing while the activity is occurring; examples of this last task are seen in the domain of human robot interaction [1, 2].

In this paper we consider the problem of parsing complex activities where we recursively define such activities to be compositions of some combination of complex (sub-)activities and primitive actions. We presume as given the temporal structure and decomposition of the activity, such as a task plan of an assembly process where there may be partially ordered or even optional steps. We further assume that probabilistic low level visual detectors are provided or learned from training examples; these detectors provide noisy evidence as to

Email addresses: `namvo@gatech.edu` (Nam N. Vo), `afb@cc.gatech.edu` (Aaron F. Bobick)

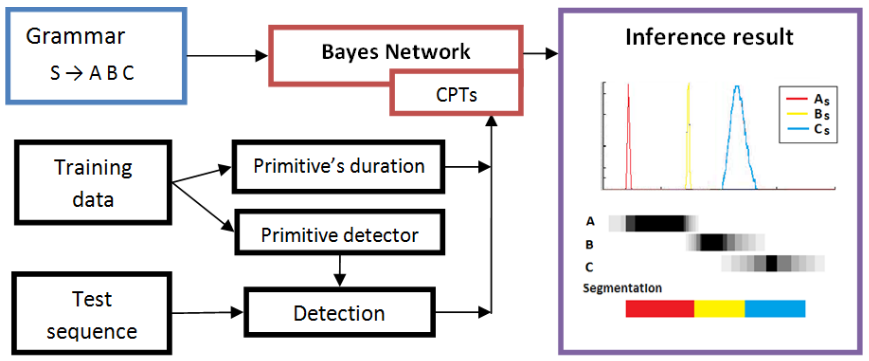


Figure 1: Overview of our framework

occurrence of some primitive action. The goal is to develop a probabilistic representation of such activities that leverage the temporal constraints and local sensing and which allows the system to assess, at any moment in time, the probability as to which actions have occurred or will occur and when.

This is an enhanced version of our previous conference paper [3]. We have added more detail and discussion, as well as additional experiments to validate the proposed method.

We organize this paper as follows. The related work is discussed in the next section. After describing the grammar representation for modeling the temporal structure of the activity in section 3, we introduce our graphical model in section 4 and the inference mechanism in section 5. In section 6 and 7, we perform different activity recognition experiments. Finally We conclude in section 8.

2. Related Works

Within computer vision community, the problem of recognizing short, low-level action has been being actively studied. Different video feature such as STIP, HOG3D, Cuboid [4], Dense Trajectories [5] were proposed and proved successful within the popular Bag-of-Word framework. Also, spatial and temporal structure can be exploited [6, 7, 8, 9] When we consider long video sequences that might contain one or multiple short action segments, this results in the challenging task of action detection: temporally segmenting the sequence or localizing the actions. Leveraging the same visual feature, different adaptations were made to handle this task. In [10, 11], SVM is used for classification and segmentation is chosen to maximize the total SVM score using dynamic programming. Pirsiavash and Ramanan [12] discover temporal pattern of the action via generating a Context-Free-Grammar; action localization is then done by a new efficient parsing technique. Most recently, Huang et al [13] proposed Sequential Max-Margin Event Detectors to handle the temporal segmentation task. This Max-Margin approach is able to scale well with respect to number of action classes.

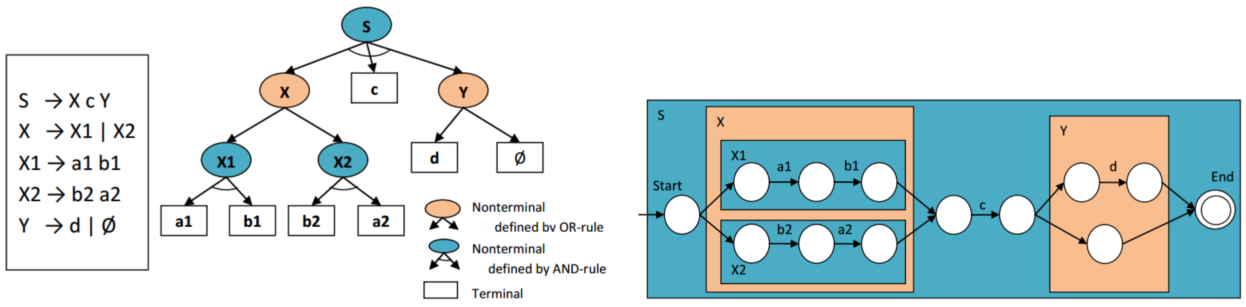
In this paper, we consider the complex activity that can be meaningfully decomposed into components, which can in turn is composed of smaller components and so on. The task is then to parse the sequence with a learnt or given temporal structure of the activity. A popular line of approaches is to first detect potential actions in a pre-processing step, then reason about their temporal composition. String and language parsing technique can be applied to such discreet set of detected actions. In [14, 15], traditional stochastic context free grammar parsing algorithm was adapted for Computer Vision problems; adjustments were made to handle different types of errors. To the same end, Damen et al [16] proposed Attribute Multiset Grammars which can encode richer constraints on activity structure. For parsing, an automatically generated Bayesian Network is used to find the detections that correspond to the best explanation. Albanese et al [17] used Probabilistic Petri Net to detect interesting human activities. In [18], Probabilistic Suffix Tree was used to learn the pattern of symbols (or "actionlets") for early detection of on-going activity. Similarly in [19, 20], AND-OR grammar is learnt from example strings of symbols, each represents an action according to the grammar's language. The learnt grammar discovers the pattern of actions and can be used for prediction. Different from these approaches, we assume detection of primitive actions is not a discrete set of events, but more general: a "heatmap" that represents the action likelihood for every interval, hence our framework can principally handle wide range of detection uncertainties.

The most related to our work are Dynamic Bayes Network (DBN) approaches, such as Hidden Markov Model (HMM), where detection of actions and their temporal composition are jointly inferred. In [21, 22], special DBNs were proposed, in which the system's state encodes when each action starts and ends. Inference by a particle filter is done in streaming mode. While the current state can be inferred, it is computationally infeasible to derive the distribution of the start and end of actions at arbitrary points in the past or future (prediction) using all available observation up till the current time. Koppula et al [23] introduce Anticipatory Temporal Conditional Random Field, which is an undirected graphical model designed to run online like a DBN. Prediction is done by extending the network into the future and perform simulation. Most recently, Wei et al [24] proposed method for modeling sequence that could incorporate rich information (duration and object/human poses) about each action. However, these approaches have to resort to approximate inference since it is infeasible to explore every possible state of every timesteps. Our framework can be considered a class of segment model and HSMM [25, 26]. By reasoning on the timings, it has the similar rich capability while permitting exact inference.

3. Grammar

First, we introduce the probabilistic context free grammar notation for modeling complex activity in this section. Note that grammar induction is not the paper's main focus, we assume that the activity's grammar is one of the inputs (however we employed simple grammar induction techniques in the experiments when the activity's grammar is not available).

The complex composite activity can be represented in hierarchical fashion: the activity consists of several actions, which can in turn consist of even smaller actions and so on. Therefore we define two types of actions: the action that is represented as a composition of other actions, and the primitive action which is explicitly modeled using learned duration model and visual detector. The whole activity is the top-level composition.



(a) (b) (c)

Figure 2: Example activity: "$S \to (ab \mid ba)\ c\ (d \mid \emptyset)$". Probabilities associated with the OR-rules were not shown in this figure. Other equivalent representations: (a) the compiled grammar, (b) the AND-OR tree, (c) Acyclic Finite State Machine

We use a stochastic grammar to model this hierarchical structure. The grammar is formally defined as G = (S, T, N, R) where: T is the set of terminal symbols, corresponding to the primitives, N is the set of non-terminal symbols, corresponding to the compositions, S is the starting symbol, corresponding to the top-level activity, R is the set of probabilistic production rules, which define the compositions. The stochastic component of the grammar is reflected in the probabilities associated with these production rules.

3.1. Compile the grammar

Before generating *SIN*, a preprocessing step is necessary to convert the original grammar to a "compiled version" that satisfies three constraints:

1. Each production rule must be either an AND-rule or an OR-rule. Mixing of AND and OR operations in one rule is not permitted. However, such rule can be trivially converted to several pure AND-rules and OR-rules. Note that since the grammar is stochastic, each symbol on the right hand side of the OR-rule is associated with a probability and they sum to 1.
2. Every symbol can only appear on the right hand side of a production rule at most once. That is every copy of a single action that appears in more than one rule must be a distinct instance. However, these instances will share detectors and duration models (described later) making the system no more difficult to implement.
3. The grammar cannot generate arbitrary long strings since our Bayes network will cover all possible sequences. This means rules causing loop such as: "$S \to SA \mid A$" are not allowed. Explicitly unrolling such loops to a maximum number of times can be done to avoid this situation.

Two of these constraints are merely syntactic and do not restrict the structure of the top level activity. They help to simplify the implementation of the graphical model generation process. The last constraint places a string-length limitation that bounds the length of time it takes to complete the activity. This is a reasonable assumption since we are always working with a finite time window. In addition, though the length is not known in advance, most activities we consider can not be arbitrary long.

An example grammar is show in Figure 2.a. The top-level activity is a partially ordered sequence of the actions a, b in any order, followed by action c, ending with an optional action d. Figure 2.b displays the AND-OR tree of the grammar.

4. Network Generation

We describe how to generate the network that will reason about timings in sequential data. The input will be the compiled grammar of the activity. First, for every symbol A in the grammar, we define the hidden random variables A_s and A_e representing the starting time and ending time of the action A, and let Z^A be the observations of the detector associated with action A; we will describe Z^A shortly. Our formulation is defined on a discrete and bounded representation of time where $1 \leq A_s \leq A_e \leq T$, where T is defined to be large enough to cover all variations in the activity's length. Depending on the course of actions as permitted by the grammar, action A may happen or it may not. We employ the special value -1 to denote the case when action A does not happen. We will use the notations $(\exists A)$ and $(!A)$ to stand for the case A happens $(1 \leq A_s \leq A_e \leq T)$ and A does not happen $(A_s = A_e = -1)$.

We now design a network that includes nodes A_s, A_e and observations Z^A for every symbol A in the grammar. The basic idea is that SIN is constructed in a hierarchical fashion, similar to the AND-OR tree (Figure 2.b). To do so, we describe how to construct it recursively for the three cases of action primitives, AND-rules, and OR-rules. We then show a recursive message passing algorithm to perform exact inference on the constructed network; the output of the inference are the posterior distributions of the start and the end of every action $P(A_s|Z), P(A_e|Z)$ including the possibility that the action does not occur $(A_s = A_e = -1)$.

Observation notation: When multiple observation are together such as Z^M, Z^N, we will group them together as $Z^{M,N}$. In addition, we will use $Z^{pre(A)}$ and $Z^{post(A)}$ to stand for the observation of all actions that happen before and after an action A, respectively.

4.1. The primitive v

The portion of the network that corresponds to a primitive v is shown in Figure 3 (a). There are two conditional probabilities required for this component:

The conditional probability $P(v_e|v_s)$: represents the prior information about the duration of action v. In our implementation we define: $P(v_e|v_s) \propto N(v_e - v_s; \mu_v, \sigma_v)$ if $v_e - v_s \geq dmin_v$, or 0 otherwise, where $N(.;.)$ is the Gaussian density function and μ_v, σ_v are parameters learned from labeled training data. Note that the Gaussian is truncated to avoid too small (or even negative) duration. For the special case when the action does not happen the duration is defined as: $P(v_e = -1|v_s = -1) = 1$.

Likelihood $P(Z^v|v_s, v_e)$: each primitive has a visual detector that outputs a detection score $F_v[\alpha, \beta]$ representing the evidence that the action starts at time α and ends at time β for every possible interval (α, β) of the range [1, T] (covering the entire activity). Then the

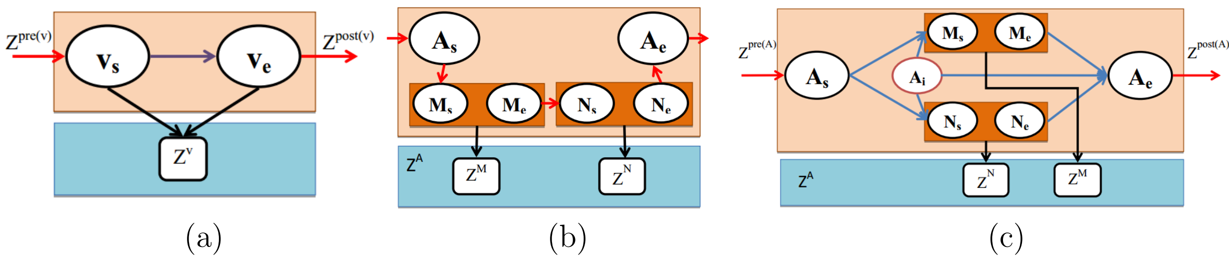


Figure 3: Structure of the network: (a) primitive v, (b) composition A defined by AND-rule A → M N , (c) composition A defined by OR-rule A → M | N

likelihood can be computed based on that detection: $P(Z^v|v_s = \alpha, v_e = \beta) = h_v F_v[\alpha, \beta]$ for some constant h_v. Calculation of F_v can be assumed to be a black box procedure.

We also need to define the likelihood for the special case $P(Z^v|v_s = -1, v_e = -1)$ which can be written as $P(Z^v|!v) = h_v F_v[-1, -1]$. We assign to $F_v[-1, -1]$ a "null value" defined as the expected detection score (Alternatively if the detector can detect if the action does not happen, it can be incorporated into this likelihood).

More detail on the design of the primitives will be discussed in section 5.4.

4.2. The composition defined by AND-rule $A \to MN$

This rule defines the action A to be the sequence of sub-action M and N (possibly more). The network is shown in Figure 3.b. Here we make some important assumptions: (1) the start and end of the composition are the start of the first action and the end of the last action in the sequence respectively ($A_s = M_s, A_e = N_e$), (2) the end of one action is equal the start of the next action in the sequence ($N_s = M_e$),[1] (3) the observation of the action consists of all observations of its sub-actions $Z^A = Z^{M,N}$

4.3. The OR-rule composition $A \to M \mid N$

The OR-rule defines a composite action A to be either M ($\exists M$ and $!N$, which means $A_s = M_s, A_e = M_e, N_s = N_e = -1$) or N ($\exists N$ and $!M$, which means $A_s = N_s, A_e = N_e, M_s = M_e = -1$), more accurately it is an exclusive OR operation. Figure 3.c shows the OR network structure.

The standard approach to realizing this "OR" condition in a Bayes network is to use the multiplexer CPT, with a "selector" variable [27] which we write as A_i in our network. $A_i \in \{M, N\}$ indicate which case it is ($\exists M$ or $\exists N$). The prior probability $P(A_i|\exists A)$, or equivalently $P(\exists M|\exists A)$ and $P(\exists N|\exists A)$, is extracted from the grammar's production rule. Note that it can be manually defined or learned from training data (usually we will choose $P(A_i = M|\exists A) = P(A_i = N|\exists A) = 0.5$, unless otherwise stated).

Implementation trick: for every composition A (both AND and OR), we can define $P(Z^A|!A) = \prod_{\text{M in A}} P(Z^M|!M)$. Note that scaling the likelihood $P(Z^v|v_s, v_e)$ does not affect the final result. Therefore in the implementation we could choose the scaling such

[1]If time between actions is needed we can insert a special DUMMY action between them.

that $h_v F_v[-1,-1] = 1$ for every primitive v, then we can safely ignore the factors $P(Z^A|!A)$ for every A.

5. Inference

5.1. Exact inference by forward backward message passing

Input: Beside the CPT $P(v_e|v_s)$ and $P(Z^v|v_s, v_e)$ for every primitive v (described in section 4.1), we need 3 more inputs: (1) The prior $P(\exists S)$, (2) $P(S_s|\exists S)$: the constraint about the start, and (3) $P(Z^{end}|S_e, \exists S)$: the constraint about the end. We set $P(\exists S) = 1$ to make following formulation simple (though rule such as "$S \to A \mid \emptyset$" can be used to emulate the case where the activity does not happen all together). For the task of activity segmentation, we can have: the start is the first time step/frame and the end is the last time step/frame of the test sequence (experiment in section 7). On the other hand, we can assume uniform distributions about the start and the end of the activity (experiment in section 6). In that case, our framework effectively performs detection and parsing at the same time.

The algorihtm: has 4 main steps as following, we refer the readers to the appendix section for more detailed description:

1. Forward phase: Starting from $P(S_s|\exists S)$, the propagation is performed from the high level actions to their subactions recursively, from the start of the action to the end, integrating all observations in the process. The output is $P(A_s, Z^{pre(A)}|\exists A)$, $P(A_e, Z^{pre(A),A}|\exists A)$ for every action A.
2. Backward phase: similarly, starting from $P(Z^{end}|S_e, \exists S)$, we compute $P(Z^{post(A)}|A_e, \exists A)$ and $P(Z^{A,post(A)}|A_s, \exists A)$ for every action A (propagation in the opposite direction to the first step).
3. Compute the posteriors: by multiplying forward and backward messages, we get $P(A_s, Z|\exists A)$, $P(A_e, Z|\exists A)$ for every action A. Normalization is performed to get $P(A_s|Z, \exists A)$, $P(A_e|Z, \exists A)$. The probability of observation $P(Z)$ can also be obtained.
4. Compute the happening probability: we find $P(\exists A|Z)$ for every action A.

Output: the probability of action A happening $P(\exists A|Z)$, and if that is the case, the distribution of the start and the end $P(A_s|Z, \exists A)$, $P(A_e|Z, \exists A)$, or even the joint of them if A is a primitive (optionally one can choose to compute the joint posterior of the start and the end of every action, though this will significantly increase the process time).

Computational complexity: The inference complexity is linear in number of nodes (number of actions in the grammar: K) and the size of CPT (T^2), hence it is $O(K.T^2)$.

5.2. Interpreting the result for recognition, detection/prediction and segmentation

First if a symbol A is on the right hand side of an OR-rule, then $P(\exists A|Z)$ is the posterior probability associated with that OR-rule. Hence we can do action recognition and infer the most probable course of actions.

Secondly we can compute $P(A_s|Z)$, $P(A_e|Z)$:

$$P(A_s|Z) = P(\exists A|Z) P(A_s|Z, \exists A) \tag{1}$$

for values between 1 and T (note that $P(A_s = -1|Z) = P(!A|Z) = 1 - P(\exists A|Z))$. These distributions are shown in the experiment in section 5. Using these distributions, prediction of when an action starts or ends can be made by picking the expected value, or the value that maximize the posterior. Even better, one could consume this whole distribution to account for the inferred uncertainty depending on specific application.

These results can be mapped back to the original grammar: to compute the distribution of actions' timing in the original grammar, one can combine the distributions of separate actions in the compiled version corresponding to the same action in the original version.

For the task of labeling frames, the probability of a time step t having the label of primitive v can be computed easily:

$$P(label_t = v|Z) = \sum_{\alpha=1}^{t} \sum_{\beta=t}^{T} P(v_s = \alpha, v_e = \beta|Z) \tag{2}$$

We obtain the distribution of the label of time step t. If A is a composition, $P(label_t = A|Z)$ can be found be summing over all its subactions. Therefore temporal segmentation can be done in any hierarchy level by choose the label that maximize the probability. We perform activity segmentation in term of primitives in the experiments to demonstrate this feature. Even though this approach maximizes the expected frame classification accuracy, the result labeling might not represent a valid sequence of actions (this phenomenon also presents in methods such as HMM). An alternative is to derive the parsing with highest probability (the most probable course of actions), estimates the start and the end of the actions in that sequence, and then labels the frames.

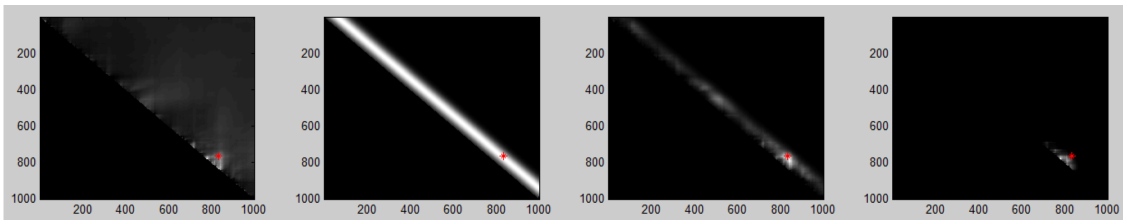
5.3. Parsing in streaming mode and future prediction

Our method is nativey an offline process (i.e. it is assumed that the entire video sequence is given), however we can addapt it to run in streaming mode if desired. This can be done by constructing the entire network at the beginning, with all likelihoods initialized using the expected detection score (the "null value" $F_v[-1, -1]$). As new observations are obtained, likelihoods are recomputed and the inference process is re-performed. Note that each inference is independent of each other, our graphical model is not running online like a DBN. This strategy is also easily applied to partially observed sequence: if calculation of $F_v[\alpha, \beta]$ for some α, β cannot be done due to observation needed not available, they will keep the prior value $F_v[-1, -1]$.

5.4. Design of the primitives detectors

The primitive actions are the basic building blocks of the activity. The primitive detectors are the only component that processes the test sequence and it is particularly important. Not only does it affect the action localization result, it impacts the OR-rule situations. For example given $A \to M|N$, a strong detection of subactions in M can make $P(\exists M|Z)$ higher, while diminishing $P(\exists N|Z)$ at the same time.

We assume this procedure is a black-box so that making use of different kinds of detectors is possible. For example it can be driven by explicitly detecting the start and the end of



(a) (b) (c) (d)

Figure 4: Example of one action in the CMU activity experiment: (a) the detection score matrix corresponding to the likelihood $P(Z^v|v_s, v_e)$, the vertical dimension is the start time, the horizontal dimension is the end time (b) the duration factor corresponding to the conditional probability $P(v_e|v_s)$, the duration has mean of 59 and variance of 33, (c) $P(v_e, Z^v|v_s)$: the product of the 2 previous factor, and (d) the joint posterior $P(v_s, v_e|Z)$: the output of our framework. The red mark is the groundtruth: the action starts at timestep 764 and ends at timestep 835. Note that all these matrices are upper triangular.

the action (experiment in section 6 and 6.2). An alternative way is to perform sliding-window-detection using statistics/features computed over the $[\alpha, \beta]$ segment (experiment in section 7.1 and 7.3). Ideally if the action starts at α_0 and ends at β_0 then $F_v[\alpha_0, \beta_0]$ would be high and $F_v[\alpha, \beta]$ would be low for every other α, β value. Note that the calculation $P(Z^v|v_s = \alpha, v_e = \beta) \propto F_v[\alpha, \beta]$ can leverage all the observation data available, not just the segment $[\alpha, \beta]$.

If it is a likelihood based detector, then the score can be used directly. If it is a discriminative method, then a post-processing step to calibrate the score is needed (because each detector might output different kinds of confidence scores). For example one can normalize the SVM-score and apply a sigmoid function, or apply a exponential function to the negative of the energy-score in order to obtain a score that better indicates the likelihood. The likelihood value 0 is usually discouraged as it could nullify the likelihood of other actions.

As the factor $P(v_e|v_s)$ accounts for the duration of the action and the factors $P(Z^v|v_s, v_e)$ accounts for the visual information of the action, the visual detector do not need to concern about duration. Although one could derive a detector like that (i.e. combine the 2 factors into 1 single factor $P(v_e, Z^v|v_s)$), we find that keeping these 2 factors separate is more flexible. In figure 4, we show some examples of the input factors ($P(v_e|v_s)$ and $P(Z^v|v_s, v_e)$) and the output ($P(v_s, v_e|Z)$) of framework.

The special value $F_v[-1, -1]$: This value represents how likely the action v does not happen. Common detection strategy with a classifier is to perform non-maxima suppression and thresholding on the heatmap F_v to obtain a set of detections. The $F_v[-1, -1]$ plays similar role as the threshold: a high value means the action more likely does not happen. Informally $F_v[\alpha, \beta]/F_v[-1, -1] > 1$ means segment $[\alpha, \beta]$ is a positive and the confidence is proportional with that ratio value. Where $F_v[\alpha, \beta]/F_v[-1, -1] = 1$ basically means "nothing is known about $[\alpha, \beta]$" (we made use of this in streaming mode). In our implementation, we choose $F_v[-1, -1]$ to be the expected detection score so that it has above properties.

Intuitively, $F_v[\alpha, \beta]$ and $F_v[-1, -1]$ put relative weights on the probability of sequence

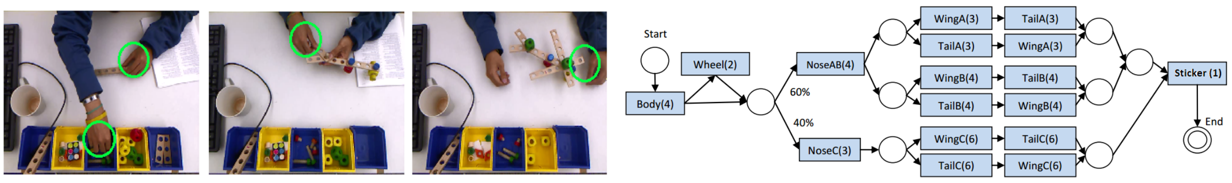


Figure 5: The toy assembly task. (a) Example input frames and hand detection result. (b) The temporal structure in the form of state machine. Each box is an AND composition of a number of primitives (shown in the bracket). The subject first assembles the body part; follow by the optional wheel part and one of the two nose parts. Then the wing part and tail part can be done in any order. Finally the subject puts the sticker on the model. There are 40 different primitive actions and 12 complete task variations.

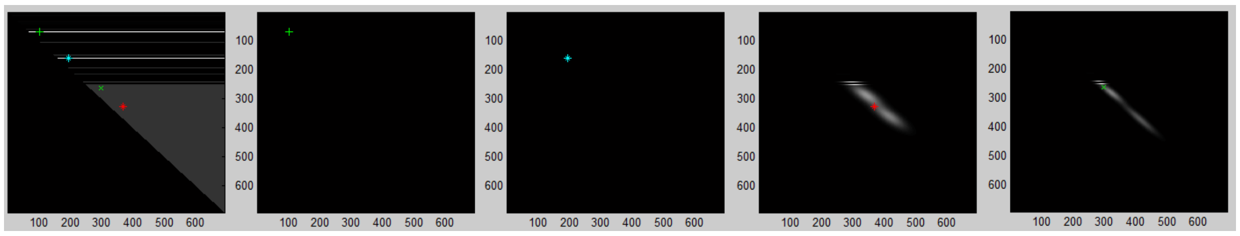


Figure 6: The toy assembly experiment: on the left is the primitive detection result of actions involving bin 4 (the vertical dimension is the start time, the horizontal dimension is the end time). There are 4 such actions, the markers are the groundtruth. On the right are the output joint posteriors of each of them. The system is running in streaming mode and the current timestep in this example is 250 (observation after this point is not yet available). The first 2 actions are detected, hence perfectly localized (coinciding with the markers). The system outputs a rough prediction of 2 later actions.

where the action happens and the sequences where it does not, respectively, when we are considering the OR-rule. For example if $F_v[\alpha, \beta]/F_v[-1, -1]$ is very big for some value α, β and these values are also temporally consistent with overall activity's structure, this would contribute to increase the posterior probability of the sequence where action v does happen. Note that: (1) scaling F_v won't change above ratio, hence won't change the inference result, (2) if the grammar does not have any OR-rules, then the value $F_v[-1, -1]$ does not matter.

6. Toy Assembly Task Experiment

To demonstrate the capability of SIN, we designed a simple toy assembly task, where the human operator takes wooden pieces provided in 5 different bins in the workspace and puts them together to construct an airplane model. The task's structure is shown in Figure 5 and the subjects will follow this recipe. Our dataset consists of 29 sequences; each one is about 2-3 minutes long. Results are generated by performing 20 trials where on each trial 3 sequences (1 of each airplane model A, B and C) are randomly selected for testing and the remaining are for training.

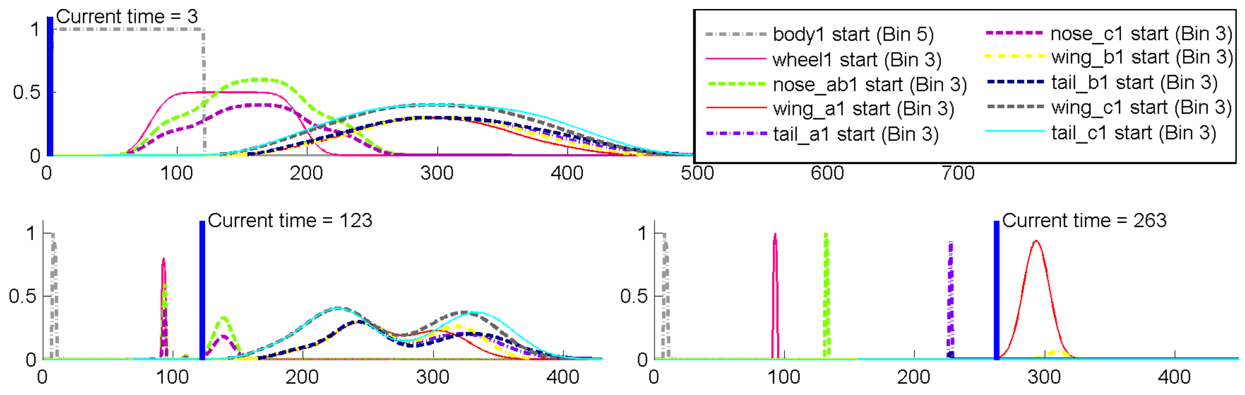


Figure 7: (best view in color) Toy assembly experiment: example posterior distributions of the starts of some actions (first primitive of each part) at 3 different time steps (computed using Eq.1, note that the special value (-1) is not shown). In order to observe them easily, each distributions is scaled so that its peak have the same value as the probability that action would happen (e.g. for *body*1, it is 1; for *nose_ab*1, it's 0.6 in the first timestep and about 0.99 in the last timestep; for *nose_c*1, it's 0.4 and then 0.01)

Each primitive action is defined as getting a piece from a bin and assembling it. The start of the action is defined as when the hand reaches the piece. In order to detect such actions, first we implement a simple color blob detector to detect the hand positions in the input frame (note that its performance is not extremely accurate: it fails to to detect the correct hands roughly 30% of the time). Then we can compute $F_v[\alpha, \beta] \propto N(H_\alpha; \mu_v, \sigma_v) + u_v$, where H_t is the position of the hands at frame t, N(.;.) is the Gaussian density function and parameters μ_v, σ_v are learned, and u_v is a small uniform component representing the likelihood in case the hand detector fails. Notice that: (1) in this case the action detectors reduce to special case: event detectors of the actions' starts; (2) actions corresponding to different pieces in the same bin will end up having similar detectors (our method naturally resolves this ambiguity). One example of detection result is shown in figure 6.

Qualitative Result: in Figure 7, some example posterior distribution outputs when running our method on a sequence in streaming mode are shown (we encourage readers to watch the supplementary video [3]). At first no observation is available; the distributions are determined by the prior information about the start of the task (which we set to be a uniform in first 30s) and duration models of primitives. In the second plot, some first actions (Body and Nose) are detected; however it is still not clear which model is being done, hence the distributions of all possible actions overlap both in the past and future. Finally, these uncertainties are resolved when the subject is about to finish TailA part. It is recognized that model A is being assembled and the next actions going to be WingA; distributions of model B and C's actions have all diminished. As we can see as time progresses: (1) ambiguities both in the past and future are resolved, (2) the distributions get tighter, hence the prediction of when the actions are going to happen becomes more certain. The system can also output the joint posterior of the start and end of each action, one example is shown in figure 6.

In figure 8, we show the final result on 1 test sequence, where all actions that happened are

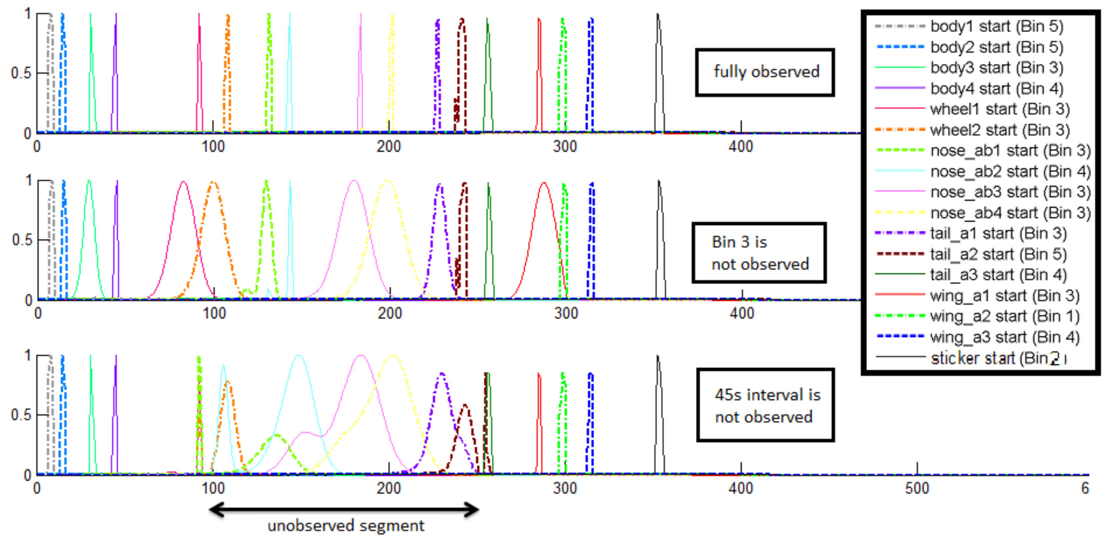


Figure 8: Toy assembly experiment: the parsing result of 1 test sequence in 3 cases: fully observed, bin 3 is not observed and a 45s interval is not observed. Here we only show the distributions of the starts of all actions that actually happen.

identified and localized. To demonstrate that the system can handle noise and uncertainty, we show the result of the same sequence in 2 other cases: one where we completely disable the detectors of all actions involving bin 3, the other we disable all sensing during a 45s interval. In both cases, the system is still able to correctly recognize the sequence of actions and roughly estimate the timings of unobserved actions, thanks to known temporal constraints.

Quantitative Result: we can use the mean of the distributions as the timing estimation, and then the event localization error can be defined as the difference between this estimation and the true timing. Figure 9 shows how the result changes as more observation is available: the classification of the model being assembled (A, B or C) gets better, the average localization error of every actions' start time decreases, and the entropy of those distributions (representing the uncertainty) decreases. When the whole sequence has been seen, the average localization error is less than 1 second. We also performed segmentation in offline mode (all observation is available) and the accuracy is 91.8%.

Note that by taking the mean of the distributions, we are making a guess that minimizes the expected squared distance error. On the other hand, one can integrate a custom cost function over the distribution in order to make a prediction that minimizes the expected cost. Earlier version of the framework was used in a Human Robot Collaboration experiment [28, 29] where the inference runs online in real-time. The robot was required to *anticipate* when a human would need a given part so that it could make a plan as to when to fetch bins and when to remove them. The goal was to create a plan that minimizes the human operator's idle time; in that situation the robot considered the entire timing distribution not just the mean.

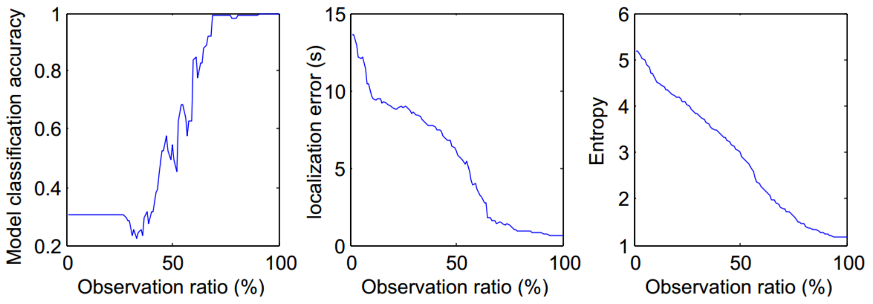


Figure 9: Result on our toy assembly dataset: (a) accuracy of model classification, (b) average localization error and (c) entropy of all actions' start

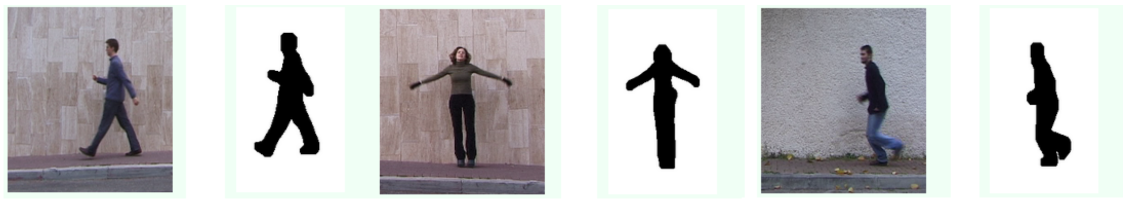


Figure 10: Example frames and corresponding foreground masks from the Weizmann action dataset

7. Recognition and Segmentation Experiments

We conducted temporal segmentation experiments on 3 different activity datasets. To apply our method, 2 components must be defined: the grammar for the activity and the visual detectors for the primitive actions. In the following sectons, we describe each experiment in detail.

7.1. Weizmann dataset

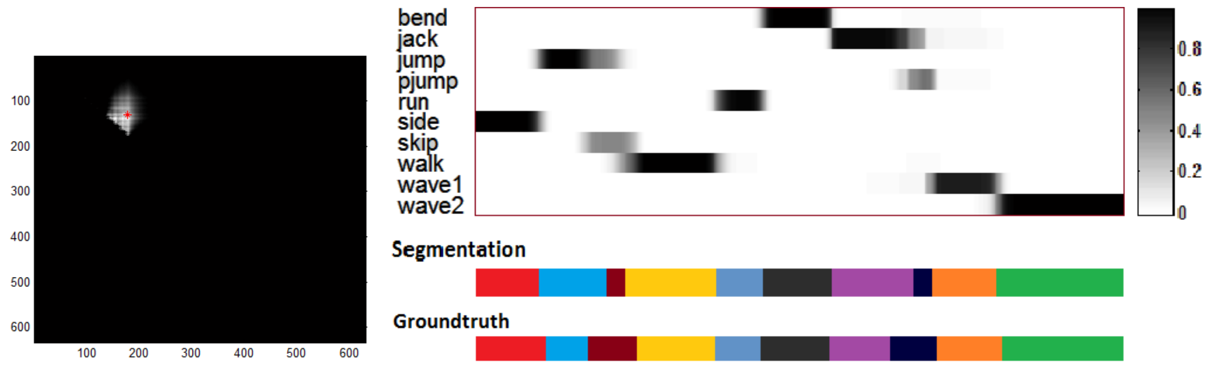
This dataset has 93 sequences of 10 actions: walk, run, jump, side, bend, wave1, wave2, jump, jumpjack, skip (figure 10); we will consider these to be primitives [30]. To create composite activities, we concatenate 10 different actions to create long sequences in the manner of [11]. We randomly choose 1 long sequence for testing and the remaining for training.

We firt describe how we implement the primitive detector: like [11] we extract the foreground mask for each frame, compute the distance transform, and then perform k-mean clustering with k = 100 to form a codebook of visual words. For every possible interval $[\alpha, \beta]$ of the test sequence, we compute the normalized histogram of visual words $h[\alpha, \beta]$. To detect a primitive v, we compute the distance score $d_v[\alpha, \beta]$ as the χ^2 distance to the nearest neighbor (in the set of positive training examples). Finally we define the similarity score $F_v[\alpha, \beta] \propto (max(10^{-5}, 1 - d_v[\alpha, \beta]))^2$. One detection example is shown in figure 11 (a).

Next we manually define the grammar assuming that the activity is a sequence of 30 unknown actions, each of which is either one of the 10 primitives or empty:

$S \rightarrow AAAAAAAAAAAAAAAAAAAAAA...$

$A \rightarrow walk \mid run \mid jump \mid side \mid bend \mid wave1 \mid wave2 \mid jump \mid jumpjack \mid skip \mid \emptyset$



(a) (b)

Figure 11: Weizmann dataset experiment: (a) Example detection score matrix F_{run} of action [run] on one test sequence (the vertical dimension is the start time, the horizontal dimension is the end time). The red mark is the truth (the action starts at timestep 130 and ends at timestep 179) (b) Example Segmentation result of 1 test sequence. On the top, the probability of each frame's label (corresponding to each column), computed using Eq.2, is shown. Segmentation is done by choosing the label with highest probability. The groundtruth is shown at the bottom.

Figure 12: Example frames from the GTEA dataset

This grammar covers a wide range of different sequences including the true sequence of 10 actions. Note that further increasing the maximum number of actions will not affect the segmentation result because the primitives' duration models would eliminate the possibilities where each action is too short or too long.

The task is to recognize and segment at the same time, as in [11]. Our segmentation accuracy is 93.6%, while state-of-the-art result is 69.7% and 87.7% by discriminative methods in [10] and [11]. We can assert the confidence about each's frame label using its distribution, one example shown in figure 11 (b).

7.2. *GeorgiaTech Egocentric Activity dataset*

The GeorgiaTech Egocentric Activity dataset (GTEA) [31, 32] consists of 7 high level activities such as making a cheese sandwich or making coffee (figure 12); each action is performed by 4 subjects. There are 61 primitives (such as take spoon, take cup, open sugar, scoop sugar spoon, open water, pour water, etc). Following [32], 16 sequences are used for training and 7 for testing.

For detection, we obtained the beginning state detection scores S_B and ending state detection scores S_E of every primitive from the author [32] (the classification accuracy is 39.7%). Since these raw scores are not a good indicator of the likelihood, we define our

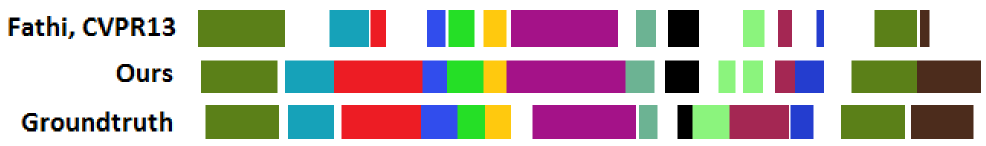


Figure 13: Example Segmentation result on GTEA of the activity: making Cheese Sandwich.

Figure 14: Example frames from CMU Multi-Modal Activity Database

detection score of a primitive v as $F_v[\alpha, \beta] \propto (S_B[v, \alpha]S_E[v, \beta])^2$ to magnify the difference between positive and negative detections. We also test a 2nd setting, where we use $F_v[\alpha, \beta] \propto (S_B[v, \alpha]S_E[v, \beta])^{10}$.

The grammar is very important and design of a good grammar is not trivial. We derive our grammar using training sequences in a very simple way:

$S \rightarrow Activity1 \mid Activity2 \mid ...$

$Activity1 \rightarrow Sequence1 \mid Sequence2 | ...$

$Sequence1 \rightarrow p\_action1\ p\_action2\ p\_action3 ...$

...

This exemplar-like approach effectively matches the testing sequence with all the training data to find similar sequences (even though they are not exactly the same). Note that this grammar does not generate the test sequences.

Our segmentation accuracy is 51% in the first detection score and 58% in the 2nd setting, compare with [32]'s 42% and [31]'s 33%. One example result is shown in Figure 13.

Unlike [32], our method models the global structure of the activity and is able to natively classify high level activity using posterior probabilities associated with the first OR-rule. In this experiment, our method correctly classifies the high level activity label 6 out of 7 test sequences.

7.3. CMU Multi-Modal Activity Database

This dataset has different kitchen activities performed by several subjects [33]. A number of sensor data recorded during the activity is given, we will only use the egocentric video data (figure 14). In this experiment, we work on 13 sequences of [making brownie] activity, where the action label are available [34]. The task is also performing temporal segmentation, we do it in leave-one-out cross-validation setting and report the average segmentation accuracy across 13 runs. There are 43 primitive actions; each sequence is 2 minutes long and has 60 actions on average.

First the grammar is needed. Since the activity is much longer (in term of string length) and the number of training sequences is limited, the simple technique in the GTEA experiment is not adequate to model the temporal structure's variation. Therefore we derive a slightly different strategy. For each training example, we break it into N parts of equal length. The activity is then defined as a sequence of N parts, each part can be taken from the pool of training examples:

$S \rightarrow Part1\ Part2\ Part3...$

$Part1 \rightarrow Sequence1Part1 \mid Sequence2Part1|...$

$Part2 \rightarrow Sequence1Part2 \mid Sequence2Part2|...$

$Sequence1Part1 \rightarrow p\_action1\ p\_action2\ p\_action3...$

...

This grammar cover a lot of variations including all training example sequences. Note that (1) not all sequence generated by the grammar is valid, and (2) similar to the GTEA experiment, the grammar likely does not generate the test sequence, however it might generate something close (in term of string distance).

For the primitive action detectors, we use Dense Trajectory [5] as feature. A codebook is learnt and for each video segment, a histogram of visual words is computed. Then detection scores are computed based on the χ^2 distance to positive examples similar to experiment in section 7.1. However, we perform 2 additional steps: (1) raise the score to the power of M = 10, 50, 100, 300 , and (2) divide the detection score by the sum of score of all detectors. We found that this normalization step helps to make the detectors more discriminative. Figure 4 shows an example.

The segmentation result is shown in table 1; one example result of the frame label's posteriors is shown in figure 15. Previous work [34] on the same task achieved 32%; though it is not directly comparable since we consider all 43 primitive actions provided in the annotation, while [34] classified 35 actions. First observe that big value of parameter M improves the performance to a certain extend. Secondly, increasing N, which means increasing the grammar's variation, helps even though this means the grammar is becoming more ambiguous. Lastly, even with the groundtruth grammar (i.e. the test sequence is used to "train" the grammar), the performance is only at 70%, hinting that significant improvement requires better primitive detectors.

	M = 10	M = 50	M = 100	M = 300
Grammar N = 1 part	41.36	49.14	48.51	49.06
Grammar N = 4 parts	43.17	52.82	53.79	53.14
Grammar N = 8 parts	45.41	54.82	55.58	55.05
Grammar N = 16 parts	45.88	57.35	59.13	57.51
Grammar N = 32 parts	44.64	57.41	58.91	**60.21**
Grammar Truth	59.89	70.03	70.04	70.18

Table 1: Segmentation result on the CMU activity dataset.

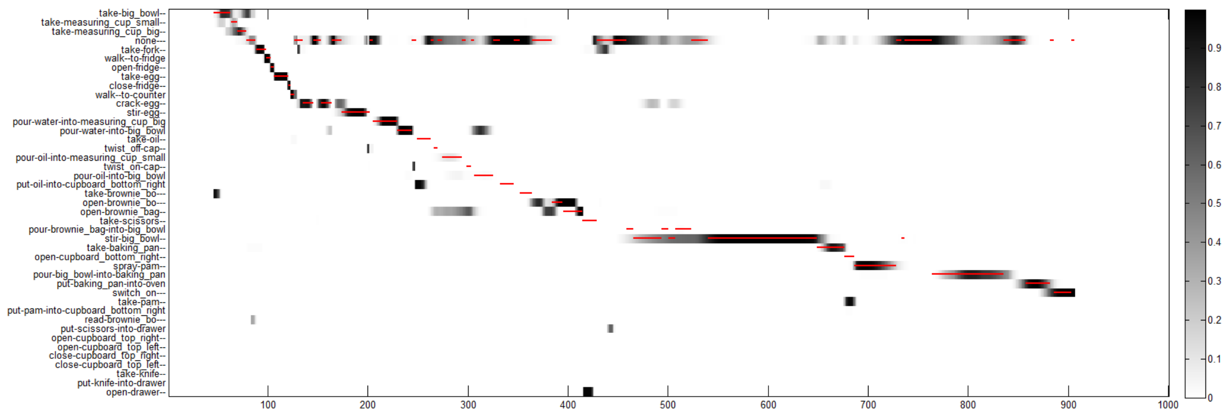


Figure 15: Example result of the frame label posterior from 1 sequence in CMU dataset experiment, red lines mark the groundtruth label.

8. Conclusion

In this paper, we introduced *Sequential Interval Network*, a novel graphical model for parsing complex structured activity sequence (or time series data in general). First, the temporal structure of the activity is presented in form of the stochastic context free grammar, which allows multiple levels of composition using the AND-rule and OR-rule. Next, the network is generated, where the hidden variables are the timings of the action components, allowing reasoning on interval/segmental level. Finally, we proposed an exact inference algorithm to compute the posteriors of the actions' timings and the frames' label.

We applied the framework for future prediction, recognizing and temporally segmenting the sequence of actions. Our method can handle noise and uncertainty in a principle and probabilistic manner. In such cases, it performance degrades gracefully, as demonstrated in the experiments where some detectors are disabled, or where a non-perfect grammar is used. In order to better understand and improve, future work is to further study and extend the grammar formulation and the primitive detectors.

Acknowledgement

We'd like to thank Alireza Fathi for providing data on GTEA dataset. This work was supported in part by BMW project #RD441

References

[1] G. Hoffman, C. Breazeal, Cost-Based Anticipatory Action Selection for HumanRobot Fluency, IEEE Transactions on Robotics 23 (5) (2007) 952–961. doi:10.1109/TRO.2007.907483.
URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4339531`

[2] M. Huber, A. Knoll, When to assist?-Modelling human behaviour for hybrid assembly systems, in: Robotics (ISR), 2010.
URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5756792`

[3] N. N. Vo, A. F. Bobick, From stochastic grammar to bayes network: Probabilistic parsing of complex activity, in: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE, 2014, pp. 2641–2648.

[4] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, C. Schmid, et al., Evaluation of local spatio-temporal features for action recognition, in: BMVC, 2009.

[5] H. Wang, A. Klaser, C. Schmid, C.-L. Liu, Action recognition by dense trajectories, in: CVPR, 2011.

[6] K. Tang, L. Fei-Fei, D. Koller, Learning latent temporal structure for complex event detection, in: CVPR, 2012.

[7] J. C. Niebles, C.-W. Chen, L. Fei-Fei, Modeling temporal structure of decomposable motion segments for activity classification, in: ECCV, 2010.

[8] M. R. Amer, S. Todorovic, Sum-product networks for modeling activities with stochastic structure, in: CVPR, 2012.

[9] X. Yang, Y. Tian, Action recognition using super sparse coding vector with spatio-temporal awareness, in: ECCV, 2014.

[10] Q. Shi, L. Wang, L. Cheng, A. Smola, Discriminative human action segmentation and recognition using semi-markov model, in: CVPR, 2008.

[11] M. Hoai, Z.-Z. Lan, F. De la Torre, Joint segmentation and classification of human actions in video, in: CVPR, 2011.

[12] H. Pirsiavash, D. Ramanan, Parsing videos of actions with segmental grammars, 2014.

[13] D. Huang, Y. Wang, S. Yao, F. De la Torre, Sequential max-margin event detectors, in: European Conference on Computer Vision (ECCV), 2014.

[14] Y. A. Ivanov, A. F. Bobick, Recognition of visual activities and interactions by stochastic parsing, Pattern Analysis and Machine Intelligence, IEEE Transactions on.

[15] D. Moore, I. Essa, Recognizing multitasked activities from video using stochastic context-free grammar, in: AAAI/IAAI, 2002, pp. 770–776.

[16] D. Damen, D. Hogg, Explaining activities as consistent groups of events, International journal of computer vision.

[17] M. Albanese, R. Chellappa, V. Moscato, A. Picariello, V. Subrahmanian, P. Turaga, O. Udrea, A constrained probabilistic petri net framework for human activity detection in video, Multimedia, IEEE Transactions on.

[18] K. Li, J. Hu, Y. Fu, Modeling complex temporal composition of actionlets for activity prediction, in: ECCV, 2012.

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

[20] M. Pei, Y. Jia, S.-C. Zhu, Parsing video events with goal inference and intent prediction, in: Computer vision (iccv), 2011 ieee international conference on, IEEE, 2011, pp. 487–494.

[21] Y. Shi, Y. Huang, D. Minnen, A. Bobick, I. Essa, Propagation networks for recognition of partially ordered sequential action, in: CVPR, 2004.

[22] B. Laxton, J. Lim, D. Kriegman, Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video, in: CVPR, 2007.

[23] H. S. Koppula, A. Saxena, Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation, ICML, 2013.

[24] P. Wei, Y. Zhao, N. Zheng, S.-C. Zhu, Modeling 4d human-object interactions for event and object recognition, in: ICCV, 2013.

[25] K. P. Murphy, Hidden semi-markov models (hsmms), unpublished notes.

[26] S.-Z. Yu, Hidden semi-markov models, Artificial Intelligence.

[27] D. Kollar, N. Friedman, Probabilistic graphical models: principles and techniques, The MIT Press, 2009.

[28] K. P. Hawkins, N. Vo, S. Bansal, A. Bobick, Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration, in: Proceedings of the IEEE-RAS International Conference on Humanoid Robots, 2013.

[29] K. P. Hawkins, S. Bansal, N. N. Vo, A. F. Bobick, Anticipating human actions for collaboration in the presence of task and sensor uncertainty, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, 2014.

[30] M. Blank, L. Gorelick, E. Shechtman, M. Irani, R. Basri, Actions as space-time shapes, in: ICCV, 2005.

[31] A. Fathi, A. Farhadi, J. M. Rehg, Understanding egocentric activities, in: ICCV, 2011.

[32] A. Fathi, J. M. Rehg, Modeling actions through state changes, in: CVPR, 2013.

[33] F. De la Torre, J. K. Hodgins, J. Montano, S. Valcarcel, Detailed human data acquisition of kitchen activities: the cmu-multimodal activity database (cmu-mmac), in: Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research, in conjuction with CHI 2009, 2009.

[34] E. H. S. Taralova, F. De la Torre, M. Hebert, Temporal segmentation and activity classification from first-person sensing, in: IEEE Workshop on Egocentric Vision, in conjunction with CVPR 2009, 2009.

Appendix A. SIN inference

Step 0: we apply the implementation trick explained in section 4.3 to safely ignore the factors $P(Z^A|!A)$ for every A.

Step 1 - Forward phase: Starting from $P(S_s|\exists S)$, the propagation is performed from the high level actions to their subactions recursively, from the start of the action to the end, integrating all observations in the process. The output is $P(A_s, Z^{pre(A)}|\exists A)$, $P(A_e, Z^{pre(A),A}|\exists A)$ for every action A.

For primitive v, given $P(v_s, Z^{pre(v)}|\exists v)$: we can multiply it with the duration factor $P(v_e|v_s)$ and visual observation factor $P(Z^v|v_s, v_e)$ to get $P(v_s, v_e, Z^{pre(v),v}|\exists v)$. Then marginalization can be done to get $P(v_e, Z^{pre(v),v}|\exists v)$.

$$P(v_s, v_e, Z^{pre(v),v}|\exists v) = P(v_s, Z^{pre(v)}|\exists v)P(v_e|v_s)P(Z^v|v_s, v_e) \tag{A.1}$$

$$P(v_e, Z^{pre(v),v}|\exists v) = \sum_{t=1}^{T} P(v_s = t, v_e, Z^{pre(v),v}|\exists v) \tag{A.2}$$

For AND-rule $A \to MN$: given $P(A_s, Z^{pre(A)}|\exists A)$, the variable M_s has the same distribution:

$$P(M_s = t, Z^{pre(M)}|\exists M) = P(A_s = t, Z^{pre(A)}|\exists A) \tag{A.3}$$

Recursively perform forward phase on M to get $P(M_e, Z^{pre(M),M}|\exists M)$. Next the variable N_s has the same distribution:

$$P(N_s = t, Z^{pre(N)}|\exists N) = P(M_e = t, Z^{pre(M),M}|\exists M) \tag{A.4}$$

So we can perform forward phase on N to get $P(N_e, Z^{pre(N),N}|\exists N)$. The final result is the same as distribution $P(A_e, Z^{pre(A),A}|\exists A)$:

$$P(A_e = t, Z^{pre(A),A}|\exists A) = P(N_e = t, Z^{pre(N),N}|\exists N) \tag{A.5}$$

For OR-rule $A \to M \mid N$: given $P(A_s, Z^{pre(A)}|\exists A)$, we can represent the distribution of A_e in term of M_e and N_e. For example if A is M, then the distribution $P(M_s, Z^{pre(M)}|\exists M)$ is the same as A_s:

$$P(M_s = t, Z^{pre(M)}|\exists M) = P(A_s = t, Z^{pre(A)}|\exists A) \tag{A.6}$$

We can therefore perform forward phase on M to get $P(M_e, Z^{pre(M),M}|\exists M)$. Similarly we can find $P(N_e, Z^{pre(N),N}|\exists N)$. Then distribution of A_e is:

$$\begin{aligned}P(A_e = t, Z^{pre(A),A}|\exists A) = P(\exists M|\exists A)P(Z^N|!N)P(M_e = t, Z^{pre(M),M}|\exists M)\\ +P(\exists N|\exists A)P(Z^M|!M)P(N_e = t, Z^{pre(N),N}|\exists N)\end{aligned}$$

Step 2 - Backward phase: Similarly, starting from $P(Z^{end}|S_e, \exists S)$, we compute $P(Z^{post(A)}|A_e, \exists A)$ and $P(Z^{A,post(A)}|A_s, \exists A)$ for every action A (propagation in the opposite direction to the first step).

For primitive v, given $P(Z^{post(v)}|v_e, \exists v)$:

$$P(v_e, Z^{v,post(v)}|v_s, \exists v) = P(Z^{post(v)}|v_e, \exists v)P(v_e|v_s)P(Z^v|v_s, v_e) \tag{A.7}$$

$$P(Z^{v,post(v)}|v_s, \exists v) = \sum_{t=1}^{T} P(v_e = t, Z^{v,post(v)}|v_s, \exists v) \tag{A.8}$$

For AND-rule $A \to MN$: given $P(Z^{post(A)}|A_e, \exists A)$, then N_e has the same distribution:

$$P(Z^{post(N)}|N_e = t, \exists N) = P(Z^{post(A)}|A_e = t, \exists A) \tag{A.9}$$

Recursively perform backward phase on N to get $P(Z^{N,post(N)}|N_s, \exists N)$, then M_e has the same distribution:

$$P(Z^{post(M)}|M_e = t, \exists M) = P(Z^{N,post(N)}|N_s = t, \exists N) \tag{A.10}$$

Recursively perform backward phase on M to get $P(Z^{M,post(M)}|M_s, \exists M)$, then A_s has the same distribution:

$$P(Z^{A,post(A)}|A_s = t, \exists A) = P(Z^{M,post(M)}|M_s = t, \exists M) \tag{A.11}$$

For OR-rule $A \to M \mid N$: given $P(Z^{post(A)}|A_e, \exists A)$, then M_e and N_e has the same distribution:

$$P(Z^{post(M)}|M_e = t, \exists M) = P(Z^{post(A)}|A_e = t, \exists A) \tag{A.12}$$

$$P(Z^{post(N)}|N_e = t, \exists N) = P(Z^{post(A)}|A_e = t, \exists A) \tag{A.13}$$

Recursively perform backward phase on M and N to get $P(Z^{M,post(M)}|M_s, \exists M)$ and $P(Z^{N,post(N)}|N_s, \exists N)$. Then:

$$\begin{aligned}P(Z^{A,post(A)}|A_s = t, \exists A) = P(\exists M|\exists A)P(Z^N|!N)P(Z^{M,post(M)}|M_s = t, \exists M)\\ +P(\exists N|\exists A)P(Z^M|!M)P(Z^{N,post(N)}|N_s = t, \exists N)\end{aligned}$$

Step 3 - Compute the posteriors: by multiplying forward and backward messages:

$$P(A_s, Z^{pre(A),A,post(A)}|\exists A) = P(A_s, Z^{pre(A)}|\exists A)P(Z^{A,post(A)}|A_s, \exists A) \tag{A.14}$$

$$P(A_e, Z^{pre(A),A,post(A)}|\exists A) = P(A_e, Z^{pre(A),A}|\exists A)P(Z^{post(A)}|A_e, \exists A) \tag{A.15}$$

These are equivalent to $P(A_s, Z|\exists A)$ and $P(A_e, Z|\exists A)$ thanks to step 0. Normalization is done to obtain the posterior $P(A_s|Z, \exists A)$ and $P(A_e|Z, \exists A)$.

If v is a primitive, we can have the joint:

$$P(v_s, v_e, Z|\exists v) = P(v_s, Z^{pre(v)}|\exists v)P(v_e|v_s)P(Z^v|v_s, v_e)P(Z^{post(v)}|v_e, \exists v) \tag{A.16}$$

Also we can find

$$P(Z) = \sum_t P(S_s = t, Z) \tag{A.17}$$

Step 4 - Compute the happening probability: starting with $P(\exists S|Z) = P(\exists S) = 1$, we find $P(\exists A|Z)$ for every action A recursively.

For AND-rule $A \to MN$: given $P(\exists A|Z)$, then:

$$P(\exists M|Z) = P(\exists N|Z) = P(\exists A|Z) \tag{A.18}$$

For OR-rule $A \to M \mid N$: given $P(\exists A|Z)$, we compute:

$$P(\exists M, Z|\exists A) = P(\exists M|\exists A) \sum_{t>0} P(M_e = t, Z|\exists M) \tag{A.19}$$

$$P(\exists N, Z|\exists A) = P(\exists N|\exists A) \sum_{t>0} P(N_e = t, Z|\exists N) \tag{A.20}$$

$$P(\exists M|Z) = P(\exists A|Z) \frac{P(\exists M, Z|\exists A)}{P(\exists M, Z|\exists A) + P(\exists N, Z|\exists A)} \tag{A.21}$$

$$P(\exists N|Z) = P(\exists A|Z) \frac{P(\exists N, Z|\exists A)}{P(\exists M, Z|\exists A) + P(\exists N, Z|\exists A)} \tag{A.22}$$

Step 5 (optional) compute the joint posterior of the start and the end: first recursively compute $P(A_e, Z^A|A_s)$.

For primitive v:

$$P(v_e, Z^v|v_s) = P(v_e|v_s)P(Z^v|v_s, v_e) \tag{A.23}$$

For AND-rule $A \to MN$:

$$P(A_e = \beta, Z^A|A_s = \alpha) = \sum_{t>0} P(M_e = t, Z^A|M_s = \alpha)P(N_e = \beta, Z^A|N_s = t) \tag{A.24}$$

For OR-rule $A \to M \mid N$:

$$\begin{aligned} P(A_e = \beta, Z^A|A_s = \alpha) = {} & P(\exists M|\exists A)P(Z^N|!N)P(M_e = \beta, Z^A|M_s = \alpha) \\ & + P(\exists N|\exists A)P(Z^M|!M)P(N_e = \beta, Z^A|N_s = \alpha) \end{aligned}$$

Then for each A:

$$P(A_s, A_e, Z|\exists A) = P(A_s, Z^{pre(A)}|\exists v)P(A_e, Z^A|A_s)P(Z^{post(A)}|A_e, \exists A) \tag{A.25}$$