# From Stochastic Grammar to Bayes Network: Probabilistic Parsing of Complex Activity

Anonymous CVPR submission

Paper ID 83

## Abstract

*We propose a probabilistic method for parsing complex activities that are defined as composition of sub-activities. The temporal structure is represented by a string-length limited stochastic context-free grammar. Given the grammar, a Bayes network is generated where the variable nodes correspond to the start and end times of component actions, and the network integrates information about duration of each primitive action, visual detection results for each primitive action, and the activity's temporal structure. At each moment in time during the activity, message passing is used to perform exact inference yielding the posterior probabilities of the start and end times for each different actions. We provide demonstrations of this framework being applied to various vision tasks such as action prediction, classification of the high-level activities or temporal segmentation of a test sequence; the method is also applicable in Human Robot Interaction domain where continually prediction of human's actions is needed.*

## 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 activity has been performed and often to parse or segment the input sequence into its component activities. 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 [7, 8].

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 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 likelihood as to which actions have occurred or will occur and when.

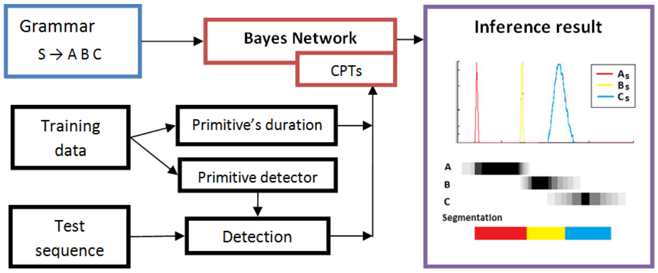


Figure 1. Overview of our framework

An overview of our framework is shown in Figure 1. A stochastic-context-free-grammar is used to represent the activitys structure: AND operations in the grammars production rules encode ordered composition of sub-actions while the OR operations permit variation in the course of actions. Next we develop a method to generate a discrete-time Bayes network whose hidden nodes are actions timings, observed nodes are the output of primitive actions detectors, and edges encodes dependencies between action-action or action-detection. Given the networks conditional probability tables (CPT) computed using the learned duration models and the visual detector outputs, we describe how to perform exact inference by a message-passing algorithm. Our framework has several advantages: (1) The grammar offers a hierarchical representation of the activity, allowing multiple layers of abstraction when defining action compositions (2) The primitive actions detector is assumed to be a blackbox and can be engineered independent of the grammar. (3) The information obtained from the inference output is very rich: the posterior probability if an action happens and when it happens or ends. The probability of an action being active at a time step can also be inferred for every action and every time step both in the past and in the arbitrarily far future.

## 2. Related Works

Broadly, previous work in activity recognition can be characterized as belonging to one of two classes of analysis. The first focuses on the short, low-level actions. There is a rich literature in recognizing them. Common approaches use BOW framework with video feature such as STIP, HOG3D, Cuboid [19], Dense Trajectories [18]. When these short time actions appear in a sequence of such primitives, there is the more challenging tasks of segmenting the sequence or localizing the actions within the sequence. For example, Tang et al [17] use variable-duration hidden Markov model to exploit temporal structure for action recognition and event detection. In [14, 6], SVM is used for classification and segmentation is chosen to maximize the total SVM score using dynamic programming. These approaches assume the activity is a Markov process that does not easily permit the consideration of a more global temporal structure.

The second type of activity representation and recognition methods considers more complex activities that can be meaningfully decomposed into smaller components. A variety of models have been proposed to represent the compositional structure and to perform the video analysis. In [1], Albanese et al used Probabilistic Petri Net to detect interesting human activities. Si et al [16] learn an AND-OR grammar from example strings of symbols, each represents an action according to the grammar's language. The learned grammar discovers the pattern of actions and can be used for prediction. This, however, assumes that the string is not noisy i.e. the primitive action detectors are perfect. In [13], Probabilistic Suffix Tree was used to learn the pattern of symbols (or ”actionlets”) for early detection of on-going activity. In [9], a stochastic context free grammar parsing technique was proposed to parse a discrete set of events, which could potentially contains false positives. To the same end, Damen et al [3] 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 corresponds to the best explanation. Different from these approaches, we assume detection of primitive actions is not a discrete set, but more general: a ”heatmap” that represents the action likelihood for every interval, hence our framework can handle wide range of uncertainties, such as false negative detections.

The approaches most related to our work are Dynamic Bayes Network (DBN) methods [15, 12] in which the system's state ecodes what actions have happened and when the current action started. 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 [11] introduce Anticipatory Temporal Conditional Random Field, which is an undirected graphical model designed to run online like a DBN and also uses a particle filter to do inference. Prediction is done by extending the network into the future by a fixed, small number of frames.

## 3. Modeling complex activity structure by a Stochastic Grammar

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.

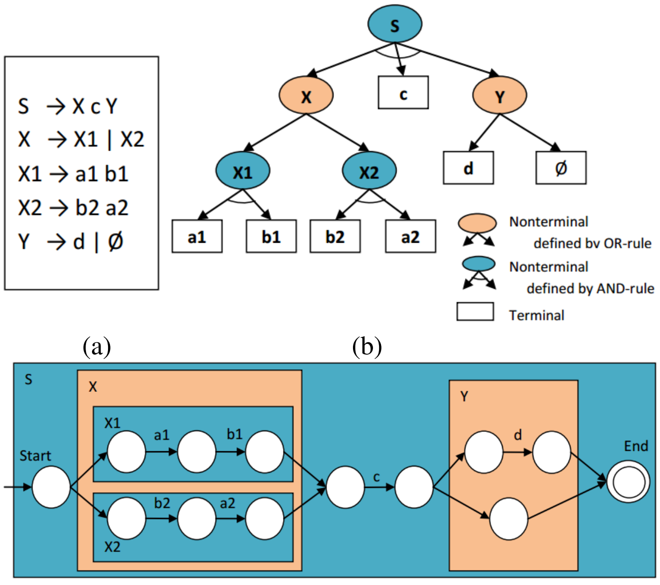
We use a stochastic grammar to model this hierarchical structure. The grammar is formally defined as G = (S, T, N, R) where: T is the set of terminal symbols, corresponding to the primitives, N is the set of non-terminal symbols, corresponding to the compositions, S is the starting symbol, corresponding 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 the corresponding Bayes network, a preprocessing step is necessary to convert the original grammar to a “compiled version” that satisfies three constraints. Two of these constraints are merely syntactic and do not restrict the structure of the top level activity. The last constraint places a string-length limitation that bounds the length of time it takes to complete the activity.

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.



(a) (b)

(c)

Figure 2. Example activity: "$S \rightarrow (ab \mid ba)\; c\; (d \mid \emptyset)$". 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

3. The grammar cannot generate arbitrary long strings since our Bayes network will cover all possible sequences. This means rules causing loop such as: "$S \rightarrow SA \mid A$" are not allowed. Explicitly unrolling such loops to a maximum number of times can be done to avoid this situation.

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. Bayes network generation and inference

Our input will be the compiled grammar of the activity. First we define random variables $A_s$ and $A_e$ representing the starting time and ending time for every action $A$, and let $Z^A$ be the observations of the detector associated with action A; we 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 activitys length. Depending on the course of actions as permitted by the grammar, action $A$ might 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 ($A_e > A_s > 0$) 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 the network is constructed in a hierarchical fashion, similar to the AND-OR tree (Figure 2.b). To do so, we describe how to construct the network 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 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 ($A_s = A_e = -1$).

## 4.1. The primitive $v$

The portion of the network that corresponds to a primitive $v$ is shown in Figure 3.a. We use the notation $Z^{pre(A)}$ and $Z^{post(A)}$ to stand for the observation of all actions that happen before $A$ and after $A$, respectively. There are two conditional probabilities required for this component:

**The condition 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 $\mu_v, \sigma_v$ 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: $P(v_e = -1|v_s = -1) = 1$.

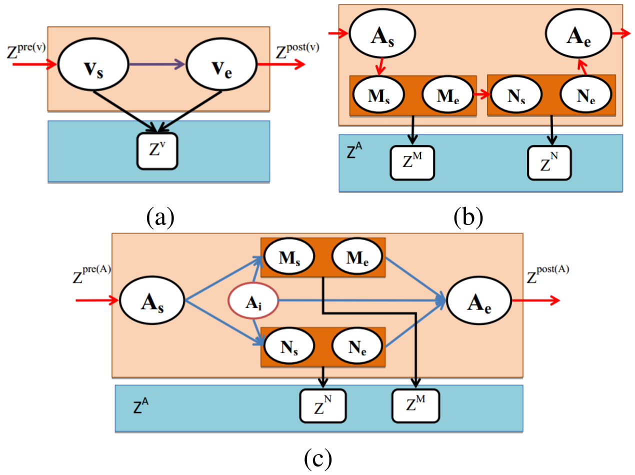
**Likelihood** $P(Z^v|v_s, v_e)$ **:** each primitive has a visual detector that outputs a detection score $F_v[\alpha, \beta]$ representing the evidence that the action starts at time $\alpha$ and ends at time $\beta$ for every possible $(\alpha, \beta)$ of the range [1, T] (covering the entire activity).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

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

This rule defines the action $A$ to be the sequence of subaction $M$ 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 $Z^A = Z^{M,N}$

[1] If time between actions is needed we can insert a special PAUSE action between them; the detector would be declared constant for all start and end values.



(a) (b)

(c)

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

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

Figure 3.c shows the OR network structure. 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 $A_s = N_s, A_e = N_e, M_s = M_e = -1$).

The standard approach to realizing this "OR" condition in a Bayes network is to use the multiplexer CPT, with a "selector" variable [10] 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).

Finally for every composition A, 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 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.

### 4.4. Exact inference by message passing

Given the constructed network, we now compactly show a 4-step inference process (a more explicit description will be given in the supplemental document). Beside the CPT $P(v_e|v_s)$ and $P(Z^v|v_s, v_e)$ described in section 4.1, we need three 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 "$S \rightarrow A \mid \emptyset$" 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 6). On the other hand, we can assume uniform distributions about the start and the end of the activity (experiment in section 5). In that case, our framework effectively performs detection and parsing at the same time.

**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 $P(v_e, Z^{pre(v),v}|\exists v)$ .

For AND-rule $A \rightarrow MN$: given $P(A_s, Z^{pre(A)}|\exists A)$, the variable $M_s$ has the same distribution. Recursively perform forward phase on M to get $P(M_e, Z^{pre(A),M}|\exists M)$. Next the variable $N_s$ has the same distribution, so we can perform forward phase on N to get $P(N_e, Z^{pre(A),M,N}|\exists N)$. The final result is the same as distribution $P(A_e, Z^{pre(A),A}|\exists A)$.

For OR-rule $A \rightarrow M \mid N$: 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 $A_s$, we can therefore perform forward phase on M to get $P(M_e, Z^{M,pre(M)}|\exists M)$. Similarly we can find $P(N_e, Z^{N,pre(N)}|\exists N)$, then distribution of $A_e$ is:

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

**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).

**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. If v is a primitive, we can have the joint distribution:
$P(v_s, v_e, Z|\exists v) = P(v_s, v_e, Z^{pre(v),v}|\exists v)P(Z^{post(v)}|v_e, \exists v)$
Also we can find $P(Z) = \sum_t P(S_s = t, Z)$

**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 $A \rightarrow MN$: given $P(\exists A|Z)$, then $P(\exists M|Z) = P(\exists N|Z) = P(\exists A|Z)$

For OR-rule $A \rightarrow M \mid N$: given $P(\exists A|Z)$, we compute (apply similar formulas for N):

$$P(\exists M, Z|\exists A) = P(\exists M|\exists A) \sum_{t>0} P(M_e = t, Z|\exists M) \tag{1}$$

$$P(\exists M|Z) = P(\exists A|Z) \frac{P(\exists M, Z|\exists A)}{P(\exists M, Z|\exists A) + P(\exists N, Z|\exists A)} \tag{2}$$

**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.

### 4.5. 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 $P(A_s|Z)$, $P(A_e|Z)$:

$$P(A_s|Z) = P(\exists A|Z) \frac{P(A_s, Z|\exists A)}{\sum_{t>0} P(A_s = t, Z|\exists A)} \tag{3}$$

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:

$$P(label_t = v|Z) = \sum_{\alpha=1}^{t} \sum_{\beta=t}^{T} P(v_s = \alpha, v_e = \beta|Z) \tag{4}$$

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. Alternative way for segmentation is to derive the parsing with highest probability (the most probable course of actions), estimate the start and the end of the actions in that sequence, then labels the frames.

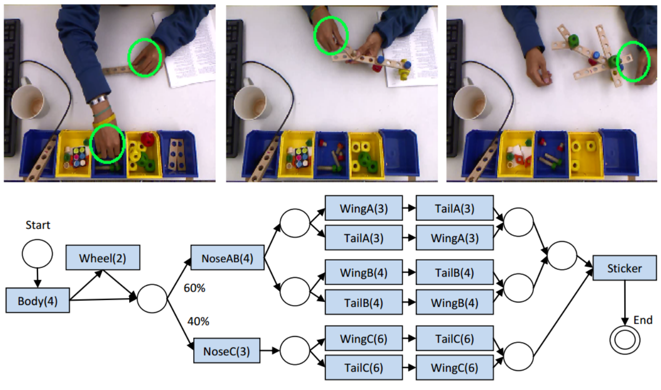


Figure 4. The toy assembly task. (a) Example input frames and hand detection result. (b) The temporal structure in the form of state machine. Each box is an AND composition of a number of primitives (shown in the bracket). There are 40 different primitive actions and 12 complete task variations.

### 4.6. Implementation Explanation

The primitive detector is 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 $A \rightarrow M|N$, 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 using of different kinds of detectors is possible. 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 scores). For example one can normalize the svm score and apply a sigmoid function to get a score that better indicates the likelihood. The likelihood value 0 is usually discouraged as it could nullify the likelihood of other actions.

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

**Parsing in streaming mode:** 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.

## 5. Toy assembly task experiment

To demonstrate the capability of the our method, 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 tasks structure is shown in Figure 4 and the

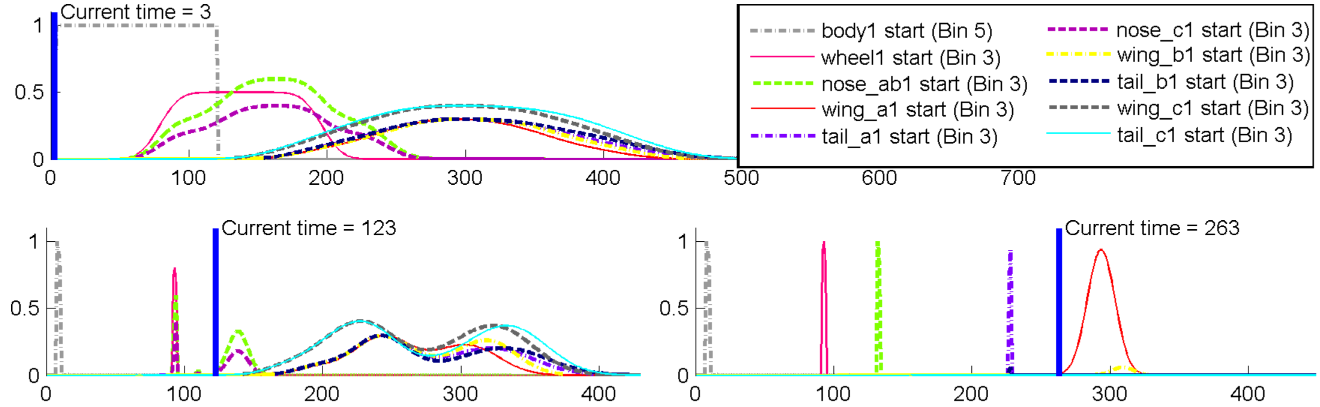


Figure 5. (best view in color) Example posterior distributions of the starts of some actions (first primitive of each part) at 3 different time steps (computed using Eq.3, note that the special value (-1) is not shown). In order to observe them easily, each distributions is scaled so that its peak have the same value as the probability that action would happen (e.g. for $body1$, it is 1; for $nose\_ab1$, it's 0.6 in the first timestep and about 0.99 in the last timestep)

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.

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 $\mu_v, \sigma_v$ 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).

**Qualitative Result:** in Figure 5, 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). 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.

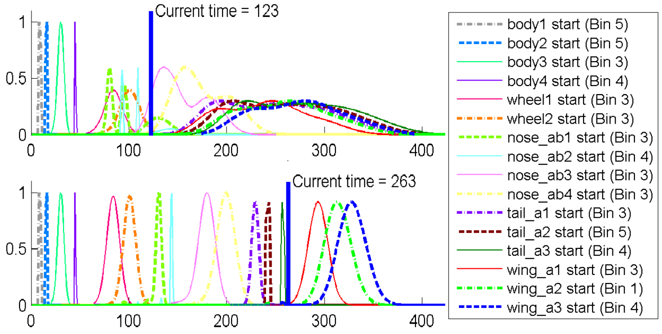


Figure 6. In this example, detectors of actions reaching for pieces in bin 3 are disabled. Here we show all actions that actually happen in this sequence.

Our method can perform robustly in the presence of noise (i.e. some detectors are weak and generate false positives or false negatives detections). In Figure 6, we show the result on the same sequence, only this time we completely disable the detectors of all actions involving bin 3. There are great uncertainty at the beginning, but eventually the system is able to recognized the sequence of actions. The timings of bin 3's actions are estimated based upon the detection of other actions and the temporal constraint between them (e.g. $body3$ is between $body2$ and $body4$).

**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 7 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

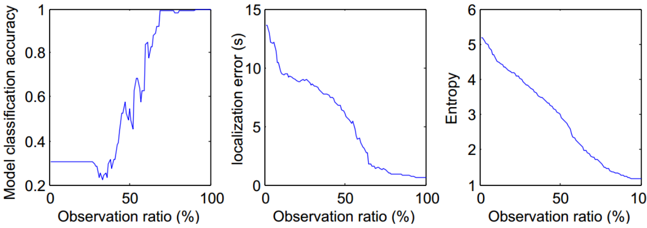


Figure 7. Result on our toy assembly dataset: (a) accuracy of model classification, (b) average localization error and (c) entropy of all actions' start

observation is available) and the accuracy is 91.2%.

Note that by taking the mean of the distributions, we are making a guess that minimizes the squared 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. In earlier work [?] we developed a linear-chain only method (no grammar) to apply in a Human Robot Collaboration context where the inference runs online in realtime. In that joint-assembly scenario, 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.

## 6. Recognition and Segmentation experiment

Here we describe two activity segmentation experiments. The first is on data we constructed from a known data set of actions to illustrate how to apply our approach to segmentation. The second experiment is on an existing dataset for which segmentation results have been reported.

### 6.1. Weizmann dataset

This dataset has 93 sequences of 10 actions: walk, run, jump, side, bend, wave1, wave2, jump, jumpjack, skip; we will consider these to be primitives [2]. To create composite activities, we concatenate 10 different actions to create long sequences in the manner of [6]. We randomly choose 1 long sequence for testing and the remaining for training.

We firt describe how we implement the primitive detector: like [6] 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 $d[\alpha, \beta]$ as the $\chi^2$ distance to the nearest neighbor (in the set of positive training examples). Finally we define the similarity score $F_v[\alpha, \beta] \propto (max(10^{-5}, 1 - d[\alpha, \beta]))^2$

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:

| Method | Segmentation Accuracy |
|---|---|
| [14] | 69.7% |
| [6] | 87.7% |
| Ours | 93.6% |

Table 1. Result on Weizmann dataset experiment

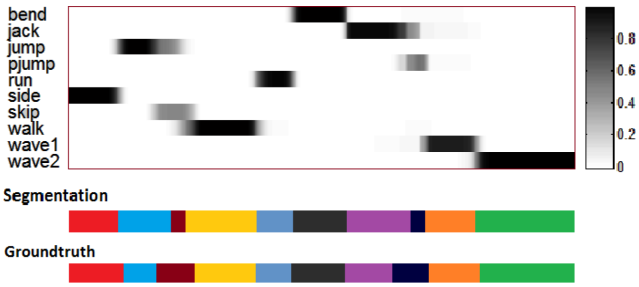


Figure 8. Example Segmentation result on the Weizmann dataset. On the top, the probability of each frame's label (corresponding to each column), computed using Eq.4, is shown. Segmentation is done by choosing the label with highest probability. The groundtruth is shown at the bottom.

$$\begin{aligned} &S \rightarrow AAAAAAAAAAAAAAAAAAAAA... \\ &A \rightarrow walk \mid run \mid jump \mid side \mid bend \mid \\ &\qquad wave1 \mid wave2 \mid jump \mid jumpjack \mid skip \mid \emptyset \end{aligned}$$

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 [6]. Our result is shown in Table 1, in comparison with discriminative methods in [6] and [14]. One advantage is that we can assert the confidence about each's frame label using the output distributions, one example shown in Figure 8.

### 6.2. GeorgiaTech Egocentric Activity dataset

We consider this a more challenging dataset [5]. It consists of 7 high level activities such as making a cheese sandwich or making coffee; 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 [5], 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 [5]. Since these raw scores are not a good indicator of the likelihood, we define our 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 try a 2nd setting, where we use $F_v[\alpha, \beta] \propto (S_B[v, \alpha] S_E[v, \beta])^{10}$.

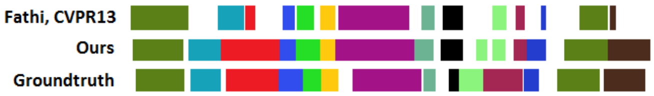


Figure 9. Example Segmentation result on GTEA of the activity: making Cheese Sandwich.

The grammar is very important and design of a good grammar is not trivial (this will be discussed in next section). 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 examplar-like approach effectively matches the testing sequence with all the training data to find similar sequences (even though they are not exactly the same).

Our segmentation accuracy is 51% in 1st setting and 59% in the 2nd setting, compare with [5]'s 42% and [4]'s 33%. One example result is shown in Figure 9. It should be noted that this dataset is very challenging: [5] reported a primitive action classification accuracy of 39.7% (where random chance is 1.6%).

Unlike [5], our method models the global structure of the activity and is able to natively classify high level activity using posterior probabilities associated with the OR-rule. In this experiment, our method correctly classifies the high level activity label of 6 out of 7 test sequences.

## 7. Conclusion

We have presented a novel framework for modeling complex composite activity using a Stochastic Grammar. Parsing a sequence is done by performing message passing on the generated Bayes Network. As shown in the experiments, our method allows different kinds of queries, it outputs the posterior distributions of: (1) all actions' timing, which can be used for localization/prediction of actions/events, (2) strings realized by the grammar (the sequence of actions), which can be used for classifying high level activity or deriving sequence that best explains the observation, and (3) frames' label which can be used for activity segmentation.

For future work, we consider a dynamic time resolution approach for time series to speed up the inference speed. That way, even sequences with length of hours can be processed efficiently. While the grammar is flexible and can be constructed using expert's knowledge, there might be different way to do that and choosing a good grammar is important. Moreover, it is desirable to learn it from training data, so the problem of grammar induction is very relevant to our work. Finally, the grammar can be extended to realize multiple streams of actions going on at the same time. This will be useful for modeling process such as complicated interaction between multiple agents.

## References

[1] M. Albanese, R. Chellappa, V. Moscato, A. Picariello, V. Subrahmanian, P. Turaga, and O. Udrea. A constrained probabilistic petri net framework for human activity detection in video. *Multimedia, IEEE Transactions on*, 2008. 2

[2] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *ICCV*, 2005. 7

[3] D. Damen and D. Hogg. Explaining activities as consistent groups of events. *International journal of computer vision*, 2012. 2

[4] A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In *ICCV*, 2011. 8

[5] A. Fathi and J. M. Rehg. Modeling actions through state changes. In *IEEE CVPR*, 2013. 7, 8

[6] M. Hoai, Z.-Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *IEEE CVPR*, 2011. 2, 7

[7] G. Hoffman and C. Breazeal. Cost-Based Anticipatory Action Selection for HumanRobot Fluency. *IEEE Transactions on Robotics*, 23(5):952–961, Oct. 2007. 1

[8] M. Huber and A. Knoll. When to assist?-Modelling human behaviour for hybrid assembly systems. In *Robotics (ISR)*, 2010. 1

[9] Y. A. Ivanov and A. F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2000. 2

[10] D. Kollar and N. Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009. 4

[11] H. S. Koppula and A. Saxena. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. ICML, 2013. 2

[12] B. Laxton, J. Lim, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *IEEE CVPR*, 2007. 2

[13] K. Li, J. Hu, and Y. Fu. Modeling complex temporal composition of actionlets for activity prediction. In *ECCV*. 2012. 2

[14] Q. Shi, L. Wang, L. Cheng, and A. Smola. Discriminative human action segmentation and recognition using semi-markov model. In *IEEE CVPR*, 2008. 2, 7

[15] Y. Shi, Y. Huang, D. Minnen, A. Bobick, and I. Essa. Propagation networks for recognition of partially ordered sequential action. In *IEEE CVPR*, 2004. 2

[16] Z. Si, M. Pei, B. Yao, and S.-C. Zhu. Unsupervised learning of event and-or grammar and semantics from video. In *ICCV*, 2011. 2

[17] K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *IEEE CVPR*, 2012. 2

[18] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *IEEE CVPR*, 2011. 2

[19] 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. 2