Hierarchical Multi-channel Hidden Semi Markov Models *

Pradeep Natarajan and **Ramakant Nevatia**
Institute for Robotics and Intelligent Systems,
University of Southern California,
Los Angeles, CA 90089-0273
{pnataraj, nevatia}@usc.edu

Abstract

Many interesting human actions involve multiple interacting agents and also have typical durations. Further, there is an inherent hierarchical organization of these activities. In order to model these we introduce a new family of hidden Markov models (HMMs) that provide compositional state representations in both space and time and also a recursive hierarchical structure for inference at higher levels of abstraction. In particular, we focus on two possible 2-layer structures - the Hierarchical-Semi Parallel Hidden Markov Model (HSPaHMM) and the Hierarchical Parallel Hidden Semi-Markov Model (HPaHSMM). The lower layer of HSPaHMM consists of multiple HMMs for each agent while the top layer consists of a single HSMM. HPaHSMM on the other hand has multiple HSMMs at the lower layer and a Markov chain at the top layer. We present efficient learning and decoding algorithms for these models and then demonstrate them first on synthetic time series data and then in an application for sign language recognition.

1 Introduction

Our goal is to develop methods for inferring activities given observations from visual and other sensory data. Activities have a natural hierarchical structure where combinations of simpler activities form higher level, more complex activities. In general, multiple agents may be involved who take parallel actions but whose temporal (and spatial) relations are important for inference of higher level activities. We present a mathematical formalism for recognizing such multi-agent activities that can take advantage of both their inherent hierarchical organization and typical durations. Such models can have a wide range of applications such as surveillance, assistive technologies like sign language recognition and intelligent environments.

A key challenge faced by researchers in activity recognition is to bridge the gap between the observations (such as limb joints) and the high level semantic concepts (such as gestures) that need to be recognized and to handle the large variations in the duration and styles of these activities when performed by different people or even the same person at different times. Due to uncertainty inherent in sensory observations, probabilistic reasoning offers a natural approach. While several probabilistic models have been proposed over the years in various communities for activity recognition, hidden Markov models (HMMs) and their extensions have by far been the most widely used ones. They offer advantages such as clear Bayesian semantics, efficient learning and inferencing algorithms, and a natural way to introduce domain knowledge into the model structure. [Bui *et al.*, 2004] presented an extension of the hierarchical hidden Markov model (HHMM) for inferring activities at different levels of abstraction. [Hongeng and Nevatia, 2003] used the hidden semi-Markov model (HSMM) for modeling activity durations, while [Duong *et al.*, 2005] presented the switching hidden semi-Markov model (S-HSMM) to exploit both the hierarchical structure as well as typical durations for activity recognition. [Vogler and Metaxas, 1999] introduced the parallel hidden Markov model (PaHMM) for recognizing complex 2-handed gestures in ASL while [Brand *et al.*, 1997] introduced the coupled hidden Markov model (CHMM) and applied it to recognize tai chi gestures as well as multi-agent activities.

We believe that in most real applications we need models that combine the features of all of these models and introduce a new class of models, namely the hierarchical multi-channel hidden semi-Markov models. In particular we focus on two model structures - the hierarchical semi parallel hidden Markov model (HSPaHMM) and the hierarchical parallel hidden semi-Markov model (HPaHSMM) and present efficient decoding and learning algorithms for them. We validate the utility of these models by testing on simulated data as well as for the real task of continuous sign language recognition.

The rest of the paper is organized as follows - In the next section we define the parameters of HSPaHMM and HPaHSMM formally. Then in the sections 3 and 4 we present efficient decoding and learning algorithms for them. Finally in section 5 we present the experimental results.

*This research was partially funded by Advanced Research and Development Activity of the U.S. Government under contract MDA-904-03-C-1786.

2 Model Definition and Parameters

We begin by defining a standard HMM model λ by the tuple (Q, O, A, B, π) where, Q is the set of possible states, O is the set of observation symbols, A is the state transition probability matrix ($a_{ij} = P(q_{t+1} = j | q_t = i)$), B is the observation probability distribution ($b_j(k) = P(o_t = k | q_t = j)$) and π is the initial state distribution. It is straightforward to generalize this model to continuous (like gaussian) output models.

This can be extended to a hierarchical hidden Markov model (HHMM) by including a hierarchy of hidden states. A HHMM can be formally specified by the tuples- $\lambda' = (Q_d, O_d, A_d, B_d, \pi_d)$ where $d \in 1..H$ indicates the hierarchy index.

In traditional HMMs, the first order Markov assumption implies that the duration probability of a state decays exponentially. The hidden semi-Markov models (HSMM) were proposed to alleviate this problem by introducing explicit state duration models. Thus a HSMM model can be specified by the tuple- $\lambda'' = (Q, O, A, B, D, \pi)$ where, D contains a set of parameters of the form $P(d_i = k)$, i.e. the probability of state *i* having a duration *k*.

In the basic HMM, a single variable represents the state of the system at any instant. However, many interesting activities have multiple interacting processes and several multi-channel HMMs have been proposed to model these. These extensions basically generalize the HMM state to be a collection of state variables ($S_t = S_t^1, , S_t^C$). In their most general form, such extensions can be represented as- $\lambda''' = (Q^C, O^C, A^C, B^C, \pi^C)$ where Q^c and O^c are the possible states and observations at channel c respectively and π^c represents the initial probability of channel c's states. A^C contains the transition probabilities over the composite states ($P([q_{t+1}^1, .., q_{t+1}^C] | [q_t^1, .., q_t^C])$), and B^C contains the observation probabilities over the composite states ($P([o_t^1, .., o_t^C] | [q_t^1, .., q_t^C])$). In this form, the learning as well as inferencing algorithms are exponential in C, and also result in poor performance due to over-fitting and large number of parameters to learn. The various multi-channel extensions typically introduce simplifying assumptions that help in factorizing the transition and observation probabilities. Parallel hidden Markov models (PaHMM) factor the HMM into multiple independent chains and hence allow factorizing both A^C and B^C. Coupled hidden Markov models (CHMM) factor the HMM into multiple chains where the current state of a chain depends on the previous state of all the chains.

The hierarchical multi-channel hidden semi-Markov models that we propose try to combine all the above characteristics into a single model structure. In the most general form, they can be described by a set of parameters of the form- $\lambda'''' = (Q_d^c, O_d^c, A_d^c, B_d^c, D_d^c, \pi_d^c)$ where, $d \in 1..H$ is the hierarchy index and c is the number of channels at level d, and the parameters have interpretations similar to before. Each channel at a higher level can be formed by a combination of channels at the lower level. Also, the duration models at each level is optional. Further, the channels at each level in the hierarchy maybe factorized using any of the methods discussed above (PaHMM, CHMM etc). It can be seen that λ'''' presents a synthesis of λ', λ'' and λ'''. HSPaHMM has 2 layers with multiple HMMs at the lower layer and HSMM at the upper layer and has the following set of parameters-

$$\lambda_{lower}^{HSPaHMM} = (Q_{lower}^C, O_{lower}^C, A_{lower}^C, B_{lower}^C, \pi_{lower}^C)$$
$$\lambda_{upper}^{HSPaHMM} = (Q_{upper}, O_{upper}, A_{upper}, B_{upper}, D_{upper}, \pi_{upper})$$

HPaHSMM contains multiple HSMMs at the lower layer and a single Markov chain at the upper layer and hence has the following set of parameters-

$$\lambda_{lower}^{HPaHSMM} = (Q_{lower}^C, O_{lower}^C, A_{lower}^C, B_{lower}^C, D_{lower}^C, \pi_{lower}^C)$$
$$\lambda_{upper}^{HPaHSMM} = (Q_{upper}, O_{upper}, A_{upper}, B_{upper}, \pi_{upper})$$

Figure 1 illustrates the various HMM extensions discussed. The key difference between HSPaHMM and HPaHSMM is that HSPaHMM models the duration for the entire low-level PaHMM with the top-level HSMM state while HPaHSMM models the duration of each state in each low-level HSMM. Thus HSPaHMM requires fewer parameters, while HPaHSMM is a richer structure for modeling real events.

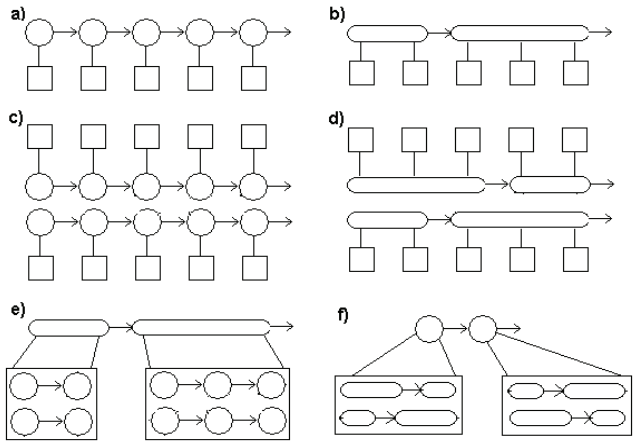

Figure 1: Structure of a)HMM b)HSMM c)PaHMM d)PaHSMM e)HSPaHMM f)HPaHSMM

3 Decoding Algorithm

In this section, we use the following notations - Let *C* be the number of channels in lower layer, *T* the number of frames or observations in each channel, *N* the number of states in each channel of the lower layer[1] and *W* the number of lower level HMMs. Then, the top-level HMM will have one state for every lower level HMM as well as a state for every possible transition between lower level HMMs. This is because in applications like sign language recognition where each word is modeled by a multi-channel HMM/HSMM each transition between the words in the lexicon is distinct. Each of the *W* HMMs can transition to any of the *W* HMMs giving a total of W^2 transition states. Thus the top-level HMM has a total of $W * (W + 1)$ states.

3.1 HSPaHMM Decoding

The decoding algorithm for HSPaHMM works by traversing the top-level HSMM and at each time segment finding the maximum likelihood path through the low-level PaHMM

[1]We have assumed that all the lower level HMMs have the same number of states for simplicity of notation. It is straightforward to extend the algorithms presented to cases where the low-level HMMs have varying number of states.

corresponding to the upper HSMM's state. Traversing the top-level HSMM states takes $O(W^2(W+1)^2T^2)$ time and each call to the lower PaHMM takes $O(CN^2T)$ time giving a total complexity of $O(W^2(W+1)^2N^2T^3)$. We can reduce this complexity if the duration models for the top-level HSMM are assumed to be uniform or normal. In this case, we can define parameters M and Σ for each state in the top-level HSMM such that we only need to consider state durations in the range $[M-\Sigma, M+\Sigma]$. The values of M and Σ can be obtained from the lower and upper thresholds in the case of uniform distributions or from the mean and variance in normal distributions. In this case since we need to check only 2Σ durations at each instant and each low-level PaHMM has M observations on average, traversing the top-level HSMM takes $O(W^2(W+1)^2T\Sigma)$ and each lower level PaHMM takes $O(CN^2M)$ giving a total inference complexity of $O(CW^2(W+1)^2N^2TM\Sigma)$. Still, even for small values of W the inference complexity become prohibitively large as it varies with $W^2(W+1)^2$. We can get around this by storing only the top *K* states of the upper-HSMM at each instant. Then, since each state can only transfer to $O(W)$ states (transition states can transfer only to the destination states while states corresponding to lower-level PaHMMs can transfer to W transition states), the overall inference complexity becomes $O(CWKN^2TM\Sigma)$. Figure 2 illustrates the decoding algorithm and Algorithm 1 presents the pseudocode.

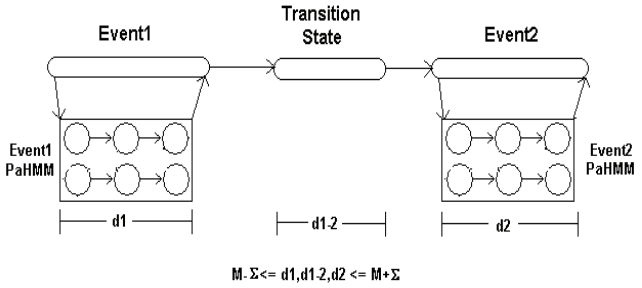

Figure 2: Decoding HSPaHMM

Algorithm 1 HSPaHMM Decoding

Top-level HSMM has W*(W+1) states
We use the following indices for states-
- Top-level states 0..W-1 have low-level PaHMMs corresponding to them.
- Top-level state $W + w_1W + w_2$ is the transition state for the $w_1 \rightarrow w_2$ transition where $w_1, w_2 \in [0..W-1]$
Beam(t)=Top K states at time t.
State i's duration $\in [M(i)-\Sigma(i), M(i)+\Sigma(i)]$
δ_t^i = probability of max-likelihood path to state i at time t.
$P^t(m|l)$ = Probability of transitioning to state m from state l.
$P^o(O_{t1}..O_{t2}|m)$ = Probability of observing $O_{t1}..O_{t2}$ in state m.
$P^d(d|m)$ = Probability of spending duration d in state m.
$maxPath(m, t1, t2)$ = Function to calculate probability of max-likelihood path through low-level PaHMM of state m from time $t1$ to $t2$. See [Vogler and Metaxas, 1999] for details.
for $i = 1$ to T **do**
  for $j = 1$ to K **do**
    $l \leftarrow j^{th}$ state in $Beam(i)$
    if $l < W$ **then**
      for $m = (l+1) * W$ to $(l+1) * W + W$ **do**
        for $n = M(m) - \Sigma(m)$ to $M(m) + \Sigma(m)$ **do**
          $\delta_{i+n}^m \leftarrow \delta_i^l * P^t(m|l) * P^d(n|m) * P^o(O_{i+1}..O_{i+n}|m)$
          if δ_{i+n}^m in top K paths at time $i+n$ **then**
            add m to $Beam(i+n)$
          end if
        end for
      end for
    else
      $m \leftarrow Remainder(l/W)$
      for $n = M(m) - \Sigma(m)$ to $M(m) + \Sigma(m)$ **do**
        $\delta_{i+n}^m \leftarrow \delta_i^l * P^d(n|m) * maxPath(m, i+1, i+n)$
        if δ_{i+n}^m in top K paths at time $i+n$ **then**
          add m to $Beam(i+n)$
        end if
      end for
    end if
  end for
end for
return δ of maximum probability state in Beam(T)

3.2 HPaHSMM Decoding

For decoding HPaHSMMs, we have to traverse the top-level Markov chain and at each state call the corresponding low-level PaHSMM. This procedure can be simplified by stringing together the low-level PaHSMMs and also the transition states into one large compound PaHSMM and then traversing it. Since the W low-level PaHSMMs have N states each and there are W^2 transition states in the top level, the compound PaHSMM has $W*(W+N)$ states. Note that the transition states can have their own duration and output models. Since each channel of the compound PaHSMM is a HSMM taking $O(W^2(W+N)^2T^3)$ [2] time, the entire decoding algorithm takes $O(CW^2(W+N)^2T^3)$. If the duration models are normal or uniform we only need to consider state durations in the range $[m-\sigma, m+\sigma]$ where the values of m and σ are defined for each state of every PaHSMM. The time complexity then becomes $O(CW^2(W+N)^2Tm\sigma)$. Further by storing only the top k states in the compound PaHSMM at each instant, we can reduce the complexity to $O(CkW(W+N)Tm\sigma)$. Figure 3 illustrates the decoding algorithm and Algorithm 2 presents the pseudocode for HPaHSMM.

[2] [Murphy, 2002] points out that HSMM inference complexity can be reduced to $O(W^2(W+1)^2T^2)$ by pre-calculating output probabilities for all states and intervals. However, since in beam search we only store only a small subset of the states at each time interval, this procedure actually increases run time in our case.

4 Embedded Viterbi Learning

In many applications like sign language recognition and event recognition the training samples typically contain a sequence of words/events which are not segmented. In such cases, we string together the individual word/event HMMs and then re-estimate the parameters of the combined HMM using the traditional Baum-Welch algorithm. Thus the individual events are segmented automatically during the re-estimation pro-

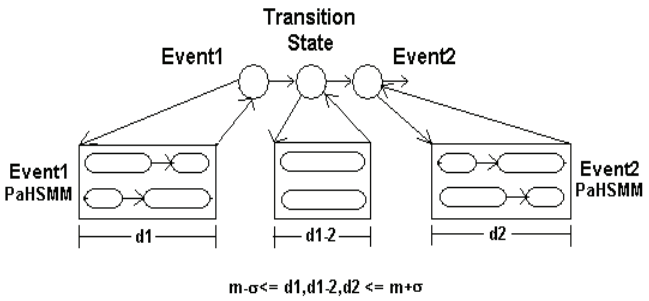

Figure 3: Decoding HPaHSMM

cess. In constrained HMM structures like the the left-right HMM (where each state can transition only to itself or to one state higher), the re-estimation procedure can be simplified by calculating the Viterbi path through the HMM at each iteration and re-estimating the parameters by a histogram analysis of the state occupancies. Since in our applications we are primarily interested in left-right HMMs, we adopted a Viterbi-like embedded training algorithm for HSPaHMM and HPaHSMM. Algorithm 3 presents the pseudocode for the learning algorithm where at each iteration we call the corresponding decoding algorithm and then re-estimate the parameters using a simple histogram analysis of state occupancies.

Algorithm 2 HPaHSMM Decoding

Beam(c,t)=Top k states at time t in channel c.
State i's duration $\in [m(i) - \sigma(i), m(i) + \sigma(i)]$
$minTh = \min(m(i) - \sigma(i)); maxTh = \max(m(i) + \sigma(i))$
$\delta_t^{c,i}$ = Probability of max-likelihood path to state i in channel c at time t.
$P_c^t(m|l)$ = Probability of transitioning to state m from state l in channel c.
$P^o(O_{t1}..O_{t2}|c,l)$ = Probability of observing $O_{t1}..O_{t2}$ in state l in channel c.
$P^d(d|c,l)$ = Probability of spending duration d in state l in channel c.
Beam(c,1) set using initialization probabilities of states $\forall$ c.
for $i = 2$ to T **do**
  for $j = minTh$ to $maxTh$ **do**
    for $l = 1$ to C **do**
      for $m = 1$ to $W * (W + N)$ **do**
        for $n = 1$ to k **do**
          p $\leftarrow n^{th}$ state in $Beam(l, i-1)$
          $\delta_{i+j}^{l,m} = \delta_{i-1}^{l,p} * P_c^t(m|p) * P^d(j+1|l,m) * P^o(O_i..O_{i+j}|l,m)$
          if $\delta_{i+j}^{l,m}$ in top k paths at time $i+j$ **then**
            add m to $Beam(l, i+j)$
          end if
        end for
      end for
    end for
  end for
end for
return $\Sigma_{c=1}^{C}\ \delta$ of maximum probability state in Beam(c,T)

5 Experiments

To evaluate learning and decoding in HSPaHMM and HPaHSMM we conducted two experiments: one with synthetic data, other with real data for a sign language (ASL) recognition task. In both cases, we compare our results with PaHMM without any duration models and beam search on the states. All run time results are on a 2GHz Pentium 4 Windows platform with 2GB RAM, running Java programs.

5.1 Benchmarks on Synthetic Data

In this experiment, we used a discrete event simulator to generate synthetic observation sequences. Each event can have C=2 channels/agents, and each channel can be in one of N=5 states at any time. State transitions were restricted to be left-to-right so that each state *i* can transition only to state (*i+1*). The states had 3-dimensional Gaussian observation models with the means uniformly distributed in [0,1] and the covariances were set to I/4. Further, each state had gaussian duration models with means in the range [10,15] and variances set to DPARAM=10. We then built a top-level event transition graph with uniform inter-event transition probabilities. Continuous observation sequences were then generated by random walks through the event transition graph and the corresponding low-level event models. Random noise was added in between the event sequences as well as at the beginning and end of the observation sequence. Figure 4 illustrates this procedure for a 2 event top-level transition graph. As can be seen, this setup corresponds to the HPaHSMM model structure. Observation sequences were generated using the setup described above using a 5-event top-level transition graph such that each sequence had 2-5 individual events and each event occurred at least 30 times in the entire set of sequences producing in total 50-60 sequences. We then randomly chose a set of training sequences such that each word occurred at least 10 times in the training set and used the rest as test sequence. Thus the training set contained 10-20 sequences and the test set contained 40-50 sequences. The data generators were then discarded and then the following models were trained - 1)HPaHSMM with randomly initialized output models, left-right low-level PaHSMMs and low-level duration models (parameters m, σ - see section 3.2) set accurately using the corresponding simulator parameters. The beam-size k was set manually. 2) HSPaHMM with randomly initialized output models, left-right low-level PaHMMs and top-level duration models whose means (M) were set by summing over the means of the corresponding low-level PaHSMMs in the simulator and set the parameters K and Σ (see section 3.1) manually. 3) PaHMM with the output models initialized randomly and decoding performed by a Viterbi Beam Search. Each model was trained by re-estimating the output models using *Embedded Viterbi learning* described in Section 4 until the slope of the log-likelihood curve fell below 0.01 or when 100 training iterations were completed. We then ran the learned models on the test sequences and obtained accuracy measures using the metric $(N - D - S - I)/N$ where N=number of events in test set, D=no. of deletion errors, S=no. of substitution errors, I=no. of insertion errors.

Since the accuracy as well as complexity of the decoding algorithms depend on manually set parameters (K,Σ for HSPaHMM and k for HPaHSMM) we first investigated their

Algorithm 3 Embedded Viterbi Learning

numTrain = number of training samples
$P_c^t(j|i)$ = Probability of transitioning from state i to state j in channel c.
$P_c^o(o_i|j)$ = Probability of observing symbol o_i in state j in channel c.
$P_c^d(d|i)$ = Probability of spending a duration d in state i in channel c.
for $i = 1$ to $numTrain$ **do**
  jointHMM ← String together HMMs of words/events forming training sequence i.
  repeat
    maxPath ← State sequence of maximum probability path through jointHMM obtained using decoding algorithm.
    n_i^c ← No. of times channel c is in state i in maxPath.
    $n_{i,j}^c$ ← No. of times channel c transitions from state i to state j in maxPath.
    n_i^{c,o_j} ← No. of times o_j is observed in state i channel c in maxPath.
    $n_i^{c,d}$ ← No. of times state i in channel c spends duration d in maxPath.
    Re-estimate parameters using the following equations
    $P_c^t(j|i) \leftarrow n_{i,j}^c / n_i^c$
    $P_c^o(o_i|j) \leftarrow n_j^{c,o_i} / n_j^c$
    $P_c^d(d|i) \leftarrow n_i^{c,d} / n_i^c$
  until convergence
  Split HMMs and update corresponding word/event HMMs.
end for

effects. To do this, we varied the parameters and ran 50 iterations of the train-test setup described above for HSPaHMM and HPaHSMM for each parameter value. Figures 5-6 show these variations.

As can be seen, while increasing Σ in HSPaHMM produces a significant drop in frame rate, it does not affect the accuracy. On the other hand, increasing the beam size(K) produces a significant increase in accuracy at the cost of slower speed. For HPaHSMM, increasing the beam size does not improve accuracy. Based on these observations we ran a set of 50 tests comparing HSPaHMM (with $\Sigma = 1$, $K = 11$), HPaHSMM (with $k = 10$) and PaHMM. Table 1 summarizes the average accuracies and speeds. Thus, HSPaHMM produces a huge jump in performance when compared to plain PaHMM without affecting the speed. While HSPaHMM's accuracy is still lower than HPaHSMM, it is 3 times faster and thus serves as a good mean between HPaHSMM and PaHMM.

Model	Accuracy	Speed
$HPaHSMM$	83.1%(N=124, D=17, S=1, I=3)	12.0
$HSPaHMM$	63.7%(N=124, D=3, S=36, I=6)	40.2
$PaHMM$	4.8%(N=124, D=38, S=53, I=27)	39.1

Table 1: Model Accuracy(%) and Speed(fps)

5.2 Application to Continuous Sign Language Recognition

Sign language recognition, besides being useful in itself, provides a good domain to test hierarchical multi-agent activities; Both hands go through a complex sequence of states simultaneously, each sign has distinct durations, and there is hierarchical structure at the phoneme, word and sentence level.

We experimented with a set of 50 test sentences from a larger dataset used in [Vogler and Metaxas, 1999](provided by Dr.Vogler); the sequences were collected using a $MotionStar^{TM}$ system at 60 frames per second. Vocabulary is limited to 10-words; each sentence is 2-5 words long for a total of 126 signs. The input contains the (x, y, z) location of the hands at each time instant; from these we calculate the instantaneous velocities which are used as the observation vector for each time instant.

We model each word as 2-channel PaHMM (for HSPaHMM) or PaHSMM (for HPaHSMM) based on the *Movement-Hold(MH)* model [Liddell and Johnson, 1989] which breaks down each sign into a sequence of "moves" and "holds". During a "move" some aspect of the hand is changed while during a "hold" all aspects are held constant. The MH model also identifies several aspects of hand configuration like location (chest, chin, etc), distance from body, hand shape, kind of movement (straight, curved, round) etc. With these definitions, we can encode the signs for various words in terms of constituent phonemes. For example, in the word "I", a right-handed signer would start some distance from his chest with all but his index finger closed and end at the chest. This can be encoded in the MH model as $(H(p0CH)M(strToward)H(CH))$, where *p0CH* indi-

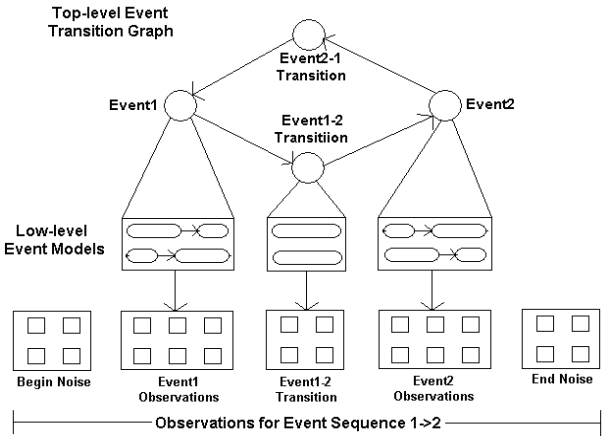

Figure 4: Structure of Event Simulator

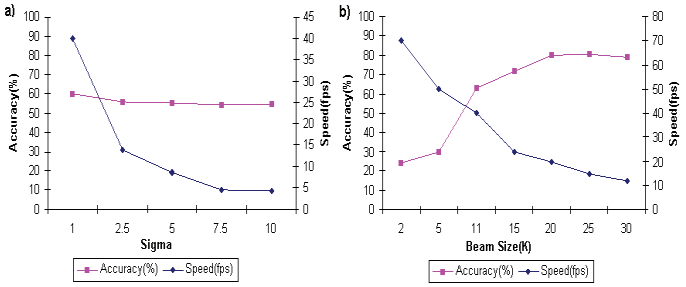

Figure 5: Variation of HSPaHMM Speed(frames/sec or fps) and Accuracy(%) with a)Sigma, Beam Size=11 b)Beam Size, Sigma=1

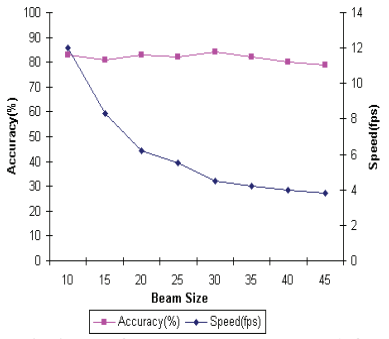

Figure 6: Variation of HPaHSMM Speed(fps) and Accuracy(%) with Beam Size

cates that the hand is within a few inches in front of the chest at the start, *strToward* indicates that hand moves straight perpendicular to the body and *CH* indicates that the hand ends at the chest. Similar transcriptions can be obtained for more complex 2 handed signs by considering both hands as well as hand shape.

We model the observation probabilities in the hold states as a normal distribution with $\mu = 0$ while the move states are modeled as a signum function. Further, we set the inflection point of the signum to be the same as the Gaussian's variance. The intuition behind this choice is that during the hold states the configuration of the hand remains constant with some random noise, while we have a "move" whenever the hand's position changes above the noise threshold during an instant.

We specified the duration models for the move states based on the distance between the starting configuration and the ending configuration and the frame rate. In order to do this we separated the possible hand locations into 3 clusters - those around the abdomen(AB), those around the chest(CH) and those around the face/forehead(FH). We approximately initialized the hold state and intra-cluster transition times by looking at a few samples and set the inter-cluster transition time to be twice the intra-cluster transition time. We modeled the duration as normal distribution centered around these means and $variance = 2.5$ so that the *Th* in the decoding algorithm is reasonably small. For the upper level HSMM in HSPaHMM, we set the means(M) by adding the means of the individual states and set $\Sigma = 2.5$. Thus, we approximately set a total of 4 parameters for the entire set up. Table 2 shows the word accuracy rates. These results indicate

Model	Accuracy	Speed
$HPaHSMM$	78.6%(N=126, D=6, S=17, I=4)	2.1
$HSPaHMM$	67.46%(N=126, D=11, S=22, I=8)	6.4
$PaHMM$	18.25%(N=126, D=7, S=23, I=73)	7.3

Table 2: Word Accuracy Rates(%) and Speed(fps)

that including duration models significantly improves the results. HSPaHMM provides a good high-speed alternative to HPaHSMM. Further, HSPaHMM produces better results than PaHMM because the top-level HSMM restricts the number of word transitions and hence reduces the number of insertion errors. These results were obtained without requiring any additional training data using very simple features. For comparison, existing algorithms for continuous sign language recognition [Vogler and Metaxas, 1999] [Starner *et al.*, 1998] require training sets with 1500-2000 signs for a test set of $\approx$ 400 signs. [Bowden *et al.*, 2004] reports good results on isolated word recognition with a 49 word lexicon that uses just one training sequence per word, but the words in both the test and train sequences are pre-segmented; this is a much easier task than continuous sign language recognition demonstrated here.

6 Conclusion

We have presented a new class of HMMs that provides a hierarchical multi-channel structure with explicit duration models; we focused on two instances - HSPaHMM and HPaHSMM and presented efficient decoding and learning algorithms for them. We rigorously analyzed their performance on synthetic multi-channel time series data and also presented a real application. The methods are not specific for a task domain and should apply to a wide variety of multi-agent activity recognition.

References

[Bowden *et al.*, 2004] R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and M. Brady. A linguistic feature vector for the visual interpretation of sign language. *European Conference on Computer Vision*, 1:91– 401, 2004.

[Brand *et al.*, 1997] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. *Computer Vision and Pattern Recognition*, pages 994–999, 1997.

[Bui *et al.*, 2004] H. Bui, D. Phung, and S. Venkatesh. Hierarchical hidden markov models with general state hierarchy. *Proceedings of the National Conference in Artificial Intelligence*, pages 324–329, 2004.

[Duong *et al.*, 2005] T.V. Duong, H.H. Bui, D.Q. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. *Computer Vision and Pattern Recognition*, 1:838–845, 2005.

[Hongeng and Nevatia, 2003] S Hongeng and R. Nevatia. Large-scale event detection using semi-hidden markov models. *International Conference on Computer Vision*, 2:1455, 2003.

[Liddell and Johnson, 1989] S.K. Liddell and R.E. Johnson. American sign language: The phonological base. *Sign Language Studies*, 64:195–277, 1989.

[Murphy, 2002] K. Murphy. Hidden semi-markov models (hsmms). Technical report, UBC, 2002.

[Starner *et al.*, 1998] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[Vogler and Metaxas, 1999] C Vogler and D.N. Metaxas. Parallel hidden markov models for american sign language recognition. *International Conference on Computer Vision*, pages 116–122, 1999.