

# Coupled Hidden Semi Markov Models for Activity Recognition

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

## Abstract

*Recognizing human activity from a stream of sensory observations is important for a number of applications such as surveillance and human-computer interaction. Hidden Markov Models (HMMs) have been proposed as suitable tools for modeling the variations in the observations for the same action and for discriminating among different actions. HMMs have come in wide use for this task but the standard form suffers from several limitations. These include unrealistic models for the duration of a sub-event and not encoding interactions among multiple agents directly. Semi-Markov models and coupled HMMs have been proposed in previous work to handle these issues. We combine these two concepts into a coupled Hidden semi-Markov Model (CHSMM). CHSMMs pose huge computational complexity challenges. We present efficient algorithms for learning and decoding in such structures and demonstrate their utility by experiments with synthetic and real data.*

## 1 Introduction

Automated systems for modeling and recognizing daily activities can have a wide range of applications such as surveillance, assistive technologies such as sign language recognition and intelligent environments. To this end, a significant amount of research has been devoted to represent, annotate and recognize the activities performed by a user. The key challenges faced by researchers in this area include bridging the gap between the raw sensor readings and the high level semantic concepts that need to be recognized and to model the uncertainties in the observed data.

Several probabilistic models have been proposed over the years in various communities for activity recognition. [1, 2, 3] use static Bayesian Networks (BNs) for mapping a variety of numerical visual properties to event concepts. [4] presents an integrated-plan recognition model based on Dynamic Bayesian Networks (DBNs). Hidden Markov Models (HMMs) and their extensions have by far been the most widely used probabilistic framework in activity recognition as they offer several advantages such as clear Bayesian semantics, efficient learning and inferencing algorithms etc. These works typically extend the basic HMM framework by introducing hierarchical state representation to bridge the gap between low level data and high level semantics or by

factorizing the HMM into multiple channels to model interacting processes or by introducing explicit duration models.

HMMs are basically a class of Dynamic Bayesian Networks (DBN) where there is a temporal evolution of nodes. A HMM model  $\lambda$  is specified by the tuple  $(Q, O, A, B, \pi)$  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  $\pi$  is the initial state distribution. It is straightforward to generalize this model to continuous (like gaussian) output models.

[5] introduced the hierarchical hidden Markov model (HHMM) to model complex multi-scale structure by including a hierarchy of hidden states and [6] adopted it for activity recognition. The abstract hidden Markov model (AHMM) [7] describes an extension where each state of the HMM depends upon a hierarchy of actions.

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 ( $\lambda'$ ) can be specified by the tuple  $(Q, O, A, B, D, \pi)$  [3] applied HSMMs to recognize video events and also presented an algorithm to reduce inference complexity under certain assumptions. [8] presented the switching hidden semi-Markov model (S-HSMM) which is a two-layered extension of HSMM and applied to activity recognition and abnormality detection.

In the basic HMM, a single variable represents the state of the system at any instant. However, many interesting processes 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 ( $S_t = S_t^1, \dots, S_t^C$ ). In their most general form, such extensions can be represented as  $\lambda'' = (Q^C, O^C, A^C, B^C, \pi^C)$  where  $Q^i$  and  $O^i$  are the possible states and observations at channel  $i$  respectively and  $\pi^i$  represents the initial probability of channel  $i$ 's states.  $A^C$  contains the transition probabilities over the composite states ( $P([q_{t+1}^1, \dots, q_{t+1}^C] | [q_t^1, \dots, q_t^C])$ ), and  $B^C$  contains the observation probabilities over the composite states ( $P([o_t^1, \dots, o_t^C] | [q_t^1, \dots, q_t^C])$ ). 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. Factorial hidden Markov models (FHMM) [9] factor

the hidden state into multiple variables which are nominally coupled at the output. This allows factorizing  $A^C$  into  $C$  independent  $N \times N$  matrices ( $N$ =number of states in each channel). Parallel hidden Markov models (PaHMM) [10] factor the HMM into multiple independent chains and hence allow factorizing both  $A^C$  and  $B^C$ . Coupled hidden Markov models (CHMM) [11] factor the HMM into multiple chains where the current state of a chain depends on the previous state of all the chains. In CHMMs, each channel has its own observation sequence and hence  $B^C$  can be factorized. Like FHMM and PaHMM, they allow each channel to evolve independently and hence we have,

$$P([q_{t+1}^1, \dots, q_{t+1}^C] | [q_t^1, \dots, q_t^C]) = \prod_{i=1}^C P([q_{t+1}^i | [q_t^1, \dots, q_t^C])$$

Further, they assume that the interaction between channels  $i$  and  $j$  is independent of the interaction between  $i$  and  $k$ . Hence we can further simplify the transition probabilities-

$$P(q_{t+1}^i | [q_t^1, \dots, q_t^C]) = \prod_{j=1}^C P(q_{t+1}^i | q_t^j)$$

With this simplification, we can have a  $N \times N$  transition matrix for each pair of channels  $i$  and  $j$  and hence factorize  $A^C$  into  $C^2$   $N \times N$  matrices. One issue with this formulation is that the RHS does not sum to 1 over all  $q_{t+1}^c$  and needs to be normalized. Alternate methods [12, 13] for factorizing replace the joint probability with a weighted sum of the marginal probabilities-

$$P(q_{t+1}^c | [q_t^1, \dots, q_t^C]) = \sum_{c'=1}^C \theta_{c,c'} P(q_{t+1}^c | q_t^{c'})$$

This introduces  $C^2$  additional parameters to learn/set. We chose to use Brand's original formulation as the conditional independence assumption suited our applications and also did not need any additional parameters. We handle the normalization issue by scaling and normalizing the forward and backward variables as described in section 2.

The extensions discussed above address various limitations in the basic HMM framework for modeling daily activities and have been proved to be effective in specific domains. However in most real world problems, we would need models that combine the features of all these 3 classes (hierarchical, semi, multi-channel) of extensions. The SHSMM is a step in this direction and introduced a hierarchical structure with explicit duration models. In our work, we introduce the coupled hidden semi-Markov model (CHSMM) where the HMM has compositional state in both space and time. The combined model produces 20-30% increase in performance on simulated data with a wide range of parameters and  $\approx 55\%$  improvement in our real tests.

The rest of the paper is organized as follows - in section 2 we present the formal definition of CHSMM and efficient learning and inferencing algorithms, in section 3 we demonstrate our model first on simulated time-series data, then in a simulated visual surveillance task and finally for the real task of continuous sign-language recognition. Conclusions and future directions are given in section 4.

## 2 CHSMM: Definition, Learning and Decoding Algorithms

Learning and decoding algorithms for a CHSMM have complexity that is inherently much higher than for an HSMM or a CHMM. We describe relatively efficient algorithms below.

### 2.1 Model Definition

CHSMM is basically a multi-channel HMM similar to CHMM where each channel has states with explicitly specified duration models. This can be specified by the tuple-

$$\lambda''' = (Q^C, O^C, A^C, B^C, D^C, \pi^C) \quad (1)$$

where the parameters  $Q^C, O^C, A^C, B^C, \pi^C$  are defined as before.  $D^C$  contains a set of parameters of the form  $P(d_i^c = k | q_t^c = i)$ , i.e the probability of state  $i$  in channel  $c$  having a duration  $k$ . Further,  $A^C$  and  $B^C$  can be factorized in any of the methods discussed before. In this paper, we will focus on the *causal coupling* used in CHMMs. Figure 1 shows the relationships between various HMM models discussed.

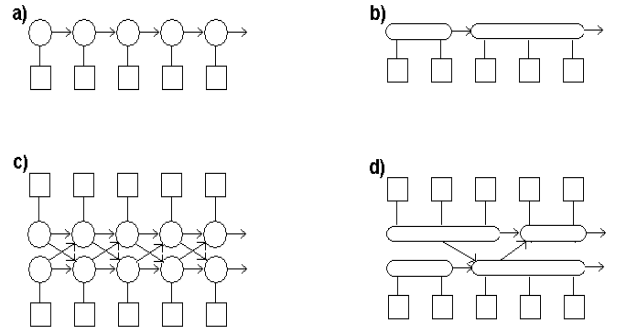


Figure 1: Structure of a)HMM b)HSMM c)CHMM d)CHSMM

### 2.2 Learning Algorithm

For a normal HMM, Baum-Welch algorithm is the standard for parameter re-estimation; it requires calculation of a forward ( $\alpha_{i,t}$ ) and a backward ( $\beta_{i,t}$ ) variable for each state  $i$  and time  $t$ . Thus we need to calculate  $N \times T$  variables and each one requires  $O(N)$  time to compute giving an overall complexity of  $O(TN^2)$ . A direct generalization of this algorithm to CHSMM would require calculation of forward and backward variables of the form  $\alpha_{[i1..iC],[t1..tC]}$  and  $\beta_{[i1..iC],[t1..tC]}$  respectively, where  $[i1..iC]$  and  $[t1..tC]$  are the states and end-times of channels  $1..C$ . Thus we need to compute  $N^C * T^C$  variables and hence the procedure is inherently exponential in  $C$ . In the next paragraph, we describe our polynomial time approximation algorithm that

allows computation of each channel's variables independently.

Let us denote by  $\alpha_{i,ts,te}^c$ , the probability of channel  $c$  being in state  $i$  from time  $ts$  to  $te$ , given the observations  $o_1$  to  $o_{te}$  on channel  $c$ . Also, let  $brick(c, i, tsc, tec)$  denote the probability of channel  $c$  being in state  $i$  for a duration of  $tec - tsc + 1$  and observing the sequence  $(o_{tsc}, \dots, o_{tec})$ . Then we have,

$$brick(c, i, tsc, tec) = p(d_i^c = tec - tsc + 1) \prod_{t=tsc}^{tec} p(o_t = k | q_t^c = i) \quad (2)$$

where,  $p(d_i^c = tec - tsc + 1)$  is the probability of channel  $c$  in state  $i$  having a duration  $tec - tsc + 1$  and  $p(o_t = k | q_t^c = i)$  is the probability of observing  $k$  in channel  $c$  and state  $i$ . Then by assuming that the channels evolve independently and influence each other only during state transitions, the forward variables can be calculated recursively using the following equation-

$$\alpha_{i,ts,te}^c = brick(c, i, ts, te) * \left[ \sum_{ic=1}^N \sum_{ts'=ts-1-Th}^{ts-1} P(q_{ts}^c = i | q_{ts-1}^c = ic) \alpha_{ic,ts',ts-1}^c \right] * \prod_{\substack{c'=1 \\ c' \neq c}}^C \left[ \sum_{ic'=1}^N \sum_{tsc'=ts-Th-1}^{ts-1} \sum_{tec'=tsc'+Th}^{ts-1} P(q_{ts}^c = i | q_{ts-1}^c = ic') * \alpha_{ic',tsc',tec'}^{c'} \right] \quad (3)$$

The second term in (3) corresponds to the intra-channel influences while the third term corresponds to the cross-channel influences. Figure 2 illustrates the forward procedure for a 2-channel CHSMM. As can be seen in order to compute  $\alpha_{i,ts,te}^c$ , we must compute the same channel influences by summing over all  $\alpha$ 's that end at  $(ts - 1)$  and the cross channel influences by summing over all  $\alpha$ 's that contain  $(ts - 1)$ .

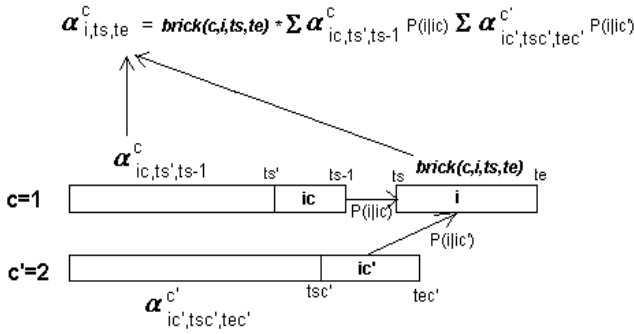


Figure 2: Calculation of  $\alpha_{i,ts,te}^c$

The derivation of (3) is discussed in greater detail in the supplementary material. The RHS takes  $O(CNT^2)$  to calculate, and since there are  $C * N * T^2$  variables to calculate, the overall complexity of the forward algorithm is  $O(C^2N^2T^4)$ . While, this is polynomial, the  $T^4$  complexity is still expensive for long sequences. If it is possible to

restrict  $|te - ts + 1| < Th, \forall c, i$  then, we can reduce the forward algorithm's complexity to  $O(C^2N^2TTh^3)$  and for sufficiently small  $Th$ , the complexity becomes  $O(C^2N^2T)$ . This is possible if the duration model is uniform (where the maximum upper-bound is  $Th$ ), or normal (where 96% of the probability is within  $2\sigma$  of the mean and hence can serve as the bound). Further, by using Brand's argument [14] to choose the maximum probability state as sidekicks, we can reduce the complexity to  $O(C^2NT)$ . The backward variables  $\beta_{i,ts,te}^c$ , can similarly be calculated based on the following equation-

$$\beta_{i,ts,te}^c = brick(c, i, ts, te) * \sum_{ic=1}^N \sum_{te'=te+1}^{te+1+Th} P^c(ic|i) \beta_{ic,te+1,te'}^c * \prod_{\substack{c'=1 \\ c' \neq c}}^C \sum_{tsc'=te-Th}^{te} \sum_{tec'=te}^{tsc'+Th} P^c(i|ic') \alpha_{ic',tsc',tec'}^{c'} \quad (4)$$

Note that for the cross-channel influences we are using the  $\alpha$ -values to weight the transition probabilities because it is the forward variables that affect state transitions.

Using the forward and backward variables, we can re-estimate the parameters as follows with a suitable normalization factor-

1)Initial Probability:

$$\bar{\pi}_i^c = \pi_i^c * \sum_{te=1}^{te=Th} \beta_{i,1,te}^c \quad (5)$$

2)Transition Probability:

$$\overline{P(s_{t+1}^c = i | s_t^c = j)} = \sum_{t=1}^T \left[ \sum_{ts'=t-Th}^t \sum_{te'=t}^{ts'+Th} \alpha_{j,ts',te'}^{c'} * P(s_{t+1}^c = i | s_t^c = j) * \sum_{te=t+1}^{t+1+Th} \beta_{i,t+1,te}^c \right] \quad (6)$$

3)Output Probability:

$$\overline{P(o_k^c | s_t^c = i)} = \sum_{t=1}^T \sum_{\substack{te=t \\ o_t^c = o_k^c \\ t \leq t' \leq te}}^{t+Th} [\alpha_{i,t,te}^c * \sum_{j=1}^N P(s_{te+1}^c = j | s_{te}^c = i) * \sum_{te'=te+1}^{te+1+Th} \beta_{j,te+1,te'}^c] \quad (7)$$

4)Duration Probability:

$$\overline{P(d_i^c = d | s_t^c = i)} = \sum_{ts=1}^T [\alpha_{i,ts,ts+d}^c * \sum_{j=1}^N P(s_{ts+d+1}^c = j | s_{ts+d}^c = i) * \sum_{te=ts+d+1}^{ts+d+1+Th} \beta_{j,ts+d+1,te}^c] \quad (8)$$

We skip a more detailed explanation of the above equations due to space limitations, but it can be seen that they involve parameter re-estimation using the forward and backward variables calculated by summing out the free variables.

Numerical underflow with increasing length of sequences is a crucial problem that must be addressed in any implementation of HMMs. To address this issue in CHSMMs, at each time  $t$  and channel  $c$ , we define a scaling parameter,

$$scale(c, t) = \sum_i \sum_{ts} \alpha_{i,ts,t}^c$$

Then, we scale the forward and backward variables as follows-

$$\hat{\alpha}_{i,ts,te}^c = \frac{\alpha_{i,ts,te}^c}{\prod_{t=ts}^{te} scale(c, t)}; \hat{\beta}_{i,ts,te}^c = \frac{\beta_{i,ts,te}^c}{\prod_{t=ts}^{te} scale(c, t)}$$

With this definition, if  $\bar{\alpha}_{i,ts,te}^c$  and  $\bar{\beta}_{i,ts,te}^c$  are the un-scaled variables, then-

$$\hat{\alpha}_{i,ts,te}^c = \frac{\bar{\alpha}_{i,ts,te}^c}{\prod_{t=0}^{te} scale(c, t)}; \hat{\beta}_{i,ts,te}^c = \frac{\bar{\beta}_{i,ts,te}^c}{\prod_{t=ts}^T scale(c, t)}$$

Now, since the re-estimation equations (5)-(8) involve products of the form,

$$\hat{\alpha}_{i,ts,te}^c * \hat{\beta}_{i,te+1,te'}^c = \frac{\bar{\alpha}_{i,ts,te}^c * \bar{\beta}_{i,te+1,te'}^c}{\prod_{t=0}^T scale(c, t)}$$

the scaling factor can be factored out and hence the re-estimation procedure is unaffected.

### 2.3 Decoding Algorithm

To make an inference in a graphical model, we need an algorithm to find the maximum a posteriori probability path, given the observations. An exact dynamic programming algorithm (see appendix 4) works by calculating variables of the form  $\Delta_{[i1..iC],[ts1..tsC],[te1..teC]}$  which denotes the log-likelihood of the maximum probability path such that channel  $c$  is in state  $ic$  from time  $ts$  to  $te$  for  $c$  in  $[1..C]$ . Then we have,

$$\begin{aligned} \Delta_{[i1..iC],[ts1..tsC],[te1..teC]} &= \\ & \log(brick(c, ic, tsc, tec)) + \\ & \max_{j,ts'c'} (\Delta_{[i1..j..iC],[ts1..ts'..tsC],[te1..tsc-1..teC]}^c + \\ & \log(P^c(ic|j))) + \sum_{\substack{c'=1 \\ c' \neq c}}^C (\log(P(i|ic'))) \end{aligned} \quad (9)$$

By calculating the  $\Delta$ 's for each  $[i1..iC],[ts1..tsC],[te1..teC]$  we can find the MAP path from  $\max_{[i1..iC],[ts1..tsC]} (\Delta_{[i1..iC],[ts1..tsC],[T..T]}^c)$ . This takes  $O(N^C T^{2C+1})$  time and  $O(N^C T^{2C})$  memory since the RHS in (9) takes  $O(NT)$  time for each  $\Delta$  and there are  $O(N^C T^{2C})$  variables to calculate and store. We can simplify this by finding the best path through each channel independently, and approximating the inter-channel interactions by using the highest probability path at that instant as follows - let us denote by  $\delta_{i,ts,te}^c$ , the log-likelihood of the maximum probability path such that channel  $c$  is in state  $i$  from time  $ts$  to  $te$ . Then we have,

$$\begin{aligned} \delta_{i,ts,te}^c &= \log(brick(c, i, ts, te)) + \\ & \max_{j,ts'} (\delta_{j,ts',ts-1}^c + \log(P^c(i|j))) + \\ & \sum_{\substack{c'=1 \\ c' \neq c}}^C \max_{ts',te',ic'} (\delta_{ic',ts',te'}^{c'} + \log(P(i|ic'))) \end{aligned} \quad (10)$$

where,  $ts' \leq ts$  and  $te' \geq te$ .  $\max_{c=1}^C [\max_{i=1}^N (\delta_{i,ts,T}^c)]$  gives the approximate maximum log-likelihood probability and by storing the  $(j, ts')$  values that produce the maximum in equation (10), we can reconstruct the path through the CHSMM. Term 2 in equation (10) models intra-channel influences while term 3 approximates the inter-channel interactions. This algorithm takes  $O(C^2 N^2 T^4)$  and  $O(CNT^2)$  memory and produces satisfactory performance as demonstrated in section 3.

### 2.4 Beam Search for continuous Activity Recognition

The decoding algorithm tries to find the maximum probability path through a CHSMM for a given observation sequence assuming that the sequence is pre-segmented. However in most real applications, the activities occur as part of a much longer stream. In such cases we combine the CHSMMs for each activity to form a compound CHSMM and add additional states for transitions between activities. We then run the decoding algorithm on this compound CHSMM and hence segment the individual activities automatically. However, even for a modest set of activities the state space of the compound CHSMM becomes prohibitively large. Hence at each step, we store only the top  $k$  states and discard the rest giving a time complexity of  $O(C^2 N k T T h^3)$ , where  $N$ =no. of states in compound CHSMM. Typical values for  $k$  are about 5-10% of  $N$ .

## 3 Experiments

To test learning and decoding in CHSMM we conducted three experiments: In the first experiment we compared CHSMM with PaHMM and CHMM on the basis of computation time and accuracy of the models using synthetic data. In the second experiment we tested the performance of our inferencing algorithm using models learned from labeled data in a simulated visual surveillance task. The third is with real data for a sign language (ASL) recognition task. In all cases, we compare our results with alternative methods. All run time results are on a 2GHz Pentium 4 Windows platform with 2GB RAM, running Java programs.

### 3.1 Benchmarks on Synthetic data

We follow the methodology described in [9] and [14] to generate synthetic data. We constructed a discrete event simulator to generate synthetic observation sequences. Each event can have multiple(C) channels/agents; each channel can be in one of  $N$  states at any time. Cartesian product transition matrices are built from random intra and inter-channel transition probabilities, then perturbed with

uniform noise and renormalized so that while there is some interaction between the processes, the overall system is not strictly conformant to our coupling model (i.e. the coupling model in test data is more general). The states have 3-dimensional Gaussian observation models with the means uniformly distributed in [0,1] and the covariances set to I/4. Further, each state has uniform/gaussian duration models with randomly distributed parameters. Random walks through each event model were used to generate forty sequences of five different event models containing hundred observations each. Half of these were used for training and the other half for testing.

The training data was used to train the following multi-channel HMMs: 1)  $CHSMM^{acc}$  with duration models set manually to correct values 2)  $CHSMM^{per}$  where the duration models were initialized after perturbing the parameters from the simulator with  $\approx 30\%$  error 3)  $CHSMM^{ran}$  where the duration models were learned from random initialization 4) PaHMM 5) CHMM. All the parameters were learned from randomly initialized values using the learning algorithm described except for the duration models in models 1 and 2. Each model was trained until the slope of the log-likelihood curve fell below 0.01 or when 100 training iterations were completed. Since [11] had already demonstrated that CHMM outperforms FHMM, LHMM and Cartesian-product HMM in a similar setup, we did not compare our model with them. In order to estimate the recognition accuracy of an HMM-model, we took the test sequence from each event and ran it on the learned models. If the learned model corresponding to the generating event produced the maximum log-likelihood, then the sequence was counted as classified correctly. The ratio of correctly classified sequences to total number of test sequences gives the accuracy for that model.

In the first set of tests, we initialized the event models with  $C=2$  and  $N=5$ , such that the states had uniform duration models and the maximum possible duration( $Th$ ) for a state was 10 and conducted 50 trials using the setup described. We then repeated the experiment with Gaussian duration models. Table 1 shows the average accuracy and learning time for the various HMMs.

As can be seen from these results, all variations of

	Uniform		Normal	
	Accuracy	Time	Accuracy	Time
$CHSMM^{acc}$	97.42%	673s	90.8%	661s
$CHSMM^{per}$	96.49%	213s	89.2%	472s
$CHSMM^{ran}$	91.04%	500s	84.8%	286s
PaHMM	70.90%	53s	69.3%	51s
CHMM	70.77%	173s	61.5%	340s

Table 1: Accuracy values with Uniform and Normal Duration models

CHSMM outperform CHMM and PaHMM. Further, even

if there is a large error( $\approx 30\%$ ) in initializing the duration parameters, there is not a huge change in performance, but there is a significant drop with randomly initialized duration models. In order to understand these results, we calculated the mean and standard deviation of log-likelihood (LL) on training data, mean log-likelihood on test data generated from the event that generated the training data (Class Test), and mean log-likelihood on other test data (Cross Test) as shown in Table 2. Although CHMM has the highest

	Mean TrainLL	$\sigma$ TrainLL	Class Test	Cross Test
$CHSMM^{acc}$	-276.3	41.8	-292.3	-406.1
$CHSMM^{per}$	-302.1	47.4	-318.9	-424.9
$CHSMM^{ran}$	-376.2	42.8	-394.6	-504.7
PaHMM	-306.0	57.6	-337.0	-427.6
CHMM	-276.9	118.7	-262.7	-376.3

Table 2: Mean and Variance of log-likelihood

train and test likelihoods on average, it also has the highest variance. Hence there is a higher confusion between class and non-class test sequences resulting in lower classification accuracy. PaHMM has a lower variance, but the class-nonClass separation is also small resulting in lower accuracy. The CHSMM variations on the other hand have low variance and high class-nonClass separation resulting in high accuracy rates. This also indicates that the mean likelihood probability in itself is not a suitable metric to measure a model/algorithm's classification capability.

We repeated these experiments by varying T, Th, N. The accuracy values were in these ranges -  $CHSMM^{acc}$  (90-100%),  $CHSMM^{per}$  (85-95%),  $CHSMM^{ran}$  (80-90%), PaHMM (65-80%), CHMM (60-75%). Although variation in T, Th and N did not affect the accuracies, increase in Th produced a significant increase in the computation time for  $CHSMM^{acc}$ . This is expected as the complexity of the learning and inferring algorithms vary with  $Th^3$ . Also, CHMM has surprisingly high learning time because it takes many more iterations to converge. Figure 3 shows these variations.

These results indicate that incorporating explicit dura-

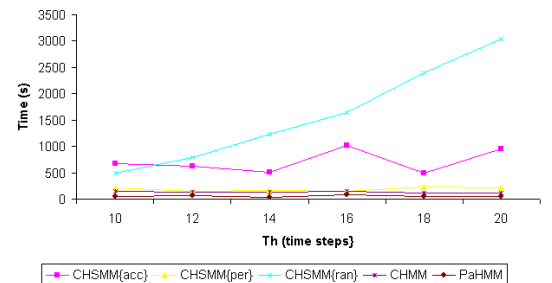


Figure 3: Learning Time Vs Th

tion models produce a 20-30% improvement in classification accuracy over CHMM and PaHMM while modeling multiple interacting processes. Further, the learning algorithm is not sensitive to initialization of transition and observation probabilities, but some prior initialization of duration probabilities produces much better models. These improvements come at the cost of increased computation time, which can be partly offset by restricting the memory(Th) of the models.

### 3.2 Modeling Human Interactions

In this experiment, we demonstrate the ability of our inferencing algorithm to accurately segment state durations. We built a simulator similar to [15] to simulate 3 different events between 2 interacting agents. Each agent can be in one of 5 states - walk at normal speed, walk slowly, run (walk fast), stand (chat) and change direction. The speed and change in direction from the previous time step were used as observations. Further, the observation model for each state was normally distributed around the mean speed for that state (e.g average human walking speed=1.2m/s). We defined 3 interaction events between the agents - *follow*, *approach+talk+continue separately*, *approach+talk+continue together*. Figure 4 shows the sequence of states in which the agents are in each of these events. The duration models for the agent states in each

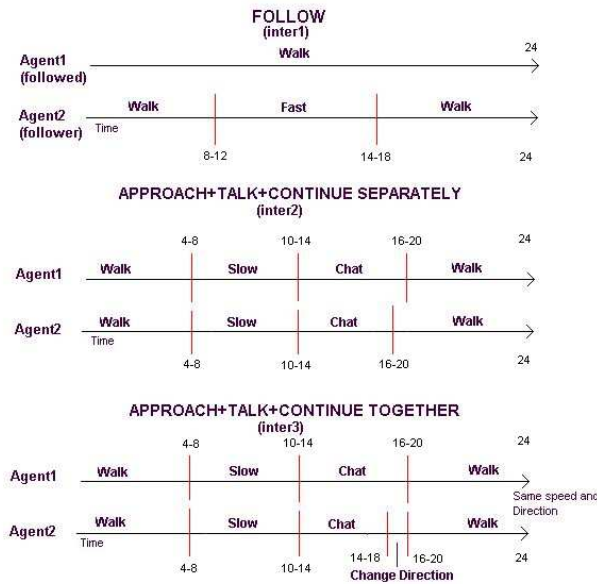


Figure 4: Timeline of modeled Human Interactions

event were uniformly distributed in the ranges shown in the figure. For example in the *follow* event, agent2 spent approximately 1/3rd of the total time in the first walk phase, the next 1/3rd in fast-walk state and the final 1/3rd in the walk state, with a duration noise of about 4 time steps. We

then generated test and train sequences of length 24 for each event and trained CHSMM and CHMM models similar to experiment 1. CHSMM consistently had  $\approx 100\%$  accuracy for all events with all values of the simulator’s parameters. [15] report 90-100% accuracy in a similar set up. However in our tests CHMM’s performance varied between 65-100% depending on the initialization of parameters indicating that CHMM is sensitive to initialization. Since the possible sequence of states in the above event definitions are quite constrained, we can learn the parameters by a histogram analysis of the state sequences that generated the observations. We implemented such a learning algorithm and tested the state segmentation capability of our inferencing algorithm. The CHSMM models always produced nearly perfect state segmentation, while CHMM models had poor results. This is primarily due to the fact that CHSMM allows explicit specification of duration models. Figures 5-6 show the state segmentation results for CHSMM and CHMM respectively for a particular instance of INTER3. The left figure shows the segments for the first agent and the right for the second agent.

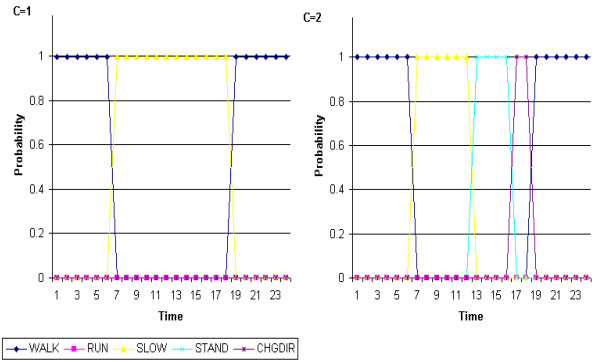


Figure 5: CHSMM State Segmentation for INTER3

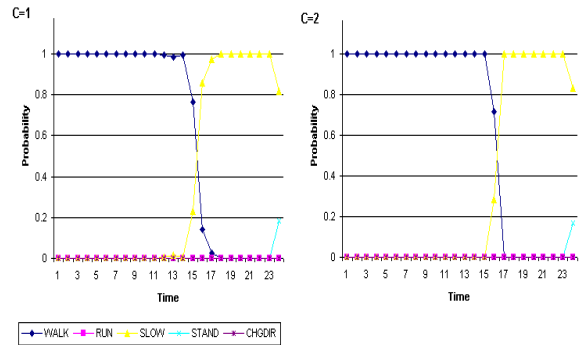


Figure 6: CHMM State Segmentation for INTER3

### 3.3 Application to continuous sign language recognition

Sign language recognition, besides being useful in itself, provides a good domain to test multi-agent activities; both hands go through a complex sequence of states simultaneously and the configuration of one hand has a significant influence on the other. Further there can be a significant variation in the numerical tracks of the same sign between different signers and also between different instances for the same sign acted by a signer. In the rest of this section first, we provide a brief summary of ASL representations, then how we encode the problem in various HMMs and finally present the recognition results.

Like spoken languages ASL has a limited number of phonemes. [16] describes a *segmental model* which breaks down each sign into a sequence of "moves" and "holds". Moves are segments where some aspect of the signer's hand configuration changes while holds are those when all aspects of the hand configuration are unchanged. [16] 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, it is possible to 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 motion of the right hand can be encoded in the MH model as  $(H(p0CH)M(strToward)H(CH))$ , where  $p0CH$  indicates 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.

For our experiments, we used a set of 50 test sentences from a larger dataset used in [17]; the sequences were collected using a combination of a vision-based tracking system and a *MotionStar<sup>TM</sup>* system at 60 frames per second. We have a vocabulary of 10-words and 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 a 2-channel CHSMM, with one channel for each hand. Further, each state of each channel corresponds to a phoneme in the MH transcription of that word and set the initial and transition probabilities with these constraints. We model the observation probabilities using a normal distribution with  $\mu = 0$  for the hold states and a signum function for the move states. 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 set the variance of the hold states' normal distribution and the inflection point of the move states' signum distribution to a single value based on a simple analysis of a few sequences. 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. Further, in order to allow for some variation in individual instances of these moves, 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. Thus, we approximately set a total of 4 parameters for the entire set up.

We then combine these word-CHSMMs to form a compound CHSMM and add additional states for every possible word transition. We ran the beam search algorithm (with  $N \approx 600$ ,  $k=20$ ) on the combined CHSMM for each test sequence without any additional training/tuning. Table 2 shows the word accuracy rates for CHSMM. For comparison we have included PaHMM's accuracy rates as well for the same feature set and output model and the transition probabilities were set using a flat start. We did not include CHMM's accuracy rates as the approximate decoding algorithm presented in [14] assumes that the probability mass at each channel is concentrated in a single state while calculating channel interactions. This assumption is not valid in our domain as many states can have nearly equal probabilities in the initial time steps and choosing only one of them prunes out the other words resulting in extremely poor performance.

Model	Accuracy	Details
<i>CHSMM</i>	83.3%	$N = 126, D = 5, S = 14, I = 2$
<i>PaHMM</i>	18.25%	$N = 126, D = 7, S = 23, I = 73$

Table 3: Word Accuracy Rates: N=no. of words, D=no. of deletion errors, S=no. of substitution errors, I=no. of insertion errors, Accuracy=(N-D-S-I)/N

These results were obtained without requiring any additional training data using very simple features. For comparison, existing algorithms for continuous sign language recognition [10, 18] require training sets with 1500-2000 signs for a test set of  $\approx 400$  signs enacted by a single signer. Such large training sets are impractical to obtain for real sign and action recognition applications. [19] 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.

## 4 Conclusion and Future Work

We have presented a new graphical model, CHSMM, and efficient algorithms for learning and decoding algorithms for this model. We have demonstrated the advantages of the new formalism by rigorous analysis on synthetic data and also on a real application (continuous ASL recognition). These results are promising and we plan to build on our current work in two directions - on the implementation side we plan to apply these methods to a larger set of activity recognition problems in a variety of domains. On the theoretical side, we plan to develop a hierarchical version of CHSMM that can recognize at higher levels of abstraction.

## References

- [1] P. Remagnino, T. Tan, and K. Baker. Multi-agent visual surveillance of dynamic scenes. *IVC*, 16:529–532, 1998.
- [2] S. Intille and A. Bobick. Recognizing planned, multi-person action. *Computer Vision and Image Understanding*, 81(3):414–445, 2001.
- [3] S Hongeng and R. Nevatia. Large-scale event detection using semi-hidden markov models. *International Conference on Computer Vision*, 2:1455, 2003.
- [4] J. Yin, X. Y. aaai, and Q Yang. High-level goal recognition in a wireless lan. *Proceedings of the National Conference in Artificial Intelligence*, pages 578–584, 2004.
- [5] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, page 32, 1998.
- [6] 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.
- [7] H. Bui, S. Venkatesh, and G. West. Policy recognition in the abstract hidden markov model. *Journal of Artificial Intelligence Research*, pages 17:451–499, 2002.
- [8] 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.
- [9] Z Ghahramani and M.I. Jordan. Factorial hidden markov models. *Advances in Neural Information Processing Systems*, 8, 1996.
- [10] C Vogler and D.N. Metaxas. Parallel hidden markov models for american sign language recognition. *International Conference on Computer Vision*, pages 116–122, 1999.
- [11] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. *Computer Vision and Pattern Recognition*, pages 994–999, 1997.
- [12] J. Ghosh. S. Zhong. Hmms and coupled hmms for multi-channel eeg classification. *IEEE Int. Joint Conf. on Neural Networks*, pages 1154–1159, 2002.
- [13] M. Jordan L. Saul. Mixed memory markov models: decomposing complex stochastic processes as mixtures of simpler ones. *Machine Learning*, pages 75–87, 1999.
- [14] M. Brand. Coupled hidden markov models for modeling interacting processes. Technical Report 405, MIT Media Lab Vision and Modeling, 1996.
- [15] Barbara Rosario, Nuria Oliver, and Alex Pentland. A synthetic agent system for bayesian modeling human interactions. In *Proceedings of the Third International Conference on Autonomous Agents (Agents’99)*, pages 342–343, Seattle, WA, USA, 1999. ACM Press.
- [16] S.K. Liddell and R.E. Johnson. American sign language: The phonological base. *Sign Language Studies*, 64:195–277, 1989.
- [17] Christian Vogler and Dimitris N. Metaxas. Handshapes and movements: Multiple-channel american sign language recognition. In *Gesture Workshop*, pages 247–258, 2003.
- [18] 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.
- [19] Bowden R, Windridge D, Kadir T, Zisserman A, and Brady M. A linguistic feature vector for the visual interpretation of sign language. *European Conference on Computer Vision*, 1:91– 401, 2004.