Activity Recognition using Dynamic Bayesian Networks with Automatic State Selection

Justin Muncaster
Computer Science Department,
University of California, Santa Barbara
jmunk@cs.ucsb.edu

Yunqian Ma
Honeywell Labs, 3660 Technology Drive,
Minneapolis, MN 55113
yunqian.ma@honeywell.com

Abstract

Applying advanced video technology to understand activity and intent is becoming increasingly important for intelligent video surveillance. We present a general model of a d-level dynamic Bayesian network to perform complex event recognition. The levels of the network are constrained to enforce state hierarchy while the d^{th} level models the duration of simplest event. Moreover, in this paper we propose to use the deterministic annealing clustering method to automatically discover the states for the observable levels. We used real world data sets to show the effectiveness of our proposed method.

1. Introduction

Activity recognition in video analytics has been developed fast in recent years. There are a wide range of applications of activity recognition. Security applications are most prevalent, including the identification of terrorist actions and threats and the monitoring of individuals within shopping stores, office buildings, hospitals, and banks. Activity recognition attempts to extract semantic activity information from low level video processing. A variety of algorithms exist for processing of video sequences for key low-level functions, such as motion detection and object tracking, allowing one to identify simple activities such as walking or running [1,2].

Besides atomic activity recognition, recently researchers have shown interest in recognizing more complex behaviors such as loitering or fighting. Such events typically require several sub events that occur in sequence before the complex event can be correctly decided and many times only slight variations in sequences distinguish one complex event from another. For example, suppose we would like to recognize events in a retail store using the following three categories: Buying item, stealing item, and browsing items. One might choose to decompose these events into lower-level events. One might define buying an item in terms of the lower level events holding an item, taking item to cashier, paying for item. We cannot distinguish between the three high level events by simply looking at any one atomic event (say, holding the item). Instead, one must consider the sequence of sub-events (e.g., holding item, concealing item, exiting store) to be able to recognize a complex behavior such as stealing. The decomposition of complex events into sequences of simpler events suggests a hierarchical representation for events.

There are many papers that have tried to classify activities at the semantic level. In [3], Hongeng et al. recognizes multi-state activities using a hidden Markov model (HMM) and a particular form of static Bayesian network; [4] defines a series of rules, e.g. entry violation, escort, theft, possess, belong; Vu et al. [5] uses a declarative model and a logic based approach to recognize predefined activities. Cupillard et al. [6] use a finite state automation to recognize 'a group fighting' event. Ryoo and Aggarwal [7] use a context-free grammer based representation to represent composite actions and interactions. Duong et al. [8] use a switching duration hierarchical semi-Markov movel (S-HSMM) to model complex activities and they imposed Coxian distribution for duration model of the states.

For the low-level events, if using a rule-based approach, there usually exist a multitude of possible low-level sequences for which it is difficult to define appropriate rules. Additionally, for any high level event there are a large number of low-level sequences that may define the high level event, some of which are more likely than others. This suggests the need for a model rich enough to represent and score hierarchical sequences in a reasonable fashion, without the need to exhaustively consider rules for each possibility.

In this paper, we consider the model in [8], which we believe to be a strong approach. We believe that the important aspect of the S-HSMM [8] is the hierarchy itself and that all other added dependencies are specific to the domain. This will motivate us to start at the basic hierarchical model and augment it from there.

In [8] the authors modeled observations by partitioning the feature space into Q^2 discrete states, each corresponding to a cell on a $Q \times Q$ grid. The observed features corresponded to a track location and they were mapped to the square region of space where the

observation was located. There are some shortcomings of this approach:

- There may potentially be many states that rarely or never get observed and will consequently lack training data.
- The approach is insensitive to variations within a cell.
- Noisy track estimates at the edges of the cell boundaries can cause errors in the observed cell.

Although the $Q \times Q$ grid in [8] is intuitive, the partition is somewhat arbitrary and is not in any way based on training data. Because of this, we believe, a highly variant portion of the space may be represented by only a single cell and many cells may cover an uninteresting portion of the space. Also it is assumed one observes the low-level region itself rather than a tracker's noisy estimate of the low-level region. This could be problematic when the tracker reports a position that is near the boundary of two regions because it is quite possible that the estimate is a mistake.

In this paper, we view the problem of defining low level events as one of finding a discrete number of representative clusters in a continuous feature space. We believe that such clusters are natural representation of low level events. Recent studies also use the assumption that similar activities in video sequences are clustered close to each other and different activities are far apart. For example, in [9] the authors show that the video sequences containing sitting, crawling, and standing up are close to their corresponding reference activity sequence.

In this paper, our major contribution resides on two parts. First, we propose a (Dynamic Bayesian Network) DBN framework in which we systematically add dependencies in order to keep the number of parameters reasonably tractable. We propose to use the minimum complexity necessary for hierarchy and then to add model complexity to the DBN framework only as needed. We divide the DBN into several levels, each representing events at a different granularity. Lower level states represent the atomic activities that make up increasingly complex events. Finally, we recognize the last level as the duration model to account for the fact that different atomic activities take varying amounts of time.

Our second contribution is that we address the problem of having to explicitly define the states of lower-level variables. Often training data is only available for the highest level (the activity we which to recognize) and lower-level activities are left unlabeled. In this paper we propose to use deterministic annealing [10], an automatic, unsupervised technique, to discover low-level states and eliminate the need of arbitrarily partitioning the space of low-level activities.

2. DBN framework for activity recognition

We begin with a probabilistic model of activities. For a set of N random variables $X^{(1:N)} = \left(X^{(1)}, X^{(2)}, ..., X^{(N)}\right)$, the joint probability distribution $\Pr\left(X^{(1:N)} = j\right)$, where j refers[1] to a joint assignment to a set of random variables. Without loss of generality, suppose that the first M variables, $X^{(1:M)}$, are hidden and that the remaining variables, $X^{((M+1):N)}$, are observed. Inference is then defined to be the computation of the marginal probability $\Pr\left(X^{(i)} = j^{(i)} \mid X^{((M+1):N)} = e\right)$.

A Bayesian network is a directed, acyclic graph where each vertex is in one-to-one correspondence with a random variable in the probabilistic model. The lack of edges corresponds to conditional independencies between variables. Using the conditional independencies, the joint distribution is given by:

$$\Pr\left(X^{(1:N)} = j\right) = \prod_{i=1}^{N} \Pr\left(X^{(i)} = j^{(i)} \mid pa\left(X^{(i)}\right) = j^{*}\right),$$

where j^{*} refers to the subset of the assignment to j that pertains to $pa\left(X^{(i)}\right)$. The above factorization efficiently reduces the number of parameters to a manageable number and allows one to perform inference, i.e. compute $\Pr\left(X^{(1)} = j \mid X^{((M+1):N)} = e\right)$, at a manageable speed. While in principle this probability can be computed by "summing out" all of the hidden variables, this quickly grows intractable. Several algorithms (e.g. [11,12]) exist that exploit conditional independence in the model and perform inference efficiently.

A Dynamic Bayesian Network [13,14] is a Bayesian network that is extended to include a variable for some discrete number of time slices T. A DBN is defined using two slices of the standard Bayesian network and adding additional "temporal" edges from nodes in $X_{t-1}^{(1:N)}$ to nodes in $X_{t}^{(1:N)}$ which specify how the variables may change over time. The parameters associated with variables in the "prior" time slice $X_{t-1}^{(1:N)}$ will have parameters that specify the initial distribution $\Pr\left(X_{1}^{(1:N)}\right)$ and the parameters in the "current" time slice, $X_{t}^{(1:N)}$, will be used to define the transition model from the previous state to the current state, or $\Pr\left(X_{t}^{(1:N)} \mid X_{t-1}^{(1:N)}\right)$. For many models we can do inference quickly by using conditional independence. A range of exact and approximate inferences algorithm exist in [14,15,16].

2.1. 'Bare-bone' Hierarchical DBN

In this section we discuss a specific type of DBN, 'bare-bones hierarchical DBN' (HDBN) as our base

[1] When considering an assignment to a set of variables we consider the enumeration of all possible combinations and use a single index to refer to a particular one.

model for activity recognition in Fig. 1a. From this we wish to cast all other dependencies as domain-specific, that is, in section 2.2 we discuss particular constraints and extensions to the model that are suited for activity recognition.

We define a HDBN as a DBN where variables can be grouped into D levels. At a given time each level has two particular variables $X_t^{(d)}$ and $E_t^{(d)}$ which will control the hierarchical structure. At time t the variable $X_t^{(d)}$ represents the state of the system at the d^{th} level and the binary-valued variable $E_t^{(d)}$ represents whether or not the process for the d^{th} level has run until its end. A HDBN then must enforce the following constraints:

- *Blocking constraint:* The state of the d^{th} level cannot change until the $(d+1)^{th}$ level has terminated. That is, for all $d < D$, $E_{t-1}^{(d+1)} = 0 \Rightarrow X_t^{(d)} = X_{t-1}^{(d)}$.
- *Termination constraint:* The d^{th} level may not terminate until the $(d+1)^{th}$ level has terminated. This is to say that for all $d < D$, $E_t^{(d+1)} = 0 \Rightarrow E_t^{(d)} = 0$.

The above constraints give a hierarchical representation to the state sequences. The constraints can be represented in a DBN by adding dependencies and requiring certain parameter values in the conditional probability tables. The dependencies are specified by the following edges:

$$\left\{ X_{t-1}^{(d)} \rightarrow X_t^{(d)}, E_t^{(d+1)} \rightarrow E_t^{(d)}, E_{t-1}^{(d+1)} \rightarrow X_t^{(d)} \right\}$$

And the constraints are enforced with the following constraints on conditional probability table (CPT) values for all $d < D$:

$$\Pr\left(X_t^{(d)} = j \mid X_{t-1}^{(d)} = i, E_{t-1}^{d+1} = 0 \right) = \delta_{i,j}, \qquad (1)$$

$$\Pr\left(E_t^{(d)} = 0 \mid E_t^{(d+1)} = 0 \right) = 1.$$

Where $\delta_{i,j} = 1$ if $i = j$ and equals 0 otherwise. We show an HDBN with additional dependencies in Fig. 1b [14,17]. To be an HDBN for the general case where variables may have additional dependencies, the CPT satisfies for $d < D$:

$$\Pr\left(X_t^{(d)} = j \mid X_{t-1}^{(d)} = i, E_{t-1}^{(d+1)} = 0, pa^*\left(X_t^{(d)}\right) = \cdot \right) = \delta_{i,j} \quad (2)$$

$$\Pr\left(E_t^{(d)} = 0 \mid E_t^{(d+1)} = 0, pa^*\left(E_t^{(d)}\right) = \cdot \right) = 1.$$

where $pa^*(X)$ denotes the rest of the parents of X that have not already appeared in the list of conditioning variables.

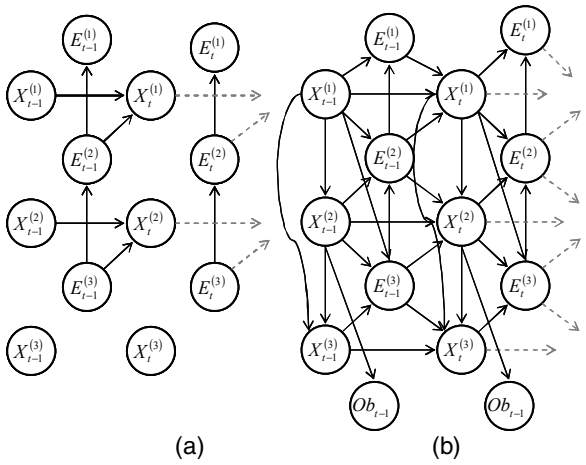

Figure 1: (a) The bare-bones HDBN (b) A more typical example of an HDBN/HHMM.

2.2 Proposed framework

We extend a 3-level the 'bare bone' HDBN (Fig.1a) with various constraints effective in modeling activities. We define the semantics of each level and add appropriate dependencies in Fig. 2. Our first level X_t^{HL} represents the high-level activity that we are attempting to classify. The number of states for the high-level activity will depend on the particular domain. The second level X_t^{LL} will represent the low-level activity that is occurring and generating observations. It is always hidden and will not be labeled in the training data, however we expect to have an observed variable, Y_t^{LL}, that depends on it. The last level X_t^{PH} represents further subdivisions of the low-level activity and serves as the duration model for the low-level activity. By requiring that each low-level activity go through a sequence of *phases*, we are able to model varying durations for low-level events in a probabilistic manner.

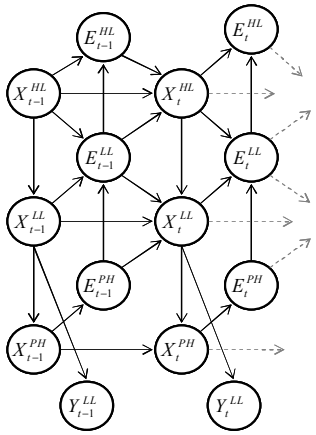

Figure 2: Model specific HDBN for activity recognition.

i. End of each level depends on the state of the level

For each level d we will add the edges $X_t^{(d)} \rightarrow E_t^{(d)}$, which assumes that the end-of-sequence marker depends on the state of the sequence. This will complete the dependencies for the end of the high-level sequence, E_t^{HL}. The CPT is then

$$\Pr\left(E_t^{HL} = 1 \mid X^{HL} = i, E_t^{LL} = 1\right) = p_{i \rightarrow end}^{HL}. \quad (3)$$

This parameter represents the probability of ending a high-level sequence given that we are at high-level state i and all of the previous low-level states have ended. This parameter will be given a value during the training phase.

ii. End of low-level sequence depends on the higher-level context

We will add an edge $X_t^{HL} \rightarrow E_t^{LL}$ to model our assumption that whether or not a particular low-level event marks the end of a sequence depends on the high-level state. This dependency may exist when two high level events both have the same low level event in their sequences. For example, a stopped track may mark the end of the low level sequence for one high-level activity yet may not mark the end of a low-level sequence for another. With this dependency the CPT for E_t^{LL} will have the following parameters:

$$\Pr\left(E_t^{LL} = 1 \mid X_t^{LL} = i, X^{HL} = k, E_t^{PH} = 1\right) = p_{i \rightarrow end}^{LL,k} \quad (4)$$

where $p_{i \rightarrow end}^{LL,k}$ represents the probability of ending the low-level sequence from state i while in context k.

iii. Lower-level states depend on immediate context

We will add the edge $X_t^{(d)} \rightarrow X_t^{(d+1)}$ for $d < D$ to model the assumption that the way a lower level event transitions depends directly on the level immediately above it. As seen in [14], frequently the lower levels also depend on even higher level context (i.e. X_t^{PH} would depend on X_t^{HL}), however we do not assume this our activity model. This offers a major savings in the number of parameters to learn, and we feel that dependencies from $X_t^{(d)}$ to $X_t^{(d+\ell)}$ with $\ell > 1$ should not be added unless the particular domain warrants it. From this we have the following prior distributions for each level:

$$\Pr\left(X_1^{HL} = j\right) = \pi_j^{HL},$$

$$\Pr\left(X_1^{LL} = j \mid X_1^{HL} = k\right) = \pi_j^{LL,k},$$

$$\Pr\left(X_1^{PH} = j \mid X_1^{LL} = k\right) = \pi_j^{PH,k}.$$

Given that the lower-level sequence is not finished, we have the following transition parameters for transition from high-level state i to state j:

$$\Pr\left(X_t^{HL} = j \mid X_{t-1}^{HL} = i, E_{t-1}^{LL} = 1\right) = a_{i \rightarrow j}^{HL}.$$

iv. Low level and high level transition probabilities depend on whether the sequence just ended.

We will place the edges $E_{t-1}^{HL} \rightarrow X_t^{HL}$ and $E_{t-1}^{LL} \rightarrow X_t^{LL}$ to model the assumption that our transition probabilities are different if we just ended a sequence. This is assumption is built into each level of the HHMM model and we believe that it fits for the low-level and high-level activity modeling. We believe this assumption is appropriate and that if a sequence has ended then the probability distribution for starting a new sequence should be different than that of normal transitions. This gives the following transition model parameters for the high-level:

$$\Pr\left(X_t^{HL} = j \mid X_t^{HL} = i, E_{t-1}^{HL} = 0, E_{t-1}^{LL} = 1\right) = a_{i \rightarrow j}^{HL},$$

$$\Pr\left(X_t^{HL} = j \mid X_t^{HL} = i, E_{t-1}^{HL} = 1, E_{t-1}^{LL} = 1\right) = \pi_j^{HL}.$$

The low-level CPT is similar, however it also depends on the high level context:

$$\Pr\left(X_t^{LL} = j \mid X_t^{LL} = i, X^{HL} = k, E_{t-1}^{LL} = 0, E_{t-1}^{PH} = 1\right) = a_{i \rightarrow j}^{LL,k},$$

$$\Pr\left(X_t^{LL} = j \mid X_t^{LL} = i, X^{HL} = k, E_{t-1}^{LL} = 1, E_{t-1}^{PH} = 1\right) = \pi_j^{LL,k}.$$

Here, in high-level context k, $a_{i \rightarrow j}^{LL,k}$ is the transition probability of going from state i to state j and $\pi_j^{LL,k}$ is the probability of starting a sequence in state j.

v. Lowest-level follows a Coxian phase-distribution that depends on low-level context

To model complex durations of low-level events we use the discrete Coxian distribution. This means that every low level event must pass through some number of *phases* before it can transition to the next low level event. We fix the number of phases n^{PH} for each low-level event. Next, we insist that the phase model ends exactly when it reaches the final phase. This gives the parameters:

$\Pr\left(E_t^{PH} = 1 \mid X_t^{PH} = i\right) = \delta_{i,n^{PH}}$, and,

$$\Pr\left(X_t^{PH} = j \mid X_{t-1}^{PH} = i, X_t^{LL} = k\right) = \begin{cases} 1 - a_{i \rightarrow i+1}^{PH,k} & j = i, i < n^{PH} \\ a_{i \rightarrow i+1}^{PH,k} & j = i+1, i < n^{PH} \\ \pi_j^{PH,k} & i = n^{PH} \\ 0 & otherwise \end{cases} \quad (5)$$

where, in context k, $p_{i \rightarrow i+1}^{PH,k}$ is a free parameter representing the probability of advancing to the next phase from phase i and $\pi_j^{PH,k}$ is the probability of starting the sequence in phase j.

This representation of the Coxian distribution is slightly different from the HHMM model of [8] because we have a state devoted to flagging the end of the phase-distribution. The important thing, however, is that we have the same effect of modeling complex duration distribution for the low-level event. Since we never observe the phase it doesn't matter that we have a distinct final state devoted to the end of the sequence. We choose this formulation due to the simpler formulas and reduction in the sizes of CPTs for X_t^{PH} and E_t^{PH}.

vi. Observations are multivariate Gaussian and are based on the low-level state.

To perform inference on our model we will need some observations. Thus we add a new variable, Y_t^{LL}, which represents an observation of the low level event. Typically, our observations are some continuous-valued feature-vector, $\mathbf{y}_t^{LL}$. During the training phase we will take all of our feature vectors and partition the feature space into clusters, where each cluster represents a low-level state. We then assume that a particular observation, $\mathbf{y}_t^{LL}$, is drawn from a Gaussian distribution, whose probability distribution is given by a conditional Gaussian density

$$\Pr\left(Y_t^{LL} = \mathbf{y}_t^{LL} \mid X_t^{LL} = k\right) = N\left(\mathbf{y}_t^{LL}; \boldsymbol{\mu}_k^{LL}, \boldsymbol{\Sigma}_k^{LL}\right) \quad (6)$$

where $\boldsymbol{\mu}_k^{LL}$ is a mean vector and $\boldsymbol{\Sigma}_k^{LL}$ is a covariance matrix for context k. These parameters can be readily estimated by our cluster methods in section III.

3. State selection using deterministic annealing

To calculate $\boldsymbol{\mu}_k^{LL}$ and $\boldsymbol{\Sigma}_k^{LL}$ in formula (6), we are given a large set of feature points from our training data as illustrated in Fig. 3(a). We run a clustering algorithm (deterministic annealing) to find several representative points as shown in Fig. 3(b). These clusters will each correspond to a low-level state in our model, illustrated in (c). So, we will use the cluster results to estimate our observation likelihoods $\Pr\left(Y_t^{LL} = \mathbf{y}_t^{LL} \mid X_t^{LL} = k\right)$ by fitting a Gaussian to each cluster, shown pictorially in (d).

Following [10], we define the clustering problem as follows. We are given a distortion measure $d(\cdot,\cdot)$ and a set of training data Ω. We wish to find a set of up to k cluster centers $C = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k\}$ so as to minimize

$$D = \sum_{\mathbf{y} \in \Omega} \Pr(\mathbf{y})\, d(\mathbf{y}, NC(\mathbf{y})), \quad (7)$$

Where $NC(\mathbf{y})$ denotes the nearest cluster center to $\mathbf{y}$. Deterministic annealing attempts to find an assignment of cluster centers that will find a global minimum to the above distortion function. To do this the technique relaxes the problem constraints and allows each point to be assigned to *all* clusters with some probability so as to minimize a modified distortion function. To prevent points from always being assigned to the nearest neighbor (thus minimizing D), an additional constraint is added on the entropy the assignments. This extra constraint forces points to be assigned in some degree to all cluster centers.

The random assignments and the entropy constraint introduce a notion of randomness that allows the algorithm to explore many solutions. The amount of randomness is controlled by a "temperature" term, T, which starts out high, decreases to a minimum temperature $T_{\min}$, and then ends by clustering at $T = 0$. Initially, this forces each point to be equally assigned to all cluster centers. At $T = T_{\min}$ we will have fixed the number of clusters and we run one more iteration at zero temperature where D is minimized by a "hard" assignment, giving us a solution to the original problem. For details regarding deterministic annealing, see [10].

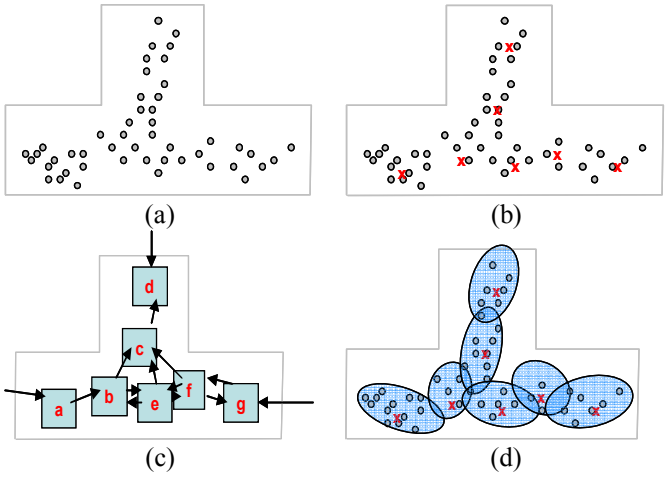

Figure 3: Using clustering to define low-level events. (a) We have as input many points in the feature space. (b) We run a clustering algorithm to partition the space. (c) Each cluster center represents a low level state from which we can learn transition probabilities. (d) We fit a Gaussian to each cluster center to represent the observation likelihood for each low-level event.

Combining Section 2.2 and deterministic annealing above, our proposed algorithm for training the model is given in Table 1. The inputs are:

- Number of high-level states, n^{HL}.
- Max number of low-level states, n^{LL}.
- Number of phases for each low-level state, n^{PH}.
- Minimum annealing temperature $T_{\min}$.

- A set of L training sequences, $S = \left\{ Y_{1:\ell_1}^1, Y_{1:\ell_2}^2, \ldots, Y_{1:\ell_L}^L \right\}$.

Where $Y_t^i = \left(x_t^{HL}, \mathbf{y}_t^{HL} \right)$ is an observation at time t in the i^{th} sequence.

Table 1: Proposed Algorithm

1. Cluster features using deterministic annealing with maximum number of clusters $n_{\max}^{LL}$ and minimum temperature $T_{\min}$

$$\left(\boldsymbol{\mu}_{1:n^{LL}}^{LL}, \boldsymbol{\Sigma}_{1:n^{LL}}^{LL}, n^{LL} \right) \leftarrow clusterDA\left(n_{\max}^{LL}, T_{\min} \right)$$

2. Create 3-level HDBN model with model with $X_t^{HL}, X_t^{LL}, X_t^{PH}$ representing the state of each level and $E_t^{HL}, E_t^{LL}, ..., E_t^{PH}$ representing the end-of-sequence for each level.

a. Augment basic structure and set number of states

$$bnet \leftarrow createDbnModel\left(n^{HL}, n^{LL}, n^{PH} \right)$$

b. Set conditional probability distribution prior probabilities for EM algorithm. Initialize unknown parameters to random values.

i. Observations: Use $\boldsymbol{\mu}_{1:n^{LL}}^{LL}$ and $\boldsymbol{\Sigma}_{1:n^{LL}}^{LL}$ as parameters to Gaussian distributions.

ii. Initial time slice

$$\Pr\left(X_{t-1}^{HL} = j \right) = \pi_j^{HL}$$

$$\Pr\left(X_{t-1}^{LL} = j \mid X_{t-1}^{HL} = k \right) = \pi_j^{LL,k}$$

$$\Pr\left(X_{t-1}^{PH} = j \mid X_{t-1}^{LL} = k \right) = \pi_j^{PH,k}$$

iii. Transition probabilities:

$$\Pr\left(X_t^{HL} = j \mid X_{t-1}^{HL} = i, E_{t-1}^{LL} = e^{LL}, E_{t-1}^{HL} = e^{HL} \right) = \begin{cases} \delta_{i,j} & e^{LL} = 0 \\ a_{i \to j}^{HL} & e^{LL} = 1, e^{HL} = 0 \\ \pi_j^{HL} & e^{LL} = 1, e^{HL} = 1 \end{cases}$$

$$\Pr\left(X_t^{LL} = j \mid X_{t-1}^{LL} = i, X_t^{HL} = k, E_{t-1}^{PH} = e^{PH}, E_{t-1}^{LL} = e^{LL} \right) = \begin{cases} \delta_{i,j} & e^{PH} = 0 \\ a_{i \to j}^{LL,k} & e^{PH} = 1, e^{LL} = 0 \\ \pi_j^{LL,k} & e^{PH} = 1, e^{LL} = 1 \end{cases}$$

$$\Pr\left(X_t^{PH} = j \mid X_{t-1}^{PH} = i, X_t^{LL} = k \right) = \begin{cases} 1 - a_{i \to i+1}^{PH,k} & j = i, i < n^{PH} \\ a_{i \to i+1}^{PH,k} & j = i+1, i < n^{PH} \\ \pi_j^{PH,k} & i = n^{PH} \\ 0 & otherwise \end{cases}$$

iv. End-of-sequence probabilities

$$\Pr\left(E_t^{HL} = 1 \mid X^{HL} = i, E_t^{LL} = e^{LL} \right) = \begin{cases} 0 & e^{LL} = 0 \\ p_{i \to end}^{HL} & e^{LL} = 1 \end{cases}$$

$$\Pr\left(E_t^{LL} = 1 \mid X_t^{LL} = i, X^{HL} = k, E_t^{PH} = e^{PH} \right) = \begin{cases} 0 & e^{PH} = 0 \\ p_{i \to end}^{LL,k} & e^{PH} = 1 \end{cases}$$

$$\Pr\left(E_t^{PH} = 1 \mid X_t^{PH} = i \right) = \delta_{i,n^{PH}}$$

c. Do EM learning:

$$bnet \leftarrow learnParamsEM\left(S \right)$$

4. Experiment results

We tested our algorithm using the video clips of a shopping center in Portugal that we found in [18]. We identify three high level activities: Entering the shop, leaving the shop, and passing the shop. For each video we ran a tracker developed by [2] to track people. We randomly set aside 8 tracks for training (3 entering, 3 leaving, 2 passing) and 6 tracks for testing. We labeled the training data with the appropriate high-level event. We used a four-dimensional feature space: x-position, y-position, x-velocity, y-velocity. The velocity estimate was done using fixed-lag differences of the positions and all features were smoothed with a Gaussian kernel.

We hypothesized that because people tend to walk at approximately the same speed, clustering of the space amounted to only considering few regions on common paths and a small number of directions. In the hallway, for example, the tracks would typically only go to the left or to the right, so we hypothesized this would form two clusters in the feature space for a given region in physical space. Figure 4b shows a visualization of how we expected the features to cluster.

For clustering, we scale model expressiveness with the availability of training data. We found experimentally that 16 states did a good job in recognizing the activities in our data set, so we ran the deterministic annealing algorithm with $T_{\min} = 0$ to find 16 clusters which we used as the low-level events. Figure 5 shows some features and the clusters projected down to various subspaces.

After running the clustering algorithm on the set of all features, we set up the rest of our DBN for learning. Experimentally, we chose to use 4 phases for our Coxian model (which is effectively 3 phases because the last phase denotes the end of the phase distribution). We also chose to insist that all low-level events went though all n^{PH} phases and so we fixed $\pi_j^{PH,k} = \delta_{j,1}$. Our training data consisted only of feature points from tracks of persons in video and high-level labels of their activity. To

do learning we used the expectation-maximization (EM) algorithm with randomly initialized free parameters and used the Boyen-Koller smoothing algorithm [15] to do the inference portion of EM.

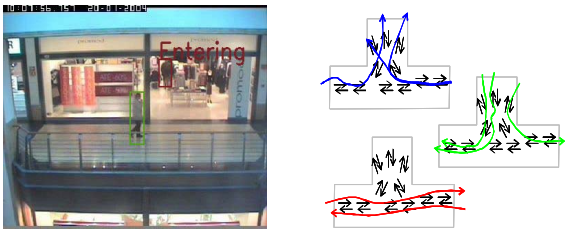

Figure 4: (a) Result for 'Entering' event (b). Pictorial representation of low-level states that result from clustering and the possible transitions that form high level events.

In the testing phase, we used the feature points from our tracker and did inference using the Boyen-Koller filtering algorithm [15] to compute the filtered marginal probabilities for each frame. That is, the probability reported at time *t* is the probability for each high level event given all of the measurements up until time *t* (as opposed to using future information). The results for 3 activities are shown in figure 6.

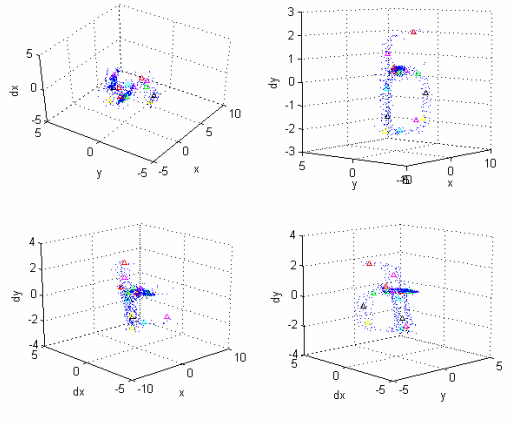

Figure 5: Clustering results.

Our results are promising. Consider the sequence for the "Entering" track (a). In this sequence the person walks along the pathway and approaches the door from frame 0 to around frame 150. As expected, during this time we are unsure as to whether the user is entering the store or merely passing. It appears we favor the possibility that the user is entering the store, which we believe is because the users velocity is in fact pointing towards going in the store, increasing the likelihood of being in a low-level state that would represent entering the shop. Alternatively, this could be simply because there is a greater prior probability that the user enters the store. Eventually the user has "turned the corner" at the entryway and almost immediately our algorithm is able to classify the person as entering with a high degree of certainty.

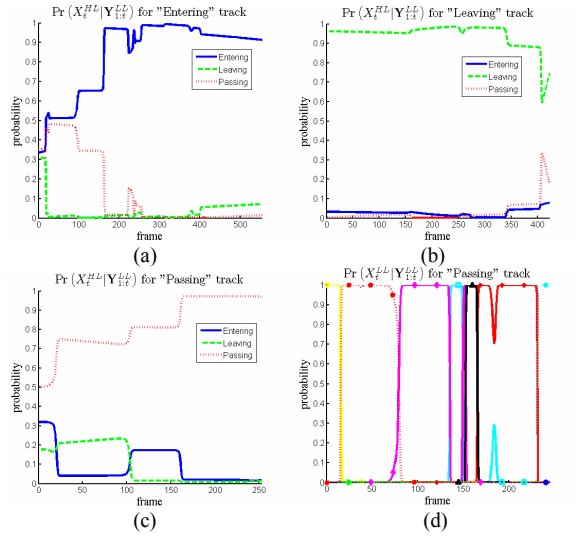

Figure 6: Results.

The "leaving" track is relatively easy to discern and we had little trouble with this high level event. We show some results in 6b. The results of a "passing" event are shown in figure 6c. This figure shares characteristics with the entering event in that there are clear points where one event is discerned from another. In this track the user passes the store from right to left. Initially there is uncertainty between entering and passing, which gets quickly resolved as the user passes the doorway. As the track goes past the doorway the uncertainty is than between passing and leaving, however this gets resolved without a problem. In figure 6d we show the filtered probabilities for the low-level events of the passing person. Here we can see that the user indeed passes from one cluster to another. Each jump in probability for the high level event has a corresponding low level event transition, suggesting that the data fits our model as expected.

Finally, in figure 7 we have two video sequences with tracks that we could not label with one of our high level events. These videos both had the following behavior: A person would exit the store by walking out and turning the corner. Next, the person would stop, pause for a short time, and turn around and go right back into the store. The trajectory of the person is unlike any trajectory seen in our training data. However, as we can see in figure 7, we are able to infer a meaningful label for both tracks where this occurs.

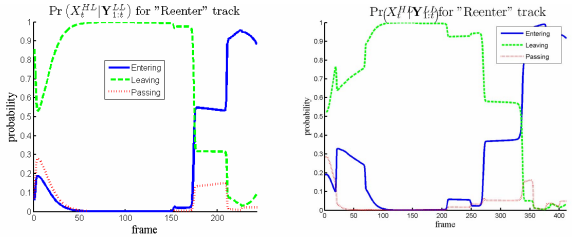

Figure 7: Results for leaving and re-entering.

The results of figure 7 also suggest that our duration model is working effectively. Notice that the track in Figure 7a is shorter (i.e., has fewer frames) than the track in Figure 7b. In fact, each low-level event also occurred for a shorter period of time. We believe that because the uncertainty associated with the duration of a low level event is effectively modeled, we are able to handle this case where events occur at different speeds.

Although the qualitative results of these graphs are representative of our overall results on many runs, in some cases where the classification was not correct or not clear. We believe that in most cases this was due to a lack of training data. We did many runs of our algorithm where in each run we would randomly partition our data set into training data and testing data. Sometimes our training data would be unrepresentative of the activities – for example, we may train the *passing* event with only with tracks of people passing from the left to the right. In this case when we tested on a track of a person passing from right to left we observed a chaotic result. We believe that for the high level events that we have defined we can solve this problem with more training data. However, the open question remains on what to do with abnormal sequences that do not fit any of the training data. Although figure 7 shows good results for mildly abnormal sequences, we ultimately wish to be able to classify very abnormal activities, such as loitering or fighting. We are working on ways to augment our model to be able to recognize such abnormal activities without having to rely on labeled data of abnormal activities.

5. Conclusions

We have presented a model for recognizing complex activities in video. We have given a lightweight representation of a hierarchical model in which we add domain-specific dependencies to and give a way to partition the space of observations into states in our model. Our method is robust to tracking errors and natural variations in high level activities because we model each state sequence probabilistically. We have tested our algorithm on video and our qualitative results suggest we can successfully classify normal activities and mildly abnormal activities in video sequences. We plan to use our technique to classify more complex activities and in future work we plan on recognizing more abnormal behaviors.

We would like to thank Dr. Isaac Cohen at Honeywell labs for many valuable discussions on this work.

6. References

[1] Y. Ma, B. Miller, P. Buddharaju and M. Bazakos: "Activity Awareness: from Predefined Events to New Pattern Discovery". ICVS 2006: 11.

[2] Y. Ma, Q. Yu and I. Cohen, "Multiple Hypothesis Target Tracking using Merge and Split of Graph's Nodes", ISVC 2006, LNCS 4291, pp. 783-792.

[3] S. Hongeng, F. Bremond and R. Nevatia, "Bayesian Framework for Video Surveillance Application", ICPR 2000, Vol I, pp. 164-170, September 2000

[4] V. D. Shet, D. Harwood and L. Davis, "Multivalued Default Logic for Identity Maintenance in Visual Surveillance" ECCV 2006.

[5] V. Vu, F. Brémond, M. Thonnat, "Automatic Video Interpretation: A Recognition Algorithm for Temporal Scenarios Based on Pre-compiled Scenario Models". ICVS 2003: 523-533.

[6] F. Cupillard, A. Avanzi, F. Bremond & M. Thonnat, "Video Understanding for Metro Surveillance", The IEEE ICNSC 2004 in special session on Intelligent Transportation Systems, Taiwen, 2004.

[7] M. Ryoo and J. Aggarwal, Recognition of Composite Human Activities through Context-Free Grammar Based Representation, pp 1709- 1718, CVPR 2006.

[8] T. V. Duong, H. H. Bui, D. Q. Phung, and S. Venkatesh. "Activity recognition and abnormality detection with the switching hidden semimarkov model". CVPR 2005.

[9] F. Porikli and T. Haga, "Event detection by Eigenvector Decomposition using object and feature frame", CVPR workshop, 2004.

[10] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210—2239, 1998.

[11] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgqan Kaufmann, 1988.

[12] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. J. Royal Stat. Soc., Ser. B, 50(2):157--224, 1988.

[13] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence 5*, 142–150.

[14] K. Murphy. "Dynamic Bayesian Networks: Representation, Inference, and Learning." Ph.D. thesis, UC Berkeley, 2002.

[15] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *UAI*, 1998.

[16] A. Doucet, N. de Freitas, K. Murphy, S. Russell, "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks".

[17] S. Fine, Y. Singer, and N. Tishby. The hierarchical Hidden Markov Model: Analysis and applications. *Machine Learning*, 32:41, 1998.

[18] CAVIAR dataset: http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1