Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2007, Article ID 74243, 13 pages
doi:10.1155/2007/74243

Research Article

Event Detection Using "Variable Module Graphs" for Home Care Applications

Amit Sethi, Mandar Rahurkar, and Thomas S. Huang

Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801-2918, USA

Received 14 June 2006; Accepted 16 January 2007

Recommended by Francesco G. B. De Natale

Technology has reached new heights making sound and video capture devices ubiquitous and affordable. We propose a paradigm to exploit this technology for home care applications especially for surveillance and complex event detection. Complex vision tasks such as event detection in a surveillance video can be divided into subtasks such as human detection, tracking, recognition, and trajectory analysis. The video can be thought of as being composed of various features. These features can be roughly arranged in a hierarchy from low-level features to high-level features. Low-level features include edges and blobs, and high-level features include objects and events. Loosely, the low-level feature extraction is based on signal/image processing techniques, while the high-level feature extraction is based on machine learning techniques. Traditionally, vision systems extract features in a feed-forward manner on the hierarchy, that is, certain modules extract low-level features and other modules make use of these low-level features to extract high-level features. Along with others in the research community, we have worked on this design approach. In this paper, we elaborate on recently introduced V/M graph. We present our work on using this paradigm for developing applications for home care applications. Primary objective is surveillance of location for subject tracking as well as detecting irregular or anomalous behavior. This is done automatically with minimal human involvement, where the system has been trained to raise an alarm when anomalous behavior is detected.

1. INTRODUCTION

Even with the US population rapidly aging, a smaller proportion of elderly and disabled people live in nursing homes today compared to 1990. Instead, far more depend on assisted living residences or receive care in their homes [1]. Majority of people who need long-term care still live in nursing homes, however the proportion of nursing home beds declined from 66.7 to 61.4 per 10 000 population. According to the author, these changing trends in the supply of long-term care can be expected to continue because the demand for home- and community-based services is growing. These healthcare services besides being expensive may often be emotionally traumatic for the subject. Large number of these people who live here can perform basic day-to-day tasks, however need to be under constant supervision in case assistance is required. In this paper, we show how current technology can enable us to monitor these subjects in an environment which is most amicable—their own home. Today's digital technology has made sound and video capture devices affordable for a common user. Also there has been tremendous progress in research and development in the fields of image and video compression, editing, and analysis software leading to its effective usability and commercialization.

However, success in developing general methods of analyzing video in a wide range of scenarios remains elusive. The main reason for this is the number of parameters affecting various pixels in a video or across videos. Moreover, the sheer amount of raw data in video streams is voluminous. Yet, the problem of image or video understanding especially for complex event detection task at hand is often ill-posed, making it difficult to solve the problems based on the given data alone. It is, therefore, important to understand the nature of the generation of the visual data itself and to understand the features of visual data that human users would be interested in, and how those features might be extracted. Relation of features amongst each other and how the modules extracting them might interact with each other is vital in designing vision systems.

We elaborate on a recently proposed framework [2] based on factor graphs. It relaxes some of the constraints of the traditional factor graphs [3] and replaces its function nodes by modified versions of some of the modules that have been developed for specific vision tasks. These modules can be easily formulated by slightly modifying modules developed for specific tasks in other vision systems, if we can match the input and output variables to variables in our graphical structure. It also draws inspiration from product of experts [4], and free energy view [5] of the EM algorithm [6]. We present some preliminary results for tracking and event detection applications and discuss the path for future development.

Outline of this paper is as follows. Section 2 introduces factor graphs, thereby generalizing to variable module or V/M graphs in Section 3. V/M graphs are explored extensively, thus establishing the theoretical background in Section 4. We demonstrate use of V/M graphs for home care applications, especially complex event detection and subject tracking in Section 5.

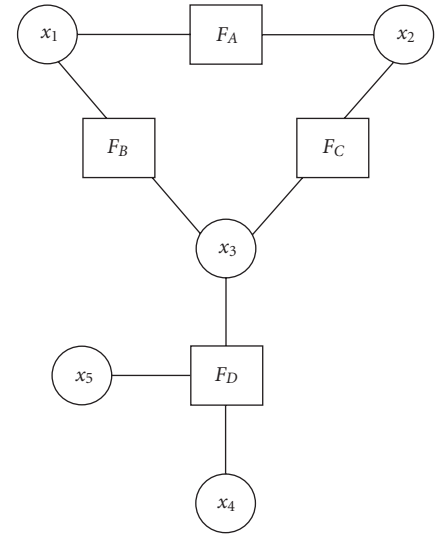


Figure 1: Example factor graph.

2. ALGORITHMS

2.1. Factor graphs

In order to understand V/M graphs, we briefly explain factor graphs. A factor graph is a bipartite graph that expresses a structure of factorization of a function into product of several local functions, thus making it efficient to represent the dependencies between random variables. Factor graph has a variable node for each random variable x_i, factor node for each local function f_j, and a connecting edge between variable node x_i and factor node f_j only if x_i is an argument of f_j. A factor (function) of a product term can selectively look at a subset of dimensions while leaving the other dimensions that are not in the subset for others factors to constrain. In other words, only a subset of variables may be part of the constraint space of a given expert. This leads to the graphs structure of a factor graph, where the edges between a factor function node and variable nodes exist only if the variable appears as one of the arguments of the factor function,

$$p(x_1,x_2,x_3,x_4,x_5) \propto \begin{pmatrix} & p_A(\ x_1,\ x_2,\ x_3,\ x_4,\ x_5\ ) \\ \times & p_B(\ x_1,\ x_2,\ x_3,\ x_4,\ x_5\ ) \\ \times & p_C(\ x_1,\ x_2,\ x_3,\ x_4,\ x_5\ ) \\ \times & p_D(\ x_1,\ x_2,\ x_3,\ x_4,\ x_5\ ) \end{pmatrix} \propto \begin{pmatrix} & f_A(\ x_1,\ x_2 & & & &) \\ \times & f_B(\ & x_2,\ x_3 & & &) \\ \times & f_C(\ x_1, & & x_3 & &) \\ \times & f_D(\ & & x_3,\ x_4,\ x_5\ ) \end{pmatrix}. \tag{1}$$

In (1), $f_A(x_1,x_2)$, $f_B(x_2,x_3)$, $f_C(x_1,x_3)$, and $f_D(x_3,x_4,x_5)$ are the factor functions of the factor graph. The factor graph in (1) can be expressed graphically as shown in Figure 1.

Inference in factor graphs can be made using a local message passing algorithm called the sum-product algorithm [3]. The algorithm reduces the exponential complexity of calculating the probability distribution over all the variables into more manageable local calculations at the variable and function nodes. The local calculations depend only on the incoming messages from the nodes adjacent to the node at hand (and the local function, in case of function nodes). The messages are actually distributions over the variables involved. For a graph without cycles, the algorithm converges when messages pass from one end of the graph to the other and back. For many applications, even when the graph has loops, the messages converge in a few iterations of message passing. Turbo codes in signal processing make use of this property of convergence of loopy propagation [7]. The message passing clearly is a principled form of feedback or information exchange between modules. We will make use of a variant of message passing for our new framework because exact message passing is not feasible for complex vision systems.

3. V/M GRAPH

We develop a hybrid framework to design modular vision systems. In this new framework, which we call *variable/module graphs* or *V/M graphs* [2, 8], we aim to borrow the strengths of both modular and generative designs. From the generative models in general and probabilistic graphical models in particular, we want to keep the principled way to explain all the information available and the relations between different variables using a graphical structure. From the modular design, we want to borrow ideas for local and fast processing of information available to a given module as well as online adaptation of model parameters.

3.1. Replacing functions in factor graphs with modules

Modules in modular design constrain the joint-probability space of observed and hidden variables just as the factor functions in factor graphs. However, there are crucial differences. Without loss of generality, we will continue our discussion on graphical models based on factor graphs, since

many of the other graphical models can be converted to factor graphs.

Modules in modular design take (probability distributions of) various variables as inputs, and produce (probability distributions of) variables as outputs. Producing an output can be thought of as passing a message from the module to the output variable. This is comparable to part of the message passing algorithm in factor graphs, that is, passing a message from the function node to a variable node. This calculation is done by multiplying messages from all the other variable nodes (except the one that we are sending the message to) to the factor function at the function node, and marginalizing the product over all the other variables (except the one that we are sending the message to). Processing of a module can be thought of as an approximation to this calculation.

However, the notion of a variable node does not exist in modular design. Let us, for a moment, imagine that modules are not connected to each other directly. Instead, let us imagine that every connection that connects output of a module to the input of another module is replaced by a node connected to the output of the first module and input of the second module. This node represents the output variable of the first module, which is the same as the input node of the second module. Let us call this the *variable node*.

In other words, a cascade of modules in a modular system is nothing but a cascade of approximations to function nodes (separated by variable nodes, of course). If we generalize this notion of interconnection of module or *module nodes* via variable nodes, we get a graph structure. We refer to his bipartite graph as *variable/module graph*. Thus, if we replace the function nodes in a factor graph by modules, we get a variable/module graph—a bipartite graph in which the variables represent one set of nodes (called variable nodes), and modules represent the other set of nodes (called module nodes).

4. SYSTEM MODELING USING V/M GRAPHS

A factor graph is a graphical representation of the factorization that a product form represents. Since the variable/module graph can be thought of as a generalization of the factor graph, what does this mean for the application of product form to the V/M graph? In essence, we are still modeling the overall constraints on the joint-probability distribution using a product form. However, the rules of message passing have been relaxed. This makes the process an approximation to the exact product form [8]. To see how we are still modeling the joint-distribution over the variables using a product form, let us start by analyzing the role of modules. A module takes the value of the input variable(s) x_i and produces a probability distribution over the output variable(s) x_j. This is nothing but the conditional distribution over the output variables given the input variable, or $p(x_j \mid x_i)$. Thus, each module is nothing but an instantiation of such conditional density functions.

In a Bayesian network, similar conditional probability distributions are defined, with an arrow representing the direction of causality. This makes it a simple case to define the module as a/an (set of) arrow(s) going from the input to the output, converting the whole V/M graph into a Bayesian network, which is another graphical representation of the product form. Also, since the Bayesian network can always be converted into a factor graph [9], we can convert a V/M graph into a factor graph. However, processing modules are many times arranged in a bottom-up fashion, whereas the flow of causality in a Bayesian network is top-down. This is not a problem, since we can use Bayes rule to reverse the flow of causality. Once we have established a module as an equivalent of a conditional density, manipulation of the structure is easy, and it always remains in the purview of product form modeling of the joint distribution. However, the similarity between V/M graphs and probabilistic graphical models ends here on a theoretical level. As we will in Section 4.1, the inference mechanisms that are applied in practice to graphical models are not applied in the exact same manner to V/M graphs. One of the reasons for this is that modules do not produce a functional form of the conditional density functions. They only produce a black box that we can sample output (distribution) from for given sample points of input, and not the other way around. Thus, in practice, application of Bayes rule to change the direction of causality is not as easy as it is in theory. We use comodules, at times, for flow of messages in the other direction to a given module.

4.1. Inference

In a factor graph, calculating the messages from variable nodes to function nodes, or the belief at each variable node is usually not difficult. When the incoming messages are in a nonparametric form, any kind of resampling algorithm or nonparametric belief propagation [10] can be used. What is more difficult is the integration or summation associated with the marginalization needed to calculate the message from a function node to a variable node. Another difficulty that we face here is the complexity with which we can design the local function at a function node. Since we also need to calculate the messages using products and marginalization (or sum), we need to devise functions that model the subconstraint as well as lend themselves to easy and efficient marginalization (or approximation thereof). If one is to break the function down into more sub-functions, there is a tradeoff involved between network complexity and function complexity for a manageable system. This is where we can make use of the modules developed for other systems. The output of a module can be viewed as a marginalization operation used to calculate message sent to the output variable. Now, the question arises what we can say about the message sent to the input variable. If we really cannot modify the module to send a message to what was the input variable in the original module, we can view it as passing a uniform message (distribution) to the input variable. To save computation, this message can be totally discounted during calculations that require combination of this message with other messages. However, in this framework, we encourage modifying existing modules to pass information backwards

as well. A way to do this is to associate a comodule with the module that does the reverse of the processing that the module does. For example, if a module takes in a background mask and outputs probability map of the position of a human in the frame, the comodule will provide some probability map of pixels belonging to background or foreground (human) given the position of human to this comodule.

In case the module node is a deterministic function, the probability function of the output variable will be treated as a delta function. Although there are definite merits of a stricter definition of a V/M graph for a stringent mathematical analysis, it might result in loss of applicability and flexibility to workable systems at this point. By introducing modified modules as approximation to functions and their message calculation procedures, we get computationally cheap approximations to complex marginalization operations over functions that will be difficult to perform from first principles or statistical sampling, the approach used with generative models until now. Whether this kind of message passing will converge or not even for graphs without cycles remains to be seen in theory, however, we have found the results to be convincing for the applications that we implemented it for as shown in Section 5.

4.2. Learning

There are a few issues that we would like to address while designing learning algorithms for complex vision systems. The first issue is that when the data and system complexity are prohibitive for batch learning, we would really like to have designs that lend themselves to online learning. The second major issue is the need to have a learning scheme that can be divided into steps that can be performed locally at different modules or function nodes. This makes sense, since the parameters of a module are usually local to the module. Especially in an online learning scheme, the parameters should depend only on the local module and the local messages incident on the function node.

We will derive learning methods for V/M graphs based on those for probabilistic graphical models. Although methods for structure learning in graphical models have been explored [11, 12], we will limit ourselves for the time being to parameter learning. In line with our stated goals in the paragraph above, we will consider online and local parameter learning algorithms for probabilistic graphical models [13, 14] while deriving learning algorithms for V/M graphs.

Essentially, parameter adjustment is done as a gradient ascent over the log likelihood of the given data under the model. While formulating the gradient ascent over the cost function, due to the factorization of the joint-probability distribution, derivative of the cost function decomposes into a sum of terms, where each term pertains to local functions. A similar idea can be extended to our modified factor graphs or V/M graphs.

Now, we will derive a gradient-ascent-based algorithm for parameter adjustment for V/M graphs. Our goal is to find the model parameters that maximize the data likelihood $p(D)$, which is a standard goal used in the literature [6, 13], since (observed) data is what we have and seek to explain, while the rest of the (hidden) variables just aid in modeling the data. Each module will be represented by a conditional density function $p_{\omega_i}(x_i \mid N_i)$. Here, x_i represents the output variable of the ith module, N_i represents the input set of variables to the ith module, and ω_i represents the parameters associated with the module. We will make the assumption that data points are independently identically distributed (*i.i.d.*), which means that for data points d_j (where j ranges from 1 to m, the number of data points) and the data likelihood $p(D)$, (2) holds,

$$p(D) = \prod_{j=1}^{m} p(d_j). \tag{2}$$

In principle, we can choose any monotonically increasing function of the likelihood, and we chose the $\ln(\cdot)$ function to convert the product into a sum. This means that for the log likelihood, (3) holds,

$$\ln p(D) = \sum_{j=1}^{m} \ln p(d_j). \tag{3}$$

Therefore, when we maximize the log likelihood with respect to the parameters ω_i's, we can concentrate on maximizing the log likelihood of each data point by gradient ascent, and adding these gradients together to get the complete gradient of the log likelihood over the entire data. Thus, at each step we need to deal with only one data point, and accumulate the result as we get more data points. This is significant in developing online algorithms that deal with limited (one) data point(s) at a time. In case where we tune the parameters slowly, this is in essence like a running average with a forgetting factor.

Now, taking the partial derivative of the log likelihood of one data point d_j with respect to a parameter ω_i, we get

$$\begin{aligned}
&\frac{\partial \ln p(d_j)}{\partial \omega_i} \\
&\quad = \frac{(\partial/\partial\omega_i)\, p(d_j)}{p(d_j)} \\
&\quad = \frac{(\partial/\partial\omega_i)\left(\int_{x_i,N_i} p(d_j \mid x_i, N_i)\, p(x_i, N_i)\, dx_i\, dN_i\right)}{p(d_j)} \\
&\quad = \frac{(\partial/\partial\omega_i)\left(\int_{x_i,N_i} p(d_j \mid x_i, N_i)\, p(x_i \mid N_i)\, p(N_i)\, dx_i\, dN_i\right)}{p(d_j)} \\
&\quad = \frac{\int_{x_i,N_i} (\partial/\partial\omega_i)\left(p(d_j \mid x_i, N_i)\, p(x_i \mid N_i)\, p(N_i)\right) dx_i\, dN_i}{p(d_j)} \\
&\quad = \frac{\int_{x_i,N_i} p(N_i)\,(\partial/\partial\omega_i)\left(p(d_j \mid x_i, N_i)\, p(x_i \mid N_i)\right) dx_i\, dN_i}{p(d_j)}.
\end{aligned} \tag{4}$$

Since we will get $p(d_j \mid x_i, N_i)$ as a result of message passing, and we will get $p(x_i \mid N_i)$ as the output of the processing module, all these computations can be done locally at the module i itself. The probability densities $p(d_j)$ and $p(N_i)$ are nonnegative functions that only scale the gradient computation, and not the direction of the gradient. With V/M graphs,

when we are not even expecting to calculate the gradient, we will only try to do a generalized gradient ascent by going in the direction of positive gradient. It suffices that as an approximate greedy algorithm, we move in the general direction of increasing $p(x_i \mid N_i)$ and hope that $p(d_j \mid x_i, N_i)$, which is a marginalization of the product of $p(x_k \mid N_k)$ over many k's, will follow an increasing pattern as we spread the procedure over many k's (modules). The greedy algorithm should be slow enough in gradient ascent that it can capture the trend over many j's (data points) when run online. This sketches the general insight into the learning algorithm. The sketch is in line with a similar derivation for Bayesian network parameter estimation in [13], where the scenario is much better defined than it is for V/M graphs. In Section 4.4, we provide another viewpoint to justify the same steps.

4.3. Free-energy view of EM algorithm and V/M graphs

For generative models, the EM algorithm [6] and its online, variational, and other approximations have been used as the learning algorithm of choice. Online methods work by maintaining sufficient statistics at every step for the q-function that approximates the probability distribution p of hidden and observed variables. We use a free-energy view of the EM algorithm [5] to justify a way of designing learning algorithms for our new framework. In [5], the online or incremental version of EM algorithm was justified using a distributed E-step. We extend this view to justify local learning at different module nodes. Being equivalent to a variational approximation to the factor graph means that some of the concepts applicable to generative models, such as variational and online EM algorithms, can be applicable to the V/M graphs. We use this insight to compare inference and learning in V/M graphs to the free-energy view of EM algorithm [5].

Let us assume that X represents the sequence of observed variables x_i, and Y represents the sequence of hidden variables y_i. So, we are modeling the generative process $p(x_i \mid y_i, \theta)$, with some prior on y_i; $p(y_i)$, given system parameters θ (which is the same for all pairs (x_i, y_i)). Due to the Markovian assumption of x_i being conditionally independent of x_j given Y, when $i \neq j$, we get

$$p(X \mid Y, \theta) = \prod_i p(x_i \mid y_i, \theta). \tag{5}$$

We would like to maximize the log likelihood of the observed data X. EM algorithm does this by alternating between an E-step as shown in (6) and an M-Step shown in (7) in each iteration with iteration number t,

$$\text{compute distribution: } q^t(y) = p(y \mid x, \theta^{(t-1)}), \tag{6}$$

$$\text{compute arg max: } \theta^{(t)} = \arg\max_{\theta} E_{q^t}[\log P(x, y \mid \theta)]. \tag{7}$$

Going by the free-energy view of the EM algorithm [5], the E- and M-steps can be viewed as alternating between maximizing the free energy with respect to the q-function and the parameters θ. This is related to the minimization of free energy in statistical physics. The formulation of free energy F is given in

$$F(q, \theta) = E_q[\log(x, y \mid \theta)] + H(q) = -D(q \| p_\theta) + L(\theta). \tag{8}$$

In (8), $D(q\|p)$ represents the KL-divergence between q and p given by (9), and $L(\theta)$ represents the data likelihood for the parameter θ. In other words, the EM algorithm alternates between minimizing the KL-divergence between q and p, and maximizing the likelihood of the data given the parameter θ,

$$D(q \| p) = \int_y q(y) \log \frac{q(y)}{p(y)} dy. \tag{9}$$

The equivalence of the regular form of EM and the free-energy form of EM has already been established in [5]. Further, since y_i's are independent of each other, the $q(y)$ and $p(y)$ terms can be split into products of different $q(y_i)$'s and $p(y_i)$'s, respectively. This is used to justify the incremental version of EM algorithm that incrementally runs partial or generalized M-steps on each data point. This can also be done using sufficient statistics of the data collected until that data point, if it is possible to define sufficient statistics for a sequence of data points.

Coming back to the message passing algorithm, for each data point, when message passing converges, the beliefs at each variable node give a distribution over all the hidden variables. If we look at the q-function, it is nothing but an approximation of the actual distribution over the variable p, and we are trying to minimize the KL-divergence between the two. Now, we can get the same q-function from the converged messages and beliefs in the graphical model. Hence, one can view message passing as a localized and online version of the E-step.

4.4. Online and local M-step

Now, let us have a look at the M-step. M-step involves maximizing the likelihood with respect to the parameter θ. When performed online for a particular data point, it can be thought of as a stochastic gradient ascent version of (7). Making use of the sufficient statistics will definitely improve the approximation of the M-step since it will use the entire data presented until that point, instead of a single data point. Now, if we take the factorization property of the joint-probability function into account, we can also see that the M-step can be distributed locally for each component of the parameter θ associated with each module or function node. This justifies the localized parameter updates based on gradient ascent shown in [13, 14]. This is another critical insight that will help us to use the online learning algorithms devised for various modules to be used as local M-steps in our systems. Due to the integration involved with the marginalization over the hidden variables while calculating the likelihood, this will be an approximation of the exact M-step. Determining the conditions where this approximation should work will be part of our future work.

One issue that still remains is the partition function. With all the local M-steps maximizing one term of the likelihood in a distributed fashion, it is likely that the local terms increase infinitely, while the actual likelihood does not. This problem arises when appropriate care is not taken to normalize the likelihood by dividing it with a partition function. While dealing with sampling-based numerical integration methods such as MCMC [15], it becomes difficult to calculate the partition function. This is because methods such as importance sampling and Gibbs sampling used in MCMC deal with surrogate q-function, which is usually a constant multiple of the target q-function. The multiplication factor can be assessed by integrating over the entire space, which is difficult. There are two ways of getting around this problem. One way was suggested in [4] as maximizing the contrastive divergence instead of the actual divergence. The other way is to put some kind of local normalization in place while calculating messages sent out by various modules. As long as the multiplication factor of the q-function does not increase beyond a fixed number, we can guarantee that maximizing the local approximation of the components of the likelihood function will actually improve system performance.

In the M-step of the EM algorithm, we minimize $Q(\theta, \theta^{(i-1)})$ with respect to θ. In the proof given by (10), we show how this minimization can be distributed over different components of the parameter variable θ,

$$\begin{aligned} Q(\theta, \theta^{(i-1)}) &= E[\log p(X, Y \mid \theta) \mid X, \theta^{(i-1)}] \\ &= \int_{h \in H} \log p(X, Y \mid \theta) f(Y \mid X, \theta^{(i-1)}) dh \\ &= \int_{h \in H} \left(\sum_{i=1}^{m} \log p(x_i, y_i \mid \theta_i) \right) f(Y \mid X, \theta^{(i-1)}) dh \\ &= \sum_{i=1}^{m} \int_{h \in H} \log p(x_i, y_i \mid \theta_i) f(Y \mid X, \theta^{(i-1)}) dh, \end{aligned} \tag{10}$$

$$\text{M-step: } \theta^{(i)} \longleftarrow \arg\max_{\theta} Q(\theta, \theta^{(i-1)}). \tag{11}$$

4.5. Probability distribution function softening

Until now, PDF softening was only intuitively justified [4]. In this section, we revisit the intuition, and justify the concept mathematically in

$$\begin{aligned} & D(q \parallel p) \\ &= \int_{x \in X} q(x) \log \frac{q(x)}{p(x)} dx \\ &= \int_{x \in X} q(x) \log q(x) dx - \int_{y \in X} q(y) \log p(y) dy \\ &= \int_{x \in X} q(x) \log \frac{\prod_i q_i(x)}{\int_{w \in X} \prod_j q_j(w) dw} dx - \int_{y \in X} q(y) \log p(y) dy \\ &= \int_{x \in X} q(x) \left(\sum_i (\log q_i(x)) - \log \left(\int_{w \in X} \prod_j q_j(w) dw \right) \right) dx \\ &\quad - \int_{y \in X} q(y) \log p(y) dy \\ &= \int_{x \in X} \sum_i (q(x) \log q_i(x)) - q(x) \log \left(\int_{w \in X} \prod_j q_j(w) dw \right) dx \\ &\quad - \int_{y \in X} q(y) \log p(y) dy \\ &= \sum_i \left(\int_{x \in X} q(x) \log q_i(x) dx \right) \\ &\quad - \int_{z \in X} q(z) \log \left(\int_{w \in X} \prod_j q_j(w) dw \right) dz - \int_{y \in X} q(y) \log p(y) dy \\ &= \sum_i \left(\int_{x \in X} q(x) \log q_i(x) dx \right) \\ &\quad - \log \left(\int_{w \in X} \prod_j q_j(w) dw \right) \int_{z \in X} q(z) dz - \int_{y \in X} q(y) \log p(y) dy \\ &= \sum_i \left(\int_{x \in X} q(x) \log q_i(x) dx \right) - \log \left(\int_{w \in X} \prod_j q_j(w) dw \right) \\ &\quad - \int_{y \in X} q(y) \log p(y) dy. \end{aligned} \tag{12}$$

As shown in (12), if we want to decrease the KL-divergence between the surrogate distribution q and the actual distribution p, we need to minimize the sum of three terms. The first term on the last line of the equation is minimized if there is an increase in the high-probability region as defined by q, which is actually a low-probability region for an individual component q_i. This means that this term prefers diversity among different q_i's, since q is proportional to the product of q_i's. Thus, the low-probability regions of q need not be low-probability regions of a given q_i. On the other hand, the third term is minimized if there is an overlap between the high-probability region as defined by q and the high-probability region defined by p and between the low-probability region as defined by q and the low-probability region defined by p. In other words, surrogate distribution q should closely model the actual distribution p.

Hence, overall, the model seeks a good fit in the product, while seeking diversity in individual terms of the product. It also seeks not-so-high-probability regions of individual q_i's to overlap with high-probability regions of q. When p has a peaky (low-entropy) structure, these goals may seem conflicting. However, this problem can be alleviated if the individual experts cater to different dimensions or aspects of the probability space, while each individual distribution has high enough entropy. This justifies softening the PDFs. This can be done by adding a high-entropy distribution such as a uniform distribution (which has provably the highest entropy), by raising the distribution to a fractional power, or by raising the variance of the peaks. Intuitively, this means that we want to strike a balance between useful opinion expressed by

an expert and being overcommitted to any particular solution (high-probability region).

4.6. Prescription

With the discussion on the theoretical justification of the design of V/M graphs complete, in this section we want to summarize how to design a V/M graph for a given application. In Section 5, we will present experimental results of successful design of vision systems for complex tasks using V/M graphs.

To design a V/M graph for an application, we will follow the following guidelines.

(1) Identify the variables needed to represent the solution.
(2) Identify the intermediate hidden variables.
(3) Suitably breakdown the data into a set of observed variables.
(4) Identify the processing modules that can relate and constrain different variables.
(5) Ensure that there is enough diversity in the processing modules.
(6) Lay down the graphical structure of the V/M graph similar to how one would do that for a factor graph, using modules instead of function nodes.
(7) Redesign each module so that it can tune online to increase local joint-probability function in an online fashion.
(8) Ensure that the modules have enough variance or leniency to be able to recover from mistakes based on the redundancy provided by the presence of other modules in a graphical structure.
(9) If a module has no feedback for a variable node, this can be considered to be a feedback equivalent of a uniform distribution. Such a feedback can be dropped from calculating local messages to save computation.

Once the system has been designed, the processing will follow a simple message passing algorithm while each module will learn in a local and online manner. If the results are not desirable, one would want to replace some of the modules with better estimators of the given task, or make the graph more robust by adding more (and diverse) modules, while considering making modules more lenient.

5. EXPERIMENTS

In this section, we report design and experimental results of several applications related to home care applications under the broad problem of automated surveillance. We focus on security and monitoring of home care subjects, and hence the targeted applications are automatic event detection and abnormal event detection. Thus, an alarm would be raised in case of abnormal activity, for example, like subject falling down. Event is a high-level semantic concept and is not very easy to define in terms of low-level raw data. This gap between the available data and the useful high-level concepts is known as the *semantic gap*. It can be safely said that the vision systems, in general, aim to bridge the semantic gap in visual data processing. Variables representing high-level concepts such as events can be conveniently defined over lower-level variables such as position of people in a frame; provided that the defining lower-level variables are reliably available. For example, if we were to decide whether a person came out or went in through a door, we can easily decide this if the sequence of the position of the person (and the position of the door) in various frames in the scene was available to us. This is the rationale behind modular design, where in this case, one would devise a system for person tracking, and the output of the tracking module would be used by an event detection module to decide whether the event has taken place or not.

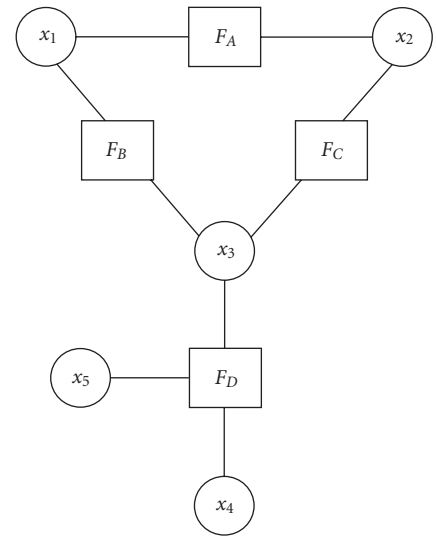


Figure 2: V/M graph for single-target tracking application.

The scenario that we considered for our experiments is related to the broad problem of automated surveillance. Without loss of generality, we assume a fixed camera in our experiments. In the following experiments, we concentrate on several applications of V/M graphs in the surveillance setting. We will proceed from simpler tasks to increasingly complex tasks. While doing so, many times we will incrementally build upon previously accomplished subtasks. This will also showcase one of the advantages of V/M graphs; namely, easy extendability.

5.1. Application: person tracking

We start with the most basic experiment, where we build an application for tracking a single target (person) using a fixed indoor camera. In this application, we identify five variables that affect inference in a frame. The intensity map (pixel values) of the frame (or, the observed variable(s)), the background mask, the position of the person in the current frame, the position of the person in previous frame, the velocity of the person in previous frame. These variables are represented as x_1, x_2, x_3, x_4, and x_5, respectively in Figure 2. All nodes except x_1 are hidden nodes. The variables exchange information through modules F_A, F_B, F_C, and F_D. Module

F_A represents the background subtraction module that maintains an eigenbackground model [16] as system parameters, using a modified-version online learning algorithm for performing principal component analysis (PCA) as described in [17]. While it passes information from x_1 to x_2, it does not pass it the other way round, as image intensities are evidence, hence fixed. Module F_C serves as the interface between the background mask and the position of the person. In effect, we run an elliptical Gaussian filter, roughly of the size of a person/target, over the background map and normalize its output as a map of the probability of a person's position. Module F_B serves as the interface between the image intensities and the position of the person in the current frame x_3. Since it is computationally expensive to perform operations on every pixel location, we sample only a small set of positions to confirm if the image intensities around that position resemble the appearance of the person being tracked. The module maintains an online learning version of eigenappearance of the person as system parameters based on a modification of a previous work [18]. It also does not pass any message to x_1. The position of the person in the current frame is dependent on the position of the person in the previous frame x_4 and the velocity of the object in the previous frame x_5. Assuming a first-order motion model, which is encoded in F_D as a Kalman filter, we connect x_3 to x_4 and x_5. x_4 and x_5 are assumed fixed for the current frame, therefore F_D only passes the message forward to x_3 and does not pass any message to x_4 or x_5.

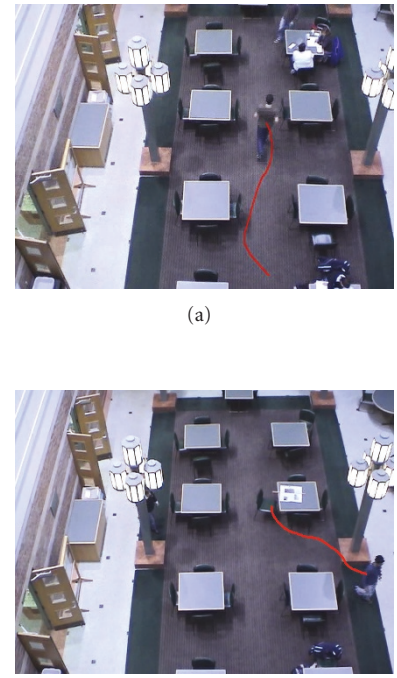
5.1.1. Message passing and learning schedule

The message passing and learning schedule used was as follows.

(1) Initialize a background model.
(2) If a large contiguous foreground area is detected, initialize a person detection module F_C, and tracking-related modules F_B and F_D.
(3) Initialize the position of the person in the previous frame as the most likely position according to the background map.
(4) Initialize the velocity of the person in the previous frame to be zero.

For every frame,

(1) propagate a message from x_1 to F_A as the image;
(2) propagate a message from x_1 to F_B as the image;
(3) propagate messages from x_4 and x_5 to F_D;
(4) propagate a message from F_D to x_3 in the form of samples of likely position;
(5) propagate a message from F_A to x_2 in form of a background probability map after an eigenbackground subtraction;
(6) propagate a message from x_2 to F_C in the form of a background probability map;
(7) propagate a message from F_C to x_3 in the form of a probability map of likely positions of the object after filtering of x_2 by an elliptical Gaussian filter;
(8) propagate a message from x_3 to F_B in the form of samples of likely position;
(9) propagate a message from F_B to x_3 in the form of probabilities at samples of likely position as defined by the eigenappearance of the person maintained at F_B;
(10) combine the incoming messages from F_B, F_C, and F_D at x_3 as the product of the probabilities at the samples generated by F_D;
(11) infer the highest probability sample as the new object position measurement. Calculate current velocity;
(12) update online eigenmodels at F_A and F_B;
(13) update motion model at F_D.



(a)

(b)

Figure 3: Tracking sequences after using color information.

5.1.2. Results

We ran our person tracker in both single-person and multi-person scenarios using grey-scale indoor sequences 320×240 in dimensions using a fixed camera. People appeared to be as small as 7×30 pixels. It should be noted that no elaborate initialization and no prior training were done. The tracker was required to run and learn on the job, fresh out of the box. The only prior information used was the approximate size of the target, which was used to initialize the elliptical filter. Some of the successful results on difficult sequences are shown in Figure 3. The trajectory estimation depends on the tracking estimate, however we did not notice serious deficiencies in this approach in our experimentation.

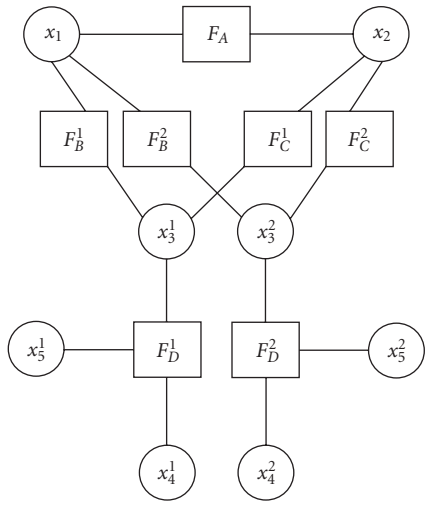


FIGURE 4: V/M graph for multiple-target tracking application (here, two targets).

(a)

(b)

FIGURE 5: Different successful tracking sequences involving multiple targets and using color information.

The tracker could easily track people successfully after complete but brief occlusion, owing to the integration of a background subtraction, eigenappearance, and motion models. The system successfully picks up and tracks a new person automatically when he/she enters the scene, and gracefully purges the tracker when the person is no longer visible. As long as a person is distinct from the background for some time during a sequence of frames, the online adaptive eigenappearance model successfully tracks the person even when they are subsequently camouflaged into the background. Note that any of the tracking components in isolation would fail in difficult scenarios such as a complete occlusion, widely varying appearance of people, and background camouflage.

To alleviate the problem of losing track because of occlusion, coupled with matching of background objects in appearance, we changed our model to include more information. Specifically, we used color frames, instead of greyscale frames. The V/M graphs remain the same, as shown in Figure 2.

5.2. Application: multiperson tracking

To adapt the single-person tracker developed in Section 5.1 for multiple targets, we need to modify the V/M graph depicted in Figure 2. In particular, we will need at least one position variable for each target being tracked. We will also need one variable representing the position in the previous frame and one representing the velocity in the previous frame for each object. On the module side, we will need one module each for each object representing the appearance matching, elliptical filtering on the background map, and Kalman filter. The resulting V/M graph is shown in Figure 4. The message passing and learning schedule were pretty much the same as given in Section 5.1.1, except that the steps specific to the target were performed for each target being tracked.

5.2.1. Results

We ran our person tracker to track multiple-person greyscale indoor sequences 320×240 in dimensions using a fixed camera. People appeared to be as small as 7×30 pixels. It should be noted that no elaborate initialization and no prior training were done. The tracker was required to run and learn on the job, fresh out of the box. The results are shown in Figure 5.

6. TRAJECTORY PREDICTION FOR UNUSUAL EVENT DETECTION

A tracking system can be an essential part of a trajectory modeling system. Many interesting events in a surveillance scenario can be recognized based on trajectories. People walking into restricted areas, violations at access controlled doors, moving against the general flow of traffic are examples of few interesting events that can be extracted based on trajectory analysis. With this framework, it is easy to incrementally build a trajectory modeling system on top of a tracking system with interactive feedback from the trajectory models to improve tracking results.

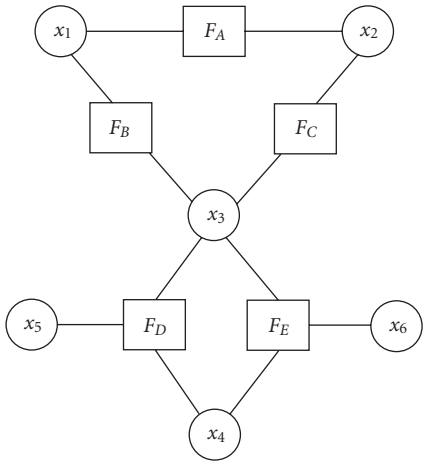


Figure 6: V/M graph for trajectory modeling system.

6.1. Trajectory modeling module

We add a trajectory modeling module F_E connected to x_3 and x_4 which represent the positions of the object being tracked in the current frame and the previous frame, respectively. The factor graph of the extended system is shown in Figure 6.

The trajectory modeling module stores the trajectories of the people, and predicts the next position of the object based on previously stored trajectories. The message passed from F_E to x_3 is given in

$$p_{\text{traj}} \propto \alpha + \sum_i w_i x_i^{\text{pred}}. \tag{13}$$

In (13), p_{traj} is the message passed from F_E to x_3, α is a constant added as a uniform distribution, i is an index that runs over the stored trajectories, w_i is the weight calculated based on how close is the trajectory to the position and direction of the current motion, and x_i^{pred} is the next point to the current closest point on the trajectory to the object position in the previous frame. The predicted trajectory is represented by variable x_6.

6.2. Results

This is a very simple trajectory modeling module, and the values of various constants were set empirically, although no elaborate tweaking was necessary. As shown in Figure 7, we can predict the most probable trajectory in many cases where similar trajectories have been seen before.

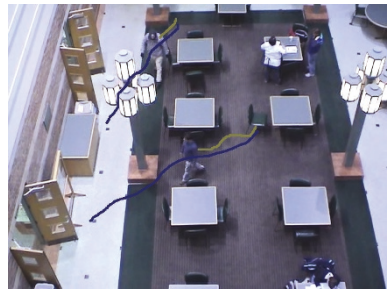
Other approaches to trajectory modeling such as vector quantization [19] can be used to replace the trajectory modeling module in this framework.

7. APPLICATION: EVENT DETECTION BASED ON SINGLE TARGET

The ultimate goal for automated video surveillance is to be able to do automatic event detection in video. With trajectory analysis, we move closer to this goal, since there are many events of interest that can be detected using trajectories. In this section, we present an application to detect the event whether a person went in or came out of a secure door. To design this application, all we have to do is to add an event detection module that is connected to the trajectory variable node, and add an event variable node to the event detection module. The event detection module can work according to simple rules based on the target trajectory.

We show the V/M graph used for this application in Figure 8. The event detection module applies some simple

(a)



(b)

Figure 7: Sequences showing successful trajectory modeling. Object trajectory is shown in green, and predicted trajectory is shown in blue.

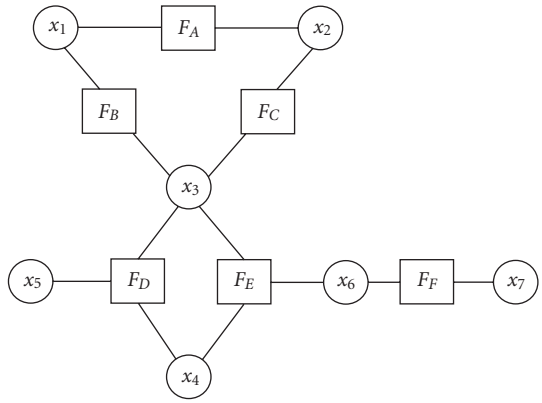


Figure 8: V/M graph for single-track-based event detection system.

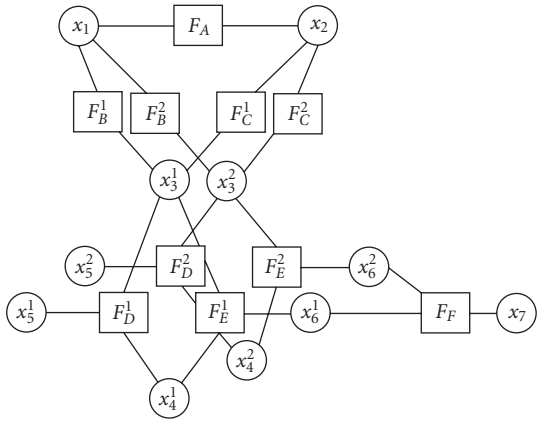


FIGURE 9: V/M graph for multiple-track-based event detection system.

rules on the trajectory to decide whether the person came out or went in. Specifically, it checks the direction of the vector from the start point of the trajectory to its endpoint and divides the direction space into two sets to make the decision. The decision is taken only when the track is lost, and not while the object is still being tracked. Thus, the event variable has three states, "no event," "came out," and "went in."

7.1. Results

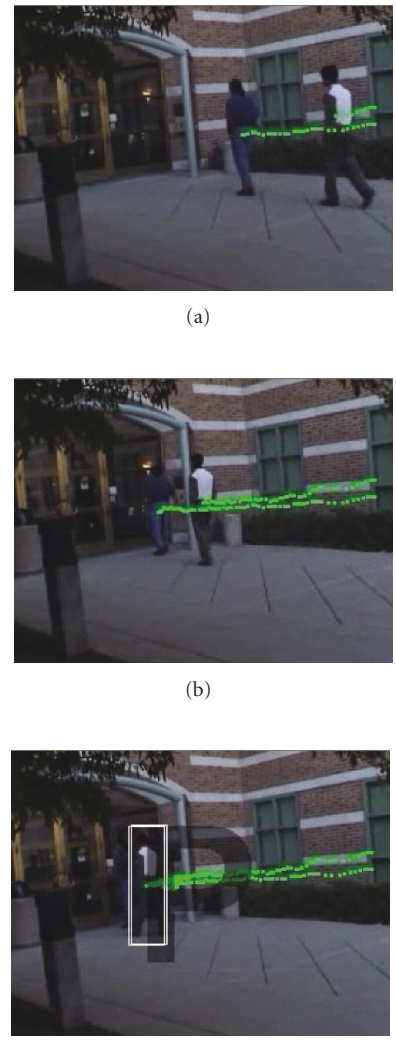
The results were quite encouraging. We got 100% correct event detection results owing to reasonable tracking performance. Some results are shown in Figure 3.

In theory, one could also design an event detection system that can give a feedback to the trajectory variable module. However, we will assume this to be uniform distribution in the following example, and we will not use it in any calculations.

8. APPLICATION: EVENT DETECTION BASED ON MULTIPLE TARGETS

We also designed applications for event detection based on multiple trajectories. Specifically, we designed applications to detect two people meeting in a café scenario, and piggybacking and tailgating at secure doors. The event detection module worked according to simple rules based on the trajectories of the targets.

We show the V/M graph used for this application in Figure 9. The event detection module applies some simple rules on the trajectories of two targets to decide whether the event has taken place or not. Specifically, to detect two people meeting, it checks that the trajectories of the two people converge and stay together for a while to make the decision. For detecting piggybacking or tailgating, it checks whether the trajectory of the two targets started together or not in order to infer whether the person swiping the card was aware of the presence of the other person behind him/her. If she/he was, then it is piggybacking, else it is tailgating.



(a)

(b)

(c)

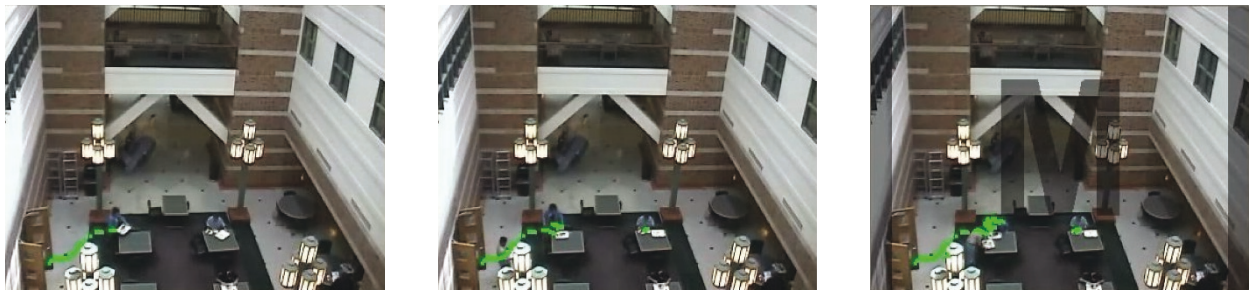
FIGURE 10: Sequence showing a detected "piggybacking" event. The first two images show representative frames of the second person following the first person closely, and the third image represents the detection result using an overlayed semitransparent letter "P."

8.1. Results

We implemented three different multitarget event detection systems, one for each type of AN event. Two of these were for detecting conditions at a secure door entry point into a building, that is, tailgating and piggybacking. The system could pick up 80% of the instances tailgating and piggybacking from a total of 5 examples in the video shot. The results are shown in Figures 10 and 11. Sample result for the event detection system for the third type of event ("meeting for lunch") is also shown in Figure 12. The results are preliminary examples of the potential of the system, and are

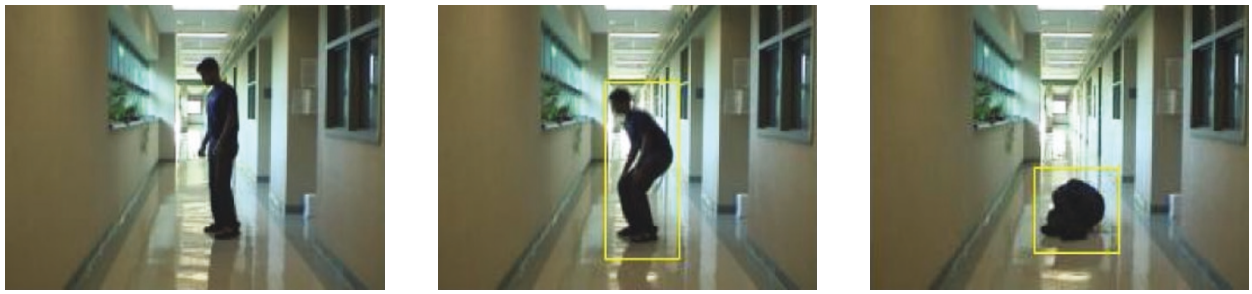
(a) (b) (c)

Figure 11: Sequence showing a detected "tailgating" event. The first two images show representative frames of the second person following the first person at a distance (sneaking in from behind), and the third image represents the detection result using an overlayed semitransparent letter "T."



(a) (b) (c)

Figure 12: Sequence showing a detected "meeting for lunch" event. The first two images show representative frames of the second person following the first person to the lunch table, and the third image represents the detection result using an overlayed semitransparent letter "M."



(a) (b) (c)

Figure 13: Proposed future work.

by no means indicative of how it compares to other event detection systems. The main difficulty in a comparison of different event detection systems is the lack of commonly agreed upon video data that can be used benchmark different systems in the research community.

9. CONCLUSION AND FUTURE WORK

In this paper, we have elaborated on a new framework for designing complex visual systems. We demonstrated effective use of these paradigms for home care and broad surveillance applications. We are working on extending out current work on using multiple modalities [20] in this framework. Also we are exploring using low-level features for abnormal event detection as shown in Figure 13.

ACKNOWLEDGMENTS

This work was supported in part by Advanced Research and Development Activities (ARDA) under Contract MDA904-03-C-1787 and in part by National Science Foundation Grant CCF 04-26627.

REFERENCES

[1] C. Harrington, S. Chapman, E. Miller, N. Miller, and R. Newcomer, "Trends in the supply of long-term-care facilities and beds in the United States," *Journal of Applied Gerontology*, vol. 24, no. 4, pp. 265–282, 2005.

[2] A. Sethi, M. Rahurkar, and T. S. Huang, "Variable module graphs: a framework for inference and learning in modular vision systems," in *Proceedings of IEEE International Conference on Image Processing (ICIP '05)*, vol. 2, pp. 1326–1329, Genova, Switzerland, September 2005.

[3] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001, special issue on codes on graphs and iterative algorithms.

[4] G. E. Hinton, "Products of experts," in *Proceedings of the 9th International Conference on Artificial Neural Networks (ICANN '99)*, vol. 1, pp. 1–6, Edinburgh, UK, September 1999.

[5] R. M. Neal and G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*, M. I. Jordan, Ed., pp. 355–368, Kluwer Academic Publishers, Norwell, Mass, USA, 1999.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the *EM* algorithm," *Journal of the Royal Statistical Society. Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[7] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's "belief propagation" algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 140–152, 1998.

[8] A. Sethi, *Interaction between modules in learning systems for vision applications*, Ph.D. thesis, University of Illinois at Urbana-Champaign, Champaign, Illinois, USA, 2006.

[9] B. J. Frey and N. Jojic, "A comparison of algorithms for inference and learning in probabilistic graphical models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 9, pp. 1392–1416, 2005.

[10] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)*, vol. 1, pp. 605–612, Madison, Wis, USA, June 2003.

[11] D. Heckerman, "A tutorial on learning with Bayesian networks," in *Learning in Graphical Models*, pp. 301–354, MIT Press, Cambridge, Mass, USA, 1999.

[12] D. Margaritis, *Learning Bayesian network model structure from data*, Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa, USA, 2003.

[13] J. Binder, D. Koller, S. Russell, and K. Kanazawa, "Adaptive probabilistic networks with hidden variables," *Machine Learning*, vol. 29, no. 2-3, pp. 213–244, 1997.

[14] E. Bauer, D. Koller, and Y. Singer, "Update rules for parameter estimation in Bayesian networks," in *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI '97)*, pp. 3–13, Providence, RI, USA, August 1997.

[15] W. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*, Chapman & Hall, London, UK, 1996.

[16] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian computer vision system for modeling human interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843, 2000.

[17] Y. Li, L.-Q. Xu, J. Morphett, and R. Jacobs, "An integrated algorithm of incremental and robust PCA," in *Proceedings of International Conference on Image Processing (ICIP '03)*, vol. 1, pp. 245–248, Barcelona, Spain, September 2003.

[18] J. Lim, D. A. Ross, R.-S. Lin, and M.-H. Yang, "Incremental learning for visual tracking," in *Advances in Neural Information Processing Systems (NIPS '04)*, Vancouver, British Columbia, Canada, December 2004.

[19] N. Johnson and D. Hogg, "Learning the distribution of object trajectories for event recognition," in *Proceedings of the 6th British Conference on Machine Vision (BMVC '95)*, vol. 2, pp. 583–592, Birmingham, UK, September 1995.

[20] A. Kushal, M. Rahurkar, F.-F. Li, J. Ponce, and T. S. Huang, "Audio-visual speaker localization using graphical models," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR '06)*, vol. 1, pp. 291–294, Hong Kong, August 2006.

Amit Sethi was born in Punjab, India. He went to Indian Institute of Technology, New Delhi, for B.Tech. degree in electrical engineering. He completed M.S. degree in general engineering, and Ph.D. degree in electrical engineering from University of Illinois at Urbana-Champaign. His academic research interests include machine learning, computer vision, event detection in videos, pattern recognition, and visual perception in humans. He is currently employed with ZS Associates, a sales and marketing consulting firm. He solves sales force sizing, structure, and performance tracking issues for his firm's clients. He also deals with customer choice modeling through survey-based conjoint analysis and primary research.

Mandar Rahurkar is fourth-year Ph.D. student at University of Illinois at Urbana-Champaign. His academic interests include machine learning applied to information retrieval and computer vision and multimodal signal processing.

Thomas S. Huang received his Sc.D. degree from MIT in 1963. He is William L. Everitt Distinguished Professor in the University Of Illinois, Department of Electrical and Computer Engineering and the Coordinated Science Lab (CSL); and he is a Full-Time Faculty Member in the Beckman Institute Image Formation and Processing and Artificial Intelligence Groups. His professional interests are computer vision, image compression and enhancement, pattern recognition, and multimodal signal processing. He is a recipient of IST and SPIE Imaging Scientist of the Year Award (2006), IEEE Jack S. Kilby Signal Processing Medal (2000) (corecipient with A. Netravali); International Association of Pattern Recognition, King-Sun Fu Prize (2002); Honda Lifetime Achievement Award (2000); Professor, Center for Advanced Study, UIUC; IEEE Third Millennium Medal (2000). He is a Fellow of the ACM, IEEE, and IAPR.