

VARIABLE MODULE GRAPHS: A FRAMEWORK FOR INFERENCE AND LEARNING IN MODULAR VISION SYSTEMS

Amit Sethi, Mandar Raturkar and Thomas S. Huang

ECE Department
University of Illinois-Urbana Champaign
email: {asethi, raturkar, huang}@ifp.uiuc.edu

ABSTRACT

We present a novel and intuitive framework for building modular vision systems for complex tasks such as surveillance applications. Inspired by graphical models, especially factor graphs, the framework allows capturing the dependencies between different variables in form of a graph. This enforces principled coordination and exchange of information between different modules. Breaking away from the traditional probabilistic graphical models the framework allows flexibility of design in individual modules by allowing different learning and inference mechanisms to work in a common setting. It also allows easy integration of more modules into an already functional system. We demonstrate the ease of building a complex vision system within this framework by designing a fully automatic multi-target tracking system for a video surveillance scenario. Favorable results are obtained for the tracking application.

1. INTRODUCTION

Complex vision tasks such as analyzing a surveillance video require multiple levels of processing. Depending on the objectives, the task can be divided into subtasks which can be handled by different modules. Significant advances have been made in designing individual modules such as background subtraction [1, 2] and tracking [3, 4]. This work aims to fill requirement of a framework needed to coordinate the processing and learning of different modules in a principled manner.

In generative models the observed variables are modeled to be dependent on the states of some hypothetical hidden variables. The dependence (or conditional independence) structure is captured by the connectivity of the nodes of a graph in graphical models such as factor graphs. However, the inference and learning mechanism on factor graphs and its derivatives (such as Bayesian networks) is limited to be

Bayesian probabilistic or related. Our framework does away with this restriction while maintaining the advantage of capturing the data dependency in a graph structure.

There has been some related work in combining different probability models that explain data. For example, Product of Experts [5] deals with models where the joint probability distribution of all the variable can be modeled as a product of different terms given by ‘experts’, or modules. Learning takes place by maximizing the data likelihood, or alternatively, minimizing the contrastive divergence [5]. However, it assumes that it is easy to perform Gibbs sampling on the variables. There has been other hierarchical models that represent the data using layers of hidden variables such as Helmholtz Machines [6], and layered linear structures based on Kalman filters [7]. The structure of these models is very restrictive making it difficult to come up with models useful for complex vision tasks. Our framework is very unrestrictive, yet principled enough to make assembling of complex vision systems very intuitive.

Theoretical background is discussed in Section 2, followed by the VM-Graphs in Section 3. Implementation of a tracking system using this framework is presented in Section 4. Results of the tracking system are described in Section 5. We conclude with a discussion on future work in Section 6.

2. THEORETICAL BACKGROUND

A factor graph is bipartite graph that can be partitioned into two sets of nodes [8]; *variable node* for each variable x_i , a *factor node* for each local function f_j . Factor graphs represent the joint probability distribution over the set of variables as a product of local probability distributions or ‘factors’ functions. Each of the factor functions use only a subset of the variables as arguments. An edge connects a variable node to factor node if and only if the variable is an argument of the factor function. Consider a function of 5 variables: $g(x_1, x_2, x_3, x_4, x_5)$ where joint probability between these variables can be expressed as:

This work was supported in part by National Science Foundation (USA) Grant CCF 04-26627, and in part by Advanced Research and Development Activity (USA) Contract MDA 904-03-C-1787.

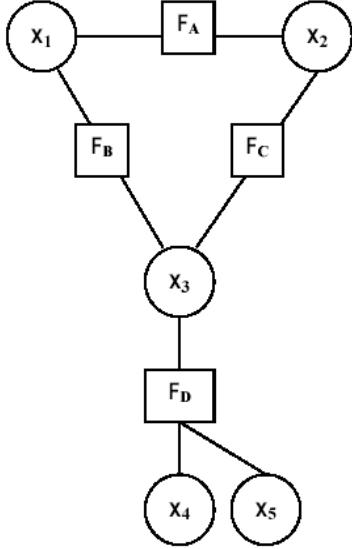


Fig. 1: Graphical representation for both the Factor Graph and the VM-Graph of the tracking example.

$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1, x_2) \times f_B(x_1, x_3) \times f_C(x_2, x_3) \times f_D(x_3, x_4, x_5) \quad (1)$$

In Eq. 1, the g is referred as a global (joint probability distribution) function and f_A , f_B , f_C , and f_D are the local (factor) functions. Factor graph corresponding to Eq. 1 is shown in Fig. 1.

Inference on probability distribution of all the variables based on all the evidence can be made by passing local ‘messages’ between nodes of the factor graph. The algorithm describing the mathematics of this message passing is known as the sum-product algorithm or belief propagation [8]. Learning schemes for parameters of Bayesian networks when all variables observed have been devised [9]. In case of some hidden variables, the use of EM-Algorithm to estimate the parameters of graphical models has also been made [10]. However, these schemes work on the assumption that batch data is available in advance to train the model. For applications where huge amount of training data is needed, it may not be practical to have batch training. Moreover, the model may be required to learn on the job (online learning) when deployed in a new scenario. Advances in online learning of parameters in probabilistic graphical models have also been made [11].

However, there are many problems in designing factor graphs for complex vision problems. First, the joint probability function may not be easily expressible as a product of its factor distributions. Second, the dependencies between

variables may not be known in form of a probability distribution, yet we may have non-probabilistic modules that we may want to integrate into the system. Third, many times it is not clear how a variable should affect another variable, and it would be best not to try to devise a complete message passing from every variable to all the other variables. Fourth, except for some work [11] that presents local online adaptation of parameters in a restricted set of graphical models, there is no framework to utilize the local connectivity of graphical models explicitly while devising a learning algorithm. Fifth, there is no easy way to incorporate fast low-level or preprocessing methods such as image filtering into factor graphs making them unnecessarily complicated while dealing with large structured data such as images. Thus, there is a need to relax some of the limiting restrictions of probabilistic graphical models while still utilizing the connectivity structure of the graph while making inferences and learning.

In the next section we generalize the idea behind factor graphs further to define a framework to build complex vision systems as an interconnection of modules called the VM-Graphs, or Variable-Module Graphs. VM-Graphs allow more flexibility in data processing, information representation, and learning than factor graphs, while retaining the structured representation of dependencies in the form of graphs.

3. VM-GRAPHS

In a VM-Graph, a variable node is connected to other variable nodes through module nodes. A background subtraction subsystem is an example of a module that connects the variable representing the image intensity and the variable representing the background mask. Each module processes information about the variables adjacent to it based on the parameters of the system. In a learning problem, we would want to estimate the parameters that are unknown or changing during the course of operation to optimize a cost function. The information exchange between the variables is done according to a scheduling algorithm. Variable nodes, module nodes, and scheduling algorithm in a VM-Graph are generalized analogs of variable nodes, function nodes, and message passing sequence respectively in a factor graph. In addition to these direct analogs of the ingredients of a factor graph, a VM-Graph also allows learning methods for module nodes to improve local cost functions. Now we explain the framework in more detail.

The variables interface with each other indirectly via module nodes. Tasks for the variable nodes are: 1) Combine information about the variables coming from various modules and present it to the user on demand. Note that combining this information need not mean taking a simple normalized product of probability distributions over the

variable (as in the case of factor graphs), as long as the right inference is represented by a local maxima in the messages and their combination. 2) Send the combined information to the module nodes it is connected to. Note that sending this information to a module need not mean taking a simple product over all the messages (probability distributions) to the variable node by the other modules.

A module node is connected only to variable nodes that serve as its (input or output) arguments. The module node has two functions. **1.** Combine the information available to it from all the variable nodes that it is adjacent to and present it in a form useful to any of these variable nodes. This process need not be limited to marginalization of the joint probability distribution over these variables as in a factor graph as long as the right inference represents a local maxima in the messages and their combination (No opinion about a variable is passed as a uniform distribution). **2.** Learn the internal parameters of the module based on the information available by the variable nodes adjacent to it. Parameters common to more than one module node have to be learnt by considering the set of these module nodes as a super-module node.

Thus, the framework can incorporate non-probabilistic modules while building a system. The learning and inference is local and can be online. This makes it easier on the memory requirements, algorithmic complexity, and time complexity. Minimizing contrastive divergence as in Product of Experts [5] is but one way of learning in such a scenario and this method can be thought of as its generalization.

4. APPLICATION TO PERSON TRACKING

In the person tracking application, we identify five variables that affect inference in a frame. The intensity map (pixel values) of the frame, 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 Fig 1. 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 online learning version of eigen background model [2] as system parameters. 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_B 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 dimensions of a person over the background map and normalize its output as a map of the probability of a person's position. Module F_C 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 represent the appearance of a person. The module maintains an online learning version of eigen-appearance of the person as system parameters based on a modification of a previous work [4]. 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 from x_4 and x_5 to x_3 .

The scheduling algorithm is as follows:

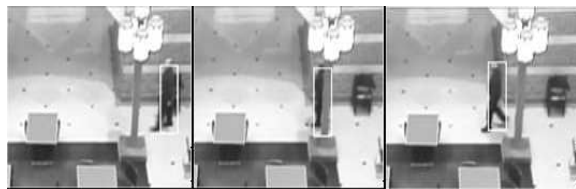
1. Initialize a background model.
2. If a large contiguous foreground area is detected, initialize a person detection module F_B , and tracking related modules F_C and F_D .
3. Initialize the position of the person in the previous frame as the most likely position according to the background map.
4. 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 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 eigen background subtraction.
6. Propagate a message from x_2 to F_C in form of a background probability map.
7. Propagate a message from F_C to x_3 in 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 form of samples of likely position.
9. Propagate a message from F_B to x_3 in form of probabilities at samples of likely position as defined by the eigen appearance of the person maintained at F_B .
10. Combine the incoming messages from F_B , F_C , and F_D to 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 eigen models at F_B , and F_C .
13. Update motion model at F_D .

5. RESULTS



(a) Occlusion of the object.



(b) Nearly camouflaged object.



(c) Multiple objects.

Fig. 2: Results of the tracking application (white rectangles around targets) zoomed in and cropped for clarity.

We ran our person tracker in both single person and multi-person scenarios using monochrome indoor sequences 320 by 240 in dimensions using a fixed camera. People appeared to be as small as 7 by 30 pixels. Note that no elaborate setup was made, and no prior training was done. The tracker was required to run and learn on the job, fresh out of the box. The tracker could easily track people successfully after brief but complete occlusion, owing to the integration of a background subtraction, eigen appearance, and motion models. The system successfully picks up and tracks new persons automatically when they enter 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 eigen appearance 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 a complete occlusion, widely varying appearance of people, and background camouflage. A slight modification of the model also

yields a multiple person tracker. The results are shown in Fig. 2. Running on unoptimized MATLAB code on a 2GHz computer, the system performs at about 2 frames a second.

6. CONCLUSIONS AND FUTURE WORK

Framework for building modular vision systems for complex tasks is presented. We generalize over existing graphical models to allow more flexibility of design of complex vision systems which are specifically geared towards making use of the structural dependency of a variables in a system as in case of a graphical model. Ease of combining modules with their own learning system is demonstrated by a building a tracking system within this framework.

In future, current implementation will be extended for complicated tasks such as behavior analysis and combining multiple modalities in surveillance scenarios. We will also be exploiting the time-based correlation between data in consecutive frames to a larger extent.

7. REFERENCES

- [1] C. Stauffer and W.E.L. Grimson, "Adaptive background mixture models for real-time tracking," *Proceedings of the CVPR*, pp. 246–242, 1999.
- [2] N.M. Oliver, B. Rosario, and A.P. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Trans on PAMI*, pp. 831–843, 2000.
- [3] M. Isard and A. Blake, "Condensation; conditional density propagation for visual tracking," *International Journal of Computer Vision*, pp. 5–28, 1998.
- [4] J. Lim, D. Ross, R.S. Lin, and M.H. Yang, "Incremental learning for visual tracking," *Proceedings of the NIPS*, 2004.
- [5] G. Hinton, "Product of experts," *Proceedings of the ICANN*, pp. 1–6, 1999.
- [6] G.E. Hinton, P. Dayan, B.J. Frey, and R.M. Neal, "The wake sleep algorithm for unsupervised neural networks," *Science*, vol. 268, pp. 1158–1161, 1995.
- [7] R.P.N Rao and D.H. Ballard, "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects.," *Nature*, vol. 2, pp. 79–87, 1999.
- [8] F.R. Kschischang, B.J. Frey, and H.A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, 2001.
- [9] Z. Ghahramani, "Learning dynamic Bayesian networks," *Lecture Notes in Computer Science*, vol. 1387, pp. 168–197, 1998.
- [10] N. Jojic and B.J. Frey, "Learning flexible sprites in video layers.," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001.
- [11] J. Binder, D. Koller, S. Russell, and K. Kanazawa, "Adaptive probabilistic networks with hidden variables," *Machine Learning*, pp. 213–244, 1997.