The Dynamic Tree [1] (DT) Bayesian Network is a powerful analytical tool for image segmentation and object segmentation tasks. Its hierarchical nature makes it possible to analyze and incorporate information from different scales, which is desirable in many applications. Having a flexible structure enables model selection, concurrent with parameter inference. In this paper, we propose a novel framework, dynamic factor graphs (DFG), where data segmentation and fusion tasks are combined in the same framework. Factor graphs (FGs) enable us to have a broader range of modeling applications than Bayesian networks (BNs) since FGs include both directed acyclic and undirected graphs in the same setting. The example in this paper will focus on segmentation and fusion of 2D image features with a linear Gaussian model assumption.

Index Terms— dynamic factor graphs, sum-product algorithm, linear Gaussian models, data fusion, data segmentation

1. INTRODUCTION

The Dynamic tree (DT) [1] was originally invented in order to model images in a probabilistic and hierarchical manner. The DT is developed from earlier work on Tree-Structured Belief Networks (TSBN) [2], which provide a natural way of modeling images within a probabilistic framework. The TSBN has a desirable hierarchical structure, but suffers from “blocky” segmentation due to its fixed parent-child structure. The DT was invented to solve this problem by allowing the structure of the network to change in order to fit the underlying spatial distribution. A common algorithm to infer a distribution at a node in both TSBNs and DTs is Pearl’s message passing [3], which gives an attractive linear-time algorithm when the directed graph structure does not contain any loops.

Bayesian networks (BNs) and Markov random fields (MRFs) are classes of probabilistic graphical models corresponding to directed acyclic graphs (DAGs) and undirected graphs (UGs), respectively. Both the DT and TSBN were invented based on the BN, so both are limited to only DAGs. In order to incorporate both a DAG and a UG together in the same framework, the factor graph (FG) [4] was invented along with its time-efficient inference algorithm sum-product algorithm [4], which is similar to Pearl’s message passing. In this paper, we propose a novel methodology that inherits the flexible structure capability of the DT and FG. The new architecture is called a dynamic factor graph (DFG). The DFG can incorporate both a DAG and UG simultaneously, and is capable of changing structure (model selection). To demonstrate the power of DFGs, an example application on data fusion will be illustrated.

Data fusion is an increasingly important and much needed technology due to the availability of many sensors for remote sensing applications. Moreover, there is a need for data fusion algorithms that are able to provide both parameter estimation and the corresponding uncertainty in the parameter estimates. This requirement is ideally suited for the probabilistic-based framework proposed in this paper. We develop a novel probabilistic framework for data fusion based on the DFG, which is inherently probabilistic. Additionally, the new framework is designed to incorporate multiple features and multidimensional data.

In the next section an overview of factor graphs and dynamic factor graphs is presented. A description of the DFG for data fusion is presented next, and the DFG structure optimization is then introduced. The remaining sections explain the incorporation of extracted multiple features into the DFG framework for segmentation and sensor fusion. Finally, results are presented.
2. DYNAMIC FACTOR GRAPHS

In general, a factor graph (FG) is a bipartite graph that expresses how a “global” function of many variables decomposes into a product of “local” functions [4]. In this scenario, the global function is the full joint distribution over all the variables in the structure and the local functions can be thought of as conditional distributions over subsets of the variables. The conditional distribution is obtained by applying the conditional independence property of the network. The circles and the squares in Fig. 1 represent variable nodes and function nodes, respectively. There is a variable node for each variable and a function node for each factor. The variable node for \( V \) is connected to the function node \( f_i \) if and only if \( V \) is an argument of \( f_i \). In Fig. 1 a set of random variables \( \{A,B,C,D,E\} \) is connected as shown. The joint probability distribution over all the variables is

\[
P(A, B, C, D, E) = f_1(A)f_2(B)f_3(C|A, B)f_4(D|C)f_5(E|C)
\]

\[
= P(A)P(B)P(C|A, B)P(D|C)P(E|C)
\]

2.2. Dynamic factor graphs

A dynamic factor graph is a factor graph whose joint probability \( P(Z, V) \) includes a random variable \( Z \) that defines the network structure. The random variable \( Z \) is in the form of a matrix whose \( i \)th row \( j \)th column element is written as \( Z_{ij} \). Each element \( Z_{ij} \) indicates the connection between the \( i \)th child node and the \( j \)th parent node. \( Z_{ij} = 1 \) when such a connection is present, and 0 otherwise. The random variable \( V \) is the set of nodes in the network structure. Normally the nodes in a network can be separated into 2 types: 1) hidden nodes and 2) observed nodes. Let \( V = \{X, Y\} \) where \( X \) and \( Y \) are the set of hidden nodes and observed nodes in the network, respectively. \( X \) and \( Y \) can be any type of random variable. In many applications they are taken to be discrete-valued, but in this paper we take them to be continuous-valued random variables, which is generally more difficult [5].

So far, \( Z \) describes qualitatively how variables in the network interact but does not indicate the strength of each interaction. To describe such a quantity, the conditional probability distribution (CPD) for each parameter is required. The CPD between a child node \( i \) with a parent node \( j \) is usually written in the form \( P(V|V_j) \). In this paper we assume the CPD takes the form of a linear Gaussian model which will be discussed below.

The goal of using a DFG is to calculate the optimal structure \( Z^* \) that can best explain the given set of observations \( Y \). That is we want \( Z^* = \text{argmax}_Z P(Z|Y) \). In terms of the data fusion problem proposed in this paper, \( Z^* \) encodes a non-redundant set of objects at the root nodes and also connects each leaf node upward to its most likely root node. It is also worth mentioning that the posterior distribution of each root node captures the (fused) information obtained by combining all the information from its corresponding leaf nodes.

3. DATA FUSION FRAMEWORK

3.1. Overview of the algorithm

In this DFG framework, the data and the underlying structure are represented by a factor graph. It is usually the case that having multiple features can enhance the classification and segmentation result. This framework is designed in order to incorporate multiple features, each of which can take the form of a multidimensional feature vector. Moreover, the random variables are also assumed multivariate normally distributed. The overview of the framework is illustrated in Fig. 2.

3.2. Mathematical model of the problem

Let \( D \) denote \( \{D_1, D_2, ..., D_k\} \) a set of \( N \) acquired data objects (sonar images in this case). From each acquired datum \( D_i \) \( F \) multidimensional feature vectors \( D_i^{(1)}, D_i^{(2)}, ..., D_i^{(F)} \) are extracted. The architecture of this framework comprises \( F \) blocks, each of which contain the same number of nodes and share the same topological structure \( Z \). Let \( X \) and \( Y \), random variables in the whole network, denote the hidden variables and observed variables, respectively. Additionally, the random variable \( X \) can be separated into \( F \) independent random variables according to the feature extraction. That is \( X = \{X^{(1)}, X^{(2)}, ..., X^{(F)}\} \), where \( X^{(f)} \) denotes the hidden variable \( X \) that contributes to feature \( f \in \{1, ..., F\} \). Furthermore, we can think of \( X^{(f)} \) as a set of hidden variables (nodes) contained in the block \( f \) allowing us to further factorize the
variable as \( X^{(f)} = \{X_1^{(f)}, X_2^{(f)}, \ldots, X_M^{(f)} \} \) where \( M \) denotes number of hidden variables in each feature. The same factorization applies to \( Y \), that is \( Y = \{Y^{(1)}, Y^{(2)}, \ldots, Y^{(P)} \} \) and \( Y^{(f)} = \{Y_1^{(f)}, Y_2^{(f)}, \ldots, Y_N^{(f)} \} \).

In each block, the number of the leaf nodes is equal to the number \( N \) of acquired data (e.g. images in this application). Each of the feature vector \( D_i^{(f)} \) will be plugged into the corresponding leaf node \( Y_i^{(f)} \), which is the \( i \)-th leaf node in the block \( f \). At this point we also say \( Y_i^{(f)} \) is observed and take a vector value \( D_i^{(f)} \). Again, the structure matrix is \( Z = (Z_{ij}) \), where \( Z_{ij} = 1 \) when the child node \( i \) is connected to the parent node \( j \) and 0 otherwise.

The main goal of DFG is to calculate the optimal structure \( Z^* \). Hence we have \( Z^* = \text{argmax}_Z P(Z|Y) = \text{argmax}_Z (\log P(Y|Z) + \log P(Z)) \).

We assume that each feature \( f \) is mutually independent so the conditional distribution can be rewritten as

\[
P(Y|Z) = \int P(X, Y|Z) dX = \int \prod_{f=1}^{F} P(X^{(f)}, Y^{(f)}|Z) dX^{(f)} = \prod_{f=1}^{F} P(Y^{(f)}|Z).
\]

By taking \( \log P(Y|Z) = \sum_{f=1}^{F} \log P(Y^{(f)}|Z) \), finally, we will have \( Z^* = \text{argmax}_Z (\sum_{f=1}^{F} \log P(Y^{(f)}|Z) + \log P(Z)) \).

Here we call the quantity inside the argmax the cost function, and we use it to find the structure \( Z^* \) that maximizes the cost function.

In real-world applications it is likely that each feature would be weighted differently by human experts. Also we want the optimal structure to be as simple as possible, which means that we would penalize structures with more complexity. Consequently, we modify the original cost function to get the final cost function \( C(Z) \) as follows

\[
C(Z) = \sum_{f=1}^{F} \omega_f \log P(Y^{(f)}|Z) + \log P(Z) + \text{pen}(Z)
\]

The first term of \( C(Z) \) corresponds to the likelihood of the structure \( Z \) given the evidence \( Y \), and user-defined weights \( \omega_f \) is the user-defined weights with the constraint \( \sum_{f=1}^{F} \omega_f = 1 \). The second term corresponds to the prior probability of the structure which can be obtained from, for example, simple segmentation. This helps us to shape the resulting structure. The last term \( \text{pen}(Z) \) is the penalty term, which is a function of the structure \( Z \).

In order to calculate the optimal structure \( Z^* \) such that \( Z^* = \text{argmax}_Z C(Z) \), the sum-product algorithm is exploited to calculate the joint \( P \left( X_i^{(f)}, Y^{(f)} \right) \), and hence we can calculate the likelihood of the structure given the evidence as \( P \left( Y^{(f)}|Z \right) = \int_{X_i^{(f)}} P \left( X_i^{(f)}, Y^{(f)}|Z \right) dX_i^{(f)} \), which is tractably computable. Recall that a structure \( Z \) determines a unique factorization over the entire joint, so it would be difficult to infer a distribution at a node without the sum-product algorithm. For discrete-valued \( X \), we can obtain similar form by simply switching \( \int_X \) to \( \sum_X \).

3.3. DFG structure optimization

The structure \( Z \) is initialized with the structure in Fig. 4 (a), and then the structure is perturbed repeatedly and progressively. The cost function \( C(Z) \) is calculated for each and every structure \( Z \) until the iteration stops. At the end the structure that gives maximum value of \( C(Z) \) is the optimal structure \( Z^* \). In general, large optimization problems can be solved successfully using a stochastic search algorithm [6]. The structure optimization in this paper is an NP-hard problem, and simulated annealing (SA) [7] is chosen to be implemented in this work to solve the problem.

3.3. Interpretation of the model

DFGs have hierarchical properties so the information can be visualized at different scales. This enables an interpretation of the data based on a generative model, which is often
desirable. In the sensor fusion application illustrated by Fig. 4, we have 3 levels to consider: 1) root nodes represent the estimate of the true feature, 2) middle-layer nodes represent sensor platform ID, and 3) leaf nodes represent the realization of the features.

![Learning curve in SA](image)

(a) Fig. 5 (a) The learning curve in SA. The black curve represents the evaluated cost function that is accepted. The grey line tracks the unaccepted ones. (b) The temperature schedule of SA used in this experiment.

A root node of the structure corresponds to a unique target that generates the feature vectors at its corresponding leaf nodes. For example, in Fig. 4 (b) there are 3 root nodes, and that means the algorithm can detect 3 targets from the 9 redundant images. The left-most tree in Fig. 4 (b) contains 1 root node, 4 leaf nodes and 2 middle-layer nodes, which means that the images \( \{D_1, D_2, D_3, D_4\} \) are from the same real target represented by the root node. Additionally the structure indicates that \( \{D_1, D_2\} \) and \( \{D_3, D_4\} \) are taken from platform #1 and #2, respectively.

The reason to pick the initial structure as in Fig. 4 (a) becomes clear in that we prepare for the worst case scenario where each leaf node is distinct from other leaf nodes. In other words, there are no redundant images.

4. EXPERIMENTAL RESULTS

To demonstrate the DFG algorithm, we use the sensor fusion problem as an example. Here we assume all parameters (means, covariance matrices, \( w_f \)) of the DFG are known from the learning algorithm. We have 9 sonar images obtained over 3 real targets taken by 2 sensor platforms. Obviously, the 9 images are redundant since in the reality there are only 3 targets. We extracted 2 continuous-valued features (x-y location and x-y correlation length) from each image, each feature is 2D. The 2 features are plotted in Fig. 3. We also assume that the underlying generative model of the features is linear Gaussian. That is \( P(A = a|B = b) = N(a; \mu_a + W_{B \rightarrow A}(b - \mu_b), \Sigma_a) \) if B is the only parent node of A, and \( P(B = b) = N(b; \mu_b, \Sigma_b) \) if B is a root node. From the experiment and from the human expert, we know that a proper penalty function is \( \text{pen}(Z) = -2r - r^2 \) which put penalty more in the more complicated structure, in this case, indicated by the number of root nodes \( r \). The SA parameters are set as follows. \( K = 1, T_0 = 25, T_{\text{limit}} = 0.1, \alpha = 0.9. \)

The resulting optimal structure we get from the DFG in this experiment is illustrated in Fig. 4, which perfectly matches the ground truth in Fig. 3 (a) and (b). The posterior distribution at each root node describes the likelihood of the fused location. The maximum a posteriori (MAP) at each root node is actually the estimated (or fused) location of each group (plotted as the black star in Fig. 3). The covariance matrix of the posterior probability determines estimation uncertainty (plotted as the grey ellipse in Fig. 3). The number of iterations depends on the SA setting in which case the temperature schedule is shown in Fig. 5. The runtime depends on the size of the network, and the number of parameters defining the underlying distribution. Here the run-time per iteration is about 0.8 sec on Intel® Pentium® D CPU 2.80GHz, 4 GB RAM.

5. CONCLUSION

The DFG is a powerful data segmentation and fusion algorithm that is capable of model selection at the same time as data segmentation and data fusion. Furthermore, the entire framework is based on Bayesian estimation. The linear-time inference can be achieved on tree-structure networks via the existing time-efficient sum-product algorithm. The framework can be implemented on both DAGs and UGs since it is based on a FG. This will help it accommodate more complex models in other applications.

ACKNOWLEDGEMENTS

This work was supported by a grant from the Office of Naval Research (N00014-07-1-0698). The authors also wish to thank J. Tory Cobb for helpful discussions.

6. REFERENCES