Deformable Bayesian Network: A Robust Framework for Underwater Sensor Fusion

K. Kampa, E. Hasanbelliu, J. T. Cobb, Member, IEEE, J. Principe, Fellow, IEEE, K. C. Slatton, Senior Member, IEEE

Abstract

The dynamic tree (DT) graphical model is a popular analytical tool for image segmentation and object classification tasks. A DT is a useful model in this context because its hierarchical property enables the user to examine information in multiple scales and its flexible structure can more easily fit complex region boundaries compared to rigid quadtree structures such as tree-structured Bayesian networks. This paper proposes a novel framework for data fusion called a deformable Bayesian network (DFBN) by using a DT model to fuse measurements from multiple sensing platforms into a non-redundant representation. The structural flexibility of the DFBN will be used to fuse common information across different sensor measurements. The appropriate structure update strategies for the DFBN and its parameters for the data fusion application are discussed. A real-world example application using sonar images collected from a survey mission is presented. The fusion results using the presented DFBN framework are shown to outperform state-of-the-art approaches such as the Gaussian mean shift and spectral clustering algorithms. The DFBN's complexity and scalability are discussed to address its potential for a larger data set.

Index Terms
dynamic tree, tree-structured, Bayesian network, deformable structure, sum-product algorithm, sonar, sensor fusion

I. INTRODUCTION

In many underwater sensing applications, human operators are required to manually combine or fuse multiple measurements of the same region or object into a single observation with no redundant information. This is usually the case in underwater surveys, where the path trajectories of diverse sensing systems overlap the same sensed region many times, producing a heterogeneous set of measurements from the same target. The answer most commonly sought regarding a particular sensed target is a label and an accurate geographic position linked to the set of corresponding sensor data gathered from the sensing systems. In this paper, we use probabilistic graphical models...
Figure 1: Underwater sensing platforms survey a field of targets (stars). The targets are seen multiple times from a variety of passes. The sensor fusion task is to correctly associate target labels between the various sensing platforms and provide an estimated position. In this figure, the y-direction and x-direction of a sample image is associated with their corresponding platform direction (denoted by arrow) and side-scan direction respectively.

as the analytical framework for automatically combining redundant sonar image target information gathered from multiple underwater sensing platforms.

Fig. 1 depicts an underwater sensing scenario where a platform with a side-looking sonar has sensed a group of targets on various passes during a typical survey mission. When all the survey data is collected, an automated target recognition algorithm and/or a human operator reviews all the sonar images and marks locations in the imagery where targets are discovered. Following this review, the multiple target instances must be combined or fused into knowledge about the target type and location on the seafloor. In this fusion scenario, there is uncertainty about each target’s absolute position derived from imprecision in the platform navigation system. Thus, targets that are located close to one another must be discriminated by their structures. Additionally, images of a given target are distorted between passes due to various ranges and sensing angles thus any clustering or classification algorithm must be invariant or anticipate such distortions. These distortions are depicted in the small image chips in Fig. 1 where the three targets look different depending on the location and heading of the sensing platform. The graphical model formulation we describe in this paper uses the seabottom texture and target image features along with location information to discriminate and group targets located within the navigational uncertainty of the sensing platform.

A spectrum of statistical approaches have been proposed to resolve this particular problem as explained comprehensively in [1], however, due to space limitations, we shall address prior work related to graphical models. Various graphical model formulations have been applied to different sensor fusion tasks, and one of the first attempts
can be found in [2] where heterogeneous sensors worked cooperatively to achieve a common goal. In the paper, the data fusion problem is mapped into a graphical model and the message-passing scheme is derived to fuse the data efficiently. Moreover, combining several sensors using graphical models yields robustness to the framework as reported in [3].

Approaches to transform data association problems into inference problems on graphical models are formally discussed at length in [4]. The work focuses on efficiently solving the problem using local message-passing algorithms, which can be seen as solving optimization problems in a distributed manner where the information exchanges are allowed only among the neighboring nodes on the graph. It is a well-known fact that message-passing algorithms give approximate and exact inference on graphs with and without loops, respectively. Recently, generalized belief propagation has been applied on sensor network localization problems on graphs with loops [5]. Graphical models have been applied in tracking problems with complex topology of sensor networks as explored in [6], and the target locations are inferred accurately using message-passing algorithms. In the cases where complex probability distributions are utilized to capture more details of the system, nonparametric message-passing algorithms are required for efficient inference [7].

Graphical model approaches have been applied to traditional decentralized data fusion [8], a framework where there is no single central fusion center and the communications only happen on a node-to-node basis. In such frameworks, the network of sensors is represented by a graph, and the communications can be calculated efficiently using message-passing algorithms [9], [10].

In general, the graphical models can be classified into two main categories: 1) fixed structure and 2) flexible structure. Many of proposed sensor fusion applications mentioned earlier fall into the first category, where the graph structure is fixed throughout the learning and inference process. However, the main drawback of this approach is the discrepancy between the data sample and the fixed graph structure. It is true that the fixed graph structure can be learned a priori in a model selection process, but in some situations the training data samples are obtained at great expense. Frequently, the training data samples are not available a priori for traditional model selection. In this case the data-graph discrepancy significantly affects the resulting accuracy. Therefore, it is crucial for a framework of this type to be capable of performing model selection simultaneously with the inference process. Hence, the graphical models in the second category, where the graph structure is flexible and driven by the data samples, are developed to mitigate the discrepancy. Some successful examples of such models include the dynamic tree (DT) graphical model, a probabilistic graphical model whose structure is flexible along the inference and learning process, which has been successfully implemented in image segmentation applications [11]–[13]. Furthermore, the analytical framework provides a flexible hierarchical structure which is desired in many image processing applications, e.g., texture segmentation across complex region boundaries.

In addition to its hierarchical structure, DTs are based on a probabilistic framework, using probability distributions to model the relationships amongst the sensor measurements and thus explicitly providing an estimate of uncertainty of the DT maximum a posteriori (MAP) solution. In other words, the solution of a DT is in terms of a posterior marginal distribution of each node rather than a single value. The DT solution can be contrasted with an unsupervised
clustering routine such as $k$-means clustering where the solution does not have an explicit measure of uncertainty. Moreover, our proposed method outputs the comprehensive relationship among observations in the dataset in terms of a graph structure which contains rich information regarding the data. Such graph structures can also be beneficial in many high-level applications requiring a good representation of the inter-relationship among the observations.

In this paper we extend the merits of the DT framework to solve an underwater sensor fusion problem. We propose a hierarchical DT architecture for the sensor fusion task named deformable Bayesian networks (DFBNs). The ultimate goal of the proposed framework is to reduce the redundancy in the given data set by means of clustering algorithms whose outputs are tree-structured graphs. The summarized flow diagram of the framework is illustrated in Fig. 2. The DFBN flow diagram presented here starts with raw measurements from mission surveys. The measurements then undergo the feature extraction process, whose details are problem-dependent, that produces a feature vector for each object of interest. The DFBN framework takes as input the feature vectors and automatically outputs a graph structure from which a number of the clusters and estimated associations can be interpreted directly. The flow diagram of Fig. 2 is significantly detailed in Alg. 2.

The details of the graph-structural interpretation are depicted in Fig. 3. The top of the figure illustrates the sensing scenario, where each tree represents a single target of interest. The bottom of the figure illustrates the structure of each tree. The root node, at the top level of the tree, represents the target in question, the intermediate level of the tree represents the particular sensing platform that collected the measurements on the target, and the leaf nodes, at the bottom level of the tree, are represented by the raw measurements. Prior knowledge about sensor platform identification and navigation information is easily incorporated into this hierarchical formulation and will ease the computational burden on the algorithm.

It is important to contrast the sensor fusion approach just explained with the large body of research in simultaneous localization and mapping (SLAM), which has the capacity to accurately determine spatial locations of targets within the field of view of a mobile sensing platform. SLAM is a popular state-space estimation framework, having rigorous probabilistic interpretation, that builds a map of the target locations and a reconstructed sensor platform path by estimating target and platform positions (with associated uncertainty) within a Bayesian filtering framework (e.g., Kalman, extended Kalman, unscented Kalman, fast Kalman and particle filters) [14]–[19]. With appropriate sensors, accurate dynamic models of the sensing platform, and correct landmark association between sensing passes (a.k.a.
Scenario Level - Forest of dynamic trees, each of which is an actual target

Figure 3: DFBN sensor fusion architecture. In a sensing scenario data is organized by a forest of dynamic trees. Each tree represents the information known about a target sensed in the environment. Within each tree, sensor measurements (leaf nodes) are linked to the true state of the target (root nodes) through an intermediate set of nodes that define the uncertainty interjected by a particular sensor.

The data association problem), a highly accurate spatial map of a sensed area can be constructed [20]–[25].

The approach we present here can be seen as a subset of the SLAM framework with the following special assumptions. First, we assume that there is no continuous stream of sensor and platform navigation data; we receive only snapshots of relevant targets in the environment discarding their orders, therefore, the dynamic model of the sensing platform is not utilized in the solution. Finally, the multiple views of a target can occur at disparate times using different sensing platforms, e.g., combining current with historical measurements. Within our sensor fusion paradigm, the measurements we receive depend upon the image characteristics of the targets of interest, rather than the usefulness of the targets as landmarks in deriving a spatial map. Nevertheless, SLAM and DT sensor fusion share the need for the data association [26]–[28] and there may be applications where they can be beneficially coupled.

It is also important to contrast our proposed method with the joint compatibility branch and bound (JCBB) algorithm [27], a widely-adopted approach to data association in computer vision and SLAM communities due to its robustness and efficiency. Generally JCBB performs data association by evaluating the compatibility between two sets of partially overlapping observations, namely joint compatibility, which makes JCBB more robust than the compatibility calculated from each pair of single observations. The heuristic approach using Mahalanobis distance to confine the search domain enables efficient computation using JCBB over joint compatibility methods in general. JCBB is specifically designed to incorporate the platform dynamic information with observations in an online manner. In fact, the JCBB can also be implemented in batch mode, however, special care should be taken as the predicted complexity of $O(1.53^n)$ suggests that for $n > 25$ (where $n$ is the number of observations), the complexity will be considerably high. In most of our batch mode scenarios, the number of data samples can easily exceed such
a number. On the other hand, our proposed method is designed specifically for a static paradigm with no platform
dynamics available at all, which can be regarded as a subset of JCBB, and is also applicable to data batch clustering
in general.

This paper uses ideas of both the SLAM and graphical model communities, so occasionally there might be
some terminological and notational confusions which are explained as follows. First, the word measurement and
observation refer to the raw measurement acquired from the sensor platforms, and in some contexts both words are
used interchangeably. Second, the term feature refers to the discriminative numerical information extracted from
each measurement, and is normally arranged in vector form, and hence is called a feature vector. Third, in the
graphical model community, \(X\) and \(Y\) commonly refer to hidden and observed variables respectively whereas in
SLAM community they are commonly used to describe either 2D (x and y) coordinates or the state and observed
variables of a discrete-time dynamical system. To avoid confusion, here we use the mathematical font of \(X\) and
\(Y\) or \(x\) and \(y\) to denote the hidden and observed variables, respectively, and regular font \(x\) and \(y\) to describe 2D
coordinates. Nevertheless, the meaning of the variables will be clear in the context of their appearance.

The rest of the paper presents a solution to the sensor fusion problem of combining redundant sensor information
using the analytical framework of DFBNs. Summaries of the properties of Bayesian networks and DTs are presented
in the next section. A description of the DFBN sensor fusion architecture is then introduced. Procedures for
optimizing the DFBN and thus finding the sensor fusion solution are described next. The remaining sections explain
the incorporation of target and sonar image texture features into the DFBN framework for sensor measurement
discrimination and include results from a real-world sensing scenario. The results from DFBNs are compared with
state-of-the-art methods, and the computational complexity is discussed. Additionally, the DFBN performance is
evaluated on two simulated datasets, ONRUF1 and ONRUF2700, whose number of observations is considerably
large. Finally, the paper is concluded in the last section.

II. PROBABILISTIC GRAPHICAL MODELS

In a Bayesian network graphical model, the joint probability of a set of random variables (or nodes) is defined
by the connections of the graph, which in turn, define the conditional probability relationships amongst the nodes.
In Fig. 4, a set of discrete random variables \(A, B, C, D\) is connected as shown. A directed link from parent node
to child nodes defines the probability state of the child conditioned on the parent. For example, the arrow from
node \(A\) to node \(C\) denotes the conditional probability \(P(C|A)\). Root nodes or nodes which do not have a parent
are defined through a prior probability. Through this notation, the joint probability of the graph in Fig. 4 can be
factored as

\[
\]  

(1)

Normally, there are two types of variables or nodes in a graphical model: 1) observed or evidential variables,
and 2) hidden or unobserved variables. Most of the time, the goal is to infer the probability distribution of a subset
of hidden nodes given the evidential nodes via posterior distribution derived from Bayes’ rule and marginalizing
out the hidden nodes. For example, to infer the probability of the states of random variable \(C\) in Fig. 4 when the
Figure 4: Bayesian network graphical model. Nodes A, B, C, and D are random variables, and links between the nodes define conditional probability relationships.

values at nodes $B = b$ and $D = d$, it requires that we sum over the probability mass function of variable $A$ in the numerator, and $A$ and $C$ in the denominator of (1). This expression, i.e., the posterior, can be written according to Bayes’ rule as

$$p(c|b,d) = \frac{\sum_{a \in A} p(a)p(b|a)p(c|a)p(d|b,c)}{\sum_{(a,c) \in A \times C} p(a)p(b|a)p(c|a)p(d|b,c)}.$$

The computational complexity of marginalization increases as the number of nodes, states, and links increases in the Bayesian network. In most inference cases, the probability mass function (or probability density function in the case of continuous random variables) at each node in the Bayesian network needs to be updated after the introduction of new evidence into the network. When the number of hidden nodes to be marginalized is large, the normalization factor, i.e., the denominator in (2), can be computationally intractable. Fortunately, there are several algorithms that can make marginalization tractable. For tree-structured networks, where each child node has only one parent, the marginalization or belief update at each node can be carried out exactly, i.e., exact inference, using a fast sum-product algorithm, a more generalized version of belief propagation and Pearl’s message passing [29]. The sum-product algorithm greatly reduces computation time and provides a standard programming framework to accomplish the marginalization over the entire network. It is worth mentioning that the method also works with any network in general; it gives exact inference for tree-structured networks and approximate inference for networks containing cycles, in which case it is known as loopy belief propagation [30]. In this work, we restrict the framework to tree-structured networks and use the sum-product algorithm for its computational advantages.

A. Deformable Bayesian Networks (DFBNs)

A dynamic tree (DT) graphical model [11], is a tree-structured Bayesian network whose joint probability $p(X, Y, Z)$ includes a random variable $Z$ that defines the structure of the network, which can be viewed as the linkage between nodes in the tree. To avoid the confusion with dynamic Bayesian networks\(^1\) and because of some extensions that differ from the traditional DT architecture, the name of this architecture is called deformable Bayesian networks (DFBNs). Let $Y = \{y_j\}_{j \in \mathcal{E}}$ denote the observed data nodes, where $\mathcal{E}$ is the set of indices of the observed nodes, and

\(^1\)To clarify further, dynamic Bayesian networks focus on the time-series aspect of Bayesian networks and have a fixed graph structure. Dynamic tree graphical models add flexibility to the graph structure by defining the linkage of the model as an additional random variable.
and let \( X = \{ x_j \}_{j \in H} \) denote hidden nodes, where \( H \) is the set of indices of the hidden nodes. The total number of nodes in the entire network will be \( N = |H| + |E| \). We set the index for each node in the network in ascending order from root to leaf. In other words, the index of a parent node is smaller than that of its children. A model with a fixed structure \( Z \) is defined by the joint distribution \( p(X, Y | Z, \theta) \). Since DFBN is a directed acyclic graph (DAG), the joint factorizes over the nodes, i.e., \( p(X, Y | Z, \theta) = \prod_j p(u_j | pa_j, \theta_j, Z) \), where \( u_j \in X \cup Y \), \( pa_j \) is the set of parents of \( u_j \), and \( \theta_j \in \theta \) parameterizes the edges (conditional probability) pointing toward the child node \( u_j \). The structure of the model is determined by the variable \( Z = \{ z_{ji} | j, i \in \{ H \cup E \} \} \), where \( z_{ji} = 1 \) when the child node \( u_j \) is connected to the parent \( u_i \), otherwise \( z_{ji} = 0 \). Normally, \( Z \) will be written in the form of a \( N \times N \) matrix whose element in \( j^{th} \) row and \( i^{th} \) column is \( z_{ji} \). All possible state configurations and all possible structures of the DFBN can be defined as a joint probability distribution whose functional form can be factorized as

\[
P(X, Y, Z | \theta, \pi) = P(X, Y | Z, \theta) P(Z | \pi),
\]

where \( \theta_j \) denotes the conditional probability of the arrow edge from \( u_i \) toward \( u_j \), and \( \pi_{ji} \) denotes the probability that the child node \( u_j \) is connected to the parent node \( u_i \). For convenience, both types of parameters can be collectively denoted as \( \theta = \{ \theta_{ji} \}_{j,i \in \{ H \cup E \}} \) and \( \pi = \{ \pi_{ji} \}_{j,i \in \{ H \cup E \}} \) respectively.

### B. Inference in the DFBN

The goal of a DFBN is to search for the optimal structure \( Z^* \) that can explain the evidence best, that is, the structure that maximizes the posterior marginal \( P(Z | Y, \theta, \pi) \) [11]. From this point on, it will be understood that the probabilities are given with the model parameters \( \theta \) and \( \pi \), but we will not write them explicitly. The posterior of the structure can be calculated from

\[
p(Z | Y) = \frac{\int p(X, Y, Z) dX}{\sum_Z \int p(X, Y, Z) dX}.
\]

The numerator can be calculated efficiently using the sum-product algorithm, but the denominator requires summation over all possible structures \( Z \) which might make the computation intractable when the number of nodes is large. There are some approximation techniques and heuristic optimization techniques to avoid the computational problem, for instance, simulated annealing and variational approximation, which will be discussed in detail subsequently. One way to avoid the calculation of the denominator is to realize that the denominator is effectively a constant depending only on the evidence \( Y \), and consider the maximum a posteriori (MAP) problem as an optimization problem where we do not have to explicitly evaluate the denominator

\[
Z^* = \arg \max_Z P(Z | Y),
\]

which is equivalent to

\[
Z^* = \arg \max_Z \{ \log P(Y | Z) + \log P(Z) \}.
\]

The likelihood of a structure \( Z \) given the evidence \( Y \) can be calculated from

\[
P(Y | Z) = \int P(X, Y | Z) dX.
\]
which can be evaluated exactly and efficiently using the sum-product algorithm since the model is a tree. In this section we adopt a relatively simple simulated annealing optimization framework to solve for the best structure. The details of this are found in Section IV.

Once the optimal structure $Z^*$ is determined, we can gather the information at the root node $x_r$ of each tree from its corresponding leaf nodes via probabilistic inference

\[ P(x_r|Y, Z^*) = \frac{P(x_r, Y|Z^*)}{P(Y|Z^*)}, \]  

\[ = \frac{\int P(X, Y|Z^*)dX_{\mathcal{H} \setminus \{r\}}}{\int P(X, Y|Z^*)dX}, \]  

(8)

(9)

where $r$ is the index of a root node, and $X_{\mathcal{H} \setminus \{r\}}$ denotes the set of all hidden variables except $x_r$. Note that the denominator in (9) is the same as in (7), and the numerator of (9) can be obtained one step before (7) using the sum-product algorithm as shown in Section IV-C1.

In this paper, we restrict the conditional probability between nodes to linear multivariate Gaussian. For a child node $x_k$ connected to the parent $x_j$, the conditional distribution is defined as $p(x_k|x_j) = \mathcal{N}(x_k; W_{kj}x_j + \mu_{kj}, \Lambda_{kj})$ where $W_{kj}$ is the loading matrix (weight matrix), $\mu_{kj}$ is the translation vector, and $\Lambda_{kj}$ is the precision matrix of the node $x_k$ with respect to the parent $x_j$. For more robust performance, the parameters sharing the same parent will be tied to have the same value, i.e., $W_{kj} = W_j$, $\mu_{kj} = \mu_j$ and $\Lambda_{kj} = \Lambda_j$.

C. DFBN for the Data Clustering and Fusion Algorithm

DFBNs have the ability to infer the information at each node in the form of a distribution using well-established statistical inference algorithms. The structure $Z$ of the DFBN can be deformed according to the evidence, indicating that model selection is intrinsically included in the DFBN. Therefore, the DFBN approach described here is appropriate to data clustering and fusion where the data have an underlying interpretable structure. In terms of the sensor fusion problem we propose in this paper, the MAP solution $Z^*$ yields a non-redundant set of target root nodes connected to the most likely sensing platform nodes, which are connected to the most likely set of measurements that originate from the particular sensing platforms. Hence, the most important part of translating a DFBN framework into the sensor fusion problem is to define the sensor-target relationships in terms of the structure variable $Z$ and the model parameters $\theta$ and $\pi$ that define the internodal relationships so that the DFBN solutions are within the realm of possibility, and more importantly, are the most likely answer of a priori constraints.

The next section will discuss the sensor fusion problem within the framework of DFBNs and detail the steps necessary to optimize the structure given sensor measurements on leaf nodes of each tree.

III. SENSOR FUSION FRAMEWORK USING DFBN

Using this framework, an algorithm was developed that takes the raw sensor measurements and sensor identification from an underwater sensing mission and automatically builds a tree for each unique target using a structural optimization algorithm combined with the sum-product algorithm. The algorithm outputs a forest or group of trees, each having an inferred probability distribution of the summarized target measurement at the root node. Each node
in the network is a random variable representing the feature. In this paper, we will use 3 features: 1) x-y target location, 2) corresponding seabottom texture parameters, and 3) target shape parameters. The details of the feature extraction algorithm will be discussed in the subsequent section. In this application, we also want to know the true or summarized location of an individual target which can be obtained from the root node of each tree. Thus, the DFBN sensor fusion framework yields two valuable pieces of information: 1) a target location (and other measurements) in terms of a probability distribution function, and 2) a parent-child relationship that links the target to the sensing platforms and raw measurements that were used to gather the information about its position.

In the sensor fusion DFBN framework, there are multiple levels of nodes and the number of nodes in each layer depends on the physical model of the problem. For example, in our scenario shown in Fig. 3, we have 3 levels:

1) **Leaf level**: Each node in the leaf level represents feature vectors extracted from measurements. This information will be combined in the upper level.

2) **Intermediate level**: Each node in the intermediate level represents the fused information collected from the level below. Furthermore, each node in this level provides the information about the platform or sensor taking the measurements, so this level is also called Sensor ID or Platform ID.

3) **Root level**: Each tree represents a single target, thus each root node represents the probability distribution of fused information for each target.

Let us assume that the measurement in the diagram is location, then each root node of the tree will be the summarized location of each target gathered from all the measurements at its corresponding leaf nodes. In fact, this framework can be seen as two simultaneous processes:

1) **Probabilistic clustering**: Each structure $Z$ can be interpreted as a result of a clustering process because it implies the parent-child relationship between two nodes from adjacent levels. The number of trees in the final structure indicates the number of actual targets. Having a set of child nodes under the same parent (or grandparent) implies that they are from the same physical target. For example, in Fig. 5, we can interpret that there are 10 measurements taken, but after all the redundancy is reduced by the DFBN, there are only 3 individual targets. Each tree has only one root node, and it represents a single target. So, each tree and its root node can be viewed as a resulting cluster and its corresponding centroid respectively.

2) **Bayesian data fusion**: Each root node serves as the centroid of the cluster, so it represents the fused information of the remaining members of the cluster, particularly, its corresponding children at the leaf level. Unlike other clustering algorithms, in this framework, the information is combined via hierarchical Bayesian probabilistic inference. Consequently, the fused information is in the form of a posterior probability which provides both estimation and uncertainty.
A. Reformulating the DFBN for Multiple Features

To improve clustering performance we extract several independent features from the data. From this point on we abuse notation by using $P(X, Y, Z|\theta, \pi)$ as the joint probability of all features $f \in \{1, \ldots, F\}$:

$$P(X, Y, Z|\theta, \pi) = P(Z|\pi) \prod_{f=1}^{F} P(X^{(f)}, Y^{(f)}|Z, \theta_f), \quad (10)$$

where $X^{(f)}$, $Y^{(f)}$ and $\theta_f$ are the hidden variables, observed variables and the edge parameters for the features $f$ respectively, and the optimization problem can be written as

$$Z^* = \arg\max_Z J(Z), \quad (11)$$

where the objective function $J(Z)$ is defined as:

$$J(Z) = \sum_f \log P(Y^{(f)}|Z, \theta_f, \pi) + \log P(Z|\pi). \quad (12)$$

Furthermore, each feature contributes differently to the clustering, so we want to assign a scalar weight $w_f$ to each of the features $f$. Finally, after modifying the objective function, we have our augmented objective function$^2$:

$$J(Z) = \sum_f w_f \log P(Y^{(f)}|Z, \theta_f, \pi) + \log P(Z|\pi) \quad (13)$$

The first term can be calculated efficiently using the sum-product algorithm [29], [31]. The second term is the structure probability, where we can incorporate other penalty functions like structure complexity or initial condition to the objective function. Note that the structure $Z$ is restricted to be the same for all the features. Fig. 6 is an illustration of the parallel DFBN structure and the three boxes from left to right are the three features: location, seabottom texture, and target shape. Note that more features can be added to the network as needed.

In the sensor fusion framework described here, we assume a uniform prior distribution on each root node. As mentioned earlier, the model parameters $\theta$ parameterize linear multivariate Gaussian relationships between each parent-child pair. Each parameter has physical meaning. For instance, the precision matrices of the edges between the root and intermediate nodes for Feature 1 are defined by the navigational uncertainty of the sensing platform.

$^2$This new objective function may not be a valid probability, but it has physical meaning. For instance, in 1D Gaussian we will have $w_f \log \mathcal{N}(x_f; \mu_f, \sigma_f^2) \approx \log \mathcal{N}(x_f; \mu_f, \sigma_f^2 w_f)$. 

Figure 5: (a) The initial structure for DFBN. (b) The optimal structure after structure optimization via simulated annealing. (c) The simplified structure is the structure when irrelevant nodes are removed.
Figure 6: Parallel DFBN sensor fusion structure. The objective function $J(Z)$ is a weighted sum of the DFBNs formed for 1) x-y target location, 2) corresponding seabottom texture parameters, and 3) target shape parameters. Ultimately, the Bayesian information criteria (BIC) is used as the objective function in order to penalize the structure complexity.

Given a set of measurements, the optimal graph structure is one in which the objective function $J(Z)$ in (13) is maximized. A general algorithm for finding the optimal graph structure is shown in Alg. 1.

IV. DFBN ALGORITHM IMPLEMENTATION

This section describes the detail of the DFBN implementation. First, the features and its extraction algorithm are explained. Then the graph structure initialization using vector quantization is introduced. Next, the efficient inference algorithm and the suggested choice of parameters setting for the objective function is addressed. At the end of this section, a stochastic-search optimization framework and heuristic structure update strategy are presented.

A. Feature Extraction

Features extracted from raw measurements occupy the leaf nodes of DFBN1, DFBN2, and DFBN3 and are ultimately used to resolve the structure of the tree. The reader should note that the DFBN framework is not restricted to type or the number of features to use. In this paper, we will use 3 fundamental features described as follows. In DFBN1, Feature 1, the geographic position of the target, is taken directly from the sensor measurement. In DFBN2,
Algorithm 1 Optimal Graph Structure

1: Input parameters:
   • structure parameters \( \pi \),
   • edge parameters (conditional probability) \( \theta \) and
   • weights \( w_1, w_2, w_3 \) from training process.

2: Input variables:
   • a set of \( N \) possible graph structures and
   • evidence on leaf nodes.

3: for \( i = 1 : N \) do
4:   Assign structure \( Z_i \).
5:   Calculate \( P(Z_i | \pi) \).
6:   Calculate the structure likelihood \( P(Y^{(f)} | Z_i, \theta_f, \pi) \), for feature \( f = 1, 2, 3 \) using sum-product algorithm.
7:   Calculate \( J(Z_i) \)
8: end for

9: Maximum value of \( J(Z) \) over all \( i \) gives globally optimal solution. The optimal structure \( Z^* \) is the structure that maximizes the objective function.

Feature 2, three sonar image seabottom texture parameters of correlation lengths in the x-and y-direction, and the K-distribution shape parameter, are extracted via the algorithm described in [32]. In DFBN3, the target shape is represented as an ellipsoid and the two radii are extracted as Feature 3 to describe the target shape characteristics. The 3 features extracted from the 27 samples are plotted separately in Fig. 7. Among three features, the scatter plot of Feature 1 seems to be the most separable. However, being very discriminative does not guarantee the correct clustering result, for instance, this can lead to over-segmenting the data. On the other hand, the other features might give information about which samples should be merged into the same cluster. Therefore, further information gathered from the other features is critically important for an accurate result.

For DFBN1, each of the target locations is assumed to be a multivariate Gaussian random variable with the mean centered about the measured geographic location and the covariance matrix determined by the navigational accuracy of the sensor. For DFBN2, the texture features are assumed to be drawn from a multivariate Gaussian distribution with the covariance matrix commensurate with the number of samples used to estimate the texture features [33]. For DFBN3, the target shape parameters are also assumed multivariate Gaussian distributed.

1) Seabottom Texture Extraction: The seabottom textures that surround a target of interest contain discriminatory information. These textures can be considered a context in which a given target occurs. Fig. 8 depicts a cylindrical target with the same orientation and range from the sensor placed in four different seabottom environments. While extracted target features from the cylinder should be the same for the four images, features based on the texture of the seabottom should be different. Seabottom texture features are extracted in the target fusion scheme to provide
Figure 7: The overview of the 3 features extracted from 27 sensor measurements. Samples with the same shape come from the identical target; here we have 5 actual targets. The same sample intensity indicates that they are obtained from the same sensor platform; in this case, there are 2 sensor platforms.

another set of features with different information than that provided by the geographic location and target features in the sonar image snippet.

Seabottom features are extracted from the sonar intensity image based on parameterized models of the autocorrelation function of the image pixels [34], [35]. The normalized ACF of a sonar intensity image is defined in spatial lag x- and y-directions denoted by $\tau_x$ and $\tau_y$ respectively as [32]

$$R_I(\tau_x, \tau_y) = \mu_I \left( 1 + [R_h(\tau_x, \tau_y)]^2 \right) + \frac{1}{\nu} \cdot \sum_i \eta_i \cdot |l_{x_i} l_{y_i}|^{1/2} \cdot$$

$$\left[ e^{-\frac{1}{2} [\tau_x \tau_y] \Phi_i^{-1} [\tau_x \tau_y]^T} + [R_h(\tau_x, \tau_y)]^2 \right]$$

where $\mu_I$ is the mean of the intensity image, $R_h(\tau_x, \tau_y)$, is the ACF of the imaging point spread function imposed by the beamformer or image formation algorithm, $\nu$ is the shape parameter of the single-point image statistics, $\eta_i$ is mixture parameter corresponding the spatial correlation parameters $l_{x_i}$ and $l_{y_i}$, and $\Phi_i$ is composed of a rotation
Figure 8: Four side-look sonar images of a cylinder in different seabottom environments.

matrix

$$\Psi_i = \begin{bmatrix} \cos \gamma_i & \sin \gamma_i \\ -\sin \gamma_i & \cos \gamma_i \end{bmatrix},$$  \hspace{1cm} (17)

a diagonal spatial correlation parameter matrix

$$\Xi_i = \begin{bmatrix} l^2_{x,i} & 0 \\ 0 & l^2_{y,i} \end{bmatrix},$$  \hspace{1cm} (18)

and a diagonal matrix of imaging point spread function parameters

$$B = \begin{bmatrix} \beta^2_{x} & 0 \\ 0 & \beta^2_{y} \end{bmatrix}$$  \hspace{1cm} (19)

via the following equation

$$\Phi_i = \Psi_i (\Xi_i + 2B) \Psi_i^T.$$  \hspace{1cm} (20)

The mixing parameter is constrained such that $$\sum_i \eta_i = 1.$$ In this application a single component or $$i = 1$$ is used in (16) for all features extracted.

The algorithm for estimating the parameters in (16) is explained in detail in [32]. We are interested primarily in three parameters to distinguish between different image seabottom texture: $$l_x$$ the spatial correlation length in the $$\tau_x$$ spatial lag direction, $$l_y$$ the spatial correlation length in the $$\tau_y$$ spatial lag direction, and $$\nu$$ the shape parameter of the single-point $$K$$-distributed probability density function [36]. The estimated $$\gamma$$ in the rotation matrix $$\Psi$$ also plays an important role in the feature extraction by reflecting the change in sensing angle between multiple looks.
at the same target. If the underlying texture is not angularly dependent on the sensor for the given sea bottom area, the set of texture features will be rotation invariant. There are instances where this assumption will not hold, e.g., a sensor angle boresight to the troughs in a sand ripple field [37], but for many seabottom types it is valid.

For this application, prior to seabed texture parameter estimation we removed a patch of the sonar image snippet containing the target and replaced it with a patch of seabed extracted above the target region. This simple two-dimensional “bootstrap” step is intended to give better estimates of the shape parameter and x and y correlation lengths in a seabed region by removing the highly correlated target and shadow pixels from the image snippet.

B. Graph Structure Initialization

In many situations, we can use the prior knowledge about the targets’ geographic location to facilitate the algorithm, and the sensor platform ID is known for each measurement, so we can use a divide-and-conquer strategy to reduce the computational time. This section describes the DFBN initialization and divide and conquer strategy.

1) Geographic Data Clustering via Vector Quantization: Unsupervised clustering via vector quantization, or more specifically Linde-Buzo-Gray vector quantization (LBG-VQ) [38], [39], is used to initially cluster the sensor data by geographic proximity. The geographic constraint is reasonable because it is not necessary to fuse targets that are too distant from one another assuming a reasonable accuracy in the sensor’s navigation system. The output of the algorithm yields subgroups of sensor measurements that are in close proximity to one another and reduces the possible structures of the DFBN. LBG-VQ is an iterative algorithm that meets two criteria: 1) the nearest neighbor condition, where each member of a cluster is assigned a centroid that is the closest centroid of the member, and 2) a distortion criteria, where the sum of the distance from all the points in a cluster to a valid centroid is less than some threshold value.

A sample problem is now introduced to illustrate the use of the LBG-VQ clustering algorithm in the DFBN initialization. Referring to Fig. 9, an initial map of targets are detected with sensors from two different sensing platforms and have locations defined by x and y. In Step 1 of the algorithm, the LBG-VQ algorithm clusters targets on the x-y input space with an output that assigns centroids (stars) to possible geographic centers of the targets. In Step 2, the targets are separated according to the assigned sensor platform ID. In this step, a DFBN forest is initialized with the centroids as the root nodes and the target locations as the leaf nodes. The leaf nodes are partitioned by centroid assignment. In Step 3, the centroids are linked to the corresponding leaf nodes to initialize the DFBN forest.

There are several alternatives to geographical data clustering algorithms, for instance, Gaussian mean shift, hierarchical clustering, k-mean, Gaussian mixture models and spectral clustering. However, keep in mind that the candidate algorithms are computationally inexpensive, do not over-segment the data, and most importantly, do not require a pre-defined number of clusters as this pre-processing step is unsupervised. From the algorithms mentioned above, the last three need cross-validation to find the number of clusters, and thus are not practical for our application here. Plus it is much simpler to pick a scalar threshold in LBG-VQ than choose the kernel size matrix (with possibly many degrees of freedom) in the Gaussian mean shift algorithm. In light of these considerations, LBG-VQ was
Figure 9: LBG-VQ clustering to initialize the DFBN forest. Targets are first clustered by their locations then connected into an initial tree based on centroid membership and sensing platform association.

chosen for initial data clustering due to its simplicity and ability to automatically determine cluster cardinality.

2) Divide and Conquer Strategy: Most of the sensing scenarios contain the vehicle-observation correspondence, however, there are some scenarios where such information can be missing or corrupted; and our framework is designed to incorporate such scenarios in general. When the sensor platform ID is known for each measurement, we can reduce the computational time of learning the structure by grouping and applying the data fusion framework to each group. With this simplifying assumption, we need only 2 layers instead of 3 at a time. The summarized procedure is illustrated in Fig. 10. First, we group measurements according to their sensor platform ID (pfm#) as in Fig. 10(b), and at this point we can bypass the platform ID level, so we will have only 2 levels: 1) measurement and 2) object/target level. Next, we apply our data fusion algorithm for each group separately and get the result which looks like Fig. 10(c). However, there might be some duplicated targets across the platforms. In other words, an individual target might be sensed by several platforms, so we will have to merge the result from the previous step again by letting the root nodes from the previous step become leaf nodes, and add the third level which will be the individual root. The same data fusion algorithm is run again, and the final result is shown in Fig. 10(d). The output in Fig. 10(d) is then used as the initial input to the DFBN algorithm greatly increasing the accuracy and speeding the computation of the final solution.
C. Probabilistic Inference for DFBN

This section describes the sum-product algorithm, the efficient inference algorithm using factor graph representation, which provides exact inference for DFBN. Furthermore, the choice of prior probability of the structure $Z$ and penalty for structure complexity are described subsequently.

1) Sum-product algorithm for DFBN: Given a fixed structure $Z$, we use a factor graph [40], [41] representation and sum-product algorithm to marginalize the hidden variables in DFBN. The sum-product rule is a generalized case of Pearl’s message passing algorithm and uses 2 types of nodes: 1) a variable node for each random variable and 2) a factor node for each local function, namely the conditional probability in this case. In the operation of the sum-product algorithm, for each tree, we first send the message from each evidential node upward to its corresponding factor node. Each of the factor nodes then sends the message upward to its corresponding parent variable node. The two processes are performed alternatively until the message reach the root of the tree. Next, the message is sent from the root to the leaf nodes in the similar manner but downward. Each message transmitted in the network represents the local knowledge or belief which is the probability that the nodes in the network attempt to share among one another in order to combine and get the global knowledge for the network, namely the posterior marginal at each node. Once the process is completed, each edge in the network contains two types of messages: upward and downward, and the posterior marginal distribution $p(x_j|Y)$ for a particular node $x_j$ given the evidences $Y$ can be carried out by multiplying all the incoming messages together.

The sum-product algorithm gives efficient and exact inference for DFBN because of its tree structure. The computational complexity is linear and polynomial in terms of the number of samples and the dimensionality of feature respectively. The complexity will be discussed again in Section V. More detail of the sum-product algorithm can be found in [40].
2) Structure Prior Probability $P(Z)$: There are many ways to come up with the prior probability of the structure. In this data fusion application, geographical location is one strong feature which we can exploit to define the prior of the structure. The Gaussian kernel is used to determine the associations among targets:

$$p(z_{ji} = 1) = \frac{\exp\left(-\frac{1}{2\sigma^2} \|r_j - r_i\|^2\right)}{\sum_{k \in V_{pa(j)}} \exp\left(-\frac{1}{2\sigma^2} \|r_j - r_k\|^2\right)}$$

(21)

where $r_j$ denotes the location vector (x-y coordinate) of the node $X_j$, $V_{pa(j)}$ is the index set of nodes in the same level as the parent of node $X_j$, and $\sigma^2$ is the variance, or the kernel width, of the prior structure. Large $\sigma^2$ suppresses the effect of the affinity between two targets, where the probability $p(z_{ji} = 1)$ tends to the uniform distribution over the parent $X_i$. In this experiment, we use a uniform prior distribution for the structure, that is, $p(z_{ji} = 1) = \frac{1}{|pa(j)|}$ where $|pa(j)|$ denotes the cardinality of the set of possible parents of $j$.

3) Penalties for structural complexity: A more complex structure tends to fit a broad range of models equally, so it is hard to distinguish the best model from the rest. Thus, it is usually the case that the structure complexity has to be regularized or penalized by some criteria so that the best model is sufficiently simple yet still able to explain the observations well. Bayesian information criterion (BIC) is a criterion for model selection among a class of parametric models with different numbers of parameters [42]. In this case, the parameters of interest are the dimensionality of the data, the number of edges and the number of trees from graph structure $Z$. The objective function derived from the BIC is given by:

$$BIC(Z) = -2 \log p(Y|Z) + K(Z) \log(#\text{sample})$$

(22)

where $K(Z)$ denotes the complexity of structure $Z$ which is

$$K_{2-\text{layer}}(Z) = #\text{sample} + #\text{root} \cdot \left(\frac{D^2}{2} + \frac{3D}{2}\right)$$

(23)

for a 2-layer structure and

$$K_{3-\text{layer}}(Z) = #\text{sample}$$

(24)

$$+ #\text{intermediate} \cdot \left(\frac{D^2}{2} + \frac{3D}{2} + 1\right)$$

(25)

$$+ #\text{root} \cdot \left(\frac{D^2}{2} + \frac{3D}{2}\right)$$

(26)

for a 3-layer structure respectively, where $D$ is the feature dimensionality, $#\text{sample}$ is the number of leaf nodes, $#\text{intermediate}$ is the number of nodes in the middle layer, and $#\text{root}$ is the number of root nodes. The new objective function, $BIC(Z)$, is small when the structure $Z$ gives high log-likelihood and has an uncomplicated structure. Thus the optimal structure $Z^*$ is obtained when $BIC(Z)$ is minimized.

D. DFBN Optimization Scheme

This section describes the exhaustive search algorithm which allows all possible structures to be evaluated, then simulated annealing, a stochastic-search optimization framework for DFBN, which provides better run-time than
using an exhaustive search in general is elaborated. Also, we propose a heuristic method which significantly speeds up the simulated annealing algorithm when there are a large number of samples.

1) Exhaustive Search: It can be shown using Bell’s number [43] that solving for the globally optimal structure of the DFBN using exhaustive search algorithm requires a search over an extremely large number of possible linkage combinations for a relatively small number of input nodes and is computationally infeasible. For example, with only 10 measurements from a single sensor, the possible number of 2-layer DFBN representations/structures is 115,975. By enforcing some a priori constraints on the problem, such as a geographic threshold mentioned earlier in this section for possible target combinations, and initializing via unsupervised clustering, the number of possible DFBN solutions is reduced, and thus, the search over the possible structures is less time-consuming. In this paper, we recommend using an exhaustive search strategy when the number of samples is less than 9, but in the case of having 9 or more, stochastic-search optimization, which will be explained subsequently, is recommended.

Table I: Number of possible structures of the 2-layer DFBN vs. number of samples. It can be shown that the number of possible structures of the 2-layer DFBN is represented by Bell’s number.

<table>
<thead>
<tr>
<th># of samples</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td># of possible structures</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>15</td>
<td>52</td>
<td>203</td>
<td>877</td>
<td>4140</td>
<td>21147</td>
<td>115975</td>
</tr>
</tbody>
</table>

2) DFBN Optimization via Simulated Annealing: Simulated annealing is an alternative optimization scheme to exhaustive search, which can drastically speed up the algorithm when there are a large number of samples. After the DFBN forest is initialized by the LBG-VQ clustering algorithm, the optimum structure is calculated using simulated annealing. Simulated annealing (SA) is a stochastic search algorithm used to find maxima in large optimization problems and draws its name from the method of allowing magnetic systems to find low-energy structures through annealing [44], [45]. In the DFBN sensor fusion framework, parent-child connections are changed at random and accepted if the new structure either results in a higher log posterior or randomly exceeds a threshold governed by the decreasing “temperature” of the annealing process. The stochastic nature of SA allows for “unfavorable” structures which have low BIC($Z$) values frequently in early iterations of the algorithm in hopes of avoiding local maxima. At later iterations, the algorithm performs like a stochastic greedy search and only accepts new structures that increase the MAP value. The structure acceptance is governed by a temperature parameter that decreases as the number of iterations increases.

Following the outline of the DFBN optimization algorithm described in Section III, a summary of the DFBN sensor fusion is presented in Alg. 2. The algorithm requires inputs of the feature vectors extracted from the sensor measurements, conditional probability distribution at each edge, an initial structure, an annealing schedule with an initial temperature and a structure update procedure. The random structure is a simple methodology to update the structure in SA by randomly picking a child node and randomly changing its corresponding parent as presented in Alg. 3.
Algorithm 2 Summary of the DFBN optimization algorithm

1: Determining the following user-defined parameters according to Tab. II

\[ I_{max} \] Maximum number of iterations allowed for the whole process
\[ T_0 \] Initial temperature
\[ \alpha \] Temperature decay constant, \( 0 < \alpha < 1 \)
\[ T_{limit} \] Limiting temperature
\[ P_{max} \] Maximum number of proposed structure changes allowed before changing temperature
\[ A_{max} \] Maximum number of accepted structure changes allowed before changing temperature

2: Initializing parameters

- Reset proposed structure changes counter \( P \leftarrow 0 \)
- Reset accepted structure changes counter \( A \leftarrow 0 \)
- Initialize temperature \( T \leftarrow T_0 \)

3: \( Z \leftarrow \) Initialize parallel DFBN structure via LBG-VQ procedure in Section IV-B1

4: Set the best structure \( Z^* \leftarrow Z \)

5: Calculate \( -\text{BIC}(Z) \) using sum-product algorithm

6: Set \( E_a \leftarrow -\text{BIC}(Z) \) and

7: Set the best score \( E^* \leftarrow E_a \)

8: repeat

9: \( Z \leftarrow \) structure update algorithm \( \triangleright \) e.g., random structure as in Alg. 3 or directed perturbation as in Alg. 4

10: Recalculate \( -\text{BIC}(Z) \)

11: Set \( E_b \leftarrow -\text{BIC}(Z) \)

12: if \( \exp \left( \frac{E_b - E_a}{kT} \right) \geq \text{Uniform_random}[0,1] \) then

13: \( E_a \leftarrow E_b \)

14: \( A \leftarrow A + 1 \)

15: else if \( E_b \geq E^* \) then

16: \( E^* \leftarrow E_b \)

17: \( Z^* \leftarrow Z \)

18: end if

19: \( P \leftarrow P + 1 \)

20: if \( A > A_{max} \) or \( P > P_{max} \) then

21: \( T \leftarrow \alpha T \)

22: \( A \leftarrow 0 \)

23: \( P \leftarrow 0 \)

24: end if

25: until \( T < T_{limit} \) or \( I_t > I_{t_{max}} \)

26: Return \( Z^* \) \( \triangleright \) The optimal structure
3) **Structure Update using Directed Perturbation:** The DFBN can be accelerated significantly when a series of appropriate structures is proposed to SA. Inspired by Gibbs sampling, we proposed an intuitive stochastic methodology, called directed perturbation, to update the structure in SA optimization scheme. First, a random child node is selected, then the affinity measure is carried out between each pair of the child node and all of the parent candidates in the upper level. A multinominal distribution is then formed to represent the probabilities that the child node would connect to each of the parent candidates. The successful parent is sampled from the distribution which is updated every time the structural changes have been made. This heuristic approach enables faster convergence than using the random structure. Empirically, it is sufficient to use 100 iterations in SA with directed perturbation as opposed to 1000 iterations in SA with random structure update in order to get a good result in the scenario with about 20 samples. The algorithm is summarized in Alg. 4.

**Algorithm 3** Random structure approach for graph structure update

1: **procedure** RANDOMSTRUCTURE(structure \(Z\))
2: randomly pick a child node \(j\) in structure \(Z\)
3: list all parent candidates \(A\) of the child node \(j\) excluding its current parent
4: randomly pick a parent node \(i^*\) from \(A\)
5: remove the connection between \(j\) and its current parent
6: make a connection between \(j\) and \(i^*\) in the new structure \(Z_{new}\)
7: Return \(Z_{new}\)
8: **end procedure**

**Algorithm 4** Directed perturbation for graph structure update

1: **procedure** DIRECTEDPERTURBATION(structure \(Z\))
2: randomly pick a child node \(j\) in structure \(Z\)
3: list all parent candidates \(A\) of the child node \(j\) excluding its current parent
4: for \(i \in A\) do
5: calculate the likelihood \(A_j(i)\) that the child \(j\) and the parent \(i\) are connected using (21)
6: end for
7: [optional] re-normalize \(A_j(i)\) over \(i\) to ensure the multinominal distribution
8: sample the successful parent \(i^*\) from \(A_j(i)\)
9: remove the connection between \(j\) and its current parent
10: make a connection between \(j\) and \(i^*\) in the new structure \(Z_{new}\)
11: Return \(Z_{new}\)
12: **end procedure**
V. EXPERIMENT: UNDERWATER SENSING SCENARIO

The DFBN algorithm’s ability to automate the sensor fusion task was demonstrated against a side-looking sonar dataset comprising 27 different sonar images with associated location measurements of 5 targets sensed multiple times via different sensor tracks. The objective of the experiment is to correctly categorize the 27 images into the 5 target classes. The sonar images were created by a high-resolution sonar stripmap imaging system integrated onto an unmanned underwater vehicle. The multiple images of each target were the result of overlapping sensing tracks where the target of interest was placed in the sensor field of view at various orientations and ranges.

Given the sonar image dataset described in the previous paragraph, the parameters needed to initialize the DFBN are: 1) the structure prior, 2) the contribution weight for each feature, and 3) the linear Gaussian parameters. In these experiments, all the parameters are learned from a training dataset of 15 images from the survey dataset, where the ground truth labels for all images are already known. For each experiment, the parallel DFBN structures were initialized using the worst case “pillar” structure and the structure prior probability is assumed uniformly distributed, i.e., $p(Z_i) = p(Z)$, thus simplifying the calculation of the objective function. The weights for 3 features are equal, i.e., each weight is $1/3$, each based on results from a previously simulated scenario as a training example.

For the linear Gaussian parameters we assume identity weights matrix $W_j^{(f)} = I$, zero translation $\mu_j^{(f)} = 0$ and full precision matrix $\Lambda_j^{(f)}$ for every node and every feature. The precision matrices can be estimated from the sonar images during the training process. The sonar image texture features of x correlation length $l_x$ and y correlation length $l_y$, were estimated from the sonar images using the procedure described in [32].

Since we have geographical clustering as a preprocess, the number of samples is reduced significantly within a group as shown in Fig. 11. The computational cost of DFBN using the exhaustive search, denoted by DFBN+All depends on the number of possible structures created from the nodes within a group. The Tab. I shows the number of possible structures of a 2-layer DFBN when the number of nodes is increased. In the case where the number of samples is less than 9, we can perform an exhaustive search efficiently over the structure domain. When the number of samples is greater or equal to 9, SA can perform faster and provide competitively good results. In addition to DFBN+All, we experiment on two types of structure update algorithms implemented for SA: 1) random structure update and 2) directed perturbation, and hence denoted by DFBN+Rand and DFBN+DP respectively. The parameters for the SA optimization algorithm of both approaches are given in Tab. II.

In order to compare the performance of DFBN, we use Gaussian mean shift (GMS) and spectral clustering [46] as baseline methods since they have been adopted in many data clustering applications. Like DFBN, GMS requires only the covariance matrix of each feature as an input, and more importantly, it does not require any information about the number of clusters. On the other hand, spectral clustering requires such information, therefore, we use [47], an information-theoretic-based procedure, to determine the optimal number of clusters, which is less convenient than DFBN where the number of cluster is determined automatically in the process. The optimal DFBN graph structures obtained from the data in Fig. 7 are illustrated in Fig. 12. The empirical results obtained from using DFBN+All, DFBN+Rand and DFBN+DP are identical, and hence are depicted identically in Fig. 11(c), provides
Table II: Parameters used in SA optimization. The second and the third column represent the parameters used in the DFBN with the random initial structure approach and directed perturbation approach respectively.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>DFBN+Rand</th>
<th>DFBN+DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_0$</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$T_{limit}$</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>$k$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$A_{max}$</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$P_{max}$</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>$I_{max}$</td>
<td>1000</td>
<td>100</td>
</tr>
</tbody>
</table>

Table III: Estimated relative x-y locations and uncertainty using DFBN for each detected target according to the results shown in Fig. 11.

<table>
<thead>
<tr>
<th>detected target#</th>
<th>group#</th>
<th>root node#</th>
<th>estimated locations</th>
<th>locational uncertainties</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>(38.99, 3.84)</td>
<td>3.95 1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.25 1.26</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>(0, 0)</td>
<td>2.51 0.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.77 0.78</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>(16.00, 2.24)</td>
<td>3.81 1.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.18 1.19</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>(67.67, 57.16)</td>
<td>3.95 1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.25 1.26</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>(80.82, 72.25)</td>
<td>3.91 1.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.23 1.24</td>
</tr>
</tbody>
</table>

some important information to us. First, SA converges to the global optimal solution within the given number of iterations. Second, the directed perturbation strategy can supply appropriate structures to SA, which in turn permits the DFBN to converge to the global optimal solution in a shorter time. The clustering results of using all the approaches mentioned earlier are shown in Fig. 11. Each tree structure in Fig. 12 can be interpreted as clustering results, and each root node distribution captures the estimated location of the actual target depicted as red dots in Fig. 11(c). The estimated locations and uncertainties obtained from DFBN (as described in Section. II-B) are listed in Tab. III. The run-time is compared in Tab. IV. All the experiments are run in MATLAB® r2010a on an Intel® Core™2 Duo CPU E8400 @ 3.00GHz machine with Linux Ubuntu operating system.
Figure 11: Comparing the clustering results (feature#1 only). The dashed boxes represent groups of x-y locations separated by geographic data clustering. Shaded ellipses represent ground truth cluster; the feature vectors (points) under the same shaded ellipse are from the same actual target. The unfilled closed contours represent resulting clusters obtained from each method.
Figure 12: The best graph structure obtained at the final stage of DFBN. From left to right are Group1, 2 and 3 respectively. The graph structures are interpreted as the clusters in Fig. 11

Table IV: Computational time for GMS, spectral clustering (SPC), DFBN with exhaustive search, random initial structure, and directed perturbation. Note that when the number of samples per group is less than 9, only DFBN+All is implemented due to the efficiency, hence n/a (not available) appears in some entries of the table.

<table>
<thead>
<tr>
<th>group#</th>
<th>#sample</th>
<th>run-time for each algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>GMS</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1 sec</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>1 sec</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>1 sec</td>
</tr>
</tbody>
</table>

The empirical results show that DFBN outperforms GMS with the same information of parameters, i.e., covariance matrices in the Gaussian kernels. In this case, the number of samples are sparse, and GMS tends to overestimate the number of clusters despite using the parameters from the training data. Similar drawbacks are also found in spectral clustering where the optimal kernel size and the optimal number of clusters are learned from the training data and information-theoretic approach.

On the other hand, DFBN performs better because it considers a cluster as a structure which generally contains more information than the notion of membership in traditional clustering frameworks. Structures within the DFBN contain information that is smoothly combined both locally (bottom-up) and globally (top-down) via the sum-product algorithm. Moreover, the Bayesian framework assists with stabilizing DFBN in case of sparse data whereas others do not, and therefore, the DFBN does not require a large number of samples.

The deformable structure of DFBN not only adjusts its structure to fit with the given data, but also searches clustering configurations in graph-structure space which cannot be done in traditional distribution-based clustering algorithms. This behavior is similar to model selection.
Despite the fact that the linear multivariate Gaussian model is used in this paper, there is no limitation on the conditional probability model used in a DFBN in general. For instance, the linear Gaussian DFBN would over-segment clusters whose actual uncertainties are from a Gaussian mixture model, so it might be more appropriate to model the conditional probability distribution of the DFBN with Gaussian mixture model instead. This flexibility empowers DFBN to be versatile to a broad array of applications.

Although DFBN outperforms both GMS and spectral clustering, its run-time is sensitive to the choice of structure update strategy, and inappropriate choice can result in a poor clustering result or inefficient run-time. The cost of an inference for a structure using the sum-product algorithm is $O(|\mathcal{E}|D^2)$ (recall $|\mathcal{E}|$ and $D$ denote the number of samples in the data and the dimensionality of the feature respectively). The exhaustive search depends on the number of possible structure bounded by $O\left(|\mathcal{E}|^{\log(|\mathcal{E}|+1)}\right)$ [48]. When the number of sample is small, DFBN+All, whose complexity is $O(|\mathcal{E}|D^2 \left[\frac{|\mathcal{E}|}{\log(|\mathcal{E}|+1)}\right]^{|\mathcal{E}|})$, can be still carried out, however, that might not be the case when the number of sample is large. The first effort to overcome the problem is to use DFBN+Rand, whose complexity is $O(|\mathcal{E}|D^2)I_{nax}^{(Rand)}$ where $I_{nax}^{(Rand)}$ denotes the maximum number of iterations for SA using random structure, resulting in better run-time in group2 as shown in Tab. IV. However, a shortcoming of DFBN+Rand is the randomness of the proposed structures which requires large number of iterations to converge to a good solution. Such a hindrance is overcome by having more control over the proposed structures resulting in DFBN+DP mentioned earlier in previous section. DFBN+DP, whose complexity is $O(|\mathcal{E}|D^2)I_{nax}^{(DP)}$ where $I_{nax}^{(DP)}$ denotes the maximum number of iterations for SA using directed perturbation, mitigates the problem by providing appropriate structures to SA, thus, in general, requires significantly less number of iterations to converge, i.e., $I_{nax}^{(DP)} << I_{nax}^{(Rand)}$. There are many more possible strategies to accelerate DFBN, for instance, greedy structure update, Gibbs sampling, variational approximation inference on possible structures. These issues are still open for future research.

VI. EXPERIMENT ON ONRUF1 DATASET

In this experiment we evaluate the proposed framework on a simulated dataset where the location of each target is known precisely. ONRUF1, a simulated dataset, comprises 5 geographical groups of targets, each group containing 3 targets (which may or may not be of the same target type), thus 15 distinct targets overall. The details of each group are as follows:

- Group1 – three cylinders
- Group2 – three cones
- Group3 – three blocks
- Group4 – one cylinder, one cone and one block
- Group5 – one cylinder, one cone and one block

These groups of targets are placed on a seabottom sand background. For each target there are 24 observations due to the 24 different vehicle tracks, resulting in 72 samples for each group and 360 in total. The same set of features are extracted from each data sample whose x-y location is shown in Fig. 13. In this experiment, we evaluate the performance of DFBN+DP, best from the previous experiment, on ONRUF1 dataset. The implementation is
slightly changed from the previous experiment because we implement the entire framework without relying on the Bayesian Network toolbox (BNT) [49] for inference as opposed to the previous experiment, resulting in a significant improvement in the run-time as 360 samples can be processed in less than 1 minute. All the experiment parameters remain the same as that of the previous experiment.

Due to the size of the dataset, the topology of the partition cannot be properly represented using graphics, and therefore a similarity measure is then needed to evaluate performance. Since this is an unsupervised clustering framework, it does not make much sense to use a confusion matrix to evaluate the clustering performance because a confusion matrix requires a predefined output class, which is not the case in our application. Furthermore, in an unsupervised clustering framework, the number of resulting clusters is not necessarily the same as that of the ground truth clusters. Therefore, we adopt Adjusted Rand Index (ARI) [50], [51] as our measure of performance, which is a measure of the consistency between two data partitions, namely the resulting partition and the ground truth partition, by counting the fraction of pairs of data samples that are consistent over those that are not. The value of an ARI ranges between $[0, 1]$, where higher is better.

The clustering results for each group are illustrated in Fig. 14. The ground truth and the predicted target locations are represented by the black stars and black circles respectively. The uncertainty, captured by the covariance matrix, associated with each location is magnified by 3 (for visual purpose) before being plotted as an ellipse. In each data group, the predicted label for each sample is represented by a solid shape (e.g., square, upper-triangle, lower-triangle, left-triangle), whereas the corresponding ground truth label is represented by the surrounding shape. A sample represented by the same inner and outer shape implies correct labeling, and otherwise implies mislabeling. Tab. V demonstrates the performance of DFBN on the ONRUF1 dataset. The number of clusters (denoted by $N_{detected}$ in the second row) in each group is correctly predicted by the proposed framework except in group3 where such a number is overestimated as 4 as opposed to 3. The ARI calculated from group1 to group5 are shown in the third row of Tab. V. In group1, the simplest scenario among all five, the clustering can recover the ground truth clusters perfectly, resulting in unity ARI value. In group2, 4 and 5, a few samples are mismatched, hence the ARI values are smaller than unity. However, in group3 there are mismatches in both clustering and the number of clusters, and therefore resulting in the low ARI value. In the 4th and 5th rows, $loc_{true}$ and $loc_{out}$ represent the true and estimated location of each target respectively, more specifically each line of which representing the (x,y) coordinate of each target. The averaged locational error measured between each corresponding pair of true and estimated location of each group, denoted by $error_{avg}$ and illustrated in the 6th row, indicates the error is less than 1m in every group except in group3, where the most ambiguous scenario occurs in the feature space.

Finally, the estimated covariance matrix for each estimated location, denoted by $cov_{out}$, is shown in the last row of the table. As mentioned before, these covariance matrices are then multiplied by 3 before plotting in Fig. 14 for demonstrative purpose. Due to the complexity of the sensing scenario in the dataset, the true uncertainty of each target cannot be summarized in terms of covariance matrix or any known parametric distribution, and hence it does not make much sense to show such measures in Tab. V. However, our proposed model assumes multivariate Gaussianity for each feature, and therefore it is valid to represent the uncertainty using the covariance matrix.
background combinations on the dataset ONRUF2700 which contains 2700 different scenarios. Like ONRUF1, this experiment aims to evaluate the performance of DFBN+DP in a multitude of target scenarios and seabottom background combinations on the dataset ONRUF2700 which contains 2700 different scenarios. Like ONRUF1,

![Figure 13: The dataset ONRUF1. Each geographical group contains 3 targets, whose ground truth locations are represented by the stars, and each observation is illustrated by a circle.](image)

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N_{\text{detected}})</td>
<td>3/3</td>
<td>3/3</td>
<td>4/3</td>
<td>3/3</td>
<td>3/3</td>
</tr>
<tr>
<td>ARI</td>
<td>1.00</td>
<td>0.92</td>
<td>0.63</td>
<td>0.84</td>
<td>0.80</td>
</tr>
<tr>
<td>loc(_{\text{true}})</td>
<td>(-80.00, 11.00)</td>
<td>(50.00, 0.00)</td>
<td>(-20.00, -54.00)</td>
<td>(40.00, -68.00)</td>
<td>(40.00, 60.00)</td>
</tr>
<tr>
<td>loc(_{\text{out}})</td>
<td>(-68.00, 5.00)</td>
<td>(58.00, 5.00)</td>
<td>(-15.00, -46.00)</td>
<td>(40.00, -60.00)</td>
<td>(44.00, 68.00)</td>
</tr>
<tr>
<td>loc(_{\text{out}})</td>
<td>(-50.00, 0.00)</td>
<td>(70.00, -5.00)</td>
<td>(-12.00, -50.00)</td>
<td>(52.00, -68.00)</td>
<td>(52.00, 64.00)</td>
</tr>
<tr>
<td>(\text{error}_{\text{avg}})</td>
<td>0.44</td>
<td>0.66</td>
<td>3.70</td>
<td>0.88</td>
<td>0.87</td>
</tr>
<tr>
<td>(\text{cov}_{\text{out}})</td>
<td>\begin{pmatrix} 0.21 &amp; -0.01 \ -0.01 &amp; 0.11 \end{pmatrix} &amp; \begin{pmatrix} 0.13 &amp; -0.03 \ -0.03 &amp; 0.14 \end{pmatrix} &amp; \begin{pmatrix} 0.10 &amp; 0.18 \ 0.16 &amp; 0.08 \end{pmatrix} &amp; \begin{pmatrix} 0.13 &amp; 0.07 \ 0.07 &amp; 0.11 \end{pmatrix} &amp; \begin{pmatrix} 0.24 &amp; -0.03 \ -0.03 &amp; 0.16 \end{pmatrix}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 14: (Best viewed in color) The clustering results on the ONRUF1 dataset. The ground truth and the predicted target locations are represented by the black stars and black circles respectively. A sample represented by the same inner and outer shape implies correct labeling, and otherwise implies mislabeling.

The ONRUF2700 dataset contains 5 groups of 3 targets in the same way listed in ONRUF1. On the other hand, ONRUF2700 has 3 types of background: seagrass (G), rock (R) and sand (S), each of which has 2 realizations generated using similar sets of parameters, and thus we have G1, G2, R1, R2, S1 and S2. When accounting for the possible ways such background patterns can be put into the 3 targets in each group, there are over a hundred possible scenarios. We focus our attention on a subset of them by carefully selecting 18 different seabottom background combinations for 3 targets as illustrated in Tab. VI. Combinations 1-6 present the same type of background to all 3 targets in the group, and henceforth is referred to as same-background. Combinations 7-12 present the same type but different realization of seabottom background to each target, and therefore is called similar-background. The rest of the combinations present different background to each target, so they are referred to as different-background.
Table VI: All 18 combinations of seabottom background. The 3 sub-columns indicate that the corresponding background occurs under the target1, 2 and 3 in each group respectively.

<table>
<thead>
<tr>
<th>Same-background (comb. 1-6)</th>
<th>Similar-background (comb. 7-12)</th>
<th>Different-background (comb. 13-18)</th>
</tr>
</thead>
<tbody>
<tr>
<td>target1</td>
<td>target2</td>
<td>target3</td>
</tr>
<tr>
<td>G1</td>
<td>G1</td>
<td>G1</td>
</tr>
<tr>
<td>G2</td>
<td>G2</td>
<td>G2</td>
</tr>
<tr>
<td>R1</td>
<td>R1</td>
<td>R1</td>
</tr>
<tr>
<td>R2</td>
<td>R2</td>
<td>R2</td>
</tr>
<tr>
<td>S1</td>
<td>S1</td>
<td>S1</td>
</tr>
<tr>
<td>S2</td>
<td>S2</td>
<td>S2</td>
</tr>
</tbody>
</table>

Furthermore, there are 6 possible ways to permute 3 targets within the same group, which creates 6 possibilities for each group and totally $6 \times 5 = 30$ possibilities for all 5 groups, resulting in $30 \times 18 = 540$ scenarios. In order to eliminate the location variability of the dataset, we placed each group on all five group locations, meaning objects of group 1 were placed on those of groups 2, 3, 4 and 5, so, location does not bias the DFBN solution. Thus we will have $540 \times 5 = 2700$ scenarios in total.

We evaluate DFBN+DP in all the 2700 scenarios using the same setting used in the previous experiment, so the run-time of this experiment is about the same as before. Instead of showing the results case by case, we hereby summarize the performance by computing the averaged ARI over all the cases, resulting in $ARI_{overall} = 0.77$. The results also show the DFBN+DP estimates correctly the number of targets in 1812 scenarios, and overestimates the number of targets to 4, 5, 6 and 7 in 549, 258, 59 and 22 scenarios respectively. It is interesting that the underestimate case has not occurred in the experiment, which suggests that the objective function of DFBN+DP slightly favors more complex structures than simpler ones. This is still an open problem for future work on the objective function.

**VIII. Conclusion**

In many survey-based sensing tasks, multiple measurements are taken of the same target of interest over a period of time. At the end of these surveys, the redundancy in the information must be removed to declutter data and information displays to the end-user. The DFBN architecture yields a hierarchical analytical framework to automate the task of combining similar targets that have discriminating measurements. Measured geographic locations and features extracted from both the target and the corresponding seabottom texture were used to fuse the redundantly sensed targets via optimization of the DFBN structure or nodal linkage. The DFBN framework was compared against GMS and spectral clustering algorithm and demonstrated significant improvement over these algorithms. The resulting tree structure obtained from DFBN is a rich representation of clusters which shows more details of interrelationship among the data samples than traditional clustering methods such as k-mean and GMM.

The DFBN’s run-time is sensitive to the choice of structure update strategies, and improper choice of such strategies can result in undesired clustering result or inefficient run time. Several update strategies were proposed
that worked well in the various data scenarios. DFBN+All, a modification which evaluates all possible structures, is suitable when the number of samples is small. DFBN+Rand, a simulated annealing optimization regime with random structure update strategy which encourages globally optimal solutions, is suited for larger optimization problems. A third strategy, DFBN+DP, uses directed perturbation to greatly improve the run-time over DFBN+Rand while attaining the desired solution.

The DFBN+DP was evaluated on the simulated dataset ONRUF1 and ONRUF2700, where each scenario contains 360 observations. The former dataset is a single scenario where the seabottom backgrounds are the same for all targets. The estimated number of the targets can be recovered correctly in all but one target, in addition, the location of targets have been estimated with an error less than 1m. The Adjusted Rand Index is used to measure the consistency between the resulting and the ground truth partitions. We further tested DFBN+DP on a multitude of scenarios in the dataset ONRUF2700, resulting in satisfactory results. With a computational cost $O(|E|D^2 I_{max})$, the DFBN+DP takes less than 1 minute per each scenario containing 360 observations, and hence has potential use with large data sets and is an avenue of future research.

REFERENCES


Kittipat Kampa received the B.S. degree in electrical engineering from Chulalongkorn University in 2001 and the M.S. and Ph.D. degrees in electrical and computer engineering from University of Florida in 2006 and 2011, respectively. From 2001 to 2003 he worked as an embedded system design engineer in Integrated-circuit Design and Applications Research (IDAR), Bangkok, Thailand. From 2003 to 2009 he worked at Adaptive Signal Processing Laboratory and National Center for Airborne Laser Mapping (NCALM), where his research focused on signal processing and machine learning with applications to remote sensing data, specifically LiDAR 3D point cloud. Since 2009 he has been a member of Computational NeuroEngineering Laboratory (CNEL), where his research interests include machine learning, information theoretic learning, and probabilistic graphical models, with applications to computer vision, data clustering and data fusion.

Erion Hasanbelliu received a B.S. and M.S. degree in computer science from Jacksonville State University in 2002 and 2004, respectively. He also received a M.S. degree in electrical and computer engineering (ECE) from University of Florida in 2008. Currently, he is pursuing a Ph.D. degree in ECE from UF, and working at the Computational NeuroEngineering Laboratory. His research interests include signal processing, machine learning, information theoretic learning, and pattern recognition.

J. Tory Cobb (S’99–M’01) received the B.S. degree in electrical engineering from the United States Coast Guard Academy in 1994 and the M.S. degree in electrical engineering from Auburn University in 2001. From 1994 to 1999 he was an active duty officer in the Coast Guard. Since 2001 he has been employed as a Research Engineer at the Naval Surface Warfare Center, Panama City, Florida. He has served as Principal Investigator or Co-Principal Investigator for various automatic target recognition and sensor fusion projects funded by the Office of Naval Research. Concurrently, he is pursuing the Ph.D. degree in electrical and computer engineering at the University of Florida. His research interests include statistical modeling of sonar signals with applications to automatic target recognition.
**Jose C. Principe** is Distinguished Professor of Electrical and Biomedical Engineering at the University of Florida, Gainesville, where he teaches advanced signal processing and machine learning. He is BellSouth Professor and Founding Director of the University of Florida Computational Neuro-Engineering Laboratory (CNEL). His research interests are centered in advanced signal processing and machine learning, Brain Machine Interfaces and the modeling and applications of cognitive systems. He has authored 5 books and more than 200 publications in refereed journals, and over 380 conference papers. He has directed 65 Ph.D. dissertations and 67 Master’s theses.

**K. Clint Slatton** (S'95–M'02–SM'07) received the B.S. and M.S. degrees in aerospace engineering and the M.S. and Ph.D. degrees in electrical engineering from the University of Texas (UT), Austin in 1993, 1997, 1999, and 2001, respectively.

From 2002 to 2003, he was a Postdoctoral Fellow with the Center for Space Research, UT, where he worked on multiscale data fusion techniques for interferometric synthetic aperture radar (InSAR) and airborne laser swath mapping (ALSM) measurements. He was also with the Radar Sciences Section, NASA Jet Propulsion Laboratory. He is currently an Associate Professor with the Department of Electrical and Computer Engineering and the Department of Civil and Coastal Engineering, University of Florida, Gainesville, where he is the Director of the Adaptive Signal Processing Laboratory. His research interests include remote sensing applications of sonar, ALSM, and InSAR as well as high-dimensional data segmentation, multiscale data fusion, graphical models, and photonics, with sponsored projects from the National Science Foundation, the U.S. Army, the U.S. Navy, and National Oceanic and Atmospheric Administration.

Dr. Slatton is a member of the American Geophysical Union, SPIE, Sigma Xi, Tau Beta Pi, and Sigma Gamma Tau. He was a 2006 winner of the Presidential Early Career Award for Scientists and Engineers for work on predicting signal propagation in highly cluttered environments using remotely sensed geometry from ALSM. He was a recipient of the NASA Graduate Student Research Program Fellowship.