DOI:10.1145/1831407.1831431

Nonparametric Belief Propagation

By Erik B. Sudderth, Alexander T. Ihler, Michael Isard, William T. Freeman, and Alan S. Willsky

Abstract

Continuous quantities are ubiquitous in models of real-world phenomena, but are surprisingly difficult to reason about automatically. Probabilistic graphical models such as Bayesian networks and Markov random fields, and algorithms for approximate inference such as belief propagation (BP), have proven to be powerful tools in a wide range of applications in statistics and artificial intelligence. However, applying these methods to models with continuous variables remains a challenging task. In this work we describe an extension of BP to continuous variable models, generalizing particle filtering, and Gaussian mixture filtering techniques for time series to more complex models. We illustrate the power of the resulting nonparametric BP algorithm via two applications: kinematic tracking of visual motion and distributed localization in sensor networks.

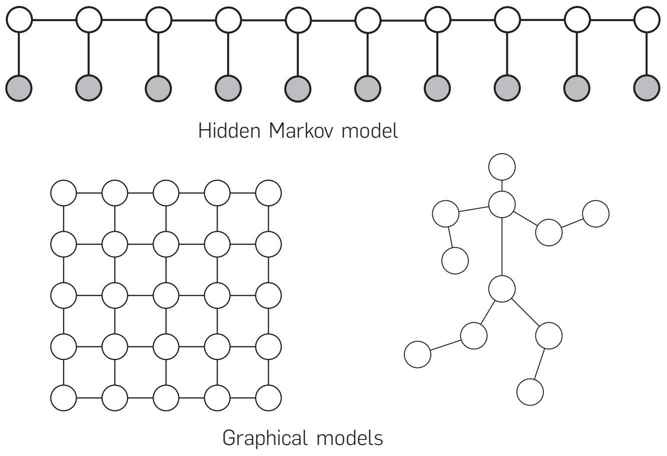
1. INTRODUCTION

Graphical models provide a powerful, general framework for developing statistical models in such diverse areas as bioinformatics, communications, natural language processing, and computer vision.[28] However, graphical formulations are only useful when combined with efficient algorithms for inference and learning. Such algorithms exist for many discrete models, such as those underlying modern error correcting codes and machine translation systems.

For most problems involving high-dimensional continuous variables, comparably efficient and accurate algorithms are unavailable. Alas, these are exactly the sorts of problems that arise frequently in areas like computer vision. Difficulties begin with the continuous surfaces and illuminants that digital cameras record in grids of pixels, and that geometric reconstruction algorithms seek to recover. At a higher level, the articulated models used in many tracking applications have dozens of degrees of freedom to be estimated at each time step.[41, 45] Realistic graphical models for these problems must represent outliers, bimodalities, and other non-Gaussian statistical features. The corresponding optimal inference procedures for these models typically involve integral equations for which no closed form solution exists. It is thus necessary to develop families of approximate representations, and algorithms for fitting those approximations.

In this work we describe the *nonparametric belief propagation* (NBP) algorithm. NBP combines ideas from Monte Carlo[3] and particle filtering[6, 11] approaches for representing complex uncertainty in time series, with the popular belief propagation (BP) algorithm[37] for approximate inference in complex graphical models. Unlike discretized approximations to continuous variables, NBP is not limited to low-dimensional domains. Unlike classical Gaussian approximations, NBP's particle-based messages can represent, and consistently reason about, the multimodal distributions induced by many real-world datasets. And unlike particle filters, NBP can exploit the rich nonsequential structure of more complex graphical models, like those in Figure 1.

Figure 1. Particle filters assume variables are related by a hidden Markov model (top). The NBP algorithm extends particle filtering techniques to arbitrarily structured graphical models, such as those for arrays of image pixels (bottom left) or articulated human motion (bottom right).



We begin in Section 2 by reviewing graphical models, BP, Monte Carlo methods, and particle filters. Section 3 then develops the two stages of the NBP algorithm: a belief fusion step which combines information from multiple particle sets, and a message propagation step which accounts for dependencies among random variables. We review a pair of previous real-world applications of NBP in Section 4: kinematic tracking of visual motion (Figures 6 and 7) and distributed localization in sensor networks (Figure 8). Finally, we conclude in Section 5 by surveying algorithmic and theoretical developments since the original introduction of NBP.

2. INFERENCE IN GRAPHICAL MODELS

Probabilistic graphical models decompose multivariate distributions into a set of local interactions among small subsets of variables. These local relationships produce conditional independencies which lead to efficient learning and inference algorithms. Moreover, their modular

The original versions of this paper were entitled "Nonparametric Belief Propagation," by E. Sudderth, A. Ihler, W. Freeman, and A. Willsky, and "PAMPAS: Real-Valued Graphical Models for Computer Vision," by M. Isard. Both appeared in the *IEEE Conference on Computer Vision and Pattern Recognition*, June 2003.

structure provides an intuitive language for expressing domain-specific knowledge about variable relationships and facilitates the transfer of modeling advances to new applications.

Several different formalisms have been proposed that use graphs to represent probability distributions.[28, 30, 47, 50] *Directed* graphical models, or *Bayesian networks*, are widely used in artificial intelligence to encode causal, generative processes. Such directed graphs provided the basis for the earliest versions of the BP algorithm.[37] Alternatively, *undirected* graphical models, or *Markov random fields* (MRFs), provide popular models for the symmetric dependencies arising in such areas as signal processing, spatial statistics, bioinformatics, and computer vision.

2.1. Pairwise Markov random fields

An undirected graph $\mathcal{G}$ is defined by a set of nodes $\mathcal{V}$ and a corresponding set of undirected edges $\mathcal{E}$ (see Figure 1). Let $\Gamma(i) \triangleq \{j \mid (i,j) \in \mathcal{E}\}$ denote the *neighborhood* of a node $i \in \mathcal{V}$. MRFs associate each node $i \in \mathcal{V}$ with an unobserved, or hidden, random variable $x_i \in \mathcal{X}_i$. Let $x = \{x_i \mid i \in \mathcal{V}\}$ denote the set of all hidden variables. Given evidence or observations y, a *pairwise MRF* represents the posterior distribution $p(x \mid y)$ in factored form:

$$p(x \mid y) \propto p(x, y) \propto \prod_{(i,j)\in\mathcal{E}} \psi_{ij}(x_i, x_j) \prod_{i\in\mathcal{V}} \psi_i(x_i, y). \qquad (1)$$

Here, the proportionality sign indicates that $p(x, y)$ may only be known up to an uncertain normalization constant, chosen so that it integrates to one. The positive *potential functions* $\psi_{ij}(x_i, x_j) > 0$ can often be interpreted as soft, local constraints. Note that the local evidence potential $\psi_i(x_i, y)$ does *not* typically equal the marginal distribution $p(x_i \mid y)$, due to interactions with other potentials.

In this paper, we develop algorithms to approximate the conditional marginal distributions $p(x_i \mid y)$ for all $i \in \mathcal{V}$. These densities lead to estimates of x_i, such as the posterior mean $\mathbb{E}[x_i \mid y]$, as well as corresponding measures of uncertainty. We focus on pairwise MRFs due to their simplicity and popularity in practical applications. However, the nonparametric updates we present may also be directly extended to models with higher-order potentials.

2.2. Belief propagation

For graphs that are acyclic or tree-structured, the desired conditional distributions $p(x_i \mid y)$ can be directly calculated by a local message-passing algorithm known as *belief propagation* (BP).[37, 50] At each iteration of the BP algorithm, nodes $j \in \mathcal{V}$ calculate messages $m_{ji}(x_i)$ to be sent to each neighboring node $i \in \Gamma(j)$:

$$m_{ji}(x_i) \propto \int_{\mathcal{X}_j} \psi_{ij}(x_i, x_j)\, \psi_j(x_j, y) \prod_{k\in\Gamma(j)\setminus i} m_{kj}(x_j) dx_j. \qquad (2)$$

The outgoing message is a positive function defined on $\mathcal{X}_i$. Intuitively, it is a (possibly approximate) sufficient statistic of the information that node j has collected regarding x_i. At any iteration, each node can produce an approximation $q_i(x_i)$ to the marginal distribution $p(x_i \mid y)$ by combining incoming messages with the local evidence potential:

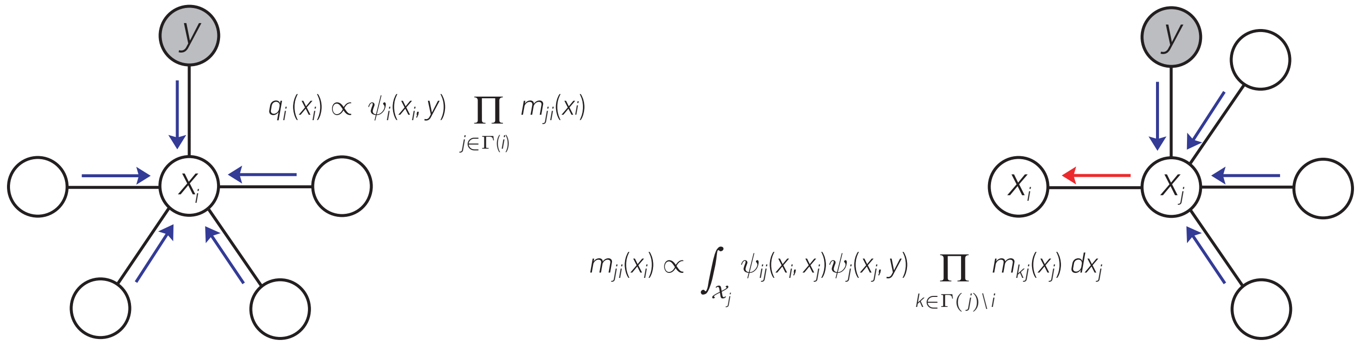
$$q_i(x_i) \propto \psi_i(x_i, y) \prod_{j\in\Gamma(i)} m_{ji}(x_i). \qquad (3)$$

These updates are graphically summarized in Figure 2. For tree-structured graphs, the approximate marginals, or *beliefs*, $q_i(x_i)$ will converge to the true marginals $p(x_i \mid y)$ once messages from each node have propagated across the graph. With an efficient update schedule, the messages for each distinct edge need only be computed once, and BP can be seen as a distributed variant of dynamic programming.

Because each iteration of the BP algorithm involves only local message updates, it can be applied even to graphs with cycles. For such graphs, the statistical dependencies between BP messages are not accounted for, and the sequence of beliefs $q_i(x_i)$ will *not* converge to the true marginals. In many applications, however, the resulting *loopy BP* algorithm[37] exhibits excellent empirical performance.[8, 14, 15, 49] Recently, several theoretical studies have provided insight into the approximations made by loopy BP, establishing connections to other *variational* inference algorithms[47] and partially justifying its application to graphs with cycles.[20, 23, 34, 50, 51]

The BP algorithm implicitly assumes messages $m_{ji}(x_i)$ have a finite parameterization, which can be tractably updated via the integral of Equation 2. Most implementations

Figure 2. Message-passing recursions underlying the BP algorithm. ***Left:*** **Approximate marginal (belief) estimates combine the local observation potential with messages from neighboring nodes.** ***Right:*** **A new outgoing message (red) is computed from all other incoming messages (blue).**


y
X_i
$q_i(x_i) \propto \psi_i(x_i, y) \prod_{j\in\Gamma(i)} m_{ji}(x_i)$
y
X_i
X_j
$m_{ji}(x_i) \propto \int_{\mathcal{X}_j} \psi_{ij}(x_i, x_j)\psi_j(x_j, y) \prod_{k\in\Gamma(j)\setminus i} m_{kj}(x_j)\, dx_j$

assume each hidden variable x_i takes one of K discrete values ($|\mathcal{X}_i| = K$), so that messages and marginals can be represented by K-dimensional vectors. The message update integral then becomes a matrix–vector product, which in general requires $\mathcal{O}(K^2)$ operations. This variant of BP is sometimes called the *sum–product algorithm*.[30]

For graphical models with continuous hidden variables, closed-form evaluation of the BP update integral is only possible when the posterior is jointly Gaussian. The resulting Gaussian BP algorithm, which uses linear algebra to update estimates of posterior mean vectors and covariance matrices, generalizes Kalman smoothing algorithms for linear dynamical systems.[2] More generally, a fixed K-point discretization sometimes leads to an effective histogram approximation of the true continuous beliefs.[13, 14] However, as K must in general grow exponentially with the dimension of $\mathcal{X}_i$, computation of the discrete messages underlying such approximations can be extremely demanding.

2.3. Monte Carlo methods

By using random samples to simulate probabilistic models, *Monte Carlo methods*[3] provide flexible alternatives to variational methods like BP. Given a target distribution $p(x \mid y)$, many inference tasks can be expressed via the expected value $\mathbb{E}_p[f(x)]$ of an appropriately chosen function. Given L independent samples $\{x^{(\ell)}\}_{\ell=1}^{L}$ from $p(x \mid y)$, the desired statistic can be approximated as follows:

$$\mathbb{E}_p[f(x)] = \int_{\mathcal{X}} f(x)p(x \mid y)dx \approx \frac{1}{L}\sum_{\ell=1}^{L} f(x^{(\ell)}). \tag{4}$$

This estimate is unbiased, and converges to $\mathbb{E}_p[f(x)]$ almost surely as $L \to \infty$. For the graphical models of interest here, however, exact sampling from $p(x \mid y)$ is intractable.

Importance sampling provides an alternative based on a *proposal distribution* $q(x)$, chosen so that $q(\bar{x}) > 0$ wherever $p(\bar{x} \mid y) > 0$. Defining the importance weight function as $w(x) = \bar{p}(x \mid y)/q(x)$, where $p(x \mid y) \propto \bar{p}(x \mid y)$ up to some potentially unknown normalization constant, the expectation of Equation 4 can be rewritten as follows:

$$\mathbb{E}_p[f(x)] = \frac{\int_{\mathcal{X}} f(x)w(x)q(x)dx}{\int_{\mathcal{X}} w(x)q(x)dx} \approx \sum_{\ell=1}^{L} w^{(\ell)} f(x^{(\ell)}), \tag{5}$$

$$x^{(\ell)} \sim q(x), \qquad w^{(\ell)} \triangleq \frac{w(x^{(\ell)})}{\sum_{m=1}^{L} w(x^{(m)})}.$$

Importance sampling thus estimates the target expectation via a collection of L *weighted* samples $\{(x^{(\ell)}, w^{(\ell)})\}_{\ell=1}^{L}$.

For high-dimensional models like the full joint distribution of Equation 1, designing tractable proposal distributions that closely approximate $p(x \mid y)$ is extremely challenging. Even minor discrepancies can produce widely varying importance weights $w^{(\ell)}$, which may in turn cause the estimator of Equation 5 to have a huge variance even for large L. Instead, we use importance sampling to *locally* approximate intermediate computations in the BP algorithm.

2.4. Particle filters

Our approach is inspired by particle filters, an approximate inference algorithm specialized for *hidden Markov models* (HMMs). As depicted graphically in Figure 1, an HMM models a sequence of T observations $y = \{y_1, \dots, y_T\}$ via a corresponding set of hidden states x:

$$p(x, y) = p(x_1)\, p(y_1 \mid x_1) \prod_{t=2}^{T} p(x_t \mid x_{t-1})\, p(y_t \mid x_t). \tag{6}$$

Recognizing this factorization as a special case of the pairwise MRF of Equation 1, the "forward" BP messages passed to subsequent time steps are defined via the recursion

$$\begin{aligned} m_{t,t+1}(x_{t+1}) &\propto \int_{\mathcal{X}_t} p(x_{t+1} \mid x_t) p(y_t \mid x_t) m_{t-1,t}(x_t) dx_t \\ &\propto p(x_{t+1} \mid y_1, \dots, y_t). \end{aligned} \tag{7}$$

For continuous $\mathcal{X}_t$ where this update lacks a closed form, *particle filters*[6, 11] approximate the forward BP messages $m_{t-1,t}(x_t)$ via a collection of L weighted samples, or particles, $\{(x_t^{(\ell)}, w_t^{(\ell)})\}_{\ell=1}^{L}$. Importance sampling is used to recursively update the particle locations and weights via a single, forward pass through the observations. A variety of proposal distributions $q(x_{t+1} \mid x_t, y_{t+1})$, which aim to approximate $p(x_{t+1} \mid x_t, y_{t+1})$, have been suggested.[6] For example, the "bootstrap filter" samples $x_{t+1}^{(\ell)} \sim p(x_{t+1} \mid x_t^{(\ell)})$, and incorporates evidence via weights $w_{t+1}^{(\ell)} \propto w_t^{(\ell)} p(y_t \mid x_t^{(\ell)})$.

For the simple algorithm sketched above, each message update introduces additional approximations, so that the expected variance of the importance weights $w_t^{(\ell)}$ increases over time. Particle filters avoid such sample depletion via a *resampling* operation, in which the highest-weight particles at time t determine a larger proportion of the outgoing message particles $x_{t+1}^{(\ell)}$. The bootstrap filter then becomes:

$$x_{t+1}^{(\ell)} \sim p(x_{t+1} \mid x_t^{(z_\ell)}), \; \Pr[z_\ell = m] \propto w_t^{(m)} p(y_t \mid x_t^{(m)}). \tag{8}$$

After such resampling, outgoing message particles are equally weighted as $w_{t+1}^{(\ell)} = 1/L$, $\ell = 1, \dots, L$. By stochastically selecting the highest-weight particles multiple times, resampling dynamically focuses the particle filter's computational resources on the most probable regions of the state space.

3. NONPARAMETRIC BP

Although particle filters can be adapted to an extremely wide range of dynamical models and observation types, they are specialized to the structure of temporal filtering problems. Conversely, loopy BP can in principle be applied to graphs of any structure, but is only analytically tractable when all hidden variables are discrete or jointly Gaussian. In this section, we describe an NBP algorithm[26, 44] that generalizes sequential Monte Carlo methods to arbitrary graphs. As in regularized particle filters,[11] we approximate the true BP messages and beliefs by nonparametric density estimates. Importance sampling and MCMC approximations then update these sample-based messages, propagating information from local observations throughout the graph.

3.1. Nonparametric representations

Consider again the BP algorithm of Section 2.2, and suppose that messages $m_{ji}(x_i)$ are approximated by a set of weighted, discrete samples. If $\mathcal{X}_i$ is continuous and these messages are constructed from independent proposal distributions, their particles will be distinct with probability one. For the message product operation underlying the BP algorithm to produce sensible results, some interpolation of these samples to nearby states is thus needed.

We accomplish this interpolation, and ensure that messages are smooth and strictly positive, by convolving raw particle sets with a Gaussian distribution, or kernel:

$$m_{ji}(x_i) = \sum_{\ell=1}^{L} w_{ji}^{(\ell)} \mathcal{N}\left(x_i; x_{ji}^{(\ell)}, \Lambda_{ji}\right), \tag{9}$$

$$q_i(x_i) = \sum_{\ell=1}^{L} w_i^{(\ell)} \mathcal{N}\left(x_i; x_i^{(\ell)}, \Lambda_i\right). \tag{10}$$

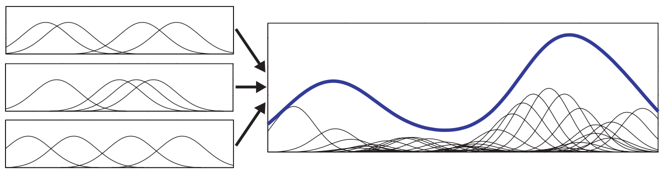
Here, $\mathcal{N}(x; \mu, \Lambda)$ denotes a normalized Gaussian density with mean μ and covariance Λ, evaluated at x. As detailed later, we use methods from the nonparametric density estimation literature to construct these mixture approximations.[42] The product of two Gaussians is itself a scaled Gaussian distribution, a fact which simplifies our later algorithms.

3.2. Message fusion

We begin by assuming that the observation potential is a Gaussian mixture. Such representations arise naturally from learning-based approaches to model identification.[14] The BP belief update of Equation 3 is then defined by a product of $d = (|\Gamma(i)| + 1)$ mixtures: the observation potential $\psi_i(x_i, y)$, and messages $m_{ji}(x_i)$ as in Equation 9 from each neighbor. As illustrated in Figure 3, the product of d Gaussian mixtures, each containing L components, is itself a mixture of L^d Gaussians. While in principle this belief update could be performed exactly, the exponential growth in the number of mixture components quickly becomes intractable.

The NBP algorithm instead approximates the product mixture via a collection of L independent, possibly importance weighted samples $\{(x_i^{(\ell)}, w_i^{(\ell)})\}_{\ell=1}^{L}$ from the "ideal" belief of Equation 3. Given these samples, the bandwidth Λ_i of the nonparametric belief estimate (Equation 10) is determined via a method from the extensive kernel density estimation literature.[42] For example, the simple "rule of thumb" method combines a robust covariance estimate with an asymptotic formula that assumes the target density is Gaussian. While fast to compute, it often oversmooths

Figure 3. A product of three mixtures of $L = 4$ 1D Gaussians. Although the $4^3 = 64$ components in the product density (thin lines) vary widely in position and weight (scaled for visibility), their normalized sum (thick line) has a simple form.



multimodal distributions. In such cases, more sophisticated cross-validation schemes can improve performance.

In many applications, NBP's computations are dominated by the cost of sampling from such products of Gaussian mixtures. Exact sampling by explicit construction of the product distribution requires $\mathcal{O}(L^d)$ operations. Fortunately, a number of efficient approximate samplers have been developed. One simple but sometimes effective approach uses an evenly weighted mixture of the d input distributions as an importance sampling proposal. For higher-dimensional variables, iterative Gibbs sampling algorithms are often more effective.[44] Multiscale KD-tree density representations can improve the mixing rate of Gibbs samplers, and also lead to "epsilon-exact" samplers with accuracy guarantees.[25] More sophisticated importance samplers[5] and multiscale simulated or parallel tempering algorithms[39] can also be highly effective. Yet more approaches improve efficiency by introducing an additional message approximation step.[19, 22, 31] By first reducing the complexity of each message, the product can be approximated more quickly, or even computed exactly. When $\psi_i(x_i, y)$ is a non-Gaussian analytic function, we can use any of these samplers to construct an importance sampling proposal from the incoming Gaussian mixture messages.

3.3. Message propagation

The particle filter of Section 2.4 propagates belief estimates to subsequent time steps by sampling $x_{t+1}^{(\ell)} \sim p(x_{t+1} \mid x_t^{(\ell)})$. The consistency of this procedure depends critically on the HMM's factorization into properly normalized conditional distributions, so that $\int p(x_{t+1} \mid x_t) dx_{t+1} = 1$ for all $x_t \in \mathcal{X}_t$. By definition, such conditional distributions place no biases on x_t. In contrast, for pairwise MRFs, the clique potential $\psi_{ij}(x_i, x_j)$ is an arbitrary nonnegative function that may influence the values assumed by either linked variable. To account for this, we quantify the *marginal influence* of $\psi_{ij}(x_i, x_j)$ on x_j via the following function:

$$\varphi_{ij}(x_j) = \int_{\mathcal{X}_i} \psi_{ij}(x_i, x_j) dx_i. \tag{11}$$

If $\psi_{ij}(x_i, x_j)$ is a Gaussian mixture, $\psi_{ij}(x_j)$ is simply the mixture obtained by marginalizing each component. In the common case where $\psi_{ij}(x_i, x_j) = \tilde{\psi}_{ij}(x_i - x_j)$ depends only on the difference between neighboring variables, the marginal influence is constant and may be ignored.

As summarized in the algorithm of Figure 4, NBP approximates the BP message update of Equation 2 in two stages. Using the efficient algorithms discussed in Section 3.2, we first draw L independent samples $\tilde{x}_j^{(\ell)}$ from a partial belief estimate combining the marginal influence function, observation potential, and incoming messages. For each of these auxiliary particles $\tilde{x}_j^{(\ell)}$, we then interpret the clique potential as a joint distribution and sample particles $x_{ji}^{(\ell)} \in \mathcal{X}_i$ from the conditional density proportional to $\psi_{ij}(x_i, x_j = \tilde{x}_j^{(\ell)})$.

Particle-based approximations are only meaningful when the corresponding BP messages $m_{ji}(x_i)$ are finitely integrable. Some models, however, contain nonnormalizable potentials that nevertheless encode important constraints. For example, the kinematic tracking and sensor localization applications considered in Section 4 both involve "repulsive" potentials, that encourage pairs of variables to *not* take similar values. In such cases, the NBP algorithm in Figure 4 instead stores the weighted particles needed to evaluate $m_{ji}(\bar{x}_i)$ at any location $\bar{x}_i$ of interest. These messages then influence subsequent iterations via importance weighting.

As illustrated in Figure 2, the BP update of message $m_{ji}(x_i)$ is most often expressed as a transformation of the incoming messages from all *other* neighboring nodes $k \in \Gamma(j)\backslash i$. From Equations 2 and 3, however, it can also be expressed as

$$m_{ji}(x_i) \propto \int_{\mathcal{X}_j} \psi_{ij}(x_i, x_j)\, \frac{q_j(x_j)}{m_{ij}(x_j)}\, dx_j. \tag{12}$$

This transformation suggests an alternative *belief sampling* form of the NBP algorithm, in which the latest belief estimate provides a proposal distribution for auxiliary particles $\tilde{x}_j^{(\ell)} \sim q_j(x_j)$. Overcounting of $m_{ij}(x_j)$ may then be avoided via importance weights $\tilde{w}_j^{(\ell)} \propto 1/m_{ij}(\tilde{x}_j^{(\ell)})$. Computationally, belief sampling offers clear advantages: computation of new outgoing messages to d neighbors requires $\mathcal{O}(dL)$ operations, versus the $\mathcal{O}(d^2L)$ cost of the approach in Figure 4. Statistically, belief sampling also has potentially desirable properties,[26, 29] but can be less stable when the number of particles L is small.[22]

Figure 4. Nonparametric BP update for the message $m_{ji}(x_i)$ sent from node j to node i, as in Figure 2.

Given input messages $m_{kj}(x_j)$ for each $k \in \Gamma(j)\backslash i$, which may be either kernel densities $m_{kj}(x_j) = \{x_{kj}^{(\ell)}, w_{kj}^{(\ell)}, \Lambda_{kj}\}_{\ell=1}^{L}$ or analytic functions, construct an output message $m_{ji}(x_i)$ as follows:

1. Determine the marginal influence $\varphi_{ij}(x_j)$ of Equation (11).
2. Draw L independent, weighted samples from the product

   $$(\tilde{x}_j^{(\ell)}, \tilde{w}_j^{(\ell)}) \sim \varphi_{ij}(x_j)\psi_j(x_j, y) \prod_{k\in\Gamma(j)\backslash i} m_{kj}(x_j).$$

   Optionally resample by drawing L particles with replacement according to $\Pr\left[\tilde{x}_j^{(\ell)}\right] \propto \tilde{w}_j^{(\ell)}$, giving evenly weighted particles.
3. If $\psi_{ij}(x_i, x_j)$ is normalizeable ($\int \psi_{ij}(x_i, x_j = \bar{x})\, dx_i < \infty$ for all $\bar{x} \in \mathcal{X}_j$), construct a kernel-based output message:

   (a) For each auxiliary particle $\tilde{x}_j^{(\ell)}$, sample an outgoing particle

   $$x_{ji}^{(\ell)} \sim \psi_{ij}(x_i, x_j = \tilde{x}_j^{(\ell)})$$

   Using importance sampling or MCMC methods as needed.

   (b) Set $w_{ji}^{(\ell)}$ to account for importance weights in steps 2–4(a).

   (c) Set Λ_i via some bandwidth selection method (see Silverman[42]).
4. Otherwise, treat $m_{ji}(x_i)$ as an analytic function

   $$m_{ji}(x_i) \propto \sum_{\ell=1}^{L} \tilde{w}_j^{(\ell)} \psi_{ij}(x_i, \tilde{x}_j^{(\ell)})$$

   parameterized by the auxiliary particles $\{\tilde{x}_j^{(\ell)}, \tilde{w}_j^{(\ell)}\}_{\ell=1}^{L}$.

4. ILLUSTRATIVE APPLICATIONS

In this section we show several illustrative examples of applications that use NBP to reason about structured collections of real-valued variables. We first show examples of kinematic tracking problems in computer vision, in which the variables represent the spatial position of parts of an object. We then show how a similar formulation can be used for collaborative self-localization and tracking in wireless sensor networks. Other applications of NBP include deformable contour tracking for medical image segmentation,[46] image denoising and super-resolution,[38] learning flexible models of dynamics and motor response in robotic control,[17] error correcting codes defined for real-valued codewords,[31, 43] and sparse signal reconstruction using compressed sensing principles.[4] NBP has also been proposed as a computational mechanism for hierarchical Bayesian information processing in the visual cortex.[32]

4.1. Visual tracking of articulated motion

Visual tracking systems use video sequences from one or more cameras to estimate object motion. Some of the most challenging tracking applications involve *articulated* objects, whose jointed motion leads to complex pose variations. For example, human motion capture is widely used in visual effects and scene understanding applications.[33] Estimates of human, and especially hand, motion are also used to build more expressive computer interfaces.[48]

To illustrate the difficulties, we consider a toy 2D object localization problem in Figure 5. The model consists of nine nodes: a central circle, and four jointed arms composed of two rectangular links. The circle node's state x_0 encodes its position and radius, while each rectangular link node's state x_i encodes its position, angle, width, and height. Each arm prefers one of the four compass directions, arms pivot around their inner joints, and geometry is loosely enforced via Gaussian pairwise potentials $\psi_{ij}(x_i, x_j)$; for details see Isard.[26]

Our goal is to find the object in a sea of clutter (white shapes in Figure 5) whose elements look exactly like components of the object. This mimics the difficulties faced in real video sequences: statistical detectors for individual object parts often falsely fire on background regions, and global geometric reasoning is needed to disambiguate them. Applied to this model, NBP's particles encode hypotheses about the pose x_i of individual object parts, while messages use geometric constraints to propagate information between parts. When all of the true object's parts are visible, NBP localizes it after a single iteration. By using Gaussian mixture potentials $\psi_i(x_i, y)$ that allow occasional outliers in observed part locations, NBP remains successful even when the central circle is missing. In this case, however, it takes more iterations to propagate information from the visible arms.

Kinematic tracking of real hand motion poses far greater challenges. Even coarse models of the hand's geometry have 26 continuous degrees of freedom: each finger's joints have four angles of rotation, and the palm

Figure 5. Detection of a toy, four-armed articulated object (top row) in clutter. We show NBP estimates after 0, 1, and 3 iterations (columns), for cases where the circular central joint is either visible (middle row) or occluded (bottom row).

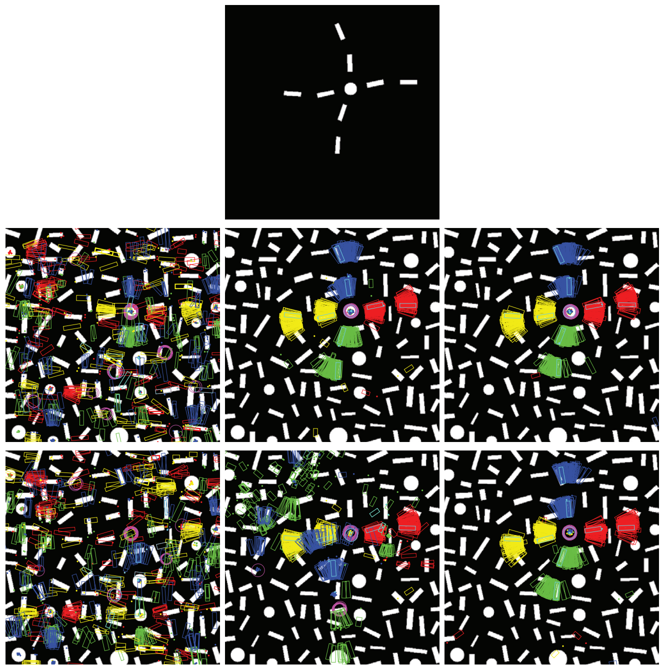
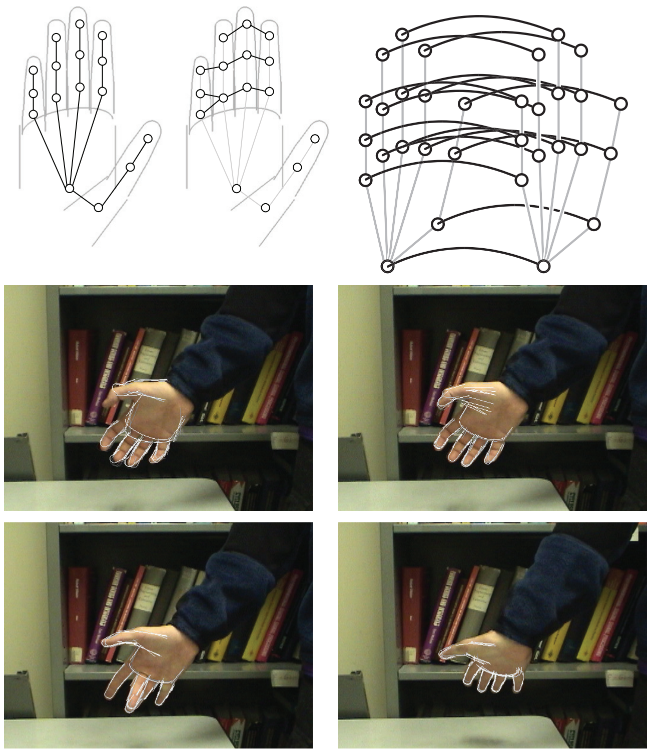


Figure 6. Articulated 3D hand tracking with NBP. ***Top*****: Graphical models capturing the kinematic, structural, and temporal constraints relating the hand's 16 rigid bodies.** ***Middle*****: Given a single input image, projected estimates of hand pose after one (left) and four (right) NBP iterations.** ***Bottom*****: Two frames showing snapshots of tracking performance from a monocular video sequence.**



may take any 3D position and orientation.[48] The graphical models in Figure 6 instead encode hand pose via the 3D pose of 16 rigid bodies.[45] Analytic pairwise potentials then capture kinematic constraints (phalanges are connected by revolute joints), structural constraints (two fingers cannot occupy the same 3D volume), and Markov temporal dynamics. The geometry of individual rigid bodies is modeled via quadric surfaces (a standard approach in computer graphics), and related to observed images via statistical color and edge cues.[45]

Because different fingers are locally similar in appearance, global inference is needed to accurately associate hand components to image cues. Discretization of the 6D pose variable for each rigid body is intractable, but as illustrated in Figure 6, NBP's sampling-based message approximations often lead to accurate hand localization and tracking. While we project particle outlines to the image plane for visualization, we emphasize NBP's estimates are of 3D pose.

Finally, Figure 7 illustrates a complementary approach to multicamera tracking of 3D person motion.[41] While the hand tracker used rigid kinematic potentials, this graphical model of full-body pose is explicitly "loose limbed," and uses pairwise potentials estimated from calibrated, 3D motion capture data. Even without the benefit of dynamical cues or highly accurate image-based likelihoods, we see that NBP successfully infers the full human body pose.

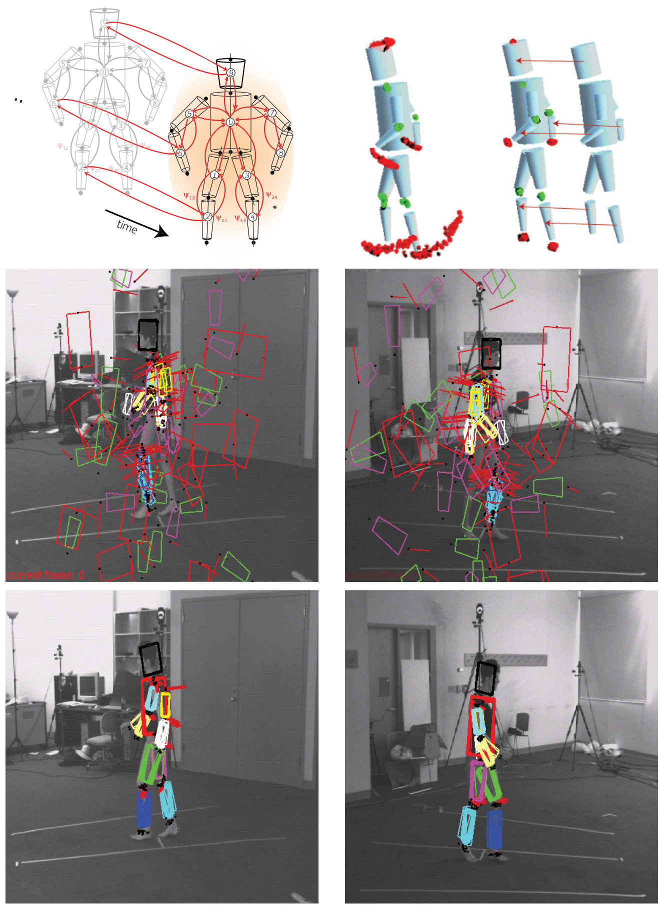
4.2. Sensor self-localization

Another problem for which NBP has been very successful is sensor localization.[22] One of the critical first tasks in using ad-hoc networks of wireless sensors is to determine the location of each sensor; the high cost of manual calibration or specialized hardware like GPS units makes self-localization, or estimating position based on local in-network information, very appealing. As with articulated tracking, we will be estimating the position of a number of objects (sensors) using joint information about the objects' relative positions. Specifically, let us assume that some subset of pairs of sensors $(i,j) \in \mathcal{E}$ are able to measure a noisy estimate of their relative distance (e.g., through signal strength of wireless communication or measuring time delays of acoustic signals). Our measurements y_{ij} tell us something about the relative positions x_i, x_j of two sensors; assuming independent noise, the likelihood of our measurements is

$$p(y \mid x) = \prod_{(i,j)\in\mathcal{E}} p(y_{ij} \mid x_i, x_j) = \prod_{(i,j)\in\mathcal{E}} \psi_{ij}(x_i, x_j).$$

We can see immediately that this likelihood has the form of a pairwise graphical model whose edges are the pairs of sensors with distance measurements. Typically we assume a small number of *anchor* sensors with known or partially known position to remove translational, rotational, and mirror image ambiguity from the geometry.

Figure 7. Articulated 3D person tracking with NBP.[41] *Top*: Graphical model encoding kinematic and dynamic relationships (left), and spatial and temporal potential functions (right) learned from mocap data. *Middle*: Bottom-up limb detections, as seen from two of four camera views. *Bottom*: Estimated body pose following 30 iterations of NBP.



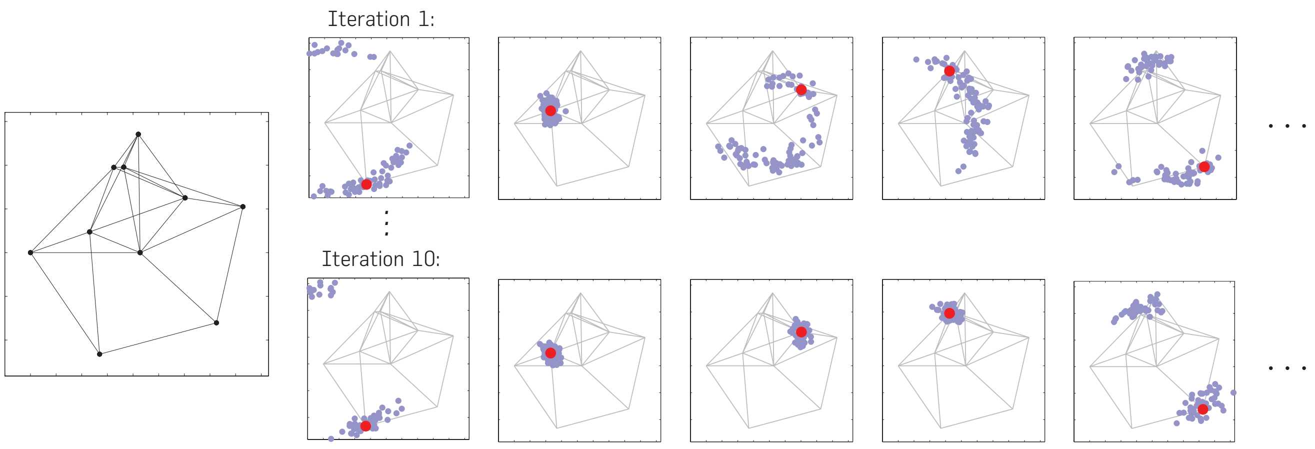
A small 10-sensor network with 24 edges is shown in Figure 8, indicating both the true 2D sensor positions (nodes) and inter-sensor measurements (edges). The beliefs obtained using NBP are displayed on the right, by showing 500 samples from the estimated belief; the true sensor positions are also superimposed (red dots). The initial beliefs are highly non-Gaussian and often fairly diffuse (top row). As information propagates through the graph and captures more of the inter-sensor dependencies, these beliefs tend to shrink to good estimates of the sensor positions. However, in some cases, the measurements themselves are nearly ambiguous, resulting in a bimodal posterior distribution. For example, the sensor located in the bottom right has only three, nearly colinear neighbors, and so can be explained almost as well by "flipping" its position across the line. Such bimodalities indicate that the system is not fully constrained, and are important to identify as they indicate sensors with potentially significant errors in position.

5. DISCUSSION

The abundance of problems that involve continuous variables has given rise to a variety of related algorithms for estimating posterior probabilities and beliefs in these systems. Here we describe several influential historical predecessors of NBP, and then discuss subsequent work that builds on or extends some of the same ideas.

As mentioned in Section 2.2, direct discretization of continuous variables into binned "histogram" potentials can be effective in problems with low-dimensional variables.[4] In higher-dimensional problems, however, the number of bins grows exponentially and quickly becomes intractable. One possibility is to use domain specific heuristics to exclude those configurations that appear unlikely based on local evidence.[8, 14] However, if the local evidence used to discard states is inaccurate

Figure 8. NBP for self-localization in a small network of 10 sensors. *Left*: Sensor positions, with edges connecting sensor pairs with noisy distance measurements. *Right*: Each panel shows the belief of one sensor (scatterplot), along with its true location (red dot). After the first iteration of message passing, beliefs are diffuse with non-Gaussian uncertainty. After 10 iterations, the beliefs have stabilized near the true values. Some beliefs remain multimodal, indicating a high degree of uncertainty in that sensor's position due to near-symmetries that remain ambiguous given the measurements.



or misleading, these approximations will heavily distort the resulting estimates.

One advantage of Monte Carlo and particle filtering methods lies in the fact that their discretization of the state space is obtained stochastically, and thus has excellent theoretical properties. Examples include statistical consistency, and convergence rates that do not depend on the dimension.[10] While particle filters are typically restricted to "forward" sequential inference, the connection to discrete inference has been exploited to define smoothing (forward and backward) algorithms,[6] and to perform resampling to dynamically improve the approximation.[35] Monte Carlo approximations were also previously applied to other tree-structured graphs, including junction trees.[9, 29]

Gaussian mixture models also have a long history of use in inference. In Markov chains, an algorithm for forward inference using Gaussian mixture approximations was first proposed by Alspach and Sorenson[1]; see also Anderson and Moore.[2] Regularized particle filters smooth each particle with a small, typically Gaussian kernel to produce a mixture model representation of forward messages.[11] For Bayesian networks, Gaussian mixture-based potentials and messages have been applied to junction tree-based inference.[12]

NBP combines many of the best elements of these methods. By sampling, we obtain probabilistic approximation properties similar to particle filtering. Representing messages as Gaussian mixture models provides smooth estimates similar to regularized particle filters, and interfaces well with Gaussian mixture estimates of the potential functions.[12, 14, 17, 38] NBP extends these ideas to "loopy" message passing and approximate inference.

Since the original development of NBP, a number of algorithms have been developed that use alternative representations for inference on continuous, or hybrid, graphical models. Of these, the most closely related is particle BP, which uses a simplified importance sampling representation of messages, more closely resembling the representation of (unregularized) particle filters. This form enables the derivation of convergence rates similar to those available for particle filtering,[21] and also allows the algorithm to be extended to more general inference techniques such as reweighted message-passing algorithms.[24]

Other forms of message representation have also been explored. Early approaches to deterministic discrete message approximation would often mistakenly discard states in the early stages of inference, due to misleading local evidence. More recently, dynamic discretization techniques have been developed to allow the inference process to recover from such mistakes by re-including states that were previously removed.[7, 27, 36] Other approaches substitute alternative, smoother message representations, such as Gaussian process-based density estimates.[40]

Finally, several authors have developed additional ways of combining Monte Carlo sampling with the principles of exact inference. AND/OR importance sampling,[16] for example, uses the structure of the graph to improve the statistical efficiency of Monte Carlo estimates given a set of samples. Another example, Hot Coupling,[18] uses a sequential ordering of the graph's edges to define a sequence of importance sampling distributions.

The intersection of variational and Monte Carlo methods for approximate inference remains an extremely active research area. We anticipate many further advances in the coming years, driven by increasingly varied and ambitious real-world applications.

Acknowledgments

The authors thank L. Sigal, S. Bhatia, S. Roth, and M. Black for providing the person tracking results in Figure 7. Work of WTF supported by NGA NEGI-1582-04-0004 and by MURI Grant N00014-06-1-0734. Work of ASW supported by AFOSR Grant FA9559-08-1-1080 and a MURI funded through AFOSR Grant FA9550-06-1-0324.

References

1. Alspach, D.L. and Sorenson, H.W. Nonlinear Bayesian estimation using Gaussian sum approximations, Morgan Kaufmann. *IEEE Trans. AC 17*, 4 (Aug. 1972), 439–448.
2. Anderson, B.D.O., Moore, J.B. *Optimal Filtering*. Prentice Hall, New Jersey, 1979.
3. Andrieu, C., de Freitas, N., Doucet, A., Jordan, M.I. An introduction to MCMC for machine learning. *Mach. Learn. 50* (2003), 5–43.
4. Baron, D., Sarvotham, S., Baraniuk, R.G. Bayesian compressive sensing via belief propagation. *IEEE Trans.Sig.Proc. 58*, 1 (2010), 269–280.
5. Briers, M., Doucet, A., Singh, S.S. Sequential auxiliary particle belief propagation. In *ICIF* (2005), 705–711.
6. Cappé, O., Godsill, S.J., Moulines, E. An overview of existing methods and recent advances in sequential Monte Carlo. *Proc. IEEE 95*, 5 (May 2007), 899–924.
7. Coughlan, J., Shen, H. Dynamic quantization for belief propagation in sparse spaces. *Comput. Vis. Image Underst. 106*, 1 (2007), 47–58.
8. Coughlan, J.M., Ferreira, S.J. Finding deformable shapes using loopy belief propagation. In *ECCV*, vol. 3, (2002), 453–468.
9. Dawid, A.P., Kjærulff, U., Lauritzen, S.L. Hybrid propagation in junction trees. In *Advances in Intelligent Computing* (1995), 87–97.
10. Del Moral, P. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer-Verlag, New York, 2004.
11. Doucet, A., de Freitas, N., Gordon, N., eds. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.
12. Driver, E., Morrell, D. Implementation of continuous Bayesian networks using sums of weighted Gaussians. In *UAI* (1995), 134–140.
13. Felzenszwalb, P.F., Huttenlocher, D.P. Pictorial structures for object recognition. *IJCV 61*, 1 (2005), 55–79.
14. Freeman, W.T., Pasztor, E.C., Carmichael, O.T. Learning low-level vision. *IJCV 40*, 1 (2000), 25–47.
15. Frey, B.J., MacKay, D.J.C. A revolution: Belief propagation in graphs with cycles. In *NIPS 10* (1998), MIT Press, 479–485.
16. Gogate, V., Dechter, R. AND/OR importance sampling. In *UAI* (2008), 212–219.
17. Grimes, D.B., Rashid, D.R., Rao, R.P. Learning nonparametric models for probabilistic imitation. In *NIPS* (2007), MIT Press, 521–528.
18. Hamze, F., de Freitas, N. Hot coupling: A particle approach to inference and normalization on pairwise undirected graphs of arbitrary topology. In *NIPS 18* (2006), MIT Press, 491–498.
19. Han, T.X., Ning, H., Huang, T.S. Efficient nonparametric belief propagation with application to articulated body tracking. In *CVPR* (2006), 214–221.
20. Heskes, T. On the uniqueness of loopy belief propagation fixed points. *Neural Comp. 16* (2004), 2379–2413.
21. Ihler, A., McAllester, D. Particle belief propagation. In *AI Stat. 12* (2009).
22. Ihler, A.T., Fisher, J.W., Moses, R.L., Willsky, A.S. Nonparametric belief propagation for self-localization of sensor networks. *IEEE J. Sel. Areas Commun. 23*, 4 (Apr. 2005), 809–819.
23. Ihler, A.T., Fisher, J.W., Willsky, A.S. Loopy belief propagation: Convergence and effects of message errors. *JMLR 6* (2005), 905–936.
24. Ihler, A.T., Frank, A.J., Smyth, P. Particle-based variational inference for continuous systems. In *NIPS 22* (2009), 826–834.
25. Ihler, A.T., Sudderth, E.B., Freeman, W.T., Willsky, A.S. Efficient multiscale sampling from products of Gaussian mixtures. In *NIPS 16* (2004), MIT Press.
26. Isard, M. PAMPAS: Real-valued graphical models for computer vision. In *CVPR*, vol. 1 (2003), 613–620.
27. Isard, M., MacCormick, J., Achan, K. Continuously-adaptive discretization for message-passing algorithms. In *NIPS* (2009), MIT Press, 737–744.
28. Jordan, M.I. Graphical models. *Stat.*

Sci. 19, 1 (2004), 140–155.
29. Koller, D., Lerner, U., Angelov, D. A general algorithm for approximate inference and its application to hybrid Bayes nets. In *UAI 15* (1999), Morgan Kaufmann, 324–333.
30. Kschischang, F.R., Frey, B.J., Loeliger, H.-A. Factor graphs and the sum-product algorithm. *IEEE Trans. IT 47*, 2 (Feb. 2001), 498–519.
31. Kurkoski, B., Dauwels, J. Message-passing decoding of lattices using Gaussian mixtures. In *ISIT* (July 2008).
32. Lee, T.S., Mumford, D. Hierarchical Bayesian inference in the visual cortex. *J. Opt. Soc. Am. A 20*, 7 (July 2003), 1434–1448.
33. Moeslund, T.B., Hilton, A., Kruger, V. A survey of advances in vision-based human motion capture and analysis. *Comput. Vision Image Underst. 104* (2006), 90–126.
34. Mooij, J.M., Kappen, H.J. Sufficient conditions for convergence of the sum-product algorithm. *IEEE Trans. IT 53*, 12 (Dec. 2007), 4422–4437.
35. Neal, R.M., Beal, M.J., Roweis, S.T. Inferring state sequences for non-linear systems with embedded hidden Markov models. In *NIPS 16* (2004), MIT Press.
36. Neil, M., Tailor, M., Marquez, D. Inference in hybrid Bayesian networks using dynamic discretization. *Stat. Comput. 17*, 3 (2007), 219–233.
37. Pearl, J. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufman, San Mateo, 1988.
38. Rajaram, S., Gupta, M.D., Petrovic, N., Huang, T.S. Learning-based nonparametric image super-resolution. *EURASIP J. Appl. Signal Process.* (2006), 229–240.
39. Rudoy, D. Wolf, P.J. Multi-scale MCMC methods for sampling from products of Gaussian mixtures. In *ICASSP*, vol. 3 (2007), III-1201–III-1204.
40. Seeger M. Gaussian process belief propagation. In *Predicting structured data* (2007), 301–318.
41. Sigal, L., Bhatia, S., Roth, S., Black, M.J., Isard, M. Tracking loose-limbed people. In *CVPR* (2004).
42. Silverman, B.W. *Density Estimation for Statistics and Data Analysis.* Chapman & Hall, London, 1986.
43. Sommer, N., Feder, M., Shalvi, O. Low-density lattice codes. *IEEE Trans. Info. Theory 54*, 4 (2008), 1561–1585.
44. Sudderth, E.B., Ihler, A.T., Freeman, W.T., Willsky, A.S. Nonparametric belief propagation. In *CVPR*, vol. 1 (2003), 605–612.
45. Sudderth, E.B., Mandel, M.I., Freeman, W.T., Willsky, A.S. Visual hand tracking using nonparametric belief propagation. In *CVPR Workshop on Generative Model Based Vision* (June 2004).
46. Sun, W., Cetin, M., Chan, R., Willsky,, A.S. Learning the dynamics and time-recursive boundary detection of deformable objects. *IEEE Trans. IP 17*, 11 (Nov. 2008), 2186–2200.
47. Wainwright, M.J., Jordan, M.I. Graphical models, exponential families, and variational inference. *Foundations Trends Mach. Learn. 1*, (2008), 1–305.
48. Wu, Y., Huang, T.S. Hand modeling, analysis, and recognition. *IEEE Signal Proc. Mag.* (May 2001), 51–60.
49. Yanover, C., Weiss, Y. Approximate inference and protein-folding. In *NIPS 16* (2003), MIT Press, 1457–1464.
50. Yedidia, J.S., Freeman, W.T., Weiss, Y. Understanding belief propagation and its generalizations. In G. Lakemeyer and B. Nebel, eds. *Exploring Artificial Intelligence in the New Millennium.* Morgan Kaufmann, 2002.
51. Yedidia, J.S., Freeman, W.T., Weiss, Y. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Trans. IT 51*, 7 (July 2005), 2282–2312.

Erik B. Sudderth (sudderth@cs.brown.edu), Brown University, Providence, RI.

Alexander T. Ihler (ihler@ics.uci.edu), University of California, Irvine.

Michael Isard (misard@microsoft.com), Microsoft Research, Mountain View, CA.

William T. Freeman (billf@mit.edu), Massachusetts Institute of Technology, Cambridge, MA.

Alan S. Willsky (willsky@mit.edu), Massachusetts Institute of Technology, Cambridge, MA.