# Nonparametric Belief Propagation

Erik B. Sudderth, Alexander T. Ihler, William T. Freeman, and Alan S. Willsky
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
*esuddert@mit.edu, ihler@mit.edu, wtf@ai.mit.edu, willsky@mit.edu*

## Abstract

*In many applications of graphical models arising in computer vision, the hidden variables of interest are most naturally specified by continuous, non-Gaussian distributions. There exist inference algorithms for discrete approximations to these continuous distributions, but for the high-dimensional variables typically of interest, discrete inference becomes infeasible. Stochastic methods such as particle filters provide an appealing alternative. However, existing techniques fail to exploit the rich structure of the graphical models describing many vision problems.*

*Drawing on ideas from regularized particle filters and belief propagation (BP), this paper develops a nonparametric belief propagation (NBP) algorithm applicable to general graphs. Each NBP iteration uses an efficient sampling procedure to update kernel-based approximations to the true, continuous likelihoods. The algorithm can accomodate an extremely broad class of potential functions, including nonparametric representations. Thus, NBP extends particle filtering methods to the more general vision problems that graphical models can describe. We apply the NBP algorithm to infer component interrelationships in a parts-based face model, allowing location and reconstruction of occluded features.*

## 1. Introduction

Graphical models provide a powerful, general framework for developing statistical models of computer vision problems [7, 8, 10]. However, graphical formulations are only useful when combined with efficient algorithms for inference and learning. Computer vision problems are particularly challenging because they often involve high–dimensional, continuous variables and complex, multimodal distributions. For example, the articulated models used in many tracking applications have dozens of degrees of freedom to be estimated at each time step [17]. 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. Thus, it is necessary to develop families of approximate representations, and corresponding methods for updating those approximations.

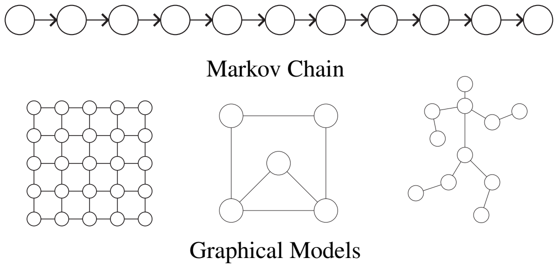



**Figure 1. Particle filters assume variables are related by a Markov chain. The NBP algorithm extends particle filtering techniques to arbitrarily structured graphical models.**

The simplest method for approximating intractable continuous–valued graphical models is discretization. Although exact inference in general discrete graphs is NP hard, approximate inference algorithms such as loopy belief propagation (BP) [16, 21, 23, 24] produce excellent empirical results in many cases. Certain vision problems, including stereo vision [20], are well suited to discrete formulations. For problems involving high–dimensional variables, however, exhaustive discretization of the state space is intractable. In some cases, domain–specific heuristics may be used to dynamically exclude those configurations which appear unlikely based upon the local evidence [2, 7]. In more challenging applications, however, the local evidence at some nodes may be inaccurate or misleading, and these approximations will heavily distort the resulting estimates.

For temporal inference problems, particle filters [5, 10] have proven to be an effective, and influential, alternative to discretization. They provide the basis for several of the most effective visual tracking algorithms [15, 17]. Particle filters approximate conditional densities nonparametrically as a collection of representative elements. Although it is possible to update these approximations deterministically using local linearizations [1], most implementations use Monte Carlo methods to stochastically update a set of weighted point samples. The stability and robustness of particle filters can often be improved by regularization methods [5, Chapter 12] in which smoothing kernels [18] explicitly represent the uncertainty associated with each point sample.

Although particle filters are often very effective, they are specialized to temporal problems whose corresponding graphs are simple Markov chains (see Figure 1). Many vision problems, however, are characterized by non–causal (e.g., spatial or model–induced) structure which is better represented by a more complex graph. Because particle filters cannot be applied to arbitrary graphs, graphical models containing high–dimensional variables may pose severe problems for existing inference algorithms. Even for tracking problems, there is often structure within each time instant (for example, associated with an articulated model) which is ignored by standard particle filters.

Some authors have used junction tree representations [12] to develop structured approximate inference techniques for general graphs. These algorithms begin by clustering nodes into cliques chosen to break the original graph's cycles. A wide variety of algorithms can then be specified by combining an approximate clique variable representation with local methods for updating these approximations [3, 11]. For example, Koller et al. [11] propose a framework in which the current clique potential estimate is used to guide message computations, allowing approximations to be gradually refined over successive iterations. However, the sample algorithm they provide is limited to networks containing mixtures of discrete and Gaussian variables. In addition, for many graphs (e.g. nearest–neighbor grids) the size of the junction tree's largest cliques grows exponentially with problem size, requiring the estimation of extremely high–dimensional distributions.

The *nonparametric belief propagation* (NBP) algorithm we develop in this paper differs from previous nonparametric approaches in two key ways. First, for graphs with cycles we do *not* form a junction tree, but instead iterate our local message updates until convergence as in loopy BP. This has the advantage of greatly reducing the dimensionality of the spaces over which we must infer distributions. Second, we provide a message update algorithm specifically adapted to graphs containing continuous, non–Gaussian potentials. The primary difficulty in extending particle filters to general graphs is in determining efficient methods for combining the information provided by several neighboring nodes. Representationally, we address this problem by associating a regularizing kernel with each particle, a step which is necessary to make message products well defined. Computationally, we show that message products may be computed using an efficient *local* Gibbs sampling procedure. The NBP algorithm may be applied to arbitrarily structured graphs containing a broad range of potential functions, effectively extending particle filtering methods to a much broader range of vision problems. For a related generalization of particle filters to graphical models, see [9].

Following our presentation of the NBP algorithm, we validate its performance on a small Gaussian network. We then show how NBP may be combined with parts–based local appearance models [4, 6, 14, 22] to locate and reconstruct occluded facial features.

## 2. Undirected Graphical Models

An undirected graph $\mathcal{G}$ is defined by a set of nodes $\mathcal{V}$, and a corresponding set of edges $\mathcal{E}$. The *neighborhood* of a node $s \in \mathcal{V}$ is defined as $\Gamma(s) \triangleq \{t \mid (s,t) \in \mathcal{E}\}$. Graphical models associate each node $s \in \mathcal{V}$ with an unobserved, or hidden, random variable $x_s$, as well as a noisy local observation $y_s$. Let $x = \{x_s \mid s \in \mathcal{V}\}$ and $y = \{y_s \mid s \in \mathcal{V}\}$ denote the sets of all hidden and observed variables, respectively. For simplicity, we consider models with pairwise potential functions, for which $p(x,y)$ factorizes as

$$p(x,y) = \frac{1}{Z} \prod_{(s,t)\in\mathcal{E}} \psi_{s,t}(x_s,x_t) \prod_{s\in\mathcal{V}} \psi_s(x_s,y_s) \qquad (1)$$

However, the nonparametric updates we present may be directly extended to models with higher–order potentials.

In this paper, we focus on the calculation of the conditional marginal distributions $p(x_s \mid y)$ for all nodes $s \in \mathcal{V}$. These densities provide not only estimates of $x_s$, but also corresponding measures of uncertainty.

### 2.1. Belief Propagation

For graphs which are acyclic or tree–structured, the desired conditional distributions $p(x_s \mid y)$ can be directly calculated by a local message–passing algorithm known as *belief propagation* (BP) [16, 24]. At iteration $n$ of the BP algorithm, each node $t \in \mathcal{V}$ calculates a message $m_{ts}^n(x_s)$ to be sent to each neighboring node $s \in \Gamma(t)$:

$$m_{ts}^n(x_s) = \alpha \int_{x_t} \psi_{s,t}(x_s,x_t)\,\psi_t(x_t,y_t) \times \prod_{u\in\Gamma(t)\setminus s} m_{ut}^{n-1}(x_t)\,dx_t \qquad (2)$$

Here, $\alpha$ denotes an arbitrary proportionality constant. At any iteration, each node can produce an approximation $\hat{p}^n(x_s \mid y)$ to the marginal distribution $p(x_s \mid y)$ by combining the incoming messages with the local observation:

$$\hat{p}^n(x_s \mid y) = \alpha \psi_s(x_s,y_s) \prod_{t\in\Gamma(s)} m_{ts}^n(x_s) \qquad (3)$$

For tree–structured graphs, the approximate marginals, or beliefs, $\hat{p}^n(x_s \mid y)$ will converge to the true marginals $p(x_s \mid y)$ once the messages from each node have propagated to every other node in the graph.

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 $\hat{p}^n(x_s \mid y)$ will *not* converge to the true marginals. In many applications, however, the resulting loopy BP algorithm exhibits excellent empirical performance [7, 20]. Recently, several theoretical studies have provided insight into the approximations made by loopy BP, partially justifying its application to graphs with cycles [21, 23, 24].

### 2.2. Nonparametric Representations

For graphical models with continuous hidden variables, analytic evaluation of the BP update integral (2) is often intractable. Instead, we represent the resulting message nonparametrically as a kernel density estimate [18]. Let $\mathcal{N}(x; \mu, \Lambda)$ denote a normalized Gaussian density with mean $\mu$ and covariance $\Lambda$, evaluated at $x$. An $M$ component mixture approximation of $m_{ts}(x_s)$ takes the form

$$m_{ts}(x_s) = \sum_{i=1}^{M} w_s^{(i)} \mathcal{N}\Big(x_s; \mu_s^{(i)}, \Lambda_s\Big) \tag{4}$$

where $w_s^{(i)}$ is the weight associated with the $i^{th}$ kernel mean $\mu_s^{(i)}$, and $\Lambda_s$ is a bandwidth or smoothing parameter. The weights are normalized so that $\sum_{i=1}^{M} w_s^{(i)} = 1$. Other kernel functions may be used [18], but in this paper we consider only mixtures of diagonal–covariance Gaussians.

In the following section, we describe stochastic methods for determining the kernel centers $\mu_s^{(i)}$ and associated weights $w_s^{(i)}$. The resulting nonparametric representation is only meaningful when the message $m_{ts}(x_s)$ is finitely integrable.[1] To guarantee this, we assume that

$$\begin{aligned} \int_{x_s} \psi_{s,t}(x_s, x_t = \bar{x})\, dx_s < \infty \qquad & \forall\, (s,t) \in \mathcal{E} \\ \int_{x_s} \psi_s(x_s, y_s = \bar{y})\, dx_s < \infty \qquad & \forall\, s \in \mathcal{V} \end{aligned} \tag{5}$$

A simple induction argument then shows that all messages are normalizable. Heuristically, equation (5) requires each potential to be "informative," so that observing one variable constrains the likely locations of the other. In most applications, this assumption is easily satisfied by constraining all variables to a (possibly large) bounded range.

## 3. Nonparametric Message Updates

Conceptually, the BP update equation (2) naturally decomposes into two stages. First, the message product $\psi_t(x_t, y_t) \prod_u m_{ut}^{n-1}(x_t)$ combines information from neighboring nodes with the local evidence $y_t$, producing a function summarizing all available knowledge about the hidden variable $x_t$. We will refer to this summary as a likelihood function, even though this interpretation is only strictly correct for an appropriately factorized tree–structured graph. Second, this likelihood function is combined with the compatibility potential $\psi_{s,t}(x_s, x_t)$, and then integrated to produce likelihoods for $x_s$. The nonparametric belief propagation (NBP) algorithm stochastically approximates these two stages, producing consistent nonparametric representations of the messages $m_{ts}(x_s)$.

[1]Probabilistically, on tree–structured graphs BP messages are likelihood functions $m_{ts}(x_s) \propto p(y = \bar{y} \mid x_s)$, not conditional densities, and are *not* necessarily integrable (e.g., when $x_s$ and $y$ are independent).

Approximate marginals $\hat{p}(x_s \mid y)$ may then be determined from these messages by applying the following section's stochastic product algorithm to equation (3).

### 3.1. Message Products

For now, assume that the potential functions of equation (1) are weighted Gaussian mixtures (such potentials arise naturally from learning–based approaches to model identification [7]). The product of $d$ Gaussian densities is itself Gaussian, with mean and covariance given by

$$\begin{aligned} \prod_{j=1}^{d} \mathcal{N}(x; \mu_j, \Lambda_j) \propto \mathcal{N}\big(x; \bar{\mu}, \bar{\Lambda}\big) \\ \bar{\Lambda}^{-1} = \sum_{j=1}^{d} \Lambda_j^{-1} \qquad \bar{\Lambda}^{-1}\bar{\mu} = \sum_{j=1}^{d} \Lambda_j^{-1}\mu_j \end{aligned} \tag{6}$$

Thus, a BP update operation which multiplies $d$ Gaussian mixtures, each containing $M$ components, will produce a Gaussian mixture with $M^d$ components. The weight $\bar{w}$ associated with product mixture component $\mathcal{N}\big(x; \bar{\mu}, \bar{\Lambda}\big)$ is

$$\bar{w} \propto \frac{\prod_{j=1}^{d} w_j \mathcal{N}(x; \mu_j, \Lambda_j)}{\mathcal{N}\big(x; \bar{\mu}, \bar{\Lambda}\big)} \tag{7}$$

where $\{w_j\}_{j=1}^{d}$ are the weights associated with the input Gaussians. Note that equation (7) produces the same value for any choice of $x$. Also, in various special cases, such as when all input Gaussians have the same variance $\Lambda_j = \Lambda$, computationally convenient simplifications are possible.

Integration of Gaussian mixtures is straightforward, so in principle each BP message update could be performed exactly using equations (6,7). In practice, however, approximations are required to avoid exponentially large numbers of mixture components. Given $d$ input mixtures of $M$ Gaussians, the NBP algorithm approximates their $M^d$ component product by drawing $M$ independent samples.

Direct sampling from this product, achieved by explicitly calculating each of the product component weights (7), requires $\mathcal{O}(M^d)$ operations. The complexity of this sampling is combinatorial: each product component is associated with $d$ labels $\{l_j\}_{j=1}^{d}$, where $l_j$ identifies a kernel in the $j^{th}$ input mixture. Although the joint distribution of the $d$ labels is complex, the conditional distribution of any individual label $l_j$ is simple. In particular, assuming fixed values for $\{l_k\}_{k \neq j}$, equation (7) can be used to sample from the conditional distribution of $l_j$ in $\mathcal{O}(M)$ operations.

Since the conditional distribution of each mixture label is tractable, we may use a Gibbs sampler [8] to draw asymptotically unbiased samples from the product density. Details are provided in Algorithm 1, and illustrated in Figure 2. At each iteration, the labels $\{l_k\}_{k \neq j}$ for $d-1$ of the input mixtures are fixed, and the $j^{th}$ label is sampled from the corresponding conditional density. The newly chosen $l_j$ is then fixed, and another label is updated. This procedure continues for a fixed number of iterations $\kappa$; more iterations lead

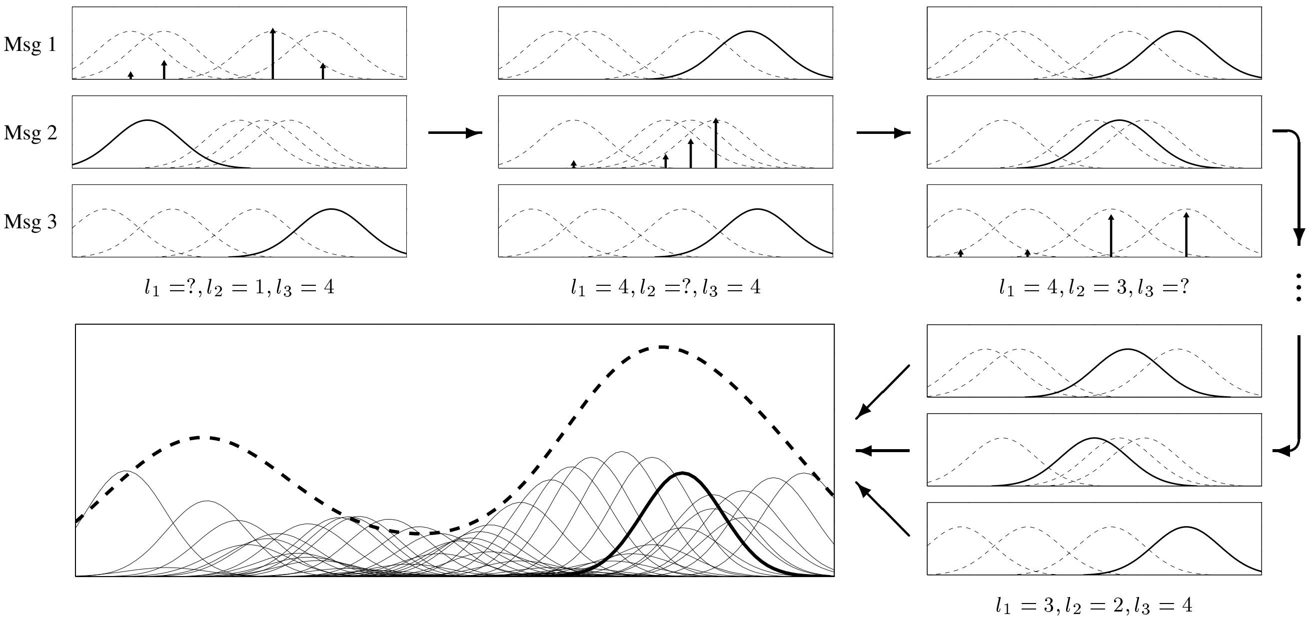



**Figure 2.** *Top row:* **Gibbs sampler for a product of 3 Gaussian mixtures, with 4 kernels each. New indices are sampled according to weights (arrows) determined by the two fixed components (solid). The Gibbs sampler cycles through the different messages, drawing a new mixture label for one message conditioned on the currently labeled Gaussians in the other messages.** *Bottom row:* **After $\kappa$ iterations through all the messages, the final labeled Gaussians for each message (right, solid) are multiplied together to identify one (left, solid) of the $4^3$ components (left, thin) of the product density (left, dashed).**

to more accurate samples, but require greater computational cost. Following the final iteration, a single sample is drawn from the product mixture component identified by the final labels. To draw $M$ (approximate) samples from the product density, the Gibbs sampler requires $\mathcal{O}(d\kappa M^2)$ operations.

Although formal verification of the Gibbs sampler's convergence is difficult, our empirical results indicate that accurate Gibbs sampling typically requires far fewer computations than direct sampling. Note that NBP uses the Gibbs sampling method differently from classic simulated annealing algorithms [8]. In simulated annealing, the Gibbs sampler updates a single Markov chain whose state dimension is proportional to the graph dimension. In contrast, NBP uses many *local* Gibbs samplers, each involving only a few nodes. Thus, although NBP must run more independent Gibbs samplers, for large graphs the dimensionality of the corresponding Markov chains is dramatically smaller.

In some applications, the observation potentials $\psi_t(x_t, y_t)$ are specified by analytic functions. The Gibbs sampler may be adapted to this case using *importance sampling* [5], as shown in Algorithm 2. At each iteration, the weight of the $i^{th}$ kernel label is rescaled by $\psi_t\left(\bar{\mu}^{(i)}, y_t\right)$, the observation likelihood at that kernel's center. Then, the final sample is assigned an importance weight to account for variations of the analytic potential over the kernel's support. This procedure is most effective when $\psi_t(x_t, y_t)$ varies slowly relative to the typical kernel bandwidth.

## 3.2. Message Propagation

The NBP algorithm's second stage propagates each sample from the message product by approximating the belief update integral (2). This stochastic integration requires a decomposition of the pairwise potential $\psi_{s,t}(x_s, x_t)$ which separates its *marginal* influence on $x_t$ from the *conditional* relationship it defines between $x_s$ and $x_t$.

The marginal influence function $\zeta(x_t)$ is determined by the relative weight assigned to *all* $x_s$ values for each $x_t$:

$$\zeta(x_t) = \int_{x_s} \psi_{s,t}(x_s, x_t)\ dx_s \qquad (8)$$

The NBP algorithm accounts for this marginal influence by incorporating $\zeta(x_t)$ into the Gibbs sampler. If $\psi_{s,t}(x_s, x_t)$ is a Gaussian mixture, $\zeta(x_t)$ is easily calculated. Alternately, importance sampling may be used, assuming $\zeta(x_t)$ can be evaluated (or approximated) pointwise. In the common case where pairwise potentials depend only on the difference between their arguments ($\psi_{s,t}(x, \bar{x}) = \psi_{s,t}(x - \bar{x})$), $\zeta(x_t)$ is constant and can be neglected.

To complete the stochastic integration, each sample $x_t^{(i)}$ from the message product is propagated to the neighboring node by sampling $x_s^{(i)} \sim \psi_{s,t}(x_s, x_t^{(i)})$, where equation (5) ensures that $\psi_{s,t}(x_s, x_t^{(i)})$ is normalizable. Finally, having produced a set of independent samples from the output message $m_{ts}(x_s)$, NBP selects a kernel bandwidth to complete the nonparametric density estimate. There are many ways to make this choice; the results in this paper use the compu-

Given $d$ mixtures of $M$ Gaussians, where $\{\mu_j^{(i)}, \Lambda_j^{(i)}, w_j^{(i)}\}_{i=1}^M$ denote the parameters of the $j^{th}$ mixture:

1. For each $j \in [1 : d]$, choose a starting label $l_j \in [1 : M]$ by sampling $p\,(l_j = i) \propto w_j^{(i)}$.
2. For each $j \in [1 : d]$,
   (a) Calculate the mean $\mu^*$ and variance $\Lambda^*$ of the product $\prod_{k \neq j} \mathcal{N}\left(x; \mu_k^{(l_k)}, \Lambda_k^{(l_k)}\right)$ using equation (6).
   (b) For each $i \in [1 : M]$, calculate the mean $\bar{\mu}^{(i)}$ and variance $\bar{\Lambda}^{(i)}$ of $\mathcal{N}(x; \mu^*, \Lambda^*) \cdot \mathcal{N}\left(x; \mu_j^{(i)}, \Lambda_j^{(i)}\right)$. Using any convenient $x$, compute the weight
   $$\bar{w}^{(i)} = w_j^{(i)} \frac{\mathcal{N}\left(x; \mu_j^{(i)}, \Lambda_j^{(i)}\right) \mathcal{N}(x; \mu^*, \Lambda^*)}{\mathcal{N}\left(x; \bar{\mu}^{(i)}, \bar{\Lambda}^{(i)}\right)}$$
   (c) Sample a new label $l_j$ according to $p\,(l_j = i) \propto \bar{w}^{(i)}$.
3. Repeat step 2 for $\kappa$ iterations.
4. Compute the mean $\bar{\mu}$ and variance $\bar{\Lambda}$ of the product $\prod_{j=1}^d \mathcal{N}\left(x; \mu_j^{(l_j)}, \Lambda_j^{(l_j)}\right)$. Draw a sample $\hat{x} \sim \mathcal{N}\left(x; \bar{\mu}, \bar{\Lambda}\right)$.

**Algorithm 1. Gibbs sampler for the product of several Gaussian mixtures.**

Given $d$ mixtures of $M$ Gaussians and an analytic function $f(x)$, follow Algorithm 1 with the following modifications:

2. After part (b), rescale each computed weight by the analytic value at the kernel center: $\bar{w}^{(i)} \leftarrow f(\bar{\mu}^{(i)})\bar{w}^{(i)}$.
5. Assign importance weight $\hat{w} = f(\hat{x})/f(\bar{\mu})$ to the sampled particle $\hat{x}$.

**Algorithm 2. Gibbs sampler for the product of several Gaussian mixtures with an analytic function.**

Given input messages $m_{ut}\,(x_t) = \{\mu_{ut}^{(i)}, \Lambda_{ut}^{(i)}, w_{ut}^{(i)}\}_{i=1}^M$ for each $u \in \Gamma(t) \setminus s$, construct an output message $m_{ts}\,(x_s)$ as follows:

1. Determine the marginal influence $\zeta(x_t)$ using equation (8):
   (a) If $\psi_{s,t}\,(x_s, x_t)$ is a Gaussian mixture, $\zeta(x_t)$ is the marginal over $x_t$.
   (b) For analytic $\psi_{s,t}\,(x_s, x_t)$, determine $\zeta(x_t)$ by symbolic or numeric integration.
2. Draw $M$ independent samples $\{\hat{x}_t^{(i)}\}_{i=1}^M$ from the product $\zeta(x_t)\psi_t\,(x_t, y_t) \prod_u m_{ut}\,(x_t)$ using the Gibbs sampler of Algorithms 1-2.
3. For each $\{\hat{x}_t^{(i)}\}_{i=1}^M$, sample $\hat{x}_s^{(i)} \sim \psi_{s,t}(x_s, x_t = \hat{x}_t^{(i)})$:
   (a) If $\psi_{s,t}\,(x_s, x_t)$ is a Gaussian mixture, $\hat{x}_s^{(i)}$ is sampled from the conditional of $x_s$ given $\hat{x}_t^{(i)}$.
   (b) For analytic $\psi_{s,t}\,(x_s, x_t)$, importance sampling or MCMC methods may be used as appropriate.
4. Construct $m_{ts}\,(x_s) = \{\mu_{ts}^{(i)}, \Lambda_{ts}^{(i)}, w_{ts}^{(i)}\}_{i=1}^M$:
   (a) Set $\mu_{ts}^{(i)} = \hat{x}_s^{(i)}$, and $w_{ts}^{(i)}$ equal to the importance weights (if any) generated in step 3.
   (b) Choose $\{\Lambda_{ts}^{(i)}\}_{i=1}^M$ using any appropriate kernel size selection method (see [18]).

**Algorithm 3. NBP update of the nonparametric message $m_{ts}\,(x_s)$ sent from node $t$ to node $s$ as in equation (2).**

tationally efficient "rule of thumb" heuristic [18].

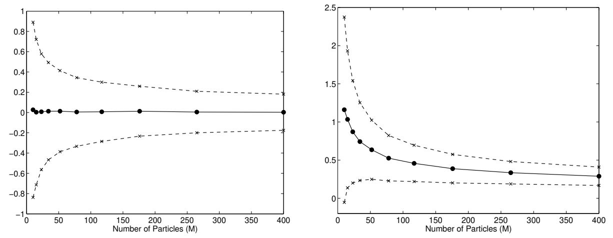
The NBP message update procedure is summarized in Algorithm 3. Note that parts of this algorithm may be simplified in certain special cases. For example, Isard [9] has proposed a related generalization of particle filters in which the sampling and bandwidth selection of steps 3-4 are replaced by a deterministic kernel placement. When each pairwise potential contains only a few kernels, this modification is an excellent way to reduce biases inherent in kernel density estimates (see Section 4). However, for graphical models with multimodal potentials containing large numbers of kernels (as in Section 5), a more sophisticated propagation step (like that provided by NBP) is necessary.

# 4. Gaussian Graphical Models

Gaussian graphical models provide one of the few continuous distributions for which the BP algorithm may be implemented exactly [23]. For this reason, Gaussian models may be used to test the accuracy of the nonparametric approximations made by NBP. Note that we cannot hope for NBP to outperform algorithms (like Gaussian BP) designed to take advantage of the linear structure underlying Gaussian problems. Instead, our goal is to verify NBP's performance in a situation where exact comparisons are possible.

We examine NBP's performance on a $5 \times 5$ nearest–neighbor grid (as in Figure 1) with randomly chosen inhomogeneous potentials. However, qualitatively similar results have also been observed on tree–structured graphs. To form the test model, we drew samples from the single correlated Gaussian defining each of the graph's pairwise potentials, and then formed a nonparametric density estimate based on these samples. Although the NBP algorithm could have directly used the original correlated potentials, sample–based models are a closer match for the information available in many vision applications (see Section 5).

For each node $s \in \mathcal{V}$, Gaussian BP converges to a steady–state estimate of the marginal mean $\mu_s$ and variance $\sigma_s^2$ after about 15 iterations. To evaluate NBP, we performed 15 iterations of the NBP message updates using several different particle set sizes $M \in [10, 400]$. We then found the marginal mean $\hat{\mu}_s$ and variance $\hat{\sigma}_s^2$ estimates implied by the final NBP density estimates. For each tested particle set size, the NBP comparison was repeated 100 times.




Mean ($\tilde{\mu}_s$) Variance ($\tilde{\sigma}_s^2$)

**Figure 3. NBP performance on a $5 \times 5$ grid with Gaussian potentials. Plots show the mean (solid) and standard deviation (dashed) of the normalized error measures of equation (9), for different particle set sizes $M$.**

Using the data from each NBP trial, we computed the error in the mean and variance estimates, normalized so each node behaved like a unit–variance Gaussian:

$$\tilde{\mu}_s = \frac{\hat{\mu}_s - \mu_s}{\sigma_s} \qquad \tilde{\sigma}_s^2 = \frac{\hat{\sigma}_s^2 - \sigma_s^2}{\sqrt{2}\sigma_s^2} \tag{9}$$

Figure 3 shows the mean and variance of these error statistics, across all nodes and trials, for different particle set sizes $M$. The NBP algorithm always provides unbiased mean estimates, but overly large variances. This bias, which decreases as more particles are used, is due to the smoothing inherent in kernel–based density estimates. As expected for samples drawn from Gaussian distributions, the standard deviation of both error measures falls as $M^{-1/2}$.

## 5. Component–Based Face Models

Like particle filters, the NBP algorithm has a wide range of potential computer vision applications. Previously, NBP was used to estimate dense stereo depth maps [19]. In this section, we use NBP to infer relationships between PCA coefficients in a component–based model of the human face. The proposed model extends the approaches of [6, 14, 22] to estimate not only the location, but also the appearance of occluded parts. Parts–based local appearance models share many features with the articulated models often used for visual tracking. However, they lack the implementational overhead associated with modern person trackers [17], for which we think NBP would also be well suited.

### 5.1. Model Construction

In this section, we use training data to construct a nonparametric graphical prior for the location and appearance of five different facial features (see Figure 5). A 10 dimensional linear basis for each feature's appearance is determined via principal components analysis (PCA) [14], as described below. The hidden variable at each of the graphical model's five nodes is then defined to be a 12 dimensional (10 PCA coefficients plus position) representation of the corresponding feature. We assume that the face's orientation and scale are known, although the model could be easily extended to estimate other alignment parameters [4].

**Figure 4. Two of the 94 training subjects from the AR face database. Each was photographed in these four poses.**

(a) (b) (c)

**Figure 5. PCA–based facial component model. (a) Masks for each of the five features. Note that the two mouth masks overlap. (b) Mean features. (c) Graphical prior relating each component's position and PCA coefficients.**

Our appearance model is based on training images from the AR face database [13]. For each of 94 people, we chose four poses showing a range of expressions and lighting conditions (see Figure 4). Five manually selected feature points (eyes, nose and mouth corners) were used to align each image and position the feature masks shown in Figure 5(a). We then found a 10 dimensional PCA basis for each feature [14], producing the mean features of Figure 5(b).

Using the PCA representation of each training subject, we determined a kernel–based nonparametric density estimate of the joint probability of those pairs of facial features which are adjacent in the graph of Figure 5(c). Figure 6 shows several marginalizations of these 20 dimensional densities, each relating a pair of PCA coefficients (e.g., the first nose and second left mouth coefficients). Clearly, simple Gaussian approximations would obscure most of this data set's structure. Finally, we approximate the true pairwise potentials relating neighboring PCA coefficients by the corresponding kernel density estimates [7]. Differences between feature positions are modeled by a Gaussian density, with mean and variance estimated from the training set.

### 5.2. Estimation of Occluded Features

In this section, we apply the graphical model of Figure 5 to the simultaneous location and reconstruction of partially occluded faces. Given an input image, we first identify the region most likely to contain a face using a standard eigenface detector [14] trained on partial face images. This step

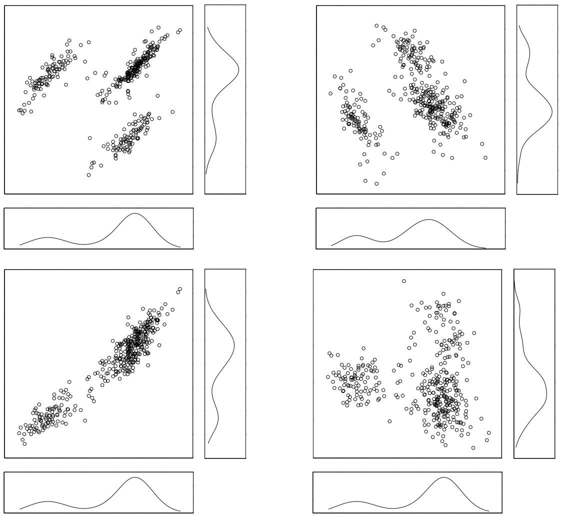


**Figure 6. Empirical joint densities of four different pairs of PCA coefficients. Each plot shows the corresponding marginal distributions along the bottom and right edges. Note the multimodal, non–Gaussian relationships.**

helps to prevent spurious detection of background detail by the individual components. For each node, we scan this region with the feature mask, producing the best PCA reconstruction $\hat{y}$ of each pixel window $y$. The observation potential is created by defining a Gaussian mixture component, with mean $\hat{y}$ and weight $\exp\left\{-||y-\hat{y}||^2/2\sigma^2\right\}$, for each $y$. To allow for outliers due to occlusion, the observation potential is augmented by a zero mean, high–variance Gaussian weighted to represent 20% of the total likelihood.

We tested the NBP algorithm on images of individuals not found in the training set. Each message was represented by $M = 100$ particles, and the Gibbs sampler used $\kappa = 100$ iterations. Total computation time for each image was a few minutes on a Pentium 4 workstation. Due to the high dimensionality of the variables in this model, and the presence of the occlusion process, discretization is intractable. Therefore, we instead compare NBP's estimates to the closed form solution obtained by fitting a single Gaussian to each of the nonparametric prior and observation potentials.

Figure 7 shows inference results for two images of a man concealing his mouth. In one image he is smiling, while in the other he is not. Using the relationships between eye and mouth shape learned from the training set, NBP correctly infers the concealed mouth's expression. In contrast, the Gaussian approximation distorts the relationships shown in Figure 6, and produces results which are indistinguishable from the mean mouth shape. Note that these results were obtained in an unsupervised fashion, without any manual labeling of the training image expressions.

Figure 8 shows inference results for two images of a woman concealing one eye. In one image, she is seen under normal illumination, while in the second she is illuminated from the left by a bright light. In both cases, the structure of the concealed eye mirrors the visible eye. In addition, NBP correctly modifies the illumination of the occluded eye to match the intensity of the corresponding mouth corner. This example shows NBP's ability to integrate information from multiple nodes, producing globally consistent estimates.

## 6. Discussion

We have developed a nonparametric sampling–based variant of the belief propagation algorithm for graphical models with continuous, non–Gaussian random variables. Our parts–based facial modeling results demonstrate NBP's ability to infer sophisticated relationships from training data, and suggest that it may prove useful in more complex visual tracking problems. We hope that NBP will allow the successes of particle filters to be translated to many new computer vision applications.

## Acknowledgments

The authors would like to thank Ali Rahimi for his help with the facial appearance modeling application. This research was supported in part by ONR Grant N00014-00-1-0089, by AFOSR Grant F49620-00-1-0362, and by an ODDR&E MURI funded through ARO Grant DAAD19-00-1-0466. E. Sudderth was partially funded by an NDSEG fellowship.

## References

[1] D. L. Alspach and H. W. Sorenson. Nonlinear Bayesian estimation using Gaussian sum approximations. *IEEE Trans. AC*, 17(4):439–448, Aug. 1972.

[2] J. M. Coughlan and S. J. Ferreira. Finding deformable shapes using loopy belief propagation. In *ECCV*, pages 453–468, 2002.

[3] A. P. Dawid, U. Kjærulff, and S. L. Lauritzen. Hybrid propagation in junction trees. In *Adv. Intell. Comp.*, pages 87–97, 1995.

[4] F. De la Torre and M. J. Black. Robust parameterized component analysis: Theory and applications to 2D facial modeling. In *ECCV*, pages 653–669, 2002.

[5] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[6] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. Submitted to *IJCV*, 2003.

[7] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low–level vision. *IJCV*, 40(1):25–47, 2000.

[8] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. PAMI*, 6(6):721–741, Nov. 1984.

[9] M. Isard. PAMPAS: Real–valued graphical models for computer vision. In *CVPR*, 2003.

[10] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *ECCV*, pages 343–356, 1996.

[11] D. Koller, U. Lerner, and D. Angelov. A general algorithm for approximate inference and its application to hybrid Bayes nets. In *UAI 15*, pages 324–333, 1999.

[12] S. L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, 1996.

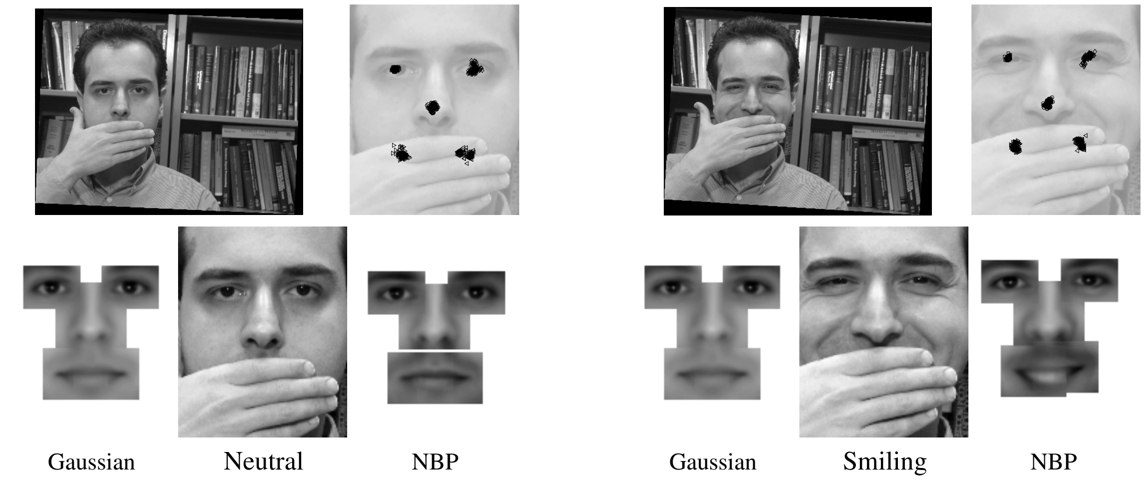



**Figure 7. Simultaneous estimation of location (top row) and appearance (bottom row) of an occluded mouth. Results for the Gaussian approximation are on the left of each panel, and for NBP on the right. By observing the squinting eyes of the subject (right), and exploiting the feature interrelationships represented in the trained graphical model, the NBP algorithm correctly infers that the occluded mouth should be smiling. A parametric Gaussian model doesn't capture these relationships.**

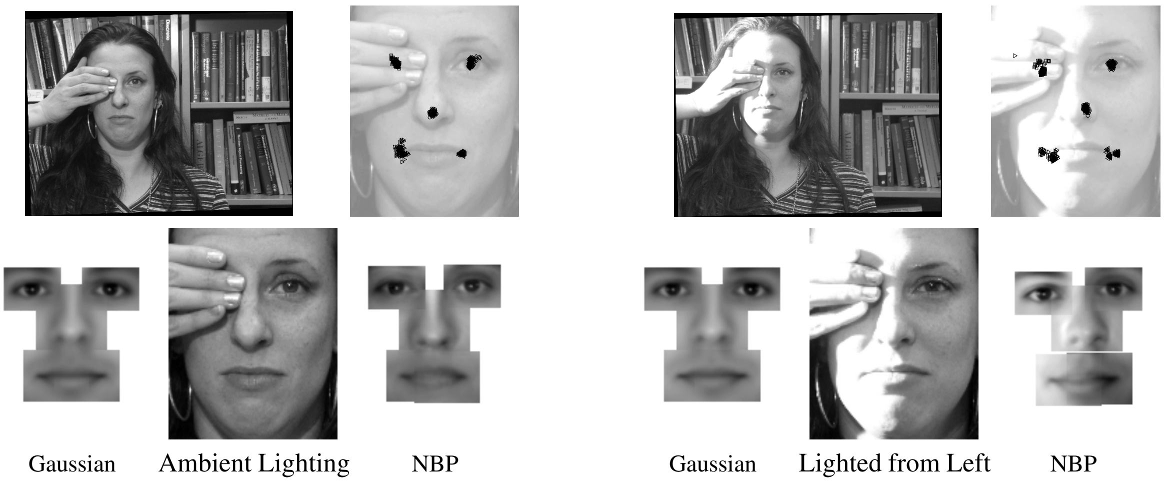



**Figure 8. Simultaneous estimation of location (top row) and appearance (bottom row) of an occluded eye. NBP combines information from the visible eye and mouth to determine both shape and illumination of the occluded eye, correctly inferring that the left eye should brighten under the lighting conditions shown at right.**


[13] A. M. Martínez and R. Benavente. The AR face database. Technical Report 24, CVC, June 1998.

[14] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Trans. PAMI*, 19(7):696–710, July 1997.

[15] O. Nestares and D. J. Fleet. Probabilistic tracking of motion boundaries with spatiotemporal predictions. In *CVPR*, pages 358–365, 2001.

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, San Mateo, 1988.

[17] H. Sidenbladh and M. J. Black. Learning the statistics of people in images and video. *IJCV*, 2003. To appear.

[18] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, London, 1986.

[19] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. Technical Report 2551, MIT Laboratory for Information and Decision Systems, Oct. 2002.

[20] J. Sun, H. Shum, and N. Zheng. Stereo matching using belief propagation. In *ECCV*, pages 510–524, 2002.

[21] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree–based reparameterization analysis of sum–product and its generalizations. *IEEE Trans. IT*, 49(5), May 2003.

[22] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *ECCV*, pages 18–32, 2000.

[23] Y. Weiss and W. T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Comp.*, 13:2173–2200, 2001.

[24] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report 2002-35, MERL, Aug. 2002.