Efficient Belief Propagation for Higher Order Cliques Using Linear Constraint Nodes

Brian Potetz [*], Tai Sing Lee

Department of Computer Science & Center for the Neural Basis of Cognition
Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213

Abstract

Belief propagation over pairwise connected Markov Random Fields has become a widely used approach, and has been successfully applied to several important computer vision problems. However, pairwise interactions are often insufficient to capture the full statistics of the problem. Higher-order interactions are sometimes required. Unfortunately, the complexity of belief propagation is exponential in the size of the largest clique. In this paper, we introduce a new technique to compute belief propagation messages in time linear with respect to clique size for a large class of potential functions over real-valued variables. We discuss how this technique can be generalized to still wider classes of potential functions at varying levels of efficiency. Also, we develop a form of nonparametric belief representation specifically designed to address issues common to networks with higher-order cliques and also to the use of guaranteed-convergent forms of belief propagation.

To illustrate these techniques, we perform efficient inference in graphical models where the spatial prior of natural images is captured by 2×2 cliques. This approach shows significant improvement over the commonly used pairwise-connected models, and may benefit a variety of applications using belief propagation to infer images or range images, including stereo, shape-from-shading, image-based rendering, segmentation, and matting.

Key words: Belief propagation, Higher order cliques, Non-pairwise cliques, Factor graphs, Continuous Markov Random Fields
PACS:

[*] Corresponding author.
Email addresses: `bpotetz@cs.cmu.edu` (Brian Potetz), `tai@cnbc.cmu.edu` (Tai Sing Lee).

1 Introduction

In the past few decades, the application of probabilistic models for solving computer vision problems has lead to significant advances. Many of these probabilistic models can be simplified by factoring large probability distributions using graphical models. Unfortunately, statistical inference in arbitrary factorized distributions is still NP-hard [1]. Recently, the method of loopy belief propagation has been shown to produce excellent results in several real-world computer vision problems [2–7]. However, this method has some drawbacks. The most serious is that the running time of belief propagation is exponential in the size of the largest graph clique. This means that for problems with many labels or real-valued variables, graphical representations are typically limited to pairwise interactions between variables. Unfortunately, for many problems in computer vision, pairwise interactions fail to capture the rich statistical distribution of the problem domain. For example, natural images exhibit rich higher-order statistics that cannot be captured by pairwise connected Markov Random Fields (MRFs). In section 3, we introduce a new technique to compute belief propagation messages in time linear with respect to clique size that works for a large class of potential functions without resorting to approximation. In section 7, we further improve on the efficiency and performance of belief propagation by presenting a nonparametric, particle-like representation of belief propagation messages that is simultaneously compatible with higher-order non-pairwise interactions and also with recent extensions to belief propagation that guarantee convergence [8].

These advancements allow us to efficiently solve inference problems that were previously unavailable to belief propagation. In section 8, we show that a prior model of natural images using 2×2 MRF cliques strongly outperforms pairwise-connected models. The ability to use more accurate models of image or range image priors has the potential to significantly aid the performance of several computer vision applications, including stereo [3], photometric stereo [4], shape-from-shading [9], image-based rendering [10], segmentation, and matting [7].

1.1 Methods of Statistical Inference

The problem of statistical inference is central to artificial intelligence and to computation in general. Statistical inference is the problem of finding the expected value $E[\vec{X}]$ of a multivariate distribution $P(\vec{X})$ (also known as the Minimum Mean-Squared Error point estimate, or MMSE), or, alternatively, finding the most likely point in a distribution (also known as the Maximum a Posteriori point estimate, or MAP). Successful methods of statistical inference can allow us to locate likely values of unknown things, such as the likely transcription of a spoken audio recording $P(text|audio)$, or the likely 3D shape of an object in a photograph $P(shape|image)$. Unfortunately, in the general case, finding the MAP or MMSE point estimate of a distribution is NP-Hard [1]. Thus, for many real-world problems with large, complex distributions, approximate methods must be used to estimate the MAP or MMSE points of a distribution.

One simple approach is to use a gradient descent or related methods on the posterior distribution to find the MAP estimate. The problem is that gradient descent can easily become stuck in local minima. This is a serious problem for all but the most simple posterior distributions. A related approach is Markov chain Monte Carlo (MCMC) sampling. In this family of algorithms, we seek to approximate the posterior distribution by generating a set of samples from this distribution. Sampling can be used to compute a MAP estimate by simply selecting the sample with the highest probability according to the model probability distribution. MMSE and other estimators can also be approximated from a sample list. Unfortunately, MCMC sampling can also be prohibitively slow, especially in high dimensional problems. Additionally, it is often difficult to determine if the algorithm has converged, or if some important portion of the state space has not yet been explored by the stochastic sampling algorithm.

One key insight that has been greatly helpful for statistical inference is to exploit local structure within a probability distribution. Specifically, many probability distributions can be *factorized*, or represented as a product of *potential functions*, each of which ranges over only a small subset of variables of the problem space $\vec{X}$:

$$p(\vec{X}) \propto \prod \phi_i(\vec{x_i}) \qquad \vec{x_i} \subset \vec{X} \tag{1}$$

Such probability distributions are often represented in the form of a *factor graph*. A factor graph is a bipartite graph in which each potential function $\phi_i(\vec{x_i})$ is represented by a factor node f, which is connected to one variable node v for each element of the vector $\vec{x}_i$. Example factor graphs are depicted in figures 2 and 3.

One popular method of statistical inference that exploits factorized probability distributions is *graph cuts* [11]. Graph cuts are a method of estimating the MAP point estimate of a factorized distribution that is based on the max-flow min-cut theorem for graphs. For graph cuts to work, the potential functions in equation 1 must meet a set of constraints [12]. Specifically, each potential function must be *regular*. For potential functions of two variables, this means that for any three variable states x_1, x_2, and α, we must have

$$\phi(x_1, x_2) + \phi(\alpha, \alpha) \geq \phi(\alpha, x_2) + \phi(x_1, \alpha) \tag{2}$$

Intuitively, regular potential functions must encourage its associated variables to be equal. Potential functions of three or more variables have similar constraints [13]. These constraints are compatible with many simple models of spatial priors for images [14]. However, many real-world applications require potential functions that violate these constraints, such as the Lambertian constraint in shape-from-shading [9], hard linear constraints of the type discussed in section 3.3, or the spatial priors of 3D shape, which include planar surfaces [15]. Graph cuts was originally designed for inference of binary random variables, but was extended to multivariate distributions

Another popular method for statistical inference for factorized distributions is *belief propagation*. Belief propagation can be used to estimate either the MAP or the MMSE of a distribution. Belief propagation works for with arbitrary potential functions, including non-regular potentials that are not available to graph cuts. Belief propagation has been used with great success in a variety of applications [16,17,2–4]. Point estimates obtained using belief propagation typically outperform gradient descent significantly, and belief propagation can succeed in cases where gradient descent is overwhelmed by local suboptimal extrema [9] Perhaps the most serious difficulty with using belief propagation is that it is slow for factors with many variables. Specifically, belief propagation requires computing messages from each factor node to each of its neighbors in the factor graph; each of these messages requires computation that is exponential in the number of neighbors of the factor node (this is also known as the clique size, equal to the number of variables in $\vec{x_i}$). In this paper, we introduce methods that reduce the amount of computation from exponential to linear in the number of neighbors. This computational shortcut makes efficient many inference tasks that were previously inaccessible to belief propagation.

2 Belief Propagation

Belief propagation is a method for estimating the single-variate marginals of a multivariate probability distribution of the form:

$$p(\vec{X}) \propto \prod \phi_i(\vec{x_i}) \qquad \vec{x_i} \subset \vec{X} \tag{3}$$

As mentioned above, factorized probability distributions are often represented in the form of a factor graph, such as the ones shown in figures 2 and 3. Sum-product belief propagation estimates the marginals $b(x_i) = \int_{X \backslash x_i} p(\vec{x}) d\vec{x}$ by iteratively computing *messages* along each edge of the graph according to the equations:

$$m^t_{i\to f}(x_i) = \prod_{g\in\mathcal{N}(i)\backslash f} m^{t-1}_{g\to i}(x_i) \tag{4}$$

$$m^t_{f\to i}(x_i) = \int_{\vec{x}_{\mathcal{N}(f)\backslash i}} \phi_f\left(\vec{x}_{\mathcal{N}(f)}\right) \prod_{j\in\mathcal{N}(f)\backslash i} m^t_{j\to f}(x_j)\, d\vec{x} \tag{5}$$

$$b^t_i(x_i) \propto \prod_{g\in\mathcal{N}(i)} m^t_{g\to i}(x_i) \tag{6}$$

where f and g are factor nodes, i and j are variable nodes, and $\mathcal{N}(i)$ is the set of neighbors of node i (see [18,19,8] for further details on classic belief propagation). The integrand of equation 5 sums over a function whose range is of dimensionality $\mathcal{N}(i)-1$, and the integral must be evaluated for each value of x_i. At each iteration of belief propagation, messages $m^t_{i\to f}(x_i)$ are passed from variable nodes to factor nodes, and messages $m^t_{f\to i}(x_i)$ are passed from factor nodes to variable nodes.

In equation 6, $b_i(x_i)$ is the estimated marginal of variable i. The expected value of $\vec{X}$ can be computed by finding the mean of each marginal. This is equivalent to the finding minimum mean-squared error (MMSE) point estimate of the distribution in equation 3. If the maximally likely value of $\vec{X}$ is desired (also known as the maximum a posteriori (MAP) point estimate), then the integrals of equation 5 are replaced by supremas. This is known as max-product belief propagation (rather than sum-product belief propagation).

When the underlying factor graph is a tree, one iteration of sum-product belief propagation is guaranteed to compute the correct marginals [19]. If the factor graph contains loops, the messages must be updated iteratively. This is known as *loopy belief propagation* (LBP). Loopy belief propagation is not guaranteed to converge, and for a long time, little was known about the quality of the approximation when convergence was reached. Loopy belief propagation originally became popular as it was applied successfully to a variety of applications, achieving outstanding empirical results in a number of fields [16,17,2–4]. Later, several theoretical results demonstrated that belief propagation could be expected to achieve high quality approximations in a variety of circumstances [20,21]. More recently, it was shown that when sum-product belief propagation converges, the resulting marginals form a minima of the Bethe free energy, a quantity from statistical physics which can be thought of as an approximate measure of the distance between a multivariate probability distribution and a set of marginals [19]. This connection between Bethe free energy and loopy belief propagation provided a sound theoretical justification for the application of belief propagation to networks with loops. Furthermore, this discovery has lead to new belief propagation methods that minimize Bethe free energy directly, and are guaranteed to converge [22,8]. The computational shortcuts we describe in section 3 are compatible with these convergent variants of be-

Fig. 1. Insert Figure 1 about here.

lief propagation, and in section 6, we will discuss these convergent methods in greater detail.

For continuous random variables, the integrals of equation 5 typically cannot be computed or represented analytically. In these cases, the beliefs $b_i(x_i)$ and messages $m_{i \to f}(x_i)$ are often approximated by discrete histograms. When messages are represented by histograms, the integrand of equation 5 is replaced by a summand:

$$m^t_{f \to i}(x_i) = \sum_{\vec{x}_{\mathcal{N}(f) \setminus i}} \phi_f \left(\vec{x}_{\mathcal{N}(f)} \right) \prod_{j \in \mathcal{N}(f) \setminus i} m^t_{j \to f}(x_j) \tag{7}$$

and the algorithm proceeds as before. In the next several sections, we will assume that messages are represented using discrete histograms. We will continue to write the belief propagation equations in continuous form, so that the error of discretization can be postponed for as long as possible. In section 7, we will discuss alternate methods of message representation, their implications for belief propagation inference in networks with higher order cliques, and ways of minimizing discretization error.

3 Efficient Belief Propagation

Belief propagation has been applied successfully to a variety of computer vision problems [2–4]. However, for many computer vision problems, belief propagation is prohibitively slow. For discrete message representations such as histograms, the computational bottleneck of belief propagation is the high-dimensional integrand in equation 5. Performing this integrand has a complexity of $\mathcal{O}(M^N)$, where M is the number of possible labels for each variable, and N is the number of neighbors of f. In many computer vision problems, variables are continuous or have many labels. In these cases, applications of belief propagation have nearly always been restricted to pairwise connected Markov Random Fields, where each potential function in equation 3 depends on only two variable nodes [2,3]. However, pairwise connected models are often insufficient to capture the full complexity of the joint distribution of the problem. In this section, we describe methods to efficiently compute belief propagation messages over continuous random variables for a wide range of higher-order (non-pairwise) potential functions.

Consider potential functions of the form

$$\phi(\vec{x}) = g(\vec{x} \cdot \vec{v}) \tag{8}$$

where $\vec{x}$ and $\vec{v}$ are vectors of length N. Factor nodes of this form will be referred to as Linear Constraint Nodes (LCNs). Normally, computing messages from such factor nodes takes $\mathcal{O}(M^N)$ time. Here, we show that, using a change of variables, this computation can be done in $\mathcal{O}(NM^2)$ time. For notational simplicity, we illustrate this using $N = 4$, although the method extends easily to arbitrary N. For shorthand, let $M_i \equiv m_{f\to i}$ and $m_i \equiv m_{i\to f}$ Then we have:

$$\begin{aligned}
M_1(x_1) &= \iiint g(v_1x_1 + v_2x_2 + v_3x_3 + v_4x_4) \\
&\qquad m_2(x_2)m_3(x_3)m_4(x_4)\,dx_2\,dx_3\,dx_4 && (9)\\
&= \iiint J\,g(v_1x_1 + y_2)\,m_2(\frac{y_2 - y_3}{v_2}) \\
&\qquad m_3(\frac{y_3 - y_4}{v_3})\,m_4(\frac{y_4}{v_4})\,dy_2\,dy_3\,dy_4 && (10)\\
&\propto \int g(v_1x_1 + y_2)\Bigg(\int m_2(\frac{y_2 - y_3}{v_2}) \\
&\qquad \Bigg(\int m_3(\frac{y_3 - y_4}{v_3})m_4(\frac{y_4}{v_4})dy_4\Bigg)dy_3\Bigg)dy_2 && (11)
\end{aligned}$$

where J is the (constant) Jacobian corresponding to the change of variables. Since belief propagation messages are only defined up to a constant for most variations of LBP, the Jacobian can be safely ignored in this case. Here we have used the change of variables:

$$\begin{aligned}
y_4 &= v_4x_4 && (12)\\
y_3 &= v_3x_3 + y_4 && (13)\\
y_2 &= v_2x_2 + y_3 && (14)\\
J &= 1/(v_2v_3v_4) && (15)
\end{aligned}$$

This allows us to perform each integrand one at a time. Since each of the $N-1$ integrands depend only on two variables, each can be computed in $\mathcal{O}(M^2)$ time. In section 7.3, we provide more technical details on how to compute these integrals for histogram-based message representations, and show that the method of computing messages described here not only results in a significant computational speed-up, but also lowers the discretization error.

The transformation of variables used above works for any vector $\vec{v}$. However, there are many possible transformations. Clever choice of the transformation of variables may allow one to reuse intermediate computations during the computation of other messages, or to embed additional nonlinear potential functions of pairs of variables y_i and y_{i+1} at no extra computational cost. The choice of transformation of variables is discussed further in section 5.

If $v_i = \pm 1$ for all i, and messages are represented as uniform-width histograms, then each integrand in equation 11 can be reduced to a $\mathcal{O}(M \log M)$ computation using discrete Fourier transforms as in [23]. Although we describe our approach for sum-product belief propagation, the same approach is valid for max-product belief propagation. For max-product belief propagation, each maximal in equation 11 can be closely approximated in linear time using the distance transform methods described in [23].

3.2 Linear Constraint Nodes and Projection Pursuit Density Estimation Methods

Systems of linear constraint nodes, of the form

$$P(\vec{x}) \approx \tilde{P}(\vec{x}) = \prod_{k=1}^{K} g_k(\vec{x} \cdot \vec{v}_k) \tag{16}$$

have been very successful in approximating multivariate, continuous probability distributions $P(\vec{x})$. Projection pursuit density estimation [24], Minimax Entropy and FRAME [25,26], Products of Experts [27], and Fields of Experts [28] all work by approximating distributions $P(\vec{x})$ as products of linear constraint nodes (as in equation 16). Previously, performing inference over these graphical models typically required using gradient descent or related methods. These approaches often struggled with local maxima. In section 8, we will show how the shortcut introduced in section 3.1 allows us to perform inference in Fields of Experts using belief propagation. Our results significantly outperform gradient descent based methods of optimization.

Products of linear potential functions have several attractive features that have lead to their success. Their factorized nature simplifies the problem of learning parameters, and several powerful methods for learning parameters g_k and $\vec{v}_k$ have been developed [27,29–31]. Additionally, systems of linear potential functions as in equation 16 are members of the exponential family of probability density models [25]. One consequence of this is that when the potential functions g_k are learned so as to minimize the KL-divergence between $P(\vec{x})$ and $\tilde{P}(\vec{x})$

$$D_{KL}[P(\vec{x})||\tilde{P}(\vec{x})] = \int P(\vec{x}) \log \frac{P(\vec{x})}{\tilde{P}(\vec{x})} d\vec{x} \tag{17}$$

(or equivalently, learned so as to maximize the log likelihood of the training data), the single-variate marginals of $\tilde{P}(\vec{x})$ projected onto each vector v_k will match those of the target distribution $P(\vec{x})$:

$$\int P(\vec{x})\delta(\vec{x}\cdot\vec{v_k}-\rho)d\vec{x} = \int \tilde{P}(\vec{x})\delta(\vec{x}\cdot\vec{v}-\rho)d\vec{x} \qquad \forall\rho, k \tag{18}$$

Furthermore, of all probability distributions that share this property (those that satisfy equation 18), $\tilde{P}(\vec{x})$ will achieve the maximal possible entropy [25]. Intuitively, this suggests that $\tilde{P}$ makes as few assumptions as possible regarding features $\vec{v}'$ that the model was not trained on.

Finally, we point out that, given enough linear potential functions, the product of those potential functions can approximate any probability distribution or desired nonlinear potential function arbitrarily well. Suppose we allow K to approach infinity. Then equation 16 becomes

$$\log\tilde{P}(\vec{x}) = \int_{|v|=1} g_{\vec{v}}(\vec{x}\cdot\vec{v})d\vec{v} \tag{19}$$

Now consider the Radon transform

$$\mathcal{R}[f(\vec{x})](\rho,\vec{v}) = \int f(\vec{x})\delta(\vec{x}\cdot\vec{v}-\rho)d\vec{x} \tag{20}$$

where $\vec{v}$ is constrained to be of unit norm. The adjoint of the Radon transform [32] has the form

$$\mathcal{R}^{\dagger}[\psi(\rho,\vec{v})](\vec{x}) = \int_{|v|=1} \psi(\vec{x}\cdot\vec{v},\vec{v})\, d\vec{v} \tag{21}$$

The Radon transform is invertible [32], and since the adjoint of an invertible function is itself invertible, equation 21 is also invertible. This means that we can always choose our potential functions $g_{\vec{v}}(\rho)$ in such a way that $\tilde{P}(\vec{x}) = P(\vec{x})$ exactly. Specifically, choosing $g_{\vec{v}}(\rho) = \mathcal{R}^{\dagger-1}[\log P(\vec{x})]$ results in a perfect reproduction of the target probability distribution $P(\vec{x})$. In practice, large values of K are often impractical. However, in our experience, all but the most pathological probability density functions $P(\vec{x})$ can be approximated well with only a small number of linear potential functions. In figure 1, we illustrate how a product of several linear potential functions can be used to approximate an arbitrary function.

Fig. 2. Insert Figure 2 about here.

3.3 Hard Linear Constraint Nodes

A subclass of linear constraint nodes that is especially useful is the *hard linear constraint node.* Hard linear constraint nodes have the form:

$$\phi(\vec{x}) = \begin{cases} 1 & \text{if } \vec{x} \cdot \vec{v} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

or equivalently, hard linear constraint nodes have a nonlinearity g that is a delta function. We refer to linear constraint nodes that are *not* of this form as *soft linear constraint nodes.*

Hard linear constraint nodes are useful because they enforce linear dependencies among the variable nodes in a graphical model. For example, a hard linear constraint node may enforce that variables a, b, and c obey the relationship $a + b = c$. This ability to enforce linear dependencies means that hard linear constraint nodes allow us to utilize overcomplete representations of the problem space $\vec{X}$. Specifically, a factor graph that uses an overcomplete representation is one that has more variable nodes than there are degrees of freedom in the underlying probability distribution. When the representation of $\vec{X}$ is overcomplete, then there must be linear dependencies among the variables of $\vec{X}$ of the form $\vec{x} \cdot \vec{v} = 0$. These dependencies must be enforced to prevent computing estimates that are internally inconsistent. Using standard belief propagation (equation 7), enforcing such constraints would be intractable. Using the methods in equation 11, these constraints can be efficiently enforced using a set of hard linear constraint nodes.

For any computational problem, finding the best way to represent the problem state space is crucial; some problems can be solved much more easily given the right representation. A single complete representation forces us to decide on only one representation, whereas overcomplete representations allow us to retain the benefits of multiple complete representations. One example of the use of overcomplete representations is multi-scale approaches in computer vision, which have been very successful in several domains. Another example can be found in the primate visual cortex, which is overcomplete by a factor of at least 200:1 relative to retinal input.

In figure 2, we demonstrate how hard linear constraint nodes may be used to exploit multiple-resolution techniques with belief propagation. Multiple-resolution methods, and similar approaches such as wavelet-domain processing and image-pyramid techniques, are all highly successful in computer vision, and have contributed to algorithms for image denoising [33], shape-from-stereo [34], motion [35], texture classification [36], region classification [37], and segmentation [38]. Previous statistical inference approaches that exploited multiple-resolution methods were limited to simple Gaussian models or gradient descent optimization methods. The use of hard linear constraint nodes makes multiple-resolution representations available to belief propagation techniques.

Another example of an overcomplete representation often used in vision is surface normal maps used to represent 3D surface shape. Such maps, or "needle maps," are typically represented using two values per pixel: $p = \frac{\partial z}{\partial x}$ and $q = \frac{\partial z}{\partial y}$. For any real surface $z(x, y)$, its gradient field must satisfy the zero curl requirement, or equivalently,

$$\frac{\partial}{\partial y}\left(\frac{\partial z}{\partial x}\right) = \frac{\partial}{\partial x}\left(\frac{\partial z}{\partial y}\right) \tag{23}$$

$$\frac{\partial}{\partial y}p = \frac{\partial}{\partial x}q \tag{24}$$

In the computer vision literature, this equality also referred to as the integrability constraint, which ensures that a surface's normal map must integrate to a valid surface z. When p and q do not satisfy this relationship, there is no surface $z(x, y)$ that is consistent with p and q. In discrete form, the integrability constraint is equivalent to

$$p(x, y) - q(x, y) + q(x + 1, y) - p(x, y + 1) = 0 \tag{25}$$

where

$$p(x, y) = z(x + 1, y) - z(x, y) \tag{26}$$

$$q(x, y) = z(x, y + 1) - z(x, y) \tag{27}$$

The integrability constraint can be enforced efficiently using a hard linear constraint node of four variables. For many problems of 3D shape inference, representing shape using a surface normal map can be a great advantage. Consider the classic problem of shape-from-shading, where the image intensity at each point restricts the surface normal to lie along some one-dimensional manifold, according to the Lambertian equation:

$$i(x, y) = \max(0, \frac{1 + pL_p + qL_q}{\sqrt{1 + p^2 + q^2}\sqrt{1 + L_p^2 + L_q^2}}) \tag{28}$$

where L_p and L_q specify the lighting direction. This relationship between p and q could be implemented as a pairwise clique in an overcomplete factor graph with the integrability constraint enforced using hard linear constraint nodes of clique-size four [9]. Alternatively, the Lambertian relationship could be enforced using cliques of size three in a complete factor graph whose variable nodes represent depth at each pixel:

$$i(x,y) = \max(0, \frac{s(x,y)}{\sqrt{1+L_p^2+L_q^2}}) \tag{29}$$

$$s(x,y) = \frac{1 + (z_{x+1,y} - z_{x,y}L_p) + (z_{x,y+1} - z_{x,y})L_q}{\sqrt{1 + (z_{x+1,y} - z_{x,y})^2 + (z_{x,y+1} - z_{x,y})^2}} \tag{30}$$

However, note that because absolute depth is completely ambiguous in shape-from-shading, the computed marginals of z should be expected to be highly uniform over a large range of depths. Even if an absolute depth is arbitrarily chosen at one node, belief propagation is then charged with the task of propagating this value to all nodes in the image. Since uncertainty compounds over space, this measure would be ineffective outside of a small radius. Thus, using an overcomplete representation in this case is essential.

Another useful application of hard linear constraint nodes is the ability to aggregate over a set of local data to compute global features, such as by summing over several variable nodes. For example, in [39], the authors seek to infer the location and activity of a person from a stream of several days worth of GPS coordinates. In order to place a prior over the number of times a given activity occurs in a single day, variable nodes representing individual activities must be summed over. In [39], techniques similar to hard linear constraint nodes are used to perform belief propagation efficiently, where a tree of variable nodes is constructed, each node summing over two children. The methods of this paper show that such a tree structure can be replaced by a single hard linear constraint factor node, by setting $\vec{v}$ in equation 8 to $[-1, 1, 1, \ldots, 1]$. This would reduce the memory requirements by half (without increasing the number of operations), and, for convergent variants of belief propagation (discussed in section 6), would reduce the number of iterations of belief propagation required. The results of this paper also show how belief propagation can be made efficient for a much larger class of potential functions, including other examples that can be used to aggregate data across many variable nodes. For example, section 5, we show how variable nodes that extract the maximum value from a stream of local variable nodes can also be made efficient.

4 Nonlinear Constraint Nodes

We now extend our method to include potential functions of the form

$$\phi(\vec{x}) = g(g_1(x_1) + \cdots + g_N(x_N)) \tag{31}$$

For the sake of brevity, we consider the case where $N = 3$, although the same method works for cliques of arbitrary size. If g_i is invertible for all i, then we can apply a change of variables to equation 5 to get:

$$\begin{aligned} M_1(x_1) =& \iint g(g_1(x_1) + g_2(x_2) + g_3(x_3)) \\ & \quad m_2(x_2)\, m_3(x_3)\, dx_2\, dx_3 \qquad (32) \\ =& \iint J(\hat{x}_2, \hat{x}_3)\, g(g_1(x_1) + \hat{x}_2 + \hat{x}_3) \\ & \quad m_2(g_2^{-1}(\hat{x}_2))\, m_3(g_3^{-1}(\hat{x}_3))\, d\hat{x}_2\, d\hat{x}_3 \qquad (33) \end{aligned}$$

where we have applied the change of variables

$$\hat{x}_2 = g_2(x_2) \qquad (34)$$

$$\hat{x}_3 = g_3(x_3) \qquad (35)$$

$$J(\hat{x}_2, \hat{x}_3) = \left(\frac{\partial}{\partial \hat{x}_2} g_2^{-1}(\hat{x}_2) \right) \left(\frac{\partial}{\partial \hat{x}_3} g_3^{-1}(\hat{x}_3) \right) \qquad (36)$$

The Jacobian $J(\hat{x}_2, \hat{x}_3)$ can be absorbed into the messages by defining

$$\hat{m}_i(\hat{x}_i) = m_i(g_i^{-1}(\hat{x}_i))\, \frac{\partial}{\partial \hat{x}_i} g_i^{-1}(\hat{x}_i) \qquad (37)$$

and so we have

$$M_1(x_1) = \iint g(g_1(x_1)+\hat{x}_2+\hat{x}_3)\hat{m}_2(\hat{x}_2)\hat{m}_3(\hat{x}_3)d\hat{x}_2 d\hat{x}_3 \qquad (38)$$

We can then apply the methods of section 3.1 to get

$$M_1(x_1) \propto \int g(g_1(x_1)+y_2) \int \hat{m}_2(y_2-y_3)\hat{m}_3(y_3)\, dy_3\, dy_2 \qquad (39)$$

where we have made the change of variables $y_2 = \hat{x}_2 + \hat{x}_3$ and $y_3 = \hat{x}_3$.

If g_i is not invertible, we can still apply the same technique if we integrate equation 5 separately for each branch of $g_i^{-1}(x_i)$. For example, if $g_i(x_i) = x_i^2$, simply integrate over the range $(-\infty, 0]$, and then over the range $(0, +\infty)$, and add the two integrals together. $g_i(x_i)$ has an inverse within both of these ranges.

Using these techniques, belief propagation can be performed efficiently for a wide range of high dimensional potential functions. These include all axis-aligned generalized Gaussian distributions and Gaussian Scale Mixtures, which are popular for natural image models and denoising [33]. Since additional nonlinear potential functions of pairs of variables y_i and y_{i+1} can be embedded into equation 39 at no additional computational cost, many non axis-aligned Gaussians and other potential functions can also be computed efficiently using these methods.

5 Transformed Variable Elimination

The computational shortcuts introduced in the previous sections can be made even more general, and to apply to an even larger class of potential functions. In this section, we widen the class of potential functions that can benefit from the efficient belief propagation techniques developed so far, and at the same time, place these techniques in a broader computational framework that provides a different perspective into how these computational speed-ups are achieved, and how these methods can be tailored to suit specific applications.

For higher-order cliques, the problem of computing messages

$$m^t_{f \to i}(x_i) = \sum_{\vec{x}_{\mathcal{N}(f)\setminus i}} \phi_f\left(\vec{x}_{\mathcal{N}(f)}\right) \prod_{j \in \mathcal{N}(f)\setminus i} m^t_{j \to f}(x_j) \tag{40}$$

is not unlike the problem of computing a single-variate marginal

$$P_i(x_i) = \sum_{X \setminus x_i} P(X) \tag{41}$$

Thus, belief propagation exploits the factorization of a high-dimensional probability distribution to decompose a difficult problem (exponential in the dimensionality of X) into several easier, but similar problems (each exponential in N, the dimensionality of the clique).

When $P(X)$ can be factorized (as in equation 3), single variate marginals can be computed efficiently using the *Variable Elimination Algorithm* [40]. Note that this algorithm differs from belief propagation in that rather than computing *all* single-variate marginals of a distribution the elimination algorithm finds the marginal of only one variable. The variable elimination algorithm works by choosing a variable $x_j \in X \setminus x_i$, and then summing over all terms ϕ_k that depend on x_j. For example, if $P(X) = f_1(x_1, x_2, x_3) f_2(x_3, x_4)$, then eliminating the variable x_4 would proceed as:

$$\begin{aligned} P_i(x_1) &= \sum_{x_2}\sum_{x_3}\sum_{x_4} f_1(x_1, x_2, x_3) f_2(x_3, x_4) & (42)\\ &= \sum_{x_2}\sum_{x_3} f_1(x_1, x_2, x_3) \sum_{x_4} f_2(x_3, x_4) & (43)\\ &= \sum_{x_2}\sum_{x_3} f_1(x_1, x_2, x_3) g(x_3) & (44) \end{aligned}$$

The variable elimination process is repeated until all variables other than x_i have been eliminated. The computational complexity of the variable elimination algorithm depends on the structure of the factorization, and also on the order of elimination chosen. When the order is optimal, the complexity of the variable elimination algorithm is $\mathcal{O}(NM^{T+1})$, where M is the number of states of each variable, and T is the *treewidth* of the Markov Random Field (MRF) underlying the factorization of $P(X)$ (see [41] for a review of the treewidth of a graph). Unless the graph is very dense, $T+1$ is typically less than the number of variable nodes in the graph, and so the variable elimination algorithm represents a substantial improvement over brute force summation to compute a single-variate marginal.

If it was possible to use the variable elimination algorithm to more efficiently compute a belief propagation message (equation 40), then it also would have been possible to further factorize the clique potential function $\phi_f(x_{\mathcal{N}(f)})$ into a product of smaller, more efficient potential functions. Thus, we can assume that ϕ_f does not factorize, and so a direct application of the variable elimination algorithm cannot help to make belief propagation more efficient. The key insight of using linear constraint nodes is that by applying a transform $\mathcal{T}$ to the space $X \setminus x_i$ we may be factorize the transformed potential function, and so be able to apply variable elimination to an otherwise unfactorable clique.

By framing the methods of section 3 in this way, we can illustrate how these methods can be extended to a larger class of potential functions ϕ_f. So far, the methods of this paper has focused on finding transforms of $\phi_f(x_{\mathcal{N}(f)})$ that result in an underlying MRF in the form of a tree. A tree has a treewidth of one. Thus, once a MRF is in tree form, variable elimination can be used to compute the marginal of any node in $\mathcal{O}(NM^2)$ time. It is also possible to consider transforms $\mathcal{T}$ that transform the clique into other graphs of bounded treewidth that still represent a computational advantage over brute force summation. This allows us to improve the performance of a wider class of potential functions ϕ_f.

Let us restrict ourselves for now to linear transforms $\mathcal{T}$. Let $\mathcal{M}$ be the inverse transform matrix, so that $\mathcal{M}\vec{y} = \vec{x}$, for $\vec{x} \in X$. $\mathcal{M}$ must be an invertible matrix, and it must preserve x_i. Without loss of generality, we assume that $i = 1$, and so the top row of $\mathcal{M}$ must be $(1, 0, 0, ..., 0)$. Using transform $\mathcal{T}$, computing belief propagation messages now becomes

$$\begin{aligned} m^t_{f\to i}(x_1) &= m^t_{f\to i}(y_1) && (45)\\ &= J_{\mathcal{M}} \int_{Y\setminus y_1} \phi_f\left(\mathcal{M}\vec{y}\right) \prod_{j\in\mathcal{N}(f)\setminus i} m^t_{j\to f}(\mathcal{M}_{j*}\cdot\vec{y})d\vec{y} && (46)\end{aligned}$$

where $J_{\mathcal{M}}$ is the Jacobian of $\mathcal{M}$, and $\mathcal{M}_{j*}$ is the j^{th} row of $\mathcal{M}$. The goal of using a transform $\mathcal{T}$ is to choose a transform such that ϕ factorizes under $\mathcal{T}$:

$$\phi_f(\vec{x}) = \phi_f(\mathcal{M}\vec{y}) = \prod_{i=1}^{K_Y} \phi_f^{(i)}(\vec{y_i}) \qquad \vec{y_i} \subset \vec{Y} \tag{47}$$

The integrand in equation 46 specifies a MRF graph G with variable nodes labeled y_1 through y_N. Each of the K_Y subsets $\vec{y_i}$ must be fully connected in G. Additionally, because of the incoming messages $m^t_{j\to f}$, for each row $\mathcal{M}_j$

of $\mathcal{M}$, the variables corresponding to the nonzero entries of $\mathcal{M}_{j*}$ must also be fully connected in G. The computational cost of computing the integral in equation 46 with messages represented as histograms will then be $\mathcal{O}(NM^{T_G})$, where T_G is the treewidth of the graph G.

5.1 Products of Linear Constraint Nodes

To illustrate the flexibility of this approach, we will now use this analysis to show that messages from a single factor node consisting of a product of K linear experts can be computed in time $\mathcal{O}(NM^{K+1})$. Suppose the potential function ϕ_f over clique $\vec{X}$ is:

$$\phi_f(\vec{x}) = \prod_{k=1}^{K} f_k(\vec{x} \cdot \vec{v}^{(k)}) \tag{48}$$

As described in section 3.2, one way to implement such a product of multiple linear constraints is by constructing a separate factor node for each constraint f_k (figure 3 is an example). Messages from those factors would then be computed independently, each in $\mathcal{O}(NM^2)$ time, using the methods of section 3.1. Alternatively, these factor nodes can be combined into one, and the methods of section 3.1 no longer apply. The underlying probability distributions represented by these two factor graphs are equivalent; only their factorizations are different. Because belief propagation exploits the structure of the factor graph to perform inference efficiently, the results of belief propagation will depend on the shape of the factor graph even if the underlying probability distribution is unchanged. As mentioned in section 2, when sum-product belief propagation converges, the resulting marginals form a minima of the Bethe free energy, a quantity from statistical physics which estimates the distance between the true multivariate probability distribution and the estimated single-variate marginals [19]. The quality of this approximation improves as larger cliques are grouped together [42]. As an extreme example, consider that any probability distribution can be represented by a factor graph with a single factor node connected to each variable node. Inference using belief propagation in such a graph would be exact, but intractable. Conversely, splitting factor nodes into multiple factors typically improves belief propagation efficiency but reduces the quality of the approximation. Thus, combining a product of multiple linear constraints (as in equation 48) into a single factor node may cause belief propagation to estimate marginals more accurately than using a separate factor node for each linear constraints. Products of multiple linear constraints within a single factor node are not eligible for the methods of section 3.1, but using the transformed variable elimination methods of this section, we can show how these messages can be computed in $\mathcal{O}(NM^{K+1})$ time. Assuming that $N << M$, this represents a computational advantage over the original brute-force approach as long as $K+1<N$.

Under transformation $\mathcal{T}$, ϕ_f of equation 48 becomes

$$\phi_f(\vec{y}) = \prod_{k=1}^{K} f_k(\mathcal{M}\vec{y} \cdot \vec{v}^{(k)}) = \prod_{k=1}^{K} f_k(\vec{y} \cdot \mathcal{M}'\vec{v}^{(k)}) \tag{49}$$

where $\mathcal{M}'$ denotes the transpose of $\mathcal{M}$. There are many transforms $\mathcal{T}$ that can reduce the computation of messages from this factor node from $\mathcal{O}(M^N)$ to $\mathcal{O}(NM^{K+1})$. Here, we will choose $\mathcal{M}$ to be an upper-triangular band matrix with bandwidth $K+1$, with row $\mathcal{M}_{1*} = (1, 0, ..., 0)$. Next, we constrain $\mathcal{M}$ so that the vector $\mathcal{M}'\vec{v}^{(k)}$ is zero everywhere except for elements 1 through $K+1$. Note that this ensures that under transform $\mathcal{T}$, in the MRF G underlying Y, y_i and y_j are only connected for $|i-j| \leq K$. This ensures that G has a treewidth of K.

The constraint that the vector $\mathcal{M}'\vec{v}^{(k)}$ is only nonzero in elements 1 through $K+1$ is equivalent to

$$\mathcal{M}_{*i} \cdot \vec{v}^{(k)} = 0 \quad \forall k \leq K, \quad K+1 < i \leq N \tag{50}$$

where $\mathcal{M}_{*i}$ is the i^{th} column of $\mathcal{M}$. By construction, column $\mathcal{M}$ is only nonzero between elements $i-K$ and i. Thus, we can achieve our constraint by setting the $(K+1)$-element vector $(\mathcal{M}_{(i-K),i}, ..., \mathcal{M}_{i,i})$ to be perpendicular to $(v_{i-K}^{(k)}, ..., v_i^{(k)})$ for all $k \leq K$, and $K < i \leq N$. Note that if the bandwidth of $\mathcal{M}$ (and thus the treewidth of G) were any smaller, this constraint could not be satisfied. Also note that for $K=1$, the transform described here matches the example transform used as an example in section 3.1. For the change of variables used in equation 10, $\mathcal{M}$ is given by

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{v_2} & -\frac{1}{v_2} & 0 \\ 0 & 0 & \frac{1}{v_3} & -\frac{1}{v_3} \\ 0 & 0 & 0 & \frac{1}{v_4} \end{pmatrix} \tag{51}$$

$$\mathcal{M}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & v_2 & v_3 & v_4 \\ 0 & 0 & v_3 & v_4 \\ 0 & 0 & 0 & v_4 \end{pmatrix} \tag{52}$$

In section 3, we mentioned that a good choice of the transform of variables may allow one to embed additional pairwise nonlinear potential functions at no additional cost. We will explain that in more detail here. Suppose that our factorized distribution $P(\vec{X})$ contains the factors $\phi_1(\vec{x})$ and $\phi_2(\vec{x})$, both ranging over the same subset of variables $\vec{x}$, where

$$\phi_1(\vec{x}) = g_1(\vec{x}\cdot\vec{v}) \tag{53}$$
$$\phi_2(\vec{x}) = g_2(\vec{x}\cdot\vec{v}_1, \vec{x}\cdot\vec{v}_2) \tag{54}$$

One approach is to implement ϕ_1 and ϕ_2 as two separate factor nodes in the factor graph. However, this requires additional computation. Additionally, unnecessarily separating overlapping factors can degrade the Bethe approximation that underlies belief propagation, reducing accuracy [42]. Combining these factors into a single factor node with potential $\phi_1\phi_2$ could be advantageous.

Let $\mathcal{M}$ be a matrix that allows messages $m^t_{\phi_1\to 1}(x_i)$ from ϕ_1 to variable node x_1 to be computed in $\mathcal{O}(NM^2)$ time (if $\vec{x}$ contains four variables, then $\mathcal{M}$ is the matrix given by equation 51). Now suppose that v_1 and v_2 both lie in the plane defined by two consecutive rows j and $j+1$ of $\mathcal{M}^{-1}$. Then, in the transformed space $\vec{y} = \mathcal{M}^{-1}\vec{x}$, the MRF corresponding to ϕ_2 consists of only a single connection joining variables y_j and j_{j+1}. This means that, under the transformed space, the two potential functions ϕ_1 and ϕ_2 have overlapping factor graphs. That allows us to combine ϕ_1 and ϕ_2 into one factor node and still compute messages efficiently.

For example, consider the four-dimensional linear constraint node

$$\phi_1(\vec{x}) = g(v_1x_1 + v_2x_2 + v_3x_3 + v_4x_4) \tag{55}$$

discussed in section 3. Using the change of variables given by $\mathcal{M}$ in equation 51, we can compute messages $M_1 = m_{f\to 1}$ efficiently according to

$$M_1(x_1) \propto \int g(v_1x_1 + y_2)\Bigg(\int m_2(\frac{y_2 - y_3}{v_2}) \\ \Bigg(\int m_3(\frac{y_3 - y_4}{v_3})m_4(\frac{y_4}{v_4})dy_4\Bigg)dy_3\Bigg)dy_2 \tag{56}$$

Now suppose that $\phi_2(\vec{x}) = h(x_3, x_4)$. Both x_3 and x_4 lie on the plane spanned by $y_3 = v_3x_3 + v_4x_4$ and $y_4 = v_4x_4$. That means that we can represent $\phi_2(\vec{x})$ as

$$\phi_2(\vec{x}) = h(\frac{y_3 - y_4}{v_3}, \frac{y_4}{v_4}) = \hat{h}(y_3, y_4) \tag{57}$$

Thus, messages from the combined factor node $\phi_1\phi_2$ can be computed as

$$M_1(x_1) \propto \int g(v_1x_1 + y_2)\Bigg(\int m_2(\frac{y_2 - y_3}{v_2}) \\ \Bigg(\int m_3(\frac{y_3 - y_4}{v_3})m_4(\frac{y_4}{v_4})\hat{h}(y_3, y_4)dy_4\Bigg)dy_3\Bigg)dy_2 \qquad (58)$$

In a previous paper by one of the authors [9], this technique was used for an application that infers 3D shape from a shaded image. The approach described here made it possible to to combine a hard linear constraint that enforced the integrability of the surface:

$$\phi_1(\vec{x}) = \delta((q_1 - q_2) + (p_1 - p_2)) \qquad (59)$$

with a spatial prior on the second order derivative of depth $\frac{\partial^2 z}{\partial x \partial y}$:

$$\phi_2(\vec{x}) = \exp(-\frac{|q_1 - q_2|}{2b}) \qquad (60)$$

5.3 Sums of Linear Constraint Nodes

It is also useful to note that some potential functions $\phi_f(\vec{x})$ which cannot be made more efficient under any transform $\mathcal{T}$ can be expressed as the *sum* of some number of efficient potential functions. For example, we may find that

$$\phi_f(\vec{x}) = \phi_{f1}(\vec{x}) + \phi_{f2}(\vec{x}) \qquad (61)$$

where ϕ_{f1} and ϕ_{f2} admit transforms that reduce each potential to a low-treewidth MRF. In such cases, the belief propagation messages $m_{f\to i}(x_i)$ can be computed by summing messages from ϕ_{f1} and ϕ_{f2}:

$$m^t_{f\to i}(x_i) = m^t_{f1\to i}(x_i) + m^t_{f2\to i}(x_i) \qquad (62)$$

Thus, if a potential is a sum of linear constraint nodes, messages $m_{f\to i}(x_i)$ can be computed in time $\mathcal{O}(bNM^2)$, where b is the number of terms in equation 61.

As an example, consider the hard constraint node that enforces that variable x_n is the maximum of several variable nodes:

$$x_n = \max_i\{x_1, \ldots, x_{n-1}\} \qquad (63)$$
$$\phi_f(\vec{x}) = \delta(x_n - \max_i\{x_1, \ldots, x_{n-1}\}) \qquad (64)$$

This type of constraint may be useful to extract a pertinent global feature from a stream of variable nodes. The potential $\phi_f(\vec{x})$ can be expressed as a sum of n MRFs with treewidths of 1. To illustrate with $N = 4$:

$$\begin{aligned}\phi_f(\vec{x}) &= \delta(x_4 - \max_i\{x_1, x_2, x_3\}) && (65)\\ &= H(x_1 - x_2)H(x_1 - x_3)\delta(x_4 - x_1) \; + \\ &\quad H(x_2 - x_1)H(x_2 - x_3)\delta(x_4 - x_2) \; + \\ &\quad H(x_3 - x_1)H(x_3 - x_2)\delta(x_4 - x_3) && (66)\end{aligned}$$

where H is defined by

$$H(x) \equiv \begin{cases} 1 & x > 0 \\ 0 & otherwise \end{cases} \tag{67}$$

Each line of equation 66 is already in the form of a tree-shaped MRF; no change of variables is needed. Specifically, if we set $\phi_{f1}(\vec{x})$ to be the first line of equation 66, then we can compute $m_{f\to i}(x_4)$ as:

$$m^t_{f\to i}(x_4) = m^t_{f1\to i}(x_4) + m^t_{f2\to i}(x_4) + m^t_{f3\to i}(x_4) \tag{68}$$

$$\begin{aligned}m^t_{f1\to i}(x_4) = &\int_{-\infty}^{\infty} \left(\int_{-\infty}^{x_1} m_2(x_2)dx_2\right)\left(\int_{-\infty}^{x_1} m_3(x_3)dx_3\right) \\ &\delta(x_4 - x_1)m_1(x_1)dx_1 && (69)\end{aligned}$$

6 Convergent Loopy Belief Propagation

One of the biggest shortcomings of loopy belief propagation is that it is not guaranteed to converge. Convergence becomes increasingly unlikely when the factor graph contains many tight loops, or when potential functions are “high energy,” or nearly deterministic [43]. The application in higher-order spatial priors in section 8 contains a high number of very tight loops. Also, applications that use hard linear constraint nodes (such as the shape-from-shading application in [9]) often have convergence problems using standard belief propagation, even using different dampening, scheduling, or reweighting techniques. In general, applications that can utilize the computational shortcuts introduced in this paper also tend to benefit greatly from techniques that improve the convergence properties of belief propagation.

Fortunately, it was recently discovered that when standard sum-product loopy belief propagation converges, the resulting marginals minimize a quantity from statistical physics known as the Bethe free energy [19]. This has lead to the development of belief propagation algorithms that minimize the Bethe free energy directly [44,8], and do so while ensuring convergence.

In the applications shown later in section 8, we use the algorithm described in [8], which modifies equations 4 and 5 by:

$$m_{i\to f}^{t}(x_i) = m_{f\to i}^{t-1}(x_i)^{\frac{1-n_i}{n_i}} \prod_{g\in\mathcal{N}(i)\setminus f} m_{g\to i}^{t-1}(x_i)^{\frac{1}{n_i}} \tag{70}$$

$$m_{f\to i}^{t}(x_i) = \int_{\vec{x}_{\mathcal{N}(f)\setminus i}} \tilde{\phi}_f\left(\vec{x}_{\mathcal{N}(f)}\right) \prod_{j\in\mathcal{N}(f)\setminus i} m_{j\to f}^{t}(x_j)\, d\vec{x} \tag{71}$$

$$b_i^t(x_i) \propto \prod_{g\in\mathcal{N}(i)} m_{g\to i}^{t}(x_i)^{\frac{1}{n_i}} \tag{72}$$

where $n_i = |\mathcal{N}(i)|$. Initially, $\tilde{\phi}_f$ is set to equal ϕ_f. Each time the estimated beliefs in equation 72 converge, $\tilde{\phi}_f$ is updated according to

$$\tilde{\phi}_f(\vec{x}_{\mathcal{N}(f)}) = \phi_f(\vec{x}_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)} b_j^{\tau}(x_j)^{\frac{n_j-1}{n_j}} \tag{73}$$

where $b_j^{\tau}(x_j)$ is the belief at variable node j the last time the algorithm converged. The algorithm continues until $b_j^{\tau}(x_j)$ itself converges.

Not only does this approach guarantee convergence, but we have found that the results are often superior to standard LBP when standard LBP does converge.

One drawback to Heskes' convergent algorithm is that it is not compatible with max-product belief propagation. However, when maximum a-posteriori point estimates are desired, we can achieve them using the approach proposed by Yuille [44], which introduces a temperature T, and replaces the energy function of equation 3 with $\prod \phi_i(\vec{x_i})^{\frac{1}{T}}$. As the algorithm converges, T is reduced. As T approaches zero, the computed marginals will approximate the "maximals" of max-product belief propagation.

Another method for improving the performance and convergence properties of the original belief propagation equation is to use *tree reweighting* methods [45,46]. Tree-reweighted extensions to belief propagation apply to both sum-product belief propagation [45] and max-product belief propagation [46]. In [46], tree-reweighting methods ensure convergence for a variant of max-product belief propagation. While this approach is typically formulated specifically for pairwise connected MRFs, a straightforward generalization to arbitrary factor graphs would utilize standard belief propagation as a central step. Thus, the computational shortcuts introduced in this paper apply directly. For tree-reweighted variants of sum-product belief propagation [45], a key component of the algorithm is a belief propagation procedure of a form similar to that in equations 70-72. Thus, the methods of this paper apply to this variation as well.

7 Message Representation for Belief Propagation

For continuous random variables, the integrals of equation 9 or 11 typically cannot be computed analytically. In these cases, the beliefs $b_i(x_i)$ and messages $m_{i \to f}(x_i)$ are often approximated by discrete histograms. Discretization error can be a serious issue for histogram representations, especially for highly kurtotic or near-certain beliefs. These errors can propagate across nodes and accumulate over iterations. Also, for some applications, adequate covering of the variable space requires many bins. Even using the techniques of section 3, a high value of message length M incurs a high computational cost. In this section, we will discuss different methods of representing messages, and how to reduce discretization error.

7.1 Parametric Message Representation

One method of representing belief propagation messages is to assume that each message and belief can be well approximated with a Gaussian [4,6]. However, for many applications, marginals are often highly non-Gaussian. This is often the case in computer vision, where natural image statistics have distributions which are highly kurtotic. Even more problematic are random variables encoding the hidden underlying parameters of a scene, such as 3D shape or surface material, which often have bimodal or multimodal messages and beliefs. The shape-from-shading application of [9] and the facial appearance model of [47] are two examples of applications of belief propagation with highly multimodal messages and marginals. Among those problems where the Gaussian approximation is effective, many can be solved more simply using linear programming or gradient descent methods. For these reasons, we will focus here on the more flexible histogram and particle-based representations.

7.2 Particle-Based Message Representations

Particle based belief propagation works by approximating each message by a set of samples, or *particles*. Each particle is associated with a mean μ and a weight w. Each message $m_{i \to f}(x_i)$ is represented with M particles, with means $\{\mu_{if}^{(m)}\}_{m=1}^{M}$ and weights $\{w_{if}^{(m)}\}_{m=1}^{M}$. In the case where the potential function ϕ_f is sufficiently simple, such as a small mixture of Gaussians, $m_{f \to i}(x_i)$ can be approximated as:

$$m_{f\to i}(x_i) \approx \tilde{m}_{f\to i}(x_i) = \sum_{m=1}^{M} w_{fi}^{(m)} \phi_f(x_i, \vec{\mu}^{(m)}_{\mathcal{N}(f)\setminus i,f}) \tag{74}$$

$$w_{fi}^{(m)} = \prod_{j\in\mathcal{N}(f)\setminus i} w_{jf}^{(m)} \tag{75}$$

where $\vec{\mu}^{(m)}_{\mathcal{N}(f)\setminus i,f}$ is a vector composed of the m^{th} particles from each message $\tilde{m}_{j\to f}$ such that $j \in \mathcal{N}(f)\setminus i$ [48]. If ϕ_f is not of a simple form, then it is helpful to perform an additional step where we define $\tilde{m}_{f\to i}(x_i)$ by sampling from equation 74, to simplify subsequent computations [47]. In this case, let $\mu_{fi}^{(m)}$ be a sample drawn from $\phi_f(x_i, \vec{\mu}^{(m)}_{\mathcal{N}(f)\setminus i,f})$. We can then approximate $\tilde{m}_{f\to i}(x_i)$ as:

$$\tilde{m}_{f\to i}(x_i) = \sum_{m=1}^{M} w_{fi}^{(m)} \mathcal{N}(x_i; \mu_{fi}^{(m)}, \Lambda_{fi}) \tag{76}$$

where $\mathcal{N}(x;\mu,\Lambda)$ is a Gaussian density function with mean μ and variance Λ.

For particle based belief propagation, the computational bottleneck lies in computing $\{\mu_{if}^{(m)}\}_{m=1}^{M}$ according to equation 4, which requires sampling from $m'_{i\to f}(x_i)$, defined as:

$$m'_{i\to f}(x_i) = \zeta_{fi}(x_i) \prod_{g\in\mathcal{N}(i)\setminus f} \tilde{m}_{g\to i}(x_i) \tag{77}$$

$$\zeta_{fi}(x_i) = \int_{\vec{x}_{\mathcal{N}(f)\setminus i}} \phi_f(\vec{x}) d\vec{x} \tag{78}$$

If $\tilde{m}_{g\to i}(x_i)$ is computed as in equation 76, this requires sampling from a product of $D-1$ mixtures of M Gaussians each, where $D = |\mathcal{N}(i)|$. A straightforward sampling method would require interpreting $m'_{i\to f}(x_i)$ a weighted mixture of M^{D-1} Gaussians, and sampling from that, which requires $\mathcal{O}(M^D)$ operations. Instead, [47] showed how Gibbs sampling could be used to sample $\{\mu_{if}^{(m)}\}_{m=1}^{M}$ from $m'_{i\to f}(x_i)$ in $\mathcal{O}(D\kappa M^2)$ steps, where κ is the number of Gibbs sampling iterations required. Note that if $\tilde{m}_{f\to i}(x_i)$ is computed as in equation 74, this step is made more difficult.

Particle-based belief propagation was originally developed for pairwise connected MRFs using standard belief propagation. Both higher-order potential functions and Heskes' convergent belief propagation pose several additional obstacles for nonparametric belief propagation. Graphs with higher-order cliques tend to be more highly connected, and thus have higher values of D. For instance, the denoising problem in section 8 uses a network with $D = 12$ (see figure 3). Particle-based belief propagation is typically considered impractical for $D > 8$ [48].

This problem is exacerbated by the adjustments made in convergent variations of belief propagation. As mentioned earlier, heavily connected graphs typical of problems with high-order cliques, as well as graphs with high-energy potential functions such as hard linear constraint nodes, all tend to benefit greatly from convergent belief propagation. The greatest obstacle to particle-based message representations imposed by convergent belief propagation is the exponentiation of messages, as in equation 70. Thus, rather than sampling $\mu_{if}^{(m)}$ from a product of D mixtures of M Gaussians (already a challenging task), samples must be drawn from such a product $r$aised to an arbitrary, fractional exponent. One more difficulty in using particle-based messages for convergent belief propagation is that the potential function ϕ_f has been replaced with $\tilde{\phi}_f$ (in equation 71), which requires sampling from a product of more Gaussian mixtures. Thus, for convergent belief propagation, equation 77 becomes

$$m'_{i\to f}(x_i) = \zeta_{fi}(x_i) \left(\frac{b_i^{\tau}(x_i)}{\tilde{m}_{i\to f}^{t}(x_i)} \right)^{\frac{n_j - 1}{n_j}} \prod_{g \in \mathcal{N}(i) \backslash f} \tilde{m}_{g\to i}^{t-1}(x_i)^{\frac{1}{n_i}} \tag{79}$$

Recall that each message $\tilde{m}_{g\to i}^{t-1}$ is represented as a mixture of Gaussians. Sampling from such a distribution would be quite challenging.

7.3 Histogram-Based Message Representations

Particle-based representations benefit from the their flexible and dynamic structure, which allow them to focus computational effort on the most likely values of a probability distribution. One way to achieve similar flexibility of representation without incurring the computational expense of sampling from complex distributions is to use histograms with variable-width bins, where each bin may have a different, possibly dynamic, width. Using variable-width bin histograms, messages are approximated as:

$$m_{f\to i}(x_i) \approx \hat{m}_{f\to i}(x_i) = \sum_{m=1}^{M} w_{fi}^{(m)} \mathrm{B}_{\beta_i^{(m-1)}}^{\beta_i^{(m)}}(x_i) \tag{80}$$

$$\mathrm{B}_{\beta_0}^{\beta_1}(x) \equiv \begin{cases} 1 & x \in [\beta_0, \beta_1) \\ 0 & otherwise \end{cases} \tag{81}$$

Variable-width bin histograms have been used successfully to improve the speed and performance of the join tree algorithm [49,50]. Here we show that such a representation, when applied to belief propagation, can overcome the obstacles encountered in applying particle-based representations to Heskes' guaranteed-convergent LBP variation [8], or to problems with highly-connected graphs. We require that each message $m_{i\to f}(x_i)$ and $m_{f\to i}(x_i)$ to and from a given variable node i must have the same bin edges $\{\beta_i^{(m)}\}_{m=1}^M$. Because of this, and because histogram bins are non-overlapping (unlike Gaussian kernels), both multiplication and exponentiation now become trivial:

$$\left(\left(\sum_{m=1}^{M} w_m \mathrm{B}_{b_{m-1}}^{b_m}(x_i)\right)\left(\sum_{m=1}^{M} w'_m \ \mathrm{B}_{b_{m-1}}^{b_m}(x)\right)\right)^{\eta}$$
$$= \sum_{k=1}^{M} (w_m w'_m)^{\eta} \mathrm{B}_{b_{m-1}}^{b_m}(x) \tag{82}$$

Thus, equation 70 can be computed efficiently, even for high values of D:

$$m_{i\to f}^t(x_i) \approx \hat{m}_{i\to f}^t(x_i) \tag{83}$$
$$\hat{m}_{i\to f}^t(x_i) \equiv \hat{m}_{i\to f}^t(x_i)^{\frac{1-n_i}{n_i}} \prod_{g\in\mathcal{N}(i)\setminus f} \hat{m}_{g\to i}^{t-1}(x_i)^{\frac{1}{n_i}} \tag{84}$$

$$= \sum_{m=1}^{M} \left(w_{fi}^{(m)}\right)^{\frac{1-n_i}{n_i}} \prod_{g\in\mathcal{N}(i)\setminus f} \left(w_{gi}^{(m)}\right)^{\frac{1}{n_i}} \mathrm{B}_{\beta_i^{(m-1)}}^{\beta_i^{(m)}}(x_i) \tag{85}$$

Using linear constraint nodes, computing messages from factor to variable nodes $m_{f\to i}(x_i)$ (as in equation 11) can be viewed as a series of convolutions of scaled histograms. For the example in section 3.1, the first step is to compute the integral

$$M_{3,4}(y_3) = \int m_3(\frac{y_3 - y_4}{v_3}) m_4(\frac{y_4}{v_4}) dy_4 \tag{86}$$
$$= [m_3(t/v_3) * m_4(t/v_4)](y_3) \tag{87}$$

where $*$ denotes convolution. $m_{f\to i}(x_i)$ can be computed as

$$M_{2,3,4}(y_2) = [m_2(t/v_2) * M_{3,4}(t)](y_2) \tag{88}$$
$$m_{f\to i}(x_i) = [g(-t) * M_{2,3,4}(t)](-v_1 x_1) \tag{89}$$

Consider the simplest case for computing $M_{3,4}(y_3)$, where m_3 and m_4 are represented by histograms with all bins of width 1 ($\beta_3^{(m)} = \beta_4^{(m)} = m$), and $v_3 = v_4 = 1$, so that no scaling is required. Often, such a convolution of histograms is approximated as a discrete convolution:

$$\hat{M}_{3,4}(y_3) = \sum_{m=1}^{M} w_{2,3}^{(m)} \mathrm{B}_{\beta_{2,3}^{(m-1)}}^{\beta_{2,3}^{(m)}}(x_i) \tag{90}$$
$$w_{2,3}^{(m)} = \sum_{m'=1}^{M} w_3^{(m-m')} w_4^{(m')} \tag{91}$$
$$\beta_{2,3}^{(m)} = m \tag{92}$$

However, this approximation can result in compounded discretization error. For example, suppose that $m_3(x) = m_4(x) = \mathrm{B}_0^1(x)$. Then $M_{3,4}(y_3)$ is a piecewise linear function that is nonzero within the interval $(-1, 2)$. However, using the approximation in equation 91, $\hat{M}_{3,4}(y_3)$ will be nonzero only within $[0, 1]$, because both $\hat{m}_3$ and $\hat{m}_4$ have only one nonzero bin. A reduction in discretization error can be achieved by discretizing $\hat{M}_{3,4}(y_3)$ *after* the integration is performed:

$$w_{2,3}^{(m)} = \frac{1}{W_m} \int_{\beta_{2,3}^{(m-1)}}^{\beta_{2,3}^{(m)}} \left(\int \hat{m}_3(\frac{y_3 - y_4}{v_3}) \hat{m}_4(\frac{y_4}{v_4}) dy_4 \right) dy_3 \tag{93}$$

$$W_m = \beta_{2,3}^{(m)} - \beta_{2,3}^{(m-1)} \tag{94}$$

In the more general case, where $\{\beta_i^{(m)}\}_{m=0}^{M}$ and $\vec{v}$ are all arbitrary, an approximation like equation 91 is more difficult. Thus, in general, equation 93 is often more accurate and more convenient.

Note that the brute-force $\mathcal{O}(M^N)$ computation of an N dimensional integral of discrete histograms such as equation 7 would typically employ a method similar to equation 91, where integration is performed *after* discretization. Thus, by using linear constraint nodes, we can reduce discretization error in addition to saving time.

To implement equation 93, first observe that the product $\hat{m}_3(\frac{y_3 - y_4}{v_3})\hat{m}_4(\frac{y_4}{v_4})$ is equal to a 2D histogram under an affine transform. Equation 93 integrates this 2D function over a rectangular region. This is equivalent to summing the areas of a set of four- to six-sided polygons, each weighted by $w_3^{(m-m')} w_4^{(m')}$. It can be shown that the total number of such polygons cannot exceed $3M^2$. Thus, equation 93 can be computed in $\mathcal{O}(M^2)$ time.

At the start of the belief propagation algorithm, the locations of histogram bin edges $\{b_m\}_{m=1}^{M}$ can be initialized based on local estimates of the marginal, such as single-variate potential functions $\phi(x_i)$. In the denoising example in section 8, the intensity value of each pixel has a single-variate Gaussian potential function whose mean is the observed (noisy) pixel intensity. In this case, we set $\{b_m\}_{m=1}^{M}$ so that each bin is equally likely under this Gaussian distribution.

In some applications, such as the denoising application, it is sufficient to hold these bin widths fixed throughout the belief propagation execution. In other applications, if the range of values x_i is especially large, or if messages are expected to be very low in entropy, then it may be beneficial to adapt the histogram bin edges to best represent the current beliefs. For Heskes' convergent variation of belief propagation, this can be most conveniently done when $b_i(x_i)$ reaches convergence, and $\tilde{\phi}_f$ is updated.

Several strategies are available for adjusting the bin locations of each variable node. One approach is to simply delete low-likelihood bins and split high-likelihood bins apart. This strategy is related to some previous works that adaptively restrict the search space of belief propagation to only those states with high predicted likelihoods [51]. Another strategy is to run a special, single iteration of belief propagation where each bin is first split into 2 or 3 bins. Following this high-resolution iteration, bins can be recombined until only M bins remain. Recombination can be performed to minimize either sum-squared error with the high-resolution message, or the KL-divergence (as used by [49] to combine two possibly multidimensional histograms). Finally, if messages are expected to approximate a particular functional form, one strategy is to fit the beliefs to some parametric function and place the histogram bins to minimize error. In the denoising application of section 8, a small performance boost can be achieved by placing bins to minimize KL-divergence with a Gaussian fitted to the latest beliefs. Despite the Gaussian arrangement of bin edges, highly non-Gaussian distributions may still be effectively represented by such a histogram. At the same time, placing bins in this way allows belief propagation to focus computational effort on the most likely and most interesting intensity values.

Regardless of the strategy used, if a variable node's bin locations are altered, it is never necessary to perform interpolation to find new values of bin weights $\{w_{fi}^{(m)}\}_{m=1}^{M}$. Beliefs and messages can be retained using the original, unaltered bin locations until the belief propagation algorithm updates that variable node according to equations 4 through 6. During that update, incoming messages can be constructed using the new bin locations.

Note that the locations of histogram bins for the intermediate messages $\hat{M}_{3,4}$ and $\hat{M}_{2,3,4}$ can also be dynamically adapted. Similar strategies that are available for adapting message bin locations are also available for setting the bin locations of intermediate messages.

Finally, we point out that histogram representations and Monte Carlo integration can be combined to retain some of the advantages of both. By storing beliefs and messages in histogram form, messages $\hat{m}_{i\to f}(x_i)$ from variable to factor nodes (equation 70) can be computed easily. At the same time, the bin weights of $\hat{m}_{f\to i}(x_i)$ can be estimated using Monte Carlo integration by drawing samples $\{\vec{\mu}_{j,f}^{(m)}\}$ from each input message $\hat{m}_{j\to f}^{t}(x_j)$ and then sampling from $\phi_f(x_i, \vec{\mu}_{\mathcal{N}(f)\setminus i,f}^{(m)})$ as in equation 74. Bin weights $w_{fi}^{(m)}$ can then be set according to the number of particles that fall within each bin. This allows messages $\hat{m}_{f\to i}(x_i)$ to be computed efficiently with little sacrifice in quality.

Fig. 3. Insert Figure 3 about here.

Table 1
Insert Table 1 about here.

Fig. 4. Insert Figure 4 about here.

8 Application to Higher-Order Spatial Priors

Several state-of-the-art computer vision algorithms use belief propagation. A number of these, including stereo [3], photometric stereo [4], shape-from-shading [9], image-based rendering [10], segmentation and matting [7] work over a grid at the pixel level. These algorithms solve ambiguous and under-constrained problems, where having a strong prior for images or 3D shape is essential. However, the computational complexity of belief propagation has constrained these algorithms to weak pairwise interactions between neighboring pixels. These pairwise interactions capture the smoothness properties of images, but they overlook much of the rich statistics of natural scenes. Finding a way to exploit a stronger model of image priors using belief propagation could greatly enhance the performance of these algorithms.

One promising recent model for capturing natural image statistics beyond pairwise interactions is the Fields of Experts model (FoE), which provides a way to learn an image model from natural scenes [28]. FoE has shown itself to be highly effective at capturing complex image statistics by performing well at image denoising and image inpainting (filling in holes) using a gradient descent algorithm. The FoE model describes the prior probability of an image as the product of several Student-t distributions:

$$p(\vec{I}) \propto \tilde{p}(\vec{I}) = \prod_{C} \prod_{i=1}^{K} \left(1 + \frac{1}{2}(\vec{I}_C \cdot \vec{J}_i)^2\right)^{-\alpha_i} \tag{95}$$

where C is the set of all (overlapping) $n \times n$ patches in the image, and $\vec{J}_i$ is an $n \times n$ filter. The parameters $\vec{J}_i$ and α_i are learned from a database of natural images.

Recently, an attempt was made at performing inference in Fields of Experts models using loopy belief propagation, and the approach was tested on an image denoising problem [5]. The authors showed that using three 2×2 Fields of Experts filters yields a significant improvement over pairwise models. In their approach, the authors mitigate the computational complexity of equation 5 by restricting the intensity at each pixel to lie within a range defined by its immediate neighbors within the noisy image. Specifically, the true intensity value of each pixel is assumed to lie between the brightest and darkest of its nearest four neighbors within the noisy image, after a slight Gaussian blur is applied. Thus, computational complexity of each message is still $\mathcal{O}(M^N)$,

Table 2
Insert Table 2 about here.

Table 3
Insert Table 3 about here.

but M (the number of possible labels) is significantly reduced (note that here, $N = 4$). One drawback of this approach is that it is particular to image denoising. In many problems requiring a strong image or range image prior such as stereo and other depth inference algorithms, it can be difficult to restrict the search space of each variable based solely on local properties of the algorithm input. We seek to develop an implementation of Fields of Experts for belief propagation that can be applied to arbitrary image or range image inference problems.

Using the methods of section 3, efficient belief propagation is possible in higher-order Fields of Experts factor graphs without relying on simplifying assumptions specific to image denoising. In this section, in order to demonstrate the viability of this approach, we apply our methods to the image denoising problem, using the same 2×2 filters as [5]. Although we use image denoising as an example problem, note that this approach is not specific to image denoising, and can be used as a spatial prior for a variety of computer vision applications.

In the denoising problem described here, we are given a natural image (such as figure 4**a**) that has been corrupted with additive Gaussian noise of known variance (such as figure 4**b**). The object is to remove this noise and recover the original image. Using the Fields of Experts model, the conditional probability of the denoised image $\vec{I}$ given the noisy image $\vec{I}_N$, is modeled by

$$p(\vec{I}|\vec{I}_N) \propto \tilde{p}(\vec{I}|\vec{I}_N) \tag{96}$$

$$\tilde{p}(\vec{I}|\vec{I}_N) = \tilde{p}(\vec{I}) \prod_{x,y} \left(\frac{1}{\sigma\sqrt{2\pi}} e^{(\vec{I}(x,y) - \vec{I}_N(x,y))^2/(2\sigma^2)} \right) \tag{97}$$

where the (unnormalized) prior $\tilde{p}(\vec{I})$ is the Fields of Experts model given in equation 95. The Fields of Experts spatial prior is implemented according to the factor graph in figure 3. The Gaussian likelihood is implemented as a prior at each node, and requires no additional messages. Note that this model is capable of performing denoising in a variety of other noise circumstances, such as non-Gaussian or multiplicative noise.

Note that in the factor graph in figure 3, the observed, noisy pixel values are not explicitly represented as variable nodes. Instead, the Gaussian likelihood potential functions are absorbed into the factor nodes neighboring each pixel, and therefore require no additional belief propagation messages.

In our implementation, each variable node's beliefs and messages are represented using 16 bins. Bin edges are initialized so that each bin has equal probability under the Gaussian distribution $P(true\_intensity|noisy\_intensity)$, and bins span the range of possible intensity values from 0 to 255. Results are reported for the case where bin edges remain static during the inference procedure, and also for the case where bin edges are occasionally updated to minimize the KL-divergence between the histogram $\hat{b}_i(x_i)$ and a Gaussian distribution fitted to the current beliefs. Intermediate messages (such as $\hat{M}_{3,4}(y_3)$ and $\hat{M}_{2,3,4}(y_2)$) were represented as histograms with 32 bins. Bin edges for intermediate messages were chosen by first computing a histogram of 96 bins, where edges were chosen to minimize the KL-divergence between the convolution of two Gaussians fit to the two convolved input messages. Then the most unlikely consecutive pairs of these bins were combined until 32 bins remained. We ran each image for 15 outer-loop iterations (15 updates of $\tilde{\phi}_f$, as in equation 73) of the convergent belief propagation algorithm described in section 6. On average, this required about 35 iterations of belief propagation. The 2×2 Fields of Experts parameters used by our model were the same as those in [5].

For comparison, we also tested 2×2 Fields of Experts using the gradient descent algorithm used in [28] (code available online). In each case, gradient descent was run for 3000 iterations using a step-size of 0.1.

Sample results from our approach are shown in figure 4. We measured the average peak signal to noise ratio (PSNR) for each algorithm over the same set of 10 images from the Berkeley segmentation database [52] that was used in [5]. Here, PSNR is defined by

$$PSNR = 20 \log_{10}(255/\sqrt{MSE}) \tag{98}$$

where MSE is the mean squared error. These results are shown in table 1. In tables 1 and 1, we also show results for five canonical images from denoising literature, as used by Portilla [33].

As shown in figure 4**c**, belief propagation over pairwise-connected Markov random fields tends to produce piecewise constant results. A 2×2 Fields of Experts model promises to correct this problem by modeling not only the statistics of neighboring pixels, but whole blocks of four pixels. However, gradient descent (fig. 4**d**) is unable to fully exploit this model, achieving signal-to-noise ratios that do not exceed those of the pairwise-connected model using belief propagation. Local minima encountered by gradient descent are likely at fault. Figures 4**e** and 4**f** show the results of our approach, which outperform both pairwise connected MRFs and gradient descent over the same statistical model (2×2 FoEs) by over a decibel of PSNR.

More importantly, using the methods of section 3, belief propagation can be performed efficiently in higher-order factor nodes without relying on domain-specific approximations or simplifying assumptions. On a 2.2GHz Opteron 275, our algorithm takes under two minutes for each iteration of belief propagation on a 256×256 image. By comparison, the method of [5] took 16 minutes per iteration on a 3GHz Xeon, and benefited from a reduced search space.

In addition to an improvement in running time, our approach also yielded some improvement in quality over the more brute-force belief propagation approach used by Lan et. al. [5]. One difference between these two methods is that, in order to reduce the search space of the problem, [5] relies on the assumption that pixel intensities in the original image should lie within some range determined by their immediate neighbors in the noisy image. Because this assumption is only applicable for image denoising, and our interest lies in developing spatial priors that can be used by any belief propagation algorithm, our approach does not restrict the search space. This assumption is violated by just under 10% of the pixels in the images tested, so it is reasonable to ask if this assumption could account for the improvement in performance achieved by our approach. However, when our linear constraints node algorithm is forced to make this same assumption and restrict the search space for each pixel according to its neighbors, performance *improves* slightly. For the suite of 10 Berkeley images tested in table 1, restricting the search space as in [5] increased PSNR from 31.62 to 31.68 for $\sigma = 10$ and from 27.40 to 27.57 for $\sigma = 20$. This improvement most likely results from focusing histogram bins on more likely intensity values. However, while this assumption may offer some small performance gain, it is important to remember that such assumptions are not available for other applications of spatial priors where belief propagation is more necessary, such as stereo [3], photometric stereo [4], shape-from-shading [9], image-based rendering [10], segmentation, and matting [7].

If the assumption made by Lan et. al. [5] to reduce the search space is not the cause of the performance gain of our approach, then it is most likely due to the convergent variant of belief propagation [43] and nonparametric message representations used by our approach. One reason that this performance gain is of interest is that although the underlying statistical model of natural images is identical between the two methods, the factor graph used by [5] is not identical to the one used by our method (seen in figure 3). The graph used in [5] uses a single factor node for all three 2×2 experts within a clique, whereas our method separates each expert into its own factor node. By separating out these factors, the Bethe free energy approximation used by belief propagation is degraded. The good performance of our approach shows that this sacrifice in the quality of the Bethe approximation was less than the advantages offered by convergent belief propagation and variable width bin histograms.

For the sake of comparison, we also present results for two state-of-the-art denoising algorithms: 5×5 FoEs using gradient descent [28], and an algorithm that uses Gaussian scale mixtures to model the joint distribution of wavelet coefficients [33]. These algorithms are designed specifically for image denois-

Table 4
Insert Table 4 about here.

Table 5
Insert Table 5 about here.

ing; they cannot easily be adapted for use as spatial priors in more complex algorithms like stereo, shape from shading, matting, and others. We present them here in tables 2 and 3, for a sense of perspective.

Belief propagation computes the single-variate marginals of each pixel value. The expected value of the denoised image, or the minimum mean-squared error (MMSE) point estimate, can be computed by taking the mean of each marginal. This approach usually yields the best results for our algorithm. In figure 4 and in tables 2 and 3, we also show results for the intensity values that maximize the marginal, or the "maximum marginal" (MM) point estimate. For fixed-width histograms, a continuous MRF that approximates intensity using only 16 bins would typically show high discretization error for point estimates computed this way. By using variable width histograms, these quality of these point estimates is nearly equal to MMSE results. As discussed in section 6, maximum a posteriori (MAP) point estimates can be computed using either non-convergent max-product belief propagation, or by performing annealing within convergent sum-product belief propagation [22]. For problems with smooth, unimodal likelihood functions like image denoising, using MAP point estimates is rarely beneficial.

In table 1, results using linear constraint nodes are presented both with and without dynamic readjustment of histogram bin locations. In each case, histogram bins are initialized so that each bin has an equal likelihood under to the Gaussian likelihood function. In the dynamic case, bins are also adjusted after each outer-loop iteration, as described earlier. This procedure takes negligible time, and yields a small but significant performance improvement. For other applications, where initial estimates of the marginals may be less accurate, or beliefs fluctuate more during inference (such as the shape-from-shading algorithm in [9]), dynamic histogram bin edge adjustments are more important to performance.

In addition to showing that LCNs allow belief propagation to efficiently capture nonpairwise aspects of the statistics of natural scenes, we are also interested in showing that belief propagation outperforms gradient descent techniques at finding maximally likely images I that optimize $\tilde{p}(\vec{I}|\vec{I}_N)$ (equation 97). In tables 4 and 5, we show the unnormalized log likelihoods $\log \tilde{p}(\vec{I}|\vec{I}_N)$ for the denoised images computed by both gradient descent and by our belief propagation approach. These algorithms both use the same 2×2 Fields of Experts model, and so both algorithms are attempting to opimize the same energy function. Because the spatial prior $P(\vec{I})$ may not be optimal, it is possible for an algorithm to achieve poor denoising results despite finding superior optima of $\tilde{p}(\vec{I}|\vec{I}_N)$. Tables 4 and 5 show that this is not the case. All variants of belief propagation with LCNs find denoised images that are significantly more likely (according to the model) than those chosen by gradient descent.

9 Conclusions

In this paper, we have introduced a way to efficiently perform belief propagation over large graph cliques, reducing computation from $\mathcal{O}(M^N)$ to $\mathcal{O}(NM^2)$ for a wide variety of potential functions. We have shown how these methods can be generalized in several ways to benefit a larger subclass of potential functions. Additionally, we have developed methods for representing belief propagation messages for continuous variables that remain computationally efficient for highly connected graphs, convergent variants of belief propagation, and the use of the computational shortcuts introduced in this paper. These message representations allow discretization error to be minimized while at the same time preserving computational efficiency.

The techniques introduced in this paper open up a wealth of powerful, higher-order statistical models for inference using belief propagation methods that would previously have been intractable. Belief propagation is a promising framework of optimization for these models, because it often outperforms gradient-based optimization methods by exploiting factorizations, and by performing optimization within the much larger search space of single-variate marginals, which is less prone to local extrema. Computer vision in particular stands to benefit greatly from higher order statistical models due to the complex statistical structure of natural images and underlying image properties. In particular, enabling belief propagation to exploit non-pairwise statistical relationships in images such as the Fields of Experts models, and to take advantage of overcomplete representations of a state-space such as multi-resolution representations, may prove especially useful in computer vision.

To illustrate our techniques, we applied belief propagation to the problem of performing statistical inference over higher order spatial priors of images. When these spatial priors are applied to image denoising, we demonstrated a significant performance increase over belief propagation in pairwise models and also over gradient-descent based methods for higher-order models. At the same time, a sizeable speed increase was shown over previous belief propagation approaches to higher-order graph cliques. The ability to exploit higher order spatial priors using belief propagation may be of great benefit to a number of computer vision tasks that seek to infer images or range images in ambiguous, uncertain circumstances, including stereo [3], photometric stereo [4], shape-from-shading [9], image-based rendering [10], segmentation, and matting [7].

Acknowledgements

We thank S. Roth and the authors of [5] for graciously sharing filters & results. This research was funded by an NSF Graduate Research Fellowship and an Interdisciplinary Training Grant in Computational Neuroscience (NIH/NIDA R09 DA023428) to B. Potetz and NSF IIS-0413211 and NSF IIS-0713206 to T.S. Lee.

References

[1] P. Dagum, M. Luby, Approximating probabilistic inference in bayesian belief networks is np-hard, Artif. Intell. 60 (1) (1993) 141–153.

[2] W. T. Freeman, E. Pasztor, O. T. Carmichael, Learning low-level vision, Int. J. Comp. Vis. 40 (1) (2000) 25–47.

[3] J. Sun, N.-N. Zheng, H.-Y. Shum, Stereo matching using belief propagation, IEEE Trans. Pattern Anal. Mach. Intell. 25 (7) (2003) 787–800.

[4] K. L. Tang, C. K. Tang, T. T. Wong, Dense photometric stereo using tensorial belief propagation., in: CVPR, 2005, pp. 132–139.

[5] X. Lan, S. Roth, D. P. Huttenlocher, M. J. Black, Efficient belief propagation with learned higher-order markov random fields, in: ECCV, 2006, pp. 269–282.

[6] N. Petrovic, I. Cohen, B. J. Frey, R. Koetter, T. S. Huang, Enforcing integrability for surface reconstruction algorithms using belief propagation in graphical models., in: CVPR, 2001, pp. 743–748.

[7] J. Wang, M. F. Cohen, An iterative optimization approach for unified image segmentation and matting, ICCV (2005) 936–943.

[8] T. Heskes, K. Albers, B. Kappen, Approximate inference and constrained optimization., in: UAI, 2003, pp. 313–320.

[9] B. Potetz, Efficient belief propagation for vision using linear constraint nodes, in: CVPR 2007: Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, Minneapolis, MN, USA, 2007.

[10] O. J. Woodford, I. D. Reid, P. H. S. Torr, A. W. Fitzgibbon, Fields of experts for image-based rendering, in: Proceedings of the 17th British Machine Vision Conference, Edinburgh, Vol. 3, 2006, pp. 1109–1108.

[11] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Trans. Pattern Anal. Mach. Intell. 23 (11) (2001) 1222–1239.

[12] V. Kolmogorov, R. Zabih, What energy functions can be minimized via graph cuts?, in: ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III, Springer-Verlag, London, UK, 2002, pp. 65–81.

[13] P. Kohli, M. P. Kumar, P. H. S. Torr, p^3 and Beyond: Solving energies with higher order cliques, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[14] S. Geman, D. Geman, Stochastic relaxation, gibbs distributions, and the bayesian restoration of images (1987) 564–584.

[15] E. B. Sudderth, A. T. Ihler, W. T. Freeman, A. S. Willsky, Nonparametric belief propagation, Tech. Rep. MIT//LIDS P-2551, MIT, Laboratory for Information and Decision Systems (2002).

[16] F. R. Kschischang, B. J. Frey, Iterative decoding of compound codes by probability propagation in graphical models, IEEE Journal of Selected Areas in Communications 16 (2) (1998) 219–230.

[17] K. P. Murphy, Y. Weiss, M. I. Jordan, Loopy belief propagation for approximate inference: An empirical study, in: UAI, 1999, pp. 467–475.

[18] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufman, San Francisco, CA, 1988.

[19] J. S. Yedidia, W. T. Freeman, Y. Weiss, Generalized belief propagation, in: NIPS, 2000, pp. 689–695.

[20] Y. Weiss, W. T. Freeman, Correctness of belief propagation in gaussian graphical models of arbitrary topology, Neural Computation 13 (10) (2001) 2173–2200.

[21] Y. Weiss, W. T. Freeman, On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs, IEEE Transactions on Information Theory 47 (2) (2001) 736–744.

[22] A. L. Yuille, Cccp algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation., Neural Computation 14 (7) (2002) 1691–1722.

[23] P. F. Felzenszwalb, D. P. Huttenlocher, Efficient belief propagation for early vision, in: CVPR, Vol. 1, 2004, pp. 261–268.

[24] J. H. Friedman, W. Stuetzle, A. Schroeder, Projection pursuit density estimation, Journal of the American Statistical Association 79 (1984) 599–608.

[25] S. C. Zhu, Y. N. Wu, D. Mumford, Minimax entropy principle and its applications to texture modeling, Neural Computation 9 (1997) 1627–1660.

[26] S. C. Zhu, Y. N. Wu, D. Mumford, Frame : Filters, random fields and maximum entropy — towards a unified theory for texture modeling, Int'l Journal of Computer Vision 27 (2) (1998) 1–20.

[27] G. Hinton, Products of experts, in: International Conference on Artificial Neural Networks, Vol. 1, 1999, pp. 1–6.

[28] S. Roth, M. J. Black, Fields of experts: A framework for learning image priors, in: CVPR, 2005, pp. 860–867.

[29] Y. Weiss, W. T. Freeman, What makes a good model of natural images?, in: CVPR 2007: Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, Minneapolis, MN, USA, 2007.

[30] M. F. Tappen, Utilizing variational optimization to learn markov random fields, in: CVPR 2007: Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, Minneapolis, MN, USA, 2007.

[31] J. M. Coughlan, A. L. Yuille, Algorithms from statistical physics for generative models of images, Image and Vision Computing 21 (1) (2003) 29–36.

[32] S. R. Deans, The Radon Transform and Some of Its Applications, John Wiley & Sons, 1983.

[33] J. Portilla, V. Strela, M. J. Wainwright, E. P. Simoncelli, Image denoising using scale mixtures of gaussians in the wavelet domain., IEEE Transactions on Image Processing 12 (11) (2003) 1338–1351.

[34] A. Witkin, D. Terzopoulis, M. Kass, Signal matching through scale space, in: Readings in computer vision: issues, problems, principles, and paradigms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987, pp. 759–764.

[35] J. R. Bergen, P. Anandan, K. J. Hanna, R. Hingorani, Hierarchical model-based motion estimation, in: ECCV '92: Proceedings of the Second European Conference on Computer Vision, Springer-Verlag, London, UK, 1992, pp. 237–252.

[36] S. Livens, P. Scheunders, G. V. de Wouwer, P. Vautrot, D. V. Dyck, Wavelets for texture analysis, an overview, in: Proc. IEE International Conference on Image Processing and Applications, Dublin, 1997, pp. 581–585.

[37] S. Kumar, M. Hebert, Discriminative random fields, International Journal of Computer Vision 68 (2) (2006) 179–202.

[38] H. Cheng, C. A. Bouman, Multiscale bayesian segmentation using a trainable context model, IEEE Transactions on Image Processing 10 (4) (2001) 511–525.

[39] L. Liao, D. Fox, H. Kautz, Location-based activity recognition, in: Y. Weiss, B. Schölkopf, J. Platt (Eds.), Advances in Neural Information Processing Systems 18, MIT Press, Cambridge, MA, 2006, pp. 787–794.

[40] N. L. Zhang, D. Poole, A simple approach to bayesian network computations, in: Proc. of the Tenth Canadian Conference on Artificial Intelligence, 1994, pp. 171–178.

[41] H. L. Bodlaender, A tourist guide through treewidth, Acta Cybernetica 11 (1993) 1–21.

[42] J. S. Yedidia, W. T. Freeman, Y. Weiss, Understanding belief propagation and its generalizations, in: Exploring artificial intelligence in the new millennium, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003, pp. 239–269.

[43] T. Heskes, On the uniqueness of loopy belief propagation fixed points., Neural Comp. 16 (11) (2004) 2379–2413.

[44] A. L. Yuille, An algorithm to minimize the Bethe free energy, in: EMMCVPR, 2001.

[45] M. Wainwright, T. Jaakkola, A. Willsky, A new class of upper bounds on the log partition function, Information Theory, IEEE Transactions on 51 (7) (July 2005) 2313–2335.

[46] M.-V. Kolmogorov, Convergent tree-reweighted message passing for energy minimization, IEEE Trans. Pattern Anal. Mach. Intell. 28 (10) (2006) 1568–1583.

[47] E. Sudderth, A. Ihler, W. Freeman, A. Willsky, Nonparametric belief propagation, in: CVPR, 2003.

[48] M. Isard, Pampas: Real-valued graphical models for comnputer vision, in: CVPR, 2003, pp. 613–620.

[49] A. Kozlov, D. Koller, Nonuniform dynamic discretization in hybrid networks, in: UAI, 1997, pp. 314–32.

[50] D. Koller, U. Lerner, D. Anguelov, A general algorithm for approximate inference and its application to hybrid bayes net, in: UAI, 1999, pp. 324–33.

[51] J. Coughlan, H. Shen, Shape matching with belief propagation: Using dynamic quantization to accomodate occlusion and clutter, in: CVPRW, 2004, p. 180.

[52] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, ICCV (2001) 416–423.

3. Figure 1

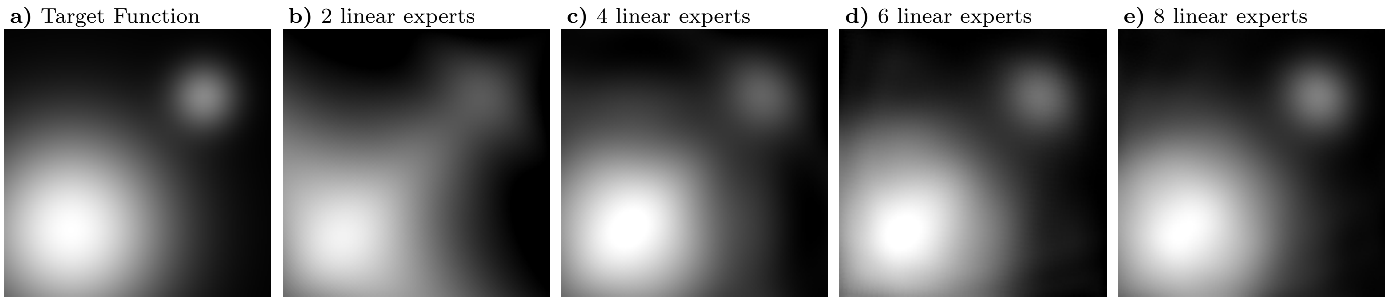

Fig. 1. Illustrating how products of linear potential functions can approximate arbitrary functions. **a)** The target potential function to be approximated: a two-dimensional mixture of Gaussians. Subfigures **b)** through **e)** show the target function approximated with an increasing number of linear potential functions. Vectors $\boldsymbol{v}_k$ were chosen to manually to be evenly spaced.

3. Figure 2

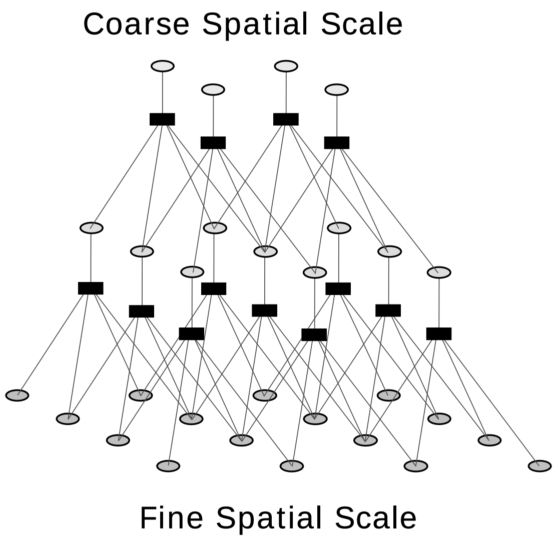

Fig. 2. A factor graph that demonstrates the use of multiple resolution inference for belief propagation. Each circle represents a variable at one of three spatial scales, and each black square represents a hard linear constraint factor node. Here, each hard linear constraint node enforces that its upper neighbor is the block average of the pixels in the next finer spatial scale. Wavelet and Laplacian image pyramids are also possible. The methods of section 3.1 reduce the number of operations required to compute belief propagation messages in this network from $\mathcal{O}(M^5)$ to $\mathcal{O}(M^2)$.

3. Figure 3

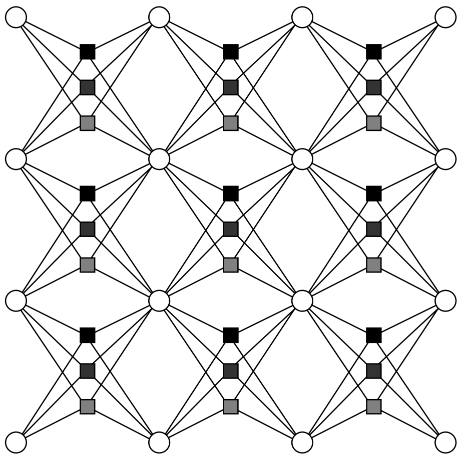

Fig. 3. A factor graph used to perform image denoising using three 2×2 Fields of Experts filters. Each variable node, shown here as circles, represents the true image intensity at a given pixel. The three gray squares represent factor nodes corresponding to the three 2×2 Fields of Experts filters.

3. Figure 4

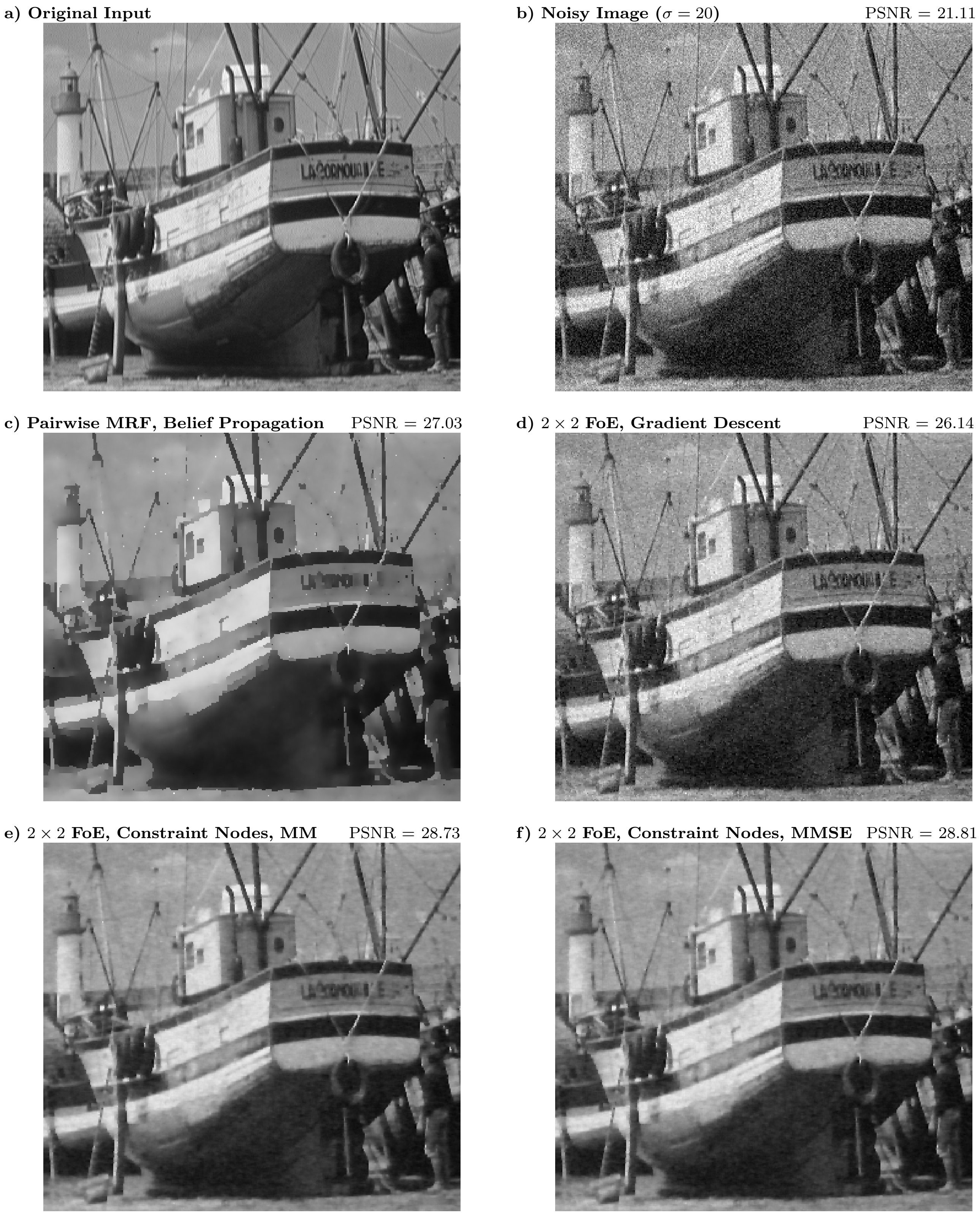
a) **Original Input**
b) **Noisy Image** ($\sigma = 20$) PSNR = 21.11
c) **Pairwise MRF, Belief Propagation** PSNR = 27.03
d) 2×2 **FoE, Gradient Descent** PSNR = 26.14
e) 2×2 **FoE, Constraint Nodes, MM** PSNR = 28.73
f) 2×2 **FoE, Constraint Nodes, MMSE** PSNR = 28.81

Fig. 4. Using higher-order Fields of Experts to perform image denoising. **a)** A cropping from the original image (from [33]). **b)** The original image with additive Gaussian noise of $\sigma = 20$. **c)** The output of belief propagation over a pairwise-connected Markov Random Field, similar to the model described in [23]. Pairwise models tend to produce piecewise constant image regions [5]. **d)** Denoising using the gradient descent algorithm employed by [28], with three 2×2 Fields of Experts filters learned from natural images. **e)** Results using the same 2×2 FoE model as **d)**, except using linear constraint nodes (the methods described in section 3) and the graphical model of figure 3. Intensity values were chosen to be the grey value with maximal marginal probability. **f)** As in **e**, except the intensity values were chosen to be the expected value of the estimated marginal.

4. Tables 1

	$\sigma = 10$	$\sigma = 20$
Noisy Input Images	28.13	22.11
Hand-tuned Pairwise MRF using belief propagation [5]	30.73	26.66
2×2 FoE using gradient descent (algorithm from [28])	30.59	26.09
2×2 FoE using belief propagation (from [5])	30.89	27.29
2×2 FoE using LCNs, Fixed Histograms, MMSE	31.51	27.29
2×2 FoE using LCNs & Adaptive Histograms, Max Marginals	31.55	27.25
2×2 FoE using LCNs & Adaptive Histograms, MMSE	31.62	27.40

Table 1
Peak signal-to-noise ratio (in decibels) for pairwise and higher-order models, averaged over the ten images from the Berkeley segmentation database [52] used in [5]. Peak signal-to-noise ratio (PSNR), in decibels, for pairwise and higher-order models, averaged over the ten images from the Berkeley segmentation database [52] used in [5]. PSNR is defined in equation 98. MMSE point estimates are taken by computing the mean of each marginal, which gives us the approximated mean of the posterior distribution. Max marginal point estimates are choosing the maximal value of each marginal. Denoising using linear constraint nodes (LCNs) with 2×2 FoEs outperforms both belief propagation on pairwise MRFs and gradient descent on identical FoEs.

4. Tables 2

$\sigma = 10$ (PSNR = 28.13)	boat	peppers	house	lena	barbara
2x2 FoE, Gradient Descent [28]	30.61	30.73	31.00	30.91	30.19
2x2 FoE, BP using LCNs, Adaptive Histograms, Max Marginal	32.30	32.95	33.92	33.49	30.11
2x2 FoE, BP using LCNs, Adaptive Histograms, MMSE	32.28	32.85	33.71	33.34	30.24
5x5 FoE, Gradient Descent [28]	33.04	34.18	35.14	35.03	32.85
Portilla et. al. [33]	33.58	33.77	35.35	35.61	34.03

Table 2
Denoising results for five canonical denoising images (used in [33]). Image noise $\sigma = 10$. *BP using LCNs* refers to belief propagation using the linear constraint node computational shortcut. State of the art denoising algorithms (bottom two rows) are also reported. Note that these algorithms are designed especially for denoising, and would be difficult to use as a spatial prior for other vision tasks like stereo, shape from shading, and others. All error values are given in peak signal-to-noise ratio (equation 98).

4. Tables 3

$\sigma = 20$ (PSNR = 22.11)	boat	peppers	house	lena	barbara
2x2 FoE, Gradient Descent [28]	26.14	26.15	26.49	26.45	25.44
2x2 FoE, BP using LCNs, Adaptive Histograms, Max Marginal	28.73	29.03	30.60	30.39	25.29
2x2 FoE, BP using LCNs, Adaptive Histograms, MMSE	28.81	29.09	30.46	30.31	25.47
5x5 FoE, Gradient Descent [28]	29.82	30.19	32.02	31.81	28.31
Portilla et. al. [33]	30.38	30.31	32.39	32.66	30.32

Table 3
Results as in table 3, except under noise with $\sigma = 20$. All error values are given in peak signal-to-noise ratio (equation 98).

4. Tables 4

$\sigma = 10$	Berkeley Suite	boat	peppers	house	lena	barbara
Original Image	-3.93	-26.62	-6.57	-6.40	-25.82	-27.84
Noisy Image	-4.15	-28.25	-7.03	-6.92	-27.78	-29.25
2x2 FoE, Gradient Descent [28]	-3.94	-26.65	-6.65	-6.52	-26.20	-27.79
2x2 FoE, BP using LCNs, Fixed Histograms, MMSE	-3.80	-25.70	-6.45	-6.31	-25.35	-26.73
2x2 FoE, BP using LCNs, Fixed Histograms, Max Marginal	-3.80	-25.68	-6.45	-6.31	-25.34	-26.67
2x2 FoE, BP using LCNs, Adaptive Histograms, MMSE	-3.80	-25.68	-6.44	-6.31	-25.33	-26.69
2x2 FoE, BP using LCNs, Adaptive Histograms, Max Marginal	-3.79	-25.62	-6.43	-6.30	-25.28	-26.62

Table 4
The (unnormalized) log-likelihood of each image reconstruction according to the 2×2 FoE model. All values are given as $\log \tilde{p}(\boldsymbol{I}|\boldsymbol{I}_N) \times 10^{-5}$, where $\tilde{p}(\boldsymbol{I}|\boldsymbol{I}_N)$ is given in equation 97. The values given for the Berkeley Suite images show the mean unnormalized log-likelihood for the ten images from the Berkeley segmentation database [52] used in [5]. The five denoising algorithms shown here all seek to optimize the same equation (i.e. equation 97 using the 2×2 FoE model). In each case, belief propagation significantly outperforms gradient descent. Thus, in addition to producing denoised images with less error, belief propagation does a better job at finding the optimum values of the FoE probability model. This means that the improvement in performance is not due to peculiarities of the FoE model. Also note that, according to the model, the denoised images computed using belief propagation have greater likelihood than the original image. This suggests that improving the model is more important than improving the method of optimization.

4. Tables 5

$\sigma = 20$	Berkeley Suite	boat	peppers	house	lena	barbara
Original Image	-4.19	-28.44	-7.03	-6.85	-27.64	-29.66
Noisy Image	-4.86	-33.39	-8.33	-8.25	-33.10	-34.04
2x2 FoE, Gradient Descent [28]	-4.29	-29.29	-7.33	-7.21	-28.93	-30.18
2x2 FoE, BP using LCNs, Fixed Histograms, MMSE	-4.00	-27.89	-7.00	-6.87	-27.55	-28.88
2x2 FoE, BP using LCNs, Fixed Histograms, Max Marginal	-4.02	-27.83	-6.99	-6.86	-27.51	-28.77
2x2 FoE, BP using LCNs, Adaptive Histograms, MMSE	-3.99	-27.85	-6.99	-6.86	-27.52	-28.82
2x2 FoE, BP using LCNs, Adaptive Histograms, Max Marginal	-3.98	-27.76	-6.97	-6.84	-27.43	-28.71

Table 5
Results as in table 4, except under noise with $\sigma = 20$. All values are given as $\log \tilde{p}(\boldsymbol{I}|\boldsymbol{I}_N) \times 10^{-5}$, where $\tilde{p}(\boldsymbol{I}|\boldsymbol{I}_N)$ is given in equation 97.