UNIVERSITY OF WISCONSIN, MACHINE LEARNING RESEARCH GROUP WORKING PAPER 06-1, SEPTEMBER 2006

Improving the Efficiency of Belief Propagation in Large, Highly Connected Graphs

Frank DiMaio and Jude Shavlik
Computer Sciences Dept.
University of Wisconsin–Madison
Madison, WI 53706
{*dimaio,shavlik*}@*cs.wisc.edu*

Abstract

We describe a part-based object-recognition framework, specialized to mining complex 3D objects from detailed 3D images. Objects are modeled as a collection of parts together with a pairwise potential function. The algorithm's key component is an efficient inference algorithm, based on belief propagation, that finds the optimal layout of parts, given some input image. Belief Propagation (BP) – a message passing method for approximate inference in graphical models – is well suited to this task. However, for large objects with many parts, even BP may be intractable. We present AggBP, a message aggregation scheme for BP, in which groups of messages are approximated as a single message, producing a message update analogous to that of mean-field methods. For objects consisting of N parts, we reduce CPU time and memory requirements from $O(N^2)$ to $O(N)$. We apply AggBP to both real-world and synthetic tasks. First, we use our framework to recognize protein fragments in three-dimensional images. Scaling BP to this task for even average-sized proteins is infeasible without our enhancements. We then use a synthetic "object generator" to test our algorithm's ability to locate a wide variety of part-based objects. These experiments show that our improvements result in minimal loss of accuracy, and in some cases produce a more accurate solution than standard BP.

1 Introduction

Several recent publications have explored the use of part-based models for recognizing generic objects in images [4, 19, 9]. These models represent physical objects as a graph: a collection of vertices ("parts") connected by edges enforcing pairwise constraints. An inference algorithm determines the most probable location of each part in the model given the image. However, this previous work has only considered simple objects with relatively few parts, often only using two-dimensional image data. We present a part-based object recognition algorithm specialized to objects with hundreds of parts in detailed, three-dimensional images.

Rich, three-dimensional data commonly arises in biological datasets, especially with recent advancements in biological imaging techniques. For example, fMRI scans produce detailed 3D images of the brain. Confocal microscopy constructs high-quality 3D images of tissues. X-ray crystallography yields a 3D electron density map, a three-dimensional "image" of a macromolecule. This three-dimensional data often contains objects comprised of many parts, connected with some complex topology. As detailed biological imagery becomes easier to acquire, techniques to accurately interpret such images are needed. For example, a vascular biologist may want to automatically locate all the blood vessels in a kidney section; a crystallographer may want to trace a piece of RNA in an electron density map. Even rich two-dimensional data, such as detailed satellite imagery, may contain complex objects that cannot be interpreted using current methods.

To effectively mine complex 3D objects, our algorithm includes an efficient message-passing inference algorithm based on *belief propagation* [15]. Message-passing algorithms are an extremely powerful tool for inference in graphical models (that is, probabilistic models defined on a graph). Belief propagation (BP) – also known as the sum-product algorithm – is a message-passing method for exactly computing marginal distributions in tree-structured graphs. In graphs with arbitrary topologies, no such optimality is guaranteed. Empirically, however, "loopy BP" often provides extremely accurate approximations, when exact inference methods are intractable [6, 13, 21].

For very large, highly-connected graphs, with large input images, even loopy BP may not offer enough efficiency. In near-fully connected graphs, with hundreds or thousands of vertices, approximations to BP's messages may be necessary to compute marginal distributions in a reasonable amount of time. We describe AggBP (for *aggregate BP*), a technique for approximating groups of BP messages with a single message. This composite message turns out to be quite similar to the message update for mean-field methods. We illustrate that, for certain types of graphs, AggBP may reduce running time in a (near-fully connected) graph with N nodes from O(N^2) to O(N).

Additionally, we provide a method for dealing with continuously-valued variables that is efficient and does not require accurate initialization. Recently, an extension to BP, nonparametric belief propagation (NBP), was introduced [18]. NBP represents variables that have continuous non-Gaussian distributions as a mixture of Gaussians. We introduce an efficient variant which alternately represents probability distributions over a continuous three-dimensional space as a set of Fourier-series coefficients. We describe efficient message passing and update algorithms in this framework.

Finally, we test our approximation techniques using both real-world and synthetic data. Our first testbed is on a real-world computer-vision task, identifying protein fragments in three-dimensional images. Interpreting these protein images is a very important step in determining protein structures using x-ray crystallography. AggBP lets us scale interpretation to large proteins in large 3D images. Our second testbed uses a synthetic object generator to test AggBP's performance locating a wide variety of objects with various part topologies.

2 Modeling 3D Objects

Following others [4], we describe a class of objects using a graphical model. Graphical models, such as Bayesian networks and Markov fields, represent the joint probability distribution over a set of variables as a function defined over some graph. A *pairwise undirected graphical model* (or *pairwise Markov field*) represents this joint distribution as a product of potential functions defined on each *edge* and *vertex* in the graph. To represent an object as a graph, then, vertices correspond to parts in the object, while edges correspond to constraints between pairs of parts.

Formally, the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $s \in \mathcal{V}$ connected by edges $(s,t) \in \mathcal{E}$. Each node in the graph is associated with a (hidden) random variable $x_s \in \mathbf{x}$, and the graph is conditioned on a set of observation variables $\mathbf{y}$. For object recognition, these x_s's are the 3D position of part s. Each vertex has a corresponding *observation potential* $\psi_s(x_s, \mathbf{y})$, and each edge is associated with an *structural potential* $\psi_{st}(x_s, x_t)$. Then, we can represent the full joint probability as

$$p(\mathbf{x}|\mathbf{y}) \propto \prod_{(s,t)\in\mathcal{E}} \psi_{st}(x_s, x_t) \times \prod_{s\in\mathcal{V}} \psi_s(x_s|\mathbf{y}) \qquad (1)$$

In many applications, this paper included, we are most concerned with finding the maximum marginal assignment, that is, the labels $x_s \in \mathbf{x}$ that maximize this joint probability for some value of $\mathbf{y}$.

To describe an object using a graphical model, one must provide three pieces of data: a *part graph*, each node's *observation potential*, and each edge's *structural potential*. Given a graph describing an object, these potential functions are learned from a set of previously solved problem instances.

For 3D object recognition, the *part graph* is fully connected; most edges are associated with identical, weak (diffuse) potentials, while a sparse subset of the graph (the "skeleton") connects very highly correlated variables. As an illustration, consider using a graphical model for recognizing people in images, as in Figure 1. In this model, a sparsely connected skeleton connects highly correlated nodes. For example, the head and body are connected in this skeletal structure, because the position of the head is highly correlated with the position of the body.

However, many other pairs of nodes – such as the left leg and the left arm – are not connected in the skeletal structure, yet their labels are not completely (conditionally) independent. There is a weak correlation: the two parts may not occupy the same location in space. As this constraint is not implicitly modeled by the chain that connects them in the skeletal structure, an edge between them is necessary. These *occupancy* edges only serve to ensure that two parts in the model do not overlap in 3D space. For example, when modeling a hand [19], occupancy edges are required to ensure two fingers do not occupy the same space. The potential associated with these edges is typically very diffuse; it is non-zero everywhere except in a small neighborhood around the origin (in the node's local coordinates).

Each part's observation potential is usually based on the application of a simple classifier. At each location in 3D space, it returns the probability that a particular part is at that location. Individual part potential functions may use template matching, color matching, edge detection, or any other method. Observation potentials need not be particularly accurate, as belief propagation is able to infer the true location using the combined power of many weak detectors.

As illustrated in the person-detector example, structural potentials are broken into two types: *skeletal potentials* (or *sequential potentials*) model the relationship between parts connected in an object's skeleton, while *occupancy potentials* model the weakly correlated relationship between all other pairs of nodes. Skeletal potentials may take an arbitrary form, learned from a set of allowable object con-

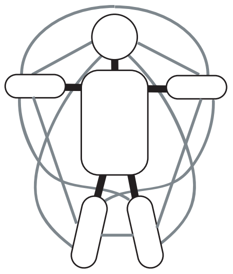

Figure 1. A sample graphical model for recognizing a person in an image. Thicker dark edges illustrate the highly-correlated "skeleton" of the model, while thinner light edges are weakly-correlated occupancy edges, which ensure two parts do not occupy the same 3D space.

formations. They may be a function of position as well as orientation of the 3D object. Occupancy potentials take the form of a step function (using a "hard collision" model) or a sigmoidal function (using a "soft collision" model). Occupancy potentials only depend on the position of the connected objects, and is only nonzero if the connected objects are sufficiently far apart.

3 Inference: Locating 3D Objects

Given an image and some object's graphical model, *inference* attempts to find the most-probable location of each of the object's parts in the image. Because our object graph is fully connected, with a number of loops, exact inference methods either will not work (e.g., tree-based methods) or are intractable (e.g., exhaustive methods). Instead, we are forced to rely on approximate inference methods. Our object-recognition framework uses *belief propagation*, a message-passing approximate inference algorithm.

3.1 Belief propagation

Belief propagation – based on Pearl's polytree algorithm [15] – computes the *marginal probability* over each x_s (the location of each part) by passing a series of local messages. The marginal probability refers to the joint probability, where all but one variable is summed out, that is:

$$b_s(x_s|\mathbf{y}) = \sum_{x_1} \dots \sum_{x_{s-1}} \sum_{x_{s+1}} \dots \sum_{x_N} P(\mathbf{x}|\mathbf{y}) \qquad (2)$$

This marginal distribution is important because it provides information about the distribution of some variable (x_s above) in the full joint distribution, without requiring one to explicitly compute the (possibly intractable) full joint distribution.

Algorithm 1: Belief propagation

```
input  : Observational potentials $\psi_s(x_s|\mathbf{y})$ and
         structural potentials $\psi_{st}(x_s, x_y)$
output: An approximation to the marginal
         $\hat{b}_s(x_s|\mathbf{y}) \approx \sum_{x_1} \dots \sum_{x_{s-1}} \sum_{x_{s+1}} \dots \sum_{x_N} P(\mathbf{x}|\mathbf{y})$
initialize accumulator, messages to 1
while $\hat{b}$'s have not converged do
    foreach part $s = 1 \dots N$ do
        $\hat{b}_s(x_s|\mathbf{y}) \leftarrow \psi_s(x_s|\mathbf{y})$
        foreach part $t = 1 \dots N$ do
            if $t \neq s$ and $\hat{b}_t$ has been updated then
                $m^n_{t\to s}(x_s) \leftarrow \int_{x_t} \psi_{st} \times \frac{\hat{b}^n_t}{m^{n-1}_{s\to t}} \, dx_t$
            end
            $\hat{b}_s(x_s|\mathbf{y}) \leftarrow \hat{b}_s(x_s|\mathbf{y}) \times m^n_{t\to s}(x_s)$
        end
    end
end
```

Pseudocode appears in Algorithm 1. At each iteration, a part in the model computes the product of all incoming messages, then passes a *convolution* of this product to its neighbors (for clarity, the message's dependence on **y** is usually dropped):

$$m^n_{t\to s}(x_s) \propto \int_{x_t} \psi_{st}(x_s, x_t) \times \psi_t(x_t|\mathbf{y}) \times \prod_{u \in \Gamma(t)\backslash s} m^{n-1}_{u\to t}(x_t) \, dx_t \qquad (3)$$

In the above, $\Gamma(t)\backslash s$ denotes all neighbors of t in the graph *excluding* s. Typically, these messages are normalized so that the probabilities sum to unity. Following Koller *et al.* [11], we assign some order to the nodes, and update the belief at each node sequentially, alternating between forward and backward passes through our ordering. At any iteration, the algorithm computes an approximation to the marginal as the product of incoming messages and the node's observation potential ψ_s,

$$\hat{b}^n_s(x_s|\mathbf{y}) \propto \psi_s(x_s|\mathbf{y}) \times \prod_{u \in \Gamma(t)} m^n_{u\to t}(x_t) \qquad (4)$$

In tree-structured graphs (graphs without cycles), this algorithm is exact. Unfortunately, for many tasks, this limitation is overly restrictive. In graphs with arbitrary topologies, however, there are no guarantees to the convergence of this algorithm – and convergence may not be to the correct solution – but empirical results show that "loopy BP" often produces good estimates in practice [14].

Several papers have explored circumstances under which loopy BP's convergence or optimality can be guaranteed. Weiss has shown a category of graphical models with a single loops in which optimality is guaranteed [22]. More recent work [24] has shown the existence of fixed-points in loopy BP, but they are neither unique or optimal. Heskes [7] has developed sufficient conditions for the uniqueness of BP's fixed-points. Others have characterized the fixed-points in loopy BP [20].

Others have explored message approximation in loopy BP. When exact message computation is intractable, stochastic approximation of messages [11] as well as message simplification [1] have been investigated. Additionally, when dealing with continuous-valued variables, some sort of approximation or simplifying assumptions must be made [9, 18]. Ihler *et al.* [8] have explored the consequences of approximating messages in BP, placing bounds on accumulated message errors as BP progresses.

A recent paper [18] investigates the special case where the labels x_t are continuously valued. Using ideas from particle filtering [3], these authors use weighted-Gaussian probability density estimates. That is, given a set of weights w_s^i, $i = 1 \dots N$, a set of Gaussian centers μ_s^i and a covariance matrix Λ_s, an estimate of the belief is given by

$$\hat{b}_s^n(x_s|\mathbf{y}) = \sum w_s^{(i)} \times \mathcal{N}(x_s; \mu_s^{(i)}, \Lambda_s) \qquad (5)$$

Message computation is implemented using an efficient Gibbs sampling routine. The Gibbs sampler [8] approximates the product of k Gaussian mixtures – each with M components – as an M-component mixture. This sampling is used to compute BP message products. For the BP convolution operation, forward sampling is employed. Their inference algorithm was applied to several vision tasks. Isard [9] makes use of a similar technique, with a sampling routine specialized to mixture-of-Gaussian edge potentials.

In the next sections, we provide several techniques to scale belief propagation for 3D part-based object recognition. One of these techniques is a message aggregation to handle large, highly connected graphs that arise in mining complex, 3D images. We also include an alternate representation for continuously-valued beliefs and potentials, which allows for efficient message computation and products.

4 Scaling Belief Propagation

Belief propagation was originally intended for small, sparsely connected graphs. In large, highly-connected graphs, the number of messages quickly becomes overwhelming. To make BP tractable in these types of graphs, we propose AggBP, which approximates some subset of outgoing messages at a single node with a single message, replacing many message computations with relatively few.

4.1 BP Message Aggregation

In the undirected graphical models used for 3D object recognition, pairs of nodes along *skeleton* edges are highly correlated. Consequently, messages along these edges have a high information content. It is important to exactly compute messages along these edges. Coarse approximations – like those used in mean field methods [10] – introduce too much error.

However, in these graphs, the *majority* of edges are *occupancy* edges, which enforce the constraint that two parts cannot occupy the same 3D space. The potential functions associated with these edges are weak – that is, nearly uniform – and messages along these edges carry little information. Along these edges, we can make some approximations; a full BP message update may be overkill.

Formally, BP's message update, given by Equation 4, can be alternately written as (again, the explicit dependence of the message on **y** is dropped for clarity):

$$m_{t\to s}^n(x_s) \leftarrow \alpha_1 \int_{x_t} \psi_{st}(x_s, x_t) \times \frac{\hat{b}_t^n(x_t|\mathbf{y})}{m_{s\to t}^{n-1}(x_t)} dx_t \qquad (6)$$

The denominator in the above, $m_{s\to t}^{n-1}(x_t)$ is a term that serves to avoid double-counting or "feedback", making the method exact in tree-structured graphs. In loopy graphs, such feedback – through the graph's loops – in unavoidable. For messages along occupancy edges this denominator carries little information, and AggBP drops it with little loss of accuracy. This gives an update equation more like the naïve mean-field theory update :

$$m_{t\to s}^n(x_s) \leftarrow \alpha_2 \int_{x_t} \psi_{st}(x_s, x_t) \times \hat{b}_t^n(x_t|\mathbf{y})\, dx_t \qquad (7)$$

The key advantage of doing this is – assuming that the structural potential ψ_{st} is identical along all occupancy edges – is *all occupancy messages outgoing from a single node are identical*. For the remainder of this section, we will refer to these approximate messages as $m_{t\to *}(x_*)$.

Assuming identical ψ_{st}'s, AggBP reduces the number of occupancy messages computed from O(N^2) to O(N) in a model with N parts. However, *updating* the belief for some part still requires multiplying all the incoming occupancy messages times all the incoming skeletal message; for an N-part model, this is still O(N^2). To reduce this complexity, we utilize the fact that each node receives this broadcast message from all but a few nodes in the graph: its neighbors (in the skeleton graph) and itself. We consider, then, send-

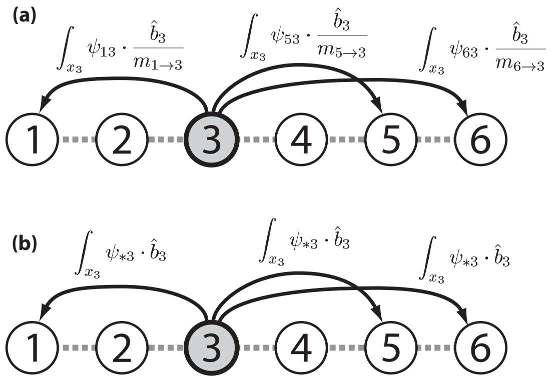
(a)
$\int_{x_3} \psi_{13} \cdot \frac{\hat{b}_3}{m_{1\to 3}}$
$\int_{x_3} \psi_{53} \cdot \frac{\hat{b}_3}{m_{5\to 3}}$
$\int_{x_3} \psi_{63} \cdot \frac{\hat{b}_3}{m_{6\to 3}}$
1 2 3 4 5 6
(b)
$\int_{x_3} \psi_{*3} \cdot \hat{b}_3$
$\int_{x_3} \psi_{*3} \cdot \hat{b}_3$
$\int_{x_3} \psi_{*3} \cdot \hat{b}_3$
1 2 3 4 5 6

Figure 2. Our message aggregation approximates (a) all the outgoing messages at node 3, with (b) a single message sent to all non-adjacent nodes. Caching these aggregate message products results in significant runtime savings.

ing all these aggregate messages to a central accumulator:

$$ACC(x_*) \leftarrow \prod_{t=1}^{N} m_{t\to *}(x_*) \tag{8}$$

This accumulator is then used to efficiently update a node's belief, by sending – in a single message – the product of all occupancy messages.

Figure 2 illustrates AggBP when the graph is a chain. For a chain, we compute the product of incoming messages, using this accumulator, as:

$$\begin{aligned}
\hat{b}_1 &\leftarrow \psi_1 \times ACC \times \frac{m_{2\to 1}}{m_{1\to *} \times m_{2\to *}} \\
\hat{b}_2 &\leftarrow \psi_2 \times ACC \times \frac{m_{1\to 2} \times m_{3\to 2}}{m_{1\to *} \times m_{2\to *} \times m_{3\to *}} \\
&\vdots \\
\hat{b}_k &\leftarrow \psi_k \times ACC \times \frac{m_{k-1\to k} \times m_{k+1\to k}}{m_{k-1\to *} \times m_{k\to *} \times m_{k+1\to *}} \\
&\vdots
\end{aligned}$$

The numerators of these message updates contain skeletal messages, while the denominators contain the approximated occupancy messages. AggBP reduces the runtime and memory requirements from O(N^2) to O(N) in a model with N parts. The storage benefit is especially appealing when the 3D space for each part is large, and storing o(N^2) messages is space-prohibitive. Section 5.1 provides a closer look at one such application where this is the case.

Algorithm 2: Aggregate belief propagation (same I/O as Algorithm 1).

```
initialize accumulator $ACC$, messages $m$ to 1
while $\hat{b}$'s have not converged do
    foreach part $s = 1 \dots N$ do
        $ACC \leftarrow ACC / m^{n-1}_{s\to *}$
        $\hat{b}_s(x_s|\mathbf{y}) \leftarrow \psi_s \times ACC$
        foreach part $t = 1 \dots N$ do
            if $s$ is skeleton neighbor of $t$ then
                if $\hat{b}_t$ has been updated then
                    $m^n_{t\to s}(x_s) \leftarrow \int_{x_t} \psi_{st} \times \frac{\hat{b}^n_t}{m^{n-1}_{s\to t}} dx_t$
                end
                $\hat{b}_s(x_s|\mathbf{y}) \leftarrow \hat{b}_s(x_s|\mathbf{y}) \times (m^n_{t\to s}/m^n_{t\to *})$
            end
        end
        // compute composite message
        $m^n_{s\to *}(x_s) \leftarrow \int_{x_t} \psi_{*s} \times \hat{b}^n_s(x_s|\mathbf{y})$
        // update accumulator
        $ACC \leftarrow ACC \times m^n_{s\to *}$
    end
end
```

Algorithm 2 gives a pseudocode overview of AggBP (notice the arguments x_s and x_t have been dropped for clarity). As we progress from node to node, instead of computing each outgoing message at a single node, we instead compute significantly fewer composite messages. The key difference from Algorithm 1 is in the inner loop, "if s is a skeleton neighbor of t." For graphs used in part-based object recognition, this loop is rarely entered, requiring significantly fewer message calculations.

Finally, when the occupancy edges all have a *different* potential function (e.g., when parts in the model are of a different size), then AggBP can still take advantage of this approximation, with additional approximation error. In this case, AggBP simply computes the broadcast message from a part t using the *average* potential function outgoing from t:

$$m^n_{t\to *}(x_*) \leftarrow \alpha \int_{x_*} \frac{\sum_{u=1}^{N} \psi_{tu}(x_t, x_*)}{N} \times \hat{b}^n_t(x_t) dx_t \tag{9}$$

Section 5.2 explores how well AggBP handles varying occupancy potentials.

4.2 Belief representation

Section 3 describes an approach to belief propagation with continuous-valued labels based on particle filtering.

However, there may be cases where these models are insufficient. In general, when using particle filtering-based methods, reasonably accurate initialization of the Gaussian centers representing the probability distribution is necessary for accurate inference [19]. In this section, we describe an alternative belief representation that uses a Fourier-series probability density estimate [17] to represent probabilities and messages. While particle-filtering-based methods tend to concentrate on high-probability space, our approach accurately represents the probability distribution over the entire space of each random variable. Efficient message passing and message computation make this representation ideal for large, highly-connected graphs.

Formally, we represent marginal distributions $\hat{b}_s^n$ as a set of 3-dimensional Fourier coefficients f_k, where, given an upper-frequency limit, K

$$\hat{b}_s^n(x_s|\mathbf{y}) \approx \sum_{k=0}^{K} f_k \times e^{-2\pi i (x_s \cdot k)} \tag{10}$$

Messages are represented using the same type of probability density estimate.

4.2.1 Message convolution

Recall from Eq. (3) that computing $m_{t\to s}$ requires integrating the product of the edge potential $\psi_{st}(x_s, x_t)$, the observation potential $\psi_s(x_s, \mathbf{y})$, and the incoming message product $\prod m_{s\to t}(x_t)$ over all x_t. While this computation is difficult in general for Fourier-based density estimates, if our edge potential can be represented as a function of the *difference* between the labels of the two connected nodes, that is, $\psi_{st}(x_s, x_t) = f(||x_s - x_t||)$, then $m_{t\to s}$ is just a convolution:

$$m_{t\to s}^n(x_s) = \big(\psi_{st} * \prod m_{s\to t}^{n-1}\big)(x_s) \tag{11}$$

This is easily computed as the product of Fourier coefficients:

$$\mathcal{F}\big[m_{t\to s}^n(x_s)\big] = \mathcal{F}\big[\psi_{st}(x_s, x_t)\big] \times \mathcal{F}\Big[\big(\prod m_{s\to t}^{n-1}(x_t)\big)\Big]. \tag{12}$$

For object recognition, *all* of our occlusion potentials may be represented in this manner. That is, the potential here only depends upon the *difference* between labels.

This computational shortcut was originally proposed by Felzenswalb for belief propagation in low-level vision [5]. Computing these message products is efficient, with running time $\mathrm{O}(K)$, where K is the high-frequency limit of the density estimate.

4.2.2 Message products

As shown in Eq. (4), computing the current belief $\hat{b}_s^n$ at a given node requires taking the product of all incoming messages $m_{t\to s}(x_s)$, and multiplying it by the observation potential $\psi_s(x_s, \mathbf{y})$. Given the Fourier coefficients of all messages, we compute this multiplication in real space:

$$\hat{b}_s^n(x_s|\mathbf{y}) = \psi_s(x_s, \mathbf{y}) \times \prod \mathcal{F}^{-1}\Big[\mathcal{F}\big[m_{t\to s}^n(x_s)\big]\Big] \tag{13}$$

As with the message convolution, this operation is fairly efficient. Each transform and inverse transform runs in time $\mathrm{O}(K \log K)$.

5 Experiments

In this section, we compare the standard "full" belief propagation algorithm with AggBP, using both real-world and synthetic datasets. The real-world task is based upon locating protein fragments in 3D images; these objects consist of a chain of "parts" (amino acids). The synthetic dataset looks at our algorithm's performance recognizing objects containing more complex part topologies.

5.1 Protein fragment identification

One application for object recognition arises from x-ray crystallography. our approach is building a graphical model for a protein, in order to identify it in a *three-dimensional* image. These three-dimensional images, or *electron density maps*, are produced when determining protein structures using x-ray crystallography. Interpreting this map – illustrated in Figure 3 – is the final step of x-ray crystallography [16]. Interpretation produces the Cartesian coordinates of every atom in the protein. It is often quite difficult and time-consuming to interpret electron-density maps: it make take weeks to months of a crystallographer's time to find every atom in the protein. Alternatively, a *backbone trace* focuses instead on predicting the location of a key carbon atom – the alpha carbon, or C$_\alpha$ – contained in each amino acid. We use AggBP to automatically determine a 3D backbone structure given an electron density map and a protein sequence.

5.1.1 Task overview

A protein is constructed as a linear chain of amino acids. Each of the 20 different naturally-occurring amino acids consists of a constant four-atom motif (the *backbone*) and a variable *sidechain*. Figure 4 illustrates a protein structure, highlighting the backbone and sidechains. Protein recognition is difficult for several reasons. The electron-density maps are experimentally determined and are often very noisy. Additionally, the protein chain is extremely flexible: proteins are typically tightly coiled together, and amino acids distant on the linear chain are often very close in three-dimensional space. A protein chain typically has

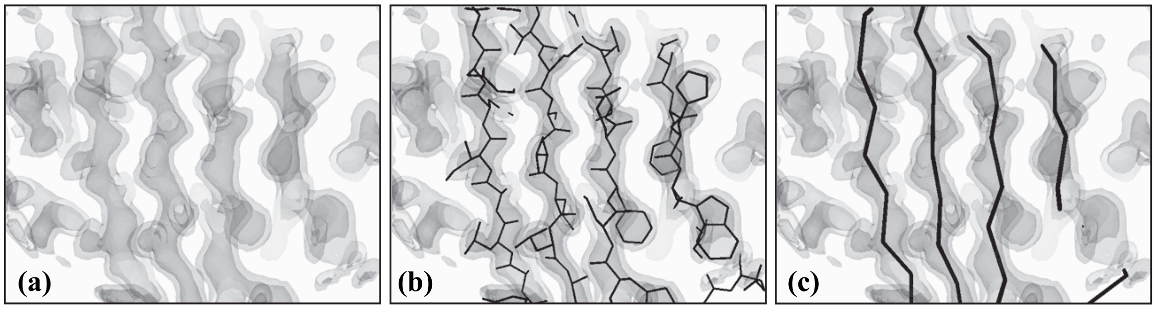

Figure 3. An overview of electron density map interpretation. Given the amino acid sequence of the protein and (a) a density map, the crystallographer's goal is to find (b) the positions of all the proteins atoms. Alternatively, (c) a backbone trace, provides the location of a key carbon atom called C$_\alpha$ that is in each amino acid.

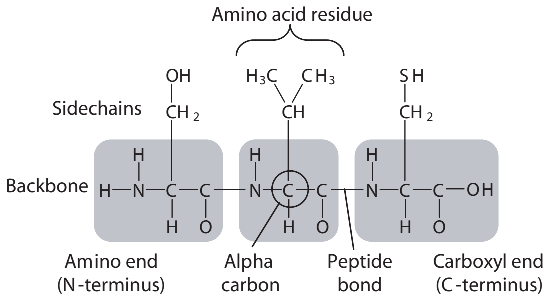

Figure 4. Proteins are constructed by amino acids condensing into a polypeptide chain. A chain of three amino acids is illustrated.

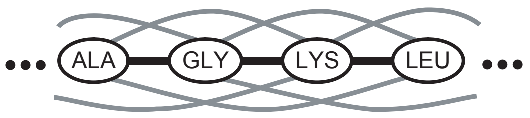

Figure 5. Our protein part graph. The probability of some conformation is the product of an observation potential, a sequential potential (dark lines), and an occupancy potential (light lines).

hundreds to thousands of amino acids, and the density map "image" usually contains more than one copy of the protein.

Figure 5 shows our encoding of a protein as a Markov field model. Each node s represents an amino-acid in the protein. The label $w_s = \{x_s, q_s\}$ for each amino-acid consists of seven terms: the 3D Cartesian coordinates x_s of the amino acid's C$_\alpha$, and four internal parameters q_s. These four internal parameters are an alternate parameterization of: (a) three 3D rotational parameters plus (b) the bend angle formed by three consecutive C$_\alpha$s. Probability distributions over Cartesian space make use of the Fourier-series based parameterization outlined in Section 4.2. The internal parameters q_s are modeled as constant-width Gaussians conditioned on the Cartesian coordinates, that is, $\hat{b}_s(q_s) = \mathcal{N}(q_s|\mu_s^i(x_s), \Lambda), i = 1 \ldots 4$.

A recent paper by this paper's authors [2] describes how protein-specific structural and observation potential functions are learned; it also compares this method to other algorithms for map interpretation. However, this previous work did not present the results shown here; nor did it describe our message approximations and aggregation. Before describing our new contributions, we briefly review the potential functions for electron density map interpretation that we introduced in our prior article, and use in our experiments in this paper.

Each node's potential function $\psi_s(w_s, \mathbf{y})$ is computed by matching a learned set of small-protein-fragment templates to the electron density map. Details of this potential function are beyond the scope of this paper, but appear elsewhere [2]. Edge potential functions are of two basic types. *Sequential* edges connect adjacent amino acids; the corresponding potential $\psi_{st}^{seq}(w_s, w_t)$ ensures that these adjacent amino acids are the proper distance apart and in the proper orientation with respect to each others. This proper distance and orientation is learned from a set of previously-solved protein structures. *Occupancy* edges connect all other pairs of amino acids, and the corresponding potential $\psi_{st}^{occ}(x_s, x_t)$ ensures two amino acids do not occupy the same space.

As the graph is fully connected, and the messages are

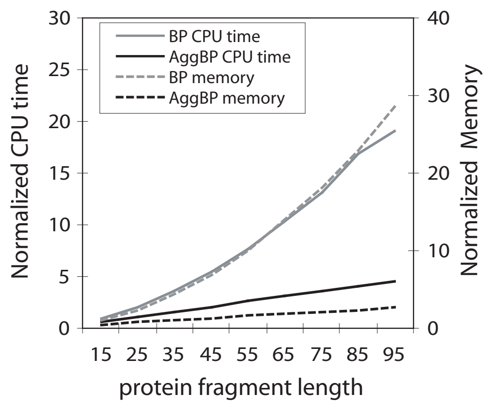

Figure 6. A comparison of memory and CPU time usage between our approximate-BP and standard BP.

continuous three-dimensional probability distributions over the entire unit cell, storage and run-time requirements are considerable. We use the AggBP's message aggregation and approximation outlined in Section 4.1. Without these shortcuts, inference in even medium-sized proteins would be computationally intractable.

5.1.2 Results

This section details some experiments locating protein fragments in electron density maps. These maps were provided to us by crystallographer George Phillips, at UW-Madison. We convoluted the maps with a Gaussian to simulate a poor-quality (3Å resolution) density map: a resolution at which other automated interpretation methods fail. The maps had been previously solved by a crystallographer, giving us the "true" solution with which to compare our predictions. For a given protein, we were provided the sequence, and we constructed a Markov field based on this sequence.

We compare standard BP inference (*exact-BP*) to AggBP on this Markov field model. Exact-BP was unable to scale to the entire protein (as many as 500 amino acids in our testset), so to compare these two methods we consider protein *fragments* of between 15 and 65 amino acids. CPU time per iteration and memory usage of the two techniques are illustrated in Figure 6. Because the actual running-time and memory usage is dependant upon the size of the density map, we normalize these values, so that exact-BP's time and memory usage at 15 amino-acids is 1.0 (in an average-sized protein, these values are 200 MB and 120 sec, respectively). We increase fragment length until our 6 GB machine began paging; Figure 6 does not include time swapping to disk; however, in larger fragments this is a serious issue.

At each of six fragment lengths, we searched for 15 different fragments of that length from 5 different proteins (in 5 different electron-density maps), for a total of 90 different target fragments. Fragments were chosen that roughly corresponded with the beginning, middle, and end of each protein chain. We ran BP and AggBP until convergence or 20 iterations (where one iteration is a single forward or backward pass through the protein). In each map, we reduced the electron density map to a small neighborhood around each fragment, then searched for that fragment. Using this reduced density map is a more-realistic model of searching for a complete protein.

Results from this experiment appear in Figure 7. We plot three different metrics – RMS deviation and log-likelihood of the maximum-marginal interpretation (i.e. the predicted backbone trace), as well as average KL-divergence [12] between the marginal distributions – as a function of iteration. As these plots show, the solutions found by these two methods differ somewhat, however, in terms of error versus the true trace, both produce equally accurate traces. More interestingly, Figure 7b shows the log-likelihood of the maximum-marginal interpretation. Under this metric, AggBP produces a better solution; perhaps because AggBP's approximation avoids overfitting the data. Figure 7d shows the RMS error as a function of protein-fragment length. Not surprisingly, both methods seem to perform slightly worse when searching for longer fragments; still, the predicted structure is fairly accurate – considering the quality of the maps – with an RMS error of under 4Å.

Finally, a scatterplot of log-likelihoods, where each of the 90 fragments is represented as a point, is illustrated in Figure 8. In this figure, points below the diagonal correspond to fragments on which our AggBP produced a more-likely interpretation. For almost every fragment, AggBP produces a solution with a greater log-likelihood than does standard BP. This difference is statistically significant; a two-tailed, paired *t* test gives a *p* value of 0.014.

5.2 Synthetic object recognition

While the protein fragment identification testbed shows the CPU and memory savings achievable by our algorithm, it uses a rather limited part topology: the skeletal structure is just a linear chain, and each part is a constant distance apart. In this section, we construct a synthetic object generator, that builds "part graphs" with varying branching factors, object sizes, and object "softness." We also explore approximation performance under various part-finder accuracies.

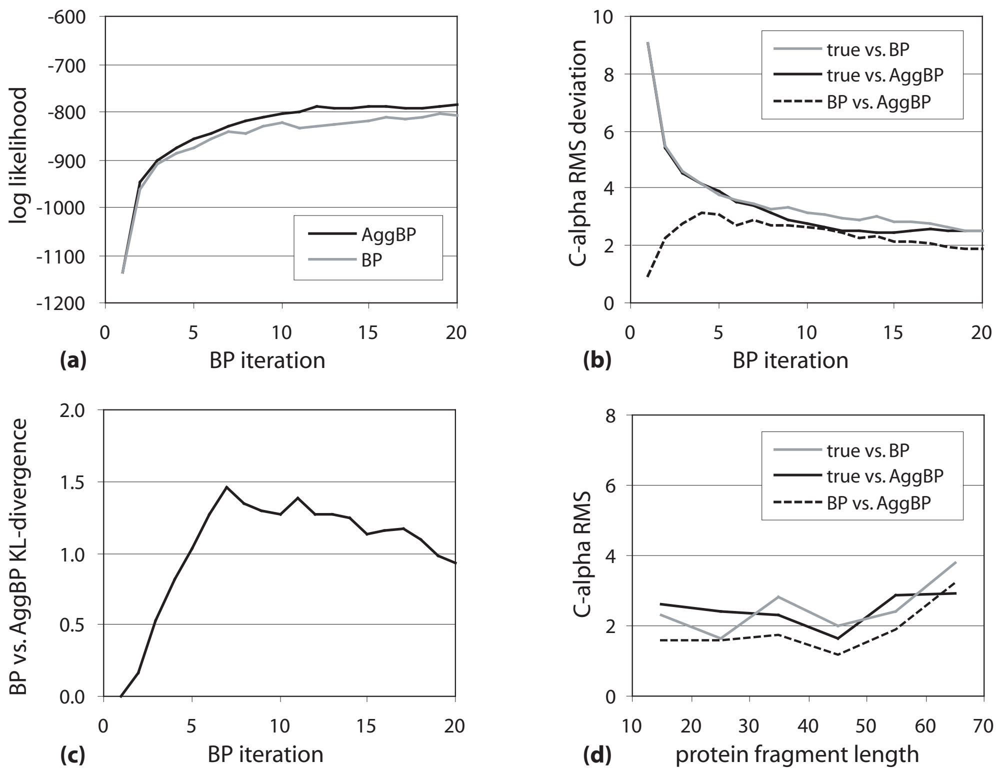

Figure 7. A comparison of our BP approximation with standard BP as the algorithm progresses, using (a) RMS deviation, (b) average KL-divergence of the predicted marginals, and (c) log-likelihood of the maximum-marginal interpretation. Additionally, (d) shows RMS error as a function of protein fragment size (at iteration 20).

5.2.1 Object generator

We have developed a synthetic object generator to better understand how well AggBP works over the range of possible tasks locating 3D objects composed of interconnected parts. This object generator lets us vary the graph topology and individual part parameters, as in Figure 9. The generator constructs objects with a predefined number of parts, arranged in a tree-structured skeleton. Given some branching factor, the skeleton is randomly assembled from the parts. As before, all pairs of nodes *not* connected in this skeleton are connected with edges enforcing *occupancy* potentials, which ensure two parts do not occupy the same three-dimensional space.

Each part in the model is given a *radius* $r_i \in \mathbf{r}$ and a *softness* $s_i \in \mathbf{s}$, from which the structural potential functions are derived. Pairs of parts directly connected in the skeleton maintain a distance equal to the sum of their radii. Other pairs may not occupy the same 3D space: they should be *at least* as far apart as the sum of their radii, although they may get closer than this distance as softness increases. This softness parameter allows these part pairs to get slightly closer than the sum of their radii with some low probability. Specifically, the softness parameter replaces the occupancy potential's step function with a sigmoid. For non-zero softness, then, the probability distribution of the distance d between two parts i and j, with radii r_i and r_j, and softness s_i and s_j, is given by:

$$p_{ij}(d) = \frac{1}{1 + exp\left(\frac{-(d-(r_i+r_j))}{s_i \cdot r_i + s_j \cdot r_j}\right)} \tag{14}$$

Our testbed generator also generates observation potentials ψ_{obs}, that is, the probability distribution of each part's location in 3D space. These would normally be generated by some type of pattern matcher in a 2D or 3D image. Our algorithm's generator assumes we have a classifier that – given a location in 3D space – returns a score. Shown in Figure 10, scores are drawn from one of two distributions: at the true location of a part, the score for that part is

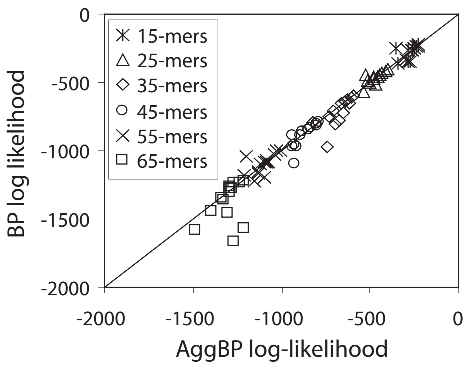

Figure 8. A scatterplot showing – for each of the 90 target fragments – the log likelihood of AggBP's trace versus the standard BP's trace. Points below the diagonal correspond to fragments where AggBP returned a more likely solution.

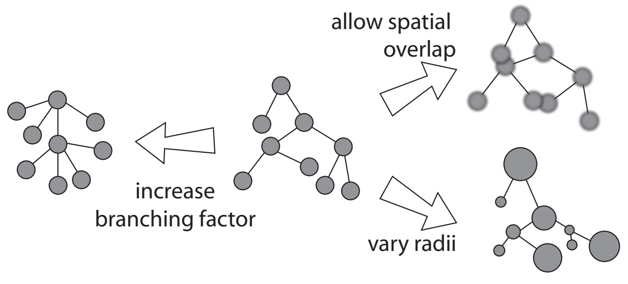

Figure 9. An illustration of the three graph topology parameters we vary using our graph generator.

drawn from one distribution, at any other location the score is drawn from another distribution.

For simplification, we assume both distributions are fixed-width Gaussians with different means. Varying the difference in means results in classifiers with varying accuracy. Given this difference in means, then, we *generate* each part's observation potential by drawing scores at random from these two distribution. Assuming the distributions are known, scores are converted into probabilities using Bayes' rule.

A specific width μ corresponds to a single part classifier, with some accuracy. In the remainder of this section, we report not this value for μ, but rather the area under the precision-recall curve (AUPRC) which it – along with the number of positive and negative examples – induces. For example, in a 40x40x40 grid, $\mu = 3.85$ corresponds to an AUPRC of 0.3.

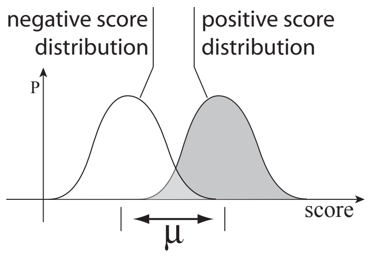

Figure 10. Observation potentials are generated by drawing scores from two distributions. The parameter μ is directly related to each part-classifier's accuracy.

5.2.2 Results

We used our generator to vary four different parameters in the model (default values are shown in parentheses):

- **branching-factor**: the average branching factor in the skeleton graph (default = 2)
- **softness**: each part's softness (default = 0.0)
- σ**(radius)**: the standard deviation of radii in the graph (default = 0)
- μ: the difference in means between the positive score distribution and negative score distribution. We report this value as the area under the associated classifier's precision-recall curve. (default area = 0.3)

In every graph, the average part radius was fixed (at 1 grid point), and each model was constructed of 100 parts.

We used our object recognition framework to search for the optimal layout of parts, given some generated object and observation potentials. As in the previous section, we used both standard belief propagation as well as AggBP, and compared the results. We assumed that part parameters – radius and softness – were known (or learned) by the algorithm. For both AggBP and standard BP, we ran until convergence or 20 iterations. Standard BP occasionally did not converge; in these cases, we took the highest-likelihood solution at any iteration. At each parameter setting, we compute the average error using 20 randomly generated part graphs.

Results from this experiment appear in Figure 11. For each of the four varied parameters we plot the RMS error (a) between standard BP and ground truth, (b) between AggBP and truth, and (c) between AggBP's solution and standard

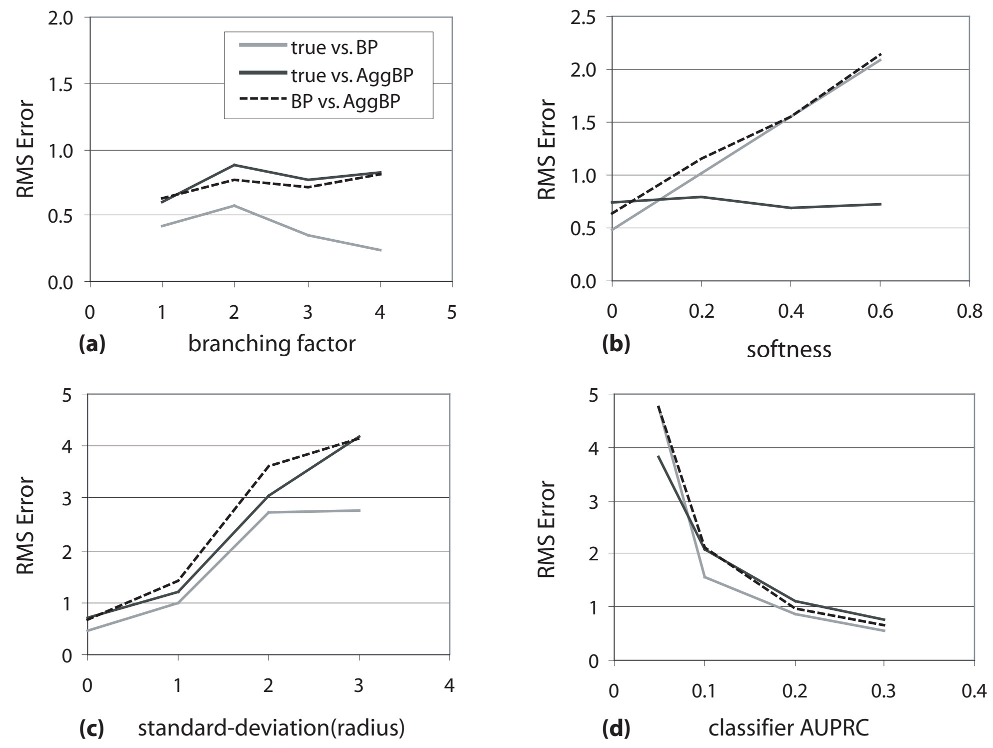

Figure 11. A comparison of our BP approximation with standard BP using the synthetic object generator. While holding other parameters fixed, we vary (a) skeleton branching factor, (b) part softness, (c) radius standard deviation, and (d) classifier AUPRC. We report the RMS error of our algorithm (AggBP) and standard BP (BP) against each other as well as ground truth.

BP's. Runtime and memory usage is almost identical to the previous experiment.

For two of the varied parameters – graph branching factor and classifier AUPRC (Figures 11a and 11d) – the solutions returned by the two methods are of comparable accuracy. Even though the solutions themselves may be quite different, they are both equally close to ground truth. Figure 11c shows that our algorithm performs reasonably well as the radii of the model's parts are varied more and more. The performance of the two algorithms is similar until the standard deviation of part radii is increased to three times the radius. Larger variations could be handled by clustering objects into multiple groups based on radius, approximating messages to each group.

The most interesting result, however, is that in Figure 11c, where the object softness is varied. Increasing object softness allows two objects to move closer than would normally be allowed with some low probability. Here, for any non-zero softness, AggBP finds a more accurate solution than standard BP. The reason for this in unclear, however, it may be due to feedback introduced by this softness, that is dampened by our approximation. When running with a non-zero softness, standard BP fails to converge quite often, giving some support to the idea that our approximation is dampening some feedback loops.

Log likelihood plots, not shown for our synthetic experiments, are very similar to the error plots. These experiments show that our method is clearly valid for a wide variety of model parameters and part topologies. In a large majority of the synthetic experiments, AggBP produced an interpretation that was as good or better than standard BP, in a small fraction of the time.

6 Conclusions and Future Work

We describe a part-based, 3D object recognition framework, well suited to mining detailed 3D image data. We introduce AggBP, a message approximation and aggregation scheme that makes belief propagation tractable in large and highly connected graphs. Using a message-approximation similar to that of mean-field methods, we reduce the number of message computations at a single node from many to just a few. In the fully connected graphs used by our object-recognition framework, we reduce the running time and memory requirements for an object with N parts from O(N^2) to O(N). Additionally, we describe an efficient probability representation based on Fourier series. Experiments on a 3D vision task arising from x-ray crystallography shows that using these improvements produce solutions as good or better then standard BP. Synthetic tests show that AggBP is accurate under a variety of object types with various part topologies, almost always producing a solution as good or better than standard BP.

It is unclear why AggBP should sometimes produce more-accurate results than standard BP. Our approximate-message computation ignores a term that serves to avoid feedback, and makes the method exact in tree-structured graphs. However, in graphs with loops, such feedback is unavoidable (through the loops of the graph). For some types of edge potentials, ignoring this term produces a more-accurate approximation, perhaps by dampening some of these feedback loops inherent in loopy belief propagation. Further investigation into this is needed.

In the future, we would like to take a more dynamic approach to message aggregation. For example, in the protein backbone-tracing task, AggBP's approximation error is highest along edges connecting amino acids that are nearby in space. If we could accurately predict which amino acids are close (in space) as BP iterates, we could *precisely* compute messages between these pairs of nodes, and *approximately* compute messages along other edges.

The results using AggBP illustrate our techniques are useful in the automatic interpretation of complex 3D image data. The shortcuts we introduce drastically increase the size of problems on which BP is tractable. In one real and one synthetic dataset, we produces accurate results with significant CPU and storage savings over standard BP. Our algorithm appears to be a powerful tool for mining large images.

Acknowledgements

This work is supported by NLM grant 1R01 LM008796 and NLM Grant 1T15 LM007359.

References

[1] J. Coughlan and S. Ferreira. Finding deformable shapes using loopy belief propagation. *Proc. ECCV*.

[2] F. DiMaio, J. Shavlik and G. Phillips (2006). A probabilistic approach to protein backbone tracing in electron density maps. *Proc. ISMB*.

[3] A. Doucet, S. Godsill and C. Andrieu (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*.

[4] P. Felzenszwalb and D. Huttenlocher (2000). Efficient matching of pictorial structures. *Proc. CVPR*

[5] P. Felzenszwalb and D. Huttenlocher (2004), Efficient belief propagation for early vision. *Proc. CVPR*.

[6] B. Frey (1998). *Graphical Models for Machine Learning and Digital Communication*. MIT Press.

[7] T. Heskes (2004). On the uniqueness of loopy belief propagation fixed points. *Neural Comp.*, 16.

[8] A. Ihler, E. Sudderth, W. Freeman, and A. Willsky (2004). Efficient multiscale sampling from products of gaussian mixtures. *Proc. NIPS*.

[9] M. Isard (2003). PAMPAS: Real–valued graphical models for computer vision. *Proc. CVPR*.

[10] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul (1999). An introduction to variational methods for graphical models. *Machine Learning*.

[11] D. Koller, U. Lerner, and D. Angelov (1999). A general algorithm for approximate inference and its application to hybrid Bayes nets. *Proc. UAI*.

[12] S. Kullback and R. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*.

[13] D. MacKay and R. Neal (1995). Good codes based on very sparse matrices. *Cryptography and Coding: 5th IMA Conference*.

[14] K. Murphy, Y. Weiss, and M. Jordan (1999). Loopy belief propagation for approximate inference: An empirical study. *Proc. UAI*.

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

[16] G. Rhodes (2000). *Crystallography Made Crystal Clear*. Academic Press.

[17] B. W. Silverman (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall.

[18] E. Sudderth, A. Ihler, W. Freeman, and A. Willsky (2003). Nonparametric belief propagation. *Proc. CVPR*.

[19] E. Sudderth, M. Mandel, W. Freeman, and A. Willsky (2004). Visual hand tracking using nonparametric belief propagation. *MIT LIDS Technical Report 2603*.

[20] S. Tatikonda and M. Jordan (2002). Loopy belief propagation and Gibbs measures. *Proc. UAI*.

[21] Y. Weiss (1996). Interpreting images by propagating Bayesian beliefs. *Proc. NIPS*.

[22] Y. Weiss (2000). Correctness of local probability propagation in graphical models with loops. *Neural Comp.*, 12.

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

[24] J. Yedidia, W. Freeman and Y. Weiss (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Trans. on Information Theory*.