APPEARS IN THE PROCEEDINGS OF THE SIXTH IEEE CONFERENCE ON DATA MINING (ICDM), 2006

Belief Propagation in Large, Highly Connected Graphs for 3D Part-Based Object Recognition

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. An efficient inference algorithm – based on belief propagation (BP) – finds the optimal layout of parts, given some input image. We introduce AggBP, a message aggregation scheme for BP, in which groups of messages are approximated as a single message. For objects consisting of N parts, we reduce CPU time and memory requirements from O(N^2) to O(N). We apply AggBP on synthetic data as well as a real-world task identifying protein fragments in three-dimensional images. These experiments show that our improvements result in minimal loss in accuracy in significantly less time.

1 Introduction

Several recent publications have explored the use of part-based models for recognizing generic objects in images [3, 13, 6]. These models represent physical objects as a graph: a collection of vertices ("parts") connected by edges enforcing pairwise constraints. An inference algorithm finds the most probable location of each part in the model given the image. Most previous work has only considered simple objects with relatively few parts. We present a part-based object recognition framework capable of identifying objects with thousands of parts.

Rich, three-dimensional image data arises in many applications; for example, biological imaging techniques, such as fMRI or confocal microscopy, produce high-quality 3D images of tissues. X-ray crystallography yields a 3D electron density map, a three-dimensional image of a macromolecule. These data sources contain objects comprised of many parts and connected with a complex topology. 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 uses an efficient message-passing inference algorithm based on *belief propagation* [11]. Unfortunately, for large, highly connected graphs, standard BP may not offer enough efficiency. In fully connected graphs, with 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*), which approximates groups of BP messages with a single message. For fully connected graphs, AggBP reduces BP's running time (in a graph with N nodes) from O(N^2) to O(N).

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 finding objects with various part topologies.

2 Part-Based Object Recognition

Following others [3], our framework describes some class of objects using a *pairwise undirected graphical model*. Pairwise undirected graphical models define the joint probability distribution of a set of variables on a graph, as the product of potential functions associated with each *edge* and each *vertex* in the graph.

2.1 Undirected graphical models

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we associate each vertex s with random variable $x_s \in \mathbf{x}$, conditioned on observa-

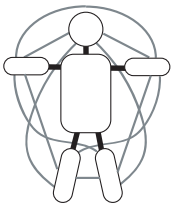

Figure 1. A sample graphical model for recognizing a person in an image. Thicker lines indicate skeletal edges, while thinner lines indicate occupancy edges.

tion variables $\mathbf{y}$. For object recognition, these x_s's describe the 3D position (and possibly orientation) 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 represent the probability of some layout of parts given the image as

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

Usually we are concerned with finding the labels $x_s \in \mathbf{x}$ that maximize this joint probability.

To describe an object in this part-based framework, one provides three pieces of data: a *part graph*, each node's *observation potential*, and each edge's *structural potential*. Given a part graph, these potential functions are typically learned from previously solved problem instances; this is described in more detail in a working paper [2] of ours.

In our object-recognition framework, the *part graph* is fully connected. The reason is best illustrated by example. Figure 1 shows a model used to recognize people in images. A sparsely connected skeleton (the thick edges) connects highly correlated nodes; the corresponding *skeletal potential* may take any form. However, any other part pair – for example, the two arms – have labels that are not completely (conditionally) independent: the parts may not occupy the same 3D space. *Occupancy* edges connecting all other pairs of parts ensure that no two parts in the model occupy the same 3D space; the corresponding *occupancy potential* is typically a step function that is only non-zero when connected parts are sufficiently distant.

2.2 Belief propagation

Given an image and this graphical model, *inference* attempts to find the most-probable location of each of the object's parts in the image. Because our object graph contains loops, exact inference methods will not work. Instead, our framework uses *belief propagation* (BP) [11], a message-passing approximate inference algorithm. BP computes the *marginal probability* over each x_s (each part's location) by passing a series of local messages. 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 $\mathbf{y}$ is usually dropped):

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 $\hat{b}_s(x_s|\mathbf{y}) \approx \sum_{x_1} \dots \sum_{x_{s-1}} \sum_{x_{s+1}} \dots \sum_{x_N} P(\mathbf{x}|\mathbf{y})$

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

$$m^n_{t\to s}(x_s) \propto \int_{x_t} \psi_{st}(x_s, x_t) \times \psi_t(x_t|\mathbf{y}) \times \prod_{u\in\Gamma(t)\setminus s} m^{n-1}_{u\to t}(x_t)\, dx_t \qquad (2)$$

This message can be thought of as indicating where node t expects to find s, based on t's belief. An approximation to the marginal (or *belief*) is given by the product of incoming messages and observation potential ψ_s:

$$\hat{b}^n_s(x_s|\mathbf{y}) \propto \psi_s(x_s|\mathbf{y}) \times \prod_{u\in\Gamma(t)} m^n_{u\to t}(x_t) \qquad (3)$$

In tree-structured graphs, BP is exact. In graphs with cycles, there are no guarantees to BP's correctness; however, BP often produces good estimates in practice [10]. Our working paper [2] discusses related work in this area.

3 AggBP: Scaling Belief Propagation

When modeling objects with hundreds of parts, the number of BP 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. In the undirected graphs used for 3D object recognition, pairs of

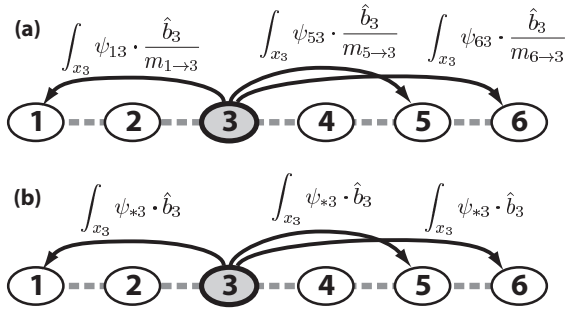

Figure 2. AggBP approximates (a) a group of messages with (b) a single message.

nodes along *skeleton* edges are highly correlated; messages along these edges have high information content. 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, then, we can make some approximations.

Formally, BP's message update, given by Equation 3, can be alternately written:

$$m^n_{t\to s}(x_s) \leftarrow \alpha_1 \int_{x_t} \psi_{st}(x_s, x_t) \times \frac{\hat{b}^n_t(x_t|\mathbf{y})}{m^{n-1}_{s\to t}(x_t)} dx_t \quad (4)$$

The denominator in the above, $m^{n-1}_{s\to t}(x_t)$ is a term to avoid double-counting, making the method exact in tree-structured graphs. In loopy graphs, such double-counting is unavoidable. Along occupancy edges this denominator carries little information, and AggBP drops it:

$$m^n_{t\to s}(x_s) \leftarrow \alpha_2 \int_{x_t} \psi_{st}(x_s, x_t) \times \hat{b}^n_t(x_t|\mathbf{y})\, dx_t \quad (5)$$

The key advantage of doing this is that, if the structural potential ψ_{st} is identical along all occupancy edges, then all occupancy messages outgoing from a single node are identical. We refer to these approximate messages as $m_{t\to *}(x_*)$, illustrated for a chain in Figure 2.

This approximation reduces the number of occupancy messages computed from O(N^2) to O(N). However, *updating* the belief for some part still requires multiplying all the incoming occupancy messages; the running time is still O(N^2). To get around this, we send these aggregate messages to a central accumulator:

$$ACC(x_*) \leftarrow \prod_{t=1}^{N} m_{t\to *}(x_*) \quad (6)$$

This accumulator is then used to update each node's belief in a constant number of operations, reducing AggBP's runtime to O(N). For a chain:

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

Algorithm 2: Aggregate belief propagation.

initialize accumulator ACC, messages m to 1
while $\hat{b}$*'s have not converged* **do**
  foreach *part* $s = 1 \ldots N$ **do**
    $ACC \leftarrow ACC / m^{n-1}_{s\to *}$
    $\hat{b}_s(x_s|\mathbf{y}) \leftarrow \psi_s \times ACC$
    foreach *part* $t = 1 \ldots N$ **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 agg. msg., send to accumulator
    $m^n_{s\to *}(x_s) \leftarrow \int_{x_t} \psi_{*s} \times \hat{b}^n_s(x_s|\mathbf{y})$
    $ACC \leftarrow ACC \times m^n_{s\to *}$
  end
end

Algorithm 2 gives a pseudocode overview of AggBP. The key difference from Algorithm 1 is in the inner loop, "if s is a skeleton neighbor of t." In our object recognition framework, the skeleton graph is sparsely connected, and this loop is rarely entered.

Finally, when the occupancy edges all have *different* potential functions, AggBP can still take advantage of this approximation, with additional approximation error. In this case, AggBP computes the broadcast message from a part t using the *average* potential function outgoing from t:

$$m^n_{t\to *}(x_*) \leftarrow \alpha \int_{x_*} \frac{\sum_{u=1}^{N} \psi_{tu}(x_t, x_*)}{N} \times \hat{b}^n_t(x_t) dx_t \quad (7)$$

4 Experiments

In this section, we compare the standard 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., while our synthetic dataset

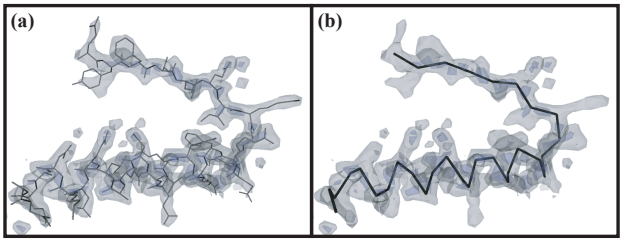

Figure 3. Given a protein's amino-acid sequence and a density map, (a) interpretation finds each atom's position. (b) A backbone trace finds a key atom in each amino acid.

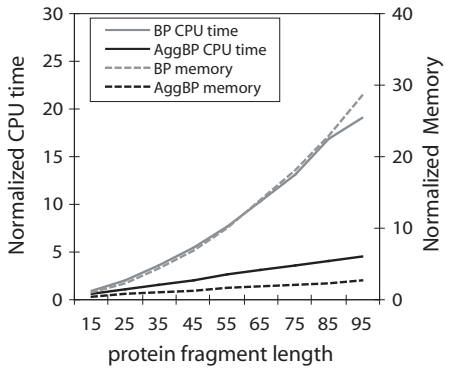

Figure 4. A comparison of memory and CPU time usage between our approximate-BP and standard BP.

allows us to explore recognizing objects with more complex graph topologies.

4.1 Protein fragment identification

One application for object recognition arises from x-ray crystallography. In determining the three-dimensional structure of a protein, crystallography produces a three-dimensional image of the protein, known as an *electron density map*. As illustrated in Figure 3, finding all the atoms (or *interpreting*) this map [12] is the final time-consuming step of x-ray crystallography. Alternatively, a *backbone trace* focuses instead on locating a key carbon atom – the alpha carbon, or C_α – contained in each amino acid. We use AggBP to automatically determine a 3D backbone trace given an electron density map and a protein sequence.

Details about this task, empirical results, and comparison to other methods is found in previous work by this paper's authors [1, 2]. Here we briefly describe backbone tracing using our object-recognition framework. We construct a graph where 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_α, and four internal rotational parameters q_s. Probability distributions over Cartesian space are represented using a discretized probability density estimate.

Protein-specific structural and observation potential functions are learned from previously solved structures [1]. 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. Edge potential functions are of two basic types: *Skeletal* edges run along the linear amino-acid chain, while *occupancy* edges connect all non-adjacent pairs of amino acids, ensuring they don't occupy the same 3D space.

4.1.1 Results

Five density maps were provided by crystallographer George Phillips, at UW-Madison. We convoluted the maps with a Gaussian to simulate a poor-quality (3Å) density map, on which other automated interpretation methods produce poor results [1]. For each map, we were provided the amino-acid sequence and the "true" crystallographer-determined solution.

Using these maps, we compare standard BP inference (ExactBP) to AggBP accuracy at tracing the protein backbone. ExactBP was unable to scale to the entire protein, so we considered locating fragments of 15 to 65 amino acids. Normalized CPU time per iteration and memory usage of the two techniques are illustrated in Figure 4; normalization sets exact-BP's time and memory usage locating a 15 amino-acid fragment to 1.0 (in an average-sized protein, about 200 MB and 120 sec, respectively).

Results from this experiment appear in Figure 5. We plot two different metrics – RMS deviation and log-likelihood of the maximum-marginal interpretation – as a function of iteration. The solutions found by these two methods differ, however, in terms of RMS error versus the true trace, both produce equally accurate traces. More interestingly, Figure 5b shows the log-likelihood of the maximum-marginal interpretation. Under this metric, AggBP produces a better solution. Figure 5c 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 6. 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.

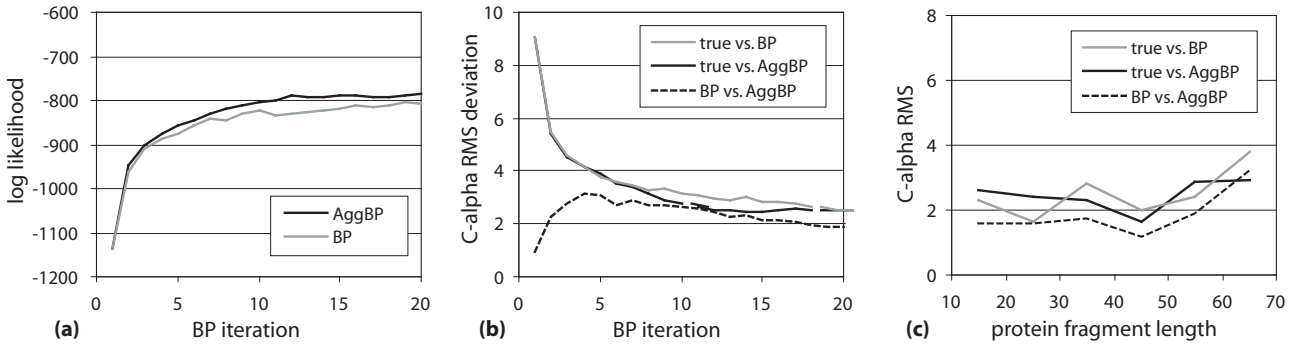

Figure 5. AggBP and ExactBP's error at each iteration, using (a) RMS deviation and (b) log-likelihood of the interpretation. Additionally, (c) shows RMS error as a function of protein fragment size.

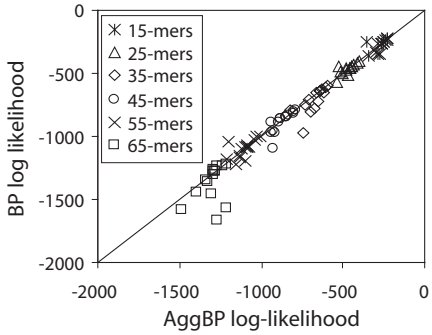

Figure 6. A scatterplot showing – for each of the 90 target fragments – the log likelihood of AggBP's trace versus ExactBP's trace.

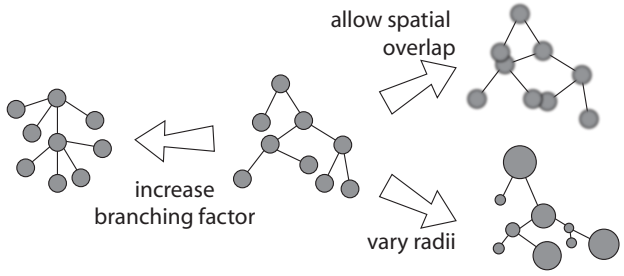

Figure 7. An illustration of parameters variable in our graph generator.

4.2 Synthetic object recognition

The previous section illustrated performance under a limited topology: the skeletal structure is a chain, and all occupancy potentials were identical. In this section, we construct a synthetic object generator that explores AggBP's performance identifying objects with varying topologies.

4.2.1 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 generator lets us vary graph topology and individual part parameters, as in Figure 7. The generator constructs objects with a predefined number of parts, in a tree-structured skeleton. Given some branching factor, the skeleton is randomly assembled from the parts. All part pairs not connected in the skeleton are connected with occupancy edges.

Each part is given a *radius* and a *softness*, from which the structural potentials are derived. Parts directly connected in the skeleton have a distance exactly equal to the sum of their radii, while no other part pairs may be closer than the sum of their radii. The softness parameter allows this occupancy constraint to be violated with low probability.

Our testbed generator also artificially generates observation potentials ψ_{obs}, the *prior* probability distribution of each part's location in 3D space. This would normally be generated by a pattern matching algorithm. Our object generator assumes we have a classifier with an area under the precision-recall curve (AUPRC) of 0.3. Further discussion of this generator appears in our working paper [2].

4.2.2 Results

Our generator varies three different model parameters:

- **branching-factor**: the average branching factor in the skeleton graph (default = 2)
- **softness**: each part's softness (default = 0)
- σ(**radius**): the standard deviation of radii (default = 0)

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

We use 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 compare AggBP versus ExactBP. We assumed that part parameters – radius and softness – are known (or are learned) by the algorithm. We run until convergence, for a maximum of

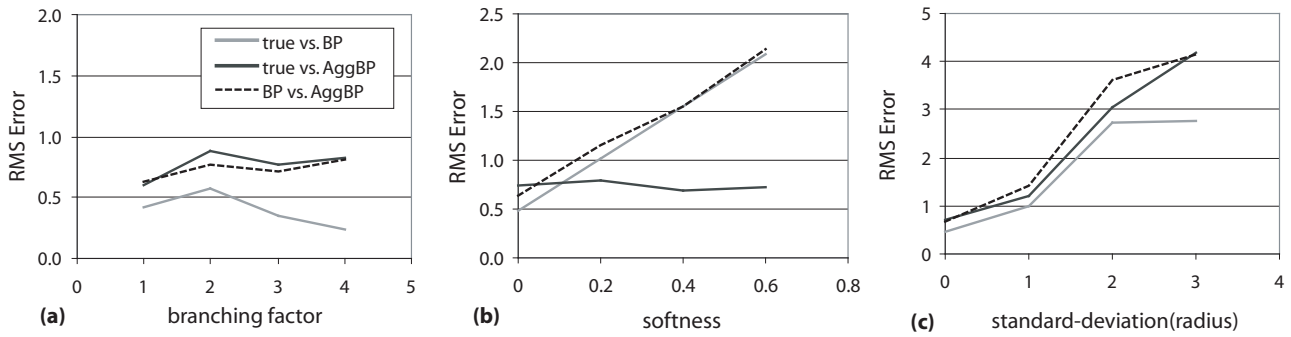

Figure 8. A comparison of ExactBP and AggBP on synthetic data. We report error as we vary (a) skeleton branching factor, (b) part softness, and (c) radius standard deviation.

20 iterations, taking the highest-likelihood solution at any iteration, and averaging over 20 random part graphs.

A comparison of the errors from ExactBP and AggBP's interpretations appears in Figure 8. Figure 8a shows both methods are equally accurate under varying branching factors. Figure 8c shows that small to moderate variance in object radii is handled well by AggBP, even though the aggregation ignores these variations. The most interesting result, however, is that in Figure 8b, where the object softness is varied. Here, for any non-zero softness, AggBP finds a more accurate solution than standard BP.

It is unclear why AggBP should sometimes produce more-accurate results than standard BP. AggBP ignores a term that serves to avoid feedback – feedback unavoidable in graphs with loops. It seems that in some cases, ignoring this term produces a more-accurate approximation, perhaps by dampening these feedback loops inherent in loopy belief propagation. Even without the improved accuracy, the computational savings makes AggBP's approximation an important technique for mining large images.

5 Conclusions

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 BP tractable in large, highly connected graphs. In the fully connected graphs used by our object-recognition framework, we reduce the runtime and memory requirements in an N-node graph from O(N^2) to O(N). Experiments on a 3D biological vision task as well as synthetic data show that AggBP produces solutions as good or better then standard BP.

Acknowledgements

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

References

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

[2] F. DiMaio and J. Shavlik (2006). Improving the efficiency of belief propagation in large, highly connected graphs. *Working Paper 06-1, UW ML Research Group*.

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

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

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

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

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

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

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

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

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

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

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

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