$(BP)^2$: Beyond Pairwise Belief Propagation Labeling by Approximating Kikuchi Free Energies

Ifeoma Nwogu and Jason J. Corso
Department of Computer Science and Engineering
University at Buffalo, SUNY
Buffalo, NY
{inwogu,jcorso}@cse.buffalo.edu

Abstract

Belief Propagation (BP) can be very useful and efficient for performing approximate inference on graphs. But when the graph is very highly connected with strong conflicting interactions, BP tends to fail to converge. Generalized Belief Propagation (GBP) provides more accurate solutions on such graphs, by approximating Kikuchi free energies, but the clusters required for the Kikuchi approximations are hard to generate. We propose a new algorithmic way of generating such clusters from a graph without exponentially increasing the size of the graph during triangulation. In order to perform the statistical region labeling, we introduce the use of superpixels for the nodes of the graph, as it is a more natural representation of an image than the pixel grid. This results in a smaller but much more highly interconnected graph where BP consistently fails. We demonstrate how our version of the GBP algorithm outperforms BP on synthetic and natural images and in both cases, GBP converges after only a few iterations.

1. Introduction

Belief propagation (BP) and its variants are established as viable methods for approximate inference on graphical models, which could also be formulated in terms of optimizing a cost function. The cost function to be optimized corresponds to a distribution that factorizes as a product of terms over cliques of the graph. The optimization problem is therefore to find the *maximum a posteriori* (MAP) configuration of the graph, which depends critically on the structure of the underlying graph [18]. For a graph without cycles (a chain or tree graph) BP computes the MAP configuration efficiently by updating a measure of the marginal distribution at each node, conditioned on information from its neighbors (passing messages from one node to another). On chains and tree graphs, message updates converge after a finite number of steps [13].

Because BP can be fast [2] and stable (does not suffer from high variance), it is fast becoming the method of choice for inference. Therefore, there is a need to continue to better understand and improve the process. BP has been known to produce inaccurate results or may even fail completely when applied on highly connected graphs with strongly conflicting interactions [21]. But in spite of these drawbacks, in computer vision, especially in low-level vision, with cycles existing on the pixel lattice, BP has been used successfully in solving a variety of problems. These include stereo, super-resolution, denoising, inpainting etc., [2, 3, 9, 25]. BP has also been used quite successfully in mid-level vision, in tracking human body parts [5, 17]. On closely investigating the reason for its success in these fields we observed that the modeling of human parts often resulted in a tree-like graph as shown in figure 1 and also, low level image processing occurs either on the square pixel lattice or variants of the lattice such as square pixel grids, using 4-connectivity. But message propagation is done often first in one direction (say horizontally), and then in the other direction. For this reason, pairwise potentials [4] often suffice, although improvements can still be made by using higher order potentials as demonstrated by Lan *et al*. [7] for image denoising on the regular pixel lattice. Since direct pairwise potentials on the nodes of a simple graph are characteristic of the Bethe energy function, the Bethe free energy approximation solution performs reasonable well when applied to these class of problems.

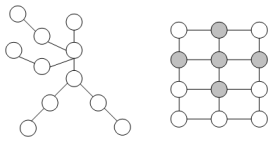


Figure 1. Left is an example of a graph of the human body parts; right is a pixel lattice showing interaction via 4-connectivity

In our vision application, we do away with the regu-

lar pixel lattice and represent an image with more meaningful statistics. This results in a graph with significantly fewer nodes and more natural neighborhood relationships between the nodes. The resulting graph is also more highly connected (with tightly coupled loops) than the regular pixel grid; hence, the standard BP algorithm is not guaranteed to converge on such a graph. A variant of BP known as *Generalized Belief Propagation* (GBP) [23] is therefore used to efficiently find a local maximum of the posterior probability in the Markov network defined over our irregular graph.

Our goal in this paper is to extend the use of BP algorithms in computer vision by implementing GBP which is better suited for inference on highly connected graphs. *Cluster graphs* are a natural representation of GBP algorithms (for optimizing Kikuchi free energies), however, the method of clustering the nodes in a regular graph to create a cluster graph is not a well defined process. This is primarily because the process of generating clusters on an undirected graph exponentially increases the size of the graph (during the triangulation process). We present a systemic, tractable approach to building cluster graphs and demonstrate the superior performance of GBP (applied on cluster graphs) over standard BP (applied on regular pairwise graphs), by using statistical inference to label images (natural and synthetic) of varying textures, colors and noise levels.

In Section 2 we discuss the theoretical background of BP and the extension to GBP using a hypergraph representation. Section 3 explains how GBP cluster graphs can be algorithmically generated from regular graphs. In section 4 we discuss how the MRF model is constructed for a highly connected graph and we show how inferences on the cluster graphs can be used for labeling image regions in section 5.

2. Beliefs, messages and hypergraphs

2.1. Standard BP on a regular graph

We consider a regular graph $G = (V, E)$, with V being the set of all variable nodes $i \in V$, corresponding to hidden states $x = x_1, \cdots, x_n$. If each hidden node is connected to an observed node y_i and $y = \{y_i\}$, then the joint probability function over the entire graph is given by

$$p(x_1, x_2 \cdots x_n | y) = \frac{1}{Z} \prod_i \phi_i(x_i, y_i) \prod_{ij} \psi_{ij}(x_i, x_j) \quad (1)$$

where $\phi_i(x_i, y_i)$ is the local "evidence" for node i and $\psi_{ij}(x_i, x_j)$ is the state transition matrix between nodes i and j and Z is a normalization constant. They are both referred to as the compatibility functions. We can write $\phi_i(x_i)$ as shorthand for $\phi_i(x_i, y_i)$.

The standard BP update rules are:

$$m_{ij}(x_j) \leftarrow k \sum_{x_i} \psi_{ij}(x_i, x_j)\phi_i(x_i) \prod_{k \in \mathcal{N}(i) \setminus j} m_{ki}(x_i) \quad (2)$$

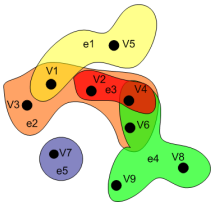
$$b_i(x_i) \leftarrow k\phi_i(x_i) \prod_{k \in \mathcal{N}(i)} m_{ki}(x_i) \quad (3)$$

where m_{ij} denotes the message node i sends to node j, k is a normalization constant, $\mathcal{N}(i) \setminus j$ refers to all the neighbors of i, except j and b_i is the approximate marginal posterior probability (or belief) at node i.

To extend the notion of message updates to Kikuchi approximations, we present a GBP algorithm described in more detail in section 2.3. The GBP algorithm propagates messages across clusters of nodes rather then only on nodes as in standard BP. GBP reduces to standard BP when every cluster in the graph strictly consists of two connected nodes.

2.2. Hypergraphs

Hypergraphs are a generalization of graphs, such that hyperedges can connect any number of nodes, as illustrated in the image on the left. Where a regular graph is a set of nodes and a binary relation (adjacency) between the nodes, a hypergraph extends the adjacency relationship to more than pairwise.



We consider a hypergraph $H = (V, h)$, where V is a set of vertices and h is a set of "hyperedges" between the vertices.

When the cardinality of the hyperedges of H is at most 2, the hypergraph is reduced to an ordinary graph whose hyperedges are regular graph edges over the same set of vertices. Our goal is to convert our regular graph G into a connection of clusters, represented by the hypergraph structure. The relations between the hypergraph and message updates are described in section 2.3.

2.3. Message updates across clusters

Yedidia *et al*. [22] showed that the **Kikuchi free energy (cluster variational method), is the value computed by approximating free energy over clusters of nodes**. It subsumes the mean field and Bethe approximations as special cases with cluster size 1 and 2. As mentioned earlier, the stationary points of Kikuchi approximation correspond to the fixed points of generalized belief propagation, a variant of the standard BP. Increasing the basic cluster size yields better approximation to the posterior marginals. However, arbitrarily large clusters can pose problems as the size of state space increases exponentially with the cluster size.

The GBP algorithm is based on messages propagating information from one region (or cluster of nodes) to another. The goal of our GBP algorithm is to create a hypergraph whose nodes are clusters and whose hyperedges exist only between clusters. The nodes existing in the intersection of clusters are called separators, and are also known in factor graphs as *factors*.

A cluster is defined as a subset of the nodes in the original regular graph G. To distinguish between the nodes of the original graph and the hypergraph, we will refer to the nodes of G as *variables*. A cluster r_β is therefore the set of variables $\{i|i \in \beta\}$ together with the the set of factors $\{a\}$ that contain some or all of the variables in β, $\{a|\mathcal{N}_a \subseteq \beta\}$, where $\mathcal{N}_a$ is the neighborhood of a.

A cluster r_α is a parent of r_β if r_α occurs first in the graph and r_β contains one or more nodes existing in r_α, i.e. there is a directed edge from r_α to r_β. Conversely, r_β is a child of r_α. A cluster r_α is an ancestor (Anc) of r_β if there is a directed path from r_α to r_β. Conversely, r_β is a descendant (Dec) of r_α.

We use a shorthand for the terms involving r_α so that $b_{r_\alpha}(x_{r_\alpha}) \doteq b_\alpha(x_\alpha)$. The hyperedges of the hypergraph are involved with message passing in the GBP setup. First, the pseudo-marginal conditional distribution, (or the belief) at a cluster $b_\alpha(x_\alpha)$ can be computed as:

$$b_\alpha(x_\alpha) = \frac{1}{Z_\alpha} \prod_{a \in r_\alpha} \psi_a(x_a) \prod_{\substack{r_\gamma \in \mathcal{U}_\alpha; \\ r_\beta \in (r_\alpha \cup Dec(r_\alpha))}} m_{\gamma\beta(x_\beta)} \quad (4)$$

where ψ_a is the corresponding clique potentials, $\mathcal{U}_\alpha$ is the set of cluster that consist of the parents of r_α and all the parents of the descendants of r_α excluding cluster in $r_\beta \in (r_\alpha \cup Dec(r_\alpha))$.

To compute the message update between a cluster r_β and its child cluster r_δ, we first compute the beliefs at $b_\alpha(x_\alpha)$ and $b_\delta(x_\delta)$ using equation (4) above. The updates can then be computed as:

$$m_{\alpha\delta}(x_\delta) \leftarrow \frac{\sum_{x_\alpha \setminus x_\delta} b_\alpha(x_\alpha)}{b_\delta(x_\delta)} \cdot m_{\alpha\delta(x_\delta)} \quad (5)$$

Messages can either be initialized randomly or to one and they are then updated iteratively as the algorithm progresses towards convergence.

3. Constructing GBP clusters

3.1. Some graph-theoretic background

Constructing the clusters for GBP inference requires the clique graph of G. Generating the clique graph of our underlying graph G guarantees that we meet the clustering conditions necessary for Kikuchi approximations.

The first step of generating the clique graph is to *triangulate* G [19]. Triangulation involves adding extra edges to G until there are no cycles of greater than three variables in the graph. Given a graph G and an ordering α on its vertices the standard way of computing a triangulation is the Node Elimination method by Parter [12]. Repeatedly choose the next vertex x in order α, and add the edges that are necessary to make the neighborhood of x into a clique in the remaining graph, thus making x *simplicial* in the resulting graph, before deleting x. The edges that are added at each step altogether define the *fill* edges of this process. The triangulated graph obtained by adding this fill to the original graph G is denoted by G^+_α. We refer to such graphs as *simplicial filled graphs*. Different orderings of the input graph result in different simplicial filled graphs. An ordering α on G is called a perfect elimination ordering (PEO) if $G^+_\alpha = G$. Consequently, α is a PEO of G^+_α. It has been shown that triangulated graphs are exactly the class of graphs that have perfect elimination orderings; hence all simplicial filled graphs are triangulated.

Simplicial filled graphs are not generally minimal. The size of the introduced fill edges depends on the order in which the vertices are processed by the elimination routine. A brute force search for the best elimination order has a complexity in time and space of $\mathcal{O}(k^n)$; where $k = |x_i|$ and n is the number of vertices in the graph. The Node elimination process has a complexity of $\mathcal{O}(n^2 k^{w+1})$ where w is a term known as the *tree width* induced by the elimination ordering. Because fill edges grow exponentially, these methods become impractical as the graph sizes get larger.

Lekkerkerker and Boland [8] introduced the concept of *LB-simplicials* where a vertex is LB-simplicial if all its substars are cliques; a substar S of x is a subset of $N(x)$ where $S = N(C)$. C is a connected component of $G(V \setminus N[x])$. $N_G(x)$ is the neighborhood of x in G. We omit subscript G when the graph context is obvious. $N_G([x]) = N_G(x) \cup x$. In figure 2(a), the substars of node 0 are $\{1, 3\}$ and $\{5, 7\}$.

We adopt an LB-Triangulation process [6] because the triangulated graph obtained from this method is linear with the size of the graph; complexity for LB-Triangulation is $\mathcal{O}(nm)$ where m is the number of edges in the original input graph. In practice, it results in significantly faster processing times with much fewer fill edges, making the algorithm very useful for low-to-mid level computer vision problems where the graph sizes can be large.

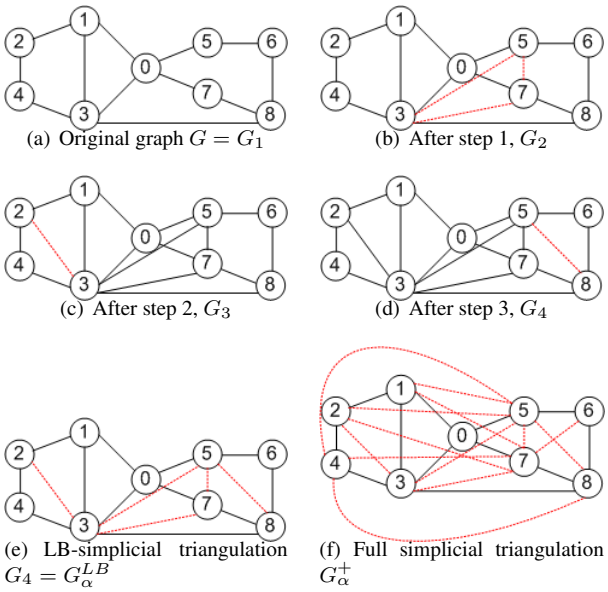
3.2. Computing the minimal graph triangulation

Lekkerkerker and Boland [8] state that a graph is triangulated iff every vertex is LB-simplicial. Based on this characterization, a *minimal triangulation* of a graph can be obtained by forcing every vertex into being LB-simplicial by locally adding edges.

Figure 2 shows how the LB-Triangulation algorithm works on a simple graph. The last row shows the outputs of the two different triangulation processes discussed.

Algorithm 1 LB-triangulation

Input: a graph G(V,E)
Output: Minimal triangulation of G i.e. G_α^{LB}
Begin
  foreach $x \in V$ *do*
    compute $N_{G_i}[x_i]$
    compute the set of connected components of $G_i(V \setminus N_{G_i}[x_i])$
    compute the neighborhood $N_{G_i}(C)$ *for each connected component* C *from previous step*
    construct the fill edges F from $N_{G_i}(C)$
End



(a) Original graph $G = G_1$ (b) After step 1, G_2 (c) After step 2, G_3 (d) After step 3, G_4 (e) LB-simplicial triangulation $G_4 = G_\alpha^{LB}$ (f) Full simplicial triangulation G_α^+

Figure 2. Steps showing LB-triangulation with fill edges shown in red dotted lines. G_α^{LB} is compared with G_α^+.

Step 1. $N_{G_1}[0] = \{0, 1, 3, 5, 7\}$. Set of connected components of $G_1(V \setminus N_{G_1}[0]) = \{\{2, 4\}, \{6, 8\}\}$. $N_{G_1}(\{2, 4\}) = \{1, 3\}; N_{G_1}(\{6, 8\}) = \{3, 5, 7\}$; $F_1 = \{\{3, 5\}, \{3, 7\}, \{5, 7\}\}$. The resulting graph G_2 is shown in figure 2(b).

Step 2. $N_{G_2}[1] = \{0, 1, 2, 3\}$. Set of connected components of $G_2(V \setminus N_{G_2}[1]) = \{\{5, 6, 7, 8\}, \{4\}\}$. $N_{G_2}(\{5, 6, 7, 8\}) = \{0, 3\}$; $N_{G_2}(\{4\}) = \{2, 3\}$; $F_2 = \{2, 3\}$. The resulting graph G_3 is shown in figure 2(c).

Step 3. Since vertex 2 is already LB-simplicial, we proceed to process vertex 3. $N_{G_3}[3] = \{0, 1, 2, 3, 4, 5, 7, 8\}$. Set of connected components of $G_3(V \setminus N_{G_3}[3]) = \{\{6\}\}$. $N_{G_3}(\{6\}) = \{5, 8\}$; $F_3 = \{5, 8\}$. The resulting graph G_4 is shown in figure 2(d). Since the remaining vertices are now LB-simplicial, the process terminates.

Once the triangulation is processed, we use the standard lexicographic breadth first search method (LBFS), to determine α, the PEO of the LB-triangulated graph.

3.3. Generating the clusters

The end-to-end process for generating the clusters is given as:

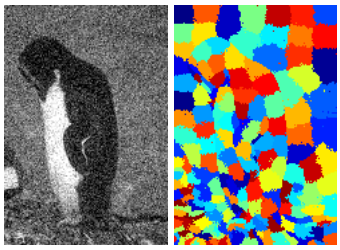
1. Minimally triangulate any planar graph G using LB-triangulation algorithm
2. Compute the elimination ordering α of the triangulated graph
3. Generate the clique graph in the order specified by α
4. The hypergraph structure is used to store the "supernodes" and hyperedges of the clique graph
5. (Optional) Compute the maximum spanning tree of the clique graph (where the edge weight is cardinality of the hypergraph i.e number of separators). This results in a clique tree and converges faster.
6. The supernodes and separators (or factors) in the hypergraph are populated with the appropriate potentials (obtained from the underlying graph).

4. Labeling with GBP on superpixels

In order to label images, they are first abstracted into a *superpixel* representation resulting in a Markov network defined on an irregular graph (not the usual pixel-lattice structure). The resulting irregular graph is then converted into a hypergraph as described in section (3) so that GBP can be used for inference. The potentials or compatibility functions for the nodes in the hypergraph are then computed and the messages are dynamically updated, until convergence. The final marginal probabilities (or beliefs) determine the final label of the superpixels in the image.

A superpixel is a homogenous segment obtained from a low-level, local, coherent grouping of the underlying pixels, which preserves most of the structure necessary for segmentation (an example is shown in figure 3). Due to the arbitrary connectivities of the superpixels, a highly irregular graph is generated when compared to the pixel-grid. But we choose to work with the superpixel grid because we believe that it is representationally more efficient: i.e. constraints exist between entire segments in a superpixel graph, while they only exist for neighboring pixels on the pixel-grid. For a local model such as the MRF model, this property is very appealing in that we can capture longer-range interactions between superpixels segments. Superpixels contain more local information than single pixels, and this puts less of a burden on the Markov network, thus relying more on the local evidence term in the MRF.

The use of superpixels is also computationally more efficient because it reduces the combinatorial complexity of images from hundreds of thousands of pixels to only a few hundred superpixels. Because superpixels are an oversegmentation of the image, there are many different well-tested and reported algorithms that accurately generate superpixels. These include the segmentation by weighted aggregates algorithm (SWA) [15], normalized cuts [16, 24], constrained Delaunay triangulation [11] to mention a few. It is very important to use near-complete superpixel maps to ensure that the original structures in the images are preserved. Therefore, we use the segmentation algorithm, *normalized cuts* [16], which was empirically validated and reported in [14].



c

Figure 3. Example of a noisy natural image with its superpixel representation

The end-to-end process for inferring labels in an image using GPB is given as:

1. Generate the superpixel representation of the underlying image. The Markov network is defined over the graph generated by the superpixel map.

2. Transform the irregular graph to a minimally triangulated graph.

3. Generate the clique graph/tree from the triangulated graph.

4. Pairwise and higher order clique potentials are derived for the model.

5. The hyperedges are populated with their corresponding beliefs based on the nodes and edges of the underlying graph G.

6. Messages are randomly initialized and updated via our message passing rules.

7. Final beliefs at the supernodes and corresponding nodes are estimated. MAP estimation lends itself easily to the labeling problem.

Empirically, we observed that the maximum clique size in our hypergraph representation is 4.

5. Experiments and results

The experiments in sections 5.1 and 5.2 involved labeling grayscale images using a simple measure of the difference between labels, a modification of the Potts model to capture the assumption that labelings should be piecewise constant. This model considered the equality or inequality of labels and we extended this to the case of three and four cliques. The three and four clique potentials were assigned values between 0 and 1 depending on how many nodes in the clique had the same label. The *fill* edges (edges added only for triangulation purposes) were also taken into consideration during the computation of the clique potentials.

In this paper, the primary focus is presenting a tractable systemic approach to implementing GBP, and we demonstrate this with the labeling problem. We use very basic models to compute the clique potentials, but it is possible that much more accurate results can be obtained with using stronger models resulting in more realizable potential functions.

5.1. Toy example using synthetic image

The fist example compares the results of labeling a simple synthetic image using different superpixel structures with BP and GBP techniques. Figure 4a shows the original synthetic image along with the degraded version of the image. Figure 4b shows one simple superpixel representation of the degraded image along with another more complex representation. Figure 4c shows the two graphical structures corresponding to the graphs in figure 4c. Only a portion of both graphs are shown, but it can be seen that one graph is a chain graph while the other is a more highly connected graph with three- and four-cliques. Figure 4d shows the results of applying standard BP on the chain graph after 5 and 18 iterations respectively. Figure 4e shows the result of applying standard BP on the irregular graph again after 5 iterations and 18 iterations respectively. Lastly, Figure 4f shows the result of applying cluster-based GBP on the irregular graph after 2 and 12 iterations.

Comments: GBP converged to a "good" solution as shown in figure 4f, although it is important to mention that the compatibility functions used for (d and e) needed to be different to include three and four-clique potentials. The primary advantage of the GBP inference was that the solution converged and remained the same. GBP by itself does not determine the quality of the final solution.

5.2. Denoising grayscale natural images (with varying levels of noise)

The labeling of the noisy penguin image was done using a similar potential function set up as in 5.2 and the label set was 0-255 discrete labels. The results of GBP were compared with efficient BP [2] and the graphshift inference

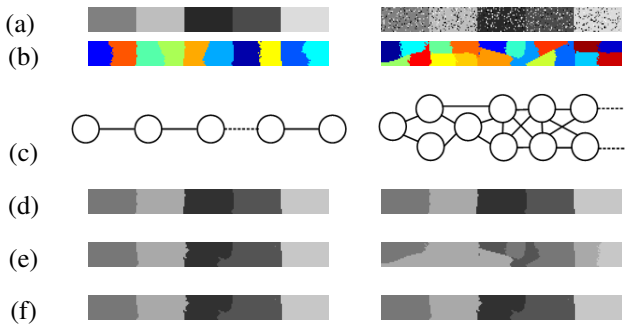


Figure 4. Comparison of the results of labeling a simple synthetic image using different superpixel structures and BP techniques. (a) Original image and its noisy version
(b) Two different superpixel representations of the **noisy** image
(c) Sections of the corresponding graphs for (b)
(d) BP after 5 and 18 iterations, on the chain graph only
(e) BP after 5 and 18 iterations, on the complex graph
(f) GBP after 2 and 12 passes, on the complex graph
(d), (e), and (f) are results of applying BP/GBP on the noisy image

algorithm[1]. We chose to compare with the graph shifts algorithm because it also performs a neighborhood-based optimization and gave very promising results on medical images. The authors provided us with their implementation. The visual results of the comparisons are shown in figure 5 for the penguin image with noise variance $\sigma = 30$. The SSD errors were computed and given in Table 1.

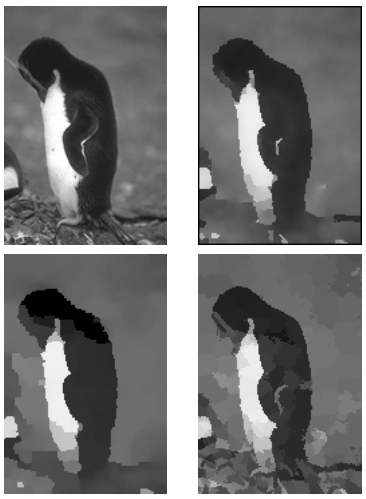


Figure 5. Top-left is original noise-free image; Remaining three images are results of denoising image in figure 3 ($\sigma = 30$). Top-right is result of efficient BP, bottom-left is from graph shifts and bottom right is GBP on superpixels.

Comments: Similar to the observation made in section 5.1, the "goodness" of the final solution is based on the potential functions used as well as the superpixel representation. MAP estimation was used to select the final labels at the nodes, giving the block effect results obtained. As the noise variance increased, GBP with superpixels outperformed the other optimizations.

Variance	Efficient BP	Graph shifts	GBP on superpixels
10	9522824	2172429	3134110
20	9701764	3032000	3897102
30	10043367	6764766	4852791

Table 1. Quantitative comparison of the SSD error between efficient belief propagation, graph-shifts and GBP superpixels.

5.3. Hole-filling for a colored natural image

This example involves a colored natural image with large holes. The superpixel representation proves useful in successfully locating the holes in the image. Attempting to select "near-white" pixels would have returned actual data pixels (non-holes) in areas such as the little boy's crown, the lady's forearm and some of the flowers in the bottom left side of the image. The use of superpixel representation succinctly captures the holes as shown in figure 7.

In order to fill the holes, the near-white superpixels are identified as the regions-of-interest. For each hole, the superpixels surrounding the hole are the labels for that round of the algorithm (we implement an inference algorithm separately for each hole). The image statistic used is the color distribution of the underlying pixels in the superpixel region, i.e the underlying color histogram. The distance between any two superpixels is measured using the Kullback-Leibler divergence (or K-L distance between them). The one-way KL-distance is given by:

$$d^*_{PQ} = \sum_i P(i) log \frac{P(i)}{Q(i)}, \tag{6}$$

d_{PQ} is the mean of d^*_{PQ} and d^*_{QP} after normalization. d_{PQ} is only a measure of the pairwise distance. Our potential function is given as:

$$\psi_{clique} = e^{-\theta(d)} \tag{7}$$

where $\theta(d)$ is an empirical extension of the pairwise KL-distances to larger cliques. For example, in a 3-clique if every KL-distance value is below a pre-set threshold, we assume that all the clique-nodes are the same and the smallest difference in distance values is assigned as the clique distance. A larger value is assigned when only two distance values are below the threshold; when only one or none of the distance values is below the threshold a value is set accordingly. For improved accuracy, the potentials can be learned from training similar natural images.

The process of updating messages across all the cliques in the hypergraph is referred to as *global propagation*. Because messages need to be stored and updated in 3- and 4-clique hypernodes, the method can be computationally expensive. So, for the hole filling problem, we implement a *local propagation* [10] by selecting a subtree involving only k ancestors and descendants of the hypernode containing the hole. Messages are locally propagated only within the subtree. The result of the inference is a histogram of the color distribution for the hole. For completeness, we use the resulting histogram to synthesize texture from the original superpixel (the label owner).

Figure 6. Example of a colored textured natural image with "holes"

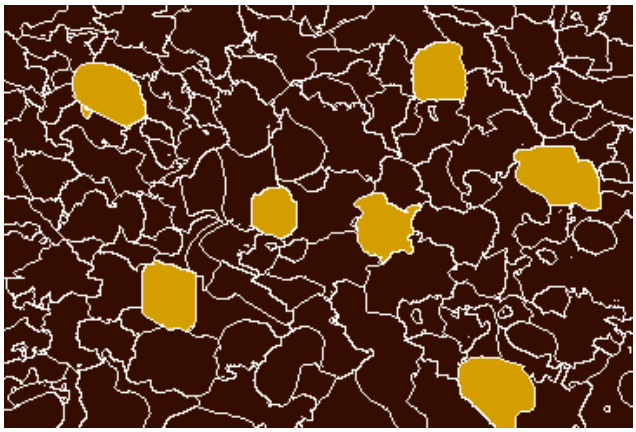


Figure 7. Superpixels accurately capture the holes in the image.

Comments: The holes in figure 6 inherited their labels (based on histograms) from a neighboring superpixel (treated as the source for texture synthesis). Texture synthesis was accomplished by a causal neighborhood and raster scan ordering after the hole-filling labels were assigned via GBP inference [20]. Pixel-wise interactions are better suited to capture fine-detail transitions in hole-filling problems, but in this paper we limit our GBP demonstrations to region-labeling problems. The result obtained for this class of images though, is visually satisfactory. *Note:* the process of texture synthesis is performed after a label has been assigned, and GBP is not used to solve the texture generation problem. In [9] the inpainting optimization problem was solved using loopy belief propagation.

Figure 8. The holes inherit the distribution of neighboring superpixels.

6. Conclusion and future work

In conclusion, we have provided a framework which makes inference tractable even with higher order BP. We extended the representational power of MRFs by introducing the use of superpixels which better capture the rich statistics of natural scenes and performed inference via generalized belief propagation. Even with simple models, the combination of these two methods outperformed the state-of-the-art solutions to problems such as denoising. Using a toy example, we demonstrated empirically how standard BP converges on chain graphs but can tend to diverge from the true solution when loops are introduced. But when inference is performed via cluster-based GBP, the solution converges. We also demonstrated the effectiveness of GBP to solve a variety of region-based vision problems on real images.

In addition, we presented the LB-triangulation algorithm, which reduces the complexity of generating clique clusters from any regular graph, from being exponential to linear (in the size of the graph). The graph size explosion due to triangulation has been one of the limitations of GBP-based algorithms.We present an end-to-end systemic approach to constructing complete cluster graphs that will approximate Kikuchi free energies, when inference is performed on the graph.

In the future, we plan to perform more detailed error and convergence analysis on the different variants of BP encountered (standard BP, cluster-based BP and tree based BP). We believe that the real-world application for this process will be in labeling and parsing three-dimensional

anatomical data (e.g. medical data). Interactions between "supervoxels" of 3D data are much more tightly coupled than in 2D data because of the increase in degrees-of-freedom. Therefore, GBP on supervoxels will likely result in more accurate labeling of 3D anatomical data.

Acknowledgements This work was supported in part by NSF-IGERT grant DGE-0333417. We also thank Yngve Villanger for the countless e-mails and all his assistance with the LB-Triangulation process.

References

[1] J. J. Corso, Z. Tu, A. Yuille, and A. W. Toga. Segmentation of Sub-Cortical Structures by the Graph-Shifts Algorithm. In *Proceedings of IPMI*, 2007. 6

[2] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. In *Proc. IEEE Conf. Comput. Vision And Pattern Recogn.*, 2004. 1, 5

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

[4] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. on PAMI*, 6:721–741, 1984. 1

[5] T. X. Han and T. S. Huang. Articulated body tracking using dynamic belief propagation. In *ICCV-HCI*, pages 26–35, 2005. 1

[6] P. Heggernes and Y. Villanger. Efficient implementation of a minimal triangulation algorithm. *LNCS Proceedings ESA 2002 - 10th European Symposium on Algorithms*, pages 550–561, 2002. 3

[7] X. Lan, S. Roth, D. P. Huttenlocher, and M. J. Black. Efficient belief propagation with learned higher-order markov random fields. In *ECCV (2)*, pages 269–282, 2006. 1

[8] C. Lekkerkerker and D. Boland. Representation of finite graphs by a set of intervals on the real line. In *Fund. Math.*, volume 51, pages 45–64, 1962. 3

[9] A. Levin, A. Zomet, and Y. Weiss. Learning how to inpaint from global image statistics. In *Proc. of IEEE Intl. Conf. Comput. Vision*, page 305, 2003. 1, 7

[10] T. Minka and Y. Qi. Tree-structured approximations by expectation propagation. In *Advances in NIPS 16*. 2004. 7

[11] G. Mori, X. Ren, A. A. Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. In *Proc. IEEE Conf. Comput. Vision And Pattern Recogn.*, volume 2, pages 326–333, 2004. 5

[12] S. Parter. The use of linear graphs in gauss elimination. In *SIAM Review*, volume 3, pages 119–130, 1961. 3

[13] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988. 1

[14] X. Ren, C. C. Fowlkes, and J. Malik. Scale-invariant contour completion using conditional random fields. In *Proc. of 10th IEEE Intl. Conf. Comput. Vision*, volume 2, pages 1214–1221, 2005. 5

[15] E. Sharon, A. Brandt, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. In *Proc. IEEE Conf. Comput. Vision And Pattern Recogn.*, volume 1, pages 469–476, 2001. 5

[16] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 22(8):888–905, 2000. 5

[17] L. Sigal, M. Isard, B. H. Sigelman, and M. J. Black. Attractive people: Assembling loose-limbed models using non-parametric belief propagation. *Advances in NIPS 16*, 2004. 1

[18] M. Wainwright, T. Jaakkola, and A. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, 14(2):143–166, 2004. 1

[19] M. Wainwright and M. Jordan. A variational principle for graphical models. In *chapter 11 in: New Directions in Statistical Signal Processing*. MIT Press, 2005. 3

[20] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH*, pages 479–488, 2000. 7

[21] M. Welling. On the choice of regions for generalized belief propagation. In *Proceedings of the 20th conf AUAI*, pages 585–592, 2004. 1

[22] J. Yedidia, W. Freeman, and Y. Weiss. Bethe free energy, kikuchi approximations, and belief propagation algorithms. MERL Tech. Report., TR2001-16., 2001. http://www.merl.com/reports/docs/TR2001-16.pdf. 2

[23] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51(7):2282–2312, 2005. 2

[24] S. Yu and J. Shi. Segmentation with pairwise attraction and repulsion. In *Proc. of 8th IEEE Intl. Conf. Comput. Vision*, July 2001. 5

[25] C. L. Zitnick and S. B. Kang. Stereo for image-based rendering using image over-segmentation. *Int. J. of Comp. Vis.*, 75(1):49–65, 2007. 1