Loopy Belief Propagation in Image-Based Rendering

Dana Sharon

Department of Computer Science
University of British Columbia

Abstract

Belief propagation (BP) is a local-message passing technique that solves inference problems on graphical models. Though [12] proved BP converges to a unique fixed point on singly connected graphs, researchers have shown that BP can also produce great results on graphs with loops. Most notable are the near Shannon-limit of error-correcting decoding and unwrapping of phase images [13, 2]. Since images can be easily represented as loopy graphs, where graph nodes are associated with pixels or image patches, many image-based rendering researchers have experimented with BP, with promising results. In this paper, I discuss an implementation that uses BP for noise removal in images. The following sections survey other image-based rendering applications of BP.

1 Introduction

Image-based rendering uses existing images of a scene in order to synthesize new images. Some image-based rendering techniques attempt to first infer knowledge of a scene (such as 3D shapes, lighting, transparency), and then use the new information in image synthesis. If 3D geometry is extracted, for example, the scene can be rendered from different viewpoints. Alternatively, if knowledge of transparency is discovered, the transparent objects can be rendered against different backgrounds.

It is reasonable to expect that image statistics (e.g. of intensities or gradients) can aid in the inference of scene information. This expectation arises from the fact that images, in general, tend to have very low variability. For example, the output of a derivative filter applied to a natural image is generally sparse, with a distribution resembling the Laplacian [1]. Belief propagation (BP) is an algorithm that uses prior probabilities of images to infer information about a scene.

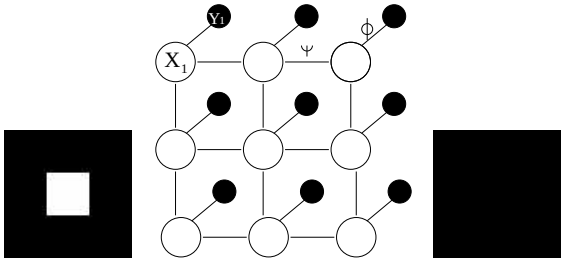


Figure 1: (a) The observed, noisy image. (b) Each pixel in the image is one node in the MRF. (b) The infered scene using BP

In section 2, I describe the BP algorithm in detail. In section 3, I show some results of an implementation of BP in the context of noise removal. Sections 4 to 8 describe other image-based rendering applications of BP. I conclude in section 9.

2 Belief Propagation

Belief propagation (BP) is a local-message passing technique that solves inference problems on graphical models. It has been shown [12] to produce exact inferences on singly connected graphs. Moreover, empirical studies [1, 14, 10] have shown that, although beliefs don't always converge on loopy graphs, when they do - they produce great results. In fact, loopy BP has shown to be the best solution (so far) for turbocodes and phase image unwrapping.[2, 13]. As an example, consider the problem of noise removal. For simplicity, assume pixels can have one of two colors: black, or white. Suppose we have a noisy 3x3 image in which all pixels are black except for one that is white (see Figure 1). The image can be considered evidence of an underlying scene that is the uncorrupted version of the image. In this scene, we could expect all pixels to be black.

Here, the pixels are nodes in a pairwise Markov

random field (MRF). The white circles, x_i, are hidden nodes representing the pixels in the scene, and the black circles, y_i, are the evidence nodes, representing the image pixels. In BP, we define compatibility, or potential functions between neighboring hidden nodes ψ, and between hidden nodes and their corresponding observed nodes ϕ. An appropriate assumption for this particular application is that, in general, neighboring pixels have the same color. This assumption can be coded into the compatibility functions. For example, we set a high compatibility between neighboring pixels that have similar colors, and low compatibility between neighboring pixels with differing colors. We set the local potentials in a similar manner. These potentials are used in messages that are propagated between the pixels to indicate what color each scene pixel should have. In general, we define the joint probability distribution for all the nodes to be

$$P(\{x\}) = \frac{1}{Z}\prod_{ij}\psi_{ij}(x_i, x_j)\prod_{i}\phi_i(x_i) \qquad (1)$$

We wish to maximize $P(\{x\})$, that is, we want to find the most likely state for all the hidden nodes x_i, given all the evidence nodes y_i. In BP, we use beliefs to approximate this probability. The belief $b_i(x_i)$of a node is

$$b_i(x_i) = k\phi_i(x_i)\prod_{k} m_{ki}(x_i) \qquad (2)$$

where the m_{ki} 's are the messages that node i receives from its neighbors. The messages are updated using the following rule:

$$m_{ij}(x_j) = \sum_{x_i}\phi_i(x_i)\psi_{ij}(x_i, x_j)\prod_{k\neq j} m_{ki}(x_i) \qquad (3)$$

In equation 3, the product of messages will not include the one from node j, the node we are passing a message to.

Both the messages, $m_{ij}(x_j)$, and the local potentials, $\phi_i(x_i)$, are vectors whose length corresponds to the number of states that node x_i can be in. The neighbouring potentials, $\psi_{ij}(x_i, x_j)$, are N x M matrices where N is the number of states that x_j can be in, and M is the number of states for node x_i. As can be seen from equation 3, the BP algorithm is exponential in the number of states. Many solutions have been proposed for cases which require continuous state spaces. One solution uses "a hybrid method that allows both good fitting and fast propagation"[4]. Here, the authors represent the states as mixtures of gaussians [3], and then use sampling techniques to reduce computation.

Pearl [12] has proved that for singly connected graphs, the belief at a node will converge to the marginal probability for that node. That is,

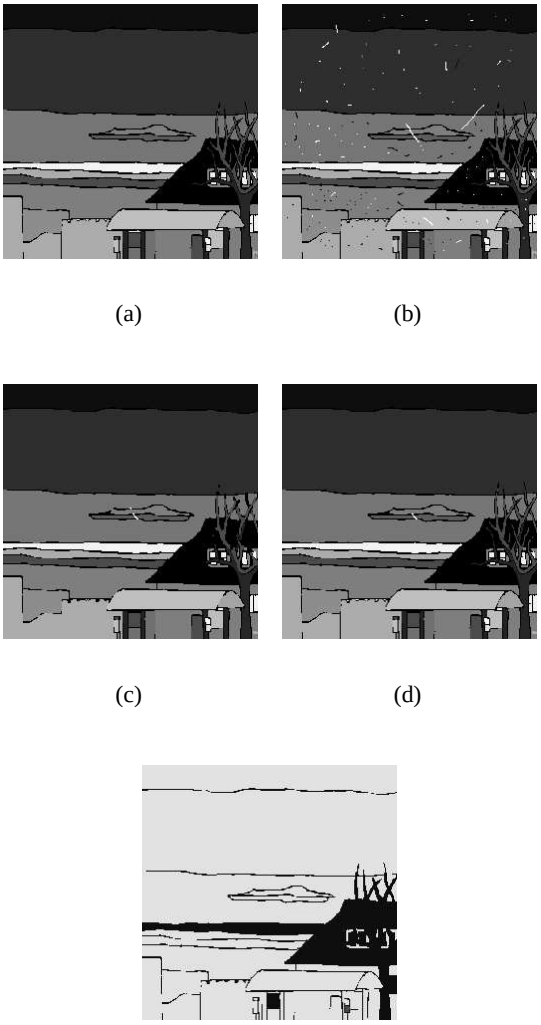
$$b_i(x_i) = k\sum_{x_j/x_i} p(x) = p_i(x_i) \qquad (4)$$

One would think that BP should not work correctly for graphs that contain loops. The messages propagated throughout the graph will end up being counted multiple times, causing evidence previously accounted for to be considered again, as new evidence. However, it turns out that "all evidence is double counted in equal amounts" [15]. In this case, the beliefs will be wrong, but the proportionality of beliefs in different states will be correct. Therefore, we cannot use individual beliefs themselves, but we can still maximize them.

3 Noise Removal using BP

I have implemented a program that used BP to remove noise in images that contain a small number of states (colors). The program takes a noisy image as input, and produces a cleaner version. I find the number of colors in the image and set those to be the states in MRF nodes corresponding to image pixels. Each node stores information about it's neighbours, such as node index, messages to be passed, potentials, etc. I make the simplifying assumption that potentials, both local and neighbouring, are identical for all nodes pairs. However, allowing for different potentials is an easy extension to add.

Figure 2 show results of running BP on a noisy Microsoft painting (b). The original image can be seen in (a), and is courtesy of Ritchie Argue (Imager lab). As can be seen in (c), running standard BP removes most of the noise. All the noise was removed, except for a little peice of the scratch in the cloud. On the other hand, the algorithm fails to around the tree area. The tree branches are so thin, that they are mistaken for noise and are replaced with the black color of the roof or the blue of the sky, which are in the background. BP is very flexible though. We can modify the local compatibilities to have strong beliefs in certain colors. (d) show an improved version of the clean image, where local compatibilities of brown, black, and yellow pixels were set to the 1. For example, if an image pixel is black k, there is very high probability that the corresponding scene pixel should also be black.



(a) (b)

(c) (d)

(e)

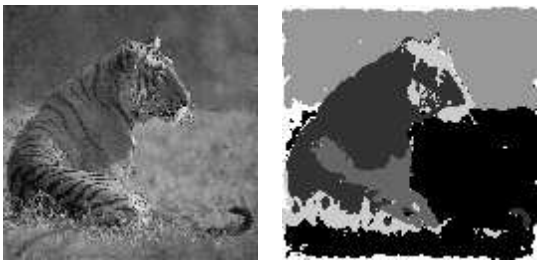
Figure 2: (a) Original image. (b) Noisy version of original (c) Standard BP on noisy image (d) BP with extremely high local potentials for black, brown, and yellow. (e) BP with funky potential functions.

Although this program is meant to be used for noise removal it can be easily extended for use in other applications. For instance, with simple tweaking of compatibilities, we can get interesting images such as the one in Figure 2(e). To produce this result, I simply negated the compatibilities to be very low if neighbors are similar, and very high if they have different colors. Other, less artistic applications, include cleaning disparity maps or image segmentation outputs.

Figure 3 illustrates how BP can be used to clean the output of image segmentations. In (a) we see the original output after segmenting an image of a tiger. Many pixels are misclassified. In (b) some of those pixels are correctly classified. The compatibilities only exist between neighboring pairs of pixels. This makes it hard to remove noise that is several pixels in diameter. A possible solution is to encode furthur assumptions into the potentials. Another solution is constructing a MRF on several scales of the image. For example, we could add a MRF to the current one, that contains image patches, rather than individual pixels. The compatibilities will then have to exist between nodes across scale as well as space. The MRF of image patches will allow BP to propagate the correct states to patches that contain noise of larger diameter.

4 Transparency

A difficult task in computer vision is identifying objects in a scene from image data. If the data contains a transparent object, the task is even harder. Image based rendering could also benefit from knowledge of transparency in an image. For instance, one could extract the transparent object and place it in a new scene. If one describes an image $I(x, y)$ as a composition of two layers, $I(x, y) = I_1(x, y) + I_2(x, y)$, the goal would be to find the best decomposition (i.e., one the separates the semi-transparent objects). With an infinite number of possible decompositions, simply checking all of them is infeasible for a decent sized image. [1] take advantage of the statistics of natural images (i.e. the sparseness of image derivatives) to define prior probabilities over image gradients. The authors then pick as the most probable decomposition, the one the minimizes the number of edges and corners, which can be identified using gradients. The probability of an image is defined in terms of the log



(a) (b)

(c)

Figure 3: (a) ,(b) Segmentation of tiger image. (c) Segmentation after BP

over the image gradient:

$$logP(I_x, I_y) = -logZ - \sum_{x,y}(|\nabla(I(x,y)|^{\alpha} + \eta c(x,y)^{\beta}) \quad (5)$$

Here, c(x,y) is the output of a corner detector. To get the probability of the decomposition, one can just sum the log probabilities of the gradients of the two image layers I_1, I_2. The authors construct a graphical model of the image, where each hidden node in the graph represents the gradient $g_i = (g_{ix}, g_{iy})^T$ of one of the layers. The gradient of the second layer is implicitly defined as $f_i = (I(x,y))^T - g_i$. The joint probability of the gradient of the first layer is given by [1]

$$P(g) = \frac{1}{Z}\mathbf{\Pi}_i \phi_i(g_i) \mathbf{\Pi}_{<ijkl>} \psi_{<ijkl>}(g_i, g_j, g_k, g_l) \quad (6)$$

where the compatibility $psi_{<ijkl>}(g_i, g_j, g_k, g_l)$ is over a 2x2 pixel neighborhood. Notice that the neighboring compatibilities are not pairwise as in the noise removal example. Thus, we could not simply construct a pairwise MRF and run BP as before. In fact, the more suitable graphical model for the transparency application is a factor graph as shown in figure 4. The circle nodes are the gradients, and the squares are clusters of the surrounding 4 nodes. The message update rules are more complicated for the factor graph, as one needs to consider two types of messages: from clusters to nodes, and from nodes to clusters. However, as is stated in [9], the two graphical models (factor graphs and MRFs) are mathematically equivalent and one can easily convert back and forth.

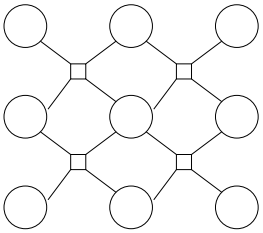
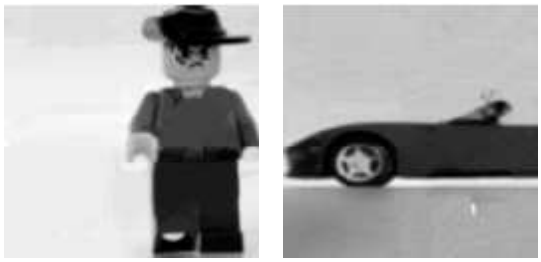


Figure 4: The factor graph above can be easily converted into a pairwise MRF, for which we detailed the BP update rules.

The following two equations of the local and neighboring compatibilities are based on the histograms of derivative filters and corner detectors, re-

(a)



(b) (c)

Figure 5: After running BP, (b),(c) are the most probable decomposition of (a) [1]

spectively. In both cases, the histogram generally resembles the exponential function where the exponent is < 1.

$$\phi_i(g_i) = e^{(-|g|^\alpha - |f|^\alpha)}. \quad (7)$$

$$\psi_{ijkl}(g_i, g_j, g_k, g_l) = e^{-\eta/T(det(g_i g_i^T + g_j g_j^T + g_k g_k^T + g_l g_l^T)^\beta + det(f_i f_i^T + f_j f_j^T + f_k f_k^T + f_l f_l^T)^\beta))} \quad (8)$$

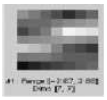
The gradient space was discretized allowing four possible states at each node: $g_i = (I_x, I_y), g_i = (0,0), g_i = (0, I_y), and g_i = (I_x, 0)$. Since BP is exponential in the number of states, the authors kept the number of candidate gradient states small. They claim that, in any case, their results were not affected by adding more possible states.

The algorithm works very well on synthetic images, and on simple natural images such as the one in figure4. However, the algorithm fails on images with more complex textures. One way the authors mean to deal with this limitation is to measure the statistics of more sophisticated edge and corner detectors, that take textures into account.

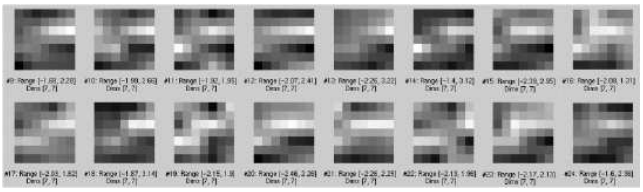
5 Super-Resolution

In super resolution, the goal is to increase the resolution, or zoom into images. Standard interpolation methods, such as cubic spline interpolation can be used for this purpose, however, they "introduce artifacts or blur edges" [14]. In [5, 14], BP on a pairwise Markov random field is used to infer high resolution details by learning from a training set containing pairs of high and low resolution images.

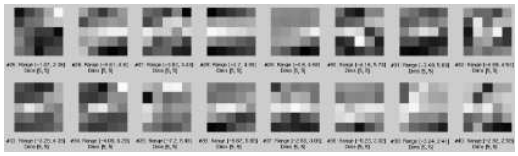
To construct an MRF from a low-resolution image, the authors first apply an initial interpolation so that the image is the same size as the required high resolution image. The image is then divided into patches, where each patch is an observed node in the MRF. The corresponding high-resolution patches are the hidden nodes. Candidate states for each patch are taken from the training set. For each low-resolution patch in the image, they search the training set for patches that best resemble the input. The high-resolution patches corresponding to the best 10-20 patches are used as possible states for the hidden nodes. Note that simply choosing the match for each patch in the final image results in a useless semi random image. The reason can be seen in figure **??**. (a) is the low resolution image patch. (b) shows the best 16 low-resolution matches. (c) shows the high resolution patches corresponding to the best matches found in the training set search. Notice that even though the low-resolution patches are similar, the high-resolution ones are fairly different. Thus, local information alone will not suffice.



(a)



(b)



(c)

Figure 6: (a) The low-resolution input. (b) Best matches from the training set. (c) Corresponding high-resolution patches [14]

To evaluate the compatibilities between neighboring hidden nodes, and between pairs of hidden/observable nodes, the image is divided so that the corresponding high-resolution patches overlap. If the overlapping pixels of two node states match, the compatibility between those states will be high. The compatibility is evaluated using the following equation

$$\psi_{ij}(x_i, x_j) = e^{(-\frac{d_{ij}(x_i, x_j)}{2\sigma^2})} \quad (9)$$

where $d_{ij}(x_i, x_j)$ is the sum of squared differences between overlapping pixels.

Figure 5 shows results of the BP algorithm for super-resolution [5]. The top-left image is the low-resolution image, whereas the bottom-right is the correct high-resolution image we wish to obtain. The top-right and bottom-left are results after cubic spline interpolation and BP, respectively. Notice that the the BP version is much sharper in the hair and eye areas.

Figure 7: top-left: low-resolution image. top-right: cubic spline interpolation. bottom-left: BP approximation. bottom-right: real high-resolution image. [14]

BP, in this context, has an inherent limitation that exists for all super resolution techniques. The high-resolution image will most likely not be the correct one. It merely provides a plausible approximation to the ideal. Some applications, however, require little more . For instance, if someone wants to enlarge a family picture, that person is not going to be concerned with the correctness of each hair strand on their daughter's head, but rather is interested in a visually pleasing enlargement of the photo.

6 Shape-Time Photography

Describing motion in a single image can be useful in a variety of applications such as action summaries, education (e.g., analyzing human motion or natural phenomena), art, etc. A primitive approach simply uses multiple-exposure. However, over-exposure is a problem. Moreover, one cannot get a sense of the 3-D changes of objects over time. Another approach, which [6] calls "layer-by-time," overcomes the limitations of multiple exposure techniques. Here, images are rendered on top of each other in sequence, while only overwriting areas of the new foreground image. With this technique, 3D information is present, but only in areas where the last foreground image does not occlude previous foreground images. In [6], the authors wish to "describe the shape relationships between foreground objects at different times". Figure 6 shows the results of the three techniques discussed above, namely, multiple exposure, "layer-by-time", and shape-time photography.

In shape-time photography, each pixel in the composite image is chosen from the input image whose pixel is in foreground compared to all other images. "This allows the objects to occlude themselves at different times, revealing 3-D shape relationships". The authors use depth information, extracted using stereo techniques. They define $L_k(t), R_k(t)$, to be the left and right values of the *k*th pixel, respectively. Moreover, they define $d_k^L(t)$ to be the distance to the surface projected onto the *k*th pixel of the left image.

Deciding which pixel $L_k(t)$ goes in the *k*th pixel of the left view of the composite image amounts to minimizing $d_k^L(t)$:

$$I_k^L = L_k(argmin_t d_k^L(t)) \tag{10}$$

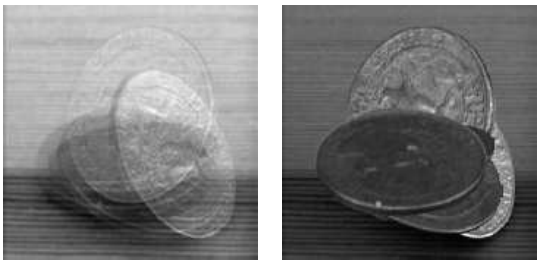
where $argmin_t d_k^L(t)$ will be referred to as a layer assignment. That is, the layer (one of the input images) which is assigned to the composite image at the *k*th pixel.

The problem with equation 10 is that even small errors in the measured distances result in unacceptable visual artifacts. To remove these errors, the authors run the BP algorithm on their initial results. BP is an appropriate solution if the following two assumptions are made: (1) Layer assignments of neighboring pixels are likely the same. (2) "Layer transitions are more likely to happen at image edges, because they may be occluding edges where a layer switch should occur"[6].

The authors build an MRF whose nodes are layer assignments at pixels. The possible states for the nodes are the indices of the input images: $t = 1, 2, ...N$, where N is the number of input images. The compatibilities, ψ, in equation 1 are interpreted as the likelihood of the transition between the layer assignments at the two nodes. Based on assumption (1) and (2) above, they evaluate these to be 1 if the two nodes are in the same layer (i.e., no transition at all is preferable). Otherwise, they set the compatibilities to be $max(E_j(t_j), E_j(t_k), E_k(t_j), E_k(t_k))$, where $E_k(t)$ is the squared magnitude of the image gradient at the image index by t and the pixel indexed by k. The local compatibility, ϕ, will be treated as the likelihood that the a certain layer in a node is in the foreground in the composite image. This probability depends on the measured depths $d_k^L(t)$ and the corresponding measured uncertainties $\sigma_k^L(t)$

Figure 6 shows the result of running BP on the initial layer assignments, where each color in the images corresponds to a different layer assignment. The

(a)



(b) (c)

(d)

Figure 8: (a) The input images. (b) Multiple-exposure summary. (c) Layer-by-time summary. (d) Shape-time summary [6]

(a) (b)

Figure 9: (a) Initial layer assignments. (b) Approximating most probable layer assignments using BP[6]

resulting assignments (b) allow for a much cleaner shape-time photograph.

7 Unwrapping Phase Images

Many applications, such as radar, satellite, and medical imaging, produce phase images in which ”each real-valued observation ... is measured modulus a known wavelength”[2]. If one could obtain the gradient field of the original surface, reconstruction would simply amount to integrating over this field.

A gradient can be found by inferring the number of wrappings between each pair of measurements. The number of wrappings, in turn, can be represented as integer ”shifts”[2]. A shift of 1 implies that the measurements have wrapped around in the positive x or y direction, a -1 shift corresponds to wrapping in the negative direction of x or y, and a shift of zero means that no wrapping has occured. Assuming that adjacent measurements are for the most part close to each other, one could infer integer shifts by minimizing the distance between ”real” measurements. Suppose, for example, that observed measurements are in the range [0,1]. If two adjacent points are measured as 0.4 and 0.5, the number of wrappings here would be zero, whereas the shift between observed measurements 0.0 and 0.9 would be 1.

Previously, the least squares and branch cut methods have been used to infer the gradient field of the original surface from a phase image. However, both techniques produce suboptimal results because of an initial greedy selection of the gradients.

In [2] the authors ”conjecture that although phase unwrapping is generally NP-hard, there exists a near-optimal phase unwrapping algorithm for Gaussian

process priors. Further, [they] believe that algorithm to be loopy belief propagation".

Let the set of all shifts be $S = \{a(x,y).b(x,y) : x = 1,...,N-1; y = 1,...,M-1\}$ and the set of all observed measurements be $\Phi = \{\phi(x,y) : 0 \leq \phi(x,y) \leq 1, x = 1,...,N; y = 1,...,M\}$. Here, a(x,y) and b(x,y) denote shifts in the x-direction and y-direction, respectively. The authors assume that the unwrapped surface can be described as a product of Gaussians:

$$P(S,\Phi) = \prod_{x=1}^{N-1}\prod_{y=1}^{M-1} e^{-(\phi(x+1,y)-\phi(x,y)-a(x,y))^2/2\sigma^2} \cdot \prod_{x=1}^{N-1}\prod_{y=1}^{M-1} e^{-(\phi(x,y+1)-\phi(x,y)-b(x,y))^2/2\sigma^2} \cdot \prod_{x=1}^{N-1}\prod_{y=1}^{M-1} \delta(a(x,y)) + b(x+1,y) - a(x,y+1) - b(x,y)) \quad (11)$$

where δ simply enforces the integrability constraint around every loop.

The graphical model constructed for phase unwrapping can be seen in figure 7 (b). Here, pixels are denoted as X, the white circles are graph nodes that correspond to shifts in the x or y direction, and black circles are clusters of the four surrounding shift nodes. The importance of the black nodes is to ensure that the four gradients represened by the shifts are integrable. This is called the "zero-curl constraint" [2], and simply means that the sum of all the shifts around each loop must equal zero.

As with the transparecny application, the graphical model here is a factor graph. Hence, the BP messages and beliefs are more complicated than equations 23 divided for which messages are

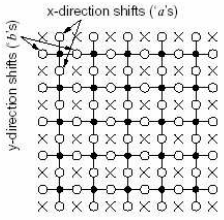
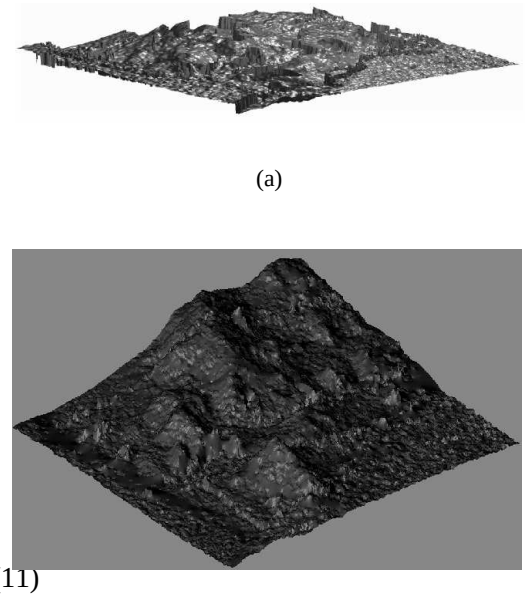


Figure 10: "A graphical model that describes the zero-curl constraints (black discs) between neighboring shift variables (white discs). 3-element probability vectors (μ's) on the relative shifts between neighboring variables (-1, 0, +1) are propagated across the network. " [2]

Figure 7 shows the result of running BP on the graph in figure 7. (a) is the wrapped topographic map,



(a)

(b)

Figure 11: (a) A wrapped topographic map measured on a 2-dimensional grid. (b) Unwrapped surface after integrating the shifts inferred by BP. [2]

and (b) is the estimated unrwapped surface. [2] show that the BP algorithm for unwrapping phase images results in much lower rconstruction errors than previous techniques.

8 Shading and Reflectance Intrinsic images

Although I have no time to discuss this application in detail, figure 12 shows some very nice results of applying BP to the recovery of shading and reflectance images from a single image.

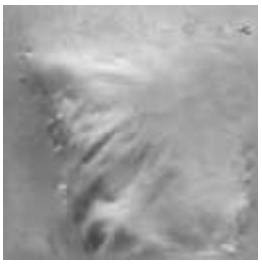
In general, [10] use color and grayscale information to classify local changes in color as resulting from shading effects or reflectance changes. BP is used to propagate infromation from pixels where classification is clear, to areas where it is ambiguous.

9 Conclusion

As discussed in the sections above, Pearl's BP algorithm can generate great results even on graph that contain loops. Researchers have since tried to find some theoretical explanations of why BP should work at all for loopy graphs and, in consequence, derived

more sophisticated and generalized BP algorithms [8, 7, 11]. In this paper, I discussed results of an implementation of belief propagation as applied to the problem of noise removal in images. I discussed other image-based rendering applications of BP. BP has been much more successful than other algorithms in unwrapping phase images. For other image-based rendering applications, BP shows results that compete well with previous techniques.

(a)



(b) (c)

Figure 12: (a) Original image (b) Shading intrinsic image. (c) Reflectance intrinsic image. [10]

References

[1] A. Zomet A. Levin and Y. Weiss. Learning to perceive transparency from the statistics of natural images. In *Neural Information Processing Systems*, 2002.

[2] R. Koetter B. J. Frey and N. Petrovic. Very loopy belief propagation for unwrapping phase images. In *Advances in Neural Information Processing Systems 14, MIT Press, Cambridge, MA*, 2001.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford, 1995.

[4] W. T. Freeman and E. C. Pasztor. Learning to estimate scenes from images. In *Advances in Neural Information Processing Systems*, volume 11, 1999.

[5] W. T. Freeman and E. C. Pasztor. Learning low-level vision. In *International Journal of Computer Vision*, volume 40, 2000.

[6] W. T. Freeman and H. Zhang. Shape-time photography. *Technical Reports of Computer Science Division, UC Berkeley*, 2002.

[7] W. T. Freeman J. S. Yedidia and Y. Weiss. Bethe free energy, kikuchi approximations and belief propagation algorithms. In *MERL, TR*, 2000.

[8] W. T. Freeman J. S. Yedidia and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems*, volume 13, 2000.

[9] W. T. Freeman J. S. Yedidia and Y. Weiss. Understanding belief propagation and its generalizations. *International Joint Conference on Artificial Intelligence in, Distinguished Lecture*, 2001.

[10] W. T. Freeman M. F. Tappen and E. H. Adelson. Recovering intrinsic images from a single image. In *Neural Information Processing Systems*, 2002.

[11] P. Pakzad and V. Anantharam. Belief propagation and statistical physics. In *Conference on Information Sciences and Systems*, 2002.

[12] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[13] D. MacKay R. McEliece and J. Cheng. Turbo decoding as an instance of pearl's 'belief propagation' algorithm. *IEEE Journal on Selected Areas in Communication*, 16:140, 1998.

[14] T. R. Jones W. T. Freeman and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 2002.

[15] Y. Weiss. Belief propagation and revision in networks with loops. *Technical Report Technical Report 1616, MIT AI lab*, 1997.