Efficient Belief Propagation for Early Vision

Pedro F. Felzenszwalb and Daniel P. Huttenlocher
Department of Computer Science, Cornell University
{pff,dph}@cs.cornell.edu

Abstract

Markov random field models provide a robust and unified framework for early vision problems such as stereo, optical flow and image restoration. Inference algorithms based on graph cuts and belief propagation yield accurate results, but despite recent advances are often still too slow for practical use. In this paper we present new algorithmic techniques that substantially improve the running time of the belief propagation approach. One of our techniques reduces the complexity of the inference algorithm to be linear rather than quadratic in the number of possible labels for each pixel, which is important for problems such as optical flow or image restoration that have a large label set. A second technique makes it possible to obtain good results with a small fixed number of message passing iterations, independent of the size of the input images. Taken together these techniques speed up the standard algorithm by several orders of magnitude. In practice we obtain stereo, optical flow and image restoration algorithms that are as accurate as other global methods (e.g., using the Middlebury stereo benchmark) while being as fast as local techniques.

1 Introduction

Over the past few years there have been exciting advances in the development of algorithms for solving early vision problems such as stereo, optical flow and image restoration using Markov random field (MRF) models. While the MRF formulation of these problems yields an energy minimization problem that is NP hard, good approximation algorithms based on graph cuts [3] and on belief propagation [10, 8] have been developed and demonstrated for the problems of stereo and image restoration. These methods are good both in the sense that the local minima they find are minima over "large neighborhoods", and in the sense that they produce highly accurate results in practice. A comparison between the two different approaches for the case of stereo is described in [9].

Despite these substantial advances, both the graph cuts and belief propagation approaches still require several minutes of processing time on today's fastest desktop computers for solving stereo problems, and are too slow for practical use when solving optical flow and image restoration problems. Thus one is faced with choosing between these methods, which produce good results but are slow, and local methods which produce substantially poorer results but are fast. In this paper we present three new algorithmic techniques that substantially improve the running time of belief propagation (BP) for solving early vision problems. Taken together these techniques speed up the standard algorithm by several orders of magnitude, making its running time competitive with local methods. In the case of stereo we obtain results with the same degree of accuracy as standard BP or graph cuts algorithms in about one second per image pair. The differences are even more pronounced for the case of visual motion estimation and image restoration. For example, for optical flow our method is competitive in speed with simple local window-based techniques and yet provides qualitatively better results, similar to robust regularization formulations (e.g., [1]).

The general framework for the problems we consider can be defined as follows (we use the notation from [3]). Let $\mathcal{P}$ be the set of pixels in an image and $\mathcal{L}$ be a set of labels. The labels correspond to quantities that we want to estimate at each pixel, such as disparities, intensities or flow vectors. A labeling f assigns a label $f_p \in \mathcal{L}$ to each pixel $p \in \mathcal{P}$. We assume that the labels should vary smoothly almost everywhere but may change dramatically at some places such as object boundaries. The quality of a labeling is given by an energy function,

$$E(f) = \sum_{(p,q)\in\mathcal{N}} V(f_p, f_q) + \sum_{p\in\mathcal{P}} D_p(f_p), \qquad (1)$$

where $\mathcal{N}$ are the edges in the four-connected image grid graph. $V(f_p, f_q)$ is the cost of assigning labels f_p and f_q to two neighboring pixels, and is normally referred to as the discontinuity cost. $D_p(f_p)$ is the cost of assigning label f_p to pixel p, which is referred to as the data cost. Finding a labeling with minimum energy corresponds to the MAP estimation problem for an appropriately defined MRF.

2 Loopy Belief Propagation

We start by briefly reviewing the BP approach for performing inference on Markov random fields (e.g., see [10]). In particular, the max-product algorithm can be used to find an approximate minimum cost labeling of energy functions in the form of equation (1). Normally this algorithm is defined in terms of probability distributions, but an equivalent computation can be performed with negative log probabilities, where the max-product becomes a min-sum. We use this formulation because it is less sensitive to numerical artifacts, and it uses the energy function definition more directly.

The max-product BP algorithm works by passing messages around the graph defined by the four-connected image grid. Each message is a vector of dimension given by the number of possible labels. Let m^t_{pq} be the message that node p sends to a neighboring node q at time t. When using negative log probabilities all entries in m^0_{pq} are initialized to zero, and at each iteration new messages are computed in the following way,

$$m^t_{pq}(f_q) = \min_{f_p} \left(V(f_p, f_q) + D_p(f_p) + \sum_{s \in \mathcal{N}(p) \setminus q} m^{t-1}_{sp}(f_p) \right) \quad (2)$$

where $\mathcal{N}(p)\setminus q$ denotes the neighbors of p other than q. After T iterations a belief vector is computed for each node,

$$b_q(f_q) = D_q(f_q) + \sum_{p \in \mathcal{N}(q)} m^T_{pq}(f_q).$$

Finally, the label f^*_q that minimizes $b_q(f_q)$ individually at each node is selected. The standard implementation of this message passing algorithm on the grid graph runs in $O(nk^2T)$ time, where n is the number of pixels in the image, k is the number of possible labels for each pixel and T is the number of iterations. Basically it takes $O(k^2)$ time to compute each message and there are $O(n)$ messages per iteration.

In this paper we show three different techniques for substantially reducing the time needed to compute the message updates in (2). First we show that the cost functions $V(f_p, f_q)$ traditionally used in early vision problems enable a new message to be computed in $O(k)$ time, often via the use of distance transform techniques. Our second result shows that for the grid graph (and any bipartite graph) essentially the same beliefs as those defined above can be obtained using only half as many message updates. Besides yielding a speedup this technique also makes it possible to compute the messages "in place", using half as much memory as the normal algorithm. This is important because BP has high memory requirements, storing multiple distributions at each pixel. Finally we present a multiscale algorithm to perform BP in a coarse to fine manner. In our multiscale approach the number of message passing iterations, T, can be a small constant, because long range interactions are captured by short paths in coarse scale graphs. In contrast, for most problems the normal algorithm requires T to be large, as it bounds the distance that information can propagate across the image. This means that in the standard algorithm T needs to grow like $n^{1/2}$ to allow for information from one part of the image propagate everywhere else.

Combining all our techniques together we obtain an $O(nk)$ algorithm that is very fast in practice. Moreover our results are as accurate as those obtained when using standard max-product BP or graph cuts algorithms to minimize energy functions of the form in equation (1). In the case of stereo we quantify this using the benchmark in [7].

3 Computing Messages

This section covers the first of our three techniques, which reduces the time required to compute a single message update from $O(k^2)$ to $O(k)$ for most low-level vision applications. We can re-write equation (2) as,

$$m^t_{pq}(f_q) = \min_{f_p} \left(V(f_p, f_q) + h(f_p) \right), \quad (3)$$

where $h(f_p) = D_p(f_p) + \sum m^{t-1}_{sp}(f_p)$. The standard way of computing the messages is to explicitly minimize over f_p for each choice of f_q. This takes $O(k^2)$ time, where k is the number of labels. However, in low-level vision problems the cost $V(f_p, f_q)$ is generally based on some measure of the *difference* between the labels f_p and f_q rather than on the particular pair of labels. In such cases the messages can often be computed in $O(k)$ time using techniques similar to the ones in [5] for pictorial structures and [6] for HMMs. This is particularly important for problems such as motion estimation and image restoration where the number of labels, k, can be in the hundreds or even thousands. These large label sets have made current algorithms impractical for such problems.

We start by considering a simple case. The Potts model [3] captures the assumption that labelings should be piecewise constant. This model considers only the equality or inequality of labels. For equal labels the cost is zero, while for different labels the cost is a positive constant,

$$V(f_p, f_q) = \begin{cases} 0 & \text{if } f_p = f_q \\ d & \text{otherwise} \end{cases}$$

With this cost function equation (3) can be expressed as,

$$m^t_{pq}(f_q) = \min \left(h(f_q), \min_{f_p} h(f_p) + d \right).$$

In this form it is apparent that the minimization over f_p can be performed once, independent of the value of f_q. Thus the

overall time required to compute the message is $O(k)$. First we compute $\min_{f_p} h(f_p)$, and then use that to compute the message value for each f_q in constant time. Note that this idea still applies when instead of a single constant d there is a constant d_{pq} for each edge in the graph. This is useful when the result of some other process, such as edge detection or segmentation, suggests that discontinuities should be penalized more or less for different pairs of pixels.

Another class of cost functions are based on the degree of difference between labels. For example, in stereo and image restoration the labels $\{0, \ldots, k-1\}$ correspond to different disparities or intensity values. The cost of assigning a pair of labels to neighboring pixels is generally based on the amount of difference between these quantities. In order to allow for discontinuities, as the values are not smoothly changing everywhere, the cost function should be robust, becoming constant as the difference becomes large. One common such function is the truncated linear model, where the cost increases linearly based on the distance between the labels f_p and f_q up to some level,

$$V(f_p, f_q) = \min(s||f_p - f_q||, d), \tag{4}$$

where s is the rate of increase in the cost, and d controls when the cost stops increasing. A similar cost function was used in a BP approach to stereo [8], although rather than truncating the linear cost they have a function that changes smoothly from being almost linear near the origin to a constant value as the cost increases.

We first consider the simpler problem of a pure linear cost without truncation given by $V(f_p, f_q) = s||f_p - f_q||$. Substituting into equation (3) yields,

$$m^t_{pq}(f_q) = \min_{f_p} \left(s||f_p - f_q|| + h(f_p)\right). \tag{5}$$

One can envision the labels as being embedded in a grid. Note that this is a grid of labels and is not related to the image grid. The grid of labels is one-dimensional in the case of stereo or image restoration, and two-dimensional in the case of motion. The minimization in (5) can then be seen as the lower envelope of k upward facing cones of slope s rooted at $(f_p, h(f_p))$. The one-dimensional case is illustrated in Figure 1. This lower envelope calculation is similar to that performed in computing a distance transform (e.g., [2]). For the distance transform the cones are placed at height 0 and occur only at selected values rather than every grid point. Despite these differences, the standard distance transform algorithm from [2] can be modified to compute messages with the linear cost.

It is straightforward to verify that the following simple two-pass algorithm correctly computes the message in equation (5) for the one-dimensional case. The two-dimensional case is similar. First we initialize the message vector with the values of h, and then update its entries sequentially. This is done "in place" so that updates affect one another,

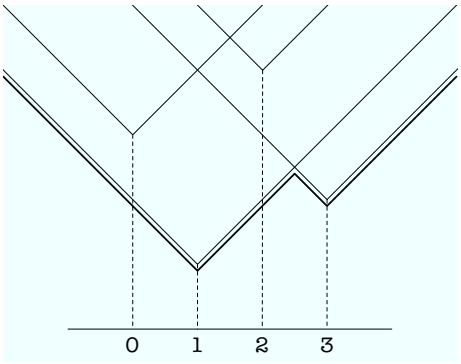


Figure 1: An illustration of the lower envelope of four cones in the case of one-dimensional labels (e.g. stereo disparity or image restoration). Each cone is rooted at location $(f_p, h(f_p))$. The darker line indicates the lower envelope.

for f_q **from** 1 **to** $k-1$:
$m(f_q) \leftarrow \min(m(f_q), m(f_q - 1) + s)$.

The backward pass is analogous,

for f_q **from** $k-2$ **to** 0 :
$m(f_q) \leftarrow \min(m(f_q), m(f_q + 1) + s)$.

Consider the example in Figure 1. The initial value of m is $(3, 1, 4, 2)$. After the forward pass we have $(3, 1, 2, 2)$, and after the backward pass we get $(2, 1, 2, 2)$. The key property that allows us to use this algorithm is that the labels are embedded in a grid, and the discontinuity cost is a linear function of distance in the grid.

Messages with the truncated linear model in equation (4) can now be easily be computed in $O(k)$ time. First we compute what the message would be with the linear model and then compute the element-wise minimum of the linear cost message and the value used for the Potts computation,

$$m_{pq}(f_q) = \min(m(f_q), \min_{f_p} h(f_p) + d).$$

Another useful cost function that can be computed in a similar manner is the truncated quadratic, which grows proportionally to $||f_p - f_q||^2$ up to some level and then becomes a constant thereafter. However we do not cover the algorithm for the truncated quadratic case here.

4 BP on the Grid Graph

In this section we show that for a bipartite graph BP can be performed more efficiently while getting essentially the

same results as the standard algorithm. Recall that a bipartite graph is one where the nodes can be split into two sets so that every edge connects pairs of nodes in different sets. If we color the grid graph in a checkerboard pattern we see that every edge connects nodes of different colors, so the grid graph is bipartite.

The main observation is that for a bipartite graph with nodes $A \cup B$, when computing the messages defined in equation (2) the messages sent from nodes in A only depend on the messages sent from nodes in B and vice versa. In particular, if we know the messages sent from nodes in A at iteration t, we can compute the messages from nodes in B at iteration $t+1$. At this point we can compute the messages from nodes in A at iteration $t+2$. Thus the messages from nodes in A at iteration $t+2$ can be computed without ever computing the messages from those nodes at iteration $t+1$. This motivates the following modification of the standard BP algorithm for bipartite graphs.

In the new scheme messages are initialized in the standard way, but we alternate between updating the messages from A and the messages from B. For concreteness let $\bar{m}^t_{pq}$ be the message sent from node p to node q at time t under this new message passing scheme. When t is odd we update the messages sent from nodes in A and keep the old values for the messages sent from nodes in B. When t is even we update the messages sent from B but not those sent from A. So we only compute half the messages in each iteration. Moreover we can store new messages in the same memory space where the old messages were. This is because in each iteration the messages being updated do not depend on each other. Using the ideas from the last paragraph it is straightforward to show by induction that for all $t > 0$, if t is odd (even) then

$$\bar{m}^t_{pq} = \begin{cases} m^t_{pq} & \text{if } p \in A \text{ (if } p \in B\text{)} \\ m^{t-1}_{pq} & \text{otherwise} \end{cases}.$$

That is, the messages $\bar{m}$ sent under the new scheme are nearly the same as the messages m sent under the standard scheme. Note that when BP converges, this alternative message passing scheme converges to the same fixed point. This is because after convergence $m^{t-1}_{pq} = m^t_{pq}$.

5 Multiscale BP

One problem with BP for many early vision problems follows from the fact that messages are updated in parallel (at least conceptually, even though the implementation is usually sequential). This implies that it takes many iterations for information to flow over large distances in the grid graph. In this section we describe a multiscale technique to circumvent this problem. An alternative approach to address this issue was discussed in [9], but that method still requires many iterations to produce good results. In contrast, our technique produces high quality results using a small fixed number of iterations.

The basic idea is to perform BP in a coarse-to-fine manner, so that long range interactions between pixels can be captured by short paths in coarse graphs. While hierarchical BP methods have been used in other work such as [11], our method differs in that we use the hierarchy only to reduce the number of message passing iterations and do not change the function that is being minimized. For instance in [11] the edges between neighboring pixels in the image grid are replaced by edges between a pixel and its parent in a quad-tree structure. This has the nice property of removing loops from the graph, but it also substantially changes the minimization problem compared with the non-hierarchical case. In particular, the quad-tree structure creates artifacts due to the spatially varying neighborhood structure.

Recall that BP works by looking for fixed points of the message update rule. Usually messages are initialized to zero (in the log-probability space). If we could somehow initialize the messages close to a fixed point one would expect to get convergence more rapidly. Our hierarchical method does exactly this; we run BP at one level of resolution in order to get estimates for the messages at the next finer level. Thus we use a coarse-to-fine computation only to speed up convergence of the original BP problem on the grid graph, without changing the graph structure or the energy function being minimized.

In more detail, we define a set of problems arranged in a coarse-to-fine manner. The zero-th level corresponds to the original labeling problem we want to solve. The i-th level corresponds to a problem where blocks of $2^i \times 2^i$ pixels are grouped together, and the resulting blocks are connected in a grid structure. Intuitively the i-th level can represent labelings where all the pixels in a block are assigned the same label. A key property of this construction is that long range interactions can be captured by short paths in the coarse levels, as the paths are through blocks instead of pixels. Figure 2 illustrates two levels of the structure.

The data costs for the coarser levels are defined in terms of the data costs from the original problem. The cost of assigning label f_b to a block b is,

$$D_b(f_b) = \sum_{p \in b} D_p(f_b), \tag{6}$$

where the sum is over pixels in the block. In practice the block costs at level i can be computed by summing the costs of four blocks from level $i-1$. The summation of negative log costs corresponds to a product of probabilities, thus the interpretation of D_b is that of the probability of observing the corresponding set of pixels given a particular label for them. It is important to note that even when pixels in a block actually prefer different labels, this is captured by the

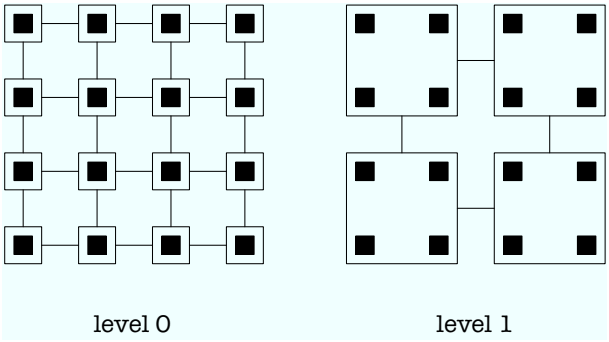


Figure 2: Illustration of two levels in our coarse-to-fine method. Each node in level i corresponds to a block of four nodes in level $i-1$.

fact that several values can have relatively low costs. For instance, if half the pixels prefer label α and half prefer label β, then each of these labels will have low cost whereas other labels will have high cost.

Our multiscale algorithm starts by running BP at the coarsest level of the hierarchy with the messages initialized to zero. After T iterations the resulting messages are used to initialize the messages at the second coarsest level. After T iterations at that level the resulting messages are used to initialize the next finer level, and so on. In the four-connected grid graph each node sends messages corresponding to the directions, *right*, *left*, *up* and *down*. Let r_p^t be the message that node p sends to the right at iteration t, and similarly let l_p^t, u_p^t and d_p^t be the messages that it sends left, up and down, respectively. Note that this is simply a different way of naming the messages, for instance if node p is the left neighbor of node q then $r_p^t = m_{pq}^t$ and $l_q^t = m_{qp}^t$. Similarly for up and down, with special care taken for boundary nodes where there are not neighbors in all directions. Messages at level $i-1$ are initialized from the messages at level i in the following way. We let the initial message that a node sends to the right to be the final message that its block sent to the right in the coarser level. Similarly for the other directions. To be precise,

$$r_{p,i-1}^0 \leftarrow r_{p',i}^T \qquad l_{p,i-1}^0 \leftarrow l_{p',i}^T$$

$$u_{p,i-1}^0 \leftarrow u_{p',i}^T \qquad d_{p,i-1}^0 \leftarrow d_{p',i}^T$$

where node p' at level i is the block containing node p at level $i-1$. When updating the messages at each level, the data costs are as defined above and the discontinuity costs at all levels are the same as that for the original problem. One could imagine other schemes for initializing messages at one level of the hierarchy based on the level above, but this simple approach produces good results in practice.

We have found that with this coarse-to-fine approach of initializing messages, it is enough to run BP for a small number of iterations at each level (between five and ten). Note that the total number of nodes in a quad-tree is just $4/3$ the number of nodes at the finest level. Thus for a given number of iterations the total number of message updates in the hierarchical method is just $1/3$ more than the number of updates in the standard single level method. In the next section we show some results of our method applied to stereo, motion estimation and image restoration. The results produced by this multiscale algorithm sometimes seem to be better than those we have obtained by running standard BP at the finest level for hundreds of iterations. We believe that in such cases the coarse to fine processing is guiding BP to a better local minimum solution, that tends to be smoother overall but still preserves real discontinuities.

Our hierarchical method differs in a subtle but important way from other techniques commonly used in computer vision, such as the Gaussian pyramid (e.g., [4]). Typically hierarchical techniques have been used so that differential methods can be applied when there are large displacements between pairs of images. These techniques are based on reducing the resolution of the image data, whereas ours is based on reducing only the resolution at which the labels are estimated. For instance consider the problem of stereo. Reducing the image resolution reduces the number of disparities that can be distinguished. By the fourth level of such a hierarchy, all disparities between 0 and 16 are indistinguishable. In contrast our method, as defined by equation (6), does not lower the image resolution but rather aggregates data costs over larger spatial neighborhoods. Thus even at a very high level of the hierarchy, small disparities are still evident if they are present over a large spatial region. This difference is crucial to solving the problem at hand, because we want to be able to propagate information about quantities such as disparities over large areas of the image in a small number of message passing iterations. In general, we need a number of levels proportional to $\log_2$ of the image diameter. In contrast a Gaussian pyramid has no useful information about displacements at levels higher than $\log_2$ of the maximum magnitude displacement (and this value is usually much smaller than the image diameter).

6 Experiments

For all the experiments shown here we used the truncated linear model for the discontinuity costs, $V(f_p, f_q) = \min(s||f_p - f_q||, d)$. In all cases we ran five message update iterations at each scale, with a total of six scales. Note that in each iteration we only updated half the messages, using the technique described in Section 4. The running times reported were obtained on a 2Ghz Pentium 4 computer.

In stereo the labels correspond to different disparities. Using the brightness constancy assumption we expect that pixels that correspond between the left and right image

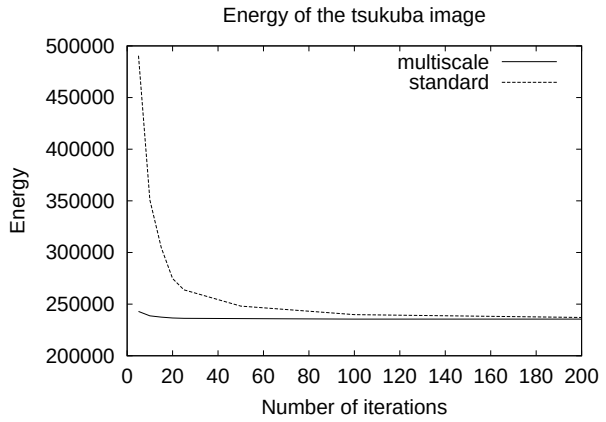


Figure 3: Energy of stereo solution as a function of the number of message update iterations.

should have similar intensities. Thus we use the following data cost for a pixel $p = (x, y)$,

$$D_p(f_p) = \min(||I_l(x, y) - I_r(x - f_p, y)||, \tau),$$

where τ denotes a truncation value. The truncation is necessary to make the data cost robust to occlusion and artifacts that violate the brightness constancy assumption (such as specularities). Figure 4 shows stereo results for the Tsukuba image pair. The running time of our algorithm for this stereo pair is about one second. In contrast, the standard BP algorithm takes a few minutes to produce similar (but patchier) solutions as reported in [9] and [8]. Figure 3 shows the value of the energy we are minimizing as a function of the number of message update iterations for our multiscale BP method versus the standard algorithm. Note how our method computes a low energy solution in just a few iterations per level, while the standard algorithm takes hundreds of iterations to obtain a similar result.

Table 1 shows evaluation results of our stereo algorithm on the Middlebury stereo benchmark [7]. For all stereo experiments we used a fixed set of parameters $s = 10$, $d = 20$ and $\tau = 20$. The input images were smoothed with a Gaussian filter of $\sigma = 0.7$ before computing the data costs.

Overall our method is ranked fifth among those in the Middlebury evaluation, making it comparable to the other global techniques. However these other techniques all run hundreds of times more slowly than our method. It is also important to note that our results are based only on the simple discontinuity and data costs defined above, whereas other methods use additional information about intensity boundaries and occlusions as well as more sophisticated data costs. We used simple cost functions because our focus here is on the algorithmic techniques, and demonstrating that they produce similar quality results much more quickly. Our techniques could be used with other costs as well.

In motion estimation the labels correspond to different displacement vectors. The data costs can be defined as in the stereo case using the brightness constancy assumption,

$$D_p(f_p) = \min(||I_0(p) - I_1(p + f_p)||, \tau).$$

Figure 5 shows optical flow results for a simple pair of images with a person walking and a static background. Note that the motion discontinuities are quite sharp. The running time of our algorithm on this image pair is about four seconds. Results on a standard image pair are illustrated in Figure 6. The energy minimization formulation of the motion estimation problem yields solutions that are regularized yet preserve discontinuities. In particular we get both smooth fields and sharp boundaries. For the motion experiments we used $s = 50$, $d = 150$ and $\tau = 50$. The input images were smoothed with a Gaussian filter of $\sigma = 1.5$ before computing the data costs.

Our last experiment in Figure 7 illustrates image restoration results. Here labels correspond to intensity values. The cost of assigning a particular intensity for a pixel is based on the difference between that intensity and the observed value,

$$D_p(f_p) = \min(||I(p) - f_p||, \tau).$$

The image restoration problem is a case where the distance transform techniques are particularly important. For this problem there are 256 labels, and algorithms that are quadratic in the label set would take a very long time to run. The running time of our algorithm for the example shown here is about three seconds. For this experiment we used $s = 1$, $d = 20$ and $\tau = 100$. The noisy image was obtained by adding independent Gaussian noise with $\sigma = 30$ to each pixel of the original input.

7 Summary and Discussion

We have presented an energy minimization method for solving MRF problems that arise in early vision based on the max-product belief propagation technique. Our method yields results of comparable accuracy to other algorithms but runs hundreds of times faster. In the case of stereo we quantified the accuracy using the Middlebury benchmark. The method is quite straightforward to implement and in many cases should remove the need to choose between fast local methods that have relatively low accuracy, and slow global methods that have high accuracy.

Our method is based on three algorithmic techniques. The first technique uses a variant of distance transform algorithms to reduce the time necessary to compute a single message update from $O(k^2)$ to $O(k)$, where k is the number of labels. The second technique uses the fact that the grid graph is bipartite to decrease both the storage requirements and the running time by a factor of two. This is particularly

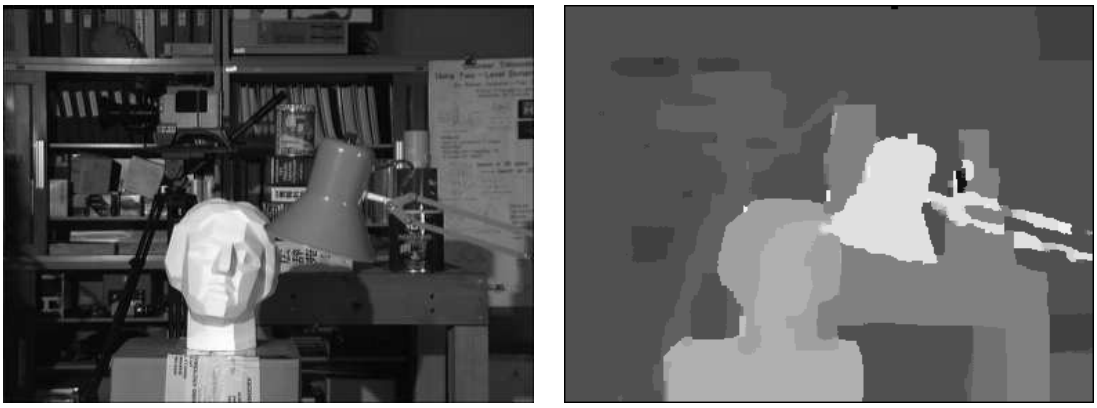


Figure 4: Stereo results for the Tsukuba image pair.

Tsukuba		Sawtooth		Venus		Map	
Rank	Error	Rank	Error	Rank	Error	Rank	Error
8	1.86	7	0.97	4	0.96	9	0.33

Table 1: Evaluation of the stereo algorithm on the Middlebury Stereo benchmark. The error measures the percentage of pixels with wrong disparities. Our method ranks in fifth place in the overall evaluation.

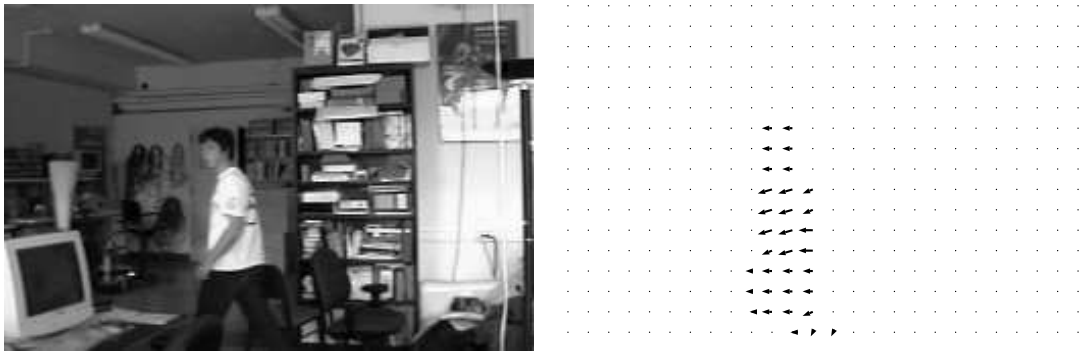


Figure 5: Optical flow results for the Lab image pair.

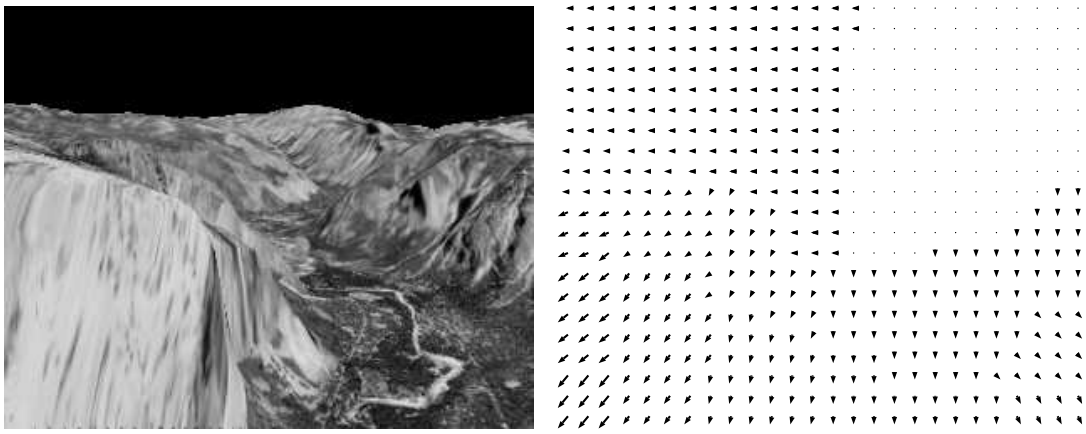
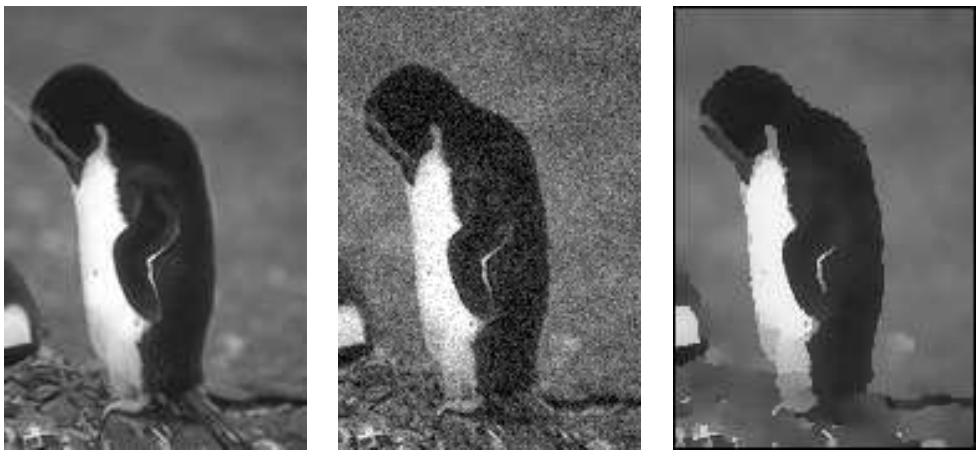


Figure 6: Optical flow results for the Yosemite image pair.



Original Corrupted Restoration

Figure 7: Restoration results for the penguin image.

important because of the relatively high memory requirements of belief propagation methods. The third technique uses a hierarchical structure to reduce the number of message passing iterations to a small constant rather than being proportional to the diameter of the image grid graph. This hierarchical technique differs from decimation based hierarchies such as the Gaussian pyramid that are commonly used in computer vision. It is used to reduce propagation time of messages rather than to solve lower resolution estimation problems.

There are a number of possible extensions to the techniques reported here. As mentioned in the Experiments section, only the simplest cost functions were used here, yet the method is applicable to a broad range of more sophisticated cost functions, including the use of discontinuity costs that vary based on evidence of a boundary or an occlusion. Another extension would be to obtain sub-pixel accuracy in the estimates of disparity or motion. As shown in [9] the sum-product belief propagation approach (as opposed to the max-product used here) can be used to obtain sub-pixel estimates of stereo disparity. Two of our three algorithmic techniques apply directly to the sum-product approach, namely the bipartite grid technique and the hierarchical message propagation technique. The distance transform technique is no longer applicable, however there is a related technique based on convolution that can be used (and has been applied to pictorial structures in [5] and HMMs in [6]).

References

[1] M.J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow-fields. *CVIU*, 63(1):75–104, 1996.

[2] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics and Image Processing*, 34(3):344–371, 1986.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[4] P.J. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communication*, 31(4):532–540, 1983.

[5] P.F. Felzenszwalb and D.P. Huttenlocher. Pictorial structures for object recognition. To appear in the International Journal of Computer Vision.

[6] P.F. Felzenszwalb, D.P. Huttenlocher, and J.M. Kleinberg. Fast algorithms for large-state-space HMMs with applications to web usage analysis. In *Advances in Neural Information Processing Systems*, 2003.

[7] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1):7–42, 2002.

[8] J. Sun, N.N. Zheng, and H.Y. Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[9] M.F. Tappen and W.T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters. In *IEEE International Conference on Computer Vision*, 2003.

[10] Y. Weiss and W.T. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):723–735, 2001.

[11] A.S. Willsky. Multiresolution markov models for signal and image processing. *Proceedings of the IEEE*, 90(8):1396–1458, 2002.