Efficient Belief Propagation for Early Vision

Pedro F. Felzenszwalb

Computer Science Department, University of Chicago

pff@cs.uchicago.edu

Daniel P. Huttenlocher

Computer Science Department, Cornell University

dph@cs.cornell.edu

Abstract

Markov random field models provide a robust and unified framework for early vision problems such as stereo and image restoration. Inference algorithms based on graph cuts and belief propagation have been found to yield accurate results, but despite recent advances are often too slow for practical use. In this paper we present some algorithmic techniques that substantially improve the running time of the loopy belief propagation approach. One of the 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 image restoration that have a large label set. Another technique speeds up and reduces the memory requirements of belief propagation on grid graphs. A third technique is a multi-grid method that 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 results that are as accurate as those of other global methods (e.g., using the Middlebury stereo benchmark) while being nearly as fast as purely local methods.

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 and image restoration using Markov random field (MRF) models. While the MRF framework yields an optimization problem that is NP hard, good approximation techniques based on graph cuts [4] and on belief propagation [12, 10] have been developed and demonstrated for problems such as 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 [11].

Despite these substantial advances however, both the graph cuts and belief propagation approaches are still computationally demanding when compared to local methods that are not attempting to solve MRF-based formulations. Thus one is faced with choosing between the MRF-

based methods, which produce good results but are relatively slow, and local methods which produce substantially poorer results but are fast. In this paper we present several 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 a comparable degree of accuracy to standard BP or graph cuts algorithms in less than one second per image pair. The differences are even more pronounced for problems such as image restoration where there are a relatively large number of possible labels for each pixel.

The general framework for the problems we consider can be defined as follows (we follow the notation in [4]). Let $\mathcal{P}$ be the set of pixels in an image and $\mathcal{L}$ be a finite set of labels. The labels correspond to quantities that we want to estimate at each pixel, such as disparities or intensities. 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 slowly almost everywhere but may change dramatically at some places such as pixels along object boundaries. The quality of a labeling is given by an energy function,

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

where $\mathcal{N}$ are the (undirected) edges in the four-connected image grid graph. $D_p(f_p)$ is the cost of assigning label f_p to pixel p, and is referred to as the data cost. $W(f_p, f_q)$ measures the cost of assigning labels f_p and f_q to two neighboring pixels, and is normally referred to as the discontinuity cost. Finding a labeling that minimizes this energy corresponds to the maximum a posteriori (MAP) estimation problem for an appropriately defined MRF (see [2, 8]).

In low-level computer vision problems the discontinuity cost W is generally based on the *difference* between labels, rather than on their actual values. For example, in stereo and image restoration the labels correspond to the possible disparities or intensity values respectively, and the cost of assigning a pair of labels to neighboring pixels is based on the degree of difference between the labels. Thus in this paper we consider the case where $W(f_p, f_q) = V(f_p - f_q)$, yielding an energy minimization problem of the form,

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

2 Loopy Belief Propagation: Max-Product

We start by briefly reviewing the BP approach for performing inference on Markov random fields (e.g., see [12]). First we consider the max-product algorithm, which can be used to approximate the MAP solution to MRF problems. Normally this technique 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 because it directly corresponds to the energy function definition in equation (1).

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

$$m^t_{p\to q}(f_q) = \min_{f_p}\left(V(f_p - f_q) + D_p(f_p) + \sum_{s\in\mathcal{N}(p)\backslash q} m^{t-1}_{s\to p}(f_p)\right) \tag{2}$$

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

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

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. It takes $O(k^2)$ time to compute each message and there are $O(n)$ messages to be computed in each iteration.

We consider three different techniques for speeding up this standard belief propagation algorithm. First we note that each message update can be expressed as a *min convolution*, and moreover that with the discontinuity costs commonly used in early vision this min convolution can be computed in $O(k)$ time. 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 multi-grid approach for performing BP in a coarse-to-fine manner. In this approach the number of message passing iterations, T, can be a small constant, because long range interactions are captured by short paths in coarse grid graphs.

Combining the three techniques together yields an algorithm that runs in $O(nk)$ time overall and is very fast in practice. In contrast the standard algorithm summarized above requires $O(nk^2)$ time per iteration and the number of iterations T is generally $O(n^{1/2})$ to allow for information to propagate across the image. Our experimental 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 [9].

3 Computing a Message Update in Linear Time

This section covers the first of the 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,

$$m^t_{p\to q}(f_q) = \min_{f_p} \left(V(f_p - f_q) + h(f_p)\right), \tag{3}$$

where $h(f_p) = D_p(f_p) + \sum m^{t-1}_{s\to p}(f_p)$. The standard way of computing this message is to explicitly minimize over f_p for each choice of f_q, which takes $O(k^2)$ time where k is the number of labels.

The form of equation (3) is commonly referred to as a *min convolution*. This operation is analogous to the standard discrete convolution operation, however in a standard convolution the sum would be replaced by a product and the min would be replaced by a sum. While standard discrete convolutions can be efficiently computed using the FFT, no such general result is known for min convolutions. However, for the cost functions $V(f_p - f_q)$ commonly used in computer vision we show that the min convolution, and thus the BP message updates, can often be computed

in $O(k)$ time. Such linear time methods are particularly important for problems such as image restoration where the number of labels, k, can be in the hundreds or more. Note that these methods do not make any approximations; they compute exactly the same result as the $O(k^2)$ brute force algorithm.

3.1 Potts Model

We start by considering a simple measure of the difference between labels, the Potts model [4], which 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,

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

With this cost function the min convolution in equation (3) can be expressed in a form where the minimization over f_p can be performed once, independent of the value of f_q,

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

Separating the minimization over f_p in this manner reduces the time necessary to compute a message to $O(k)$. 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.

3.2 Linear Model

Now we consider the case where the cost function V is based on the magnitude of the difference between labels f_p and f_q. 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. In order to allow for large discontinuities in the labeling the cost function stops growing after the difference

becomes large. For instance,

$$V(x) = \min(c|x|, d), \tag{4}$$

where c 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 [10], 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. Substituting into equation (3) yields,

$$m^t_{p\to q}(f_q) = \min_{f_p} \left(c|f_p - f_q| + h(f_p)\right). \tag{5}$$

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. For instance it is a one-dimensional grid of disparity labels in the case of stereo and a one-dimensional grid of intensity labels in the case of image restoration. The minimization in (5) can then be seen as the lower envelope of k upward facing cones of slope c rooted at $(f_p, h(f_p))$ for each grid point 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., [3]). 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 [3] can be modified to compute the min convolution with a linear cost.

It is straightforward to verify that the following simple two-pass algorithm correctly computes the message in equation (5) for the case where the labels correspond to integers $\{0, \ldots, k-1\}$. 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,

for f_q **from** 1 **to** $k-1$:

$$m(f_q) \leftarrow \min(m(f_q), m(f_q - 1) + c).$$

The backward pass is analogous,

for f_q **from** $k-2$ **to** 0 :

$$m(f_q) \leftarrow \min(m(f_q), m(f_q + 1) + c).$$

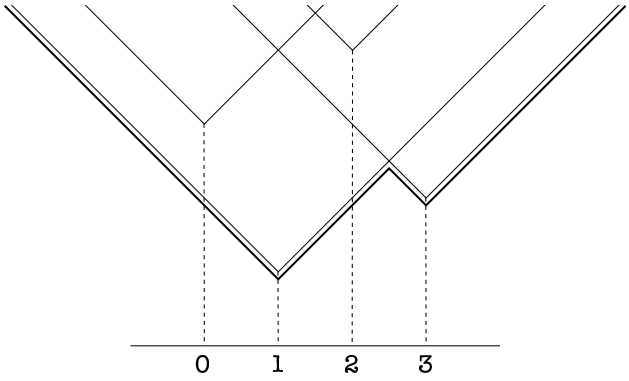

Figure 1: An illustration of the lower envelope of four cones in the case of one-dimensional labels (e.g. stereo disparity or image restoration). Each cone is rooted at location $(f_p, h(f_p))$. The darker line indicates the lower envelope.

Consider the example in Figure 1. The initial value of m is $(3, 1, 4, 2)$. With $c = 1$, the forward pass yields $(3, 1, 2, 2)$, and the backward pass yields $(2, 1, 2, 2)$. 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. If the labels are embedded in a higher dimensional grid (e.g., motion vectors in two dimensions) there is an analogous two-pass distance transform algorithm that can be used (e.g. [3]).

Message updates using the truncated linear model in equation (4) can now easily be computed in $O(k)$ time. Note that a truncated linear function is the lower envelope of a linear function and the constant function defined by the truncation value. Using algebraic properties of min convolutions (see [7]) we can compute a message under the truncated linear model in terms of a message under the linear model and a message under a constant penalty model. First we compute what the message, m', 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,

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

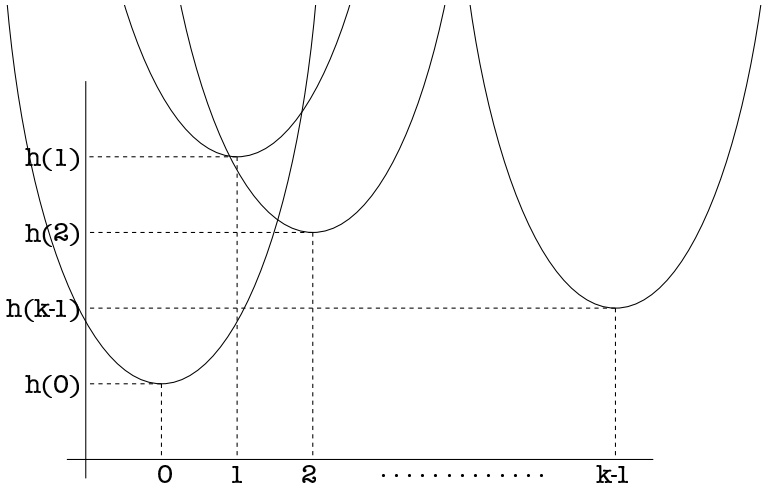

Figure 2: The min convolution as the lower envelope of k parabolas.

3.3 Quadratic Model

Another commonly used cost function is the truncated quadratic. In the case of a one-dimensional label set the cost grows proportionally to $(f_p - f_q)^2$ up to some level and then becomes a constant thereafter. As in the previous subsection, we first consider the case without truncation. Substituting into the message update equation (3), the squared Euclidean (or quadratic) cost update is given by,

$$m^t_{p\to q}(f_q) = \min_{f_p} \left(c(f_p - f_q)^2 + h(f_p) \right). \tag{6}$$

Analogous to the linear case, this can be viewed as the lower envelope of a collection of functions. Each value of f_p defines a constraint that is an upward-facing parabola rooted at $(f_p, h(f_p))$, and the overall minimization is defined by the lower envelope of these parabolas as shown in Figure 2.

Our algorithm for computing this quadratic min convolution has two steps. First we compute the lower envelope of the k parabolas just mentioned. We then fill in the values of $m(f_q)$ by checking the height of the lower envelope at each grid location f_q. Note that this approach starts with something defined on a grid (the values of h), moves to a combinatorial structure defined over the whole domain (the lower envelope of the parabolas) and then moves back to values on the grid by sampling the lower envelope. Pseudocode for the whole procedure is shown in Algorithm 1.

The main part of the algorithm is the lower envelope computation. Note that any two parabolas

defining the lower envelope intersect at exactly one point. The horizontal position of the intersection between the parabola coming from grid position q and the one from p is,

$$s = \frac{(h(p) + cp^2) - (h(q) + cq^2)}{2cp - 2cq}.$$

If $q < p$ then the parabola coming from q is below the one coming from p to the left of the intersection point s, and above it to the right of s.

We compute the lower envelope by sequentially computing the lower envelope of the first q parabolas, where the parabolas are ordered according to their corresponding horizontal grid locations. The algorithm works by computing the combinatorial structure of this lower envelope. We keep track of the structure using two arrays. The horizontal grid location of the i-th parabola in the lower envelope is stored in $v[i]$. The range in which the i-th parabola of the lower envelope is below the others is given by $z[i]$ and $z[i+1]$. The variable j keeps track of the number of parabolas in the lower envelope.

When considering the parabola from q, we find its intersection with the parabola from $v[j]$ (the rightmost parabola in the lower envelope computed so far). There are two possible cases, as illustrated in Figure 3. If the intersection is after $z[j]$, then the lower envelope is modified to indicate that the parabola from q is below all others starting at the intersection point. If the intersection is before $z[j]$ then the parabola from $v[j]$ should not be part of the new lower envelope, so we decrease j to delete that parabola and repeat the procedure.

This algorithm is a simpler version of a technique for incrementally computing the lower envelope of k parabolas in $O(k \log k)$ time [6]. That algorithm operates by sorting the parabolas into an appropriate order to be inserted into the lower envelope in amortized constant time. In our case the problem is simpler because the parabolas are all of the same shape and they are already sorted into an appropriate order.

We note that a two-dimensional quadratic min convolution can be computed by first performing a one-dimensional min convolution along each column of the grid, and then performing a one-dimensional min convolution along each row of the result (see [7]). This argument extends to arbitrary dimensions, resulting in the composition of one-dimensional min convolutions along each dimension of the underlying grid of labels. For stereo and image restoration the label space is

Algorithm $DT(h)$

1. $j \leftarrow 0$ ($*$ Index of rightmost parabola in lower envelope $*$)
2. $v[0] \leftarrow 0$ ($*$ Locations of parabolas in lower envelope $*$)
3. $z[0] \leftarrow -\infty$ ($*$ Locations of boundaries between parabolas $*$)
4. $z[1] \leftarrow +\infty$
5. **for** $q = 1$ **to** $n - 1$ ($*$ Compute lower envelope $*$)
6. $\quad s \leftarrow ((h(q) + cq^2) - (h(v[j]) + cv[j]^2))/(2cq - 2cv[j])$
7. $\quad$ **if** $s \leq z[j]$
8. $\quad\quad$ **then** $j \leftarrow j - 1$
9. $\quad\quad\quad$ **goto** 6
10. $\quad\quad$ **else** $j \leftarrow j + 1$
11. $\quad\quad\quad v[j] \leftarrow q$
12. $\quad\quad\quad z[j] \leftarrow s$
13. $\quad\quad\quad z[j + 1] \leftarrow +\infty$
14. $j \leftarrow 0$
15. **for** $q = 0$ **to** $n - 1$ ($*$ Fill in values of min convolution $*$)
16. $\quad$ **while** $z[j + 1] < q$
17. $\quad\quad j \leftarrow j + 1$
18. $\quad \mathcal{D}_h(q) \leftarrow c(q - v[j])^2 + h(v[j])$

Algorithm 1: The min convolution algorithm for the squared Euclidean distance in one-dimension.

one-dimensional. In other early vision problems such as motion estimation the label space is two-dimensional.

As in the linear case, message updates using a truncated quadratic model can also be computed in $O(k)$ time. Again we first compute what the message would be with the quadratic model and then compute the element-wise minimum of this message with the value from the Potts computation. Moreover we can compute messages under a discontinuity cost function defined by the lower envelope of a small number of linear and quadratic functions as described in [7]. Note also

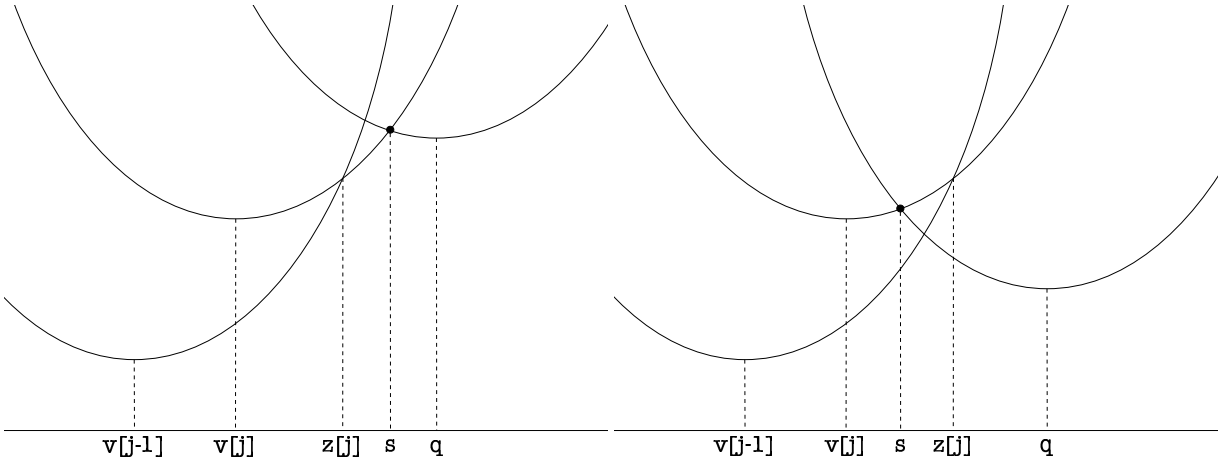

Figure 3: The two possible cases considered by the algorithm when adding the parabola from q to the lower envelope constructed so far. In (a) $s > z[j]$ while in (b) $s \leq z[j]$.

that the algorithm for the quadratic cost function could easily be modified to handle any convex discontinuity cost.

4 BP on the Grid Graph

In this section we describe how BP can be performed more efficiently for a bipartite graph while getting essentially the same results as the standard algorithm. This is analogous to the red-black techniques used for Gauss-Seidel relaxations. The main issue in using such a technique in the context of BP is establishing that it computes the correct messages. 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 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 $\bar{m}^t_{p\to q}$ be the message sent from node p to node q at iteration 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

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

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 $m^{t-1}_{p\to q} = m^t_{p\to q}$.

5 Multi-Grid BP

One drawback of using BP for many early vision problems follows from the fact that messages are updated locally and 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 multi-grid technique to circumvent this problem.

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 [14], our method differs in that we use the hierarchy only to initialize messages at successively finer levels. This makes it possible to reduce the number of message passing iterations at each level, without changing the overall problem structure. In contrast, for

example in [14] the underlying graph is changed so as to replace edges between neighboring pixels in the image grid 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 being solved. In particular, the quad-tree structure creates artifacts due to the spatially varying neighborhood structure.

BP works by looking for fixed points of the message update rule. For max-product BP the messages are usually initialized to zero (in log-probability space). If we could initialize the messages close to a fixed point one would expect to get convergence more rapidly. This is how the method described here works; we run BP at one level of resolution and then use the messages at that level in order to get estimates for the messages at the next finer level. Thus the coarse-to-fine computation speeds up convergence of the original BP problem leaving the graph structure and the energy function unchanged.

In developing the multi-grid approach we use a slightly different notation that makes the image grid Γ explicit. The problem that we want to solve is to assign a label $f_{i,j} \in \mathcal{L}$ to each location $(i,j) \in \Gamma$ while minimizing the energy,

$$E(f) = \sum_{(i,j)\in\Gamma} D_{i,j}(f_{i,j}) + \sum_{(i,j)\in\Gamma\setminus\mathcal{C}} V(f_{i,j} - f_{i+1,j}) + \sum_{(i,j)\in\Gamma\setminus\mathcal{R}} V(f_{i,j} - f_{i,j+1}), \qquad (7)$$

where $\mathcal{C}$ and $\mathcal{R}$ are respectively the last column and last row of the image grid. Equation (7) is the same as the original energy function in (1) except that it is expressed over the locations in the image grid rather than over a set of sites and neighbors.

Let $\Gamma^0, \Gamma^1, \ldots$ be a hierarchy of grids such that $\Gamma^0 = \Gamma$ and each node in Γ^ℓ corresponds to a block of $\epsilon \times \epsilon$ pixels of the original grid Γ, where $\epsilon = 2^\ell$. Intuitively the ℓ-th level represents labelings where the image pixels in each $\epsilon \times \epsilon$ 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 level grids, as the paths go through blocks instead of going through pixels. Figure 4 illustrates two levels of the structure. Now we define a hierarchy of optimization problems on the coarse grids.

Let f^ℓ be a labeling for the sites in Γ^ℓ. The energy function at level ℓ is given by,

$$E(f^\ell) = \sum_{(i,j)\in\Gamma^\ell} D^\ell_{i,j}(f^\ell_{i,j}) + \sum_{(i,j)\in\Gamma^\ell\setminus\mathcal{C}^\ell} V^\ell(f^\ell_{i,j} - f^\ell_{i+1,j}) + \sum_{(i,j)\in\Gamma^\ell\setminus\mathcal{R}^\ell} V^\ell(f^\ell_{i,j} - f^\ell_{i,j+1}), \qquad (8)$$

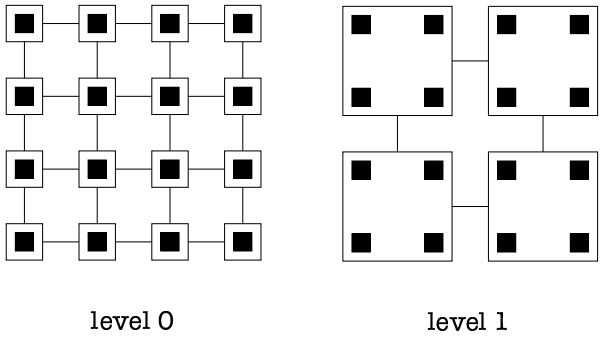

Figure 4: Illustration of two levels in the multi-grid method. Each node in level ℓ corresponds to a 2×2 block of nodes in level $\ell - 1$.

where D^ℓ and V^ℓ are the data and discontinuity costs at level ℓ. There are a number of options for how to define the costs at each level. We take an approach motivated by finite-element methods, where the full set of image pixels corresponding to each block is taken into consideration.

First consider the data cost $D^\ell_{i,j}$. Intuitively assigning a label α to a block (i, j) at level ℓ is equivalent to assigning that label to each pixel in the block, yielding a sum of the data costs for the pixels in that block,

$$D^\ell_{ij}(\alpha) = \sum_{u=0}^{\epsilon-1} \sum_{v=0}^{\epsilon-1} D_{\epsilon i+u, \epsilon j+v}(\alpha).$$

The summation of negative log costs corresponds to taking a product of probabilities, thus the data cost for an $\epsilon \times \epsilon$ block can be understood in terms of the probability of observing the corresponding set of pixels given one particular label for all of them. A given block can prefer several labels, because a cost is determined for each label of the block. 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. Note that when computing the data costs it is not necessary to always sum over the original grid Γ. Instead the calculation can be done more efficiently by summing over four data costs at the next finer level.

Now consider the discontinuity costs at level ℓ. There is no discontinuity cost between pixels inside a block, as every coarse labeling assigns the same label for such pixels. For each pair of neighboring blocks there are ϵ pairs of pixels along their boundary. In measuring the difference between labels for two neighboring blocks we use a finite difference approach, where the difference

between the labels is divided by the separation between the block centers, ϵ. This leads to,

$$V^{\ell}(\alpha - \beta) = \epsilon V\left(\frac{\alpha - \beta}{\epsilon}\right).$$

The ϵ term multiplying V takes into account the number of neighboring pixels along the boundary of two neighboring blocks, while the ϵ term in the denominator inside V takes into account the separation between blocks when measuring the difference between neighboring labels.

Different forms of discontinuity costs produce different relationships between the discontinuity costs across the problem hierarchy. For instance, using a linear cost function $V(x) = c|x|$ yields a hierarchical discontinuity cost that is independent of the level,

$$V^{\ell}(x) = c|x|,$$

as the ϵ terms cancel out. On the other hand using a quadratic cost function $V(x) = cx^2$ yields a hierarchical discontinuity cost that is weaker higher-up in the hierarchy,

$$V^{\ell}(x) = cx^2/\epsilon,$$

As mentioned before, in practice it is important to use robust discontinuity costs such as the truncated linear model in (4). We do this by truncating the discontinuity costs at each level,

$$V^{\ell}(\alpha - \beta) = \min\left(\epsilon V\left(\frac{\alpha - \beta}{\epsilon}\right), d\right).$$

Another alternative would be to truncate the individual cost functions V, but this would result in the truncation factor changing based on the level in the hierarchy, due to the multiplication of V by ϵ. In practice we have found it better to truncate the costs between blocks instead.

A simple coarse-to-fine strategy using the hierarchy of problems defined by equation (8) is to compute the BP messages for the problem at the coarsest level of the hierarchy and then use that to initialize the messages at the next level, and so on, down to the original grid. The messages at each level are a function of the same set of labels but represent interactions between different sized blocks of pixels. Given a final set of messages sent by a block at level ℓ, we initialize the messages sent by the four blocks inside it at level $\ell - 1$ to those values. This is done separately for the messages in the four directions: right, left, up and down in the grid.

We have found that with this coarse-to-fine approach 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 the hierarchy 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 finest level.

This hierarchical method differs in a subtle but important way from other multi-scale techniques commonly used in computer vision, such as the Gaussian pyramid (e.g., [5]). Typically such techniques have been used for finding displacements between pixels in pairs of images using differential methods. 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 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 Sum-Product Belief Propagation

The max-product BP algorithm is motivated by finding a labeling with maximum posterior probability, or equivalently with minimum energy. Another common formulation is based on selecting the most probable label for each pixel. There is a subtle but important difference between selecting the most probable labeling and selecting the most probable label for each pixel individually. Selecting the most probable label for each pixel minimizes the number of pixels with incorrect labels,

but the overall labeling obtained in this way could have small joint posterior probability.

The sum-product BP algorithm can be used to approximate the posterior probability of each label for each pixel. As with the max-product algorithm, the sum-product method works by passing messages around the graph defined by the neighborhood system $\mathcal{N}$. In this section we let $m^t_{p\to q}$ be the message that node p sends to a neighboring node q at iteration t of the sum-product algorithm,

$$m^t_{p\to q}(f_q) = \sum_{f_p} \left(\Psi(f_p - f_q)\Phi_p(f_p) \prod_{s\in\mathcal{N}(p)\setminus q} m^{t-1}_{s\to p}(f_p) \right) \tag{9}$$

where as above $\mathcal{N}(p)\setminus q$ denotes the neighbors of p other than q. The potential functions are defined in terms of the discontinuity costs and data costs in the energy function (1),

$$\Phi_p(f_p) = e^{-D_p(f_p)}, \quad \Psi(f_p - f_q) = e^{-V(f_p - f_q)}.$$

After T iterations a belief vector is computed for each node,

$$b_q(f_q) = \Phi_q(f_q) \prod_{p\in\mathcal{N}(q)} m^T_{p\to q}(f_q).$$

The value $b_q(f_q)$ is an approximation to the probability that the correct label for pixel q is f_q. As was true for the max-product case, 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

All of the algorithmic techniques that we discussed above for max-product also apply to the sum-product algorithm for low-level vision problems. The bipartite graph technique in Section 4 and the multi-grid technique in Section 5 both apply directly, as neither technique depends on the particular form of the messages. On the other hand, the techniques for linear-time message updates depend on the form of the message and thus do not apply directly. However there is an analogous set of techniques that we now describe.

Following the analysis for the max-product case, we can rewrite the message update rule in equation (9) as

$$m^t_{p\to q}(f_q) = \sum_{f_p} \left(\Psi(f_p - f_q)h(f_p) \right), \tag{10}$$

where $h(f_p) = \Phi_p(f_p) \prod m^{t-1}_{s \to p}(f_p)$. In this form we see that the message update computation is a convolution, which can be computed in $O(k \log k)$ time for k discrete values of f_p and f_q using the fast fourier transform (FFT).

The most commonly used compatibility functions $\Psi(f_p - f_q)$ are Gaussians or mixtures of Gaussians, and in these cases the message computation can be approximated in $O(k)$ time. The method is also very fast in practice and thus preferable to the FFT not only because of the log factor improvement but because of the low constants.

Convolution with a Gaussian can be approximated in $O(k)$ time using the box sum method in [13]. The technique uses the fact that a Gaussian can be accurately approximated by the sequential convolution of a small number of box filters. The discrete convolution of a function with k sample points with a box filter can be computed in $O(k)$ time, because each successive shift involves only a single addition and subtraction, regardless of the width of the box. To approximate Gaussian convolution the input function $h(f_p)$ is sequentially convolved with a set of such box filters. In practice only four convolutions are necessary to obtain a good approximation to a Gaussian, yielding an $O(k)$ method that is also very fast in practice.

Using the box-sum technique together with the multi-grid and bipartite graph techniques results in an $O(nk)$ algorithm for sum-product belief propagation on a grid with n nodes (or pixels). For more general potential functions Ψ, the use of the FFT yields an $O(nk \log k)$ method.

7 Experiments

In this section we show some simple experimental results to illustrate the techniques described in the paper. In these experiments we used the max-product formulation of belief propagation, or more precisely the min-sum algorithm where costs correspond to negative log probabilities. We considered both the problems of stereo and image restoration. In both cases we combined all three techniques together: the linear time message updates, the bipartite graph message passing schedule, and the multi-grid method. For the multi-grid method we used six levels in the grid hierarchy. The test images were generally around 640×480 pixels in size, making the coarsest grid have just a few blocks.

For the stereo problem the labels correspond to different disparities that can be assigned to pixels in the left image. We used a truncated linear cost function for the discontinuity term,

$$V(f_p - f_q) = \min(|f_p - f_q|, d).$$

Using the brightness constancy assumption we expect that corresponding pixels in the left and right images should have similar intensities. We assume that the images have been rectified so that disparities are horizontal shifts, and use a truncated linear model for the data cost

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

where τ is a truncation value and λ is a scaling factor. The truncation makes the data cost robust to occlusion and artifacts that violate the brightness constancy assumption (such as specularities). The scaling factor allows us to control the relative weighting of the data and discontinuity costs. More involved data costs such as those in [1] could also be employed in this framework.

Figure 5 shows stereo results for the Tsukuba image pair using our algorithm with these truncated linear cost functions. In this case we used 10 message update iterations per level. The running time was about one second on a 2GHz Pentium IV. In generating stereo results we used the following set of parameters: $d = 1.7$, $\tau = 15$ and $\lambda = 0.07$. The resulting discontinuity cost function is nearly like a Potts model, with cost zero when the labels are the same, 1 when they differ by one, and 1.7 otherwise. The input images were smoothed slightly, with a Gaussian of $\sigma = 0.7$ prior to computing the data cost. This example illustrates that the use of the hierarchical method seems to produce less variation in the output than is obtained by non-hierarchical BP techniques (for example the background is more uniformly a single disparity).

Figure 6 shows the value of the energy that is being minimized as a function of the number of message update iterations for our multi-grid 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 many more iterations to obtain a similar result. This provides some empirical evidence that the multi-grid technique is operating as intended, allowing information to propagate over long distances in few message update iterations.

Figure 7 gives empirical results of the speedups obtained by each of the techniques described in the paper. The graph compares running BP with all speedup techniques versus running BP

Figure 5: Stereo results for the Tsukuba image pair.

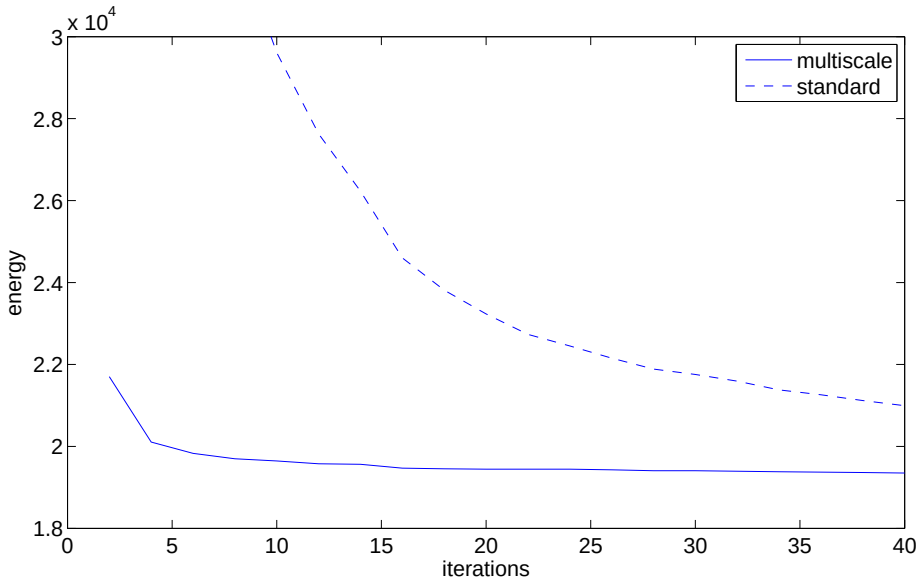

Figure 6: Energy of stereo solution as a function of the number of message update iterations.

with all but one of the techniques. In each case the running time of the algorithm is controlled by varying the number of message update iterations. We see that each speedup technique provides a significant benefit. Note how the min convolution method provides an important speedup even when the number of labels is small (16 disparities for the Tsukuba images).

Table 1 shows evaluation results of our stereo algorithm on the Middlebury stereo benchmark [9]. These results were obtained using the parameters described above. Overall our method per-

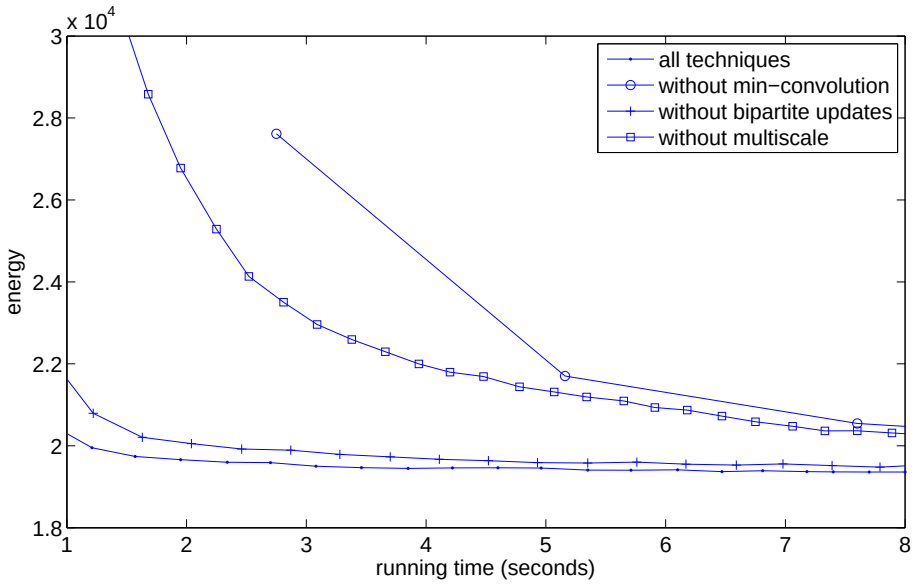

Figure 7: Energy of stereo solution as a function of running time. The graph compares running BP will all speedup techniques described in the paper versus BP with all but one of the techniques.

Tsukuba		Sawtooth		Venus		Map	
Rank	Error	Rank	Error	Rank	Error	Rank	Error
13	1.84	13	0.94	8	0.94	11	0.36

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 12th place in the overall evaluation.

forms comparably to the original graph cuts energy minimization approach [4, 9] that similarly used simple data and discontinuity costs, as well as to results in [11] that compared belief propagation with graph cuts. However these other methods run considerably more slowly, taking tens or hundreds of times longer than our algorithm. It is important to stress that this comparison is intended to demonstrate that the algorithmic techniques we have presented here produce similar quality results much faster than these other methods. Considerably more sophisticated data terms, use of occlusion information, and other techniques could be incorporated in order to improve the

accuracy of the final results.

While belief propagation and graph cuts methods are now commonly used to solve the stereo problem, these techniques have not been widely adopted for other low level vision problems such as image restoration. There is a long history of MRF-based formulations of image restoration problems (e.g., see [2, 8]), however computing solutions for these problems using previous methods is quite slow, particularly when the number of possible labels for each pixel is large. Here we consider the problem of restoring images that have 256 intensity values. We use input images that have been corrupted with additive Gaussian noise as well as by masking out regions.

In image restoration both the labels and the input data are intensities. We used a truncated quadratic for the discontinuity cost,

$$V(f_p - f_q) = \min((f_p - f_q)^2, d),$$

and a quadratic cost for the data term,

$$D_p(f_p) = \lambda \min((I(p) - f_p)^2),$$

measuring the difference between the label at a pixel and the observed intensity at that pixel.

In principle this formulation of the restoration problem should also do a good job of filling in missing data, by propagating information from other parts of the image. To demonstrate this capability we show an example of an image that was distorted by adding Gaussian noise of $\sigma = 20$, and in which in addition a rectangular region was masked out. A modified data cost function was used, where for a masked pixel the data cost is zero for any label. That is, $D_p(f_p) = 0$ when pixel p is masked. The discontinuity cost function remained unchanged. The parameters for the cost functions were: $d = 200$ and $\lambda = 0.04$. In this case we used 5 message passing iterations per level and the running time was approximately 10 seconds on a 2Ghz Pentium IV. Figure 8 shows the results of our algorithm. Note that method does a good job of filling in missing data based on the remaining image.

Corrupted Restoration

Figure 8: Restoration results with an input that has missing values.

8 Summary

We have presented three algorithmic techniques for speeding up the belief propagation approach for solving low level vision problems formulated in terms of Markov random fields. The main focus of the paper is on the max-product formulation of belief propagation, and the corresponding energy minimization problem in terms of costs that are proportional to negative log probabilities. We also show how similar techniques apply to the sum-product formulation of belief propagation. The use of our techniques yields results of comparable accuracy to other algorithms but hundreds of times faster. In the case of stereo we quantified this 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.

The first of the three techniques reduces the time necessary to compute a single message update from $O(k^2)$ to $O(k)$, where k is the number of possible labels for each pixel. For the max-product formulation this technique is applicable to problems where the discontinuity cost for neighboring labels is a truncated linear or truncated quadratic function of the difference between the labels. The method is not an approximation, it uses an efficient algorithm to produce exactly the same results as the brute force quadratic time method. For sum-product a similar technique yields an $O(k \log k)$

method for any discontinuity cost function based on difference between labels.

The second of the three techniques 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 important because of the relatively high memory requirements of belief propagation methods, which store multiple distributions at each pixel. The third of the techniques uses a multi-grid approach to reduce the number of message passing iterations to a small constant, whereas the standard method requires a number of iterations that is proportional to the diameter of the image grid.

For problems such as stereo, where the label set is relatively small, the techniques presented here provide substantial speedup. For other problems, including image restoration, where the label set can be quite large, these techniques can make an MRF-based approach tractable where it was not before. There are several opportunities for further development of our techniques. First, a general method for computing the min convolution quickly, analogous to the FFT for convolution, would broaden the applicability of fast message updates to arbitrary discontinuity cost functions based on difference between labels. Second, the lower envelope method that we have presented for the min convolution could be extended to handle problems where the labels are embedded in some space but do not lie on a regularly spaced grid. More generally, it would be interesting to consider whether other sorts of structures on the set of labels enable fast methods.

References

[1] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406, April 1998.

[2] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

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

[4] 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.

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

[6] O. Devillers and M Golin. Incremental algorithms for finding the convex hulls of circles and the lower envelopes of parabolas. *Inform. Process. Lett.*, 56(3):157–164, 1995.

[7] P.F. Felzenszwalb and D.P. Huttenlocher. Distance transforms of sampled functions. September 2004. Cornell Computing and Information Science Technical Report TR2004-1963.

[8] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] W.M. Wells. Efficient systhesis of gaussian filters by cascaded uniform filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):234–239, 1986.

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