MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Markov networks for low-level vision

William T. Freeman, Egon C. Pasztor

TR99-08   December 1999

Abstract

We seek a learning-based algorithm that applies to various low-level vision problems. For each problem, we want to find the scene interpretation that best explains image data. For example, we may want to infer the projected velocities (scene) which best explain two consecutive image frames (image). From synthetic data, we model the relationship between local image and scene regions, and between a scene region and neighboring scene regions. Three probabilities are learned, which characterize the low-level vision algorithm: the local prior, the local likelihood, and the the conditional probabilities of scene neighbors. Given a new image, we propagate likelihood functions in a Markov network to infer the underlying scene. We use a factorization approximation, ignoring the effect of loops. This yields an efficient method to infer low-level scene interpretations, which we always find to be stable. We illustrate the method with different representations, and show it working for three applications: an explanatory example, motion analysis and estimating high resolution images from low-resolution ones.

MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY
http://www.merl.com

Markov networks for low-level vision

William T. Freeman and Egon C. Pasztor
MERL, Mitsubishi Electric Research Laboratory
201 Broadway; Cambridge, MA 02139
`freeman@merl.com, pasztor@merl.com`

TR-99-08 February 1999

Abstract

We seek a learning-based algorithm that applies to various low-level vision problems. For each problem, we want to find the scene interpretation that best explains image data. For example, we may want to infer the projected velocities (scene) which best explain two consecutive image frames (image). From synthetic data, we model the relationship between local image and scene regions, and between a scene region and neighboring scene regions. Three probabilities are learned, which characterize the low-level vision algorithm: the *local prior*, the *local likelihood*, and the the conditional probabilities of scene neighbors. Given a new image, we propagate likelihood functions in a Markov network to infer the underlying scene. We use a factorization approximation, ignoring the effect of loops. This yields an efficient method to infer low-level scene interpretations, which we always find to be stable. We illustrate the method with different representations, and show it working for three applications: an explanatory example, motion analysis and estimating high resolution images from low-resolution ones.

1. First printing, TR99-08, February, 1999

Markov networks for low-level vision

William T. Freeman and Egon C. Pasztor
MERL, a Mitsubishi Electric Res. Lab.
201 Broadway
Cambridge, MA 02139
`freeman, pasztor@merl.com`

February 9, 1999

Abstract

We seek a learning-based algorithm that applies to various low-level vision problems. For each problem, we want to find the scene interpretation that best explains image data. For example, we may want to infer the projected velocities (scene) which best explain two consecutive image frames (image). From synthetic data, we model the relationship between local image and scene regions, and between a scene region and neighboring scene regions. Three probabilities are learned, which characterize the low-level vision algorithm: the *local prior*, the *local likelihood*, and the the conditional probabilities of scene neighbors. Given a new image, we propagate likelihood functions in a Markov network to infer the underlying scene. We use a factorization approximation, ignoring the effect of loops. This yields an efficient method to infer low-level scene interpretations, which we always find to be stable. We illustrate the method with different representations, and show it working for three applications: an explanatory example, motion analysis and estimating high resolution images from low-resolution ones.

1 Introduction

Our goal is to interpret images, in particular, to solve low-level vision tasks. Figure 1 shows examples of the problems we hope to address: interpreting line drawings, analyzing motion, and extrapolating resolution. Each task has input *image* data, which can be a single image, or a collection of images over time. From that, we want to estimate an underlying *scene*, which could be 3-dimensional shape, optical flow, reflectances, or high resolution detail. We will focus on low-level scene representations like these that are mapped over space. Reliable solutions to these vision tasks would have many applications in searching, editing, and interpreting images. Machine solutions might give insight into biological mechanisms.

Much research has addressed these problems, providing important foundations. Because the problems are under-determined, regularization and statistical estimation theory are cornerstones (e.g.: [21, 26, 43, 36, 39, 27]). Unfortunately, tractable solutions can be difficult to find, or are often unreliable or slow. Oftentimes the statistical models used need to be made-up or hand-tweaked. Various image interpretation problems have defied generalization from initial simplified solutions [40, 3, 38].

In part to address the need for stronger models, researchers have analyzed the statistical properties of the visual world. Several groups derived V1-like receptive fields from ensembles of images [30, 5]; Simoncelli and Schwartz [37] accounted for contrast normalization effects by redundancy reduction. Li and Atick [2] explained retinal color coding by information processing arguments. Researchers have developed powerful methods to analyze and synthesize realistic textures by studying the response statistics of V1-like multi-scale, oriented receptive fields [19, 11, 43, 36]. These methods may help us understand the early stages of image representation and processing in the brain.

Unfortunately, they don't address how a visual system might *interpret* images. To do that, one should collect statistics relating images with their underlying scene interpretations. This data is difficult to collect for natural scenes, since it involves gathering both images and the ground truth data, of the scene attributes to be estimated.

A useful alternative is to use computer graphics to generate and render *synthetic* worlds, where every attribute is known, and record statistics from those. Several researchers have done so: Kersten and Knill studied linear shading and other problems [25, 24]; Hurlbert and Poggio trained a linear color constancy estimator [20]. Unfortunately, the simplified (usually linear) models which were used to obtain tractable results limited the usefulness of these methods.

Our approach is to use general statistical models, but to make the method tractable by restricting ourselves to *local* regions of images and scenes. We follow a learning-based approach, and use Markov networks to form models of image rendering, and prior probabilities for scenes.

We ask: can a visual system correctly interpret a visual scene if it models (1) the probability that any local scene patch generated the local image, and (2) the probability that any local scene is the neighbor to any other? The first probabilities allow making scene estimates from local image data, and the second allow these local estimates to propagate. This approach leads to a Bayesian method for low level vision problems, constrained by Markov assumptions. We describe this general method, illustrate implementation choices, and show it working for several problems.

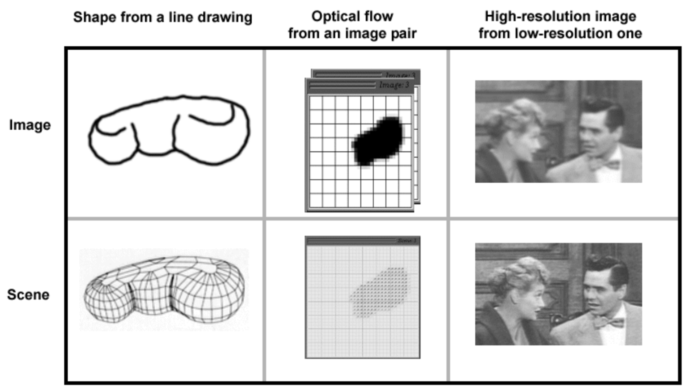

Shape from a line drawing
Optical flow from an image pair
High-resolution image from low-resolution one
Image
Scene

Figure 1: Example low-level vision problems. (The scenes shown are idealizations, not program outputs).

2 Bayesian method

We take a Bayesian approach [6, 26]. We want to find the the *posterior* probability, $P(\vec{x}|\vec{y})$: the probability of a scene, $\vec{x}$, given the image observation, $\vec{y}$. By Bayes rule (standard names shown):

$$P(\vec{x}|\vec{y}) = \frac{P(\vec{y}|\vec{x})\ P(\vec{x})}{P(\vec{y})} \quad (1)$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}. \quad (2)$$

The evidence is independent of the scene, $\vec{x}$, we want to estimate, so we treat it as a normalization constant.

The optimum scene estimate, $\hat{\vec{x}}$, depends on the loss function, or the penalty for guessing the wrong scene [6, 12]. Two different choices lead to the estimation rules used here. The Maximum A Posteriori, or MAP, estimate is the scene, $\hat{\vec{x}}$, which maximizes the posterior, $P(\vec{x}|\vec{y})$. The Minimum Mean Squared Error, or MMSE, estimate is the posterior mean. For computational convenience, we will use MMSE with inference in a continuous representation, and MAP with discrete representations.

The likelihood term, $P(\vec{y}|\vec{x})$ is the forward model. For a given scene, it asks, what is the probability that this scene renders to the observed image data? The prior probability, $P(\vec{x})$ states how probable a scene is to exist.

The space of all possible solutions is very large. In principle, to find the best scene, one needs to render each candidate scene and evaluate its likelihood and prior probability. To make practical systems, researchers typically restrict some aspect of the problem. We will make three major approximations. First, we only model the statistics of local regions, invoking the Markov assumption (next section). Second, we will use a factorization approximation (Sect. 3.1) to solve the Markov model. Finally, our preferred solution is to sample from a continuous distribution (Sect. 4.3) during the "inference phase".

3 Markov network

We place the image and scene data in a Markov network [31, 17]. We break the images and scenes into localized patches where image patches connect with underlying scene patches; scene patches also connect with neighboring scene patches, Fig. 2. (The neighbor relationship can be with regard to position, scale, orientation, etc.). This forms a network of scene nodes, each of which may have an associated observation.

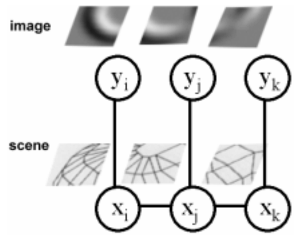


Figure 2: Markov network for vision problems. Observations, y, have underlying scene explanations, x. Connections between nodes of the graphical model indicate statistical dependencies.

Referring to Fig. 2, the Markov assumption asserts that complete knowledge of node x_j makes nodes x_i and x_k independent, or $P(x_i, x_k|x_j) = P(x_i|x_j)P(x_k|x_j)$. We say x_i and x_k are conditionally independent given x_j. (For notational convenience, we will drop the vector symbol from the random variables.) The Markov assumption also implies that $P(x_i|x_j, x_k) = P(x_i|x_j)$.

This lets us model a complicated spatial probability by a network of (tractable) probabilities governing local relationships.

To apply a Markov network to vision problems, we need to first *learn* the parameters of the network from a collection of training examples. Then, given new image data, we seek to *infer* the corresponding scene.

There are exact and approximate methods for both the learning and inference phases [17, 22, 14]. Exact methods can be prohibitively time consuming. For networks without loops, the Markov assumption allows the posterior probability to factorize, yielding efficient local rules for both learning and inference [31, 42, 22, 14]. We will adopt a *factorization* approximation, ignoring the effect of loops during both learning and inference.

3.1 Factorization

There are different, equivalent ways to use the Markov assumptions to factorize the posterior probability of a Markov network without loops [31, 42, 22, 14]. We use a particular factorization, which we believe to be new, which leads to a message passing scheme with appealing interpretations for the messages [13].

We derive the evidence propagation rules of the inference stage by calculating the messages needed to achieve the optimal Bayesian estimate at a scene node. These rules will indicate the probabilities we need to measure during the learning phase. We present the derivation for the MMSE estimator, using continuous variables, then indicate how to modify the rules for the MAP estimator, and for discrete variables.

Before any messages are passed, each node, j, has only its local evidence, y_j, available for estimating the local scene, x_j. The best that node can do is to calculate the mean of the posterior from $P(x_j|y_j) \propto P(x_j, y_j) = P(y_j|x_j)P(x_j)$. The MMSE estimate at iteration 0, $\hat{x}_{j0}$, is

$$\hat{x}_{j0} = \int_{x_j} x_j P(x_j) P(y_j|x_j). \tag{3}$$

At the first message passing iteration, node j will communicate with neighboring scenes. For simplicity of exposition, we assume scene node j has just two scene neighbors, nodes i and k. Including those observations y_i and y_k of the neighboring nodes, the MMSE estimate for x_j will be the mean over x_j of the joint posterior, $P(y_i, x_i, y_j, x_j, y_k, x_k)$, after marginalization over x_i and x_k:

$$\begin{aligned} \hat{x}_{j1} &= \int_{x_j} x_j \int_{x_i, x_k} P(y_i, x_i, y_j, x_j, y_k, x_k) && (4) \\ &= \int_{x_j} x_j \int_{x_i, x_k} P(y_i, x_i, y_j, y_k, x_k|x_j) P(x_j) && (5) \\ &= \int_{x_j} x_j P(x_j) P(y_j|x_j) && (6) \\ &\quad \times \int_{x_i} P(y_i, x_i|x_j) \int_{x_k} P(y_k, x_k|x_j). && (7) \end{aligned}$$

The factorization of $P(y_i, x_i, y_j, y_k, x_k|x_j)$ used above follows from the conditional independence defined by the Markov network. After marginalizing over x_i, the message $\int_{x_i} P(y_i, x_i|x_j)$ will become $P(y_i|x_j)$, and similarly for x_k. We call this a *region likelihood*, the likelihood given x_j of some other region of image observations, in this case, just y_i. This is a nice feature of our particular factorization: the messages passed between nodes are all region likelihoods. These are all conditionally independent of each other (given x_j), if the network has no loops.

To form the full likelihood function for x_j, we multiply together all the region likelihoods and the local likelihood. Multiplying by the local prior, $P(x_j)$, then gives the posterior for x_j, joint with y_j and all the observations that entered into the region likelihoods.

We call $P(x_j)$ the *local prior*, and $P(y_j|x_j)$ the *local likelihood*. $P(x_i|x_j)$ is a *scene conditional*. Eq. (7) then has the form,

$$\begin{aligned} \hat{x}_j = \int_{x_j} x_j \quad & \times \quad \text{[local prior]} && (8) \\ & \times \quad \text{[local likelihood]} && (9) \\ & \times \prod_{\text{neighbor links}} \text{[region likelihoods]} && (10) \end{aligned}$$

We need to know how to generate new region likelihood messages from those received on the previous iteration. Let us determine what message node x_j should pass to node x_k. To examine a more general case, assume now x_j connects to four other scene nodes (see Fig. 7). Suppose node x_j receives region likelihood messages $P(y_{R1}|x_j)$, $P(y_{R2}|x_j)$, $P(y_{R3}|x_j)$ from each scene node other than x_k. By construction, these regions ($R1$, $R2$, $R3$) are conditionally independent of each other and of y_j, given x_j. Thus the region likelihood for the union of those observations is just their product,

$$P(y_{R1}, y_{R2}, y_{R3}, y_j|x_j) = P(y_{R1}|x_j)P(y_{R2}|x_j)P(y_{R3}|x_j)P(y_j|x_j). \tag{11}$$

We seek to calculate the next region likelihood message $L_{kj} = P(y_{R1}, y_{R2}, y_{R3}, y_j|x_k)$ to pass to node x_k. At each iteration, by adding more image observations into the passed region likelihoods, the scene nodes add more observations to their joint posteriors of observations and scene value, improving their scene estimates.

We will find $P(y_{R1}, y_{R2}, y_{R3}, y_j|x_k)$ by first computing $P(y_{R1}, y_{R2}, y_{R3}, y_j, x_j|x_k)$, then marginalizing over x_j. We use $P(a, b|c) = P(a|b, c)P(b|c)$ to write

$$P(y_{R1}, y_{R2}, y_{R3}, y_j, x_j|x_k) = \tag{12}$$

$$P(y_{R1}, y_{R2}, y_{R3}, y_j|x_j, x_k)P(x_j|x_k). \tag{13}$$

Because x_j is closer in the network to the observations in the region likelihood than is x_k, by the Markov properties, we have $P(y_{R1}, y_{R2}, y_{R3}, y_j|x_j, x_k) = P(y_{R1}, y_{R2}, y_{R3}, y_j|x_j)$.

Combining our results, we have

$$P(y_{R1}, y_{R2}, y_{R3}, y_j|x_k) = \tag{14}$$

$$\int_{x_j} P(x_j|x_k)P(y_{R1}|x_j)P(y_{R2}|x_j)P(y_{R3}|x_j)P(y_j|x_j). \tag{15}$$

This tells us how to pass region likelihoods from node j to node k: (1) multiply together the region likelihoods from the *other* neighbors of node j; (2) include node j's local likelihood; (3) multiply by $P(x_j|x_k)$; and (4) marginalize over x_j. Figure 3 shows the passed messages.

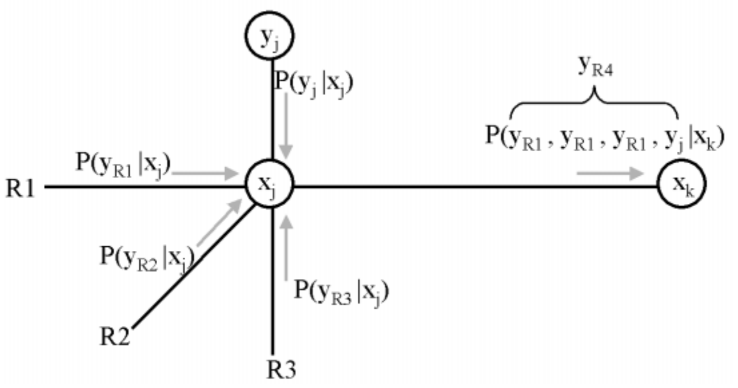


Figure 3: To send a message to node x_k, node x_j multiplies together the (conditionally independent) region likelihoods for regions $R1$, $R2$, and $R3$, giving $P(y_{R1}, y_{R2}, y_{R3}|x_j)$ (gray arrows show messages). It multiplies in its own local likelihood, also conditionally independent of the others, giving $P(y_{R1}, y_{R2}, y_{R3}, y_j|x_j)$. After multiplication by $P(x_j|x_k)$ and marginalization over x_j, that becomes $P(y_{R4}|x_k)$, node x_k's region likelihood for the new region $R4 = R1 \bigcup R2 \bigcup R3 \bigcup y_j$. That is node x_j's message to node x_k, and illustrates one iteration of recursively passing conditionally independent information to neighboring nodes.

Summarizing, after each iteration, the MMSE estimate at node j, $\hat{x_j}$ is

$$\hat{x_j} = \int_{x_j} x_j P(x_j)P(y_j|x_j) \prod_k L_{kj}, \tag{16}$$

where k runs over all scene node neighbors of node j. We calculate the region likelihoods, L_{kj}, from:

$$L_{kj} = \int_{x_k} P(x_k|x_j)P(y_k|x_k) \prod_{l \neq j} \tilde{L}_{lk}, \tag{17}$$

where $\tilde{L}_{lk}$ is L_{lk} from the previous iteration. The initial $\tilde{L}_{lk}$'s are 1.

To learn the network parameters, we measure $P(x_j)$, $P(y_j|x_j)$, and $P(x_k|x_j)$, directly from the synthetic training data.

3.2 Variations

For a discrete probability representation, we replace the integral signs with sums. To use the MAP estimator, instead of MMSE, the above arguments hold, with the following two changes:

$$\int_{x_j} x_j \quad \to \quad \mathrm{argmax}_{x_j} \tag{18}$$

$$\int_{x_i, x_k} \quad \to \quad \max_{x_i, x_k}. \tag{19}$$

3.3 Loops

If the Markov network contains loops, then the region likelihoods arriving at a node are not guaranteed conditionally independent, and the above factorization may not hold. Both learning and inference then require more computationally intensive methods. There are a range of approximation methods to choose from [22, 14].

One approach is to use non-loopy multi-resolution quad-tree networks [27], for which the factorization rules do apply, to propagate information spatially. Unfortunately, this gives results with artifacts along quad-tree boundaries, statistical boundaries in the model not present in the real problem.

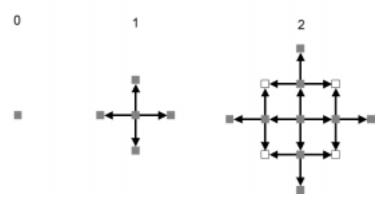
We found good results by including the loop-causing connections between adjacent nodes at the same tree level but applying the factorized learning and propagation rules, anyway. Others have obtained good results using the same approach for inference [14, 28, 42];

By "unwrapping" the computation of a loopy network, Weiss provides theoretical arguments why this works for certain networks [42]. Evidence is double-counted, but all evidence ends up being double counted equally. Fig. 4 shows how this multiple counting is distributed over space, for an array of four-connected nodes. By the (scaled) intensity, the figure shows the number of times each node contributes to the belief calculation of the center node.

Clearly, all nodes are not counted equally. One could avoid this concern by using other approximations for solving the Markov network, although those would be slower. However, in our experiments, we do not observe problems from making the factorization approximation. We prefer this method because it is fast, and we can identify clearly what the messages between nodes mean.

It may be the case that for various vision problems, the likelihood functions are essentially binary functions, either zero or not, indicating either a scene interpretation is inconsistent with the observations, or it is consistent. For such likelihood functions, the region likelihoods are always conditionally independent, as is required for the arguments of Sect. 3.1, even for a network with loops. (For binary likelihoods, the factorization of Eq. 11 will hold even if observations are counted multiple times.) The behavior of this limiting case may explain why we obtain good results using this factorization approximation.

(a)



(b)

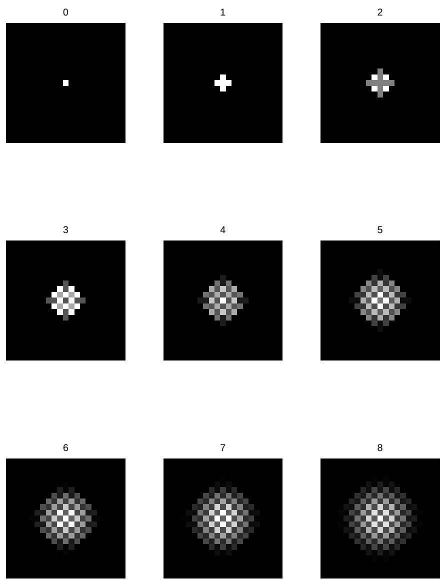


Figure 4: Plots of the number of times each node contributes a message to the belief at the center node, as a function of iteration number. (a) shows locations contributing to the belief, with arrows indicating which node each message came from. The nine images of (b) repeat that and continue to iteration 8, plotting one node per square pixel. This view of the "unwrapped" calculation shows that node data enters the belief calculation multiple times (each plot is rescaled).

4 Probability Representations

We have explored different probability representations with our method, Eqs. (16) and (17): (1) continuous, (2) discrete, and (3) a hybrid method. To explain our technique, in the rest of the paper we present example applications using each of these three probability representations.

4.1 Continuous Representation

We used mixtures of gaussians to represent continuous probability densities. We fit the densities to training data using EM (Expectation Maximization) [7].

During inference, Eqs. (16) and (17) require multiplying together probability densities. Since gaussians multiply together to give gaussians (given by the Kalman filtering formula [16]), the multiplied mixtures yield other mixtures of gaussians.

However, one must merge and prune the gaussians. We tested whether two gaussians of a mixture could be merged into one by measuring the Kullback-Leibler (KL) distance [7] between the mixture with the merged gaussians and the mixture with the non-merged gaussians. (We measured the KL distance by sampling, so there is randomness in our pruning). We repeatedly testing pairs of gaussians, never repeating comparisons of a merged mixture with comparisons made by its ancestors gaussians. This pruning allows iterative calculation with continuous densities, using a mixture of gaussians representation.

4.1.1 Toy Problem

We illustrate this representation, and our technique, for a toy problem with 1-dimensional data. Consider a 3 by 3 network of nodes (Fig. 8, (a)) laid out spatially 5 units apart. The "scene" we want to estimate is the horizontal position of each node, which is the same for each node of a column, and different by 5 units for each adjacent node in a row. The image data usually tells the scene value itself, with a little noise. However, when the image data, $y = 0$, then the underlying scene, x, can take on any value. Figure 5 shows the joint density, $P(x, y)$. This behavior is typical of some vision problems, where local image cues can determine the scene interpretation, but other image values give ambiguous interpretations.

4.1.2 Learning

During learning, we present many examples of image data, y, coupled with scene data, x. From those examples, we used EM to fit mixtures of ten gaussians to each of the probability densities that define our algorithm: the local prior $P(x_j)$, the local likelihood, $P(y_j|x_j)$, and the scene conditionals, $P(x_{left}|x)$, and $P(x_{above}|x)$. Figure 6 shows the fitted mixtures.

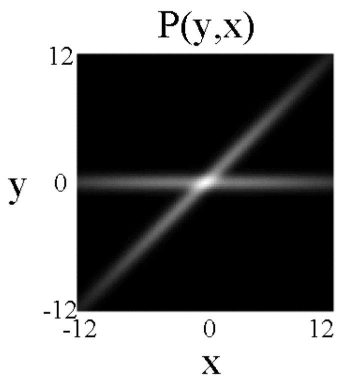


Figure 5: Illustrative toy problem, joint probability densities. x is the scene data; y is the corresponding observation. y is a good indicator of the underlying scene, except when $y = 0$; then the scene can be anything.

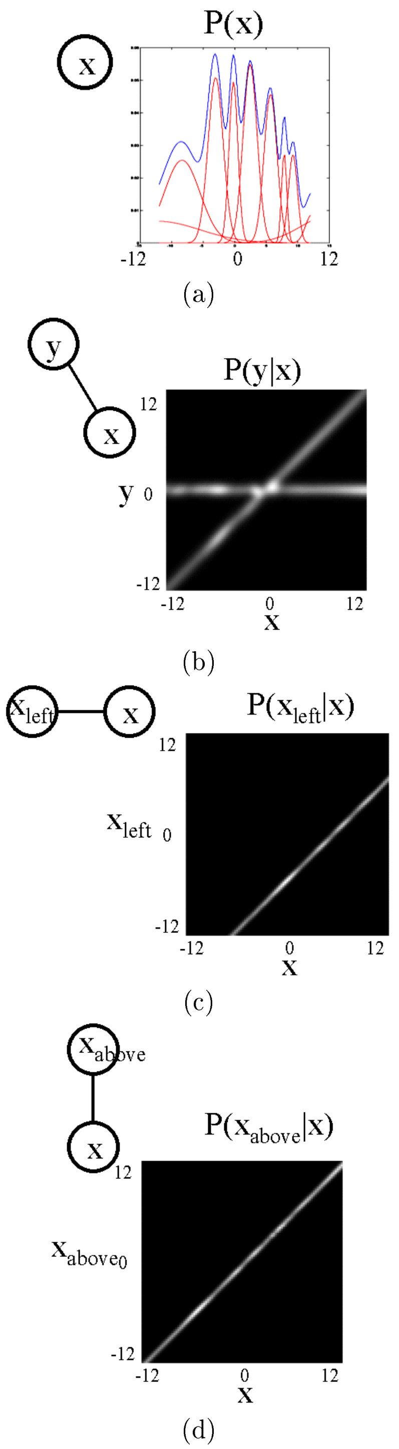


Figure 6: Outputs of learning phase for toy problem of Sect. 4.1.1. All probability densities shown are mixtures of gaussians, fitted to labelled training data. Both scene, x, and image, y, variables are 1-d here. (a) Local prior (red shows component gaussians of mixture, blue is sum). Fit is only approximate to the true prior (vertical projection of Fig. 5), but adequate for inference. (b), (c) and (d): Learned conditionals for $P(y|x)$, $P(x_{\text{left}}|x)$, $P(x_{\text{above}}|x)$. Jumps to horizontally neighboring node changes x by 5 units. Jumps to vertical neighbors leave x constant. The learned conditionals (c) and (d) both correctly model this made-up world.

4.1.3 Inference

Given particular image data (Fig. 8 (a)), we infer the underlying scene values for our toy problem by iteratively applying Eqs. (16) and (17). Figures 7 and 8 illustrate the message passing and belief calculation. Figures 9 – 12 show the posterior probabilities at each node for 0, 1, 2, and 10 iterations. By iteration 2, each node has communicated with node 5 (which has unambiguous local image information). Each node then has the correct posterior mean. After 10 iterations, the spreads of the posterior probability peaks are artificially narrowed, from treating as conditionally independent messages passed around through loops. However, the posterior means maintain their proper values.

4.2 Discrete representation

The continuous representation used above is appealing, but inference is very slow for larger problems. Furthermore, the necessary repeated pruning of gaussian mixtures reduces the modeling accuracy.

To correct those problems, one might consider using a discrete representation. Then the basic calculations in the inference equations, Eqs. (16) and (17), become (fast) vector and matrix multiplications. In this section, we show the result of a discrete representation, illustrating another application domain: motion analysis. Here the "image" is two concatenated image frames at sequential times. The "scene" is the corresponding projected velocites in each frame.

The training images were randomly generated moving, irregularly shaped blobs, as typified by Fig. 13 (a). The contrast with the background was randomized. Each blob was moving in a randomized direction, at some speed between 0 and 2 pixels per frame.

We represented both the images and the velocities in 4-level Gaussian pyramids [9], to efficiently communicate across space. Each scene patch then additionally connects with the patches at neighboring resolution levels. Figure 13 shows the multiresolution representation (at one time frame) for images and scenes.[1]

We applied the training method and propagation rules to motion estimation, using a vector code representation [18] for both images and scenes. We wrote a tree-structured vector quantizer, to code 4 by 4 pixel by 2 frame blocks of image data for each pyramid level into one of 300 codes for each level. We also coded scene patches into one of 300 codes. Figure 13 shows an input test image, (a) before and (b) after vector quantization. The true underlying scene, the desired output, is shown at the right, (a) before and (b) after vector quantization.

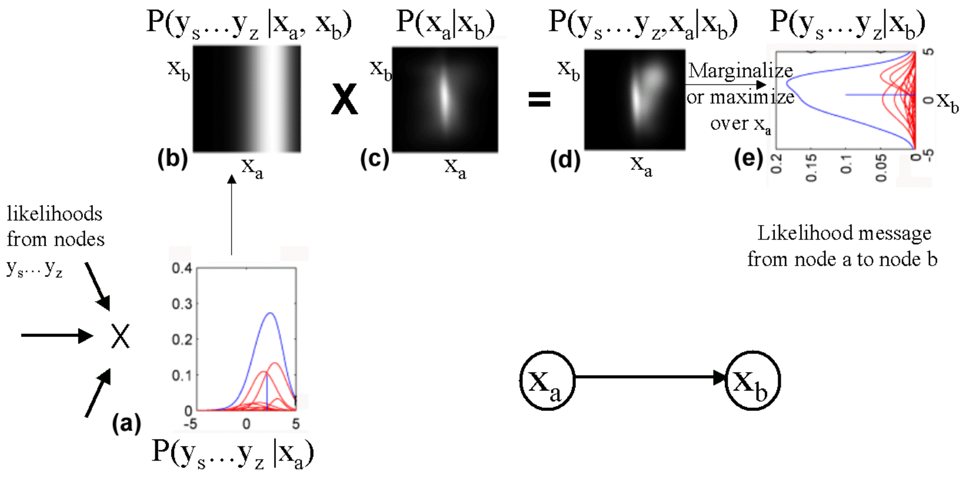
[1]To maintain the desired conditional independence relationships, we appended the image data to the scenes. This provided the scene elements with image contrast information, which they would otherwise lack.

4.2.1 Learning

During learning, we presented approximately 200,000 examples of different moving blobs, some overlapping, of a contrast with the background randomized to one of 4 values. Using co-occurance histograms, we measured the statistical relationships that embody our algorithm: $P(x)$, $P(y|x)$, and $P(x_n|x)$, for scene x_n neighboring scene x. Figures. 14 and 15 show examples of these measurements.

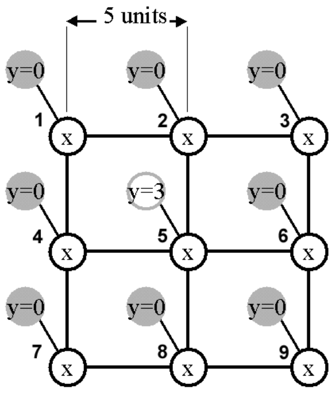
4.2.2 Inference

Figure 16 shows six iterations of the inference algorithm (Eqs. 16 and 17) as it converges to a good estimate for the underlying scene velocities. The local probabilities we learned ($P(x)$, $P(y|x)$, and $P(x_n|x)$) lead to figure/ground segmentation, aperture problem constraint propagation, and filling-in (see caption). The resulting inferred velocities are correct within the accuracy of the vector quantized representation.

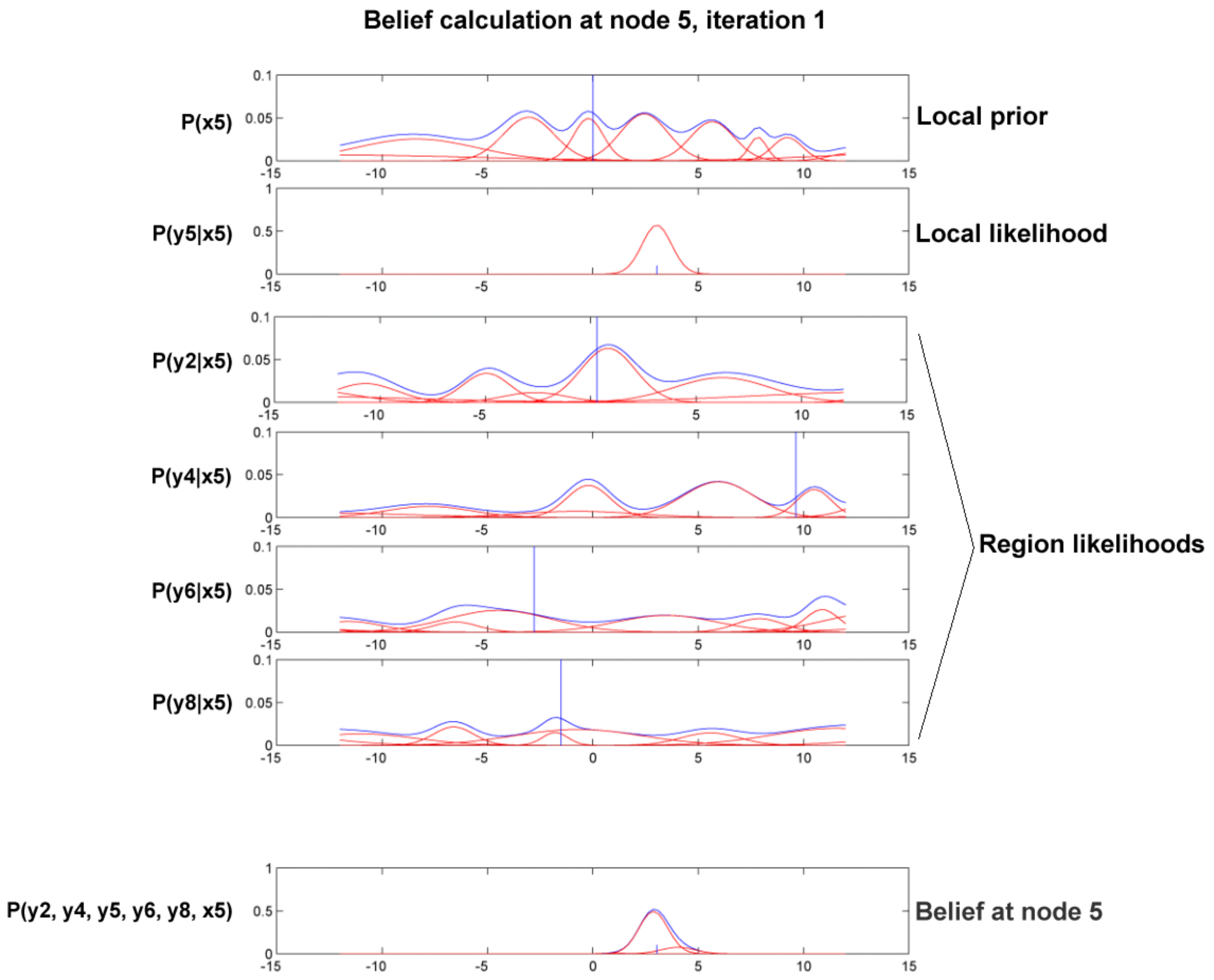


P(ys…yz |xa, xb)
P(xa|xb)
P(ys…yz,xa|xb)
P(ys…yz|xb)
xb
xa
X
=
Marginalize or maximize over xa
(b)
(c)
(d)
(e)
Likelihood message from node a to node b
likelihoods from nodes ys…yz
(a)
P(ys…yz |xa)
xa
xb

Figure 7: Graphical illustration of passing a message from node x_a to node x_b. The steps are: (a) combine conditionally independent region likelihoods arriving at x_a by multiplying them together; (b) introduce the variable x_b; (c) and (d) multiply by $P(x_a|x_b)$ to move x_a out of the conditioning variables; (e) marginalize out x_a. The result is the region likelihood for the union of all the regions multiplied together in (a), but now referenced to node x_b.



(a)



(b)

Figure 8: (a) Observed data for scene inference, for toy problem. (b) Belief calculation at a node (node 5), first iteration. The belief is the product of six probability distributions over x_5: the prior on x ($P(x_5)$), the local likelihood ($P(y_5 = 3|x_5)$), and the four region likelihoods, which are messages passed from nodes 2, 4, 6, and 8.

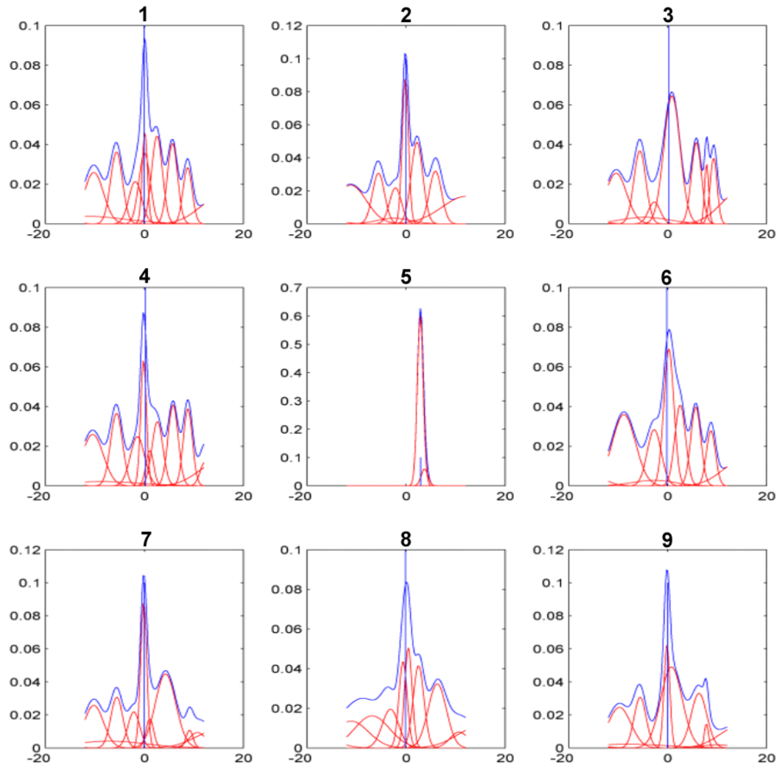


Figure 9: Next four figures: posterior distributions at each node of toy problem of Sect. 4.1.1, over several iterations. Figure 8 (a) shows the "image" data. Iteration 0: initial beliefs at each node before any message passing. The observations at all nodes have ambiguous interpretations, except for node 5, where $y = 3 \Rightarrow x = 3$, by the joint density of Fig. 5. Sampling is used when pruning the gaussian mixtures, so each pruned mixture (of the nodes other than 5) looks different.

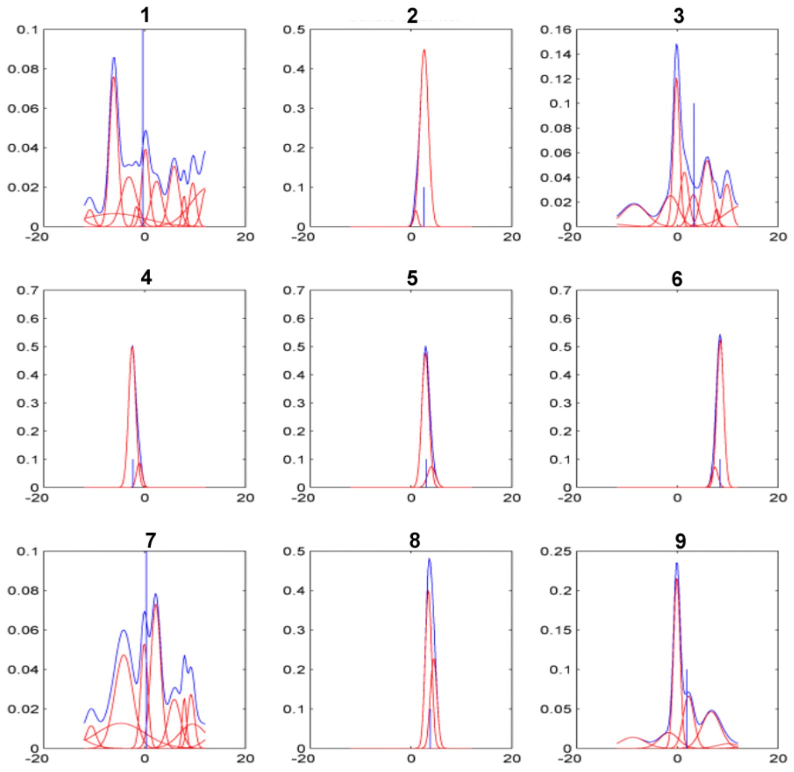


Figure 10: Beliefs at iteration 1. Nodes 2, 4, 6, and 8 have benefitted from the message passed from node 5.

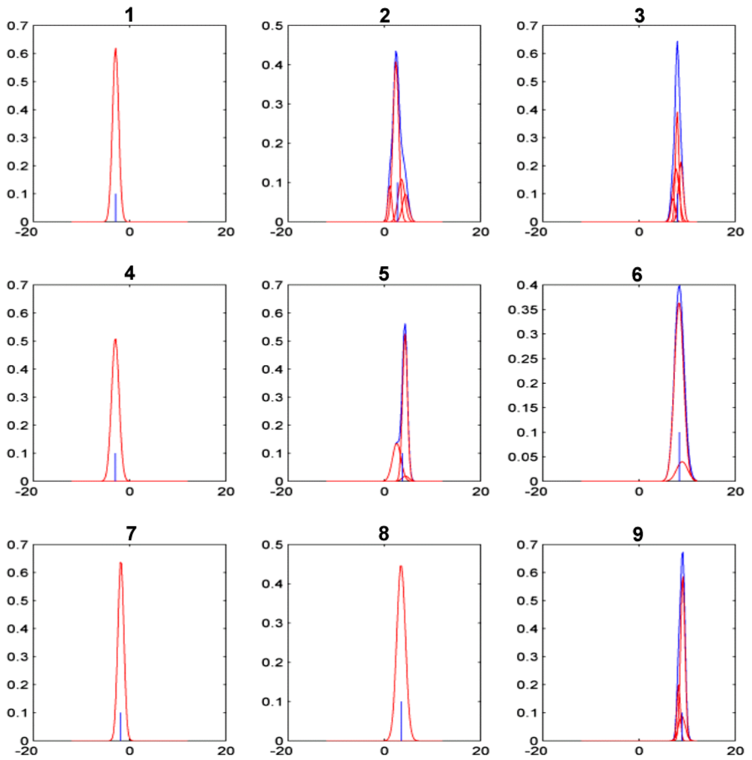


Figure 11: Beliefs at iteration 2. All nodes have heard the message from node 5. The posteriors have all attained their proper mean values (-2 for left column of nodes, 3 for middle column, 8 for right column).

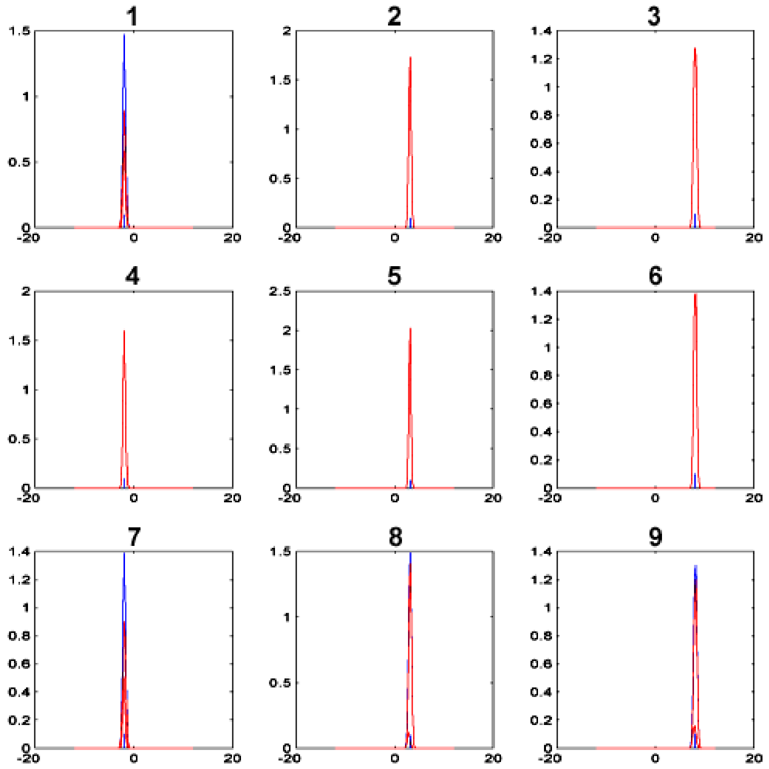


Figure 12: Beliefs at iteration 10. Running for 10 iterations artificially narrows the posterior distributions, from hearing the same information many times as if new, due to the network loops. However, the mean values are still the correct ones.

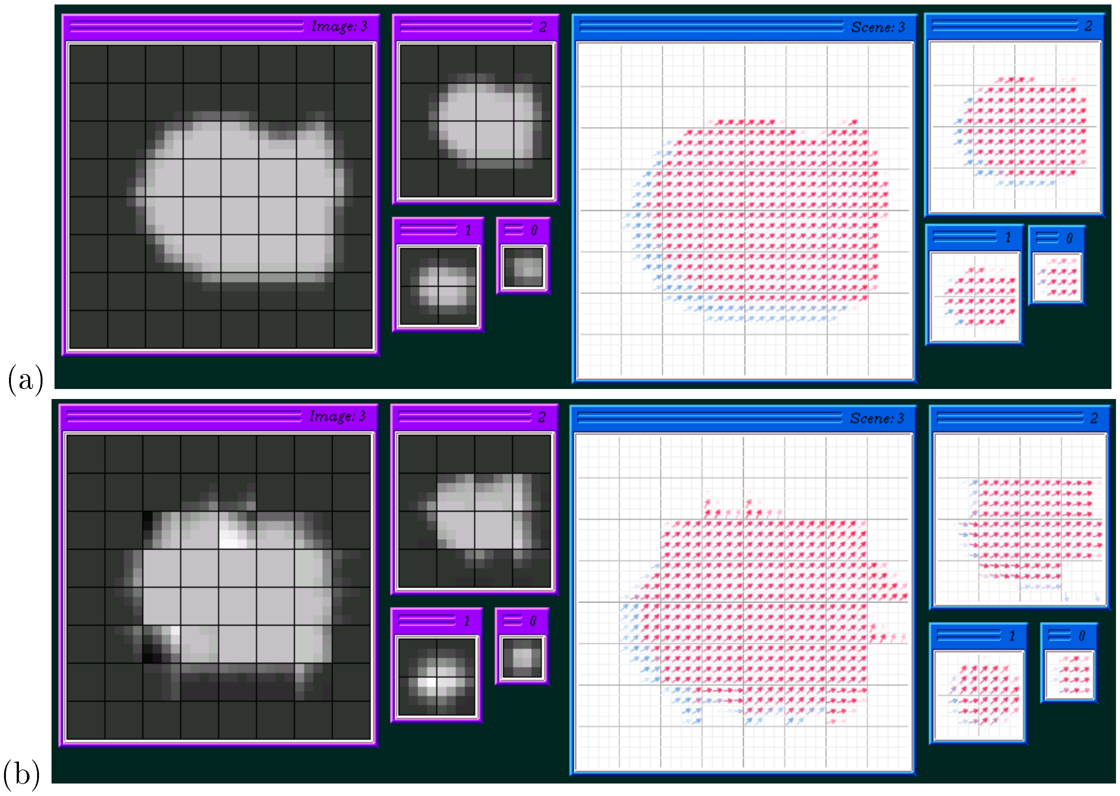


Figure 13: Motion estimation problem. (a) First of two frames of image data (in gaussian pyramid), and corresponding frames of velocity data. (b) Image and scene representations after vector quantization. The left side shows just one of two image frames. The right side shows (red) motion vectors from the second time frame obscuring (blue) motion vectors from the first. The scene representation contains both frames. Each large grid square is one node of the Markov network.

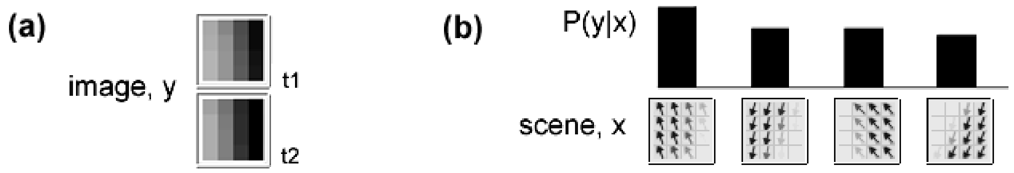


Figure 14: The local likelihood information for motion problem, vector quantized representation. Conditional probabilities are derived from co-occurance histograms. For a given image data sample, (a), the 4 most likely scene elements are shown in (b).

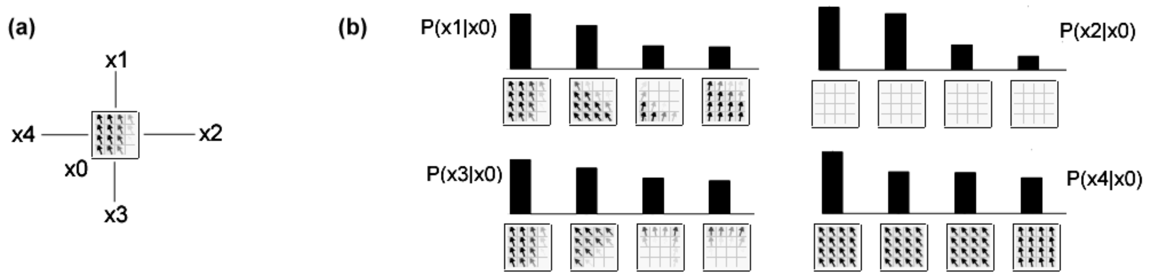


Figure 15: Some scene conditional probabilities, for the motion problem. For the given scene element, (a), the 4 most likely nodes from four different neighboring scene elements are shown. Scene elements also connect to each other across scale, not shown in this figure.

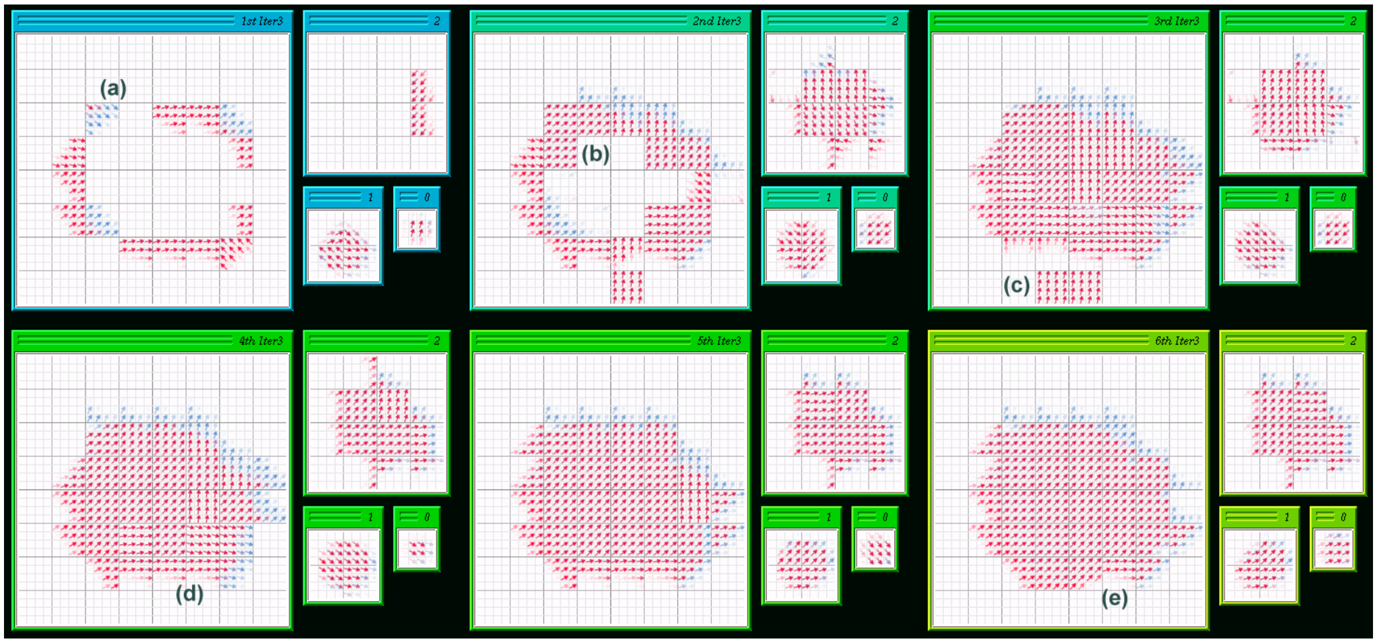


Figure 16: The most probable scene code for Fig. 13b at first 6 iterations of Bayesian belief propagation. (a) Note initial motion estimates occur only at edges. Due to the "aperture problem", initial estimates do not agree. (b) Filling-in of motion estimate occurs. Cues for figure/ground determination may include edge curvature, and information from lower resolution levels. Both are included implicitly in the learned probabilities. (c) Figure/ground still undetermined in this region of low edge curvature. (d) Velocities have filled-in, but do not yet all agree. (e) Velocities have filled-in, and agree with each other and with the correct velocity direction, shown in Fig. 13.

4.3 Continuous learning, sampled inference

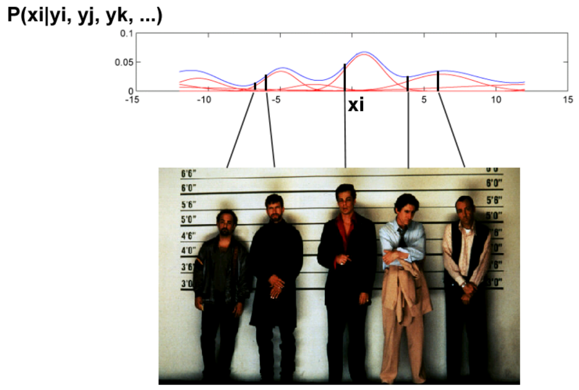


Figure 17: "Lineup of suspects" method for inference. (Photo from [1]). At each node, we gather a set of scenes, each of which explains the observed local image data. We evaluate the scene posterior only at those sample scenes. We sample appropriately from the continuous representation of the conditional probability with neighboring scenes to obtain discrete linking matrices between nodes. This allows for fast inference calculations in Eqs. 16 and 17.

Unfortunately, in a discrete representation, a faithful image representation requires so many vector codes that it becomes infeasible to measure the prior and co-occurance statistics. Note unfaithful fit of the vector quantized image and scene in Fig. 13. On the other hand, the discrete representation allows fast propagation during inference. We developed a hybrid method that allows both good fitting and fast propagation. We use a continuous representation during the learning phase, and a sampled one during inference.

We illustrate this favored method with a third application, "super-resolution". For super-resolution, the input "image" is the high-frequency components (sharpest details) of a sub-sampled image. The "scene" to be estimated is the high-frequency components of the full-resolution image, Fig. 19.

We describe the image and scene patches as vectors in a *continuous* space, and model the probability densities, $P(x)$, $P(y,x)$, and $P(x_n,x)$, as gaussian mixtures [7]. (We reduced the dimensionality of the scene and image data within each patch by principal components analysis [7]). We had approximately 20,000 patch samples from our training data, and typically used 9 dimensional representations for both images and scenes.

During inference, we evaluated the prior and conditional distributions of Eq. 17 only at a discrete set of scene values, different for each node. (This approach was inspired by the success of other sample-based methods [21, 11]). The scenes were a sampling of those scenes which render to the image at that node. We think of it as a "lineup of suspects", Fig. 17. Each node has its own set of suspects. Each scene in a node's lineup has in common the fact that it renders to the image observation at that node. We evaluate the likelihoods of the inference equations, Eqs. 16 and 17, only at those scene values for each node. This focusses our computation to just the locally feasible scene interpretations.

$P(x_k|x_j)$ in Eq. 17 becomes the ratios of the gaussian mixtures $P(x_k,x_j)$ and $P(x_j)$, evaluated at the scene samples at nodes k and j, respectively. $P(y_k|x_k)$ is $P(y_k,x_k)/P(x_k)$ evaluated at the scene samples of node k. This shows the benefit of our hybrid approach. To fit the image and scene well, each sample (suspect) has to have a very small bin of continuous parameter values that map to it. It would be prohibitive to learn the occurance and cooccurance statistics of such a fine-grain set of samples; one would have to wait forever for scene values to fall in those bins. So we do our learning in the continuous domain, where we can interpolate across parameter values, and do our inference in a discrete domain, where the calculations reduce to matrix operations. This "continuous learning/sampled inference" approach reduced the processing times from 24 hours to 10 or 15 minutes for the super-resolution problem.

To select the scene samples, we could condition the mixture $P(y,x)$ on the y observed at each node, and sample x's from the resulting mixture of gaussians. We obtained somewhat better results by using the scenes from the training set whose images most closely matched the image observed at that node. This avoided one gaussian mixture modeling step; our sampled inference actually gave better results than inference in the continuous representation.

Using 20 scene samples per node, setting up the $P(x_k|x_j)$ linking matrix for each link took about 10 minutes for these images (approximately 1000 nodes). The scene (high resolution) patch size was 3x3; the image (low resolution) patch size was 7x7. We didn't feel long-range scene propagation was critical here, so we used a flat, not a pyramid, node structure. Once the linking matrices were computed, the iterations of Eq. 17 were completed in a few minutes.

We performed experiments on two different sources of images, synthetic and natural. For a controlled training set, we trained on images of synthetically generated shaded and painted blobs, typified by Fig. 19 (a). Our training data were pairs of band-pass and high-frequency samples such as those shown in Fig. 18. We used local contrast normalization to reduce the variations we needed to model. Figure 19 illustrates the image processing steps and the reconstructed result.

Figure 20 shows the bandpassed results after 0, 1, 2, and 20 iterations of the algorithm. (This is the detail we add to Fig. 19 (c) to get Fig. 19 (d). After few iterations, the MAP estimate of the high resolution image is visually close to the actual full frequency image. The dominant structures are all in approximately the correct position. This may enable high quality zooming of low-resolution images, attempted with limited success by others [35, 32]. To illustrate that the problem is non-trivial, we include the nearest neighbor solution in Fig. 20. The choppiness there (and at iteration 0, before message passing) shows there are many different high-resolution scenes corresponding to any given low-resolution image patch.

In our second super-resolution experiment, we trained with images of tigers in the wild, from the Corel database. (Given the success of multi-resolution texture synthesis algorithms [19, 11, 43, 36], we wondered if high resolution texture scene components might be learned and synthesized). Figure 22 shows the frequency components of the image, and Figs. 24 and 25 show results. (Figure 23 is the result of a simple comparison algorithm, the nearest neighbor. The choppiness points out that given image data can have multiple possible local scene interpretations, requiring spatial propagation of some sort.) Good edge and texture synthesis in parts of Fig. 24 provide encouragement that this approach might be useful even for textured images. The solution is stable, and reaches convergence quickly.

For comparison, we show the processing result obtained using 5 samples (suspects) per node, instead of 20, Fig. 25. As might be expected, the result is a bit choppier.

To test the generalization ability, we used the probabilities trained on the tiger images with a non-tiger image, that of the teapot of Fig. 26. The tiger database contained few sharp, crisp edges, and the edges are not extrapolated well in the result. However, the results are good enough to show that the learned relationships between scenes and images generalizes some beyond strict image classes.

5 Discussion

In related applications of Markov random fields to vision, researchers often use relatively simple, heuristically derived expressions (rather than learned) for the likelihood function $P(y|x)$ or for the spatial relationships in the prior term on scenes [17, 33, 15, 24, 8, 27, 26, 34]. For other learning or constraint propagation approaches in motion analysis, see [27, 29, 23].

Weiss showed the advantage of belief propagation over regularization methods for several 1-d problems [41]; we are applying related methods to our 2-d problems.

The local probabilities we measure (local prior and likelihood, and conditional probability of neighboring scenes) have power and flexibility. For the motion problem, they lead to filling-in motion estimates in a direction perpendicular to object contours, Fig. 16 (a)–(c). For the super-resolution problem, those same corresponding probabilities lead to contour completion in the *direction* of the contours, Fig. 20 (d)–(f). The same learning machinery was used in each case, but trained for different problems, achieving different behavior, appropriate to each problem.

An appealing feature of this work is that it instantiates many of the intuitions developed vision researchers some time ago. We spatially propagate local interpretation hypotheses in the spirit of blocks world [40, 10] scene interpretation and the work of Barrow and Tenenbaum on intrinsic images [3, 4].

6 Summary

We have presented a general, learning-based method to solve low-level vision problems. The essence of the algorithm is measuring three probabilities in a synthetically generated labelled world: the local prior, the local likelihood, and the conditional relationship between scene neighbors. We placed image and scene regions in a Markov network. We used a factorization approximation to solve the network: we propagated information as if the network had no loops. This propagation is fast, and the messages passed between nodes are in-

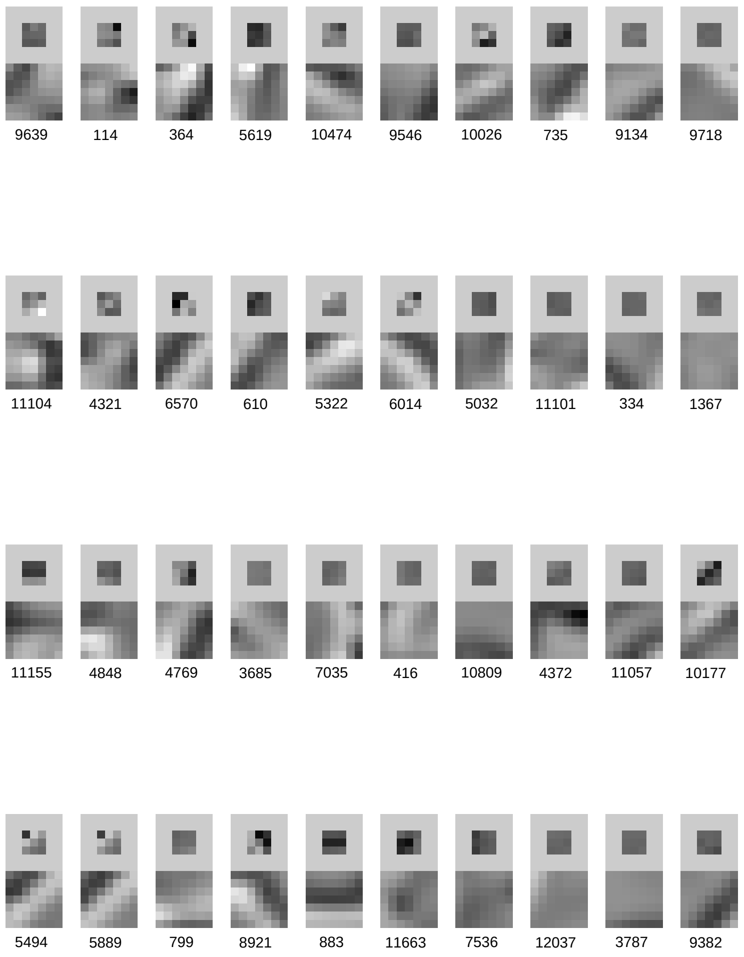


Figure 18: Training data, from images such as Fig. 19 (a), for super-resolution problem. The large squares are the image data (mid-frequency data). The small squares above them are the corresponding scene data (high-frequency data).

terpretable. Other approximation methods could be used here, if desired. In all our examples, we found this method led to stable and accurate inferences. This method is exact if the region likelihood functions are binary valued.

We explored the trade-offs between continuous and discrete representations with this method, opting for a hybrid approach: continous learning/sampled inference.

We considered three application examples: an explanatory toy problem, motion analysis, and super-resolution. In all three domains, the algorithms behaved as desired. Scene estimation by Markov networks may be useful for other low-level vision problems, such as extracting intrinsic images from line drawings or photographs.

Acknowledgements We thank E. Adelson, A. Blake, J. Tenenbaum, P. Viola, and Y. Weiss for helpful discussions.

References

[1] 20th Century Fox, 1998. http://www.geocities.com/Hollywood/8583/gallery.html.

[2] J. J. Atick, Z. Li, and A. N. Redlich. Understanding retinal color coding from first principles. *Neural Computation*, 4:559–572, 1992.

[3] H. G. Barrow and J. M. Tenenbaum. Recovering intrinsic scene characteristics from images. In A. R. Hanson and E. M. Riseman, editors, *Computer Vision Systems*, pages 3–26. Academic Press, New York, 1978.

[4] H. G. Barrow and J. M. Tenenbaum. Computational vision. *Proc. IEEE*, 69(5):572–595, 1981.

[5] A. J. Bell and T. J. Senjowski. The independent components of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[6] J. O. Berger. *Statistical decision theory and Bayesian analysis*. Springer, 1985.

[7] C. M. Bishop. *Neural networks for pattern recognition.* Oxford, 1995.

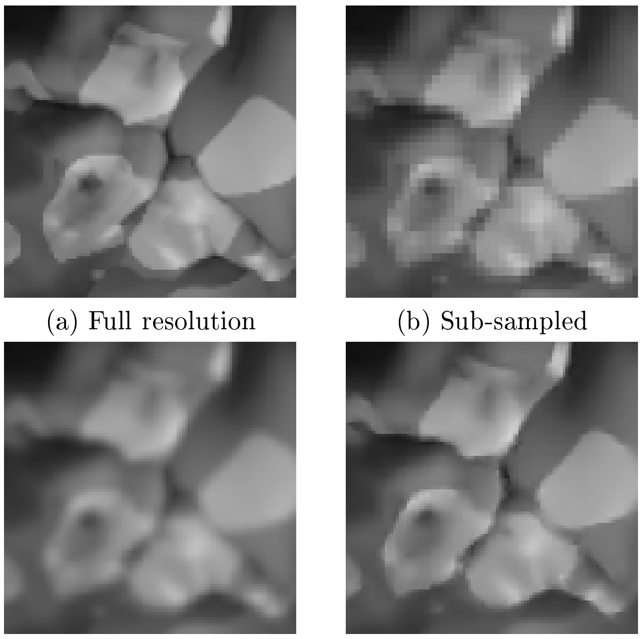


Figure 19: (a) Test image, full resolution. (b) Sub-sampled and zoomed. (c) is image (b), after blurring. (d) After 2 iterations of high frequency scene components, estimated from (c), are added to (c). Note that most sharp edges are identified and maintained as sharp edges.

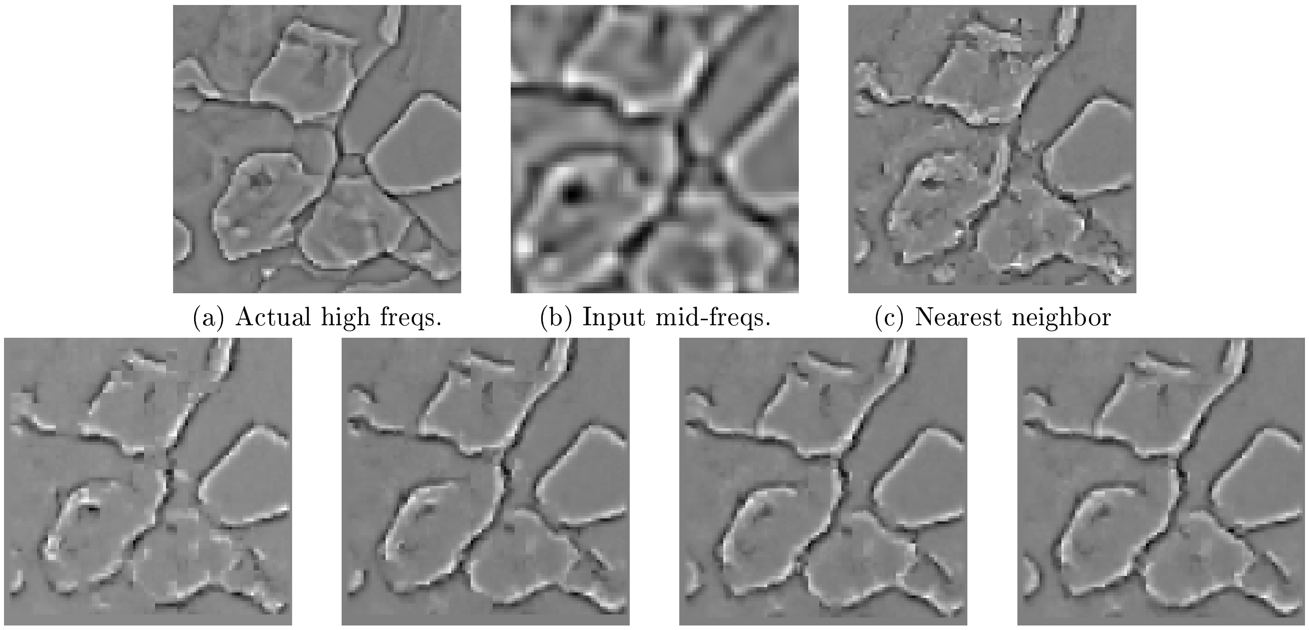


Figure 20: Bandpassed frequency components image of Fig. 19. (a) The high freqs. of Fig. 19 (a), which we seek to infer. (b) Bandpass frequencies, from which we estimate (a). (c) Nearest neighbor solution: scene patch corresponding to the nearest image patch in the training data to each image patch in (b). The choppiness shows that simply a local estimate is not sufficient. (c) Markov network inference MAP solution, iteration 0 (no message passing). (d), (e), and (f): iterations 1, 2, and 20. Note the line continuations over iterations 0, 1, and 2. The bottom row shows the stable and spatially consistent solution after very few iterations of belief propagation.

[8] M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *Proc. 4th Intl. Conf. Computer Vision*, pages 231–236. IEEE, 1993.

[9] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Trans. Comm.*, 31(4):532–540, 1983.

Figure 21: Three random draws from the Corel database set of "tigers" images, used as the training set for Figs. 24, 25, and 26. The training database consisted of 80 images like these. (The test image of Fig. 24 was excluded).

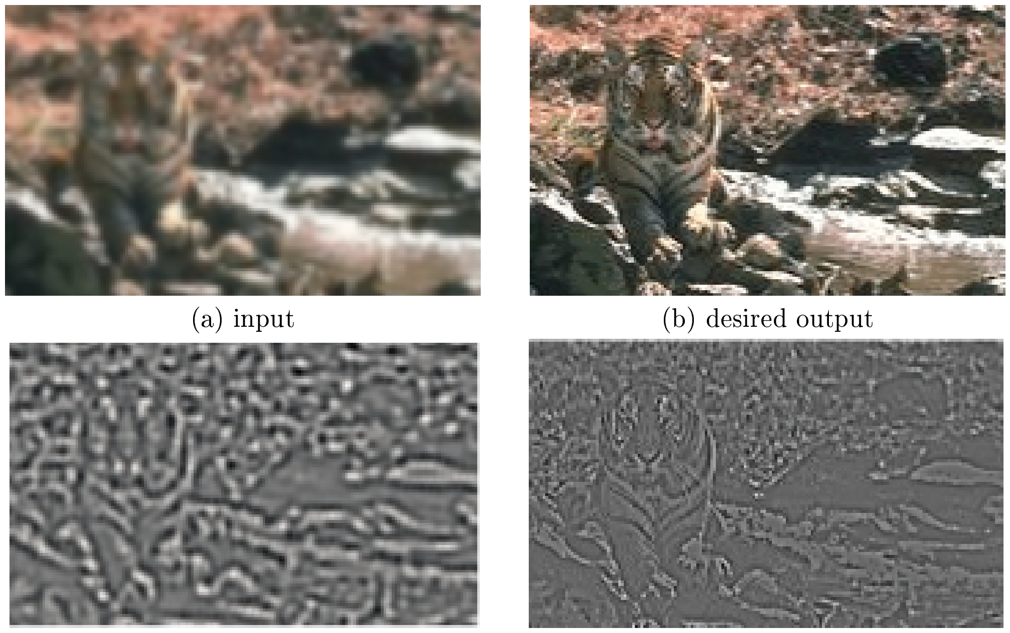


Figure 22: We want to estimate (b) from (a). The original image, (b) is blurred, subsampled, then interpolated back up to the original resolution to form (a). The missing high frequency detail, (b) - (a), is the "scene" to be estimated, (d) (this is the first level of a Laplacian pyramid [9]). The low frequencies of (a) are removed to form the input "image", (c). To reduce the variability that the model needs to learn, we form a local contrast image from the "image" band and use that to scale the image and scene bands before modeling. That scaling is undone at reconstruction.

[10] M. B. Clowes. On seeing things. *Artificial Intelligence*, 2:79–116, 1971.

[11] J. S. DeBonet and P. Viola. Texture recognition using a non-parametric multi-scale statistical model. In *Proc. IEEE Computer Vision and Pattern Recognition*, 1998.

[12] W. T. Freeman and D. H. Brainard. Bayesian decision theory, the maximum local mass estimate, and color constancy. In *Proc. 5th Intl. Conf. on Computer Vision*, pages 210–217. IEEE, 1995.

[13] W. T. Freeman and E. C. Pasztor. Learning to estimate scenes from images. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Adv. Neural Information Processing Systems*, volume 11, Cambridge, MA, 1999. MIT Press.

[14] B. J. Frey. *Bayesian networks for pattern classification.* MIT Press, 1997.

[15] D. Geiger and F. Girosi. Parallel and deterministic algorithms from MRF's: surface reconstruction. *IEEE Pattern Analysis and Machine Intelligence*, 13(5):401–412, May 1991.

[16] A. Gelb, editor. *Applied optimal estimation.* MIT Press, 1974.

[17] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[18] R. M. Gray, P. C. Cosman, and K. L. Oehler. Incorporating visual factors into vector quantizers for image compression. In A. B. Watson, editor, *Digital images and human vision.* MIT Press, 1993.

[19] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *ACM SIGGRAPH*, pages 229–236, 1995. In *Computer Graphics* Proceedings, Annual Conference Series.

[20] A. C. Hurlbert and T. A. Poggio. Synthesizing a color algorithm from examples. *Science*, 239:482–485, 1988.

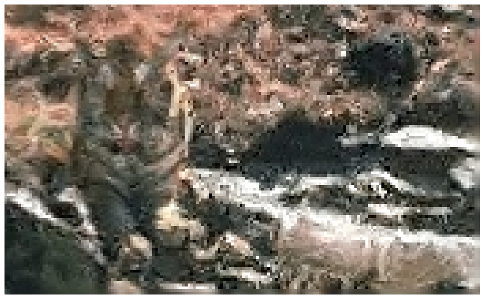


Figure 23: Nearest neighbor solution, using the high frequency components of the nearest neighbor in the training data to each image patch. The result is too choppy, showing that a solution based on local information alone is not sufficient.

[21] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. European Conf. on Computer Vision*, pages 343–356, 1996.

[22] M. I. Jordan, editor. *Learning in graphical models.* MIT Press, 1998.

[23] S. Ju, M. J. Black, and A. D. Jepson. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 307–314, 1996.

[24] D. Kersten. Transparancy and the cooperative computation of scene attributes. In M. S. Landy and J. A. Movshon, editors, *Computational Models of Visual Processing*, chapter 15. MIT Press, Cambridge, MA, 1991.

[25] D. Kersten, A. J. O'Toole, M. E. Sereno, D. C. Knill, and J. A. Anderson. Associative learning of scene parameters from images. *Applied Optics*, 26(23):4999–5006, 1987.

[26] D. Knill and W. Richards, editors. *Perception as Bayesian inference.* Cambridge Univ. Press, 1996.

[27] M. R. Luettgen, W. C. Karl, and A. S. Willsky. Efficient multiscale regularization with applications to the computation of optical flow. *IEEE Trans. Image Processing*, 3(1):41–64, 1994.

[28] D. J. C. Mackay and R. M. Neal. Good error–correcting codes based on very sparse matrices. In *Cryptography and coding – LNCS 1025*, 1995.

[29] S. Nowlan and T. J. Senjowski. A selection model for motion processing in area MT of primates. *J. Neuroscience*, 15:1195–1214, 1995.

[30] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[31] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 1988.

[32] A. Pentland and B. Horowitz. A practical approach to fractal-based image compression. In A. B. Watson, editor, *Digital images and human vision.* MIT Press, 1993.

[33] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317(26):314–139, 1985.

[34] E. Saund. Perceptual organization of occluding contours of opaque surfaces. In *CVPR '98 Workshop on Perceptual Organization*, Santa Barbara, CA, 1998.

[35] R. R. Schultz and R. L. Stevenson. A Bayesian approach to image expansion for improved definition. *IEEE Trans. Image Processing*, 3(3):233–242, 1994.

[36] E. P. Simoncelli. Statistical models for images: Compression, restoration and synthesis. In *31st Asilomar Conf. on Sig., Sys. and Computers*, Pacific Grove, CA, 1997.

[37] E. P. Simoncelli and O. Schwartz. Modeling surround suppression in V1 neurons with a statistically-derived normalization model. In *Adv. in Neural Information Processing Systems*, volume 11, 1999.

[38] P. Sinha and E. H. Adelson. Recovering reflectance and illumination in a world of painted polyhedra. In *Proc. 4th Intl. Conf. Comp. Vis.*, pages 156–163. IEEE, 1993.

[39] R. Szeliski. *Bayesian Modeling of Uncertainty in Low-level Vision.* Kluwer Academic Publishers, Boston, 1989.

[40] D. Waltz. Generating semantic descriptions from drawings of scenes with shadows. In P. Winston, editor, *The psychology of computer vision*, pages 19–92. McGraw-Hill, New York, 1975.

[41] Y. Weiss. Interpreting images by propagating Bayesian beliefs. In *Adv. in Neural Information Processing Systems*, volume 9, pages 908–915, 1997.

[42] Y. Weiss. Belief propagation and revision in networks with loops. Technical Report 1616, AI Lab Memo, MIT, Cambridge, MA 02139, 1998.

[43] S. C. Zhu and D. Mumford. Prior learning and Gibbs reaction-diffusion. *IEEE Pattern Analysis and Machine Intelligence*, 19(11), 1997.

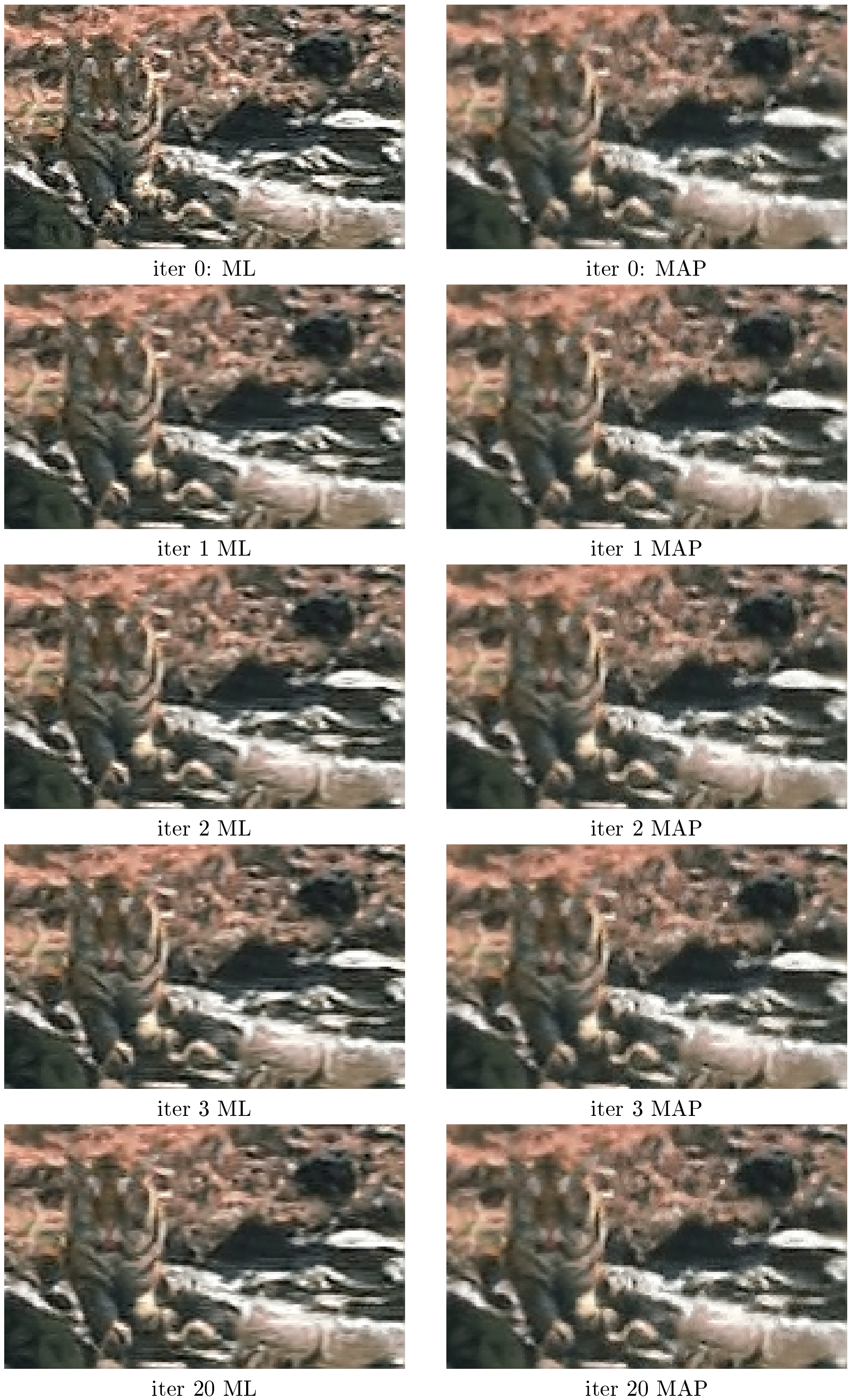


Figure 24: Reconstructed full-resolution images, using the scene estimates from belief propagation in the Markov network. We present both the maximum likelihood solution ($P(x)$ not multiplied into the final belief, Eq. 16) and the MAP solution (Eq. 16). The prior pulls estimated detail somewhat toward zero. After just a few iterations, the algorithm converges to a good solution. That solution is stable (see iteration 20 result). Compare with Fig. 22.

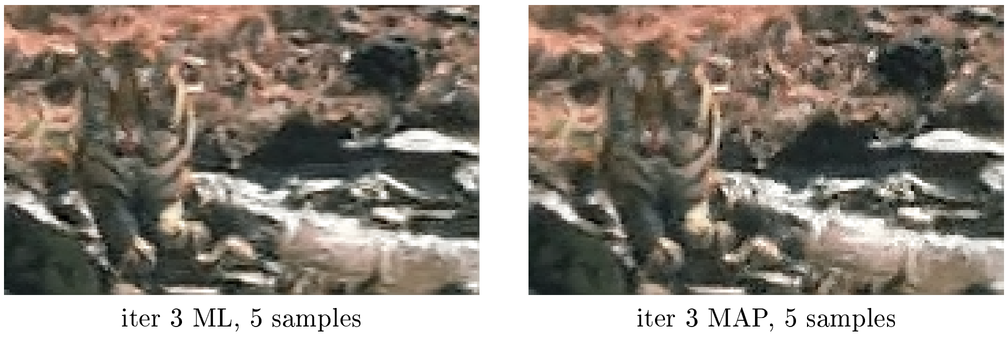


Figure 25: For a comparison of implementation choices, we show the result of using 5 scene samples at each node, instead of the 20 used in Fig. 24. The image is a bit choppier, which for the textured regions can be a preferred effect.

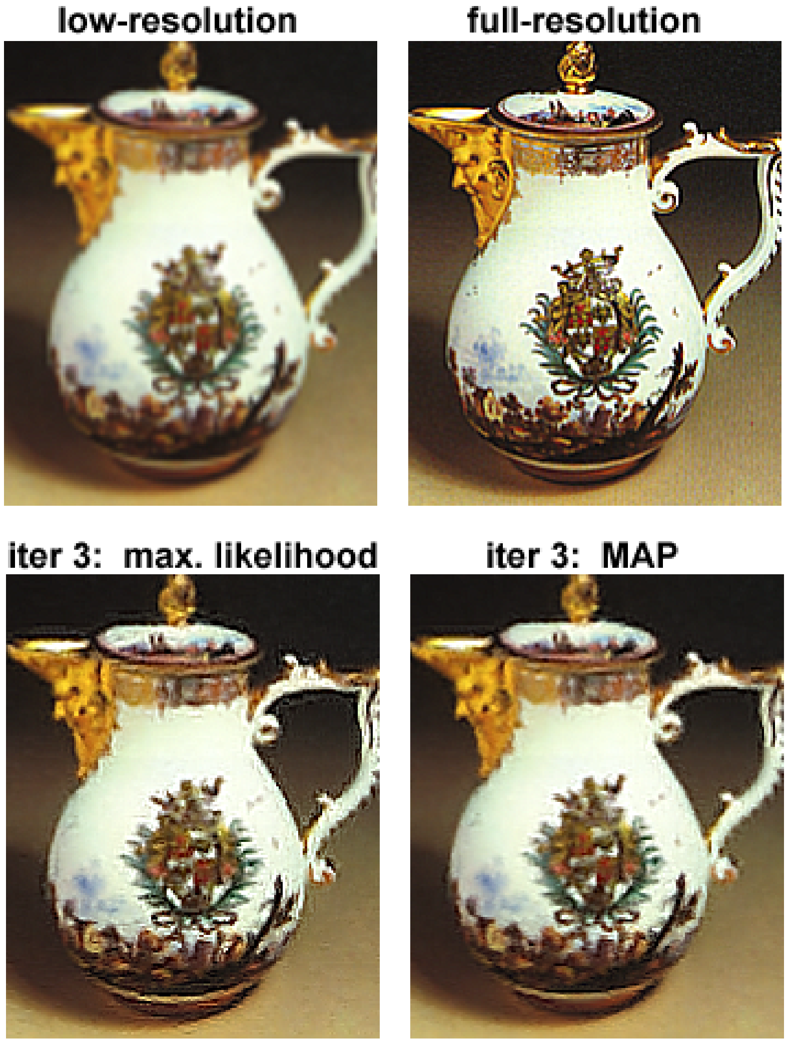


Figure 26: Result from training on the *tiger* image database and running on this *teapot* image. The input is the "low-resolution" image; the desired output is the "full-resolution" image. Crisp edges, of which there are few in the tiger training set, are not rendered well. However, the textured, painted pattern on the teapot is extrapolated reasonably well from the low-resolution original.