Proceedings of the IEEE Workshop on Statistical and Computational Theories of Vision, Fort Collins, CO, June, 1999. (Published on Web).

Learning low-level vision

William T. Freeman[1] and Egon C. Pasztor[1]

Abstract

We show a learning-based method for low-level vision problems. We set-up a Markov network of patches of the image and the underlying scene. A factorization approximation allows us to easily learn the parameters of the Markov network from synthetic examples of image/scene pairs, and to efficiently propagate image information. Monte Carlo simulations justify this approximation. We apply this to the "super-resolution" problem (estimating high frequency details from a low-resolution image), showing good results. For the motion estimation problem, we show resolution of the aperture problem and filling-in arising from application of the same probabilistic machinery.

This is also distributed as MERL TR99-12, **http://www.merl.com/reports/TR99-12/index.html**. For related technical reports, please see **http://www.merl.com/reports/TR99-08/index.html** and **http://www.merl.com/reports/TR99-05/index.html**.

1. MERL, a Mitsubishi Electric Res. Lab.
201 Broadway
Cambridge, MA 02139
`freeman, pasztor@merl.com`.

1 Introduction

We seek a machinery for learning low-level vision problems. These problems could include: motion analysis, inferring shape and albedo from a photograph, or estimating colors. For these problems, given *image* data, we want to estimate an underlying *scene*. The scene quantities to be estimated could be projected object velocities, surface shapes, reflectance patterns, colors, or missing high frequency details.

Low-level vision problems are typically under-constrained, so Bayesian [3, 19, 31] and regularization techniques [26] are fundamental. There has been much work and progress (for example, [19, 20, 12]), but difficulties remain in working with complex, real images. Typically, prior probabilities or constraints are made-up, rather than learned. A general machinery for a learning-based solution to low-level vision problems would have many applications.

A second research theme has been to learn the statistics of natural images. Researchers have related those statistics to properties of the human visual system [23, 2, 30], or have used statistical methods with biologically plausible image representations to analyse and synthesize realistic image textures [11, 6, 34, 30]. These methods may help us understand the early stages of representation and processing, but unfortunately, they don't address how a visual system might *interpret* images, i.e., estimate the underlying scene.

We want to combine those two themes of scene estimation and statistical learning. We study the statistical properties of a synthetically generated, *labelled*, visual world of images with scenes, to learn how to infer scenes from images. Our prior probabilities can be rich ones, learned from training data.

Several researchers have applied related learning approaches to low-level vision problems, but restricted themselves to linear models [17, 13], too weak for many applications. We gather full statistics on *local regions* of images and scenes. We ask whether a visual system can interpret an image if it models (1) the probability that any local scene generated the local image, and (2) the probability that any local scene neighbors any other. The first probabilities allow initial scene estimates; the second allow the estimates to propagate.

The resulting method applies the Bayesian machinery of graphical models [24, 16] to low-level vision. We were inspired by the work of Weiss [32], who pointed out the speed advantage of Bayesian methods over conventional relaxation methods for propagating local measurement information.

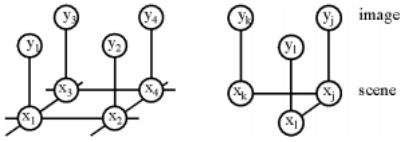
Our method uses three approximations: the Markov assumption, a factorization approximation, and sampling. It is a general machinery that should apply to various problems. We show it working for three examples: (1) joint gaussian processes, where we can compare our factorization approximation with the true answer, (2) estimating missing image details, and (3) estimating motion.

2 Markov network

For given image data, y, we seek to estimate the underlying scene, x (we omit the vector symbols for notational simplicity). We take a Bayesian approach [3]. We first calculate the posterior probability, $P(x|y)$, using Bayes rule, $P(x|y) = cP(y|x)P(x)$. For this analysis, we ignore the normalization, $c = \frac{1}{P(y)}$, a constant over x. $P(y|x)$ is the *likelihood*; $P(x)$ is

the *prior*. Under two common loss functions [3], the best scene estimate, $\hat{x}$, is the mean (minimum mean squared error, MMSE) or the mode (Maximum a posteriori, MAP) of the posterior probability.

In general, $\hat{x}$ can be difficult to compute [19] without approximations. The first of our three approximations is the Markov assumption. We divide both the image and scene into patches, and assign one node of a Markov network [24, 16] to each patch. We connect each scene patch to its corresponding image data patch, as well as to its nearest neighbors, see Fig. 1a). (We also allow connections to scene or image nodes at different resolutions or orientations). The connections indicate statistical dependencies. This network implies that knowing the scene at any position: (1) provides all the information about the image there (it has the only link to y), and (2) gives some information about nearby scenes and images (by links to nearby scene neighbors). We will call problems with this property low-level vision problems.



(a) (b)

Figure 1: (a) Markov network for vision problems. Observations, y, have underlying scene explanations, x. Connections between nodes of the graphical model indicate statistical dependencies. (b) Configuration of nodes used in explanation of factorization method, Sect. 2.1.

This network description simplifies the posterior probability [10, 9], and will allow estimating $\hat{x}$ everywhere using only local operations. Solving a Markov network involves a *learning* phase, where the parameters of the network connections are learned from training data, and an *inference* phase, when the scene corresponding to particular image data is estimated. Even using the Markov assumption, finding the posterior probability distribution for the grid-structured Markov network is computationally expensive. A variety of approximations have been proposed [10, 16].

If a network has no loops, the Markov assumption allows the posterior probability to factorize in a useful way [24, 21, 32, 16, 8]. Our second approximation will be to solve our loopy Markov network *as if there were no loops*, both during learning and inference. Thus, in Sect. 2.1, we first derive the factorization rules for networks without loops. In Sect. 3, we study the validity of our "factorization approximation" applied to networks with loops.

2.1 Factorization

There are different ways to factorize the posterior probability of a Markov network [24, 21, 32, 16, 8]. In a non-loopy network, they all give the same answer. We use a simple, valid factorization that we believe to be new, resulting in interpretable messages passed between nodes.

At each iteration, the scene estimate x_j at each node, j, is optimal, given the observations that node j has heard from. We derive our message passing algorithm by first

considering the optimal estimate, then finding what messages need to be passed to allow its computation.

Before any messages are passed, scene node j has only heard from the observation y_j. The MMSE estimate is the mean of the corresponding posterior, $P(x_j|y_j) = cP(y_j|x_j)P(x_j)$ (c is a normalization constant). We call $P(y_j|x_j)$ the *local likelihood*, and $P(x_j)$ the *local prior*.

At the next iteration, scene node j receives messages from its connecting scene nodes, say k and l (Fig. 1b). It has then "heard" from observations y_j, y_k, and y_l, since scene nodes k and l heard from observations y_k, and y_l on the previous iteration. The MMSE estimate for x_j is therefore the mean of the posterior probability, given the data it has heard from: $P(x_j|y_j, y_k, y_l) = cP(y_j, y_k, y_l|x_j)P(x_j)$. We call $P(y_j, y_k, y_l|x_j)$ a *region likelihood*, the likelihood of a region of image data, given a scene node.

Now we exploit the Markov structure. If the network has no loops, then y_j, y_k, and y_l are conditionally independent, given x_j, because they all link to x_j by independent paths.Thus we have $P(y_j, y_k, y_l|x_j) = P(y_j|x_j)P(y_k|x_j)P(y_l|x_j)$. We see that if the messages that scene nodes k and l pass to scene node j were the region likelihoods $P(y_k|x_j)$ and $P(y_l|x_j)$, then node j could compute the appropriate posterior probability by multiplying together the region likelihoods passed from its neighbors, the local likelihood, and the local prior. The passed messages should be those region likelihoods.

If we knew how to calculate the region likelihood to pass to a neighbor at the next iteration, the algorithm would be complete. What message should node j pass to node l? It should pass the region likelihood (given x_l) of the observations that node j has heard from, but node l has not: $P(y_j, y_k|x_l)$. We know this region likelihood, but given node x_j: $P(y_j, y_k|x_j) = P(y_j|x_j)P(y_k|x_j)$. (This is the local likelihood times the messages from all incoming nodes except the node being sent a message.) We want to swap x_l for x_j in $P(y_j, y_k|x_j)$. If we multiply by the probability $P(x_j|x_l)$ ("near given far"), and recognize that the Markov assumption lets us add conditioning on x_l, we have

$$\begin{aligned} P(y_j, y_k|x_j)P(x_j|x_l) &= P(y_j, y_k|x_j, x_l)P(x_j|x_l) \\ &= P(y_j, y_k, x_j|x_l). \end{aligned} \tag{1}$$

We then marginalize over x_j to find the desired region likelihood, $P(y_j, y_k|x_l)$. That is the message that node j passes to node l.

Summarizing this argument, after each iteration, the MMSE estimate at node j, $\hat{x}_j$ is

$$\hat{x}_j = \int_{x_j} x_j P(x_j) P(y_j|x_j) \prod_k L_{kj}, \tag{2}$$

where k runs over all scene node neighbors of node j. $\tilde{L}_{lk}$ is the region likelihood, given x_j, of all the observations that node k communicates to node j. We calculate L_{kj} from:

$$L_{kj} = \int_{x_k} P(x_k|x_j) P(y_k|x_k) \prod_{l \neq j} \tilde{L}_{lk}, \tag{3}$$

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.

For a discrete probability representation, we replace the integral signs with sums over the discrete states of the random variables. To use the MAP estimator, instead of MMSE, the above arguments hold, with two changes (as with [33]): the "$\int_{x_j} x_j$" in Eq. (2) becomes argmax_{x_j}, and "$\int_{x_k}$" in Eq. (3) becomes $\max_{x_k}$.

2.2 Loops

If the Markov network contains loops, then the different 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 various exact or approximate methods to choose from [4, 16, 8].

We have found good results by using a somewhat non-standard approximation: we apply the factorized learning and propagation rules to our very loopy network. Note the appeal of doing so. We can learn the network parameters, $P(x_j)$, $P(y_j|x_j)$, and $P(x_k|x_j)$, by measuring the statistics at a node and its local neighbors. Scene inference from image data involves only local computations, Eq. (3).

Others have applied the factorized propagation rules during inference to networks of one or two loops [8, 22, 33] and obtained good results. Weiss provides theoretical arguments why this works for one-loop networks [33]. We want to know: is the approximation accurate for our (larger) networks? Does it work for estimating the network parameters (learning) as well as for propagation (inference)?

3 Example 1: Joint Gaussian Processes

We study these questions with gaussian random processes. We describe the joint probabilities of all the image and scene vectors as a multi-dimensional gaussian. It is possible to design a covariance matrix which respects the statistical dependencies defined by the Markov network [16]. Thus, we can solve the problem two ways: (1) exactly, by conditioning the joint gaussian on a set of observations and finding the posterior probability, and (2) using our factorization approximation. We learn the conditional probabilities needed for propagation, $P(y_j|x_j)$ and $P(x_j|x_k)$, by marginalizing the joint gaussian.

Because our experience with real images shows convergence in few iterations, we performed our numerical experiments on relatively small grids of 3x3 and 4x4 nodes. We used randomized covariance matrices, constrained to be translationally invariant. The synthetic image data was drawn from the true joint gaussian (simulating seeing another example from the world).

Figure 2 shows an example showing typical convergence results. The dotted red curves are the true marginalized posterior at each node. The isolated gray curves are the priors; the darker curves show the belief at different iterations, ending up overlaid on the true posterior's red curve. In our experiments, after 5 iterations (typical for our applications), 87% of the runs were within 0.2 posterior probability standard deviations of the true posterior mean, $\hat{x}$; 71% were within 0.1 std. dev. On average, the factorization approximation gives a very good solution to the true posterior even for loopy networks, for jointly gaussian processes. This result encourages us to test the method on real scenes.

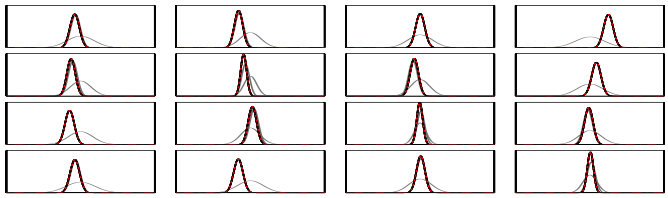


Figure 2: Showing good factorization approximation performance.

4 Probability Representations

Before continuing to applications, we need to choose a representation for the probabilities of Eqs. (2) and (3). A discrete representation allocates a discrete set of symbols to images and scenes. Propagation during inference is fast; Eq. (3) reduces to vector and matrix operations. However, the learning phase poses a problem. One needs to measure $P(y_j|x_j)$ and $P(x_k|x_j)$, using image/scene and scene/neighbor co-occurance histograms. In practise, if the symbols are few enough to measure the co-occurance histograms well from the training data, then that symbol alphabet will be too coarse to fit the image well.

This might argue for a continuous representation. One could represent the probabilities as mixtures of gaussians, and the images and scenes as vectors of real values. This approach allows images and scenes to be fit well, and the learning phase is feasible, using EM [5]. But now propagation is difficult: the mixtures must be multiplied. This requires a pruning and merging step, to avoid a runaway of gaussians in the product of mixtures, making inference slow.

We developed a hybrid approach, which we call *sampled inference*. We use a *continuous* representation during the learning phase, which allows us to fit real-world image data well, and learn the required probabilities. We use a *discrete* representation during inference, which allows fast message propagation. (We were inspired by other sample-based algorithms [14, 6].) This is the third of our three approximations.

For the image observation at each node, we form a "line-up of suspects" – a collection of scene values, each of which accounts for the observation. We search the training data for the k (usually 10) closest observations to the image at this node. The corresponding scenes from the training data forms the group of suspects[1]. We focus our computation to those scenes that explain the local image.

In this discretization, the conditional probabilities, $P(x_j|x_k)$, used in propagation, Eq. (3), become linking matrices. They are the ratio of two mixtures of gaussians, $\frac{P(x_j,x_k)}{P(x_k)}$, evaluated at the outer product of all x_j scenes with all x_k scenes. (Evaluating this ratio only at valid scenes also avoids the singularity problems of computing the conditional everywhere).

This method reduced propagation times for the super-resolution problem from 24 hours to 2 minutes, with no quality reduction. However, each image still requires 45 minutes to initialize all the linking matrices. We are studying speed-up approximations.

[1] One could also gather the candidate scenes at node j by sampling from the mixture fit to $P(y_j, x_j)$, evaluated at the observation y_j

5 Example 2: super-resolution

For the super-resolution problem, the input "image" is a low-resolution image. The "scene" to be estimated is a higher resolution image. A visually appealing solution to this problem would allow image data to be treated in a resolution-independent manner. Applications could include enlargment of digital or film photographs, upconversion of video from NTSC format to HDTV, or image compression.

At first, the task may seem impossible, after all, the high resolution data is not there. However, we can see edges in the low-resolution image that we know should remain sharp at the next resolution level. Based on the successes of recent texture synthesis methods [11, 6, 34, 30], we might expect to handle textured areas well, too.

Others [29] have used a Bayesian method, making-up the prior probability. In contrast, the Markov network will build its prior from large amounts of training data, and achieves better results. Among the non-Bayesian methods, fractal image representation [27] (Fig. 5c) only gathers training data from the one image, while selecting the nearest neighbor from training data [25] misses important spatial consistancy constraints (Fig. 3c).

We apply the Markov network as follows. The "image" we start from is a small image that has been linearly interpolated to twice the pixel resolution. The "scene" to be estimated is the high frequency image one needs to add-in to create the true image at the higher pixel resolution.

By downsampling (with anti-aliasing) training images, we can construct a training set of blurred and sharp image pairs. To ease the modeling burden, we bandpass filter the blurred image, and use its local contrast [15] to normalize both itself and the desired output highpassed image, because we believe their relationship should be contrast independent. We undo that normalization after estimation.

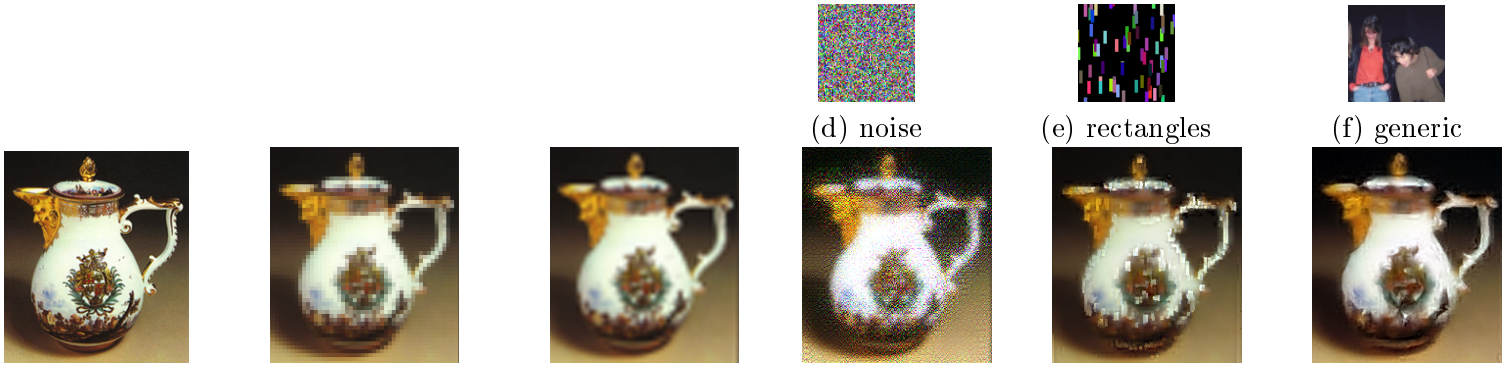
We extracted center-aligned 7x7 and 3x3 pixel patches from the images and scenes (the image patches overlapped). Applying PCA [5] to the training set, we summarize each 3-color patch of image or scene by a 9-d vector. From 40,000 image/scene pair samples, we fit 15 cluster gaussian mixtures to the local probabilities, assuming spatial translation invariance.

To infer the high frequency scene for a given image, we found the 10 training samples closest to the image data at each node (patch). The 10 corresponding scenes are the "suspects" for that node. We evaluated $\frac{P(x_j,x_k)}{P(x_k)}$ at the 100 scene points (10 x_j $\times$ 10 x_k points) to form a unique linking matrix between scene neighbors. We propagated the probabilities by Eq. (3). We found more visually pleasing reconstructions by using a maximum likelihood reconstruction (omitting the conservative local prior in Eq. (2).

Figure 3 shows the relevant full-frequency and bandpassed images, trained on simple world of random variations of the scene elements depicted. As with all these super-resolution examples, the algorithm converges after 2 or 3 iterations. Figure 4 applies our algorithm to a very low-resolution input image, 48x55 pixels, (zooming twice) showing that the results properly reflect the training data. Figure 5 shows a higher resolution test image (a, 70x70). The resulting twice zoomed estimate is very good, (d) and (e). There is little difference between using a training set taken at the same time as the test image (d), and using a set of unrelated pictures (e). Both results look much better than competing methods, cubic spline interpolation in Adobe Photoshop (b), and fractal expansion (c).

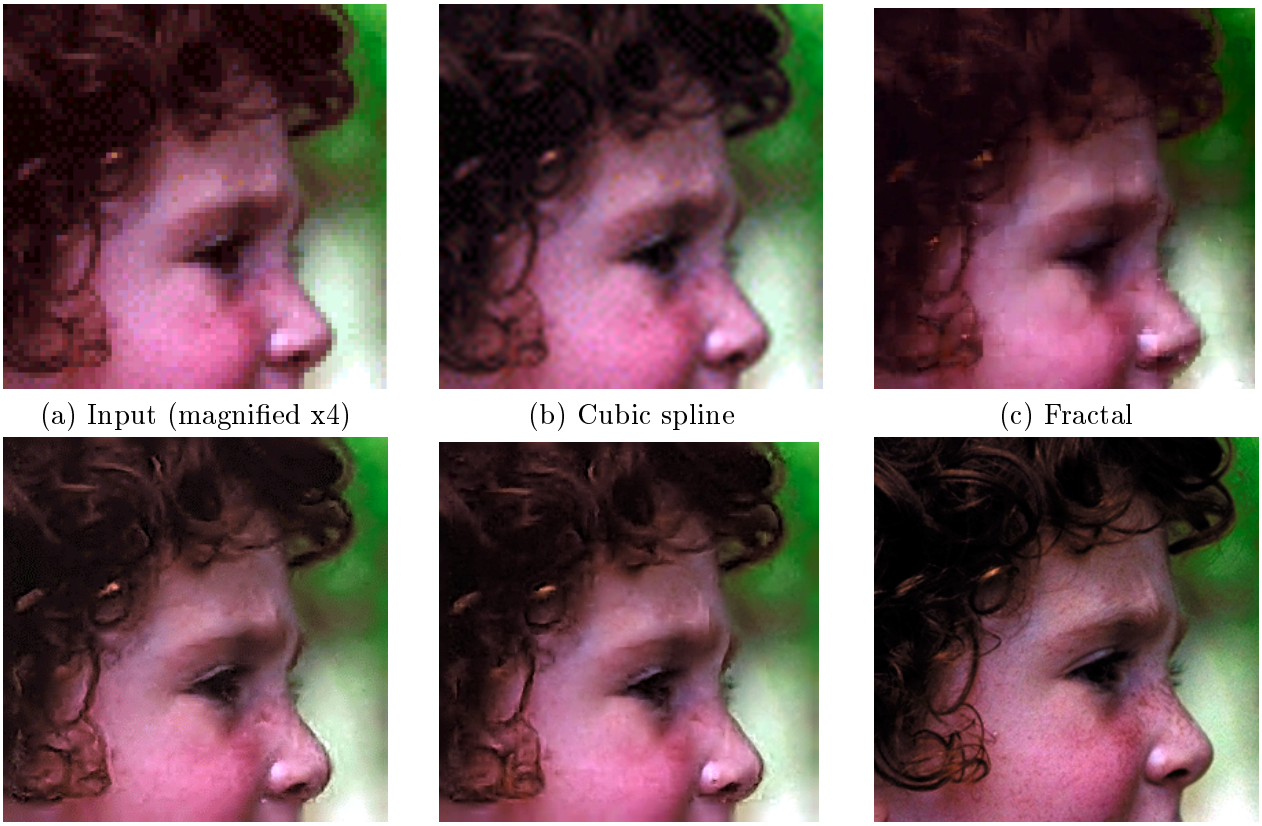
(a) (b) (c) (d) (e) (f) (g)

Figure 3: (a) Sub-sampled input. Bandpassed frequency components follow. (b) The true high freqs. (c) Nearest neighbor: use scene patch corresponding to the nearest image patch in the training data. Many high freq. scenes render to the same low-res image, giving choppiness. (d) Markov network inference MAP solution, iteration 0 (no message passing). (e), (f), and (g): iterations 1, 2, and 20. Note the line continuations over iterations 0, 1, and 2. Note the stable and spatially consistent solution after few iterations.



(a) Actual (b) Input (mag. x4) (c) Cubic spline (d) noise (g) Train: noise (e) rectangles (h) Train: rects (f) generic (i) Train: generic

Figure 4: Super-resolution example. (a) was blurred, and subsampled by 4 in each dimension to yield the low-resolution input, (b). Cubic spline interpolation to full resolution in Adobe Photoshop loses the sharp edges, (c). We recursively zoomed (b) up two factors of two using the Markov network trained on 10 images from 3 different "worlds": (d) random noise, (e) colored rectangles, and (f) a generic collection of photographs. The estimated high resolution images, (g), (h), and (i), respectively, reflect the statistics of each training world.



(a) Input (magnified x4) (b) Cubic spline (c) Fractal

(d) "Picnic" training set (e) "Generic" training set (f) Actual full-resolution

Figure 5: (a) Low-resolution input image. (b) Cubic spline 400% zoom in Adobe Photoshop. (c) Zooming luminance by public domain fractal image compression routine [27], set for maximum image fidelity (chrominance components were zoomed by cubic spline, to avoid color artifacts). Both (c) and (d) are blurry, or have serious artifacts. (d) Markov network reconstruction using a training set of 10 images taken at the same picnic, none of this person. This is the best possible fair training set for this image. (e) Markov network reconstrution using a training set of *generic* photographs, none at this picnic or of this person, and fewer than 50% of people. The two Markov network results show good synthesis of hair and eye details, with few artifacts, but (d) looks slightly better (see brow furrow). Edges and textures seem sharp and plausible. (f) is the true full-resolution image.

6 Example 3: Motion Estimation

To show the breadth of the technique, we apply it to the problem of motion estimation. The *scene* to be estimated are the projected velocities of moving objects. The *image* data are two successive image frames. We made this application before we developed sampled inference, and we used a discrete representation for both learning and inference [7].

Luettgen et.al. [21] applied a related message-passing scheme in a multi-resolution quad-tree network to estimate motion, using gaussian probabilities. While the network did not contain loops, its structure generated artifacts along quad-tree boundaries, artificial statistical boundaries of the model.

To show the algorithm working on simple test cases, we generated a synthetic world of moving blobs, of random intensities and shapes. 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, and likewise for scene patches.

During training, we presented approximately 200,000 examples of irregularly shaped moving blobs of a contrast with the background randomized to one of 4 values. Using co-occurance histograms, we measured the relevant local statistics for Eqs. (2) and (3).

Figure 7 shows six iterations of the inference algorithm (Eqs. 2 and 3) 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.

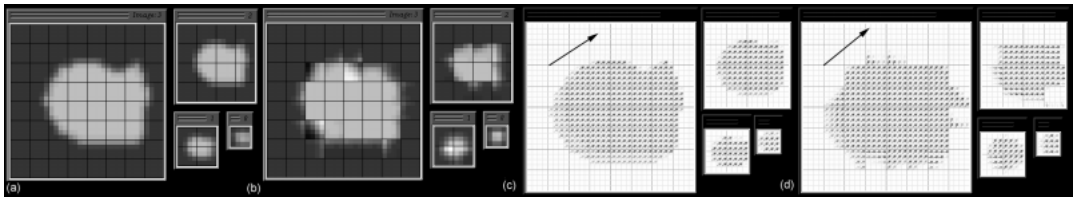


Figure 6: (a) First of two frames of image data (in gaussian pyramid), and (b) vector quantized. (c) The optical flow scene information, and (d) vector quantized. Large arrow added to show small vectors' orientation.

7 Discussion

The sampled inference propagation scheme has similarities with relaxation labelling [28]: both update a vector of beliefs through linking matrix connections to neighbors. However, the message update algorithm is quite different (the relaxation labelling algorithm is heuristically derived), as is the algorithm behavior: belief propagation converges quickly, while relaxation labelling convergence is typically slow [18].

The local probabilities our method learns, $P(y)$, $P(y|x)$, and $P(x|x_n)$, are powerful. They embody the particular vision algorithm. These probabilities, learned in the same way, but for different problems, can lead to very different propagation behavior, depending on the problems. For super-resolution, image contours often extend along the direction of the contour, Fig. 3. For motion estimation, the filling-in travels *perpendicularly* to the

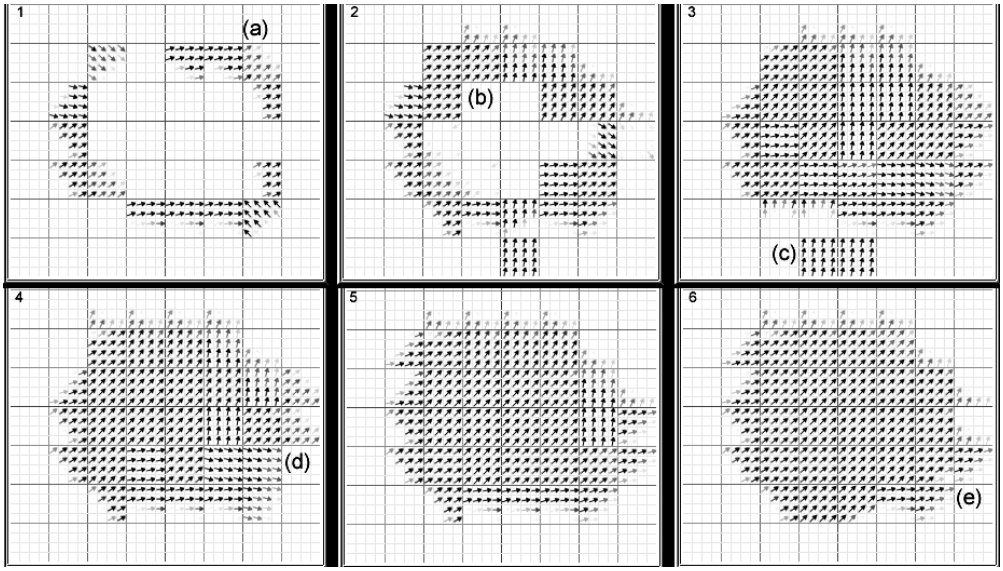


Figure 7: The most probable scene code for Fig. 6b at first 6 iterations of Bayesian belief propagation. (a) Note initial motion estimates occur only at edges. Due to the "aperture problem", initial estimates do not agree. (b) Filling-in of motion estimate occurs. Cues for figure/ground determination 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. 6.

direction of the object contour, Fig. 7. In both cases, the behavior was appropriate to the problem, and learned by the same general machinery.

8 Summary

We have applied a Bayesian learning method to the problem of scene estimation. We treat patches of images and scenes as nodes in a Markov network. From synthetic training examples, we learn the statistical relationship between local images and scenes, and between neighboring scenes.

A factorization approximation lets us learn the network parameters simply, and propagate local information by local update rules. Monte Carlo simulations show that, for random draws from gaussian processes, the estimated posterior accurately approximates the true posterior of the Markov network.

The local probabilities learned from the training data form the heart of the vision algorithm: the *local prior*, the *local likelihood*, and the joint probability for neighboring scenes. We identify the messages passed between nodes in our scheme as *region likelihoods*. These local probabilities can "learn" different propagation behavior, appropriate to each algorithm.

The technique of "sampled inference" computes the posterior probability only for a group of candidate scenes which all satisfy the image data. In practise, this speeds up the

algorithm without sacrificing quality.

The main approach of this paper–propagate local estimates to find a best, global solution– has a long tradition in computational vision [1, 28, 12, 26]. Here, we apply machine learning techniques to *learn* the potentially rich details governing the initial estimates and the propagation.

Applied to super-resolution, this method gives results that we believe are the state of the art. Applied to motion estimation, the same method resolves the aperture problem and appropriately fills-in motion over a figure. The technique may apply to related vision problems as well, such as line drawing interpretation, or distinguishing shading from reflectance.

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

References

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

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

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

[4] J. Besag. Spatial interaction and the statistical analysis of lattice systems (with discussion). *J. Royal Statist. Soc. B*, 36:192–326, 1974.

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

[6] 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.

[7] 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.

[8] B. J. Frey. *Graphical Models for Machine Learning and Digital Communication.* MIT Press, 1998.

[9] 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.

[10] 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.

[11] 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.

[12] B. K. P. Horn. *Robot vision.* MIT Press, 1986.

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

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

[15] B. Jahne. *Digital Image Processing.* Springer-Verlag, 1991.

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

[17] 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.

[18] J. Kittler and J. Illingworth. Relaxation labelling algorithms–a review. *Image and Vision Computing*, (11):206–216, 1985.

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

[20] M. S. Landy and J. A. Movshon, editors. *Computational Models of Visual Processing.* MIT Press, Cambridge, MA, 1991.

[21] 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.

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

[23] 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.

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

[25] 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.

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

[27] M. Polvere. Mars v. 1.0, a quadtree based fractal image coder/decoder, 1998. http://inls.ucsd.edu/y/Fractals/.

[28] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Trans. Systems, Man, Cybern.*, 6(6):420–433, 1976.

[29] 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.

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

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

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

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

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