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

Learning Motion Analysis

William T. Freeman, John A. Haddon† and Egon C. Pasztor*
MERL, Mitsubishi Electric Research Labs.
201 Broadway
Cambridge, MA 02139

TR-2000-32 September 2000

Abstract

We seek a learning-based algorithm that applies to various low-level vision problems. For a given problem, we want to find the scene interpretation that best explains image data. Specializing to the optical flow problem, we may want to infer the projected velocities (scene) which best explain two consecutive image frames (image).

We use synthetic data to generate examples of pairs images with their corresponding scene interpretation, the true projected velocities (optical flow). From these data, we learn candidate scene explanations for local image regions, and derive a compatibility function between neighboring scene regions. Given new image data, we propagate beliefs in a Markov network to infer the underlying optical flow. This yields an efficient method to infer low-level scene interpretations.

We first present the results of this method for a toy world of irregularly shaped blobs. Then we extend the technique to function on more realistic images, showing reasonable results.

1. First printing, TR2000-32, Sept., 2000

* John Haddon's affiliation:
Dept. Electrical Engineering and Computer Science
387 Soda Hall
University of California
Berkeley, CA 94720-1776

* Egon Pasztor's present address:
MIT Media Lab
20 Ames St.
Cambridge, MA 02139

Learning Motion Analysis

William T. Freeman* John Haddon† Egon C. Pasztor‡

September 15, 2000

Abstract

We seek a learning-based algorithm that applies to various low-level vision problems. For a given problem, we want to find the scene interpretation that best explains image data. Specializing to the optical flow problem, we may want to infer the projected velocities (scene) which best explain two consecutive image frames (image).

We use synthetic data to generate examples of pairs images with their corresponding scene interpretation, the true projected velocities (optical flow). From these data, we learn candidate scene explanations for local image regions, and derive a compatibility function between neighboring scene regions. Given new image data, we propagate beliefs in a Markov network to infer the underlying optical flow. This yields an efficient method to infer low-level scene interpretations.

We first present the results of this method for a toy world of irregularly shaped blobs. Then we extend the technique to function on more realistic images, showing reasonable results.

1 Introduction

The fundamental task of computer vision is the interpretation of images—what can we deduce about the world given one or more images of it? This understanding can be either at a high level—recognizing Albert Einstein, or identifying a chair—or at a low level—interpreting line drawings, estimating motion, or extrapolating resolution. The input is *image* data, which can be either a single image, or a collection of images over time. From that, for low-level vision problems, 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

*MERL, Mitsubishi Electric Research Labs., 201 Broadway, Cambridge, MA 02139

†Dept. Electrical Engineering and Computer Science, 387 Soda Hall, University of California, Berkeley, CA 94720-1776

‡present address: MIT Media Lab E15-385, 20 Ames St., Cambridge, MA 02139

solutions to these vision tasks would have many applications in searching, editing, and interpreting images. Machine solutions might even 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 (for example, [21, 26, 43, 34, 37, 28]). Unfortunately, solutions to these problems are often intractable; at best, they can be unreliable or slow. Often, the prior statistical models used are made up or tweaked by hand. Various image interpretation problems have defied generalization from initial simplified solutions [38, 2, 36].

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, 4]; Simoncelli and Schwartz [35] accounted for contrast normalization effects by redundancy reduction. Li and Atick [1] 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 [18, 10, 43, 34]. 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, it is necessary to collect statistics relating images with their underlying scene interpretations. For natural scenes, this data is difficult to collect, since it involves gathering ground truth data of the scene attributes to be estimated, which is often not readily available in real-world situations.

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 the underlying scene structure.

We believe that a visual system can correctly interpret a visual scene if it models (1) the probability that any local scene patch generated the local image patch, and (2) the probability that any local scene patch 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 have applied this method to a number of problems, including that of extrapolating high-resolution from low-resolution images [12]. (This is the same problem as that addressed in the chapter by Papageorgiou, Girosi, and Poggio, using a different approach). Here, we focus on the problem of optical

flow estimation—given a pair of images from a moving scene, infer the projected velocities of the objects moving in the image.

2 Markov network

We place the image and scene data in a Markov network [31, 15]. 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. 1. (In general, the neighbor relationship can be with regard to position, scale, orientation, etc. Here, we consider neighbors in position and scale). This forms a network of scene nodes, each of which may have an associated observation.

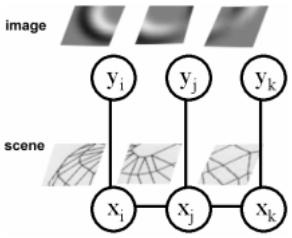


Figure 1: *Markov network for vision problems. Observations, y, have underlying scene explanations, x. Connections between nodes of the graphical model indicate statistical dependencies.*

Referring to Fig. 1, the Markov assumption asserts that complete knowledge of node x_j makes nodes x_i and x_k independent, *i.e.* $P(x_i, x_k|x_j) = P(x_i|x_j)P(x_k|x_j)$. We say x_i and x_k are conditionally independent given x_j.[1] 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 *infer* the corresponding scene.

3 Belief propagation derivation

3.1 Inference in networks without loops

For networks without loops, the Markov assumption leads to simple "message-passing" rules for computing the Maximum A Posteriori (MAP) and Minimum

[1] Note that, in general, the random variables x and y may be vector-valued; for notational convenience, we drop the vector symbol.

Mean Squared Error (MMSE) estimates [31, 40, 22]. To derive these rules, we first write the MAP and MMSE estimates for x_j at node j by marginalizing (MMSE) or taking the maximum (MAP) over the other variables in the posterior probability:

$$\hat{x}_j^{\mathrm{MMSE}} = \int_{x_j} x_j dx_j \int_{\text{all } x_i, i \neq j} P(x, y)\, dx \tag{1}$$

$$\hat{x}_j^{\mathrm{MAP}} = \arg\max_{x_j} \max_{\text{all } x_i, i \neq j} P(x, y) \tag{2}$$

y is the observed image data.

For a Markov random field, the joint probability over the scenes x and images y can be written as [5, 15, 14]:

$$P(x, y) = \prod_{\text{neighboring } i,j} \Psi(x_i, x_j) \prod_k \Phi(x_k, y_k), \tag{3}$$

where we have introduced pairwise compatibility functions, Ψ and Φ, described below. The factorized structure of Eq. (3) allows the marginalization and maximization operators of Eqs. (1) and (2) to pass through compatibility function factors with unrelated arguments. For the example network in Fig. 1, we have

$$\begin{aligned}
\hat{x}_1^{\mathrm{MAP}} &= \arg\max_{x_1} \max_{x_2} \max_{x_3} P(x_1, x_2, x_3, y_1, y_2, y_3) \\
&= \arg\max_{x_1} \max_{x_2} \max_{x_3} \\
&\quad \Phi(x_1, y_1)\Phi(x_2, y_2)\Phi(x_3, y_3)\Psi(x_1, x_2)\Psi(x_2, x_3) \\
&= \arg\max_{x_1} \Phi(x_1, y_1) \\
&\quad \max_{x_2} \Psi(x_1, x_2)\Phi(x_2, y_2) \\
&\quad \max_{x_3} \Phi(x_3, y_3)\Psi(x_2, x_3)
\end{aligned} \tag{4}$$

Each line of Eq. (4) is a local computation involving only one node and its neighbors. The analogous expressions for $\hat{x}_2^{\mathrm{MAP}}$ and $\hat{x}_3^{\mathrm{MAP}}$ use similar local calculations. Iterating those calculations lets each node j compute $\hat{x}_j^{\mathrm{MAP}}$ from the messages passed between nodes.

Assuming a general network without loops, Eqs. (1) and (2) can be computed by iterating the following steps [31, 40, 22]. The MAP estimate at node j is

$$\hat{x}_j^{\mathrm{MAP}} = \arg\max_{x_j} \Phi(x_j, y_j) \prod_k M_j^k \tag{5}$$

where k runs over all scene node neighbors of node j. M_j^k is the message sent from node k to node j. We calculate M_j^k from:

$$M_j^k = \max_{x_k} \Psi(x_k, x_j)\Phi(x_k, y_k) \prod_{l \neq j} \tilde{M}_k^l \tag{6}$$

where $\tilde{M}_k^l$ is M_k^l from the previous iteration. The initial $\tilde{M}_k^l$s are vectors of all 1's. After at most one iteration of Eq. (6) per scene node variable, Eq. (5) gives the desired optimal estimate, $\hat{x}_j^{\mathrm{MAP}}$. The MMSE estimate, Eq. (2), has analogous formulae, with the $\max_{x_k}$ of Eq. (6) replaced by $\int_{x_k}$, and $\arg\max_{x_j}$ of Eq. (5) replaced by $\int_{x_j} x_j$. For linear topologies, these propagation rules are equivalent to standard Bayesian inference methods, such as the Kalman filter and the forward-backward algorithm for Hidden Markov Models [31, 28, 39, 22, 13]. Weiss showed the advantage of belief propagation over regularization methods for several 1-d problems [39]; with the expectation of similar benefits, we apply belief propagation to our 2-d problems.

3.2 Networks with loops

In a Markov network like Figure 3, there are loops where messages can pass from node to node and return to the original node. This unfortunately means that the message passing algorithm presented above is not exactly correct.

Some theoretical justifications exist for using the message passing algorithm for networks with loops, however. In [41], it is shown that the MMSE rules give the correct means, but underestimate the variance for Gaussian distributions. For MAP rules, for arbitrary distributions, if the algorithm converges, it must converge to at least a local maximum of the posterior. [42] shows that the MMSE belief propagation equations are equivalent to the stationarity conditions for the Bethe approximation to the "free energy" of the network. This suggests that it might be acceptable to use message passing, ignoring the fact that there are loops present; and indeed, experiments give good results using this approach [27, 40].

Belief propagation algorithm	*Network topology*	
	no loops	*arbitrary topology*
MMSE rules	MMSE, correct posterior marginal probs.	For Gaussians, correct means, wrong covs.
MAP rules	MAP	Local max. of posterior, even for non-Gaussians.

Table 1: Summary of results from [41] regarding belief propagation after convergence.

4 Motion

The image analysis problem which we focus on here is that of estimating the optical flow between a pair of images. There are many existing techniques for computing optical flow from the differences between the two frames (e.g. [19, 6, 28]). We applied our method for learning low-level vision to this problem to show results on a well-studied problem. There have been many related applications

of Markov networks to vision, in the form of Markov random fields [15, 32, 14, 24, 6, 28, 26, 33]. For other learning or constraint propagation approaches in motion analysis, see [28, 29, 23].

We examined two classes of scene—first, a "blobs world", in which planar blobs translate in the plane; and then, more realistic images generated using a 3D rendering package. In both cases, the "image" consists of two concatenated image frames from sequential times. The "scene" is the corresponding projected velocities of the moving objects in the world.

4.1 Blobs world

In the blobs world, the training images were randomly generated moving, irregularly shaped blobs, as typified by Fig. 2 (a). The background was a random shade of gray, which yielded a range of contrasts with the blobs. 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 [8], to efficiently communicate across space. Each scene patch then additionally connects with the patches at neighboring resolution levels. Figure 2 shows the multiresolution representation (at one time frame) for images and scenes.[2] Figure 3 shows the corresponding Markov network for the scene nodes. (Each scene node connects to an observation node, not shown, at the corresponding scale and position). Luettgen et al [28] use a related multi-resolution Markov model to analyze motion, with exact inference. We use approximate inference, but use a more general model, allowing non-Gaussian statistics and Markov network connections between nodes at the same resolution level, which avoids discontinuities at quad-tree pyramid boundaries.

We applied the training method and propagation rules to motion estimation, using a vector code representation [16] 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 2 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.

4.1.1 Learning

For the blobs world problem, we used a different factorization of the posterior probability than Eq. (3), based on repeated applications of the rule $P(a, b) = P(a|b)P(b)$. This factorization, described in [11, 12], yields very similar belief

[2] 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.

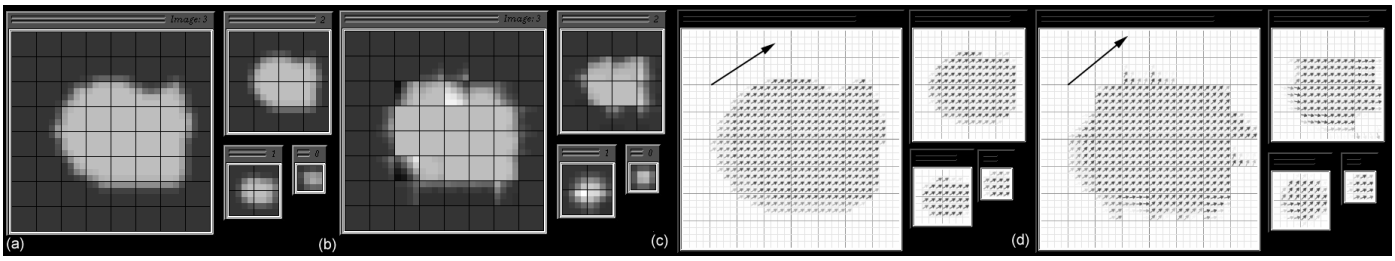


Figure 2: *(a) First of two frames of typical image data (in Gaussian pyramid). We observe the image at four different spatial scales. The blob is undergoing uniform translation in a random direction. (b) We represent the input frames by a collection of vector quantized pairs of image patches. (c) The true optical flow scene information that we hope to recover. (d) The system can only describe the estimated scene by a set of vector quantized patches of optical flow. Large arrow added to show small vectors' orientation. The goal is to estimate (c) from (a), viewed over two time steps. In our vector quantized representation, the best we can do is to estimate (d) from (b), viewed over two time frames. Figure 6 shows the velocity field estimated by our algorithm.*

propagation rules as Eq. (6), using compatibility functions learned from image-scene and scene-scene co-occurance statistics. The update and estimation rules are:

$$M_j^k = \max_{x_k} P(x_k|x_j)P(y_k|x_k) \prod_{l \neq j} \tilde{M}_k^l, \quad (7)$$

$$x_{j\,MAP} = \mathrm{argmax}_{x_j} P(x_j)P(y_j|x_j) \prod_k M_j^k. \quad (8)$$

where k runs over all scene node neighbors of node j. While the expression for the joint probability does not generalize to a network with loops, we nonetheless found good results for the motion estimation problem using these update rules.

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-occurence histograms, we measured the statistical relationships that embody the algorithm: $P(x)$, $P(y|x)$, and $P(x_n|x)$, for scene x_n neighboring scene x. Figures 4 and 5 show examples of these measurements.

4.1.2 Inference

Given a new image to analyze, we first break it into local patches of image data (image pairs). For each image pair patch, we collect a set of candidate scene interpretations from the training database. The belief propagation algorithm updates a set of probability assignments for each candidate scene interpretation,

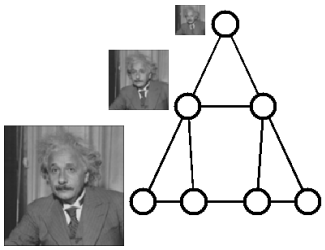


Figure 3: *Schematic illustration of multi-scale representation used for motion analysis problem. The image data is presented to the Markov network at multiple resolutions. Each scene node (representing the a patch of velocity data at some resolution and position) connects with its neighbors both in space and across scale. Each scene node also has connections (not shown) with a patch of image observations at the corresponding position and scale.*

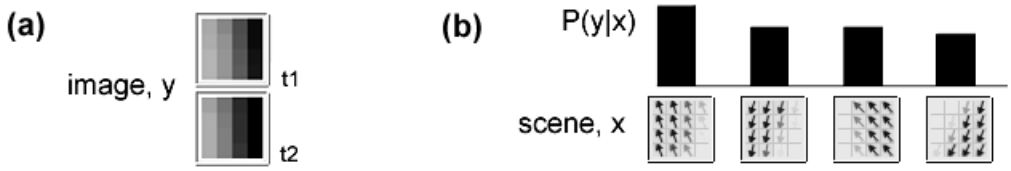


Figure 4: *The local likelihood information for motion problem, in a vector quantized representation. Conditional probabilities for the image data y (pair of image patches), given the scene x (projected velocities over a patch) are derived from co-occurence histograms. For a given image data sample, (a), the four scene elements with the highest likelihood of generating the image data are shown in (b).*

based initially only on the local image data at that patch, then taking into account more and more image data as the algorithm iterates.

Figure 6 shows six iterations of the inference algorithm 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[3] constraint propagation, and filling-in (see caption). The resulting inferred velocities are correct up to the accuracy of the vector quantized representation.

[3]The aperture problem refers to the fact that when you observe only a small piece of edge, you can only determine the component of its velocity across the edge, but not the component in the direction of the edge.

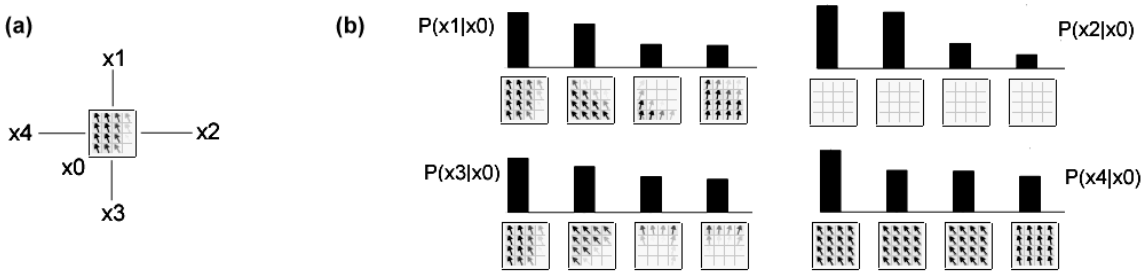


Figure 5: *Some scene conditional probabilities, for the motion problem. For the given scene element, (a), the four most likely nodes from four different neighboring scene elements are shown. For example, the top left plot of (b) shows the four motion patches most likely to appear at position* $x1$*, above the given patch of velocity data, and similarly for the three other positions relative to* $x0$*. Scene elements also connect to patches at other scales, not shown in this figure.*

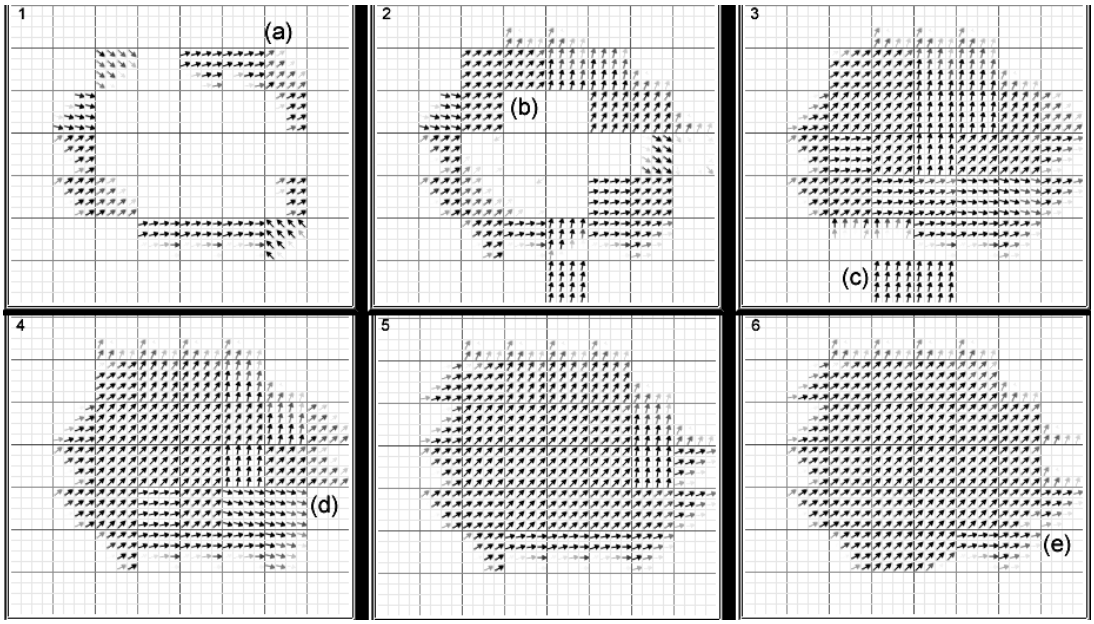


Figure 6: *The most probable scene code for Fig. 2b at first 6 iterations of Bayesian belief propagation. (a) Note initial motion estimates occur only at edges. Due to the "aperture problem", initial estimates do not agree. (b) Filling-in of motion estimate occurs. Cues for figure/ground determination 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. 2.*

4.2 Extension to more realistic images

The blob world discussed above is a very simple model of motion in real images. In order to apply the Markov network formalism to more realistic images, we generated a number of scenes using 3D Studio Max, release 2.5, a 3D modelling and rendering program. Distorted cubes in various orientations and positions were rendered from a camera directly overhead, and then from a slighly rotated and translated camera to generate a second frame. Examples of overhead images may be seen in figures 7(a) and 8(a).

For the blobs problem, we used a simple approach of storing (through our training observations) all possible local patches we would observe, and all possible scene observations for each one of those. In applying this approach to these more realistic images, we ran into a number of problems, related to the complexity of more realistic images. There are two major reasons why the blob world is easier to deal with than realistic images:

- *The background is stationary.* A stationary background means that if we observe a moving edge in an image patch, we know that one side of the edge is stationary, and we know the component of the velocity of the foreground in the direction perpendicular to the edge. Thus, the possible scene patches (optical flow) at an edge consist of a pair of one-dimensional spaces (the velocity parallel to the edge is the continuous dimension, and either side of the edge can be the foreground).

  For real images, the background may also be moving, and unless it has significant texture, it is difficult to tell which direction it is moving in. In the absence of other (global) information, the possible motions at a patch are a pair of three-dimensional spaces (we have two extra dimensions because the background can be moving at any velocity). It is more difficult both to learn and to do inference in this higher-dimensional space.

- *The blobs are untextured.* Because the blobs have no surface texture, there are relatively few possible image patches. Consider an image patch in the interior of a blob: every such patch will look identical, and we only need to describe the different possible scene motions.

  But real surfaces do have texture. To sample from all possible input data we need to sample from all possible patch textures combined with all possible motions that the patches could undergo.

4.2.1 Inference

If we were to use the same approach as we did with the blobs world, of choosing a set of image/scene pairs for each node, the number of candidates would have to be very large. There are many possible image patches in real images; we found that in order to ensure including the correct scene among the candidates, we needed to maintain a very long list of candidates.

Instead, we used the brightness constraint equation (BCE) common in optical flow work in computer vision [19], and generated a list of candidate scene explanations on the fly for each image patch of the test image. The BCE assumes that an object that moves from frame to frame will have the same brightness in each frame, possibly appearing at different points in the image.

Suppose that the pixel at point (x, y) at time t has moved to point $(x + dx, y + dy)$ at time $t + dt$. Then we have

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

If we take a Taylor series on the right to first order,

$$I(x, y, t) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt$$

In other words,

$$\nabla I \cdot v = -\frac{\partial I}{\partial t}$$

For real images, the equality will not be exact, and so it is standard practice to take a least squares estimate. Using this equation, we can evaluate the likelihood of different optical flow vectors at any pixel. (See the chapter by Weiss and Fleet for a complete discussion of likelihood functions for motion analysis). Of course, the aperture problem arises if we just consider a single pixel (we will only be able to determine one component of the velocity), but taking the sum over several pixels generally avoids this issue.

This raises the question of which pixels to sum over. Because of the difficulty mentioned above with edges in environments with nonstationary backgrounds, we assume that the optical flow is constant over a patch. This assumption is used in regularization approaches to the optical flow problem [19, 7] in which motion is smoothed across discontinuities. By making the optical flow constant over a patch, we are essentially ignoring the presence of object boundaries, but we gain significantly in that we no longer need to represent all possible optical flow vectors on either side of the boundary.

Due to the large numbers of samples that we would otherwise need to observe in training, instead of learning the statistics of neighbouring patches, we adopted a heuristic which gives us a measure of the correspondence between adjacent patches. Adjacent patches overlap by one pixel; then the compatibility of two adjacent patches can be estimated by the agreement that the two patches have on their common pixels. If the two patches have very different beliefs about the values of the pixels they share, those patches cannot be side by side.

Specifically, let k and j be two neighboring scene patches. Let d^l_{jk} be a vector of the pixels of the lth possible candidate for scene patch x_k which lie in the overlap region with patch j. Likewise, let d^m_{kj} be the values of the pixels (in correspondence with those of d^l_{jk}) of the mth candidate for patch x_j which overlap patch k. We say that scene candidates x^l_k (candidate l at node k) and x^m_j

are compatible with each other if the pixels in their regions of overlap agree. We assume that the scene training samples differ from the "ideal" training samples by Gaussian noise of covariance σ_s. Those covariance values are parameters of the algorithm. We then define the compatibility matrix between scene nodes k and j as

$$\Psi(x_k^l, x_j^m) = \exp^{-|d_{jk}^l - d_{kj}^m|^2/2\sigma_s^2} \tag{9}$$

The rows and columns of the compatibility matrix $\Psi(x_k^l, x_j^m)$ are indexed by l and m, the scene candidates at each node, at nodes j and k. We evaluate $\Phi(x_k, y_k)$ in an analogous fashion [12].

In using this heuristic, we are no longer *learning* the statistics of adjacent patches. Since for these images, we constrain our velocity samples to be uniform over each patch, the method of evaluating $\Psi(x_k, x_j)$ is equivalent to a smoothness constraint on the reconstructed motion. This is similar to other motion smoothness constraints (e.g. [19]), although enforced by Bayesian belief propagation.

Initially, we chose our optical flow vectors by sampling from the distribution of velocities consistent with the BCE. But we found that at some nodes in the network, there would be a number of adjacent patches which all had very similar samples, not because of similar structure, but simply because of the randomness of sampling. Because these samples agreed with their neighbours, and other samples may not have had such good agreement, these nodes would end up with a strong belief, based on no evidence, which was often contradictory to the correct answer.

Thus, we chose to select candidates by choosing optical flow vectors on a uniform grid, and to keep all of the candidates for each node. This solved any problems with random agreement with neighbours, since each neighbour also had the complete candidate list, so no candidate had more agreement than any other.

4.3 Results

In figures 7 and 8, we show two examples of the optical flow algorithm working on our realistic images. Observe that initially, the optical flow vectors are mostly zero (we display the average vector for each patch, weighted by its belief) except at edges, where we have a good estimate of the perpendicular velocity. After one iteration, the information at the edges begins to be propagated, and after a few iterations, the solution is close to the correct answer.

In some cases, the final solution deviates from the correct answer, especially at the image boundary. This is partially due to the fact that in this model, there is no way for a node at the boundary to have a greater magnitude than its neighbours. If the images are from a camera rotating about its optical axis, we observe the centre of the vector field rotating, and we expect the boundary of the image to be rotating more. Similarly, if the camera is moving towards the

scene, we see expansion in the middle of the image, and we expect that effect to be exaggerated towards the boundary, even in the absence of any confirming evidence. But with a Markov network, if there is no information present at the boundary of an image, the nodes there will likely share the belief of the nearest nodes with good information. Since those nodes are closer to the centre of the image, the optical flow vectors will tend to have smaller magnitude, and the optical flow on the boundary will be underestimated.

Figure 9 shows a frame from the Yosemite image sequence [17], and the resulting optical flow from our algorithm. The clouds are correctly identified as moving to the right, and there is a focus of expansion somewhat to the right of the center of the image as we fly towards that point. A few scene patches in the clouds still have some error; the lack of texture in that part of the image makes accurate optical flow estimation there exceedingly difficult.

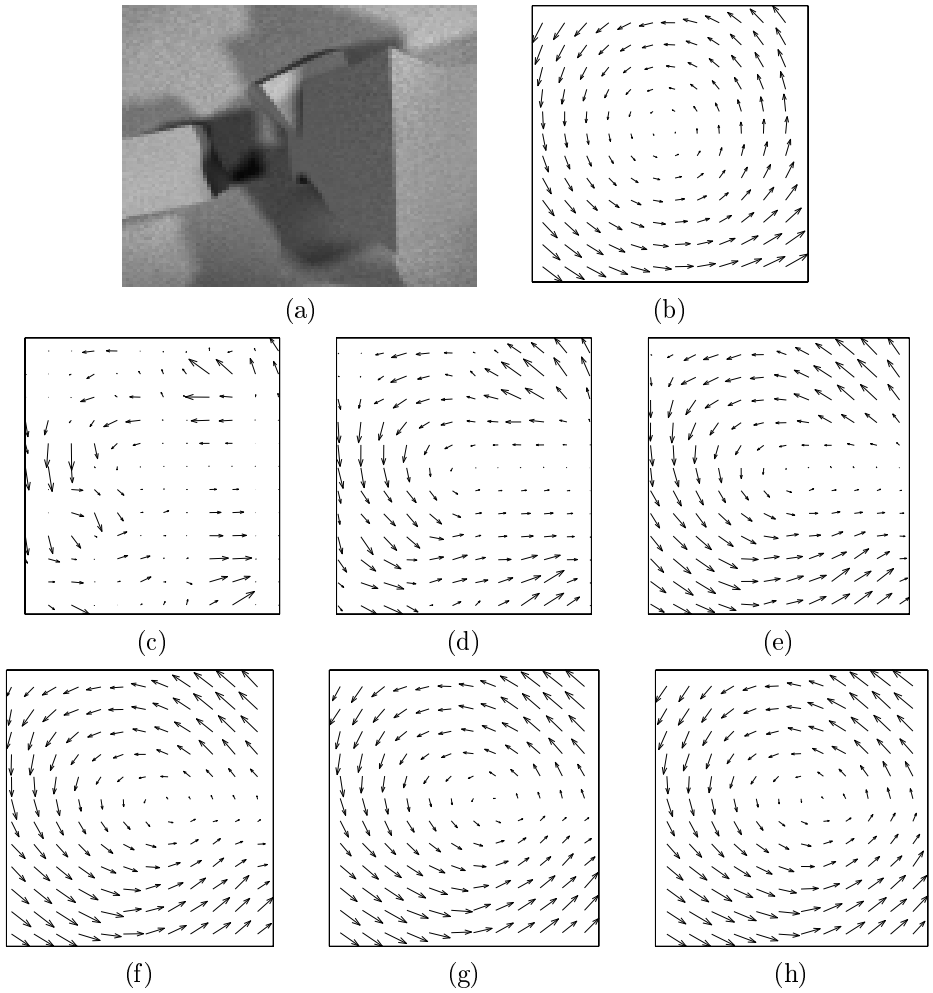


Figure 7: *Example of optical flow solution using belief propagation. (a) one frame of the two-frame input image pair. (b) The true rotational (synthetically generated) optical flow for this image. (c) The initial estimate of optical flow, with no information from a scene node's neighbors. (d)–(h) Reconstructed optical flow after 1–5 iterations of belief propagation. The initial iterations show little motion in regions where there is little image contrast perpendicular to the direction of motion, such as the right hand side, and upper left hand corner. The spatial continuity of the motion, enforced by the belief propagation, fills in the motion smoothly.*

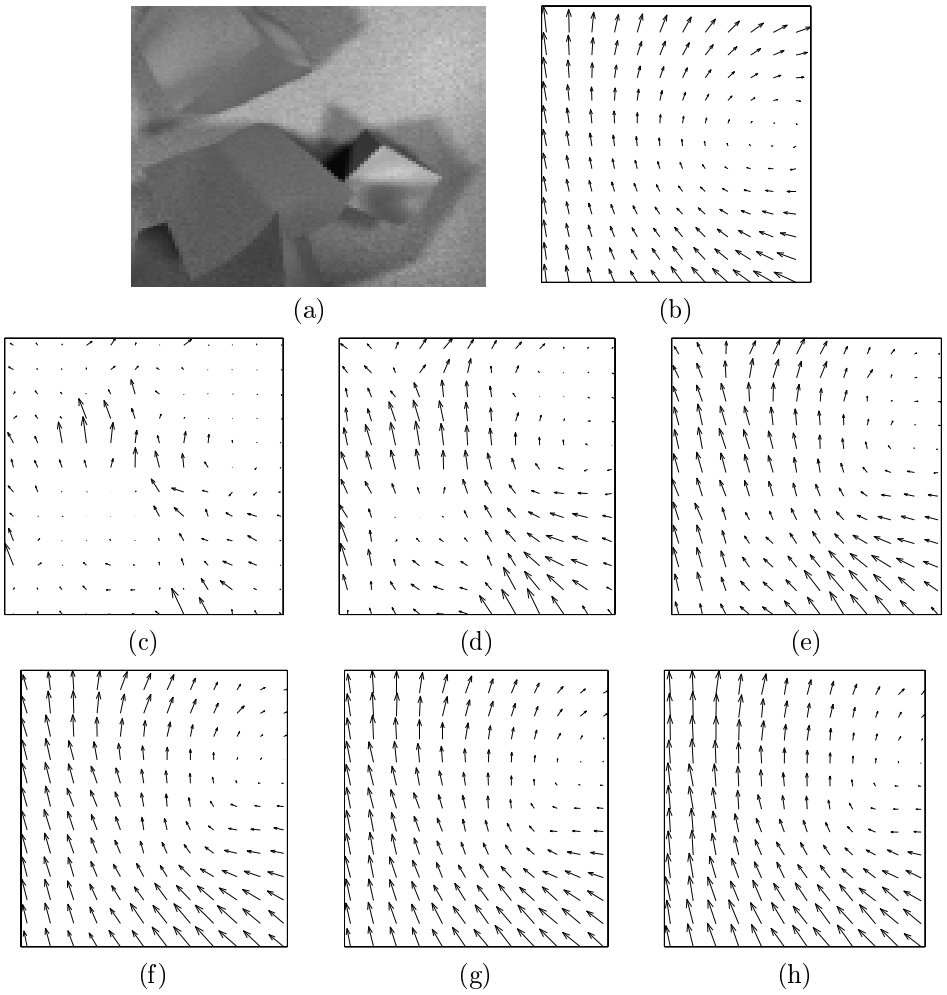
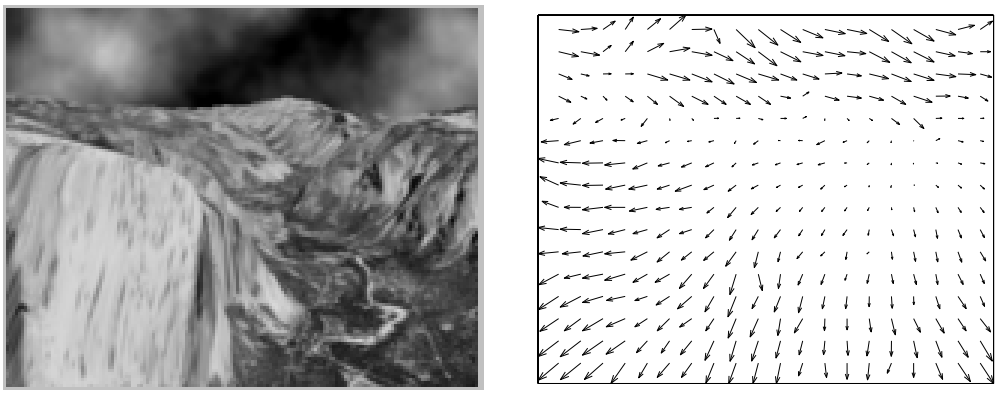


Figure 8: *Another example of optical flow solution using belief propagation. See caption to figure 7.*



(a) (b)

Figure 9: *Results of motion estimation on Yosemite sequence. (a) Single frame from sequence. (b) estimated optical flow, based on that frame and the subsequent one. The clouds are flowing to the right, and the land underneath is expanding as we fly into the valley.*

5 Summary

We have presented an algorithm for motion analysis which uses belief propagation in a Markov network to estimate the optical flow between a pair of images by communicating local information across space. The algorithm has been applied to a simple 2D blob world, as well as more realistic images generated in a 3D rendering package. The local probabilistic descriptions have power and flexibility. For the motion problem, they lead to filling-in motion estimates in a direction perpendicular to object contours and resolution of the aperture problem, Fig. 6.

The algorithm uses simple notions, which can apply to other vision problems: form hypotheses from local data, and propagate these across space. Of course, these are well-known principles in computational vision [38, 9, 2, 3]; what we propose is simple machinery to implement these principles. We have shown elsewhere the usefulness of this method for the problems of super-resolution and discriminating shading from paint [12]. From a synthetically generated world, we learn a training set of local examples of image (image pairs) and scene (projected image velocities) data. Using the machinery of Bayesian belief propagation applied to Markov networks with loops, we quickly find approximate solutions for the scene explanation which maximizes the posterior probability.

Acknowledgements We thank E. Adelson, O. Carmichael, J. Tenenbaum, and Y. Weiss for helpful discussions.

References

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

[2] 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.

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

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

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

[6] 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.

[7] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1), 1996.

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

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

[10] 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.

[11] 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. See also http://www.merl.com/reports/TR99-05/.

[12] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *Intl. J. Computer Vision*, 2000. In press. See also http://www.merl.com/reports/TR2000-05/.

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

[14] 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.

[15] 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.

[16] 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.

[17] D. Heeger. Yosemite fly-by sequence. ftp://csd.uwo.ca/pub/vision/TESTDATA/.

[18] 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.

[19] B. K. P. Horn and B. G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

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

[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] F. R. Kschischang and B. J. Frey. Iterative decoding of compound codes by probability propagation in graphical models. *IEEE Journal on Selected Areas in Communication*, 16(2):219–230, 1998.

[28] 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.

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

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

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

[35] 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.

[36] 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.

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

[38] 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.

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

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

[41] Y. Weiss and W. T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. Technical Report UCB.CSD-99-1046, Berkeley Computer Science Dept., 1999. www.cs.berkeley.edu/~ yweiss/gaussTR.ps.gz.

[42] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. Technical Report 2000–26, MERL, Mitsubishi Electric Research Labs., www.merl.com, 2000.

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