PAMPAS: Real-Valued Graphical Models for Computer Vision

M. Isard
Microsoft Research
Mountain View, CA 94043

Abstract

Probabilistic models have been adopted for many computer vision applications, however inference in high-dimensional spaces remains problematic. As the state-space of a model grows, the dependencies between the dimensions lead to an exponential growth in computation when performing inference. Many common computer vision problems naturally map onto the graphical model framework; the representation is a graph where each node contains a portion of the state-space and there is an edge between two nodes only if they are not independent conditional on the other nodes in the graph. When this graph is sparsely connected, belief propagation algorithms can turn an exponential inference computation into one which is linear in the size of the graph. However belief propagation is only applicable when the variables in the nodes are discrete-valued or jointly represented by a single multivariate Gaussian distribution, and this rules out many computer vision applications.

This paper combines belief propagation with ideas from particle filtering; the resulting algorithm performs inference on graphs containing both cycles and continuous-valued latent variables with general conditional probability distributions. Such graphical models have wide applicability in the computer vision domain and we test the algorithm on example problems of low-level edge linking and locating jointed structures in clutter.

1. Introduction

Probabilistic techniques for modelling uncertainty have found widespread success in computer vision. Their application ranges from pixel-level image generation models [13, 7] to tracking in high-level parametric spaces [11, 22] however as the size of the state-space grows the "curse of dimensionality" can cause an exponential growth in the computations necessary to perform inference. Many probabilistic vision models naturally decompose into a *graphical model* structure [14]; basic components become nodes in a graph where each node is conditionally independent of all but its near neighbours. When the nodes are image patches the neighbours may be spatially close in an image, adjacent levels in a multi-scale representation or nearby time instants in an image sequence. Alternatively many complex objects can be decomposed into graphs where the nodes are subparts of the object and an edge indicates that two subparts are physically connected. The advantage of this representation is that some formerly exponential inference computations become linear in the size of the graph.

Exact inference on graphical models is possible only in restricted circumstances [14]; the (directed) graph must be acyclic and the joint distribution over those latent variables which are not discrete must be a single multivariate Gaussian. When a model violates these restrictions, Gibbs sampling can be used to draw approximate samples from the joint distribution [9] but for most computer vision applications this technique remains intractable. Approximate inference methods can be used for Conditional Linear Gaussian models [16] including time series [3]. Recently two methods have been widely studied in the computer vision literature to perform approximate inference on more general classes of graph; loopy belief propagation [26] (LBP) can be applied to graphs with cycles, though it still only applies to discrete or jointly Gaussian random variables; and particle filtering [5] allows the use of very general distributions over continuous-valued random variables but applies only to graphs with a simple linear chain structure.

The restriction to Gaussian latent variables is particularly onerous in computer vision applications because clutter in image generation models invariably leads to highly non-Gaussian, multi-modal posterior distributions. This effect largely accounts for the popularity of particle filtering in the computer vision tracking community. In this paper we describe the PAMPAS algorithm (PArticle Message PASsing) which combines LBP with ideas from particle filtering. It has wide applicability in the computer vision domain, and increases the range of complex continuous-valued models which admit tractable probabilistic inference. We investigate the behaviour of the algorithm on test problems including illusory-contour finding and locating a jointed structure. Additionally we argue that an algorithm which performs

tractable inference over continuous-valued graphical models is a promising tool for research into unified computer vision models implementing a singe probabilistic framework all the way from the pixel level to a semantic description.

2. Belief propagation

Belief propagation (BP) has been used for many years to perform inference in Bayesian networks [14] and has recently been applied to graphs with cycles under the name of loopy belief propagation [26]. The method consists in passing *messages* between the nodes of the graph. When all the latent variables are discrete-valued the messages can be encoded as matrices, and when the continuous-valued variables are jointly Gaussian the messages can be summarised using a mean and covariance matrix.

When using models with more complex continuous-valued marginal distributions the data representation for messages becomes problematic since as they are propagated their complexity grows exponentially [18]. Techniques exist for pruning the representations when the potentials and likelihoods are exponentional distributions [16, 19, 2] and while these can be used to perform inference over Gaussian mixture models like those in this paper, they are not well suited to the complex likelihoods which arise in computer vision applications. When the graph is a chain a particle filter [5] can be used which represents the marginals with a non-parametric form, the *particle set*. Particle filtering has been widely used in computer vision largely because it performs well with common image likelihood models.

This section begins with a brief description of standard BP and goes on to describe the PAMPAS algorithm which modifies BP to use particle sets as messages and thus permits approximate inference over general continuous-valued graphical models. Sudderth et al [23, 24] have independently developed an almost identical algorithm and a comparison is presented in section 2.5.

2.1. Discrete Belief Propagation

Belief propagation can in general be stated using a pairwise MRF formulation [26]. Consider a set of latent-variable nodes $\mathcal{X} = \{X_1, \ldots, X_M\}$ paired with a set of observed nodes $\mathcal{Y} = \{Y_1, \ldots, Y_M\}$. The conditional independence structure of the problem is expressed in the neighbourhood set $\mathcal{P}$. A pair of indices $(i, j) \in \mathcal{P}$ if the hidden node X_j is not conditionally independent of X_i given the other nodes in the graph, and in this case we say X_i is a parent of X_j and by slight abuse of notation $i \in \mathcal{P}(j)$.

For tutorial simplicity, when the X_i are discrete-valued random variables we assume each takes on some value $x_i \in \{1, \ldots, L\} \equiv \mathcal{L}$. Given a fixed assignment to $\mathcal{Y}$ the observed and hidden nodes are related by observation functions $\phi_i(x_i, y_i) \equiv \phi_i(x_i) : \mathcal{L} \to \mathbb{R}$. Adjacent hidden nodes are related by the correlation functions $\psi_{ij}(x_j, x_i) : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$. The joint probability over $\mathcal{X}$ is given by

$$P(\mathcal{X}) = \frac{1}{Z}\left(\prod_{(i,j)\in\mathcal{P}} \psi_{ij}(x_j, x_i)\right)\left(\prod_i \phi_i(x_i)\right). \quad (1)$$

The belief propagation algorithm introduces variables such as $m_{ij}(x_j)$ which can intuitively be understood as a *message* from hidden node i to hidden node j about what state node j should be in. Messages are computed iteratively using the update algorithm

$$m_{ij}(x_j) \leftarrow \sum_{x_i\in\mathcal{L}} \left(\phi_i(x_i)\psi_{ij}(x_j, x_i) \prod_{k\in\mathcal{P}(i)\setminus j} m_{ki}(x_i)\right) \quad (2)$$

and the marginal distribution of X_i (the node's *belief*) is given by

$$b_i(x_i) \propto \phi_i(x_i) \prod_{j\in\mathcal{P}(i)} m_{ji}(x_i). \quad (3)$$

2.2. Belief propagation in particle networks

When a problem calls for continuous-valued random variables $\mathbf{X}_i \in \mathbb{R}^D$ then $\phi_i(\mathbf{x}_i) : \mathbb{R}^D \to \mathbb{R}$ and $\psi_{ij}(\mathbf{x}_j, \mathbf{x}_i) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ and (1) now defines a probability density. In the continuous analogues of (2) and (3) now $m_{ij}(\cdot)$ and $b_i(\cdot)$ are functions rather than L-element arrays; in fact we will assume that they are normalisable functions which can be interpreted as probability density functions. Now (2) becomes

$$m_{ij}(\mathbf{x}_j) \leftarrow \int_{\mathbb{R}^D} \psi_{ij}(\mathbf{x}_j, \mathbf{x}_i)\phi_i(\mathbf{x}_i) \prod_{k\in\mathcal{P}(i)\setminus j} m_{ki}(\mathbf{x}_i)\mathrm{d}\mathbf{x}_i. \quad (4)$$

If the structure of the network is such that the marginal distribution at each node is Gaussian, this integral can be performed exactly [19]. As noted above, if any of the m_{ij} or ϕ_i violate this conditional linear Gaussian structure exact inference is intractable and the m_{ij} must be replaced by an approximation; the PAMPAS algorithm uses particle sets.

At the heart of any algorithm for belief propagation with particle sets is a Monte-Carlo approximation to the integral in (4). We set out here a general algorithmic framework for performing this approximation and in section 2.4 describe the PAMPAS algorithm which is a specific implementation suitable for common computer vision models.

The function

$$p_{ij}^M(\mathbf{x}_i) \equiv \frac{1}{Z_{ij}}\phi_i(\mathbf{x}_i) \prod_{k\in\mathcal{P}(i)\setminus j} m_{ki}(\mathbf{x}_i) \quad (5)$$

is denoted the *foundation* of message $m_{ij}(\cdot)$ where Z_{ij} is a constant of proportionality which turns $p_{ij}^M(\cdot)$ into a probability density. A Monte-Carlo integration of (4) yields $\tilde{m}_{ij}$, an approximation of m_{ij}, by drawing N samples from the foundation $\mathbf{s}_{ij}^n \sim p_{ij}^M(\cdot)$ and setting

$$\tilde{m}_{ij}(\mathbf{x}_j) = \frac{1}{N}\sum_{n=1}^{N} \psi_{ij}(\mathbf{x}_j, \mathbf{s}_{ij}^n). \tag{6}$$

In general it is possible to pick an *importance function* $g_{ij}(\mathbf{x}_i)$ from which to generate samples $\mathbf{s}_{ij}^n \sim g_{ij}(\cdot)$ with (unnormalised) importance weights $\pi_{ij}^n \propto p_{ij}^M(\mathbf{s}_{ij}^n)/g(\mathbf{s}_{ij}^n)$ in which case the $\tilde{m}_{ij}$ is a weighted mixture

$$\tilde{m}_{ij}(\mathbf{x}_j) = \frac{1}{\sum_{k=1}^{N}\pi_{ij}^k}\sum_{n=1}^{N} \pi_{ij}^n \psi_{ij}(\mathbf{x}_j, \mathbf{s}_{ij}^n). \tag{7}$$

When the marginal distribution over $\mathbf{x}_i$ is required, samples $\mathbf{s}_i^n$ can be drawn from the estimated belief distribution

$$\tilde{b}_i(\mathbf{x}_i) = \frac{1}{Z_i}\phi_i(\mathbf{x}_i)\prod_{j\in\mathcal{P}(i)} \tilde{m}_{ji}(\mathbf{x}_i) \tag{8}$$

either directly or by importance sampling as above and these samples can be used to compute expectations over the belief, for example its mean and higher moments. By fixing a set of samples $\mathbf{s}_i^n$ at each node a Gibbs sampler can be used to generate samples from the joint distribution over the graph and this is described in section 2.4.

2.3. Choice of importance function

Some limitations of the approach in section 2.2 can be seen by considering a simple example. Suppose that we wish to perform inference on a three-node network $(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3)$ which has the chain structure:

$$\mathcal{P}(1) = (2), \quad \mathcal{P}(2) = (1,3), \quad \mathcal{P}(3) = (2).$$

Two distinct inference chains are being sampled: $\mathbf{X}_1 \rightarrow \mathbf{X}_2 \rightarrow \mathbf{X}_3$ and $\mathbf{X}_3 \rightarrow \mathbf{X}_2 \rightarrow \mathbf{X}_1$ but the information is not combined in any of the messages; there is no "mixing" between the forward and backward directions. This situation also arises whenever there is a layered structure to the graph with a set of low-level nodes passing information up to higher levels and high-level information propagating back down. Of course the belief does correctly incorporate information from both directions, but the efficiency of a Monte Carlo algorithm is strongly dependent on the sample positions so if possible we would like to use all the available information when choosing these positions.

One solution is to use importance sampling for some of the particles [16] and a natural choice of function is the current belief estimate $g(\mathbf{x}_i) = \tilde{b}_i(\mathbf{x}_i)$ in which case the importance weights are $\pi_{ij}^n = 1/\tilde{m}_{ji}(\mathbf{s}_{ij}^n)$. In the context of the simple chain given above this algorithm amounts to *smoothing* the distribution. Smoothing has been proposed for particle filters [12] but previous algorithms merely updated the particle weights in a backward pass rather than generating new particle positions localised in regions of high belief.

2.4. The PAMPAS algorithm

A property of many computer vision models is that the potentials $\psi_{ij}(\mathbf{x}_j, \mathbf{x}_i)$ can be written as a mixture of Gaussians and that the likelihoods $\phi_i(\mathbf{x}_i)$ are complex and difficult to sample from but can be evaluated up to a multiplicative constant. We now describe the PAMPAS algorithm which is specialised to perform belief propagation for this type of model; section 2.5 includes a brief sketch of a more general algorithm [24] for which these Gaussian restrictions do not apply.

For notational simplicity we describe the case that each potential is a single Gaussian plus a Gaussian outlier process though the extension to mixtures of Gaussians is straightforward. Setting λ^o to be the fixed outlier probability and μ_{ij} and Λ_{ij} the parameters of the outlier process,

$$\begin{aligned}\psi_{ij}(\mathbf{x}_j, \mathbf{x}_i) = (1-\lambda^o)\mathcal{N}(\mathbf{x}_j; f_{ij}(\mathbf{x}_i), G_{ij}(\mathbf{x}_i)) + \\ \lambda^o\mathcal{N}(\mathbf{x}_j; \mu_{ij}, \Lambda_{ij})\end{aligned} \tag{9}$$

where $f_{ij}(\cdot)$ and $G_{ij}(\cdot)$ are deterministic functions, potentially non-linear, respectively computing the mean and covariance matrix of the conditional Gaussian distribution. Each message approximation $\tilde{m}_{ij}$ is now a mixture of N Gaussians:

$$\begin{aligned}\tilde{m}_{ij}(\mathbf{x}_j) = &\frac{(1-\lambda^o)}{\sum_{k=1}^{N'}\pi_{ij}^k}\sum_{n=1}^{N'} \pi_{ij}^n \mathcal{N}(\mathbf{x}_j; f_{ij}(\mathbf{s}_{ij}^n), G_{ij}(\mathbf{s}_{ij}^n)) \\ &+ \lambda^o\mathcal{N}(\mathbf{x}_j; \mu_{ij}, \Lambda_{ij})\end{aligned} \tag{10}$$

where $N' = N - 1$ is the number of samples drawn during the Monte-Carlo integration. The message is summarised by N triplets:

$$\tilde{m}_{ij} = \{(\pi_{ij}^n, \mathbf{s}_{ij}^n, \Lambda_{ij}^n) : 1 \le n \le N\}. \tag{11}$$

where π_{ij}^N, $\mathbf{s}_{ij}^N$ and Λ_{ij}^N correspond to the outlier process and are fixed throughout the algorithm. A potential which encodes a mixture of J Gaussians with K Gaussian outliers will lead to $N' = (N-K)/J$ which is only practical for fairly small values of J.

As indicated in section 2.3 it makes sense to importance-sample some fraction of the message particles from the approximate belief distribution. In fact, because ϕ_i is assumed to be hard to sample from but easy to evaluate, we will use importance sampling for all the particles but with two

distinct importance distributions and corresponding weight functions. A fraction $(1-\nu)N'$ of the particles, by slight abuse of terminology, are denoted direct samples:

$$\begin{aligned} \mathbf{s}_{ij}^n &\sim \frac{1}{\tilde{Z}_{ij}} \prod_{k \in \mathcal{P}(i) \setminus j} \tilde{m}_{ki}(\mathbf{x}_i) \\ \pi_{ij}^n &= \phi_i(\mathbf{s}_{ij}^n) \end{aligned} \tag{12}$$

and the remaining $\nu N'$ particles we will refer to as importance samples:

$$\begin{aligned} \mathbf{s}_{ij}^n &\sim \frac{1}{\tilde{Z}_{ij}} \prod_{k \in \mathcal{P}(i)} \tilde{m}_{ki}(\mathbf{x}_i) \\ \pi_{ij}^n &= \phi_i(\mathbf{s}_{ij}^n) / \tilde{m}_{ji}(\mathbf{s}_{ij}^n). \end{aligned} \tag{13}$$

Note that if the network is a forward chain where $\mathcal{P}(i) \equiv i-1$ and $\nu = 0$ the algorithm reduces exactly to a standard form of particle filtering [11].

Both (12) and (13) require sampling from a foundation F which is the product of D mixtures indexed by a label vector $L = (l_1, \ldots, l_D)$:

$$F(\cdot) = \frac{1}{Z} \sum_L \eta_L \mathcal{N}(\cdot; \mu_L, \Lambda_L) \tag{14}$$

where from the standard Gaussian product formula

$$\Lambda_L^{-1} = \sum_{d=1}^{D} (\Lambda_d^{l_d})^{-1} \qquad \Lambda_L^{-1} \mu_L = \sum_{d=1}^{D} (\Lambda_d^{l_d})^{-1} \mu_d^{l_d} \tag{15}$$

$$\eta_L = \frac{\prod_{d=1}^{D} \pi_d^{l_d} \mathcal{N}(\cdot; \mu_d^{l_d}, \Lambda_d^{l_d})}{\mathcal{N}(\cdot; \mu_L, \Lambda_L)}. \tag{16}$$

This product F is a Gaussian mixture with N^D componenents so direct sampling is effectively infeasible for $D > 3$ for reasonable N. Methods for overcoming this difficulty are discussed in sections 2.5 and 4. The PAMPAS message update algorithm is given in figure 1.

When N belief samples $\mathbf{s}_i^n$ have been drawn for each node in $\mathcal{X}$ they can be used to sample from the joint distribution over the graph. This can be done by Gibbs sampling from the discrete probability distribution over labels $L' = (l_1, \ldots, l_M)$ where L' indexes a sample $\mathbf{s}_m^{l_m}$ at each of the M nodes. The Gibbs sampler updates each component of this label vector in turn where the marginal probability of choosing label l_d when the other labels are fixed is

$$P(l_d = n) = \phi_d(\mathbf{s}_i^n) \prod_{j \in \mathcal{P}(i)} \psi_{ji}(\mathbf{s}_i^n, \mathbf{s}_j^{l_j}). \tag{17}$$

2.5. Comparison with the NBP algorithm

Sudderth et al [23, 24] have independently developed an almost identical algorithm for performing belief propagation with the aid of particle sets, which they call Non-Parametric Belief Propagation, or NBP. In order to deal with the exponential blowup in the message product as D increases, they introduce a Gibbs sampler which is sketched below. We have used this Gibbs sampler for the central node in the jointed object model of section 3.3 which has 4 incoming messages.

1. Draw samples from the incoming message product.
   (a) For $1 \le n \le (1-\nu)(N-1)$:
      i. Draw $\tilde{\mathbf{s}}_{ij}^n \sim \prod_{k \in \mathcal{P}(i) \setminus j} \tilde{m}_{ki}(\cdot)$.
      ii. Set $\tilde{\pi}_{ij}^n = 1/(N-1)$.
   (b) For $(1-\nu)(N-1) < n \le N-1$:
      i. Draw $\tilde{\mathbf{s}}_{ij}^n \sim \prod_{k \in \mathcal{P}(i)} \tilde{m}_{ki}(\cdot)$.
      ii. Set $\gamma_{ij}^n = 1/\tilde{m}_{ji}(\tilde{\mathbf{s}}_{ij}^n)$.
   (c) For $(1-\nu)(N-1) < n \le N-1$:
      i. Set $\tilde{\pi}_{ij}^n = \nu \gamma_{ij}^n / \sum_{k=1+(1-\nu)(N-1)}^{N-1} \gamma_{ij}^k$.
2. Apply importance correction from likelihood. For $1 \le n \le N-1$:
   (a) Set $\bar{\pi}_{ij}^n = \tilde{\pi}_{ij}^n \phi_i(\tilde{\mathbf{s}}_{ij}^n)$
3. Store normalised weights and mixture components. For $1 \le n \le N-1$:
   (a) Set $\pi_{ij}^n = (1 - \pi_{ij}^N) \bar{\pi}_{ij}^n / \sum_{k=1}^{N-1} \bar{\pi}_{ij}^k$
   (b) Set $\mathbf{s}_{ij}^n = f_{ij}(\tilde{\mathbf{s}}_{ij}^n)$.
   (c) Set $\Lambda_{ij}^n = G_{ij}(\tilde{\mathbf{s}}_{ij}^n)$.

Figure 1. The message update algorithm.

The main remaining difference between the algorithms is the way in which the message foundations are interpreted. In [24] N samples $\mathbf{r}_{ij}^n$ are generated from $p_{ij}^M(\mathbf{x}_i)$ and then for each $\mathbf{r}_{ij}^n$ a single sample $\mathbf{s}_{ij}^n$ is generated by sampling from the potential $\psi_{ij}(\cdot, \mathbf{r}_{ij}^n)$. The message mixture $\tilde{m}_{ij}(\mathbf{x}_j)$ is then formed by placing identical diagonal-covariance Gaussian kernels about the $\mathbf{s}_{ij}^n$. This additional smoothing kernel leads to variance estimates which are biased upwards of their true values. This step is unnecessary when the potentials are small mixtures of Gaussians so we omit it; thus we achieve unbiased kernel estimates as well as generating less noisy samples from the message distribution, and in addition we do not have the problem of selecting a kernel width. The advantage of the technique in [24] is that it allows more general potentials which are not mixtures of Gaussians. Note that if the potentials are complex (for example a mixture of $J = 100$ Gaussians) neither approach will adequately represent the product unless N is very large. The two algorithms also differ in the application of ϕ_i and NBP is specialised to models where ϕ_i is slowly

varying with respect to the kernel bandwidth.

Sudderth et al [23, 24] propose a Gibbs sampler which performs approximate sampling from the message product F in $O(KDN)$ operations per sample, where K is the number of iterations of the sampler. A sample is drawn from F by first choosing a label vector L using a Gibbs sampler and then generating a random sample from the Gaussian F_L. The Gibbs sampler works by sampling l_d with all the other labels in L held fixed for each d in turn. With all of L but l_d fixed the marginal product-component weights are given by

$$\eta_d^n \propto \frac{\pi_d^n \mathcal{N}(\cdot; \mu_d^n; \Lambda_d^n)}{\mathcal{N}(\cdot; \mu_{L^n}, \Lambda_{L^n})} \tag{18}$$

where $L^n = (l_1, \ldots, l_{d-1}, n, l_{d+1}, \ldots, l_D)$.

We use this sampler with $K = 100$ in section 3.3 for one message product where $D = 4$. Generating an entire message using the Gibbs sampler takes $O(KDN^2)$ operations which is preferable to brute force sampling when $D > 3$ in cases where KD has the same order of magnitude as N. While [23] and [24] demonstrate that the Gibbs sampler is very effective for some applications, we have found that in clutter the algorithm may generate samples in regions with very low probability mass. This can lead to problems when importance sampling since some of the $\tilde{m}_{ji}(\mathbf{s}_{ij}^n)$ in (13) may be very small relative to other samples leading to a high weight π_{ij}^n which suppresses the other samples in the set. As a workaround we adopted a heuristic to smooth these importance weights by replacing any $\pi_{ij}^n > 0.25\nu$ by

$$\pi_{ij}^n \leftarrow (\nu - \pi_{ij}^n)/6 \tag{19}$$

and renormalising (recall from step 1c in figure 1 that $\pi_{ij}^n < \nu$). We believe that the problem of generating good samples from a product of mixtures when D is large is an open research question, and discuss this further in section 4.

3. Results

3.1. Edge linking models

We exercised the algorithm using a simple edge model. In these experiments each node in the graph corresponds to an "edgel" $\mathbf{x}_i = (x_i, y_i, \alpha_i, \lambda_i) \in \mathbb{R}^4$ where (x_i, y_i) is an image coordinate, α_i is an orientation angle and λ_i is the distance between adjacent edgels. We use a simple diagonal Gaussian conditional distribution model for $j = i+1$ where

$$\psi_{ij}(\mathbf{x}_j, \mathbf{x}_i) = \mathcal{N}(x_j; x_i + \lambda_i \cos\alpha_i, \sigma_p^2) \times \mathcal{N}(y_j; y_i + \lambda_i \sin\alpha_i, \sigma_p^2) \times \mathcal{N}(\alpha_j; \alpha_i, \sigma_\alpha^2) \times \mathcal{N}(\lambda_j; \lambda_i, \sigma_\lambda^2)$$

and the model for $j = i-1$ is the same but in reverse. A similar model has been used for pixel-based illusory contour completion [1, 25] as well as finding segmentation edges using particle filtering [21], and if the ϕ_i are Gaussian it is very similar to the standard snake model [15]. Figure 2 demonstrates the effectiveness of importance sampling using the belief distribution. The top images show messages

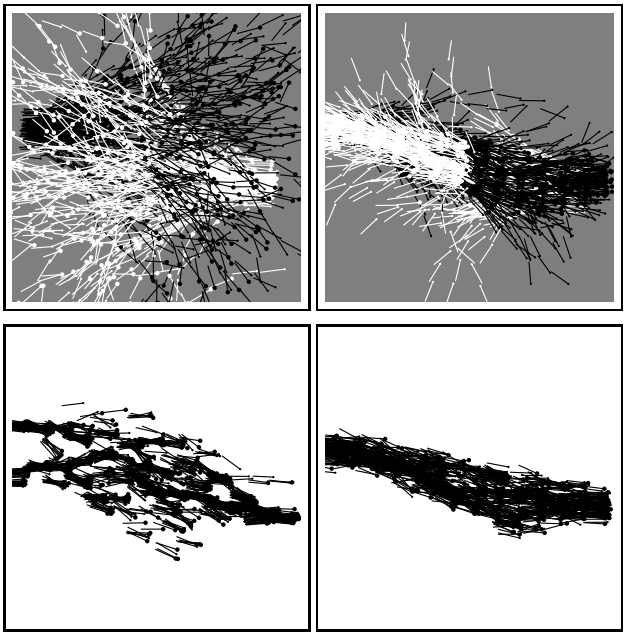

Figure 2. Importance sampling mixes information from forward and backward chains. The top images show the messages passed forwards and backwards along a chain in black and white respectively, and the bottom images show the resulting beliefs. Each displayed edgel is the mean of a Gaussian mixture component. On the left no importance sampling is used and the forward and backward messages do not interact. With importance sampling ($\nu = 0.5$) the message particles are guided to areas of high belief; this is shown on the right after 5 iterations of the algorithm.

and the bottom beliefs, and the ϕ_i are uniform to emphasise the sampling behaviour in the absence of image data. The left column shows a run of the PAMPAS algorithm with no importance sampling ($\nu = 0$) on a 16-link chain where each end of the chain is initialised from a different prior distribution. Since there is no interaction between the forward and backward messages the particles spread away from each end in a random walk producing the noisy belief distribution shown below. The right column shows the same particle sets after 5 iterations of the algorithm now using importance sampling with $\nu = 0.5$. The message particles become con-

centrated in areas of high belief making the belief estimates below less noisy.

In the absence of image measurements the above model is Gaussian and so the belief could be estimated by a variety of methods. The cluttered image data used in figure 3 causes the marginals to become highly non-Gaussian; the algorithm converges on a globally plausible solution despite strong local distractors. Here the ϕ_i used are simple robust gradient-affinity operators. The incorporation of information along the chain in both directions can be expected to improve the performance of e.g. the Jetstream algorithm [21] which is a particle-based edge finder.

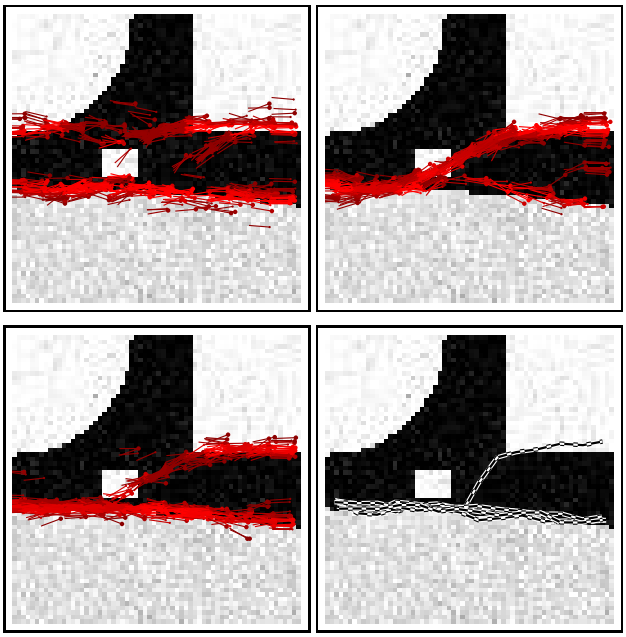

Figure 3. An edge linking model finds global solutions. The images show (in raster order) the belief estimates after 1, 5, and 10 iterations, and 7 samples from the joint graph after 10 iterations. The belief stabilises on a bimodal distribution with most of the weight of the distribution on the continuous edge as can be seen on the bottom right image.

3.2. Illusory contour matching

We tested the algorithm on an illusory-contour finding problem shown in figure 4. The input image has C "pacman"-like corner points where the open mouth defines two edge directions leading away from the corner separated by an angle C_α. The task is to match up the edges with smooth contours. We construct $2C$ chains each of four links, and each is anchored at one end with a prior starting at its designated corner pointing in one of the two edge directions. At the untethered end of a chain from corner c is a "connector" edgel which is attached to (conditionally dependent on) the connectors at the ends of the $2C-2$ other chains not starting from corner c. We use the same edgel model as in section 3.1 but now each node state-vector $\mathbf{x}_i$ is augmented with a discrete label υ_i denoting which of the $2C-2$ partner chains it matches, and the potentials at the connector are modified so that $\psi_{ij}(\mathbf{x}_j, \mathbf{x}_i) = 0$ when υ_i and υ_j do not match. This special form of potential allows us to form the message product in $O(N^2C)$ operations rather than $O(N^{2C-1})$ as a naive approach would dictate. Space does not permit a full description of the algorithm here but it is very similar to the Mixed Markov Model approach [8].

Figure 4 shows the model being applied to two scenarios with $C = 6$ in both cases. First, $C_\alpha = 11\pi/36$ which leads to two overlapping triangles with slightly concave edges when the chains are linked by the algorithm. Secondly, $C_\alpha = 7\pi/9$ which leads to a slightly convex regular hexagon. The same parameters are used for the models in both cases, so the algorithm must adapt the edgel lengths to cope with the differing distances between target corners.

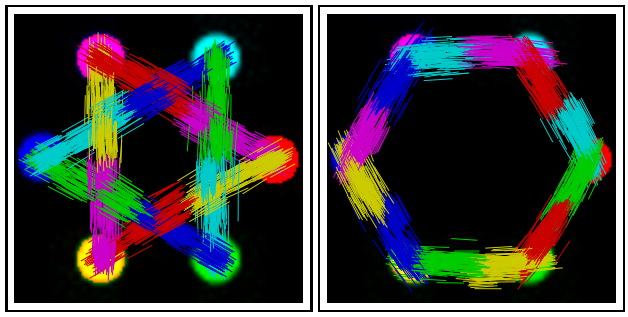

Figure 4. The PAMPAS algorithm performs illusory contour completion. Depending on the angle at which the chains leave the corner positions, the belief converges on either two overlapping triangles or a single hexagon.

3.3. Finding a jointed object in clutter

Finally we constructed a graph representing a jointed object and applied the PAMPAS algorithm (combined with the Gibbs sampler from [24]) to locating the object in a cluttered scene. The model consists of 9 nodes; a central circle with four jointed arms each made up of two rectangular links. The circle node $\mathbf{x}_1 = (x_0, y_0, r_0)$ encodes a position and radius. Each arm node $\mathbf{x}_i = (x_i, y_i, \alpha_i, w_i, h_i), 2 \leq i \leq 9$ encodes a position, angle, width and height and prefers one of the four compass directions (to break symmetry). The arms pivot around their inner joints, so the po-

tential to go from an inner arm $2 \leq i \leq 5$ to the outer arms $j = i + 4$ is given by

$$x_j = x_i + h_i \cos(\alpha_i) + \mathcal{N}(\cdot; 0, \sigma_p^2)$$
$$y_j = y_i + h_i \sin(\alpha_i) + \mathcal{N}(\cdot; 0, \sigma_p^2)$$
$$\alpha_j = \alpha_i + \mathcal{N}(\cdot; 0, \sigma_\alpha^2)$$
$$w_j = w_i + \mathcal{N}(\cdot; 0, \sigma_s^2) \;\; h_j = h_i + \mathcal{N}(\cdot; 0, \sigma_s^2)$$

and the $\psi_{1i}(\mathbf{x}_i, \mathbf{x}_1)$ from the circle outwards are similar. Going from the outer to the inner arm, the potential is not Gaussian but we approximate it with the following:

$$x_i = x_j - h_j \cos(\alpha_i) + \mathcal{N}(\cdot; 0, h_j \sigma_s |\sin((i-2)\pi/2)| \sigma_p^2)$$
$$y_i = y_j - h_j \sin(\alpha_i) + \mathcal{N}(\cdot; 0, h_j \sigma_s |\cos((i-2)\pi/2)| \sigma_p^2)$$
$$\alpha_i = \alpha_j + \mathcal{N}(\cdot; 0, \sigma_\alpha^2)$$
$$w_i = w_j + \mathcal{N}(\cdot; 0, \sigma_s^2) \;\; h_i = h_j + \mathcal{N}(\cdot; 0, \sigma_s^2).$$

The $\psi_{i1}(\mathbf{x}_1, \mathbf{x}_i)$ are straightforward:

$$x_1 = x_i + \mathcal{N}(\cdot; 0, \sigma_p^2) \quad y_1 = y_i + \mathcal{N}(\cdot; 0, \sigma_p^2)$$
$$r_1 = 0.5(w_i/\delta_w + h_i/\delta_h) + \mathcal{N}(\cdot; 0, \sigma_s^2).$$

The object has been placed in image clutter in figure 5. The clutter is made up of 12 circles and 100 rectangles placed randomly in the image. The messages are initialised with a simulated specialised feature detector: $\mathbf{x}_1$ is sampled from positions near the circles in the image and the arms are sampled to be near the rectangles, with rectangles closer to the preferred direction of each arm sampled more frequently. Each node contains $N = 75$ particles so the space is undersampled. After initialisation the PAMPAS algorithm (using the Gibbs sampler from [24] for the circle's messages and belief) is iterated without further recourse to the "feature detector" information. All of the potentials include a Gaussian outlier covering the whole image which allows samples to be generated far from the initialisation particles. After two or three iterations the belief distribution has converged on the object. In this case it might be argued that it would be more straightforward to simply detect the circles and perform an exhaustive search near each. The second example of figure 5 demonstrates the power of the probabilistic modelling approach, since now the circle at the centre of the object has not been displayed, simulating occlusion. Of course no $\mathbf{x}_1$ initialisation samples are generated near the "occluded" circle, but even after one iteration the belief at that node has high weight at the correct location due to the agreement of the nearby arm nodes. After a few iterations the belief has converged on the correct object position despite the lack of the central circle.

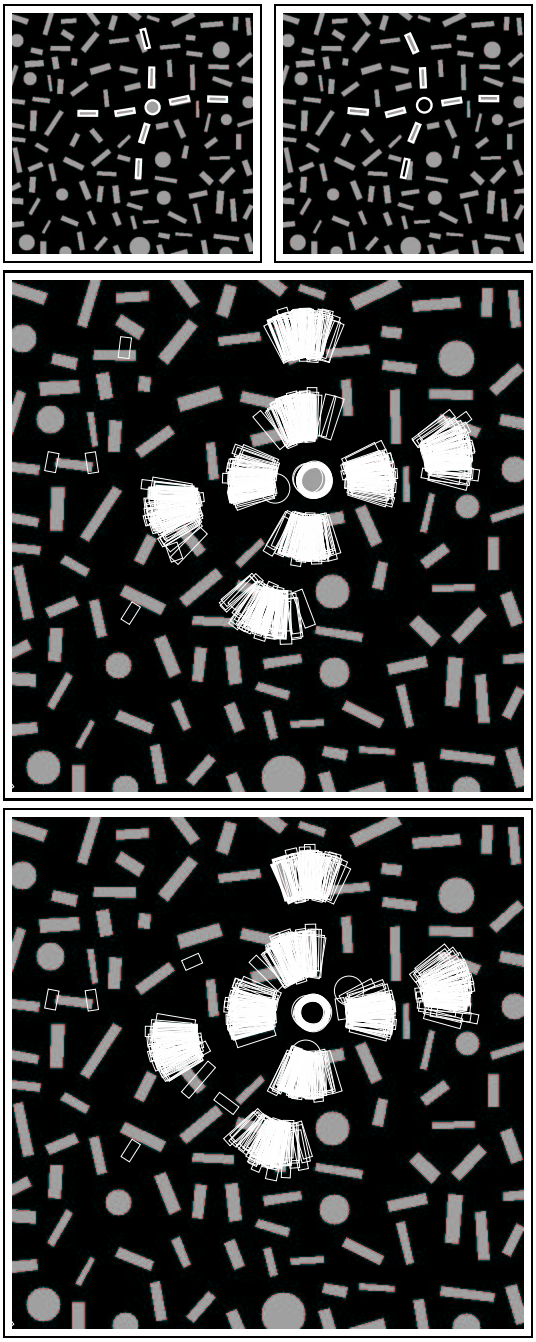

Figure 5. A jointed object is located in clutter. Two experiments are shown, with and without a simulated occlusion. The top two images show a sample from the joint distribution for each experiment, while the lower images show the full distributions. The second experiment models an occluded object with no circle present in the image where the belief has placed it. This experiment uses the Gibbs sampler from [24] to form the message product for the central node.

4. Conclusion

We have concentrated in this paper on examples with synthetic data in order to highlight the working of the PAM-

PAS algorithm. The applications we have chosen are highly suggestive of real-world problems, however, and we hope the promising results will stimulate further research into applying continuous-valued graphical models to a variety of computer vision problems. Sections 3.1 and 3.2 suggest a variety of edge completion applications. Section 3.3 indicates that the PAMPAS algorithm may be effective in locating articulated structures in images, in particular people. Person-finding algorithms exist already for locating and grouping subparts [20] but they do not naturally fit into a probabilistic framework. Recently several researchers have had promising results finding jointed structures in images using discretised graphical model representations [6, 10, 4]. Both discretising and using the PAMPAS algorithm result in approximations, and the tradeoffs between the two approaches remain to be investigated.

One obvious advantage of using a graphical model representation is that it extends straightforwardly to tracking, simply by increasing the size of the graph and linking nodes in adjacent timesteps. Another area of great interest is linking the continuous-valued graph nodes to lower-level nodes which may be discrete-valued as in [13]. In fact, researchers in the biological vision domain have recently proposed a hierarchical probabilistic structure to model human vision which maps closely onto our algorithmic framework [17].

We are performing current research, to be published elsewhere, into the problem of sampling from products of Gaussian mixtures. The Gibbs sampler of [24] is very flexible and straightforward to implement and performs well on some important models, but for cluttered distributions of the type we have encountered K must be set prohibitively large to get reliable samples. We have developed heuristic algorithms to address this clutter problem which perform well on our test applications, but are only practical for $D < 9$. The design of an efficient, generally effective sampler which scales to large D remains an open problem.

Acknowledgements

Helpful suggestions to improve this work were contributed by Michael Black, David Mumford, Mike Burrows, Martin Abadi, Mark Manasse and Ian Reid.

References

[1] J. August and S.W. Zucker. The curve indicator random field: curve organization via edge correlation. In *Perceptual Organization for Artificial Vision Systems*. Kluwer Academic, 2000.

[2] C.M. Bishop, D. Spiegelhalter, and J. Winn. Vibes: A variational inference engine for bayesian networks. In *NIPS*, 2002.

[3] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence*, 1998.

[4] J. Coughlan and S. Ferreira. Finding deformable shapes using loopy belief propagation. In *Proc. 7th European Conf. Computer Vision*, 2002.

[5] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[6] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 66–73, 2000.

[7] W.T. Freeman, E.C. Pasztor, and O.T. Carmichael. Learning low-level vision. *Intl. J. Computer Vision*, 40(1):25–47, 2000.

[8] A. Fridman. Mixed markov models. In *Proc. Second International ICSC Symposium on Neural Computation*, 2000.

[9] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[10] S. Ioffe and D. Forsyth. Human tracking with mixtures of trees. In *Proc. 8th Int. Conf. on Computer Vision*, volume 1, pages 690–695, 2001.

[11] M. Isard and A. Blake. Condensation — conditional density propagation for visual tracking. *Int. J. Computer Vision*, 28(1):5–28, 1998.

[12] M.A. Isard and A. Blake. A smoothing filter for Condensation. In *Proc. 5th European Conf. Computer Vision*, volume 1, pages 767–781, 1998.

[13] N. Jojic, N. Petrovic, B. J. Frey, and T. S. Huang. Transformed hidden markov models: Estimating mixture models of images and inferring spatial transformations in video sequences. In *Proc. Conf. Computer Vision and Pattern Recognition*, 2000.

[14] M.I. Jordan, T.J. Sejnowski, and T. Poggio, editors. *Graphical Models : Foundations of Neural Computation*. MIT Press, 2001.

[15] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proc. 1st Int. Conf. on Computer Vision*, pages 259–268, 1987.

[16] D. Koller, U. Lerner, and D. Angelov. A general algorithm for approximate inference and its application to hybrid bayes nets. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence*, pages 324–333, 1999.

[17] T.S. Lee and D. Mumford. Hierarchical bayesian inference in the visual cortex. *Journal of the Optical Society of America A*, 2002 (submitted).

[18] U. Lerner. *Hybrid Bayesian Networks for Reasoning about Complex Systems*. PhD thesis, Stanford University, October 2002.

[19] T.P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT, January 2001.

[20] G. Mori and J. Malik. Estimating human body configurations using shape context matching. In *Proc. 7th European Conf. Computer Vision*, pages 666–680, 2002.

[21] P. Perez, A. Blake, and M. Gangnet. Jetstream: Probabilistic contour extraction with particles. In *Proc. 8th Int. Conf. on Computer Vision*, volume II, pages 524–531, 2001.

[22] H. Sidenbladh, M.J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *Proc. 7th European Conf. Computer Vision*, volume 1, pages 784–800, 2002.

[23] E.B. Sudderth, A.T. Ihler, W.T. Freeman, and A.S. Willsky. Nonparametric belief propagation. Technical Report P-2551, MIT Laboratory for Information and Decision Systems, 2002.

[24] E.B. Sudderth, A.T. Ihler, W.T. Freeman, and A.S. Willsky. Nonparametric belief propagation. In *CVPR*, 2003.

[25] L.R. Williams and D.W. Jacobs. Stochastic completion fields: A neural model of illusory contour shape and salience. *Neural Computation*, 9(4):837–858, 1997.

[26] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. Technical Report TR2001-22, MERL, 2001.