LAMP-TR-***
CAR-TR-***
CS-TR-****

MDA 9049-6C-1250
September 2000

Hidden Markov Models for Images

Daniel DeMenthon, Marc Vuilleumier
and David Doermann

Hidden Markov Models for Images

Daniel DeMenthon, Marc Vuilleumier and David Doermann

Language and Media Processing Laboratory (LAMP)
University of Maryland
College Park, MD 20742-3275

Abstract

We describe a method for learning statistical models of images using a second-order hidden Markov mesh model. We show that the Viterbi algorithm approach used for segmenting Markov chains can be extended to Markov meshes. The segmental k-means algorithm can then be applied to iteratively estimate the state transition matrix and the probability densities of the observations for the model. We also describe a semi-Markov modeling technique in which the distributions of width and heights of the segmented regions are modeled explicitly. Finally, we propose a distance measure between images based on the similarity of their statistical models, for classification and retrieval tasks.

The support of this research by the Department of Defense under contract MDA 9049-6C-1250 is gratefully acknowledged.

1 Introduction

In this paper we address the problem of constructing statistical models of images using Hidden Markov modeling techniques developed in speech recognition. Contextual information in the image is encoded by a 3D array of transition probabilities. A labeling of the image pixels is produced by a global optimization over the whole image using a dynamic programming approach which closely follows the Viterbi algorithm commonly used to segment Markov chains. A segmental k-means technique is used to learn the parameters of the statistical model from the image. The primary objective of this work is the computation of a distance measure between images that is less sensitive to camera pan than techniques based on image feature vectors, yet is able to encode spatial image characteristics more specifically than current popular techniques such as histogram representations. The most important application of such a distance measure is for classification, indexing and retrieval of visual data in image and video databases. However, the statistical models produced by the proposed method can also be applied to the solution of problems for which non causal Markov random field techniques have been used [12], for example segmentation, data compression, image restoration, and more generally the calculation of probabilities in Bayesian modeling for image processing [24, 16].

2 Related Work

Markov mesh random field models were introduced by Abend, Harley and Kanal [1] and by Chow [5] in the sixties. They form a causal (or unilateral) subclass of the Markov random field (MRF) models pioneered in [3]. Some of the problems addressed in the study of MRFs are the labeling of the pixels, and the computing of joint likelihoods of the labels. In the case of non causal Markov fields, iterative solutions are available, based on the Gibbs distribution, but the computational cost is generally high [10]. In this paper we develop a method for developing a labeling using a global optimization, for the simplest Markov mesh model, a second-order model, i.e. a model in which the label likelihoods depend on the labels of only two past pixel neighbors when the pixels are observed by a raster scan. (The order of a Markov mesh model is the number of pixels which the "past" of each pixel can be reduced to.) This model has been studied extensively by Devijver. Early work by Devijver [6] developed a pixel labeling technique based on a "real time" optimization, in the sense that only the pixels seen prior to the current pixel while scanning the image in raster scan mode are used in the choice of the best label for the current pixel. That technique is an extension to a mesh of the forward algorithm used in training hidden Markov models (HMMs) on Markov chain. Later work [7] extends this type of approach to third-order Markov chains, and introduces a learning "event counting" technique which amounts to a segmental k-means estimation of model parameters. For closely related theoretical work, refer to [15, 9]. Park and Lee show that this type of approach is quite practical for offline handwritten character recognition [18]. By using a Pickard random field image model, which assumes rather restrictive local symmetries of label dependencies [19, 14], Devijver [8] develops (for binary images only) a global labeling optimization and learning technique, by deriving an expression similar to the Baum-Welch forward-backward recurrence formula. Our work also proposes a global optimization of pixel labeling and a model learning technique, but the labeling

optimization is similar to a Viterbi algorithm and the training is performed by a segmental-k means technique. Our approach does not require symmetry constraints on the Markov mesh model, and it applies to color images.

A rather different tack for applying Hidden Markov models to images can be found in the literature, the so-called "pseudo 2-D" HMM approach [13]. It does not attempt to model the local simultaneous dependencies of pixels to row and column neighbors. Instead, rows of the image are segmented by a left-right version of 1-D HMM. Then the patterns of the segmented rows can be grouped into vertical segments containing similar rows by applying an HMM in which the states (called superstates to avoid confusion with the states of the row HMMs) correspond to different families of segmentation patterns within rows. This technique is being applied in many domains, for example in optical character recognition [13], color image retrieval [17], face identification [23]. The method proposed in this paper produces a more descriptive model of image statistics, in the sense that 2-D local mesh context is described by the model. Therefore its domain of application potentially overlaps that of pseudo 2-D HMMs.

3 Notations

We consider a rectangular $U \times V$ image. For each pixel location (u, v), we have two quantities, an observable vector $\mathbf{o}_{u,v}$ and a hidden state $q_{u,v}$ which can take the discrete values $\{1, 2, \ldots, N\}$. The components of the observation vector $\mathbf{o}_{u,v}$ for a pixel are the color components in this paper, but we could instead consider DCT coefficients, or the results of a local filter. The hidden state $q_{u,v}$ can be viewed as a label for the pixel. The total number of states N used in the modeling is *specified*. The labeling of each pixel by a state provides a quantization of the observation vectors, and a segmentation of the image.

The relationship between observation vectors and states is part of the model and is expressed statistically. The modeling process will compute the probability distributions of observation values given state values, $P(\mathbf{o}|q)$.

We also define:
$R_{u,v}$ the rectangle of image pixels defined by diagonal corner pixels $(0, 0)$ and (u, v);
$\mathbf{x}_{u,v}$ the vector of combined state $q_{u,v}$ and observation $\mathbf{o}_{u,v}$ at pixel (u, v): $\mathbf{x}_{u,v} = (q_{u,v}, \mathbf{o}_{u,v})$;
$Q_{u,v}$ and $\mathbf{O}_{u,v}$ the configurations of states and observations on rectangles $R_{u,v}$;
$\mathbf{X}_{u,v}$ the rectangular configuration of joint states and observations on rectangles $R_{u,v}$.

We assume that the statistical properties don't change across the image (in other words, the random field is homogeneous). Therefore, often only relative pixel positions are important, and we can simplify pixel index notations by omitting indices for the observed pixel, and using index e for its east pixel, index n for its north pixel, and index ne for its north-east pixel.

Conditional probabilities for states at neighbor pixels are written $P(q|q_e, q_n)$, which is shorthand for the full expression $P(q = k|q_e = i, q_n = j)$, where i, j and k are the specific instances, or labels, taken by the pixel states. They define a 3D transition probability matrix.

Conditional probabilities for states at neighbor pixels will be written $P(q_{u,v}|q_{u,v-1}, q_{u-1,v})$, which is shorthand for the full expression $P(q_{u,v} = k|q_{u,v-1} = i, q_{u-1,v} = j)$, where i, j and k are the specific instances, or labels, taken by the pixel states. We will assume the modeled field to be homogeneous over the image; in other words, the probability just written will have the same

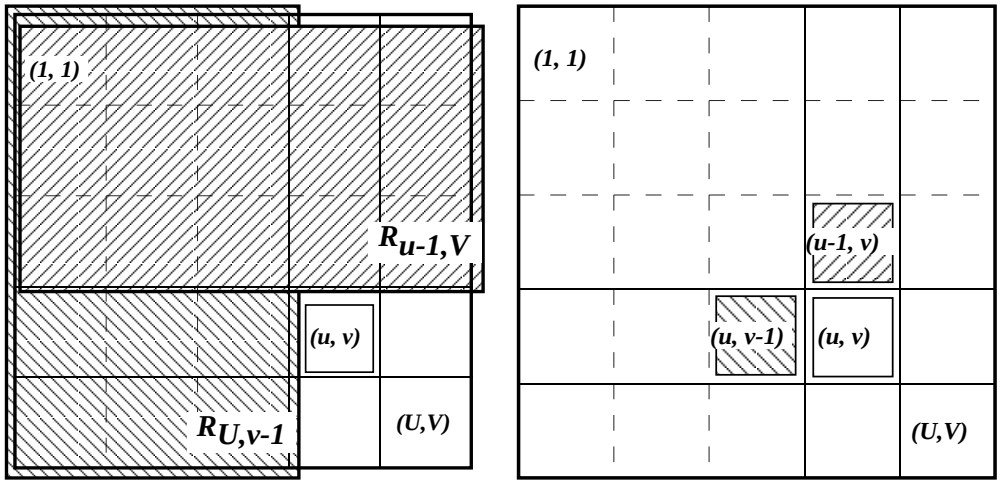


Figure 1: With a second-order causal Markov mesh model applied to an image $I_{U,V}$, the probability of seeing a state i at (u, v) conditional to the state configurations for pixels in rectangles $R_{U,v-1}$, $R_{u-1,V}$ (a) is equal to that probability conditional to the states of the left and top neighbors (b).

value for any pixel (u, v) wherever we consider the probability for the state of this pixel to be k, the state of the left pixel to be i and the state of the top pixel to be j. Therefore we will also use a_{ijk} as shorthand for $P(q_{u,v} = k|q_{u,v-1} = i, q_{u-1,v} = j)$ in discussions where geometric considerations are not important. The 3D matrix a_{ijk} is the transition matrix of the model, i.e. the probability while doing a raster scan of jumping to a pixel in state k if we just saw a pixel in state i in the same row and we had seen a pixel in state j in the same column in the previous row.

4 Markov Mesh Properties

We assume the configuration of states in the image to be driven by a homogeneous second-order Markov random mesh model. The defining property is described in Fig. 1 and can be expressed as [1]:

$$P(q_{u,v}|Q_{U,v-1} \cup Q_{u-1,V}) = P(q_{u,v}|q_{u,v-1}, q_{u-1,v}) \tag{1}$$

where we relax the notations to mean the following at the boundaries:

$$\begin{aligned}
P(q_{u,v}|q_{u,v-1}, q_{u-1,v}) &= P(q_{1,1}) \text{ if } u = v = 1 \\
P(q_{u,v}|q_{u,v-1}, q_{u-1,v}) &= P(q_{u,1}|q_{u-1,1}) \text{ if } u > v,\ v = 1 \\
P(q_{u,v}|q_{u,v-1}, q_{u-1,v}) &= P(q_{1,v}|q_{1,v-1}) \text{ if } v > u,\ u = 1
\end{aligned}$$

In other words, the probability of having a given state at pixel (u, v) conditional to the states in all the pixels on the rows above row u and all the remaining pixels to the left of column v is simply equal to the probability of having that state at pixel (u, v) conditional to the state of the pixel above (u, v) and to the state of the pixel to the left of (u, v) (Fig. 1).
Kanal and coworkers showed that the probability of seeing the configuration of states $Q_{u,v}$ over the rectangle $R_{u,v}$ is then equal to the product of the probabilities of each pixel in the rectangle, each probability being conditional to the states of the east and north neighbor pixels [1].

$$P(Q_{u,v}) = \prod_{u'=1}^{u} \prod_{v'=1}^{v} P(q_{u',v'}|q_{u',v'-1}, q_{u'-1,v'}) \tag{2}$$

This factorization is what makes this Markov model attractive. It is similar in form to the factorization of the joint probability in a 1D Markov chain using first order dependencies between states. However, the proof for this property is slightly more involved and is reproduced in the Appendix.

The observation $\mathbf{o_{u,v}}$ at pixel (u, v) is assumed to depend only on the state $q_{u,v}$ at that pixel. Therefore a second factorization follows:

$$P(\mathbf{o}_{1,1}, \mathbf{o}_{1,2}, \ldots, \mathbf{o}_{u,v}|q_{1,1}, q_{1,2}, \ldots, q_{u,v}) = \prod_{u'=1}^{u} \prod_{v'=1}^{v} P(\mathbf{o}_{u',v'}|q_{u',v'}) \tag{3}$$

or, using $\mathbf{O}_{u,v}$ and $Q_{u,v}$ to express the configuration of observations $\mathbf{o}_{i,j}$ and $q_{i,j}$ over the rectangle $R_{u,v}$,

$$P(\mathbf{O}_{u,v}|Q_{u,v}) = \prod_{u'=1}^{u} \prod_{v'=1}^{v} P(\mathbf{o}_{u',v'}|q_{u',v'}) \tag{4}$$

Now consider the event $\mathbf{X}$ defined by the joint occurrence of specific states q and observations $\mathbf{o}$ at pixels of rectangle R: By combining the factorizations of Eq. 2 and 4, it is straightforward to express $\mathbf{X}$ as the product of individual $P(\mathbf{x})$ at each pixel of the rectangle:

$$\begin{aligned}
P(\mathbf{X}_{u,v}) &= P(\mathbf{O}_{u,v}, Q_{u,v}) \\
&= P(\mathbf{O}_{u,v}|Q_{u,v})P(Q_{u,v}) \\
&= [\prod_{u'=1}^{u} \prod_{v'=1}^{v} P(\mathbf{o}_{u',v'}|q_{u',v'})][\prod_{u'=1}^{u} \prod_{v'=1}^{v} P(q_{u',v'}|q_{u'-1,v'}, q_{u',v'-1})] \\
&= \prod_{u'=1}^{u} \prod_{v'=1}^{v} P(\mathbf{o}_{u',v'}|q_{u',v'})P(q_{u',v'}|q_{u'-1,v'}, q_{u',v'-1}) \\
&= \prod_{u'=1}^{u} \prod_{v'=1}^{v} P(\mathbf{o}_{u',v'}|q_{u',v'}, q_{u'-1,v'}, q_{u',v'-1})P(q_{u',v'}|q_{u'-1,v'}, q_{u',v'-1}) \\
&= \prod_{u'=1}^{u} \prod_{v'=1}^{v} P(\mathbf{o}_{u',v'}, q_{u',v'}|q_{u'-1,v'}, q_{u',v'-1})
\end{aligned}$$

$$P(\mathbf{X}_{u,v}) = \prod_{u'=1}^{u} \prod_{v'=1}^{v} P(\mathbf{x}_{u',v'}|q_{u'-1,v'}, q_{u',v'-1}) \tag{5}$$

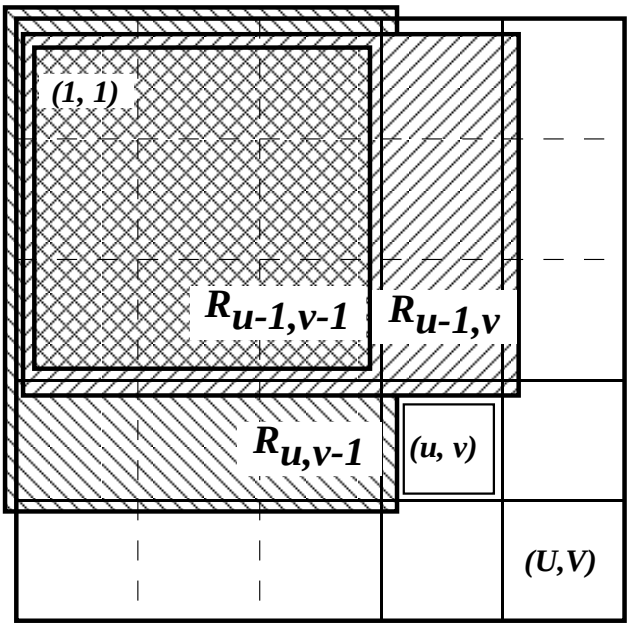


Figure 2: The joint probability for the configuration of pixel states over a rectangle R with a corner at pixel (u, v) can be expressed simply in relation to the probabilities for the configurations over rectangles R_e, R_n and R_{ne} with respective corners east, north and north-east of pixel (u, v), and to the conditional probability $P(q|q_e, q_n)$ of seeing a state q at pixel (u, v).

Therefore we have obtained a simple factorization for the probability of the event $\mathbf{X}$ defined by the joint occurrence of a specific map of states q and observations $\mathbf{o}$ at pixels of rectangle R, in terms of the individual probabilities of occurrences of states q and observations $\mathbf{o}$ (i.e. probabilities of events $\mathbf{x}$) at each pixel.

5 Recursive Expression for Joint state Probabilities

Our goal here is to develop a recursive expression of the joint probability over a rectangle that will allow us to implement for a Markov mesh an algorithm similar to the Viterbi algorithm used for Markov chains.

Theorem: The joint probability for the configuration of states Q over a rectangle R defined by diagonal pixels $(0, 0)$ and (u, v) can be expressed in relation to the probabilities for the configurations Q_e, Q_n and Q_{ne}, and to the conditional probability of seeing the state q at pixel (u, v). The expression is:

$$P(Q) = \frac{P(Q_e)P(Q_n)}{P(Q_{ne})} P(q|q_e, q_n) \tag{6}$$

Proof: We start from the right hand side of the expression and find the left hand side. Each of the three terms of the form $P(Q)$ in the right hand side is a product of probabilities for each pixel in its rectangle R, according to Eq. 2. The two probabilities in the numerator correspond to two rectangles R_e and R_n that overlap, and therefore when expanded by factorization the

numerator has the probabilities for the pixels of the overlapping rectangle appearing twice as factors (see Fig. 2). The probability of the denominator is for a rectangle R_ne that is exactly equal to the rectangle of overlap, therefore the factors appearing a second time in the numerator are eliminated by simplification with the factors of the denominator. What is left is the product of all the probabilities that correspond either to rectangle R_e or to rectangle R_n, but not both. This accounts for all the probabilities over rectangle R, except for the probability $P(q|q_e, q_n)$ corresponding to pixel (u, v). Once this factor is included as shown in Eq. 6 we have all the factors required to yield $P(Q)$ according to Eq. 6.

Now, using Eq. 5, we can write the same type of decomposition for $P(\mathbf{X})$ as for $P(Q)$ and justify it with the same proof:

$$P(\mathbf{X}) = \frac{P(\mathbf{X}_e)P(\mathbf{X}_n)}{P(\mathbf{X}_{ne})} P(\mathbf{x}|q_e, q_n) \tag{7}$$

6 Dynamic Programming Algorithm

In this section we will follow closely (for a second-order Markov model and a Markov mesh) the presentation made by Rabiner for the Viterbi algorithm applied to a first-order model of a Markov chain [21, 22]. We are looking for the single best state configuration $Q_{U,V}$ over the whole image lattice for the given observation map (image) $\mathbf{O}_{u,v}$, i.e. the state configuration that maximizes the probability $P(Q_{u,v}|\mathbf{O}_{u,v})$, or equivalently maximizes the probability $P(\mathbf{O}_{u,v}, Q_{u,v})$, which we called $P(\mathbf{X}_{u,v})$. This is a global optimization problem, but thanks to the decomposition of Eq. 7, it can be solved by dynamic programming [2]. We define the quantity

$$\delta_{u,v}(q_{u,v}) = \max_{Q_{u,v}-\{q_{u,v}\}} P(Q_{u,v}, \mathbf{O}_{u,v}) = \max_{Q_{u,v}-\{q_{u,v}\}} P(\mathbf{X}_{u,v}) \tag{8}$$

where the notation $Q_{u,v}-\{q_{u,v}\}$ represents the configuration of states for all pixels of the rectangle $R_{u,v}$, except pixel (u, v). Therefore, when maximized over $Q_{u,v} - \{q_{u,v}\}$, the expression remains a function of $q_{u,v}$.

Once we are able to compute such an expression for any (u, v), then the maximum joint probability for the whole image configuration $Q_{U,V}$ is simply:

$$\max_{Q_{U,V}}(P(Q_{U,V}, \mathbf{O}_{U,V}) = \max_{q_{U,V}} \delta_{U,V}(q_{U,V}) \tag{9}$$

We apply the recursion expression of Eq. 7 in the definition 8 of $\delta_{u,v}(q_{u,v})$ and obtain a recursive equation for $\delta_{u,v}(q_{u,v})$ by the following steps:

$$\begin{aligned}
\delta_{u,v}(q_{u,v}) &= \max_{Q_{u,v}-\{q_{u,v}\}} [\frac{P(\mathbf{X}_{u,v-1})P(\mathbf{X}_{u-1,v})}{P(\mathbf{X}_{u-1,v-1})} P(\mathbf{X}_{u,v}|q_{u,v-1}, q_{u-1,v})] \\
&\simeq \frac{\max_{\{q_{u,v-1},q_{u-1,v}\}}[\delta_{u,v-1}(q_{u,v-1})\delta_{u-1,v}(q_{u-1,v})P(\mathbf{x}_{u,v}|q_{u,v-1}, q_{u-1,v})]}{\delta_{u-1,v-1}(q^*_{u-1,v-1})}
\end{aligned}$$

$$\delta_{u,v}(q_{u,v}) = \max_{q_{u,v-1},q_{u-1,v}} [\delta_{u,v-1}(q_{u,v-1})\delta_{u-1,v}(q_{u-1,v})P(q_{u,v}|q_{u,v-1}, q_{u-1,v})] \frac{P(\mathbf{o}_{u,v}|q_{u,v})}{\delta_{u-1,v-1}(q^*_{u-1,v-1})} \tag{10}$$

where $1 \leq q_{u,v} \leq N$ and $q^*_{u-1,v-1}$ is the state that corresponds to the best score for $\delta_{u-1,v-1}(k)$. We made use of properties such as

$$\max_{Q_{u,v}-\{q_{u,v}\}} P(\mathbf{X}_{u,v-1}) = \max_{q_{u,v-1}} [\max_{Q_{u,v-1}-\{q_{u,v-1}\}} P(\mathbf{X}_{u,v-1})] = \max_{q_{u,v-1}} \delta_{u,v-1}(q_{u,v-1})$$

This expression is an approximation, in the sense that we have assumed that the maximum of the expression is obtained when the common part $P(\mathbf{X}_{u-1,v-1})$ between $P(\mathbf{X}_{u,v-1})$ and $P(\mathbf{X}_{u-1,v})$ is maximum. This common part also appears at the denominator, and its maximum is $\delta_{u-1,v-1}(q^*_{u-1,v-1})$. The approximation assumes that the optimization of δ at (u, v) should not involve pixel $(u-1, v-1)$, which is in line with the second-order Markov mesh simplification assuming that the states at these pixels are independent.

We don't know the index $q^*_{u-1,v-1}$ yet at the time we do the calculation. However the arguments $q_{u,v-1}$ and $q_{u-1,v}$ that maximize Eq. 10 for each $q_{u,v}$ are independent of the value of $\delta_{u-1,v-1}(q^*_{u-1,v-1})$. Therefore, the main purpose of the Viterbi algorithm – finding the configuration of optimal pixel states – can be achieved without attempting to compute the best possible value for $\delta_{u-1,v-1}(q^*_{u-1,v-1})$. The other purpose of the Viterbi algorithm – obtaining a correct value for the joint probability – can be directly obtained once the optimum state configuration has been computed, by applying Eq. 3.

The division by $\delta_{u-1,v-1}(q^*_{u-1,v-1})$ can be seen as a renormalization that prevents the probabilities over the rectangle $R_{u-1,v-1}$ from being counted twice, since both $\delta_{u,v-1}(q_{u,v-1})$ and $\delta_{u-1,v}(q_{u-1,v})$ include these probabilities. Therefore, even if the term $\delta_{u-1,v-1}(q^*_{u-1,v-1})$ is not directly instrumental in the computation of the optimal state configuration, we cannot simply set this term to 1, because of this renormalizing effect. For larger images, the computation would be subjected to underflow. However, any scaling technique is appropriate. One scaling technique consists of computing this term for the state that maximizes it. Another scaling technique consists of normalizing the components of each term δ as soon as it has been computed by Eq. 10.

As a third scaling technique, when the Viterbi algorithm is run in a iteration loop as described in the training procedure below, the map of best states $q^*_{u,v}$ becomes available from previous scans and is refined in the iteration process. Then the Viterbi algorithm provides a correct joint probability of states and observations at the end of the iteration, and applying Eq. 3 is not necessary.

To retrieve the state map when the bottom right corner is reached, we need to keep track of the states $q_{u,v-1}$ and $q_{u-1,v}$ that maximized Eq. 10. We store these values in the arrays $\psi^H_{u,v}(q_{u,v})$ and $\psi^V_{u,v}(q_{u,v})$ respectively, where the letters H and V are a reminder that these arrays are used to backtrack along horizontal and vertical traces of the paths. The complete backtracking procedure is described in Step 4 below.

The steps of the Viterbi algorithm can be summarized as follows:

1. Initialization

$$\delta_{1,1}(q_{1,1}) = P(q_{1,1})P(\mathbf{o}_{1,1}|q_{1,1}),\ 1 \leq q_{1,1} \leq N$$

$$\delta_{u,1}(q_{u,1}) = \delta_{u-1,1}(q_{u-1,1})P(q_{u,1}|q_{u-1,1})P(\mathbf{o}_{u,1}|q_{u,1}),\ 1 < u \leq U$$

$$\delta_{1,v}(q_{1,v}) = \delta_{1,v-1}(q_{1,v-1})P(q_{1,v}|q_{1,v-1})P(\mathbf{o}_{1,v}|q_{1,v}),\ 1 < v \leq V$$

2. Recursion

$$\delta_{u,v}(q_{u,v}) = \max_{q_{u,v-1},q_{u-1,v}} [\delta_{u,v-1}(q_{u,v-1})\delta_{u-1,v}(q_{u-1,v})P(q_{u,v}|q_{u,v-1},q_{u-1,v})] \frac{P(\mathbf{o}_{u,v}|q_{u,v})}{\delta_{u-1,v-1}(q^*_{u-1,v-1})}$$

$$(\psi^H_{u,v}(q_{u,v}), \psi^V_{u,v}(q_{u,v})) = \arg \max_{q_{u,v-1},q_{u-1,v}} \delta_{u,v}(q_{u,v})$$

3. Termination

$$P^*(Q_{U,V}, \mathbf{O}_{U,V}) \equiv \max_{Q_{U,V}}(P(Q_{U,V}, \mathbf{O}_{U,V}) = \max_{q_{U,V}} \delta_{U,V}(q_{U,V})$$

$$q^*_{U,V} = \arg \max_{q_{U,V}} \delta_{U,V}(q_{U,V})$$

4. Path (state map) backtracking

The optimal states for image pixels can be obtained by backtracking from the known best state of the bottom right pixel (U, V). However, there are multiple ways of arriving at any pixel, and each may result in a different selected state.

For example, consider an image in which each pixel can be labeled by state 1, 2 or 3 (Fig. 3). These 3 states are represented by a vertical pile of 3 boxes at each pixel, and in each state box we wrote the number of paths starting from pixel (U, V) that have reached that state when passing through the pixel. When backtracking from pixel (U, V) to pixel $(U-1, V-2)$ there are three paths.

1. Path $H - H - V$:

$$q^*_{U,V} = 2,\ \psi^H_{U,V}(2) = 3,\ \psi^H_{U,V-1}(3) = 3,\ \psi^V_{U,V-2}(3) = 2$$

2. Path $V - H - H$:

$$q^*_{U,V} = 2,\ \psi^V_{U,V}(2) = 3,\ \psi^H_{U-1,V}(3) = 3,\ \psi^H_{U-1,V-1}(3) = 1$$

3. Path $H - V - H$:

$$q^*_{U,V} = 2,\ \psi^H_{U,V}(2) = 3,\ \psi^V_{U,V-1}(3) = 3,\ \psi^H_{U-1,V-1}(3) = 1$$

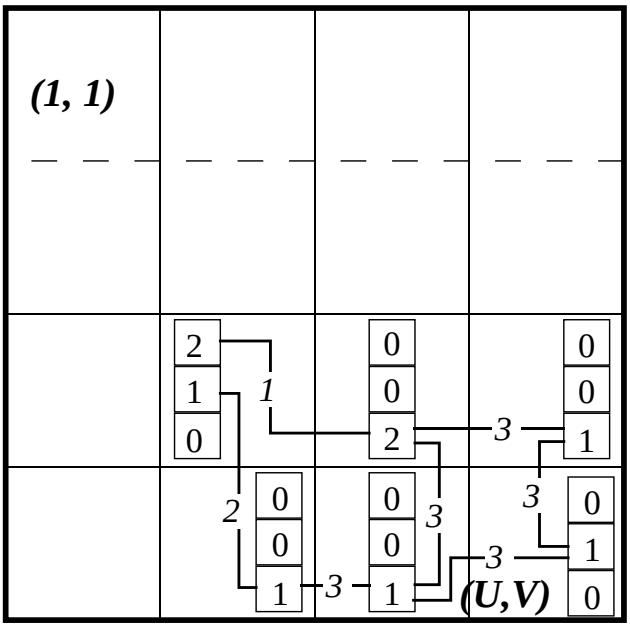


Figure 3: The best states $q^*_{u,v}$ for each pixel (u, v) are those that are reached by the highest number of backtraces from the best state of the lower right corner of the image, using backtracking matrices $\psi^H_{u,v-1}$ and $\psi^V_{u-1,v}$.

Thus for pixel $(U-2, V-1)$ we arrive at state 2 if we backtrack horizontally twice, then vertically; state 1 if we backtrack vertically, then horizontally twice; and state 1 again if we backtrack horizontally, vertically, then horizontally again.

There are ways of enforcing the consistency of alternate paths within each recursion step within the dynamic programming process, however this local decision process could reduce the quality of the optimization.

Instead, we take the approach of selecting as best state for a pixel the state which is obtained by the *largest number of backtracking paths*. For the example above, state 1 would be selected for pixel $(U-1, V-2)$ because it is reached twice, whereas state 2 is reached only once, and state 3 is not reached at all.

Therefore for each pixel (u, v), we keep the count of the number of times each state has been reached by a backtracking path. This does not require enumerating the paths, and can be performed for the whole image in time proportional to the number of pixels, by noticing that all the paths to pixel (u, v) have to come from either $(u, v+1)$ or from $(u+1, v)$, so that the path counts for (u, v) are obtained locally by transmitting the path counts at $(u, v+1)$ and at $(u+1, v)$, along the subpaths defined by $\psi^H_{u,v}$ and $\psi^V_{u,v}$.

For each pixel the path counts for each state i are kept as components i of a vector $S_{u,v}(i)$. The process starts with the pixel at the lower right corner of the image with a path count of 1 for state $q^*_{U,V}$ (found at the termination step described above) and zero for the other states. Therefore the algorithm is as follows:

(a) (b) (c)

Figure 4: Original 64 × 64 Lenna picture (figure (a)), 10-color picture segmented by proposed Viterbi and segmental k-means approach using 10 states (figure (b)), and location of ambiguous pixel states marked as grey pixels, with whiter pixels more "entropic" than darker ones (figure (c)).

4.1. Backtracking initialization

$$S_{u,v}(i) = 0, \;\; 1 \leq i \leq N, 1 \leq u \leq U, \; 1 \leq v \leq V$$

$$S_{U,V}(q^*_{U,V}) = 1$$

4.2. Backtracking recursion

$$S_{u,v}(\psi^H_{u,v+1}(i)) = S_{u,v}(\psi^H_{u,v+1}(i)) + S_{u,v+1}(i), \;\; 1 \leq i \leq N, \; 1 \leq u < U, \; 1 \leq v < V$$

$$S_{u,v}(\psi^V_{u+1,v}(i)) = S_{u,v}(\psi^H_{u,v+1}(i)) + S_{u+1,v}(i), \;\; 1 \leq i \leq N, \; 1 \leq u < U, \; 1 \leq v < V$$

4.3. Backtracking termination

$$q^*_{u,v} = \arg\max_i S_{u,v}(i), 1 \leq i \leq N, 1 \leq u < U, \; 1 \leq v < V$$

We view the multiple state solutions as additional information about the likelihood of states at different pixels. This is supported by the observation that for most pixels in the images we tested, there is no ambiguity about the best state; in other words, all the paths lead to a single state for most pixels. The pixels reached by more than one path are located at those borders between state regions where the gradients are low and it is not clear when state should be switched. For the Lenna image (Fig. 4), there are only two or three pixels per row with more than one paths, and only four pixels with three paths. Pixels with four path seem very rare (we haven't seen any yet). In the right picture of Fig. 4 we have switched pixels with multiple paths from color pixels to grey pixels, and we have used a grey scale that represents the ambiguity due

to multiple paths, using an entropy measure. The ambiguity is zero if there is only one path, and it is close to zero if there are two paths but the path count for one is a much larger proportion of the total path count for the pixel than the other path count. It is maximum if there are a larger number of paths with almost equal path counts. Entropy is a good measure of this type of ambiguity. It is computed as $-\sum_{i=1}^{N} c_i Log(c_i)$ where c_i is the path count for state i. The grey levels are normalized so that the pixel with the maximum entropy is white, and the pixel with minimum non zero entropy is black (pixels with zero entropy retain their original color).

7 Modeling of observation probabilities

As mentioned above, as is customary for HMMs in speech recognition, the distributions of observations are assumed to depend on the states of the pixels only, and not on the observations of neighbor pixels (the relationship between pixels is modeled by the statistics of the pixel states). Therefore what we model is the mean and the variability of the observations at the pixels that have been labeled by the same states. For example, if we take as observation vectors the 3 color components of the pixels, state labeling tends to segment the image into large regions of similar color, and the modeling of the color components models the variations of colors within each type of region. If the components of the observation vector are selected to be approximately independent, the distribution of each component can be independently modeled with a histogram (non-parametric discrete modeling of observations), with a normal distribution, or with a mixture of gaussians (continuous observation densities).

For color images, we have used normal distributions to model each of the three color components, and taken the observation probability of a pixel as the product of the probabilities of observations of the color components. This assumes that the color components are independent. Even as (R, G, B) components are more interdependent than components in color spaces that use luminance, hue and saturation, we have not observed qualitative improvements in segmentation when using the (Y, U, V) components of JPEG and MPEG compression instead of (R, G, B) components.

8 Training

We now address the problem of learning the model of a given image. In order to apply the Viterbi algorithm presented in Section 6, the quantities $P(q_{u,v}|q_{u,v-1}, q_{u-1,v})$ and $P(\mathbf{o}_{u,v}|q_{u,v})$ must be known before a meaningful state assignment for each pixel can be obtained. The first quantities represent the transition matrix of the model: given that the left pixel is in state $q_{u,v-1}$ and the top pixel in state $q_{u-1,v}$ what is the probability for the present pixel to be in state $q_{u,v}$ from 1 to N. The second quantities are the observation probabilities, that we assume in this presentation to be modeled by the product of the probabilities of the component of the observations, each modeled by a normal distribution. Assuming we can run the Viterbi algorithm to obtain a state assignment, then these probabilities can be easily computed, by scanning the image, and (1) tallying and normalizing the different types of state transitions between pixels to obtain the state transition matrix, and (2) averaging the values of observation components and their squares for all pixels with the same state labels to estimate the mean and variance of

observation components for pixels of identical states. This is a chicken-and-egg problem in the sense that we need these distributions to compute the Viterbi algorithm, and we need the Viterbi algorithm to compute the distributions.

This problem is solved by putting the Viterbi algorithm in an iteration loop. The iteration starts with random transition probabilities in the state transition matrix, while the gaussians of the observation component probabilities can be initially selected to overlap and cover the range of possible observations. As Viterbi segmentations and probability estimations are repeated alternately, the joint probability of the configuration of observations and states increases (although the general increase is not guaranteed to be locally monotonic as with a Baum-Welch procedure) as the system consistently converges to a stable solution. The exit criterion that we use is the maximum variation of the parameters of the model from step to step. Typically, little qualitative improvement can be observed in the image segmentation after 10 iterations. The results are a statistical model of the image, a segmentation of the image, and an evaluation of the probability of observing the image given the model. This technique is called a segmental k-means procedure [21]. It is popular in speech recognition as a much faster model training alternative to the Baum-Welch procedure, with comparable quality of results. In our case, we do not have an extension of the so-called forward-backward procedure for Markov meshes, so we cannot use a Baum-Welch approach.

9 Color segmentation with this model

One of the results of the training procedure described above is an optimal labeling of the image. Once a labeling is obtained, one may want to show the segmented image with the most likely color for each labeled pixel. For each state label, the most likely color is the color whose components are the most likely, because we have assumed that the components are independent. The distribution of each component within each state is modeled by a normal distribution. The most likely color component in each state is the mean of its normal distribution in this state. The most likely color is then obtained by combining these components. Fig. 4 shows a 64 $\times$ Lenna image (a), and a segmentation obtained using 10 states (b). Note that the number of states is the only specified parameter of this segmentation.

9.1 Distances between Images

In videos, successive frames of the same scene are typically views of the same objects whose images are shifted in the frames due to the camera's panning action, and we would like to assess that despite very different local frame pixel values after large camera pans, such frames are still similar. Statistical models of images characterize images, and allow computations of distances, yet are relatively insensitive to translation.

A distance measure can be computed as follows. We construct a probabilistic model λ_1 of one image I_1, and to find the distance to a new image I_2 we ask: what is the probability that N_2 observations $\mathbf{O}$ on the new image I_2 could have been produced by the probabilistic model λ_1 of the first image? Such a probability is generally quite small and of the form $P(I_2|\lambda_1 = 10^{-D}$, with D larger for more different images. Therefore D, which is the negative log of the probability,

$-\log P(I_2|\lambda_1)$, is an intuitive distance measure. To factor out the dependency of the total probability of the observation sequence on the number of observation, we normalize this measure by dividing by the number of observations N_2. In addition, this measure applied between an image and itself is not zero with this definition, so in most applications a measure shifted by the image distance to itself is preferable. Hence we obtain

$$D(I_1, I_2) = \frac{1}{N_2} \log P(I_2|\lambda_1) - \frac{1}{N_1} \log P(I_1|\lambda_1) \tag{11}$$

This distance measure requires the calculation of the model λ_1 of the first image only, but it is not symmetrical. When time to compute a model for the second image is available, one will prefer to compute a symmetrical distance measure D_s as the half sum of $D(I_1, I_2)$ and $D(I_2, I_1)$

$$D_s(I_1, I_2) = \frac{1}{2}[\frac{\log P(I_2|\lambda_1) - \log P(I_2|\lambda_2)}{N_2} + \frac{\log P(I_1|\lambda_2) - \log P(I_1|\lambda_1)}{N_1}] \tag{12}$$

Other interpretations of this expression exist in terms of model dissimilarity and cross-entropy of models ([21], p. 365)

9.2 Experiments with Distances

An example of distance calculation is shown in Fig. 5. We used the Lenna and Mandrill images to generate a composite third image in which a piece of the Lenna image image is replaced by a piece of the Mandrill image. We obtain an "image triangle" and compute the distances between its vertices using Eq. 12.

We compute distances between the images by obtaining a Markov model λ_i for each of the images, then by computing the probabilities of observing all the pixels of image I_j assuming that they were produced by model λ_i. The parts of the composite image that contain parts of Lenna's image are well predicted by Lenna's model, so they contribute to producing a larger joint probability and a smaller distance between Lenna and the composite image. We find a distance of 1.2. Similarly, the distance between the Mandrill image and the composite image is small, 2.0. On the other hand the Lenna and Mandrill image don't have much in common, and we find a distance of 5.9. Note that the triangle inequality is not satisfied in this example, and therefore our distance measure is not a metric. People try to find distance measures that are metrics because it allows for more efficient search, by the following reasoning: if I have an item A and am looking for similar ones in a database, if I find an item B that is *not* similar at all to A, then I don't need to check any of the items that are similar to B, because those other items, being close to B, will be unrelated to A. This reasoning is not necessarily desirable for image databases. If I look for images of cows, I would not want to miss the images of a cow that also contains the farmer, just because in the database there is a picture of the farmer and a picture of a cow, and the farmer does not look like his cow. And searches for Lenna should not fail as soon a monkey walks into the room. Therefore there is an incompatibility between the need for efficient search and the need for searches that are sensitive to partial matches and semantic content.

Another verification of the merit of this method can be obtained by taking an image, and scrambling it by swapping two pieces of the image. Example of images after 1, 5 and 10 scram-

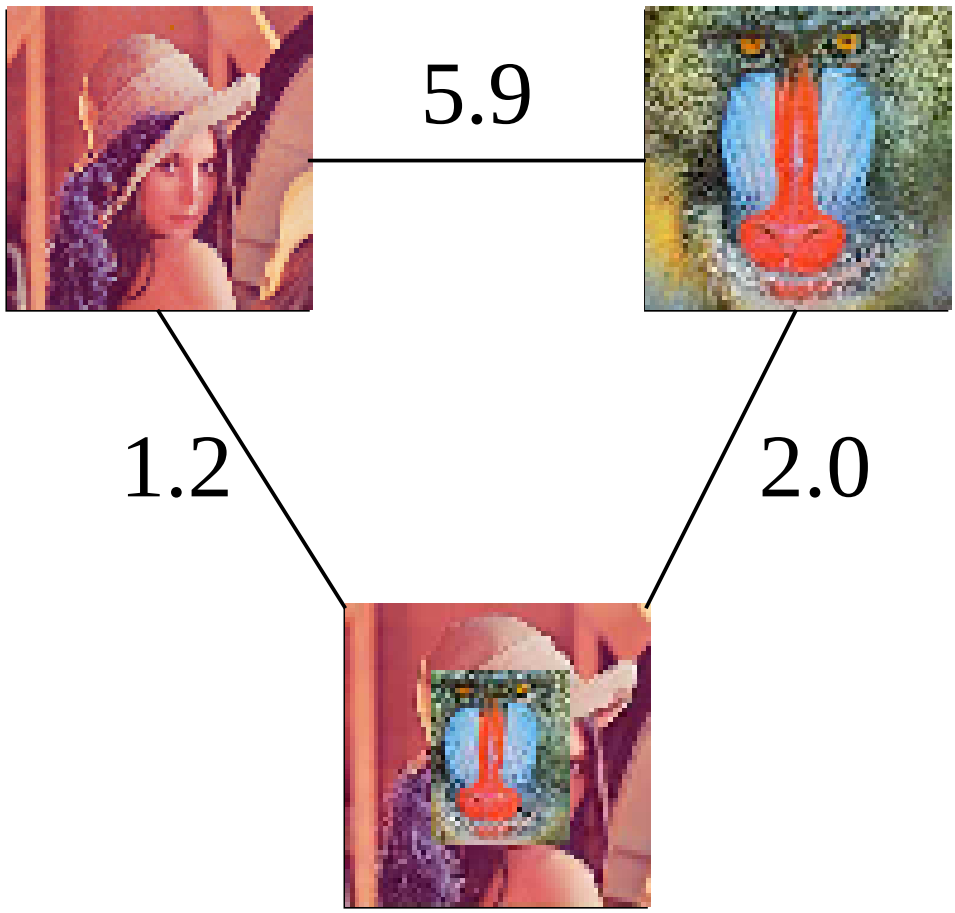
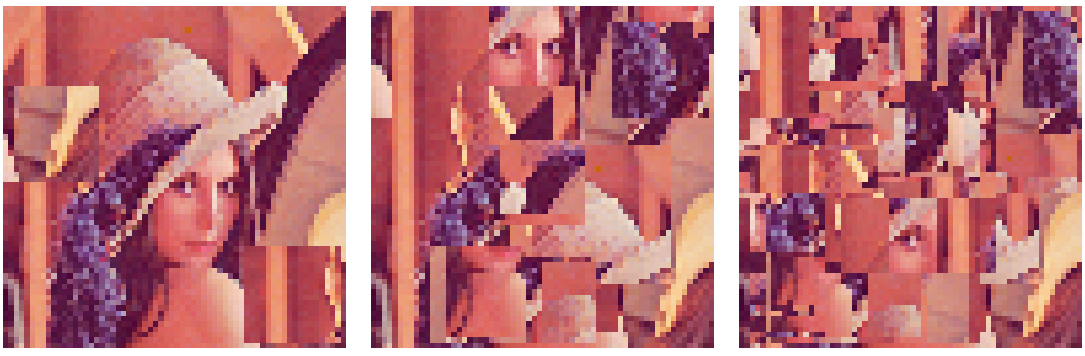


Figure 5: Distance measures between HMM models for 2 images and a composite image.



(a) (b) (c)

Figure 6: Increasingly scrambled images obtained by randomly swapping 1, 5 and 10 square regions.

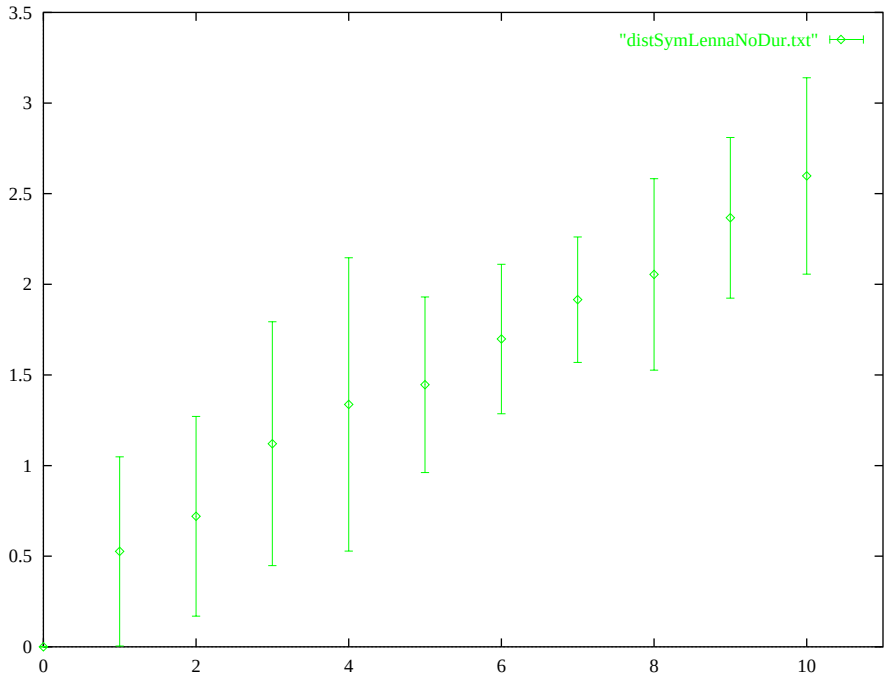


Figure 7: Distances between Lenna picture and increasingly scrambled pictures obtained by swappings of square regions. Abscissas show number of swapping, and ordinates show distances to original image. Distances are averages over 10 trials, and error bars are $\pm$ standard deviations. Some random swappings may reverse the scrambling of a previous swapping.

blings are shown in Fig. 6. As scrambling increases, the distance from the original image to the scrambled image increases as the image gets more scrambled, as shown in the plot of Fig. 7. This is an intuitive result, which would be difficult to obtain with distance measures which rely on feature matchings.

10 Semi-Markov Model with Explicit State Dimension Probabilities

In speech recognition, the probability of staying in the same state obtained by standard Markov modeling often lacks flexibility, and explicitly modeling "state durations" in analytic form was shown to improve performance [21]. The model then does not follow a Markov model as long as the model stays in the same state, and is called a semi-Markov model. For a Markov mesh modeling applied to images, similar limitations of the standard model can be observed when trying to model the width and height of regions of contiguous pixels of identical states with a Markov mesh model.

With a state transition matrix $P(q_{u,v}|q_{u,v-1}, q_{u-1,v})$, or, borrowing the simpler notation of Rabiner [21], a_{ijk} (where k is the value taken by the state $q_{u,v}$ of pixel (u, v), and i and j are the states at left and top pixels respectively), the probability of staying in the same state k horizontally along a row is

$$P_k(d_H) = (\sum_{j=1}^{N} a_{kjk})^{d_H - 1}(1 - \sum_{j=1}^{N} a_{kjk})$$

and the probability for pixels of same state i to be connected vertically is

$$P_i(d_V) = (\sum_{i=1}^{N} a_{ikk})^{d_V - 1}(1 - \sum_{i=1}^{N} a_{ikk})$$

This type of distribution is exponential. It is monotonically decreasing with increasing region size, either horizontally or vertically. Therefore small blobs are modeled as being more probable than larger blobs. Instead, for a model that is trained on an image with redish regions that are mostly 10 pixel wide and 20 pixel high, we would like the model of probability distribution of "state dimensions" to have a maximum at around 10 pixels for the horizontal dimension of contiguous red pixels, and around 20 pixels for the vertical dimension.

For modeling durations, the Gamma distribution was found to produce a good fit to observed duration histograms [4]. It assigns zero probabilities to negative values of the variable, which is very desirable for modeling durations or dimensions. This distribution is of the form

$$p(x) = \frac{\alpha^p}{\Gamma(p)} \mathrm{e}^{-\alpha x} x^{p-1} \tag{13}$$

The two free parameters of the distribution are α and p. For this application, they are positive, as seen from their estimation expressions below. The parameter α controls the vertical scaling of the distribution, while p controls whether there is a peak or not and where the peak is located. For $p < 1$, a monotonically decreasing distribution is modeled, otherwise a peak occurs at $\frac{p-1}{\alpha}$. The curve shape is less skewed and more gaussian-like as the peak is located further

from the origin. The two parameters of the Gamma distribution can be simply estimated from the mean $\overline{x}$ and variance $\overline{x^2}$ of the data x by

$$\alpha = \frac{\overline{x}}{\overline{x^2}}, \quad p = \frac{\overline{x}^2}{\overline{x^2}}$$

Burshtein proposed heuristic expressions for forcing the state durations to approximately follow a Gamma distribution in a speech recognition system [4]. However, obtaining an exact match is straightforward enough, therefore this is the technique preferred here. The cumulative probability corresponding to the Gamma distribution is called an incomplete Gamma function [20]. Its complement to 1, $P_k(d)$, represents, for a state k, the probability that the region dimension D will be more than a given dimension d. If we were in a state $q_{u,v-1} = k$ at the previous pixel of the same row and have been in this state for a distance of $d-1$ pixels, the probability of still being in the state k at pixel (u, v) is the probability that the region's horizontal dimension D will be larger than d, given the fact that it is already larger than $d-1$. Therefore

$$P_{\text{stay}}(k, d) = \frac{P_k(d)}{P_k(d-1)} \tag{14}$$

The only other possibility is that we leave state k when we go to the next pixel, therefore the expression for the probability of leaving state k is the complement to 1 of the previous probability.

We can apply this technique to model region shapes in many ways. In our current implementation, we model both the horizontal distance (in pixels) from the top left pixel of a state region to its right edges, and the vertical distance from the top left pixel to its bottom edges. To train this model within the segmental k-means framework described above, as we scan an image that has been segmented into state regions by the Viterbi algorithm described above, we compute horizontal and vertical distances for each pixel using distance information passed from neighbors. If the current pixel of state k is connected to a top or right neighbor of the same state, the information about the location of the top left pixel for that region is updated from these neighbors to the current pixel, and the vertical and horizontal distances from that top left pixel are also updated. The top left pixel of a new region of state k is a pixel of state k that is not connected to a top pixel of state k or a left pixel of state k. If the current pixel of state k has a left neighbor of state i different from k, then we went off a right edge for a state region i. We read the horizontal distance for that neighbor and use it to build up the sums used to obtain its mean and variance in regions of state i. From such means and variances, we can compute gamma distributions for these quantities, and compute the probability of staying in a state or leaving it as described above.

We define the following notations:

- S_i^H probability of staying in state i horizontally, i.e. probability that the state of current pixel be the same as state i of its left neighbor.
- L_i^H probability of leaving state i horizontally, i.e. probability that current pixel be in a state k different from state i: $L_i^H = 1 - S_i^H$
- S_j^V probability of staying in state j vertically, i.e. probability that the state of current pixel be the same as state j of its top neighbor.

- L_j^V probability of leaving state j vertically, i.e. probability that current pixel be in a state k different from state j: $L_j^V = 1 - S_j^H$

Each of these probabilities depends on state dimensions d^H or d^V, and is computed by Eq. 14 or its complement to 1. We construct tables for each state to provide these probabilities for horizontal and vertical state dimensions from 1 to the width and heigth of the image.

Assume that we have obtained by segmental k-means training the transition matrix a_{ijk} that tabulates the probability of having the current pixel (u, v) of being in state k when the left pixel $(u, v-1)$ is in state j and the top pixel $(u-1, v)$ is in state j. We wish to compute a new transition matrix a'_{ijk} that accounts for the modeling of dimensions of state regions. If k is equal to i or j (or both if i and j are equal) then the current pixel is still in the same state and we ignore the inadequate exponential distribution that the transition matrix would provide for the probability or staying in the same state, replacing it by a combination of probabilities of seeing the observed region dimensions in the horizontal and vertical directions. If however the state k of the current pixel is different from both i and j, then the information provided by the transition matrix is adequate, but must be combined with the probabilities of leaving the regions of the neighbor pixels in a consistent way so that the probabilities in the third dimension of the resulting matrix a_{ijk} add up to 1 (since for given top and left pixel states, the current pixel must be in one of the state $k = 1, \ldots, N$.) Such a combination is provided here:

1. $i = j$

   (a) $k = i \ (\Rightarrow k = j)$

   $$a'_{iii} = \frac{S_i^H S_i^V}{S_i^H S_i^V + L_i^H L_i^V}$$

   (b) $k \neq i \ (\Rightarrow k \neq j)$

   $$a'_{iik} = \frac{L_i^H L_i^V}{S_i^H S_i^V + L_i^H L_i^V} \frac{a_{iik}}{(1 - a_{iii})}$$

2. $i \neq j$

   (a) $k = i \ (\Rightarrow k \neq j)$

   $$a'_{iji} = \frac{S_i^H L_j^V}{S_i^H L_j^V + S_i^H L_j^V + L_i^H L_j^V}$$

   (b) $k \neq i, \ k = j)$

   $$a'_{ijj} = \frac{L_i^H S_j^V}{S_i^H L_j^V + S_i^H L_j^V + L_i^H L_j^V}$$

   (c) $k \neq i, \ k \neq j)$

   $$a'_{ijk} = \frac{L_i^H L_j^V}{S_i^H L_j^V + S_i^H L_j^V + L_i^H L_j^V} \frac{a_{ijk}}{(1 - a_{iij} - a_{ijj})}$$

10.1 Comparison of Field Realizations

The improvement of image modeling provided by the explicit modeling of region dimensions can be verified by comparing realizations of Markov fields obtained by the plain Markov model and by the semi Markov model.

We obtain these realizations by generating random colors from top left to bottom right along a raster scan. States are produced to follow the transition probabilities a'_{ijk} of the model trained on images, with decisions to leave regions produced by Gamma deviates. States are mapped to colors whose components are the means of the normal distributions that model the observations. In Fig. 8, we show realizations for models trained with three images, an image with ellipses of different colors and aspect ratios, the Lenna image, and the Mandrill image (left column). The fields of the middle column are obtained with the plain Markov model, and the fields of the right column are obtained with the semi-Markov model approach. Note that the region sizes, aspect ratios and contiguities are much closer to those of the regions of the original images with the semi-Markov model. A drawback of causal modeling in general is that context can be only verified from previously generated pixels in the raster scan, so there are strong diagonal directional effects for large regions. For a comparison of realizations of MRF models and causal Markov mesh models for mostly binary images, refer to [11].

Although large regions seem to be better modeled by the semi Markov model than by the standard model, we do not notice a clear advantage of the improved model in image classification using the image distances described above. For example, the diagram of distances for increasingly scrambled images of Fig. 6 is almost identical for the two models. One would have expected the semi Markov model to respond to the chopping of the larger regions by the scrambling process by producing larger distance increases than the standard model. More investigations are required.

11 Implementation Issues

Computation time for the Viterbi algorithm is proportional to $O(U \times V \times (N^3 + N \times D)$, where U and V are the height and width of the image, N is the specified number of states, and D is the number of components of the observation vectors at each pixels (3 for color images, 64 for DCT coefficients). The first term in the parenthesis of the computation cost is due to the triple loop of the algorithm in which for each pixel, each of the N components of δ (Eq. 10) must find the maximum of a term that includes one of the N components of the δ of the left pixel and one of the N components of the δ of the top pixel. The second term is due to the computation of the observation probabilities of each pixel for each possible state, $P(\mathbf{o}_{u,v}|q_{u,v})$. This second term may become of the same order as the first term if we consider large observation vectors (and smaller images), as when using DCT coefficients as observation vector components in order to find Markov models of JPEG images.

All the computations in the Viterbi algorithm are performed using the logarithmic forms of the proposed expressions. The logarithms of the N^3 elements of the transition matrix a_{ijk} and of the $U \times V \times N$ probabilities $P(\mathbf{o}_{u,v}|q_{u,v})$ at each Viterbi iteration in the segmental k-means algorithm are precomputed.

Each Viterbi iteration takes around 4 seconds for a 64×64 24 bit color image on an Ultra 1 Sun workstation, and a full HMM and an image segmentation are obtained in around 40 seconds.

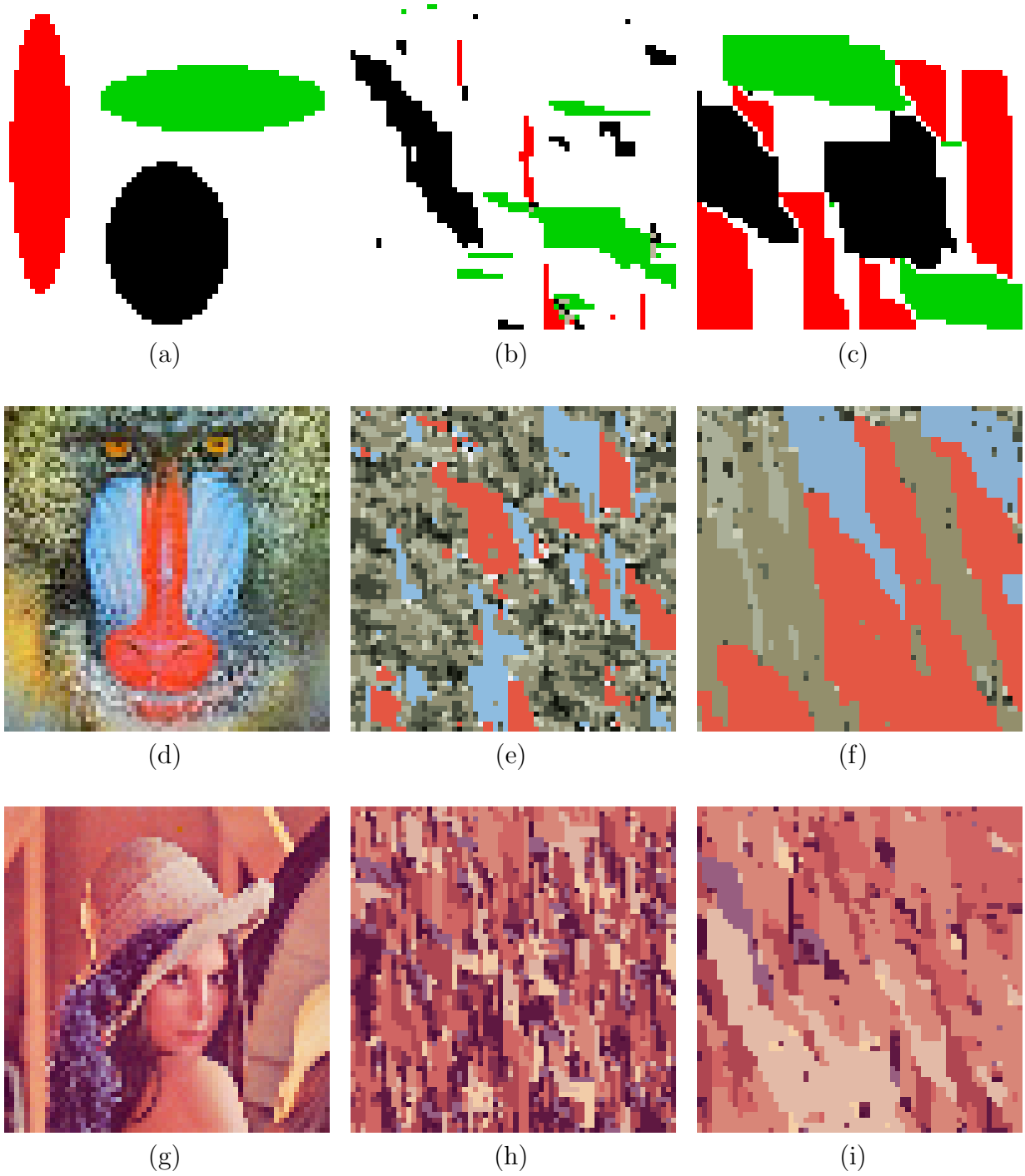


Figure 8: Realizations of Markov mesh models trained on images of the left column. Middle column: Standard HMM. Right column: HMM with state dimension modeling.

The processing of 512×512 24 bit color images would be prohibitive in time (more than 40 minutes) and memory requirements (large precomputed tables). The processing of JPEG images without decompression is more attractive and is being investigated.

Computation time is significantly longer (presently by a factor of 4) when the semi Markov model is used. The reason is that the expressions for the N^3 elements a'_{ijk} are not precomputed because they depend not only on states i, j, and k, but also on the dimensions of regions, which can be as large as the image. Required tables would be very large ($U \times V \times N^3$).

Once Markov models of individual images are obtained, the computational cost for obtaining distance measures between images is small, $O(U \times V \times D)$.

12 Conclusions and Comments

The main contribution of this paper is the decription of a training method for acquiring statistical models of images using a hidden second order Markov mesh model. This model encodes the contiguities of image regions of similar properties, and the distributions of the pixel attributes within each region. The method also provides an optimal segmentation of images when the number of labels to be used is specified. Distances between images can be computed as distances between statistical models of these images. We showed using composite images that these distances are very sensitive to common image content without the need for feature matching. We are currently extending the model to use different states for different regions, as a 2D extension of the so called left-right HMMs, in order to provide improved object recognition capability. Applications being investigated are classification in image data bases, image restoration in MPEG video streams, robust cut and transition detection in videos, change detection in streams from static cameras, and object recognition. The range of potential applications include the domains in which MRF models and pseudo 2-D HMMs have been applied, for example segmentation, data compression and image restoration for MRFs, and character recognition and face recognition for pseudo 2-D HMMs.

Appendix

The defining property for the second-order Markov mesh is

$$P(q_{u,v}|\{Q_{U,v-1} \cup Q_{u-1,V}\}) = P(q_{u,v}|q_{u,v-1}, q_{u-1,v}) \tag{15}$$

with the following boundary conditions

$$\begin{aligned} P(q_{u,v}|q_{u,v-1}, q_{u-1,v}) &= P(q_{1,1}) \text{ if } u = v = 1 \\ &= P(q_{u,1}|q_{u-1,1}) \text{ if } u > v,\ v = 1 \\ &= P(q_{1,v}|q_{1,v-1}) \text{ if } v > u,\ u = 1 \end{aligned}$$

In other words, the probability of having a given state at pixel (u, v) conditional to the states in all the pixels on the rows above row u and all the remaining pixels to the left of column v is

simply equal to the probability of having that state at pixel (u, v) conditional to the state of the pixel above (u, v) and to the state of the pixel to the left of (u, v).

Note that this definition 15 can be applied to any subimage of the image $I_{U,V}$, provided that the subimage contains pixel (u, v). For example, we can write

$$P(q_{u,v}|Q_{u,v} - \{q_{u,v}\}) = P(q_{u,v}|q_{u,v-1}, q_{u-1,v}) \tag{16}$$

To see this, we simply apply the definition 15 to the subimage that extends from $(1, 1)$ to (u, v). In this subimage, the configuration $\{Q_{U,v-1} \cup Q_{u-1,V}\}$ is identical to the configuration $Q_{u,v} - \{q_{u,v}\}$.

We want to prove that the probability of seeing the configuration of states $Q_{u,v}$ over the rectangle $R_{u,v}$ is equal to the product of the probabilities of each pixel in the rectangle, each probability being conditional to the states of the top and left neighbor pixels.

$$P(Q_{u,v}) = \prod_{u'=1}^{u} \prod_{v'=1}^{v} P(q_{u',v'}|q_{u'-1,v'}, q_{u',v'-1})$$

For $u = 2, v = 1$, the definition of a Markov mesh along with the boundary conditions yields

$$P(Q_{2,1}) \equiv P(q_{1,1}, q_{2,1}) = P(q_{2,1}|q_{1,1})$$

and for the first row and first column of pixels, the theorem amounts to the factorization of a 1D Markov chain. We assume that the theorem holds for $Q_{u,1}$ and show that it holds for $Q_{u+1,1}$.

$$P(Q_{u+1,1}) \equiv P(q_{1,1}, q_{2,1}, , q_{3,1}, \ldots, q_{u,1}, q_{u+1,1})$$

$$\begin{aligned} P(Q_{u+1,1}) &= P(q_{u+1,1}|Q_{u,1})P(Q_{u,1}) \\ &= P(q_{u+1,1}|q_{u,1})P(Q_{u,1}) \\ &= P(q_{u+1,1}|q_{u,1})P(q_{u,1}|q_{u-1,1}) \ldots \\ &\quad \ldots P(q_{u-1,1}|q_{u-2,1}) \ldots P(q_{1,1}) \end{aligned}$$

Following [1], we assume that the theorem holds for $Q_{u,v}$, and show that it holds for $Q_{u+1,v}$. Induction for $Q_{u+1,v}$ is shown in the same way because of the symmetry of the expressions.

$$\begin{aligned} P(Q_{u+1,v}) &= P(Q_{u,v})P(q_{u+1,v}, q_{u+1,v-1}, \ldots, q_{u+1,1}|Q_{u,v}) \\ &= P(Q_{u,v})P(q_{u+1,v}|q_{u+1,v-1}, \ldots, q_{u+1,1}, Q_{u,v}) \ldots \\ &\quad \ldots P(q_{u+1,v-1}|q_{u+1,v-2}, \ldots, q_{u+1,1}, Q_{u,v}) \ldots \\ &\quad \ldots P(q_{u+1,1}|Q_{u,v}) \end{aligned}$$

Using the Markov mesh defining property for each term of the factorization yields

$$
\begin{aligned}
P(Q_{u+1,v}) &= P(Q_{u,v})P(q_{u+1,v}|q_{u+1,v-1}, q_{u,v}) \ldots \\
&\quad \ldots P(q_{u+1,v-1}|q_{u+1,v-2}, q_{u,v-1}) \ldots \\
&\quad \ldots P(q_{u+1,1}|q_{u+1,1}, q_{u,1}) \\
&= P(Q_{u,v}) \prod_{v'=2}^{v'=v} P(q_{u+1,v'}|q_{u+1,v'-1}, q_{u,v'}) \\
&= \prod_{u'=1}^{u'=u+1} \prod_{v'=1}^{v'=v} P(q_{u'+1,v'}|q_{u'+1,v'-1}, q_{u',v'})
\end{aligned}
$$

Acknowledgements

The support of this research by the Department of Defense under contract MDA 9049-6C-1250 is gratefully acknowledged.

References

[1] Abend, K., Harley, T.J, and Kanal, L.N., "Classification of Binary Random Patterns", IEEE Trans. Information Theory, IT-11, pp. 538–544, 1965.

[2] Ballard, D.H., and Brown, C.M., *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982, pp. 137–143.

[3] Besag, J., "Spatial Interaction and the Statistical Analysis of Lattice Systems (with Discussion)", J. Royal Statistical Society, series B, vol. 34, pp. 75–83, 1972.

[4] Burshtein, D., "Robust Parametric Modeling of Durations in Hidden Markov Models", IEEE, 1995.

[5] Chow, C.K., "A Recognition Method using Neighbor Dependence", IRE Trans. on Electronic Computers, vol. 11, pp. 683–690, 1962.

[6] Devijver, P.A., "Probabilistic Labeling in a Hidden Second Order Markov Mesh", Pattern Recognition in Practice, II, E.S. Gelsema and L.N. Kanal eds., North-Holland, Amsterdam, 1986.

[7] Devijver, P.A., "Real-Time Modeling of Image Sequences", Proc. 10th International Conference on Pattern Recognition, 1990, pp. 194-199.

[8] Devijver, P.A., and Dekesel, M.M., "Learning the Parameters of a Hidden Markov Random Field Model: A Simple Example", Pattern Recognition Theory and Practice, P.A. Devijver and J. Kittler, eds, 1987.

[9] Fassnacht, C., and Devijver, P.A., "Image Segmentation with a Propagator Markov Mesh Model", Proc. 12th International Conference on Pattern Recognition, Jerusalem, 1994.

[10] Geman, S., and Geman, D., "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-6, no. 6, pp. 721–741, 1984.

[11] Gray, A.J., Kay, J.W., and Titterington, D.M., "An Empirical Study of the Simulation of Various Models Used for Images", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-16, no. 5, pp. 507–513, 1994.

[12] Kashyap, R.L., and Chellappa, R., "Estimation and Choice of Neighbors in Spatial Interaction Models of Images", IEEE Trans. Information Theory, vol. TI-29, pp. 60–72, 1983.

[13] Kuo, S-S. and Agazzi, O.E., "Keyword Spotting in Poorly Printed Documents using Pseudo 2-D Hidden Markov Models", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 16, no, 8, pp. 842–848, 1994.

[14] Haslett, J., "Maximum Likelihood Discriminant Analysis on the Plane using a Markovian Model of Spatial Context", Pattern Recognition, 18, pp. 287–296, 1985.

[15] Lacroix, V., "Pixel Labeling in a Second-Order Markov Mesh", Signal Processing 12, pp. 59–82, 1987.

[16] Li, S.Z., Markov Random Field Modeling in Computer Vision, Springer, 1995.

[17] Lin, H-C., Wang, L-L., and Yang, S-N., "Color Image Retrieval Based on Hidden Markov Models", IEEE Trans. Image Processing, vol. 6, no. 2, pp. 332–339, 1984.

[18] Park, H.-S., and Lee, S.-W., " An HMMRK-Based Statistical Approach for Off-line Handwritten Character Recognition", Proceedings of ICPR, pp. 320–324, 1996.

[19] Pickard, D.K., 'Unilateral Markov Fields", Advanced Applied Probability, 12, pp. 655–671, 1980.

[20] Press, W.H., Teukolsky, S.A., Vettering, W.T., and Flannery, B.P., *Numerical Recipes in C*, Second Edition, Cambridge University Press, 1992.

[21] Rabiner, L.R., and Juang, B.-H., "Fundamentals of Speech Processing", Prentice Hall, pp. 321–389, 1993.

[22] Rabiner, L.R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, 1989.

[23] Samaria, F., and Young, S., "HMM-based architecture for Face Identification", Image and Vision Computing, vol. 12, no. 8, 1994.

[24] Szeliski, R., "Baysian Modeling of Uncertainty in Low-Level Vision", Kluwer Academic Publishers, 1989.