INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa • University of Ottawa

A Novel Reduced-Complexity Approach to Hidden Markov Modeling of 2-D Processes with Application to Face Recognition

by

Hisham H. A. Othman

A thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Ottawa-Carleton Institute Electrical and Computer Engineering

School of Information Technology and Engineering

Faculty of Engineering, University of Ottawa

Ottawa, Canada, April, 2002

Acknowledgement

All thanks to God whom without his will, none would be accomplished. Full gratitude goes to my parents for their support and encouragement. Special thanks go to Dr. T. Aboulnasr for her supervision and guidance through out this work.

I would like also to thank Dr. M. Rashwan and Dr. M. Afify for introducing me to the subject in the first place.

Abstract

The 2-D Hidden Markov Model (HMM) is an extension of the traditional 1-D HMM that has shown distinctive efficiency in modeling 1-D signals. Unlike 1-D HMMs, 2-D HMMs are known for their prohibitively high complexity. This encouraged many researchers to work on alternatives such as Pseudo 2-D HMM and Embedded HMM for 2-D recognition applications to avoid the complexity problem. Those applications include, but are not limited to, Face Recognition, Optical Character Recognition, Face Detection, Image Retrieval, and Object Recognition. The Hidden Layer's complexity of a typical second-order 2-D HMM is normally in the order of (N^3). The term "Hidden Layer" refers to the computations of the probabilities of state transition and N is the number of states in the model.

In this thesis, a low complexity high performance 2-D Hidden Markov Model (HMM) is proposed and is applied to the problem of Face Recognition. The proposed model is a true 2-D HMM. The complexity of the *Hidden Layer* is brought down to the order of ($2N^2$) using a basic assumption of conditional independence between vertical and horizontal state transitions. This assumption allows replacing the 3-D state transition matrix with two 2-D transition matrices.

HMM complexity is always addressed in the literature from the Hidden Layer perspective, yet the complexity of the observation layer is not trivial. The mixtures of the proposed model are tied for lower observation layer complexity. The performance and the complexity of the proposed model with tied mixtures are investigated while applied to the problem of face recognition. The proposed face recognition system achieves

recognition rates up to 100% on the AT&T facial database with complexity that is comparable to that of 1-D HMM.

Table of Contents

List of figures

List of Tables

List of Acronyms

ASR	Automatic Speech Recognition
CHMM	Continuous-Density Hidden Markov Model
DCT	Discrete Cosine Transform
DHMM	Discrete-Density Hidden Markov Model
DWT	Discrete Wavelet Transform
EHMM	Embedded Hidden Markov Model
FFT	Fast Fourier Transform
FR	Face Recognition
HMM	Hidden Markov Model
IM	Independent- Mixture
JPEG	Joint Photographic Experts Group
KLT	Karhunen-Loeve Transform
LBG	Linde, Buzo and Gray Algorithm
MAP	Maximum A Posteriori Probability
ML	Maximum Likelihood
MMRF	Markov Mesh Random Field
MPEG	Moving Picture Expert Group
PDF	Probability Density Function
PHMM	Pseudo Hidden Markov Model
SCHMM	Semi-Continuous-Density Hidden Markov Model

Symbol	Definition
λ	The conventional 1-D HMM, $\lambda=\{\pi,A,B\}$.
χ	A state cluster in the Genonice HMM. χ also refers to the index of N_χ, which is the χ^{th} genone is the system that is associated with the state cluster χ.
$\mu_\chi^{(g)}$	The mean vector of the g^{th} Gaussian kernel in the χ^{th} genone N_χ.
$\Sigma_\chi^{(g)}$	The covariance matrix of the g^{th} Gaussian kernel in the χ^{th} genone N_χ.
$\lambda^{(k)}$	The set of parameters of the k^{th} superstate in the Embedded HMM, where $\lambda^{(k)}=\{\Pi_1,A_1,B_1\}$.
Π_0	The initial superstate distribution in the Embedded HMM, where $\Pi_0=\{\pi_{0,i}\}$.
$\pi_{0,i}$	The probability of the superstate $S_{0,i}$ to be the active state at the time 0 in the Embedded HMM.
Π_1	The initial state distribution in the Embedded HMM, where $\Pi_1=\{\pi_{1,i}\}$.
$\pi_{1,i}$	The probability of the state $S_{1,i}$ to be the active state at the time 0 in the Embedded HMM.
Π_H	The initial horizontal state probability in the proposed 2-D HMM, where $\Pi_H=\{\pi h_{m,n}\}$.
Π_V	The initial horizontal state probability in the proposed 2-D HMM, where $\Pi_V=\{\pi v_{m,n}\}$.
$\pi h_{m,n}$	The initial Horizontal state probability of $S_{m,n}$, i.e. the probability that state $S_{m,n}$ is active in the first column, in the proposed 2-D HMM.
π_i	Initial state probability in the conventional 1-D HMM.

$\Sigma_j^{(g)}$ The covariance matrix of the g^{th} multivariate Gaussian component in the observation PDF of the state S_j.

$\mu_j^{(g)}$ The mean of the g^{th} Gaussian component in the observation PDF of the state S_j.

$\mu_j^{(g)}$ The mean of the g^{th} multivariate Gaussian component in the observation PDF of the state S_j.

$\sigma_j^{(g)}$ The variance of the g^{th} Gaussian component in the observation PDF of the state S_j.

$\delta_{k,l}(m,n)$ State score in the proposed 2-D HMM; the likelihood of the past and the current observations maximized over the past state sequence terminating with $S_{m,n}$ at the observation block $O_{k,l}$.

$\zeta_{k,l}(m,n)$ The vertical predecessor of the state $S_{m,n}$ at the observation block $O_{k,l}$.

$\Psi_{k,l}(m,n)$ The horizontal predecessor of the state $S_{m,n}$ at the observation block $O_{k,l}$.

$\Sigma_{m,n}^{(g)}$ The covariance matrix of the g^{th} multivariate Gaussian component in the observation PDF of the state $S_{m,n}$.

$\mu_{m,n}^{(g)}$ The mean of the g^{th} multivariate Gaussian component in the observation PDF of the state $S_{m,n}$.

$\alpha_t(i)$ The forward variable; the joint probability that the partial observation sequence $O_1, O_2, \ldots, O_t$ occurs and that the state S_i is the current active state q_t given the model λ.

$\beta_t(i)$ The backward variable; the joint probability that the partial observation sequence $O_{t+1}, O_{t+2}, \ldots, O_T$ occurs and that the state S_i is the current active state q_t given the model λ.

$\gamma_t(i)$ The forward-backward variable; the probability of the state S_i to be the active state at the time t given observation sequence and model parameters.

$\delta_t(i)$ The state score in Viterbi algorithm; the observation likelihood maximized over the past state sequence terminating with S_i at time t.

$\xi_t(i,j)$ The joint probability of S_i and S_j to be active states at time t and $t+1$, respectively.

$\psi_t(j)$ The predecessor of state S_j at time t.

$\pi v_{m,n}$ The initial vertical state probability of $S_{m,n}$, i.e. the probability that state $S_{m,n}$ is active in the first row, in the proposed 2-D HMM.

$\mathbf{A}$ State transition matrix of the conventional 1-D HMM.

$\mathbf{A_0}$ The superstate transition matrix in the Embedded HMM, where $\mathbf{A_0}=\{a_{0,ij}\}$.

$a_{0,ij}$ The probability of the transition from superstate $S_{0,i}$ to superstate $S_{0,j}$ in the Embedded HMM.

$\mathbf{A_1}$ The state transition matrix in the Embedded HMM, where $\mathbf{A_1}=\{a_{1,ij}\}$.

$a_{1,ij}$ Tis the probability of the transition from state $S_{1,i}$ to state $S_{1,j}$ in the Embedded HMM.

a_{ij} The probabitity of state transition from state S_i to state S_j.

$\mathbf{B}$ The observation probability matrix of the conventional 1-D HMM.

$\mathbf{B_1}^{(k)}$ The observation probability matrix.

$b_j(O_t)$ The probability that the observation at time t has been generated by state S_j.

$b_{m,n}(O_{k,l})$ The probability that the observation block $O_{k,l}$ is generated by state $S_{m,n}$.

C^*_{HL} The number of multiplications in the hidden layer.

C^+_{HL} The number of additions in the hidden layer.

$c_j^{(g)}$ The weight of the g^{th} Gaussian component of the observation PDF of the state S_j.

$c_{m,n}^{(g)}$ The weight of the g^{th} Gaussian component of the observation PDF of the state $S_{m,n}$.

G_χ The number of Gaussian kernels in the χ^{th} genone N_χ.

G_s The number of Gaussian kernels in the PDF of the state S.

$h_{i,j;m,n}$ The probability of horizontal transition from state $S_{i,j}$ to $S_{m,n}$ state in the proposed 2-D HMM.

HN_p The horizontal dimension of the observation block/strip in pixels.

M The number of entries in the observation codebook in the Discrete HMM.

M_{pr} The average number of vertical predecessors per state.

N_{pr} The average numbers of vertical and horizontal predecessors per state

$m_{pr}(i,j)$ The number of vertical predecessors of state $S_{i,j}$.

N The number of states in the conventional HMM.

N_χ The χ^{th} genone in the system that is associated with the state cluster χ in the Genonic HMM, where $N_\chi=\{N_\chi^{(g)}: 1<g<G_s\}$.

$N_\chi^{(g)}$ The g^{th} Gaussian kernel in the χ^{th} genone N_χ.

N_0 The vertical number of states of the proposed HMM, and the number of vertical superstates for the 2-D PHMM and the Embedded HMM.

N_1 The horizontal number of states of the proposed HMM.

$N_1^{(k)}$ The number of horizontal states in the k^{th} superstate in the 2-D PHMM and the Embedded HMM.

N_{pr} The average number of horizontal predecessors per state.

$n_{pr}(i,j)$ The number of horizontal predecessors of state $S_{i,j}$.

N_t The total number of states in the model, excluding the number of the superstates of the 2-D PHMM and the Embedded HMM.

O A sequence of observations.

$O_{(k,l)}$ The rectangular set of observation blocks that includes all blocks starting with $O_{1,1}$ to $O_{k,l}$.

$O_{<k,l>}$ The set of observation blocks that includes all blocks starting with $O_{1,1}$ to $O_{k,l}$ excluding $O_{k,l}$.

$O_{k,l}$ The observation at the block located at the k^{th} row and l^{th} of an image in 2-D HMM.

O_t The t^{th} observation (in a 1-D observation sequence).

Q A sequence of active states.

$q_{k,l}$ The active state at the k^{th} row and l^{th} column of an image in 2-D HMM.

q_t The t^{th} active state in the conventional 1-D HMM.

S_i The i^{th} state in the conventional 1-D HMM.

$S_{i,j}$ The state at the i^{th} row and l^{th} column in the proposed 2-D HMM state constellation.

T_O The vertical number of observation blocks/strips.

T_I The horizontal number of observation blocks.

$v_{i,j;m,n}$ The probability of vertical transition from state $S_{i,j}$ to $S_{m,n}$ state in the proposed 2-D HMM.

V_k The k^{th} entry in the observation codebook in the Discrete HMM.

VN_p The vertical dimension of the observation block/strip in pixels.

Chapter 1 Introduction

Hidden Markov Models (HMM) are an efficient tool for stochastic process modeling. HMMs are widely used in many applications involving segmentation, recognition or synthesis of random signals.

HMM is also well known for its dominant role in speech applications, where it is extensively exploited in its traditional 1-D form. Efficient algorithms have been developed for HMM based on the 1-D nature of the targeted signal. Due to the emergence of digitized documents, images and video applications, the need for efficient 2-D segmentation and recognition arose significantly. The Markov Mesh Random Field (MMRF) provides a real 2-D substitution to the Markov chain, which is the base of the traditional 1-D HMM. An HMM that is based on Markov Mesh Random Field is a true 2-D HMM that is capable of handling 2-D signals without merging the two dimensions or extracting a 1-D representation of the spatial features. However, the computational burden of the MMRF-based HMM, is excessively high. This has helped the research grow in the direction of the alternative HMMs for 2-D signals.

1.1 Motivation and Problem Formulation

Given that considerable research has gone into accommodating 2-D signals in the traditional 1-D HMM and Pseudo 2-D HMMs on one hand, and the MMRF-HMM on the other hand, work needs to be done to fill the gap in between as depicted in Figure 1-1. This work aims to develop a true 2-D HMM that is robust in modeling yet low in complexity for 2-D applications. Such a model would bring together the features in both

dimensions to be modeled as a 2-D Markov source while avoiding the cost of high complexity associated with similar models.

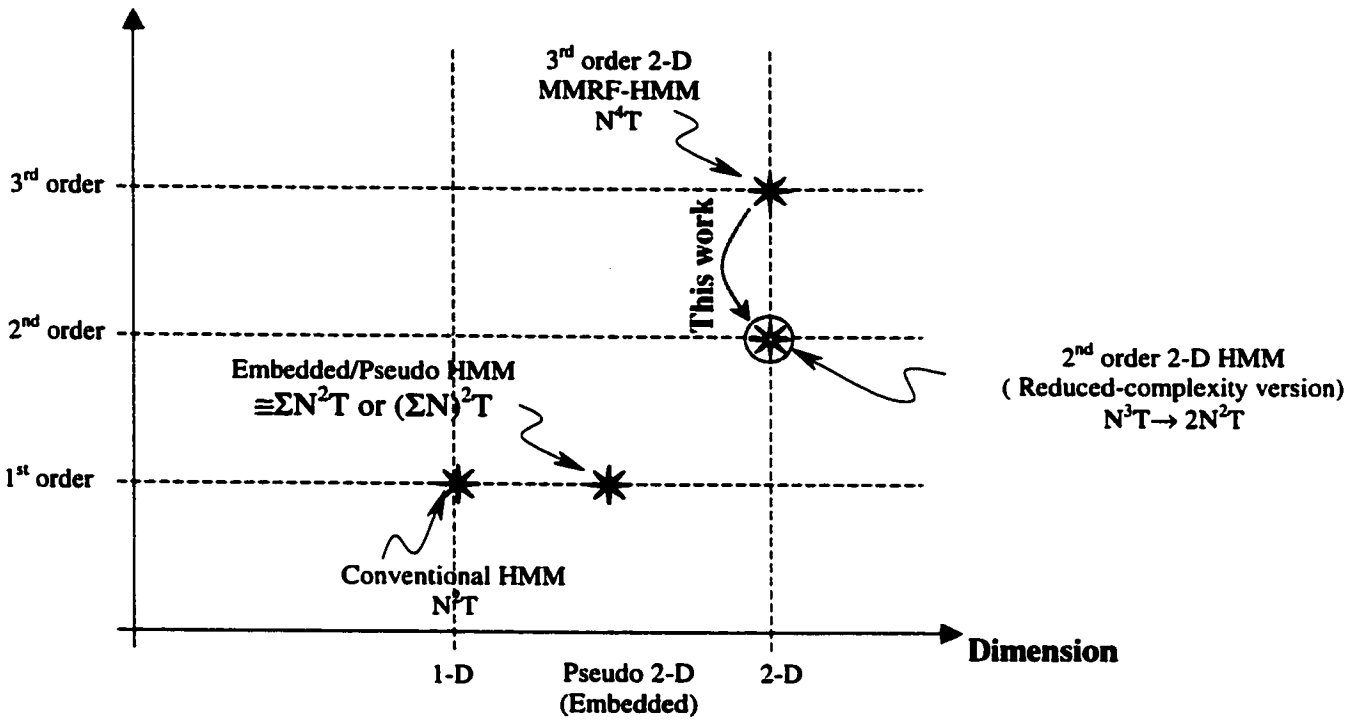


Figure 1-1. Locating the existing models on a grid of dimensionality versus order. N is the total number of states in the model and T is the total number of observation blocks in the image.

Although the aim of thesis is to propose a general model, i.e. one that is not customized to fit a certain application, a practical application is adopted to investigate the model robustness and relative complexity with respect to that of the existing 1-D, Pseudo 2-D and 2-D models. For the sake of comparison, the proposed model will be applied to the problem of Face Recognition. Face Recognition is an active area of research for which most of the existing HMMs have been tested and their performance is clearly reported in the literature. No special arrangement has been made to exploit the nature of the Face Recognition application except for the common properties that are shared by a wide set of similar applications, e.g. assuming natural images. The proposed model is a

good candidate for future work in other 2-D applications such as Document Segmentation, Optical Character Recognition, Image Retrieval, Image Database Indexing, Video Segmentation, Gesture Recognition, ... etc.

1.2 Thesis Organization

Chapter 1 presents an introduction to the general objective of this work, motivation and problem formulation while Chapter 2 includes an introduction to the Face Recognition problem and a brief survey of some of the various HMMs found in the literature. It also sheds some light on the principles and concepts of HMM concluding with a review of HMM-based Face recognition systems.

Chapter 3 presents the proposed model starting with the effect of the training data set size on a traditional 1-D HMM, then moving on to the actual proposed 2-D model, the assumptions on which its based, its performance and complexity. Chapter 3 also includes complexity-performance comparison with similar systems. Chapter 4 discusses the motivation and the benefits of tying the free parameters of the proposed model. Thesis conclusion and suggested future extensions to this work are presented in Chapter 5 along with some preliminary results in this direction.

Chapter 2 Review of Hidden Markov Models and HMM-based Face Recognition

2.1 Review of Hidden Markov Models (HMMs)

HMM has the capability of capturing statistical properties of a wide spectrum of non-deterministic signals including the class of quasi-stationary signals. Quasi-stationary signals are those signals whose statistical features are considered invariant on the short-term basis though they vary in the long term.

HMM-based approaches have a wide spectrum of applications in speech processing, video and image processing and audio-visual media. Figure 2-1 depicts some of these applications including automatic temporal video segmentation and recognition, (e.g., camera movement segmentation [1] and gesture recognition applications as in [2]).In these applications, the observations of HMM states are either vector-quantized luminance vector [3] or movement features [4]. HMM applications also include speech-lips synchronization for low-bit rate video coding [5] where the low-bit coding does not allow enough frame rate to adequately synchronize the lips' movement with the pronounced speech. HMM is used to analyze the speech and generate guiding information to synchronize the lips' frames. A similar application is text-to-visual speech synthesis for virtual reality applications [6], where the whole face is synthesized and designed to speak some pre-scripted sentences. HMM is also used for visual speech recognition, and lip reading [7][8].

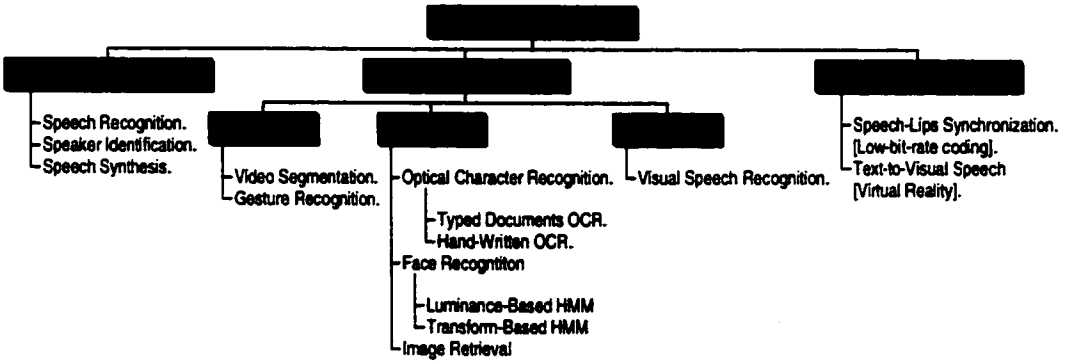


Figure 2-1. Applications of HMMs.

HMM was first introduced in late 1960s and early 1970s in mathematical journals. By mid-70s, the speech processing field made extensive use of it [9]. Today, HMM is the dominant tool for speech analysis, recognition, and synthesis. HMM has continued to gain popularity in speech as well as other media, as demonstrated above. In the following section, the fundamentals of the HMM approach are reviewed in some detail including the main structure and parameters.

2.1.1 One-dimensional Hidden Markov Model [9] (1-D HMM)

2.1.1.1 Structure and Parameters

HMM is mainly a chain of a finite number of states. Each state represents a stochastic process with a certain probability distribution, either discrete, continuous, or semi-continuous.

Only one state is active at any given time instant. The active state at that time instant is responsible for generating the observation corresponding to that instant. The generated observation obeys the probability function distribution of the emitting state. The term *observation* here is used to describe a part of the signal being modeled, e.g. a sample or a block of samples, while the term *time* is used to annotate the progressive dimension of the

signal, without loss of generality, under the assumption of that it is a temporal signal. For any other signal, the time dimension is replaced with the proper alternative dimension. The model as a whole can be seen as a multi-modal system and each state is the model representative of one of its modes.

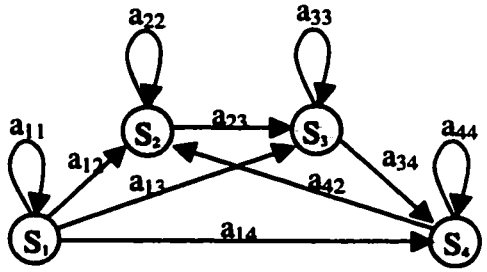


Figure 2-2. An Example of a Markov chain.

The possible state transitions are denoted by paths as shown in Figure 2-2. Each path is associated with a transition probability, a_{ij}, which expresses the probability of making transition from one state, S_i, to another state, S_j, in the direction defined by the path. We can consider the transition probability as the likelihood of the modeled process to change its mode from one to another. The transition probabilities among the states in the model are usually expressed in a matrix form. The matrix that contains all transition probabilities of a model is called *transition matrix, A*.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} \end{bmatrix},$$

where N is the number of states in the model and a_{ij} is the probabitity of state transition from state S_i to state S_j and is given by

$$a_{ij} = P[q_t = S_j | q_{t-1} = S_i] \qquad (2.1)$$

The main assumption here is that the probability for a state S_j to be the active state q_t at the time t is dependent only on q_{t-1} the active state at time $t-1$.

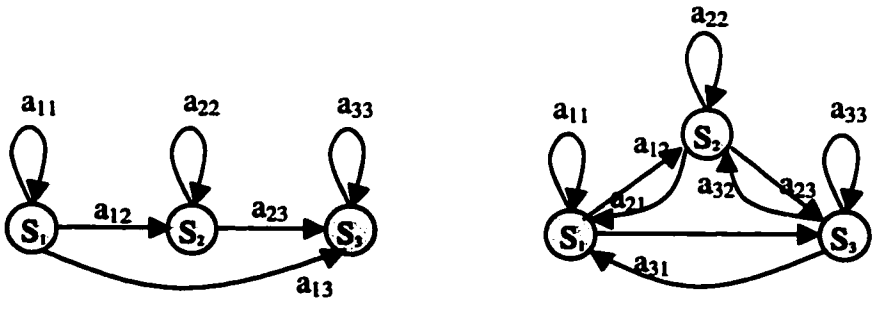
This model is referred to as *first-order HMM*. Should the current state depend on r past states, the model is said to be an r^{th}-order model whose transition matrix is $(r+1)^{th}$-dimensional.

The model's initial state probability, π_i, which defines the model behavior at the beginning of the observation sequence, is given by:

$$\pi_i = P[q_1 = S_i], \quad 1 \le i \le N \tag{2.2}$$

Back to the first-order HMM, among the most popular forms of the model are the *ergodic* HMM and the *left-to-right* HMM. An HMM is said to be ergodic if every state is reachable from every other state in a finite number of transitions. The fully-connected HMM whose Markov Chain (MC) is shown in Figure 2-3 b, is an example of the *ergodic* HMM in which every state is reachable from every other state in one transition. The left-to-right model is characterized by its transition matrix that has a zero lower-left triangle.

$$A_{Left-to-Right} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,N} \\ 0 & a_{22} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{N,N} \end{bmatrix}, \quad A_{Fully-Connected} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,N} \\ a_{21} & a_{22} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} \end{bmatrix} \tag{2.3}$$



(a) A left-to-right Markov chain. (b) A Fully-connected Markov chain.

Figure 2-3. 1-D Markov Chain examples.

HMMs are also classified according to the state probability distribution type. The simplest type is the discrete HMM whose states possess discrete probability distributions as functions of discrete observations. In this case, the observations are quantized to discrete values using a predefined finite-length codebook and mapped to its M entries. The probability of a state S_j generating an observation, O_t, is the probability of it generating the associate codebook entry V_k whose index is k. This probability is written as:

$$b_j(O_t) = P[V_k \mid q_t = S_j], \quad 1 \le j \le N, 1 \le k \le M \tag{2.4}$$

On the other hand, the most general HMM type is the continuous probability distribution HMM whose states have continuous probability density distributions which, in turn, are functions of continuous observation parameters. In most cases, the general continuous probability function is approximated by a weighted sum of finite number of Gaussian distributions. Thus, the probability of a state S_j to generate an observation O_t, is:

$$b_j(O_t) = \sum_{g=1}^{G} c_j^{(g)} G[O_t, \mu_j^{(g)}, \sigma_j^{(g)}], \quad 1 \le j \le N \tag{2.5}$$

where the observation O_t is a scalar quantity, $c_j^{(g)}$, $\mu_j^{(g)}$, and $\sigma_j^{(g)}$ are the weight, the mean , and the variance of the g^{th} Gaussian component in the observation PDF of the state S_j, respectively.

For vector observations, Equation (2.5) becomes:

$$b_j(\mathbf{O}_t) = \sum_{g=1}^{G} c_j^{(g)} G[\mathbf{O}_t, \boldsymbol{\mu}_j^{(g)}, \boldsymbol{\Sigma}_j^{(g)}], \quad 1 \le j \le N \tag{2.6}$$

where $c_j^{(g)}$, $\boldsymbol{\mu}_j^{(g)}$, and $\boldsymbol{\Sigma}_j^{(g)}$ are the weight, the mean vector, and the covariance matrix of the g^{th} Gaussian component in the observation PDF of the state S_j, respectively, and G_s is the number of Gaussian components.

Note that the weight of the mixture component must justify the constraint:

$$\sum_{g=0}^{G} c_j^{(g)} = 1, \quad 1 \le j \le N$$
$$c_j^{(g)} \ge 0, \quad 1 \le j \le N, \quad 1 \le g \le G \tag{2.7}$$

so that the area under the probability distribution function is always equal to unity.

In conclusion, an HMM model λ is sufficiently defined by the three parameters: transition matrix A, observation probability matrix B, and initial model probability π, as $\lambda=(A,B,\pi)$.

2.1.1.2 HMM Problem Definition:

Modeling a random process, whose observed outcome is O, with an HMM is subject to a certain measure, e.g. Likelihood. The likelihood of a process model to generate the observation O indicates to what extent the model fits the modeled process. Hence, the *evaluation* of the likelihood measure becomes an essential part of the model learning. In general, HMM *learning* is a process by which model parameters are estimated such that a certain quantity, e.g. likelihood, is maximized. In order to estimate model parameters, it is not enough to evaluate the likelihood of the observation O to be generated by the model, but we also need to know *how* the model generates the observation. Determining the optimal state sequence by which the model generates the observation is referred to as *decoding* process.

Hence, HMM problem is reduced to three problems:

1-Evaluation Problem.

2-Decoding Problem.

3-Learning Problem.

The following is a brief description of these problems and some basic solutions.

2.1.1.2.1 Evaluation Problem:

The term *evaluation* refers to the estimation of the probability of the observation sequence O to be generated by a given model λ, $P(O|\lambda)$. Given a model λ, observation sequence O can be generated with one state sequence Q of the many possible state sequences. State sequence Q is the list of state indices corresponding to the active state at each time instant through the time period of all observations $1 \leq t \leq T$.

This probability can be written in terms of the probability of the observation sequence given the state sequence and the model as:

$$P(O \mid \lambda) = \sum_{all\ Q} P(O \mid Q, \lambda)\, P(Q \mid \lambda), \quad Q = q_1, q_2, q_3, \ldots, q_T \qquad O = O_1, O_2, O_3, \ldots, O_T \tag{2.8}$$

The first factor of the sum term in equation (2.8), the probability of the observation sequence given the state sequence and the model, can be expressed as:

$$P(O \mid Q, \lambda) = \prod_{t=1}^{T} P(O_t \mid q_t, \lambda) \tag{2.9}$$

The right hand side is the multiplication of T factors each representing the probability of the observation at the time t, O_t, to be generated by a given active state at the same time instant t, q_t. The factor at time t depends solely on the probability distribution function of the active state q_t, $b_{q_t}(O_t)$, i.e.,

$$\prod_{t=1}^{T} P(O_t \mid q_t, \lambda) = b_{q_1}(O_1)\, b_{q_2}(O_2) \ldots b_{q_T}(O_T) \tag{2.10}$$

Note that the active state at the time instant t, q_t is one of the N states of the model λ, say S_j (the actual state is determined in the *decoding* phase). Hence, the probability distribution function of the active state q_t, $b_{q_t}(O_t)$, is, in fact, $b_j(O_t)$ which is determined by Equation (2.5).

The second factor of the sum term in Equation (2.8), $P(Q|\lambda)$, is the probability that a certain state sequence occurs. This term is observation independent; it depends only on the transition matrix, A, and the initial model probability, π, and it can be written as:

$$P(Q \mid \lambda) = \pi\, a_{q_1 q_2}\, a_{q_2 q_3}, \ldots, a_{q_{T-1} q_T} \tag{2.11}$$

Equation (2.8) suggests: *i*) evaluating the probability of the observation sequence given a state sequence and model parameters by Equation (2.9) (T multiplications per state sequence), *ii*) evaluating the probability of that state sequence given model parameters by Equation (2.11) (another T multiplications per state sequence), *iii*) multiply them (one multiplication), *iv*) summing the product over all possible state sequences (a total of $2T+1$ multiplications per state sequence). This is not practically feasible since the number of available state sequences grows exponentially with the length of the observations sequence (N^T possible state sequences), and so does the complexity. For example, the number of multiplications for a 3-state model if the observations sequence is of length 100 is $(2\text{x}100+1)(3)^{100}$. That is to say $(2T+1)N^T$. Instead, the Forward-Backward algorithm [9] and the Viterbi algorithm [9] handle the problem in a recursive manner where the probability calculation at the current observation exploits the previous calculations of the past observations.

Forward-Backward Algorithm

The main idea of the forward-backward algorithm is to convert the problem in equation (2.8) to a recursive form.

Forward Phase:

First, define forward variable $\alpha_t(i)$, as the joint probability that the partial observation sequence $O_1, O_2, \ldots, O_t$ occurs and that the state S_i is the current active state q_t given the model λ. $\alpha_t(i)$ is thus given by:

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = S_i \mid \lambda)$$

Initialization:

$$\alpha_1(i) = \pi_i \, b_i(O_1), \quad 1 \le i \le N \tag{2.12}$$

Recursion over t:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad 1 \le j \le N, 1 \le t \le T-1 \tag{2.13}$$

Termination:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.14}$$

Backward Phase:

Similarly, a backward variable $\beta_t(i)$, is defined as:

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T, q_t = S_i \mid \lambda)$$

And it can be recursively computed as follows:

Initialization:

$$\beta_T(i) = 1, \quad 1 \le i \le N \tag{2.15}$$

Recursion $t=T-1$ to $t=1$:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \quad 1 \le j \le N \tag{2.16}$$

The forward phase of the forward-backward algorithm is enough to solve the evaluation problem while backward phase helps solving the decoding problem [9].

2.1.1.2.2 Decoding problem:

The decoding problem is concerned with finding the optimal state sequence, Q, by which the model λ best generates a given observation sequence O. This can be achieved by using the forward-backward algorithm to find the state that is most likely to be the active state at each time instant t, q_t, as follows:

Let $\gamma_t(i)$ be the probability of the state of index i to be the active state at the time t given observation sequence and model parameters. $\gamma_t(i)$ can be defined as:

$$\gamma_t(i) = P(q_t = S_i \mid O, \lambda)$$

The quantity $\gamma_t(i)$ is referred to as *forward-backward variable* and can be computed by:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{T} \alpha_t(i)\beta_t(i)} \tag{2.17}$$

Then, q_t is chosen such that its probability to be the active state at time t given the observation sequence and model parameter, $\gamma_t(i)$, is maximum.

$$q_t = \arg\max_i [\gamma_t(i)], \quad 1 \le i \le N \tag{2.18}$$

The typical Forward-Backward Algorithm is subject to numerical underflow, hence, the scaling of the forward and the backward variables becomes necessary. The Viterbi algorithm is a popular and simpler alternative algorithm. It enables performing the

likelihood evaluation and decoding tasks in the log-domain, thus eliminating the need for scaling.

The complexity of the forward-backward algorithm for the 1-D first-order HMM is given by:

$C^*_{HL} = N(N+1)(T-1)+N \cong N^2T$ multiplication (2.19)

$C^+_{HL} = N(N-1)(T-1) \cong N^2T$ addition (2.20)

The number of computations above is needed to calculate the unconditional probabilities for an observation sequence, involving the state transitions. The unconditional probability will be referred to as *Hidden Layer* throughout the rest of thesis. The above complexity excludes the scaling operations and the operations of the Observation Layer, i.e. conditional probability computations of individual observations given the state. The complexity of the observation layer is not addressed at this level since it does not depend on the algorithm that handles the state transitions. Rather, it depends on the type of the observations and the observation probability model, e.g. Discrete, Continuous or Semi-Continuous.

Viterbi algorithm:

Let the state score $\delta_t(i)$ be the observation likelihood maximized over the past state sequence terminating with S_i at time t. Thus, $\delta_t(i)$ can be defined as:

$$\delta_t(i) = \max_{q_1,\ldots,q_{t-1}} P[q_1, q_2, \ldots, (q_t = i), O_1, O_2, \ldots, O_t \mid \lambda]$$

Then $\delta_{t+1}(i)$ can be computed by induction as follows:

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] b_j(O_{t+1}) \quad (2.21)$$

Equation (2.21) is the heart of Viterbi algorithm which chooses the state S_i that results in the highest score when a transition occurs from S_i at time t-1 to S_j at time t. A pointer

should be kept to such a state so the whole path, i.e., active state sequence, could be retrieved later on. A predecessor parameter $\psi_t(j)$ is defined in order to store the index of the best predecessor state of the current state S_j at time t, i.e.:

$$\psi_{t+1}(j) = \arg\max_{1 \le i \le N}[\delta_t(i)a_{ij}].$$

The algorithm can be scripted as follows:

1- Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N \tag{2.22}$$

2- Recursion $t=2$ to $t=T$:

$$\begin{aligned} \delta_t(j) &= \max_{1 \le i \le N}[\delta_{t-1}(i)a_{ij}]b_j(O_t), \quad 1 \le j \le N \\ \psi_t(j) &= \arg\max_{1 \le i \le N}[\delta_{t-1}(i)a_{ij}], \quad 1 \le j \le N \end{aligned} \tag{2.23}$$

3- Termination:

$$\begin{aligned} P(O, Q^* \mid \lambda) &= \max_{1 \le i \le N}[\delta_T(i)] \\ q_T &= \arg\max_{1 \le i \le N}[\delta_T(i)] \end{aligned} \tag{2.24}$$

where Q^* is the optimal state sequence.

4- Backtracking:

$$q_t = \psi_{t+1}(q_{t+1}), \quad t = T-1, T-2, \ldots, 1. \tag{2.25}$$

The Viterbi algorithm not only solves the problem of finding the best path, i.e., state sequence, but also provides an approximate solution to the evaluation problem considering the following approximation:

$$P(O \mid \lambda) \cong \max_{All\ Q} P(O, Q \mid \lambda) = P(O, Q^* \mid \lambda) \tag{2.26}$$

If the calculations above are carried out in the log-domain, no scaling is required and the complexity can be expressed as:

$C^{+}_{HL} \cong N(N_{pr}+1)(T-1)$ addition (2.27)

where N_{pr} is the average number of state predecessors. Thus, the number of additions is approximately $3NT$ for left-to-right HMM and N^2T for fully-connected HMM. This approximation will be used to estimate the complexity of the 2-D PHMM and Embedded HMM later in this Chapter.

2.1.1.2.3 Learning problem:

Given a model λ and a set of observation sequences, referred to as *training set*, the learning problem is concerned with determining the model parameters that maximize the likelihood of the training set. There is no exact estimation for model parameters that globally maximizes the observations likelihood, however, a local maximum can be reached. Following is a brief review of *Baum-Welch* method for HMM learning [9].

Let $\xi_t(i,j)$ be the joint probability of S_i and S_j to be active states at time t and $t+1$ respectively, i.e.:

$$\xi_t(i,j) = P[q_t = S_i, q_{t+1} = S_j \mid O, \lambda]$$

It is computed by:

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O \mid \lambda)} = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \quad (2.28)$$

Then, the forward-backward variable $\gamma_t(i)$ can be written as:

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j) \quad (2.29)$$

Note that the expected number of transitions from S_i is $\sum_{t=1}^{T}\gamma_t(i)$ while the expected number of transition from S_i to S_j is $\sum_{t=1}^{T}\xi_t(i,j)$. According to this, the model parameters (A,B,π) are estimated as follows:

π_i' is the expected number of times the active state at time $t=1$ is S_i,

a'_{ij} is the expected number of transitions from state S_i to S_j normalized w.r.t. the total expected number of transitions from state S_i,

$b'_j(k)$ is the expected number of observing codebook entry V_k by state S_j normalized w.r.t. the total expected numbers state S_j has been active.

$$\pi'_i = \gamma_1(i) \tag{2.30.a}$$

$$a'_{ij} = \frac{\sum_{t=1}^{T}\xi_t(i,j)}{\sum_{t=1}^{T}\gamma_t(i)} \tag{2.31.b}$$

For discrete probability HMM:

$$b'_j(k) = \frac{\sum_{\substack{t=1 \\ O_t \equiv V_k}}^{T}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)} \tag{2.32.c}$$

For continuous probability HMM:

$$c'^{(m)}_j = \frac{\sum_{t=1}^{T}\gamma_t(j,g)}{\sum_{t=1}^{T}\sum_{g=1}^{G}\gamma_t(j,g)} \tag{2.33.d}$$

$$\boldsymbol{\mu}_j'^{(m)} = \frac{\sum_{t=1}^{T} \gamma_t(j,g).\mathbf{O}_t}{\sum_{t=1}^{T} \gamma_t(j,g)} \tag{2.34.e}$$

$$\boldsymbol{\Sigma}_j'^{(m)} = \frac{\sum_{t=1}^{T} \gamma_t(j,g).(\mathbf{O}_t - \boldsymbol{\mu}_j^{(m)})(\mathbf{O}_t - \boldsymbol{\mu}_j^{(m)})^T}{\sum_{t=1}^{T} \gamma_t(j,g)} \tag{2.35.f}$$

where $\gamma(j,g)$ is the probability of the g^{th} Gaussian component of the state S_j to be the best to represent observation O_t.

Figure 2-4 shows an example of the time-state two-dimensional lattice and the possible state transitions in Figure 2-4.a, while an example of the forward and backward phases of the Viterbi algorithm are depicted in Figure 2-4.b and Figure 2-4.c, respectively. Figure 2-4.d shows an example of the best state sequence.

Figure 2-4 shows an example of the time-state two-dimensional lattice and the possible state transitions in Figure 2-4.a, while an example of the forward and backward phases of the Viterbi algorithm are depicted in Figure 2-4.b and Figure 2-4.c, respectively. Figure 2-4.d shows an example of the best state sequence.

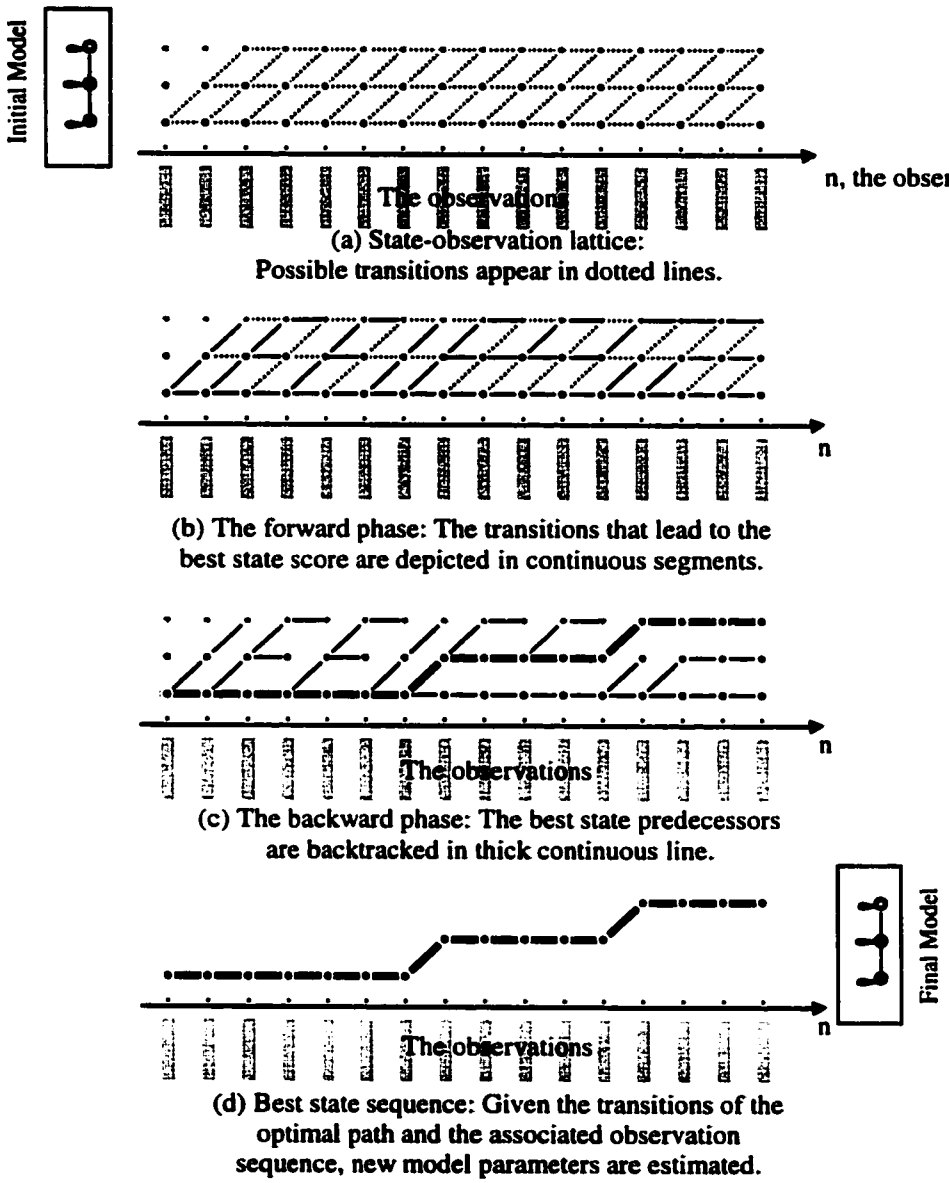


Figure 2-4. Viterbi Algorithm as applied to the HMM problem.

2.1.2 The HMM-Based Recognition System

An HMM recognizer works in two modes, training mode and recognition mode. In the training mode a sufficient database is provided for training proposes. The data is block-segmented then observation features are extracted to represent each block. An initial model is formed based on a random guess, supervised segmentation, or uniform segmentation of the training observations. Then, the initial model parameters are updated for better data-model matching with learning algorithm. The last step is repeated until the relative likelihood reaches a threshold level at which the training is said to have converged. The training procedure is illustrated in Figure 2-5.

On the recognition mode, the HMM recognizer aims to determine the model that it is most probably the one that produced the provided test image. i.e.

$$\lambda = \arg\max_{All\,\lambda} P(\lambda \mid O) \tag{2.36}$$

This is a problem of *Maximum A Postriori Probability*, known as *MAP*. *Bayes' rule* is used to reformulate the recognition problem as:

$$\lambda = \arg\max_{All\,\lambda} \frac{P(O \mid \lambda)P(\lambda)}{P(O)} \tag{2.37}$$

Unless the nature of the observation domain dictates otherwise, $P(\lambda)$ is considered equal for all λ's, i.e. all models are equally probable. On the other hand $P(O)$ does not depend on which model λ satisfies the *argmax(.)* statement. Thus Equation (2.36) can be simplified as:

$$\lambda = \arg\max_{All\,\lambda} P(O \mid \lambda) \tag{2.38}$$

Equation (2.38) expresses a *Maximum Likelihood (ML)* solution to the recognition problem. The recognition procedure is as follows: the test data is supplied, block-segmented, and transformed to the feature domain. Then, the likelihood of the model λ_i to generate the observation sequence O is computed for all λ. The model with maximum likelihood is then chosen as depicted in Figure 2-6.

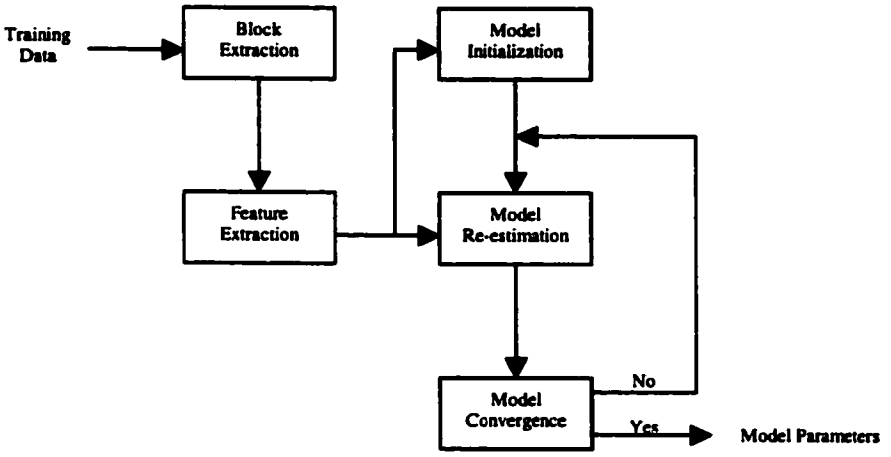


Figure 2-5. HMM Training [28]

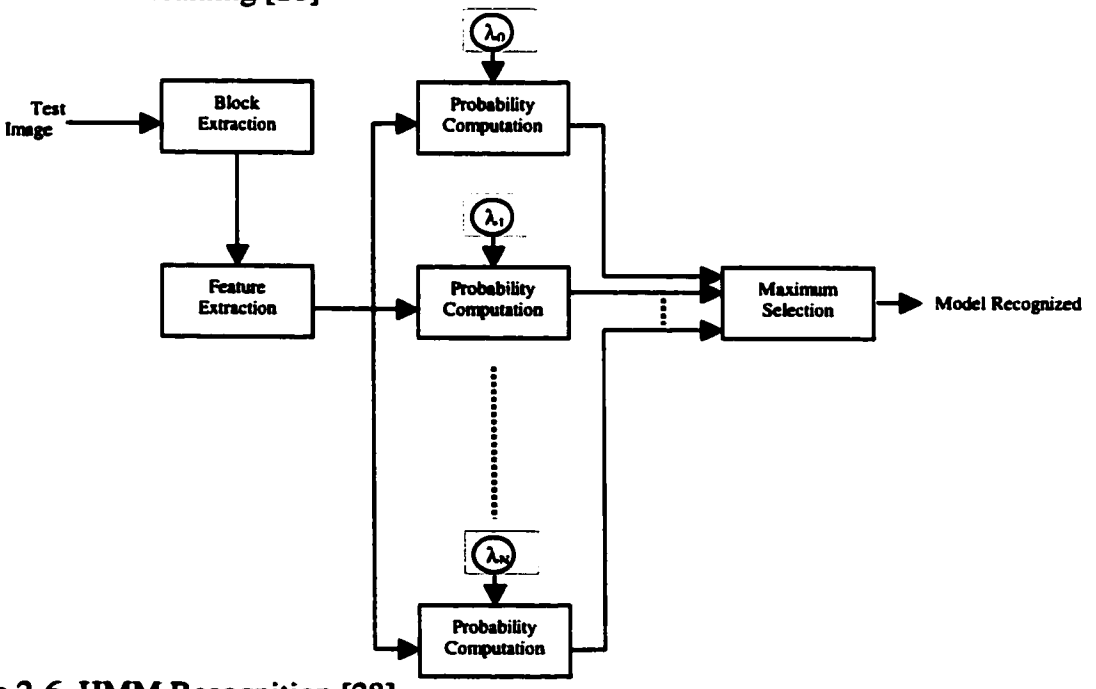


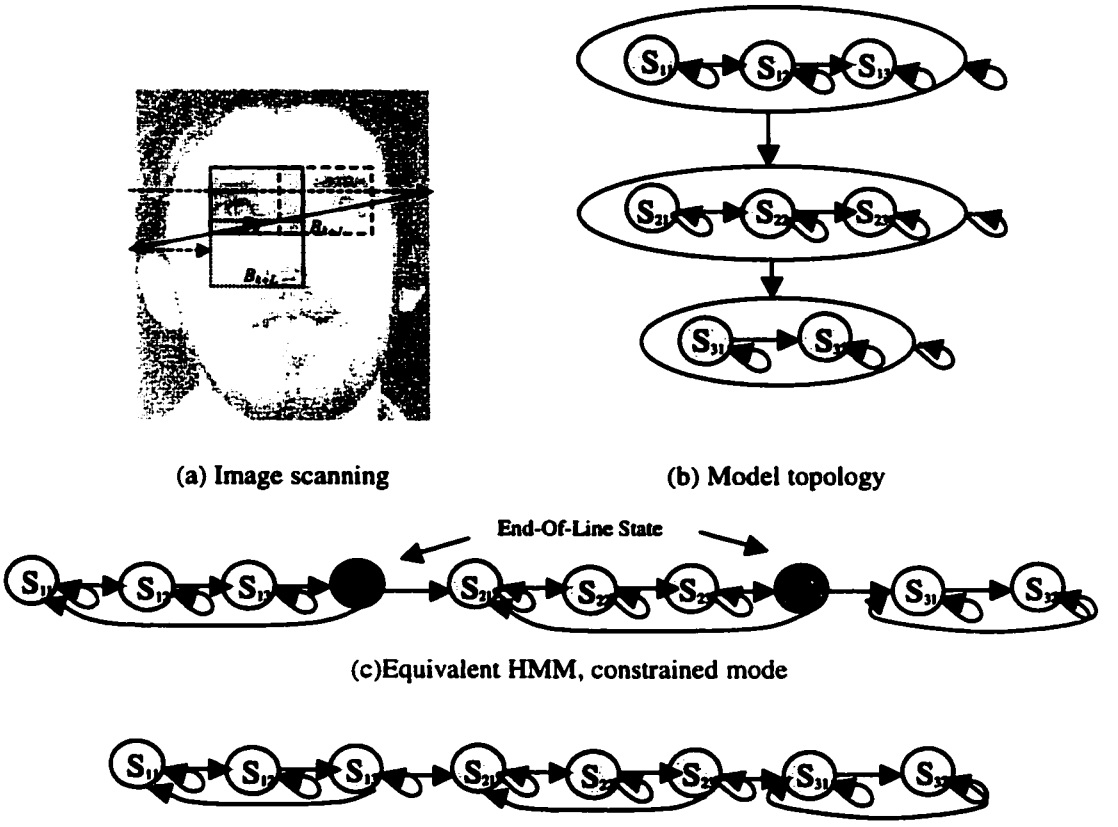
Figure 2-6. HMM Recognition [28]

2.2.1 Two-Dimensional Psuedo Hidden Markov Model (2-D PHMM)

A two-dimensional Psuedo Hidden Markov Model, 2-D PHMM, consists of number of superstates, each of them containing a Markov chain as depicted in Figure 2-7. The number of the states in the Markov chain may vary from one superstate to another. The *Model Topolgy* describes the structure of the model in terms of the state, e.g. a model with topology 2-3-3-2 contains 4 superstates including 2,3,3,2 states respectively. Transitions in 2-D PHMM are of two types. The first type is found between the consecutive superstates while the second type is found between the consecutive states within the same superstate. No transition is allowed between states in different superstates. Considering an image, the observation sequence of 2-D PHMM is a list of pixel luminance blocks scanning the image in left-to-right top-to-bottom order as in [10], binary pixels scanned in top-to-bottom left-to-right order as in [11], or group of blocks in a same color segment assigned to the superstate by color segmentation stage as in [12].

The 2-D PHMM in [10] does not exploit the relation between two vertically consecuetive *blocks*, instead, it just exploits the vertical relation between two consecuetive *rows of blocks*. This is obviously manifested in two aspects:

- The vertical transition between states in different superstates is not allowed.
- The blocks are scanned in a manner that results in a 1-D sequence of observations which is not the same dimensionality of the image, i.e. 2D.



(a) Image scanning

(b) Model topology

(c)Equivalent HMM, constrained mode

(d) Equivalent HMM, unconstrained mode

Figure 2-7. 2D-PHMM [10][9].

The same analysis applies to the algorithm in [11] but with pixels instead of blocks and exchanging the dimensions, i.e. rows and columns. The approach in [12] is slightly different, where the rows are classified by color segmentation before being assigned to the superstates. This reflects the limitation of the 2-D PHMM to run freely in two dimensions.

To conclude, 2-D PHMM is equivalent to 1-D HMM whose states are lined in one chain according to their internal order within their superstates and the order of the superstates as well. An end-of-line state may be added to the chain at the point of

superstate link to restrict each scanned line to a superstate in the constrained 2-D PHMM as shown in Figure 2-7.

Based on the discussion in Section 2.1.1.2.2, the complexity of the 2-D PHMM for the unconstrained and the constrained modes, respectively, with Viterbi algorithm is:

$$C_{HL}^{+} \cong 3\sum_{k=1}^{N_0} N_1^{(k)}(T_1T_0 - 1) \cong 3T_1T_0\sum_{k=1}^{N_0} N_1^{(k)} \quad \text{addition} \tag{2.39}$$

$$C_{HL}^{+} \cong 3\left(\sum_{k=1}^{N_0} N_1^{(k)} + N_0\right)(T_1T_0 - 1) \quad \text{addition} \tag{2.40}$$

where $N_1^{(k)}$ is the number of states in the k^{th} superstate,

N_0 is the number of superstates,

T_0 is the number of rows in the image,

T_1 is the number of blocks in each row.

2.2.2 Embedded HMM

The embedded HMM has a structure that is similar to the 2-D PHMM [13] but the image is scanned in a two-dimensional manner [14] and the 1-D Viterbi algorithm is implemented in two layers. In the first layer the 1-D Viterbi algorithm is used to estimate the probability of each sub-chain of states to generate each row individually. Then, the second layer estimates the overall probability of the main-chain of superstates based on the estimates of the preceding layer as depicted in Figure 2-8. It is worth noting that, although the 2-D PHMM and the embedded HMM share the same structure, they realize it in different ways. The embedded HMM implicitly justifies the row-superstate constraint by virtue of the 2-D scanning strategy while the 2-D PHMM either explicitly applies it by inserting an all-white block in the 1-D block sequence at the end of each row or just leaving it free. In both cases, the 2-D PHMM does not guarantee the superstate

transitions to be vertical transitions, i.e. those transitions that occur between states that correspond to vertically neighboring blocks.

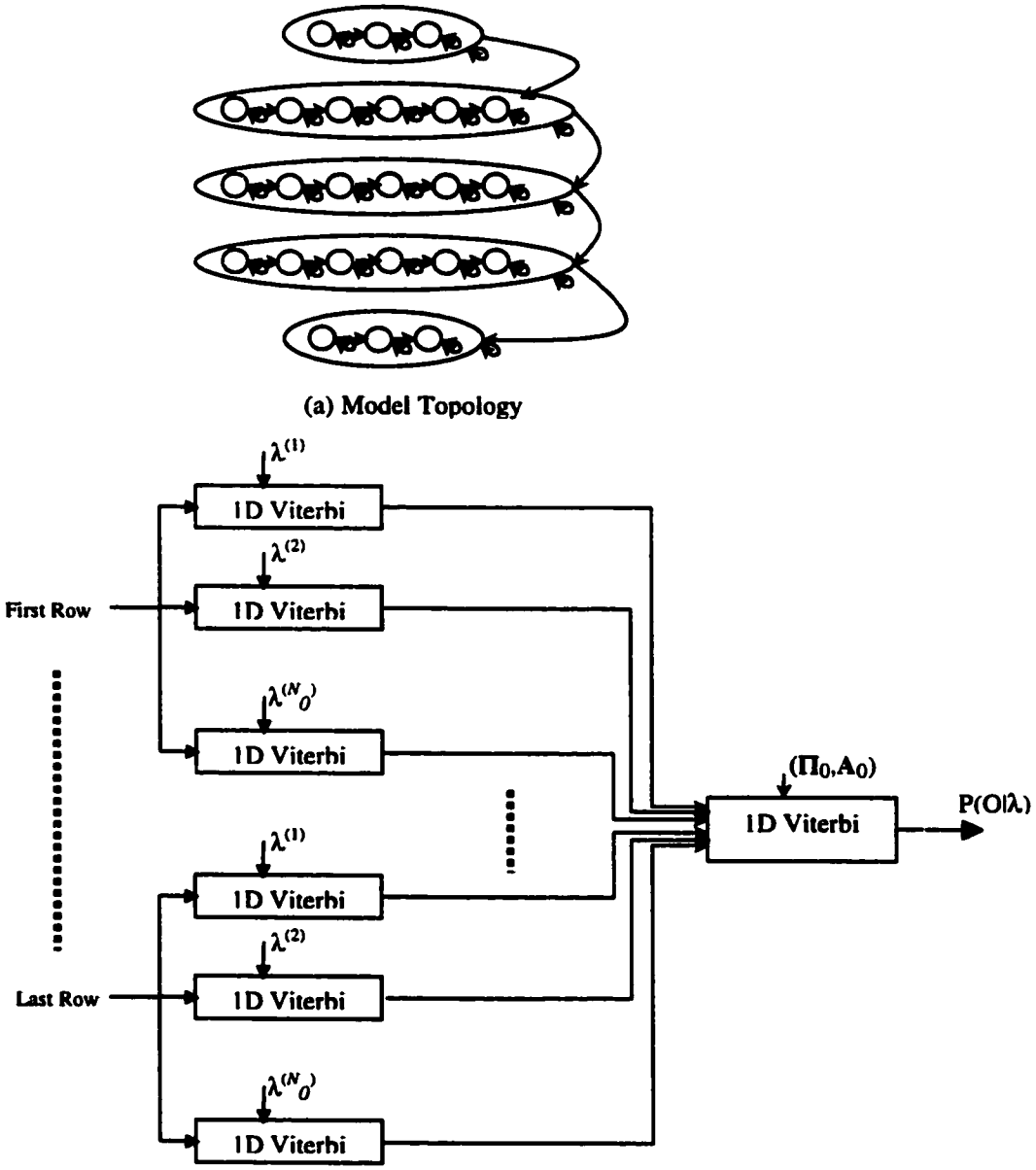


Figure 2-8. The Embedded HMM [14].

The Embedded HMM, λ, has three main components:

1- The initial superstate distribution, $\Pi_0=\{\pi_{0,i}\}$, where $\pi_{0,i}$ is the probability of the superstate $S_{0,i}$ to be the active state at the time 0.

2- The superstate transition matrix, $A_0=\{a_{0,ij}\}$, where $a_{0,ij}$ is the probability of the transition from superstate $S_{0,i}$ to superstate $S_{0,j}$.

3- The set of the embedded 1-D HMMs, $\Lambda=\{\lambda^{(k)}\}$, where $\lambda^{(k)}$ is the set of parameters of the k^{th} superstate that include :

   a- The initial state distribution, $\Pi_1=\{\pi_{1,i}\}$, where $\pi_{1,i}$ is the probability of the state $S_{1,i}$ to be the active state at the time 0.

   b- The state transition matrix, $A_1=\{a_{1,ij}\}$, where $a_{1,ij}$ is the probability of the transition from state $S_{1,i}$ to state $S_{1,j}$.

   c- The observation probability matrix, $B^{(k)}$.

The likelihood of an image to be generated by that model is equal to the likelihood of the vertical Markov chain to generate the vertical sequence of rows, i.e. each row becomes a superstate observation. The likelihood of a superstate to generate a row of blocks is obtained by evaluating the likelihood of the associated Markov chain to generate the block sequence of that row.

The complexity of the hidden layer as reported in [14] is:

$$C^{+}_{HL} \cong \sum_{k=1}^{N_0} (N_1^{(k)})^2 T_1 T_0 + N_0{}^2 T_0 \quad \text{addition} \tag{2.41}$$

where $N_1^{(k)}$ is the number of states in the k^{th} superstate,

N_0 is the number of superstates,

T_0 is the number of rows in the image,

T_1 is the number of blocks in each row.

However, if Viterbi algorithm is used in Log-domain, the complexity will be:

$$C_{HL}^{+} = 3\sum_{k=1}^{N_0} N_1^{(k)}(T_1 - 1)T_0 + 3N_0(T_0 - 1) \cong 3T_1T_0\sum_{k=1}^{N_0} N_1^{(k)} + 3N_0T_0 \text{ addition} \qquad (2.42)$$

2.2.3 Markov Mesh Random Field HMM (MMRF HMM)

The Markov Random Field (MRF) is a 2-D extension of the regular 1-D Markov chain, which is used to model the image as a 2-D observation lattice instead of the 1-D observation sequence [15]- [17]. A truly 2-D HMM based on MRF has been introduced to optical character recognition in [18]. A causal variation of the MRF, namely third-order Markov Mesh Random Field (MMRF), has been adopted in that work in order to maintain the recursive nature of the model which helps avoid the prohibitively large amount of computations due to many practical considerations. Accordingly, the past region of the current observation is defined as the region that contains observation samples in previous rows or previous columns as shown in Figure 2-9. The Markov assumption that is corresponding to the third-order Markovian process is used. It is given as:

$$P(q_{k,l}|q_{k-1,l}, q_{k,l-1}, q_{k-1,l-1}, \cdots, q_{1,1}) = P(q_{k,l}|q_{k-1,l}, q_{k,l-1}, q_{k-1,l-1})$$

where $q_{k,l}$ is the current active state.

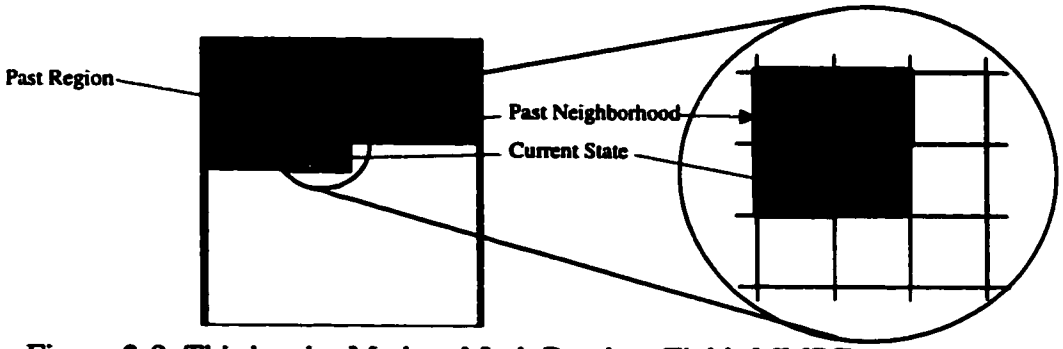


Figure 2-9. Third-order Markov Mesh Random Field (MMRF).

An observation O is recognized as a model λ according to:

$$\lambda = \arg\max_{All\ \lambda} P(O \mid \lambda)$$

The observation likelihood, $P(O|\lambda)$, is approximated by the joint probability of the observation O and the optimal state sequence, $\hat{Q}$, which is estimated by a look-ahead technique given the model λ, i.e.;

$$P(O \mid \lambda) = P(O, \hat{Q} \mid \lambda)$$

where the joint probability of the observation O and a state sequence Q given the class model λ, $P(O,Q|\lambda)$,is computed by:

$$P(O,Q \mid \lambda) = P(O \mid Q,\lambda)P(Q,\lambda)$$
$$= \prod_{k=1}^{K}\prod_{l=1}^{L} P(q_{k,l} \mid q_{k-1,l}, q_{k,l-1}, q_{k-1,l-1})P(O_{k,l} \mid q_{k,l})$$
$$= P(q_{1,1})P(q_{1,1} \mid O_{1,1})\prod_{l=2}^{L} P(q_{1,l} \mid q_{k,l-1})P(O_{1,l} \mid q_{1,l})\prod_{k=2}^{K}\prod_{l=1}^{L} P(q_{k,l} \mid q_{k-1,l}, q_{k,l-1}, q_{k-1,l-1})P(O_{k,l} \mid q_{k,l})$$

The probability of the current active state, $q_{k,l}$, given its past vertical, diagonal and horizontal neighbors, $q_{k-1,l}$, $q_{k-1,l-1}$, $q_{k,l-1}$, respectively, resembles the formal state transition probability matrix. However, it is a four-dimensional array, consequently, the complexity of the hidden layer is in the order of (N_s^4) or more, where N_s is the number of states in the model.

This model has been implemented for optical character recognition to recognize 10 digits, which are pre-segmented in bi-level patches of 64x224 pixels in [18]. The performance of the system has been investigated with both Decision Directed (DD) learning and Look-ahead-based learning.

2.3 Face Recognition Problem

The real need for robust and reliable automatic image and speech recognition, interpretation, and annotating tools has grown with the vast growth of multimedia databases. Manual classification, searching, and indexing is no longer feasible due to today's huge sizes of archived images, speech records and video sequences.

Face recognition technology provides an automated way to search, to identify or match a human face versus the contents of a pre-stored facial database [19]. It is integrated in security systems that restrict access to certain services or locations to a limited set of people recognized by face in addition to voice messages or typed passwords. Moreover, other applications of face recognition technology are found in photo-IDs like passports, credit cards...etc.

In some applications, controlling the photo capture environment is not possible. It is necessary to extract the human face from the whole scene. A face detection module is added in such cases prior to the face recognition module. A comprehensive survey of face detection methods is found in [20]. In this work, we emphasize the face recognition part assuming the photo capturing environment is controlled. This assumption is substantially fulfilled in many applications including photo-IDs, access control systems and criminal records. The term "Environment Control" covers factors that affect the appearance of the image content and even alter it in some extreme cases. They include the average object distance to the camera, the average light intensity and direction, and the position and the orientation of the object itself.

Thus, the problem is formulated as follows: identify a person by his frontal face view given a facial database that contains all possible persons, allowing for some facial expressions, small tilt, and some common changes, e.g. taking off glasses or closing eyes.

The problem of Face Recognition has been addressed by many researchers since the early 1970s. A survey of the work between 1991 and 1995 is provided in [19]. In general, face recognition systems can be classified by the type of features extracted to represent the spatial information of the facial image and the matching criterion upon which the decision is made. The extracted features include transform features, visual features and algebraic features while the matching criteria may be direct like a nearest neighbor selection, based on a suitable distance measure, or statistical-based like a maximum likelihood selection. We will briefly review some of the commonly used transforms and statistical methods.

One of the most popular transforms in face recognition is the Karhunen-Loeve Transform (KLT) [21]. In [22], the KLT is implemented to form an eigenface space using the available facial database. The face image under test is projected into the eigenface space then the projection is matched to the prepared projections of the database faces. Although the KLT is efficient in terms of power compaction and decorrelation, it is not the best choice in the sense of simplicity. A less efficient yet simpler transform, namely the Discrete Cosine Transform (DCT), is used to represent the image in the transform domain. No extra effort is needed to obtain the transform kernels since they are computed from the signal. The DCT approaches the KLT for the highly correlated first-order autoregressive process as the correlation factor approaches unity, which represents a wide spectrum of normal images [23]. The simplicity of the DCT and its adequate ability for

power compaction and decorrelation made it an essential part of the current standards like JPEG [25][26] and MPEG[26][27]. In the context of face recognition, the DCT has been implemented by Nefian [28], Tsapatsoulis [29], and many others in facial image decomposition. Discrete Wavelet Transform (DWT) [23][24] has been also used for face recognition [30] where the 2D-DWT is used to expand the face view then the histogram is calculated for the resultant wavelet coefficients. Based on the 2D-DWT image histogram moments, a search is carried out through the facial database and the best match is chosen.

In statistical recognition methods, the image features, i.e. the chrominance and luminance values of the pixels or their transformed values, are treated as random variables, which are presented in terms of their statistical properties or in the form of a stochastic model. The search process either finds the best match of the statistical properties directly, as in [30], or through a stochastic model. Examples of the stochastic model approach are the Probabilistic Decision-Based Neural Network (PDBNN) Face Recognition [31] and HMM-based Face Recognition. PDBNN consists of a finite number of weighted subnets with a probability density function (PDF) associated to each subnet. The probability density functions and the network weights are trained with reinforce/anti-reinforce learning based on the classification decision driven by the log likelihood estimate. Similarly, the HMM consists of a finite number of states. Each state has a probability density function. A probabilistic scheme is defined to govern the state transitions. A detailed discussion on the structure and the main parameters of the HMM as well as a brief survey of its recent applications follows.

2.4 HMM Applications in Face Recognition

In this section, we discuss the application of the HMM approaches reviewed in the previous sections to the face recognition problem.

Samaria [10] used 1-D HMM for face recognition. The observation of HMM states is a vector composed of luminance of the set of pixels belonging to a horizontal strip of width P pixels. The sequence of observations consists of vertically successive strips scanning the image from top to bottom. Two consecutive strips may overlap by up to one pixel less than the strip width. 85% accuracy has been reported using 5-state HMM with strip width of 10 pixels on *AT&T* database, formally known as *Olivetti Research Laboratory (ORL)* database [32].

Nefian and Hayes III [28] applied 1-D HMM to face recognition in similar a manner except for the HMM features type. They used 2D-DCT coefficients to represent the image strips instead of their pixel luminance values. The HMM feature vector consists of coefficients corresponding to DC and lower frequency bands, exploiting the fact that lower frequency bands are the most significant bands for normal images. 2D-DCT has been applied to strips of 10x92 pixels with 9 pixels vertical overlap, where the 10-pixels is vertical dimension of the strip and 92-pixels is its horizontal dimension, which is equal to the horizontal dimension of the 112x92 pixels image. Only the lower 3x13 coefficients out of the 10x92 DCT coefficients are considered for each strip to form a 39-dimensional HMM feature vector. The main contribution in [28] is the complexity reduction. A similar accuracy of 85% has been achieved. In [33] 1-D HMM with ergodic structure reached 100% when working on 10 DCT coefficients per block, where each block is 16x16 in size and 75% overlapped.

2-D PHMM has also been implemented in [10]. An accuracy of 94.5% has been achieved with 5-superstate model with topology of 3-6-6-6-3. The block size in this experiment was 10x8 with overlap 8x6. 24 states were used in this model.

Later, the embedded HMM was used in face detection and recognition [14] where the embedded HMM had 24 states arranged in 5 superstates as 3-6-6-6-3. The image was scanned with feature blocks of size 8x10 pixels and overlap of 6x8 pixels. A 2D-DCT was applied to each block where only the first 3x2 bands were used to form the observation vector. A high accuracy of 98% has been recorded by this system.

Again, 2-D PHMM has been implemented in color image retrieval in [12] where both chromatic and luminance spatial information are captured by the 2-D PHMM. Although a model of structure similar to the one in [10][9] has been used, the methodology is not the same. In order to retrieve a color image from color image database, the image is first block-segmented according to the color and luminance information, then divided into horizontal strips based on the block segmentation. The horizontal strips are mapped to the superstates of 2-D PHMM.

After training, the query is invoked by either real image or a hand-colored graph. The Viterbi algorithm is used to estimate the likelihood that the query picture was generated by each model, then the best candidates are retrieved.

It is worth noting that in [12] the 2-D PHMM recognizer is preceded by a color-based segmentor. This follows from the fact that 2-D PHMM has poor capabilities of state segmentation as Samaria has experienced two years before [10]. That is mainly because the connection between internal states in any consecutive superstates is limited. This is

one of our motivations to investigate 2-D HMM in the current work as a replacement to 2-D PHMM.

Recently, a 2-D HMM has been introduced in [18] based on third order Markov Random Field (MMRF) for off-line handwritten character recognition. When the formal decision-directed estimation algorithm was used for training, 74.9% of the testing data were correctly recognized. An estimation algorithm was proposed to enhance the recognition accuracy achieving 90.8 %. The work in [18] is based on a recognition set which consists of 10 bi-level handwritten Arabic \s 1 digits. Although the system has good performance, its complexity is rather high. It is in the order of (N_t^4) per pixel, where N_t is the total number of states in the model.

Comparatively, the 1-D HMM has the simplest structure, that is directly inherited from the left-to-right HMM applied on a 1-D observation sequence, while the MMRF HMM results in the most complex structure to attack the 2-D problem involving a 3rd order Markov solution. In between, the 2-D PHMM and the Embedded HMM offer compromise options. Although they both have similar structures, the Embedded HMM has the advantage of its ability to handle the image in its 2-D form by means of two layers of the regular 1-D Viterbi.

2.5 Conclusion

In this chapter, an introduction to the problem of face recognition is given with a brief survey of methods applied to this problem in the literature including HMM-based methods. The structure of the 1-D HMM is discussed in details along with two of the most common algorithms to solve the traditional 1-D HMM problems, i.e. evaluation problem, decoding problem and learning problem. The use of the Forward-Backward algorithm and the Viterbi algorithm as applied to the problem of the 1-D HMM is visited.

This chapter also sheds some light on the HMM-based models in higher dimensions. It was shown how the basic structure of the 1-D HMM is embedded in a Pseudo 2-D structure to form 2-D Pseudo HMM (2-D PHMM) and to form Embedded HMM (EHMM). The aspects of similarity and dissimilarity between 2-D PHMM and EHMM are highlighted. Markov Mesh Random Field-based 2-D HMM is also addressed in this chapter as an example of a true 2-D HMM. Then, the HMM-based face recognition systems are compared considering their relative complexity and correct recognition rates. Obviously, to move to a true 2-D HMM, the model complexity is an impediment factor for implementation. In the following chapter, a reduced-complexity 2-D HMM model is introduced and is applied to the face recognition problem.

Chapter 3 Proposed HMM-Based Systems for Face Recognition

It is shown in the previous chapter that only one dimension of the image is modeled as a Markov process when modeled by 1-D HMM-based models. Since images, in general, are 2-D observations that can be addressed as outcomes of some 2-D random process, it is of our best interest to consider capturing the probabilistic features of the image elements in both dimensions. This is achieved by the third-order MRF-based 2-D HMM system in [18], which is a true 2-D model that is realizable only with vast complexity. Other models have been introduced based on pseudo 2-D state structure in order to avoid the vast complexity problem, e.g. 2-D PHMM and Embedded HMM. These models break the 2-D problem into two 1-D problems, hence, they do not fully capture the probabilistic behavior of image elements, e.g. blocks, in both dimensions. They rather capture the probabilistic relationship between consecutive blocks in one dimension, say horizontal, in-detail and, at the same time, tend to roughly capture the probabilistic relationship between consecutive sets of blocks (say rows) in the other (say vertical) dimension. In such a case, the relationship between two consecutive blocks is overloaded by the behavior of the overall (vertically consecutive sets) rows, which the two blocks are members of.

In this chapter, we propose a reduced complexity 2-D HMM that handles the image as an outcome of 2-D random process, retaining the relationship between each block and its vertical and horizontal predecessors. The proposed model is applied to face recognition whereas to achieve high recognition rates, one has to ensure that the model is

sufficiently trained. We will consider the model description shortly. However, we first start by studying the impact of the size of the training data set on the recognition rate. Then, once a reasonable data size is determined, we move to the determination of the 2-D model that uses this data.

3.1 The Effect of the Training Data Set Size of on the Model Robustness

The trained model represents a class of observations based on the information provided by the training data during the training phase. The assumption of the modeling process is that the supplied training data set is wide and varied enough to cover the corresponding classes. However, the possible observation members of a certain class constitute an infinite set. Obviously, the former assumption is not sufficiently fulfilled with small or invariant training data sets. In such cases, the trained model is a training-data-specific model and it shows poor performance on testing, given that the testing data set and the training data set are exclusive, i.e. no image co-exists in both of them at the same time.

The following experiment has been conducted to investigate the effect of the training data size on the model performance using the facial database whose description is provided in Appendix A. This will be helpful later to determine the right-size data set for use in the model that we are seeking. The rest of the face recognition experiments reported in this thesis have been carried out on the same facial database unless otherwise indicated. We have used the same feature extraction scheme that is used in [28], i.e. the 112x92-pixel images are divided into 10x92-pixel horizontal strips, where the dimensions of the image and the block given above are in the form of *vertical* x *horizontal* pixels.

The consecutive strips have an overlap of 9 pixels. 2D-DCT is applied to each strip, then the first 3(vertical)x13(horizontal) coefficients are used to form a 39-D feature vector.

The observation sequence of the 1-D HMM is the list of the feature vectors that correspond to the image strips scanned from top to bottom as shown in Figure 3-1. Figure 3-1 also shows the structure of the top-to-bottom 1-D 3-state HMM. A 4-state model was trained and the experiment was repeated for 5-state model. In both cases, a weighted sum of eight Gaussian components was used to approximate the continuous probability density function. The 2D-DCT coefficients have been weighted to emphasize the lower frequency bands. The experiment was repeated with and without lower frequency emphasis and the results are plotted in Figure 3-2.

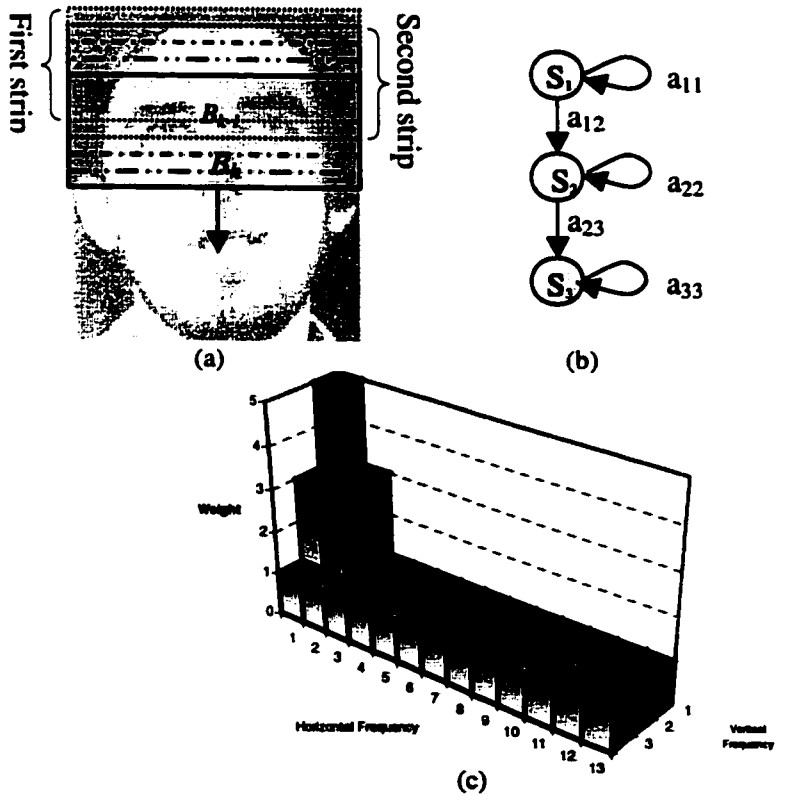


Figure 3-1. Top-to-Bottom 1-D HMM.
(a) Image scanning, (b) Model topology, (c) Low Pass Filtering (LPF) weights.

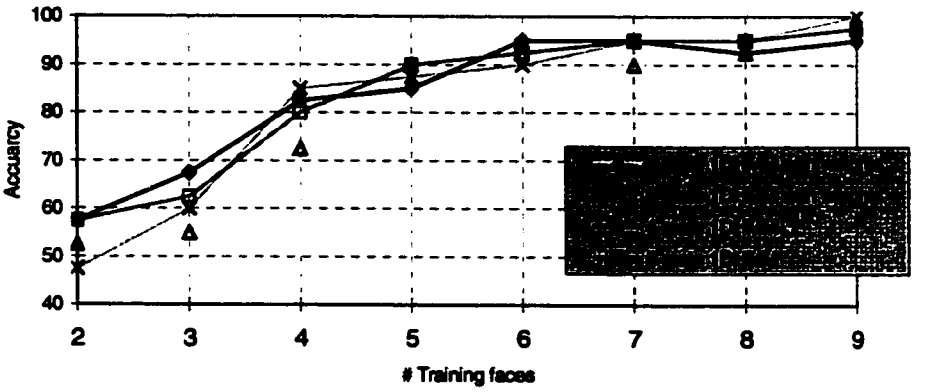


Figure 3-2. The effect of the training data size on recognition accuracy.

It is clear that, the performance of the 1-D HMM Face Recognition system is directly proportional to the size of the training data for small training data sizes. However, with larger training sets, the system performance saturates, as the newly added training faces do not contribute as much information as they would in the case of small training data sizes. The saturation takes place at almost 6 training faces with 102 observation blocks each, i.e. a total of 612 training block per model. These observations are used to train a model with 5 states, i.e. 122.4 blocks per state. Thus, if the target size of the proposed model is around 12 states, then we would need, 122.4*12=1468.8 training blocks if we want to maintain the same ratio. The proposed 2-D model is applied for image search in the compressed domain. By choice, images are represented by DCT coefficients of non-overlapped 8x8-pixel blocks, which is compatible with JPEG standards. Given that 168 blocks can be extracted per image, we estimate roughly, 1468.8/168=8.7 training images may be needed per person's model.

3.2 A Proposed Low Complexity 2-D HMM for Face Recognition

Given that we now have an estimate of the data size to be used with a given model, we now move to consider the model itself. Our objective is to propose a true 2-D HMM that provides good performance at reasonable complexity.

In this chapter, we will propose a 2-D HMM that is a rectangular constellation of states connected together by vertical and horizontal transition probabilities as shown in Figure 3-5. The main difference between a 2-D HMM and the 1-D HMM is that the latter has only one type of state transitions, thus, it does not possess 2-D localization property.

The difference between a 2-D HMM on one hand and the 2-D PHMM and the Embedded HMM on the other hand is that vertical transition is allowed between vertically consecutive states in different rows as opposed to the vertical transition between superstates in the 2-D PHMM and the Embedded HMM. This enables the model to exploit the statistical relation between two vertically consecutive blocks, thus accommodating the two-dimensional observations without converting the problem into a one-dimensional problem. In addition, it does not overload the state with the behavior of the rest of its chain, i.e. other states within the same superstate, as in the 2-D PHMM.

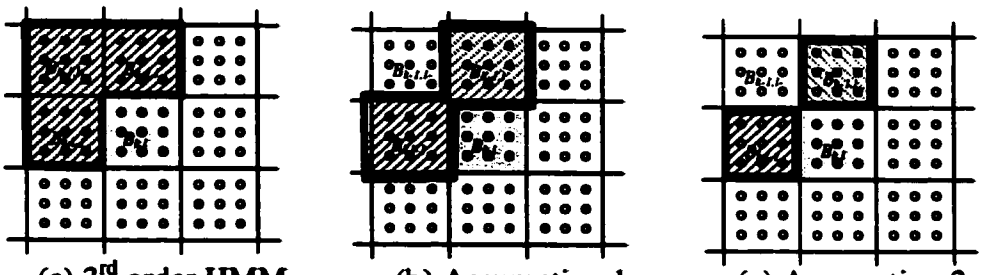
3.2.1 Design Assumptions

While 2-D HMM models are not new [18], they have not been widely used because of their excessive complexity. To simplify the model and reduce the system complexity of the proposed 2-D HMM system, two basic assumptions are made:

1) The active state at the observation block $B_{k,l}$, $q_{k,l}$, is dependent only on the immediately preceding vertical and horizontal neighbors, $q_{k-1,l}$, $q_{k,l-1}$. This assumption is equivalent to Markov assumption for the second-order source.
2) The active states of the two observation blocks in anti-diagonal neighborhood locations, e.g. $B_{k,l-1}$, $B_{k-1,l}$ are statistically independent given the current state, $q_{k,l}$, and the past observations.

The first assumption discards the explicit dependency between the diagonal neighbors while the second assumption conditionally discards the dependency between the anti-diagonal neighbors as depicted in Figure 3-3. [34] investigated the model based on assumption (2) with no knowledge of the current state, assuming unconditional independence. Inferior results were obtained as will be depicted in the comparison later in this chapter.

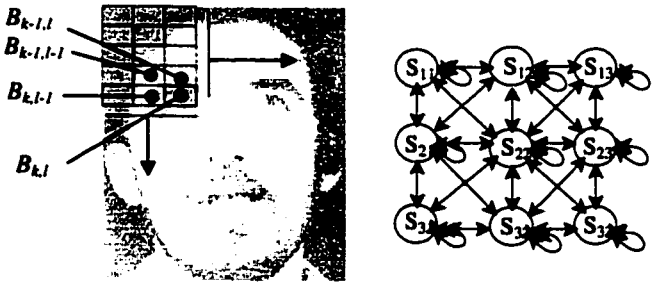
Regarding the model complexity, the first assumption reduces the complexity of the hidden layer from the order of (N_t^4) of the third-order HMM as in [18] to the order of (N_t^3) similar to the second-order HMM, where N_t is the number of states in the model. Furthermore, the second assumption enables a complexity reduction in the hidden layer to the order of $(2N_t^2)$.



(a) 3rd order HMM. (b) Assumption 1. (c) Assumption 2.

Figure 3-3. Proposed 2-D HMM main assumptions.

The scanning of the image is carried out in a 2-D manner forming a 2-D observation sequence, i.e. each block carries two indices that describe its spatial location relative to the other blocks, as depicted in Figure 3-4. Block overlapping is not supported and the block size is chosen as 8x8 pixels to maintain a compatibility with JPEG standards. It should be noted that a face detection module can be added to extract the face from the background based on a certain model, e.g. ellipse. This contributes to the improvement of the performance of the system as it is been shown in [35].



(a) (b)

Figure 3-4. 2-D HMM. (a) Image scanning, (b) Sample model topology.

3.2.2 Model Topology

The proposed model has a rectangular state constellation as in [36]. In this model, N_t states are arranged in N_I columns and N_O rows as shown in Figure 3-5. Recall that a vertical state transition describes the relationship between the state that represents the current observation block and the state that represents the vertically preceding block, i.e. the observation block that is located immediately above the current observation block. Similarly, a horizontal state transition describes the relationship between the state that represents the current observation block and the state that represents the horizontally preceding block, i.e. the observation block that is located immediately to the left of the

current observation block. In order to obtain a true two-dimensional extension of the top-to-bottom 1-D HMM mentioned in the previous section, the state transition probabilities are defined subject to the following restrictions:

i) The probability of vertical transition from state $S_{i,j}$ to a state $S_{m,n}$, $v_{i,j;m,n}$, is zero unless $S_{i,j}$ belongs to the *Vertical possible predecessor set, VPPred, of* $S_{m,n}$, i.e.

$$v_{i,j;m,n} = 0 \text{ if } S_{i,j} \notin VPPred(S_{m,n}). \tag{3.1}$$

ii) The probability of horizontal transition from state $S_{x,y}$ to a state $S_{m,n}$, $h_{x,y;m,n}$, is zero unless $S_{x,y}$ belongs to the *Horizontal possible predecessor set, HPPred, of* $S_{m,n}$, i.e.

$$h_{x,y;m,n} = 0 \text{ if } S_{x,y} \notin HPPred(S_{m,n}). \tag{3.2}$$

iii) Vertical and horizontal possible predecessor sets are formed such that the model is a top-to-bottom left-to-right 2-D HMM as a two-dimensional extension to the left-to-right no-skip 1-D HMM with one degree of flexibility in the transverse dimension, i.e.

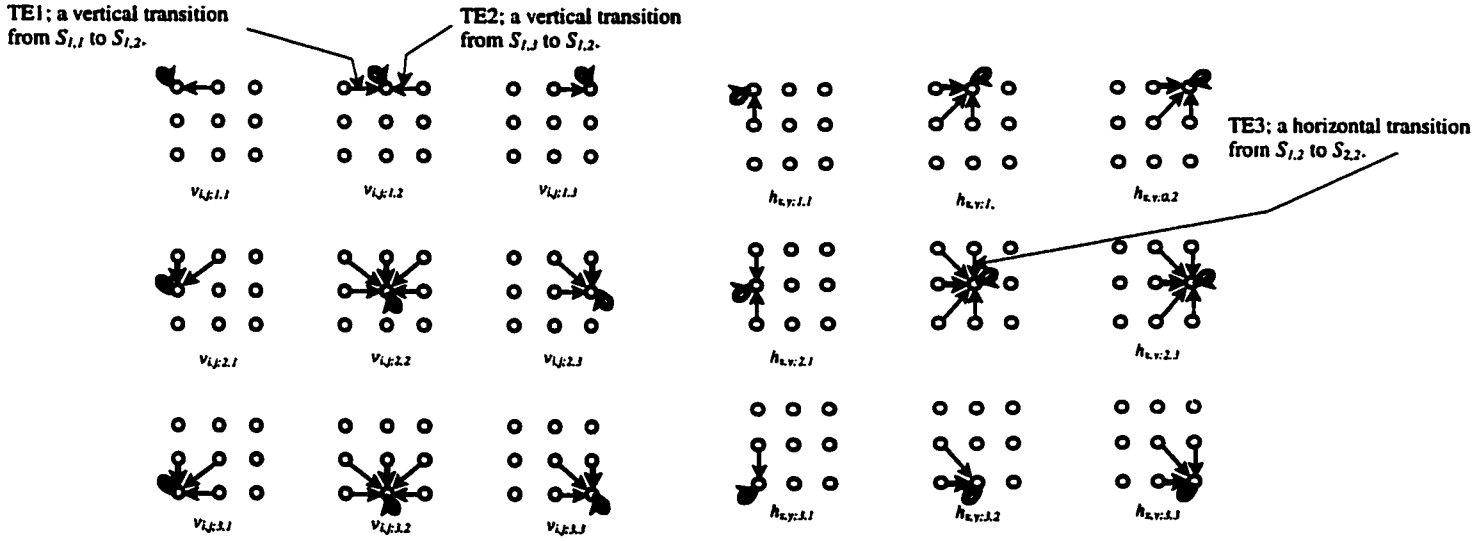
$$S_{i,j} \in VPPred(S_{m,n}) \text{ iff } ((m-1 \le i \le m) AND (n-1 \le j \le n+1)) \tag{3.3}$$

$$S_{x,y} \in HPPred(S_{m,n}) \text{ iff } ((m-1 \le x \le m+1) AND (n-1 \le y \le n)) \tag{3.4}$$

The restrictions above are depicted in Figure 3-5 for vertical and horizontal transitions based on a model of 3x3 states. In order to provide further illustration on how the model is used to recognize an image, an example based on a 12x9-block image is shown in Figure 3-6. The model is repeatedly drawn in each block location, as any of its states could be the one that has generated the block. A 2-D optimal path, produced by the recognition process, appears as a net whose nodes are generally approached by two transitions. Each node resembles the active states at the associated block. The active state, at a certain block, is the state that has most likely generated the feature block. A thick line

is added on the border between blocks that most likely have been generated by different states resulting in a number of closed regions, each of them is supported by one active state. A practical example of the outcome of the recognition process is shown in Figure 3-7.

It should be noted that once trained, the states of the model do not represent specific features of the face, rather each state will represent a homogenous area of the image exhibiting similar nature. The larger the number of states the more the homogeneity of the area.



a) Vertical transitions for 3x3 2-D b) Horizontal transitions for 3x3 2-D

Figure 3-5. Proposed 2-D HMM.
$v_{i,j;m,n}$ and $h_{x,y;m,n}$ are the vertical and horizontal transition probabilities from state $S_{i,j}$ and state $S_{x,y}$ to state $S_{m,n}$ respectively, and the indices i and j take values such that :

1) $m-1 \le i \le m$, $n-1 \le j \le n+1$ for $v_{i,j;m,n}$
2) $m-1 \le x \le m+1$, $n-1 \le y \le n$ for $h_{x,y;m,n}$

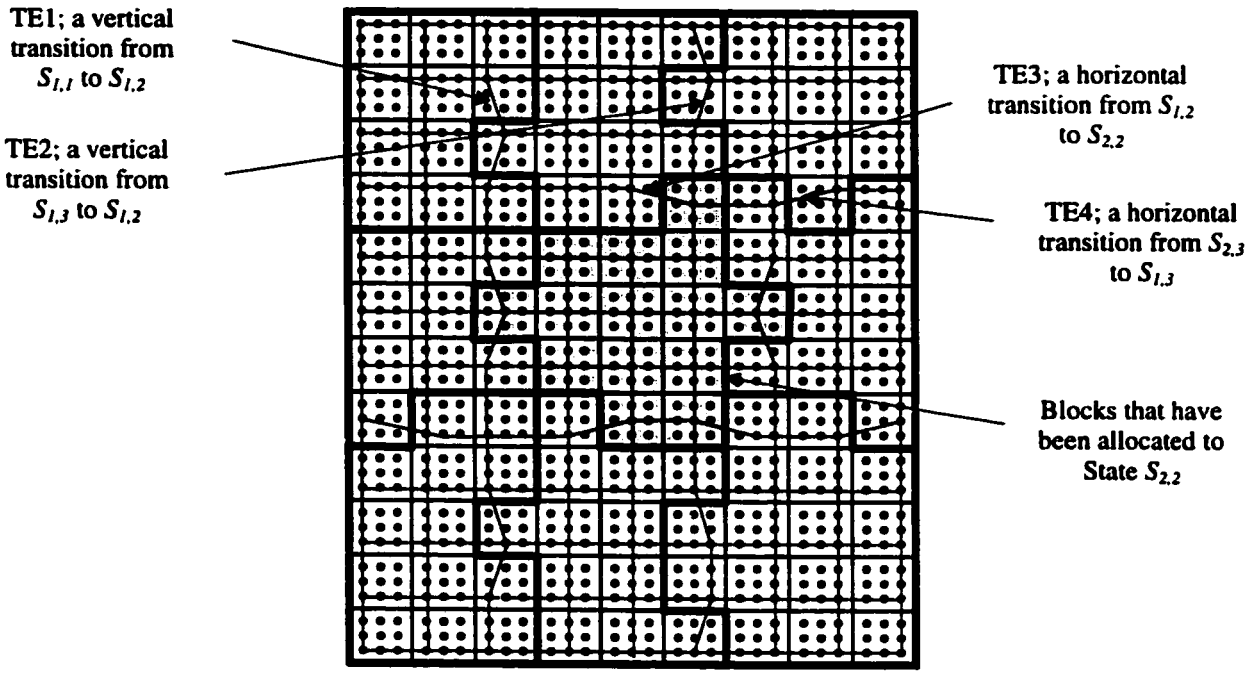


Figure 3-6. An example of state transitions based on 3x3-state model. TE1, TE2, TE3 and TE4 are some transition examples.

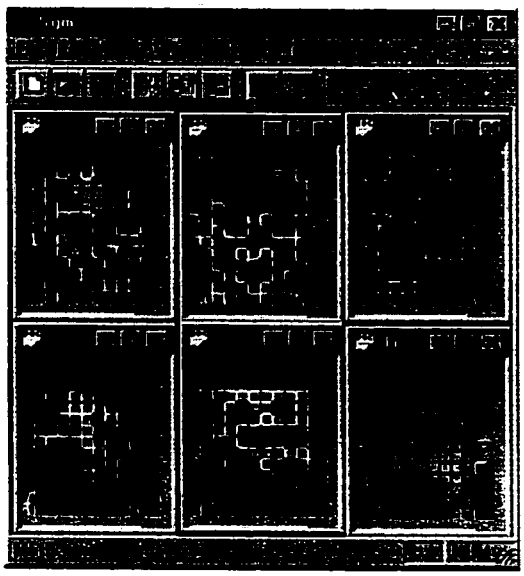


Figure 3-7. A practical example of state transitions:
The *black* vertical and horizontal line segments represent feature block borders where horizontal and vertical state transitions to the same state have taken place, respectively. The *white* vertical and horizontal line segments represent the feature block borders where horizontal and vertical state transitions to different state have taken place, respectively.

3.2.3 Framework of The Proposed System

The framework of the proposed 2-D HMM FR system includes three main tasks to be carried out, namely, the likelihood evaluation, decoding and learning. Since the implemented method is based on the Viterbi Algorithm, it is useful to jointly introduce the likelihood evaluation and state sequence decoding tasks as they are co-performed, then the learning method is given.

In the following section, the likelihood evaluation and state decoding for the proposed model are introduced. It will be shown that given the underlying assumptions in Section 3.2.1, the likelihood of the joint likelihood of an active state in a certain observation block can be given by the multiplication of two quantities. These two quantities are separately associated to the vertical and the horizontal past likelihood computations as depicted in relation (3.15). This relationship is exploited in the recursive form (3.23) as depicted in the modified Viterbi algorithm introduced later in this section. In Section 3.2.3.2, the estimation of the model parameters is given based on the state sequence decoding output from the modified Viterbi algorithm in Section 3.2.3.1.

3.2.3.1 Likelihood Evaluation and Decoding

The likelihood of observing the set of the feature vectors, $\mathbf{O}_{/T0,T1/}$, that constitutes the test image, given a model λ can be expressed as:

$$P(\mathbf{O}_{(k,l)} \mid \lambda) = \sum_{m,n=1,1}^{N_0,N_1} P(q_{k,l} = S_{m,n}, \mathbf{O}_{(k,l)} \mid \lambda) \qquad (3.5)$$

where $\mathbf{O}_{(k,l)} = \{\mathbf{O}_{r,s} : 1 \le r \le k, 1 \le s \le l\}$ and $\mathbf{O}_{r,s}$ is the feature vector of the block $B_{r,s}$.

For simplicity, the conditional probability given the model will be implicitly assumed.

Each term in the previous summation is given by:

$$P(q_{k,l} = S_{m,n}, \mathbf{O}_{\{k,l\}}) = P(q_{k,l} = S_{m,n}, \mathbf{O}_{<k,l>}) P(\mathbf{O}_{k,l} \mid q_{k,l} = S_{m,n}) \tag{3.6}$$

where $\mathbf{O}_{<k,l>} = \mathbf{O}_{\{k,l\}} - \mathbf{O}_{k,l}$

Based on assumption (1):

$$P(q_{k,l} = S_{m,n}, \mathbf{O}_{\{k,l\}}) = \sum_{i,x=1}^{N_0} \sum_{j,y=1}^{N_1} P(q_{k,l} = S_{m,n}, q_{k-1,l} = S_{i,j}, q_{k,l-1} = S_{x,y}, \mathbf{O}_{<k,l>}) \; P(\mathbf{O}_{k,l} \mid q_{k,l} = S_{m,n}) \tag{3.7}$$

where N_0 and N_1 are the vertical and horizontal number of states in the model, respectively.

$$P(q_{k,l} = S_{m,n}, \mathbf{O}_{\{k,l\}}) = \sum_{i,x=1}^{N_0} \sum_{j,y=1}^{N_1} P(q_{k-1,l} = S_{i,j}, q_{k,l-1} = S_{x,y} \mid q_{k,l} = S_{m,n}, \mathbf{O}_{<k,l>}) \cdot P(q_{k,l} = S_{m,n}, \mathbf{O}_{<k,l>}) P(\mathbf{O}_{k,l} \mid q_{k,l} = S_{m,n}) \tag{3.8}$$

Based on assumption (2) we have:

$$P(q_{k-1,l} = S_{i,j}, q_{k,l-1} = S_{x,y} \mid q_{k,l} = S_{m,n}, \mathbf{O}_{<k,l>}) = P(q_{k-1,l} = S_{i,j} \mid q_{k,l} = S_{m,n}, \mathbf{O}_{<k,l>}) P(q_{k,l-1} = S_{x,y} \mid q_{k,l} = S_{m,n}, \mathbf{O}_{<k,l>}) \tag{3.9}$$

Substituting in (3.8):

$$P(q_{k,l} = S_{m,n}, \mathbf{O}_{\{k,l\}}) = \sum_{i,x=1}^{N_0} \sum_{j,y=1}^{N_1} P(q_{k-1,l} = S_{i,j} \mid q_{k,l} = S_{m,n}, \mathbf{O}_{<k,l>}) P(q_{k,l-1} = S_{x,y} \mid q_{k,l} = S_{m,n}, \mathbf{O}_{<k,l>}) \cdot P(q_{k,l} = S_{m,n}, \mathbf{O}_{<k,l>}) P(\mathbf{O}_{k,l} \mid q_{k,l} = S_{m,n}) \tag{3.10}$$

However, the quantities $P(q_{k-1,l}=S_{i,j} \mid q_{k,l}=S_{m,n}, \mathbf{O}_{<k,l>})$ and $P(q_{k,l-1}=S_{x,y} \mid q_{k,l}=S_{m,n}, \mathbf{O}_{<k,l>})$ depend on non-causal relations. In order to maintain the recursive form of the solution, Bayes' rule [37] is used to replace them with their causal counterparts, $P(q_{k,l}=S_{m,n} \mid q_{k-1,l}=S_{i,j}, \mathbf{O}_{<k,l>})$ and $P(q_{k,l}=S_{m,n} \mid q_{k,l-1}=S_{x,y}, \mathbf{O}_{<k,l>})$, respectively.

$$P(q_{k,l} = S_{m,n}, \mathbf{O}_{\{k,l\}}) = \sum_{i,x=1}^{N_0} \sum_{j,y=1}^{N_1} \left(\frac{P(q_{k,l} = S_{m,n} \mid q_{k-1,l} = S_{i,j}, \mathbf{O}_{<k,l>}) P(q_{k-1,l} = S_{i,j}, \mathbf{O}_{<k,l>})}{P(q_{k,l} = S_{m,n}, \mathbf{O}_{<k,l>})} \right.$$

$$\cdot \frac{P(q_{k,l}=S_{m,n} \mid q_{k,l-1}=S_{x,y}, \mathbf{O}_{<k,l>})P(q_{k,l-1}=S_{x,y}, \mathbf{O}_{<k,l>})}{P(q_{k,l}=S_{m,n}, \mathbf{O}_{<k,l>})}\Bigg)$$

$$\cdot P(q_{k,l}=S_{m,n}, \mathbf{O}_{<k,l>})P(\mathbf{O}_{k,l} \mid q_{k,l}=S_{m,n}) \tag{3.11}$$

$$P(q_{k,l}=S_{m,n}, \mathbf{O}_{(k,l)}) = \sum_{i,x=1}^{N_0}\sum_{j,y=1}^{N_1}\Big(P(q_{k,l}=S_{m,n} \mid q_{k-1,l}=S_{i,j}, \mathbf{O}_{<k,l>})P(q_{k-1,l}=S_{i,j}, \mathbf{O}_{<k,l>})$$

$$\cdot P(q_{k,l}=S_{m,n} \mid q_{k,l-1}=S_{x,y}, \mathbf{O}_{<k,l>})P(q_{k,l-1}=S_{x,y}, \mathbf{O}_{<k,l>})\Big)$$

$$\cdot \frac{P(\mathbf{O}_{k,l} \mid q_{k,l}=S_{m,n})}{P(q_{k,l}=S_{m,n}, \mathbf{O}_{<k,l>})} \tag{3.12}$$

Comparing (3.6) and (3.12), it can be concluded that:

$$P(q_{k,l}=S_{m,n}, \mathbf{O}_{<k,l>}) = \sum_{i,x=1}^{N_0}\sum_{j,y=1}^{N_1}\Big(P(q_{k,l}=S_{m,n} \mid q_{k-1,l}=S_{i,j}, \mathbf{O}_{<k,l>})P(q_{k-1,l}=S_{i,j}, \mathbf{O}_{<k,l>})$$

$$\cdot P(q_{k,l}=S_{m,n} \mid q_{k,l-1}=S_{x,y}, \mathbf{O}_{<k,l>})P(q_{k,l-1}=S_{x,y}, \mathbf{O}_{<k,l>})\Big)$$

$$/\, P(q_{k,l}=S_{m,n}, \mathbf{O}_{<k,l>}) \tag{3.13}$$

Substituting back in (3.12), we get:

$$P(q_{k,l}=S_{m,n}, O_{(k,l)}) = \sqrt{\sum_{i,j=1,1}^{N_0,N_1} P(q_{k,l}=S_{m,n} \mid q_{k-1,l}=S_{i,j}, \mathbf{O}_{<k,l>})P(q_{k-1,l}=S_{i,j}, \mathbf{O}_{<k,l>})}$$

$$\cdot \sqrt{\sum_{x,y=1,1}^{N_0,N_1} P(q_{k,l}=S_{m,n} \mid q_{k,l-1}=S_{x,y}, \mathbf{O}_{<k,l>})P(q_{k,l-1}=S_{x,y}, \mathbf{O}_{<k,l>})}$$

$$\cdot P(\mathbf{O}_{k,l} \mid q_{k,l}=S_{m,n}) \tag{3.14}$$

The probability of the current active state, $q_{k,l}$, to be the state $S_{m,n}$ depends only on the past active state in the vertical direction, $q_{k-1,l}$, and on the past active state in the horizontal direction, $q_{k,l-1}$. Assume that the choice of the active state at the $(k\text{-}1)^{th}$ row is not affected by the observation blocks at the k^{th} row (i.e. the future row of blocks w.r.t.

$q_{k-1,l}$). Assume also that the choice of the active state at the $(l-1)^{th}$ column is not affected by the observation blocks at the l^{th} column (i.e. the future column of blocks w.r.t. $q_{k,l-1}$). Thus, $P(q_{k,l}=S_{m,n},\mathbf{O}_{(k,l)})$ can be expressed in terms of $P(q_{k-1,l}=S_{i,j},\mathbf{O}_{(k-1,l)})$ and $P(q_{k,l-1}=S_{x,y},\mathbf{O}_{(k,l-1)})$. Having the recursive solution in mind, at the time of evaluating $P(q_{k,l}=S_{m,n},\mathbf{O}_{(k,l)})$ at the k^{th} row and the l^{th} column, $P(q_{k-1,l}=S_{i,j},\mathbf{O}_{(k-1,l)})$ and $P(q_{k,l-1}=S_{x,y},\mathbf{O}_{(k,l-1)})$, are already obtained at the previous row and column, respectively.

Thus, the equation above can be reduced to:

$$P(q_{k,l}=S_{m,n},O_{(k,l)}) \propto \sqrt{\sum_{i,j=1,1}^{N_0,N_1} P(q_{k,l}=S_{m,n} \mid q_{k-1,l}=S_{i,j})P(q_{k-1,l}=S_{i,j},\mathbf{O}_{(k-1,l)})}$$

$$\cdot\sqrt{\sum_{x,y=1,1}^{N_0,N_1} P(q_{k,l}=S_{m,n} \mid q_{k,l-1}=S_{x,y})P(q_{k,l-1}=S_{x,y},\mathbf{O}_{(k,l-1)})}$$

$$\cdot P(\mathbf{O}_{k,l} \mid q_{k,l}=S_{m,n}) \qquad (3.15)$$

The conditional probabilities $P(q_{k,l}=S_{m,n}|q_{k-1,l}=S_{i,j})$ and $P(q_{k,l}=S_{m,n}|q_{k,l-1}=S_{x,y})$ form the vertical and the horizontal transition matrices, $\mathbf{V}=\{v_{i,j;m,n}\}$ and $\mathbf{H}=\{h_{x,y;m,n}\}$, respectively, and are obtained through the training phase as it is depicted in the next section. The probability of the current observation given the state, $P(\mathbf{O}_{k,l}|q_{k,l}=S_{m,n})$, is obtained from $b_{m,n}(\mathbf{O})$, the observation PDF of the state $S_{m,n}$, that is given by:

$$b_{m,n}(\mathbf{O})=\sum_{g=1}^{G_s} c_{m,n}^{(g)} N(\mathbf{O},\boldsymbol{\mu}_{m,n}^{(g)},\boldsymbol{\Sigma}_{m,n}^{(g)}), \quad 1\le m\le N_0, 1\le n\le N_1 \qquad (3.16)$$

where $c_{m,n}^{(g)}$, $\boldsymbol{\mu}_{m,n}^{(g)}$, and $\boldsymbol{\Sigma}_{m,n}^{(g)}$ are the weight, the mean vector and the covariance matrix of the g^{th} Gaussian component in the observation PDF of the state $S_{m,n}$, respectively, and G_s is the number of Gaussian components.

Relation (3.15) above is recursively applied throughout the image starting from the left-upper corner to the right-lower corner to estimate the likelihood of the test image. However, the consecutive multiplication of the small probability values results in an underflow problem. Re-scaling is a well-known solution to this problem but it comes with an extra computational cost. We choose to perform the computations in the log-domain to avoid the problem of underflow while keeping the system complexity low. As the multiplication process becomes an addition, the *max(.)* function can be used to replace the addition process. The relative error that results from the latter is negligible due to the order of the added numbers. The image likelihood is obtained for the best state sequence that is determined by backtracking pointers that are assigned to the candidate vertical and horizontal predecessor at each feature block. The procedure of likelihood evaluation and state sequence decoding is detailed below, where relation (3.15) is exploited recursively in (3.23).

Define the state score, $\delta_{k,l}(m,n)$, likelihood of the past and the current observations maximized over the past state sequence terminating with $S_{m,n}$ at the feature block $B_{k,l}$, as:

$$\delta_{k,l}(m,n) = \max_{q_{1,1},\ldots,q_{k-1,l},q_{k,l-1}} P[q_{1,1},\ldots,(q_{k,l}=S_{m,n}),O_{1,1},\ldots,O_{k,l} \mid \lambda]$$

Define the initial vertical and horizontal state probabilities, $\pi v_{m,n}$ and $\pi h_{m,n}$, respectively, as:

$$\begin{aligned} \pi v_{m,n} &= P(q_{1,l}=S_{m,n}) \\ \pi h_{m,n} &= P(q_{k,1}=S_{m,n}) \end{aligned} \tag{3.17}$$

1) Initialization:

$$\delta_{1,1}(m,n) = \sqrt{\pi v_{m,n}\pi h_{m,n}} \cdot b_{m,n}(\mathbf{O}_{1,1}) \tag{3.18}$$

2) Forward recursion:

First column:

$$\delta_{k,1}(m,n) = \sqrt{\max_{i,j}[\delta_{k-1,1}(i,j)v_{i,j;m,n}]\pi h_{m,n}} \cdot b_{m,n}(\mathbf{O}_{k,1}), \quad 1 \le k \le T_0 \tag{3.19}$$

$$\zeta_{k,0}(m,n) = \arg\max_{i,j}[\delta_{k-1,1}(i,j)v_{i,j;m,n}] \tag{3.20}$$

First row:

$$\delta_{0,l}(m,n) = \sqrt{\pi v_{m,n} \cdot \max_{x,y}[\delta_{1,l-1}(x,y)h_{x,y;m,n}]}\, b_{m,n}(\mathbf{O}_{1,l}), \quad 1 \le l \le T_1 \tag{3.21}$$

$$\psi_{0,l}(m,n) = [\arg\max_{x,y} \delta_{1,l-1}(x,y)h_{x,y;m,n}] \tag{3.22}$$

The rest of the image:

$$\delta_{k,l}(m,n) = \sqrt{\max_{i,j}[\delta_{k-1,l}(i,j)v_{i,j;m,n}]} \cdot \sqrt{\max_{x,y}[\delta_{k,l-1}(x,y)h_{x,y;m,n}]} \cdot b_{m,n}(\mathbf{O}_{k,l}) \tag{3.23}$$

$$\begin{aligned} \zeta_{k,l}(m,n) &= [\arg\max_{i,j} \delta_{k-1,l}(i,j)v_{i,j;m,n}] \\ \psi_{k,l}(m,n) &= [\arg\max_{x,y} \delta_{k,l-1}(x,y)h_{x,y;m,n}] \end{aligned} \tag{3.24}$$

where i,j are chosen such that $S_{i,j} \in VPred(S_{m,n})$,

x,y are chosen such that $S_{x,y} \in HPred(S_{m,n})$,

$1 \le m,i,x \le N_0$, $1 \le n,j,y \le N_1$, $1 \le k \le T_0$, $1 \le l \le T_1$

N_0 and N_1 are the vertical and horizontal number of states respectively,

T_0 and T_1 are the vertical and horizontal number of observation blocks respectively,

$v_{i,j;m,n}$ is the probability of the vertical transition from state $S_{i,j}$ to state $S_{m,n}$,

$h_{x,y;m,n}$ is the probability of the horizontal transition from state $S_{x,y}$ to state $S_{m,n}$,

$\zeta_{k,l}(m,n)$ and $\Psi_{k,l}(m,n)$ are the vertical and horizontal predecessors of the state $S_{m,n}$ at the block $B_{k,l}$.

3) Termination:

$$P(\mathbf{O}, Q^* \mid \lambda) = \max_{m,n}[\delta_{T_0,T_1}(m,n)]$$
$$q_{T_0,T_1} = \arg\max_{m,n}[\delta_{T_0,T_1}(m,n)] \tag{3.25}$$

where Q^* is the optimal state sequence.

4) Backtracking:

Last column:

$$q_{k,L-1} = \zeta_{k+1,l}(q_{k,l+1}), \quad k = T_0 - 1, T_0, -2, \ldots, 1 \tag{3.26}$$

Last row:

$$q_{K-1,l} = \psi_{k,l+1}(q_{k,l+1}) \quad l = T_1 - 1, L-2, \ldots, 1 \tag{3.27}$$

The rest of the image:

$$q_{k,l} = \underset{\zeta_{k+1,l}(q_{k+1,l}),\psi_{k,l+1}(q_{k,l+1})}{\arg\max} [\delta_{k,l}(\zeta_{k+1,l}(q_{k+1,l})), \delta_{k,l}(\psi_{k,l+1}(q_{k,l+1}))], \quad k = T_0 - 1, T_0 - 2, \ldots, 1, l = T_1 - 1, T_1 - 2, \ldots, 1$$

(3.28)

The best vertical predecessor of the $S_{m,n}$ at the block $B_{k,l}$, $\zeta_{k,l}(m,n)$, is obtained by Equation (3.20) and Equation (3.24), while the best horizontal predecessor, $\Psi_{k,l}(m,n)$, is obtained by Equation (3.22) and Equation (3.24) in the forward phase. Recall that in the 1-D Viterbi, a state $S_{m,n}$ at a block $B_{k,l}$, has only one predecessor. The predecessor at a block, say B_{k-1}, in the 1-D Viterbi becomes an active state if state S_m is chosen to be the active state at the following block B_k. In other words, there is only one candidate active state at each block, which is determined once the active state in the following block is determined in the 1-D Viterbi backtracking.

Considering the 2-D backtracking case, at each block, say $B_{k,l}$, exist a vertical predecessor of the active state at the block $B_{k+1,l}$, and a horizontal predecessor of the active state at the block $B_{k,l+1}$. This complicates the 2-D backtracking since these two

predecessors may not coincide. Having more than one active state at a block means having more than state sequence, which results in a Non-Polynomial number of state sequences. Assuming that the state that has higher score among the two predecessors results in a better state sequence, it is, consequently, appointed as active state while the other predecessor is dismissed as depicted in Equation (3.28).

The output of the decoding stage is used to estimate the frequency of the events, e.g. state transitions, which are involved in calculating the new model parameters. A simple example of the proposed model recognizing a simple image is given in Appendix B.

3.2.3.2 Learning Phase

The proposed 2-D HMM is trained through an iterative process that maximizes the likelihood of the training data being generated by the model [9] based on the modified version of Viterbi Algorithm given above.

Initialization:

- The training data images are uniformly segmented into patches which are allocated to the corresponding model states. Accordingly, the initial state PDF Gaussians are estimated by clustering the training data that are allocated to the state using the Linde, Buzo and Gray (LBG) algorithm [48].
- The state transition probabilities and the initial state probabilities are initially assumed uniform (equal).

Iteration:

- The modified Viterbi Algorithm is applied to the training data and the average log-likelihood is obtained.
- The model parameters are updated based on the decoded state sequence.

- If the relative change in the average log-likelihood is less than a certain threshold, ε, repeat the iteration, otherwise, stop.

Model parameters are modified based on the decoding phase output, i.e. the active state sequence of the training observations, referred to as *Decision-Directed* learning in [18], or *concept of counting event occurrence* in [9].

Two initial probability vectors, $\Pi_V = \{\pi v_{m,n}\}$ and $\Pi_H = \{\pi h_{m,n}\}$, two transition probability matrices, $\mathbf{V} = \{v_{x,y;m,n}\}$ and $\mathbf{H} = \{\pi h_{i,j;m,n}\}$, and the observation PDF of within the state $\mathbf{B}=\{b_{m,n}(\mathbf{O})\}$ have to be updated while training.

Updating the initial state probabilities:

$$\overline{\pi v}_{m,n} = \text{expected number of times state } S_{m,n} \text{ is active in the first row.} \tag{3.29}$$

$$\overline{\pi h}_{m,n} = \text{expected number of times state } S_{m,n} \text{ is active in the first column.} \tag{3.30}$$

Updating the state transition probabilities:

$$\overline{h}_{i,j;m,n} = \frac{\text{expected number of horizontal transitions from state } S_{i,j} \text{ to } S_{m,n}}{\text{expected number of horizontal transitions from state } S_{i,j}} = \frac{\sum_{\forall \text{ training images}} \left[\sum_{k=1}^{T_0} \sum_{l=2}^{T_1} I_{i,j}(\mathbf{O}_{k,l-1}) \cdot I_{m,n}(\mathbf{O}_{k,l}) \right]}{\sum_{\forall \text{ training images}} \left[\sum_{k=1}^{T_0} \sum_{l=2}^{T_1} I_{x,y}(\mathbf{O}_{k,l-1}) \right]} \tag{3.31}$$

$$\overline{v}_{x,y;m,n} = \frac{\text{expected number of vertical transitions from state } S_{x,y} \text{ to } S_{m,n}}{\text{expected number of vertical transitions from state } S_{x,y}} = \frac{\sum_{\forall \text{ training images}} \left[\sum_{k=2}^{T_0} \sum_{l=1}^{T_1} I_{x,y}(\mathbf{O}_{k-1,l}) \cdot I_{m,n}(\mathbf{O}_{k,l}) \right]}{\sum_{\forall \text{ training images}} \left[\sum_{k=2}^{T_0} \sum_{l=1}^{T_1} I_{x,y}(\mathbf{O}_{k-1,l}) \right]} \tag{3.32}$$

where $I_{m,n}(\mathbf{O}_{k,l})$ is an Indicator function that is defined as :

$$I_{m,n}(\mathbf{O}_{k,l}) = \begin{cases} 1 & , q_{k,l} = S_{m,n} \\ 0 & , otherwise \end{cases} \tag{3.33}$$

Updating the observation PDFs:

$$\overline{\boldsymbol{\mu}}_{m,n}^{(g)} = \frac{\sum_{\forall \text{ training images}} \left[\sum_{k=1}^{T_0} \sum_{l=1}^{T_1} I_{m,n}^{(g)}(\mathbf{O}_{k,l}) \cdot \mathbf{O}_{k,l} \right]}{\sum_{\forall \text{ training images}} \left[\sum_{k=1}^{T_0} \sum_{l=1}^{T_1} I_{m,n}^{(g)}(\mathbf{O}_{k,l}) \right]} \tag{3.34}$$

$$\overline{\boldsymbol{\Sigma}}_{m,n}^{(g)} = \frac{\sum_{\forall \text{ training images}} \left[\sum_{k=1}^{T_0} \sum_{l=1}^{T_1} I_{m,n}^{(g)}(\mathbf{O}_{k,l}) \cdot (\mathbf{O}_{k,l} - \boldsymbol{\mu}_{m,n}^{(g)})(\mathbf{O}_{k,l} - \boldsymbol{\mu}_{m,n}^{(g)})^T \right]}{\sum_{\forall \text{ training images}} \left[\sum_{k=1}^{T_0} \sum_{l=1}^{T_1} I_{m,n}^{(g)}(\mathbf{O}_{k,l}) \right]} \tag{3.35}$$

$$\overline{c}_{m,n}^{(g)} = \frac{\sum_{\forall \text{ training images}} \left[\sum_{k=1}^{T_0} \sum_{l=1}^{T_1} I_{m,n}^{(g)}(\mathbf{O}_{k,l}) \right]}{\sum_{\forall \text{ training images}} \left[\sum_{k=1}^{T_0} \sum_{l=1}^{T_1} I_{m,n}(\mathbf{O}_{k,l}) \right]} \tag{3.36}$$

where the indicator function $I_{m,n}^{(g)}(\mathbf{O}_{k,l})$ is given by:

$$I_{m,n}^{(g)}(\mathbf{O}_{k,l}) = \begin{cases} I_{m,n}(\mathbf{O}_{k,l}) & , g = \arg\max_{1 \le g' \le G_s} \left(c_{m,n}^{(g')} N(\mathbf{O}_{k,l}, \boldsymbol{\mu}_{m,n}^{(g')}, \boldsymbol{\Sigma}_{m,n}^{(g')}) \right) \\ 0 & , otherwise \end{cases} \tag{3.37}$$

3.3 Summary of the Proposed 2-D HMM FR System

The proposed system works in two phases, the training (learning) phase and the testing (recognition) phase. In this section, we will summarize the main steps as derived above both phases will be illustrated in brief.

Given a set of training dataset, the facial images that belong to one person are uniformly segmented. The number of segments in each image is equal to the number of states in the model with each segment corresponding to one state in the model. In order to

obtain an initial estimate of the PDF of each state, the observation blocks in each segment across the images of the same person are collected in one pool. This pool of observation blocks is used to obtain the initial PDF of this state corresponding to this segment. For example, for a 4-state model (2 horizontal states by 2 vertical states), each of the training images is segmented into 4 equal segments: the top-left segment, the top-right segment, the bottom-left segment and the bottom-right segment. Each of these segments includes a number of observation blocks. In order to obtain the PDF of the top-left state, the observation blocks are collected from all the top-left segments in all the training images of this person (the owner of the model). Once the observation blocks are collected in one pool, the iterative LBG algorithm is applied to the feature vectors that correspond to the observation blocks in the pool. The LBG algorithm clusters these data into a number of clusters that is equal to the number of kernels in the PDF. Each of these clusters has a centroid (mean vector) and a covariance matrix. For simplicity, off-diagonal covariance coefficients are neglected and the covariance matrix becomes a variance vector. The mean vector of each of the multivariate Gaussian kernels is set equal to a centroid of a cluster. The variance vector of that multivariate Gaussian kernel is set equal to a variance vector of the cluster. In other words, each cluster is used to initialize a PDF kernel. The ratio of the number of feature vectors in each cluster to the total number of feature vectors of the dataset of the model is used as a weight to the corresponding kernel in the weighted-sum PDF. Obviously, the sum of these weights must be 1. Similarly the sum of these ratios must be equal to 1.

The state transition probabilities are initially set to the inverse of the number of possible state transitions. Once the initial model parameters are set to their initial values, a forced

recognition run is carried out. In this forced recognition run, the model is forced to recognize its own training data. In other words, the model is forced to recognize faces that belong to its owner. The details of the recognition process will follow. During the forced recognition process the way the model recognizes each image is recorded, particularly *which state represents which observation block using which kernel.* This information is substituted in equations (3.29) to (3.37) in order to obtain a new estimate of the model parameters. Having the new estimate of the model parameters, a new forced recognition run is repeated. Each time the forced recognition process is repeated, the average likelihood over all the models in the system is recorded. The relative improvement in the average overall likelihood is monitored and when it is less than a certain threshold, the training process is aborted and the most recent estimate that came out of the most recent forced recognition run is honored as the best estimate of the model.

The recognition process is concerned with computing the likelihood of an image given a certain model. In the learning phase, the model recognizes an image of its own training dataset using the equations (3.17) to (3.25) and realizes how it has been recognized using the backtracking equations (3.26) to (3.28).

It should be noted that in the testing phase, only the equations (3.17) to (3.25) are used in order to obtain the likelihood of an image to be generated by a certain model. There is no need to use the backtracking equations (3.26) to (3.28) in the testing phase. In order to recognize a given test image in the testing phase, the recognition process is repeated for each model in the system and the one with highest likelihood is chosen to represent the test image.

3.4 Complexity of The Proposed 2-D HMM FR System

Time required for a database search is an essential factor for its commercial success. Search time depends on several factors including the speed of the machine, the size of the database, the complexity of the algorithm, and the efficiency of the implementation. Machine speed control is beyond the scope of the current work and it is dictated by the hardware conditions. The search time is directly proportional to the database size. In the current work, the size of the database we have used is 40 persons, which is the same database used by all the systems with which we will compare the proposed system. The implementation efficiency is quite dependant on the implementation environment and source code. *C language* is still one of the fastest means to implement such algorithms. Although mathematical packages like *Matlab* provide time optimized tools by supplying the commonly used algorithms in non-interpreting form, the interpreter nature is dominant as opposed to the compiler nature of C language. This leaves us with the algorithm complexity, which we will now consider in detail.

It is useful to break the face recognition task into sub-tasks in order to estimate the overall system complexity. The core of the proposed system can be divided into three main tasks; feature extraction, observation layer computations and hidden layer computations. The complexity of HMM-based algorithms is always addressed from the hidden layer point of view. This is due to the fact that, the impact of the algorithm on the complexity mainly appears on the hidden layer, irrespective of the nature the observations and the type of the state PDF. However, the observation layer and the feature extraction layers are not trivial tasks in most of cases. Thus, we still need to shed

some light on the complexity of these two, given the nature of the facial images as observations.

a) Hidden Layer:

The hidden layer calculations are concerned with the recurrent computations of the likelihood of the observation, for a given model. The computational complexity of this layer solely depends on the algorithm that is used to evaluate the likelihood, given the conditional state likelihood of the observation block. Based on the discussion in section 2.1, the approximate upper-bound of the number of additions and multiplications is $2N^2T$, each for the Forward-Backward algorithm. Practically, additional calculations are normally needed for likelihood scaling to avoid numeric underflow. Considering the Viterbi-based likelihood evaluation algorithm given earlier in this chapter, the calculations of the hidden layer are carried out in the log-domain with no multiplications. The upper-bound number of additions is approximately $2N^2T$. A lower upper-bound for the proposed model can be expressed as:

$$C^{+}_{Hidden\ Layer} \cong (T_1 T_0)(N_1 N_0)(M_{pr}+N_{pr}+1) \tag{3.38}$$

where M_{pr} and N_{pr} are the average numbers of vertical and horizontal predecessors per state, respectively, which are given by:

$$M_{pr} = \frac{\sum_{i=1}^{N_0}\sum_{j=1}^{N_1} m_{pr}(i,j)}{N_0 N_1} \tag{3.39}$$

$$N_{pr} = \frac{\sum_{i=1}^{N_0}\sum_{j=1}^{N_1} n_{pr}(i,j)}{N_0 N_1} \tag{3.40}$$

where $m_{pr}(i,j)$ and $n_{pr}(i,j)$ are the numbers of the vertical and horizontal predecessors of the state $S_{i,j}$, respectively.

Considering the model topology described in section 3.2.2 with $N_0=6$ and $N_1=2$,

M_{pr}=(1x(2x2)+5x(2x4))/12=44/12 predecessors.

N_{pr}=(1x(2+4)+4x(3+6)+ 1x(2+4))/12=48/12 predecessors.

Thus,

$C_{Hidden\ Layer}$=17,472 additions.

b) Observation Layer:

The observation layer calculations are concerned with the computations of the likelihood of each individual observation block $O_{k,l}$ to be generated by each individual state, i.e. $b_{m,n}(O_{k,l})$. Those calculations are independent on the hidden layer calculations while the opposite is not true. The computational load of the observation layer depends on the type of the state PDF as well as the size of the feature vector. Considering the continious PDF HMM, where G_s Gaussians kernels are used, $b_{m,n}(O_{k,l})$ is given by:

$$b_{m,n}(\mathbf{O}_{k,l}) = \sum_{g=1}^{G_s} \frac{c_{m,n}^{g}}{[2\pi]^{\frac{V}{2}} \boldsymbol{\Sigma}^{\frac{1}{2}}} \exp\left(-\frac{(\mathbf{O}_{k,l} - \boldsymbol{\mu}_{m,n}) \boldsymbol{\Sigma}^{-1} (\mathbf{O}_{k,l} - \boldsymbol{\mu}_{m,n})^{T}}{2} \right) \tag{3.41}$$

where V is the number of DCT coefficients in the feature vector.

The DCT is known by its property of nearly decorrelating natural images, which produces a near-diagonal covariance matrix. If the off-diagonal elements are ignored, the joint probability of the feature vector can be obtained by multiplying the Gaussian PDFs of the vector elements.

$$b_{m,n}(\mathbf{O}_{k,l}) = \sum_{g=1}^{G_s} \frac{c_{m,n}^{g}}{[2\pi]^{\frac{V}{2}} \prod_{v=1}^{V} \sigma_v^{(g)}} \prod_{v=1}^{V} \exp\left(-\frac{(o_{k,l} - \mu_{m,n}^{(g)})^2}{2(\sigma_v^{(g)})^2} \right) \tag{3.42}$$

For consistency, the observation layer computations are carried out in the log-domain and directly plugged-in the hidden layer computations. Hence the quantity $Log(b_{m,n}(O_{k,l}))$ is obtained by:

$$Log\left(b_{m,n}(\mathbf{O}_{k,l})\right) = \max_{1\le g\le G_s}\left[\frac{1}{2}\left(Log(\tilde{c}^{\,g}_{m,n}) - \sum_{v=1}^{V}\frac{(o_{k,l}-\mu^{(g)}_{m,n})^2}{\left(\sigma^{(g)}_{v}\right)^2}\right)\right] \tag{3.43}$$

where $\tilde{c}^{\,g}_{m,n} = \dfrac{\left(c^{g}_{m,n}\right)^2}{[2\pi]^V \prod_{v=1}^{V}\left(\sigma^{(g)}_{v}\right)^2}$

The number of multiplications and additions, respectively, are:

$$C^{*}_{Observation\ Layer} = (T_0T_1)\,(N_0N_1)G_s(2V) \tag{3.44}$$

$$C^{+}_{Observation\ Layer} = (T_0T_1)\,(N_0N_1)G_s(2V) \tag{3.45}$$

For $T_0=14$, $T_1=12$, $N_0=6$, $N_1=2$, $G_s=4$, and $V=9$, the number of operations is:

$C_{Observation\ Layer}$= 145,152 operations.

c) Feature Extraction:

The 2-D DCT can be obtained through the Discrete Fourier Transform (DFT) using a Fast algorithms such as Fast Fourier Transform (FFT). The FFT of T_0T_1 observation blocks of M_pxN_p pixels requires a number of complex additions and multiplications given by:

$$C^{+}_{Block\_FE} = (T_0T_1)[N_p(M_p\,log_2M_p) + M_p(N_p\,log_2N_p)] \tag{3.46}$$

$$C^{*}_{Block\_FE} = (T_0T_1)[N_p(M_p.log_2M_p) + M_p(N_p.log_2.N_p)]/2 \tag{3.47}$$

For $T_0=14$, $T_1=12$, $M_p=8$ and $N_p=8$, the number of operations is:

$C^{+}_{Block\_FE}$= 64,512 complex addition.

$C^{*}_{Block\_FE}$= 32,256 complex multiplication.

However, if the images are JPEG coded, no transformation is required. Minimal processing is needed to retrieve the entropy-coded block-based 2-D DCT coefficients, which is basically a table look-up process.

Thus, the total computational load for the local database search is:

$$C_{MAP} = C_{\text{Feature Extraction}} + N_{pf} * (C_{\text{Observation Layer}} + C_{\text{Hidden Layer}}) \quad (3.31)$$

where N_{pf} is the number of persons included in the local database.

3.5 Performance of The Proposed System

The proposed 2-D HMM FR system is examined for different values of the structural parameters, namely number of states per model and number of kernels per state PDF. Additional images are provided for training in order to compensate for the shortage of the training feature blocks of the model states due to the non-overlapping restriction. The 40 facial HMMs are trained by 9 facial images per person while one testing image is used per person. The test is repeated 5 times with different test images and the results are averaged over a total of 200 test images for 40 persons. Test images are not members of the training data set at any time. The feature vector is composed of the 2-D DCT coefficients of the lowest nine frequency bands of the non-overlapped 8x8 pixels observation blocks. The number of Gaussian kernels of the state PDFs varies between 2 and 64 in this experiment.

Number of States	Number of PDF Kernels					
	2	4	8	16	32	64
4[2x2]	66	82.5	90	94.5		88
6[2x3]	82.5	87.5	92			79.5
6[3x2]	79	90.5			94	81
8[4x2]	90	94			92	63.5
9[3x3]	86.5	94.5			86	46
10[5x2]	93	93.5			76	34.5
12[4x3]	93.5				73.5	26.5
12[6x2]	93				77.5	26
14[7x2]	94.5				72.5	23.5
15[5x3]	94.5			86.5	33	6
18[6x3]				88.5	31.5	4.5
21[7x3]				89	28	4

Table 3-1. Recognition rate for different numbers of states and PDF kernels.

The results in Table 3-1 show that the performance generally improves with the increase of the number of kernels of the state PDFs to a maximum then it starts to degrade at some point. This is a typical behavior that was shown by the various models throughout this experiment. The performance improvement within the first portion is due to the increase in the degrees of freedom of the state PDF enabling better representation of the data space.

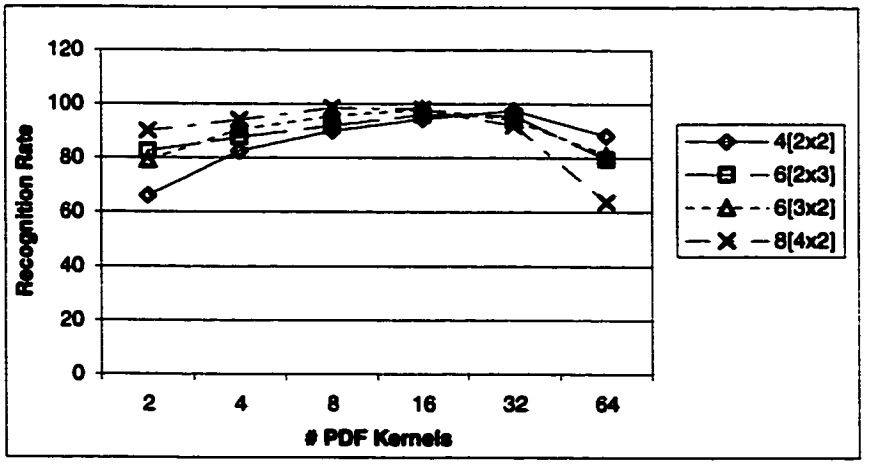


Figure 3-8. Recognition rate for different numbers of PDF kernels and few states.

The performance tends to degrade afterwards because of the overfitting problem, i.e. the model becomes too training-data-related to be able to recognize the testing examples. The maximum point takes place at relatively large number of kernels for small model sizes, i.e. models with fewer states, as depicted in Figure 3-8. For example, 2x2-state model reaches its maximum for the 32 kernels due to the need for flexible state PDF to handle the wide range of data represented by each state. The increased number of kernels helps improve the resolution achievable with fewer states.

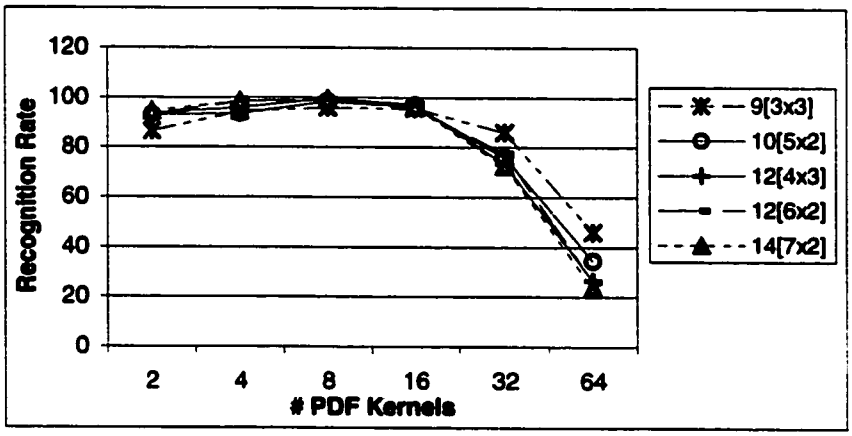


Figure 3-9. Recognition rate for different numbers of PDF kernels and moderate numbers of states.

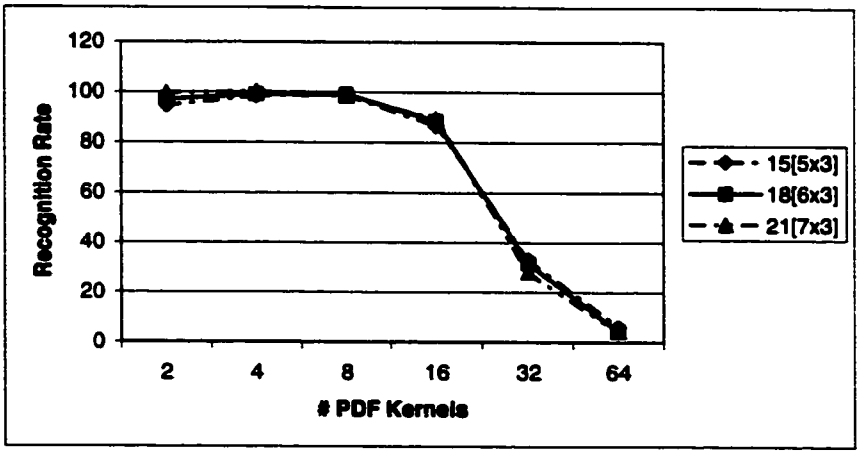


Figure 3-10. Recognition rate for different numbers of PDF kernels and large numbers of states.

On the contrary, the maximum of the performance graph takes place at a small number of kernels for models with large number of states, as shown in Figure 3-9 and Figure 3-10, since there is no need for the increased resolution provided by the large number of state PDF kernels. In this case, the more the kernels that are provided the more the model overfits the training data.

Although a 100% recognition rate can be achieved with a 21-state model, smaller numbers of states would be preferable from the complexity point of view, with some performance degradation to be expected. For example, a 12-state model that is almost half-size of the 21-state model with 4-kernel PDF delivers a recognition rate of 98.5% trading 1.5% accuracy with significant complexity saving. This trade is made possible due to the nonlinear relationship between the number of states and the model performance.

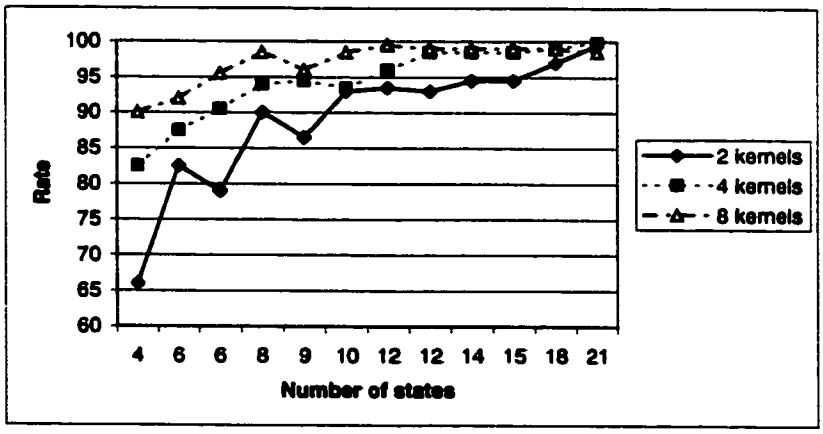


Figure 3-11. Recognition rate for different numbers of states and few PDF kernels.

The results in Table 3-1 are re-presented in Figure 3-11 and Figure 3-12 to show how the performance changes with the number of states for various number of kernels/state. For smaller numbers of kernels in the PDFs, the performance tends

to saturate as the number of states is excessively increased as depicted in Figure 3-11. Again, at large number of the PDF kernels, the model manifests overfitting problem with the increase of the number of states as shown in Figure 3-12.

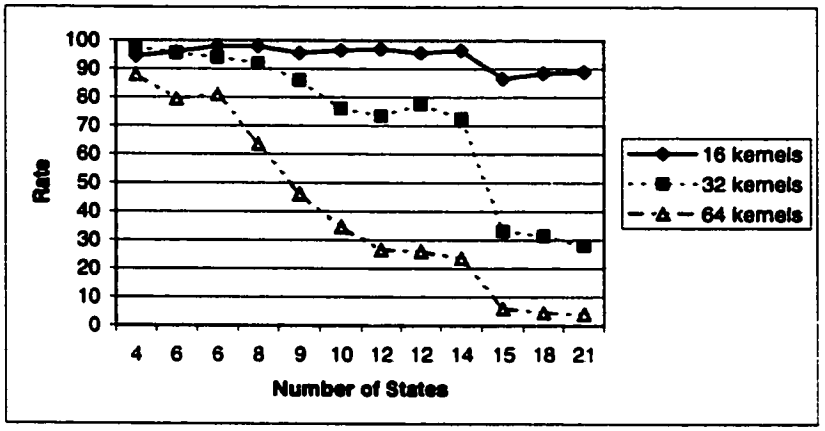


Figure 3-12. Recognition rate for different number of states and large number of PDF kernels.

3.6 Complexity-Performance Comparison

In this section, we present a complexity-performance comparison among various HMM-based FR systems found in the literature. The comparison is based on the number of additions needed at the observation layer and the hidden layer.

The complexity comparison in Table 3-2 assumes all the computations in the observation layer and the hidden layer are performed in the log-domain and that Viterbi-based algorithm has been used in the hidden layer. It should be noted that:

(1) The expression of the complexity of the hidden layer of the first two 1-D HMMs in Table 3-2 is different than that of the 1-D HMM in the third row of the table due to the difference in their topology. The first two models are top-to-bottom HMM with average number of state predecessors of 2. The third model is a fully-connected HMM with average number of state predecessors of N_0.

(2) The complexity of the 2-D PHMM in the fourth row is equivalent to that of one top-to-bottom 1-D HMM that has total number of states equal to the sum of the inner states, $\sum_{k=1}^{N_0} N_1^{(k)}$, and works on a 1-D observation sequence that is equal in length to T_1T_0.

(3) The complexity of the Embedded HMM is equivalent to T_0 times the complexity of N_0 count of left-to-right 1-D HMMs in addition to that of one top-to-bottom 1-D HMM.

(4) Both LC 2-D HMM and the proposed 2-D HMM have similar complexity load that is comparable to that of the 1-D HMM.

Method	Rate	N_0	N_1	AOL	AHL
1-D HMM Strips_Lum [9]	85.0%	5	1	$(T_1T_0)(N_1N_0)(2V)$	$3N_0(T_0)$
1-D HMM Strips_DCT [28]	85.0%	5	1	$(T_1T_0)(N_1N_0)G_s(2V)$	$3N_0(T_0)$
1-D HMM Blocks DCT [33]	100.0%	5	1	$(T_1T_0)(N_1N_0)G_s(2V))$	$N_0(N_0+1)(T_1T_0)$
2-D PHMM_Lum [10]	95.0%	5	3,6,6,6,3	$(T_1T_0)(N_1N_0)(2V)$	$3T_1T_0\sum_{k=1}^{N_0} N_1^{(k)}$
Embedded HMM [14]	98.0%	5	3,6,6,6,3	$(T_1T_0)(N_1N_0)G_s(2V)$	$\sum_{k=1}^{N_0}(N_1^{(k)})^2T_1T_0 + N_0^2T_0$
LC 2-DHMM [34]	72.5%	6	2	$(T_1T_0)(N_1N_0)G_s(2V)$	$N_1N_0(N_{tpr}+1)(T_1T_0)$
Proposed 2-D HMM	98.5%	6	2	$(T_1T_0)(N_1N_0)G_s(2V)$	$N_1N_0(N_{tpr}+1)(T_1T_0)$

N_0 : the vertical number of states of the 1-D HMM, the LC 2-D HMM and the proposed HMM, and the number of vertical superstates for the 2-D PHMM and the Embedded HMM.
N_1 : the horizontal number of states of the LC 2-D HMM and the proposed HMM, and the number of states in the consecutive superstates of the 2-D PHMM and the Embedded HMM.
N_{tpr} : the average number of predecessor states in the LC 2-D HMM and the proposed HMM.
T0,T1 : the vertical and horizontal numbers of observation blocks/strips, respectively.
Gs : the number of the state PDF kernels.
AOL : the number of additions in the observation layer.
AHL : the number of additions in the hidden layer.

Table 3-2. Abstract complexity-performance comparison.

The abstract complexity evaluation provides a rough indication about the relative complexities. However, the practical complexities depend on the typical values which would vary for the different systems targeting their optimal performance point. For example, 5 states and 90% overlap is preferred for the 1-D top-to bottom HMM while 12 states 0% overlap is preferred for the proposed 2-D HMM.

Table 3-3 shows typical values of model structural parameters, i.e. number of states and the resultant recognition rate with the associated complexity in terms of the number of addition needed in the observation layer and the hidden layer. The results in Table 3-3 are also presented in graphically in Figure 3-13 and Figure 3-14.

Method	Rate	N0	N1	Nt	HNp	VNp	HO	VO	T0	T1	V	Gs	AOL	AHL	AOHL	MOL
1-D HMM Strips_Lum	85.0%	5	1	5	102	10	n.a.	90%	102	1	920	1	938400	1515	939915	938400
1-D HMM Strips_DCT	85.0%	5	1	5	102	10	n.a.	90%	102	1	39	1	39780	1515	41295	39780
1-D HMM Blocks DCT	100.0%	5	1	5	16	16	75%	75%	25	21	10	1	52500	15720	68220	52500
2-D PHMM_Lum	95.0%	5	3,6,6,6,3	24	10	8	80%	75%	43	52	80	1	8586240	160920	8747160	8586240
Embedded HMM	98.0%	5	3,6,6,6,3	24	10	8	80%	75%	43	52	6	1	643968	158526	802494	643968
LC 2-DHMM	72.5%	6	2	12	8	8	0%	0%	14	12	9	4	145152	17472	162624	145152
Proposed 2-D HMM	98.5%	6	2	12	8	8	0%	0%	14	12	9	4	145152	17472	162624	145152

Table 3-3. Typical complexity-performance comparison.
N0 is the vertical number of states of the 1-D HMM, the LC 2-D HMM and the proposed HMM, and the number of vertical superstates for the 2-D PHMM and the Embedded HMM.
N1 is the horizontal number of states of the LC 2-D HMM and the proposed HMM, and the number of states in the consecutive superstates of the 2-D PHMM and the Embedded HMM.
Nt is the total number of states in the model, excluding the number of the superstates of the 2-D PHMM and the Embedded HMM.
HNp is the horizontal dimension of the observation block/strip in pixels.
VNp is the vertical dimension of the observation block/strip in pixels.
VO is the percentage overlapping between vertically consecutive observation blocks/strips.
HO is the percentage overlapping between horizontally consecutive observation blocks/strips.
T0 is the vertical number of observation blocks/strips.
T1 is the horizontal number of observation blocks.
Gs is the number of the state PDF kernels.
AOL is the number of additions in the observation layer.
AHL is the number of additions in the hidden layer.
AOHL is the total number of additions in the observation layer and the hidden layer.
MOL is the number of multiplications in the observation layer.
n.a. Not Applicable.

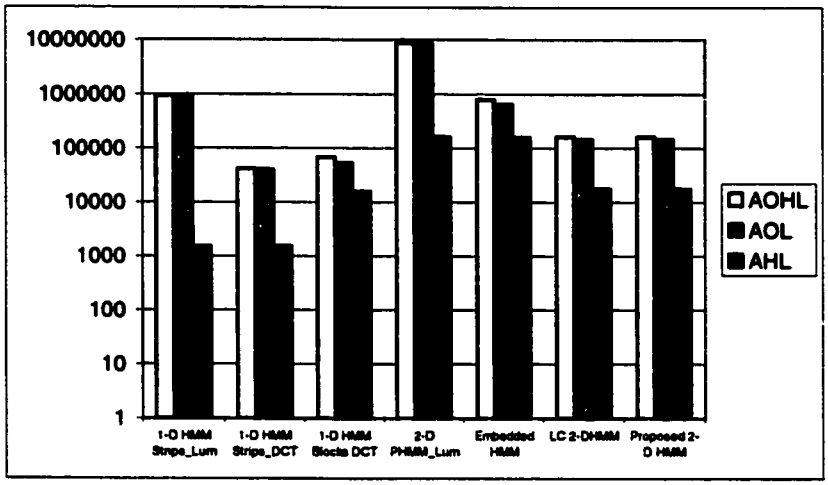


Figure 3-13. Complexity comparison: AOL is the number of Additions in the Observation Layer, AHL is the number of Additions in the Hidden Layer and AOHL is their sum.

The luminance-based 2-D PHMM is an early implementation of the HMM to the FR problem whose complexity is, in principle, less than that of the MMRF-HMM with a good recognition rate of 95%. However, the large number of observation blocks and the considerably large number of states increase the complexity compared to the luminance-strip-based 1-D HMM that is introduced by the same author. The latter was built with a small number of states, e.g. 5 states, and the number of observation strips was reasonable, though highly overlapped. This small number of states, in addition to the simplicity of the top-to-bottom 1-D HMM model structure, helped reduce the complexity compared with the luminance-based 2-D PHMM. Switching from the 2-D PHMM structure to the top-to-bottom 1-D HMM structure came with the cost of degrading the recognition rate to 85%. On the other hand, the complexity of the observation layer of Embedded HMM FR is significantly less than that of the luminance-based 2-D PHMM due to the reduction of the feature vector size based on the transform properties discussed earlier in this chapter. The complexity of the hidden layer of the Embedded HMM is slightly less that that of the

luminance-based 2-D PHMM which has similar structure, though the vertical component of $N_0^2 T_0$ is added, as $\sum_{k=1}^{N_0} (N_1^{(k)})^2$ is always less than or equal to $N_t^2 = \left(\sum_{k=1}^{N_0} N_1^{(k)} \right)^2$.

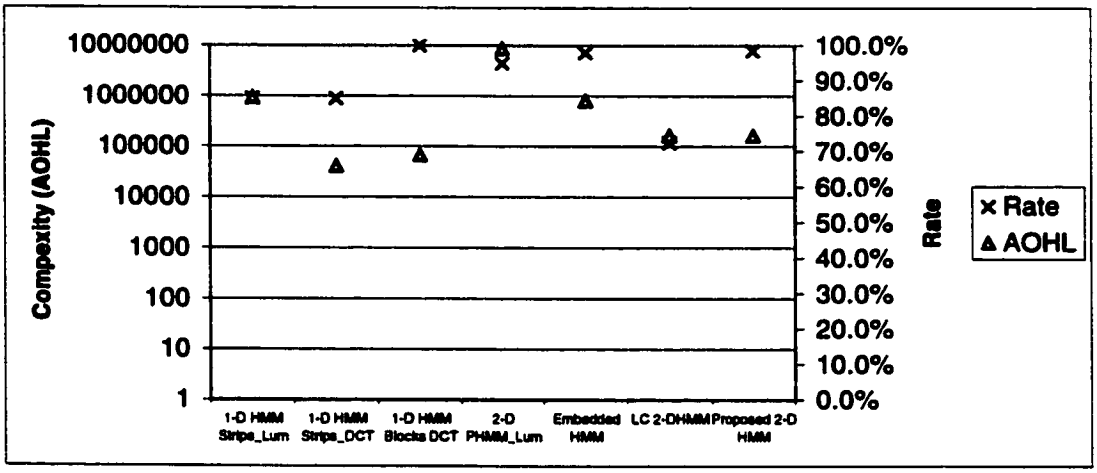


Figure 3-14. Complexity-Performance comparison: AOHL is the number of additions in the observation and hidden layer.

The DCT-strip-based 1-D HMM FR system also exploits the DCT properties in reducing the observation layer complexity while the hidden layer benefits from the top-to-bottom 1-D HMM structure. As seen before, replacing Embedded HMM with the DCT-strip-based 1-D HMM costs a performance decrease from 98% to 85%. The DCT-block-based 1-D HMM FR system costs more than the DCT-strip-based 1-D HMM in the observation layer due to having more observation blocks. It also costs more in the hidden layer due to using an ergodic model structure as opposed to the simple top-to-bottom structure in return for a reported 100% recognition rate. The complexity of the proposed 2-D HMM is similar to that of the LC 2-D HMM which is lower than that of the Embedded HMM and 2-D PHMM, but higher than that of 1-D HMM. The proposed 2-D

HMM achieves recognition rate of 98.5% which is significantly high compared with the other forms of HMM-based face recognition systems.

Considering the actual execution time, the proposed 2-D HMM FR system consumes 1.025 second on average to train one 6x2x4 model and 0.5 second on average to test one image with such a model using a P350MHz PC. That is comparable to the 1-D HMM FR system in [33] that consumes 23 seconds on average to train one subject model and 2.1 seconds on average to test one image with such a model using a P200MHz PC. It should be noted that the system in [33] has a feature extraction layer that involves carrying out DCT transformation as opposed to the proposed system that perform the search in the transform domain. This constitutes an additional computational load, specially with the large number of observation blocks that results from the large block overlap (75% in [33]). This computational load is not shown in the complexity comparison in this thesis since it is focused on the complexity of the hidden layer and the complexity of the observation layer. This explains the system in [33] consumes more processing time compared with proposed system.

The typical time figures above are presented just to indicate a rough trend, keeping in mind that there are practical considerations other than model complexity may affect the time of execution.

3.6.1 Complexity of the Proposed 2-D HMM vs. the 2-D MMRF HMM

The proposed 2-D HMM has a hidden layer complexity in the order of $(2N_t^2)$ in general, which is considerably lower than the complexity of the MMRF 2-D HMM introduced in [18], (N_t^4), assuming nontrivial model size. It is comparable to the hidden layer complexity of the 1-D HMM, (N_t^2).

Performance-wise, the 6-state 16-grey level 2-D MMRF HMM introduced in [18] has achieved a reported accuracy of 90.8% on recognizing 10 objects, namely handwritten digits. Considering the proposed 2-D HMM with 6-state in 3x2 constellation and 16 Gaussian kernels PDF, a recognition rate of 98% has been achieved over the facial database that contains 40 persons. We only consider the comparison on the abstract level, i.e. considering the two HMM recognizers performing a MAP search among different objects. Obviously, the physical nature of the searched objects is different in the two problems which imposes some constraints on the comparison. In other words, there is no guarantee we will have the same performance if we reverse the databases. When comparing the two systems the following factors should be taken into consideration:

- The extracted features: N-grey level luminance, M-bands DCT coefficients, ... etc.
- The extracting fashion: pixels, strips, blocks, ... etc.
- The non-stationary behavior: digits are highly non-stationary, thus, block-based feature extraction is not suitable for digit or character recognition in general, while images show lower degree of non-stationarity.
- The overall complexity: the complexity figures in the literature are per feature unit. That is to say, in [18], the complexity is in the order of (N_t^4) per pixel, where the number of pixels in each segmented digit is 5376, 64x224. On the other hand, the complexity of the proposed 2-D HMM is in the order of $(2N_t^2)$ per block, where the number of blocks is just 168, 14x12.

Figure 3-15 depicts the comparison of the three different 6-state HMM recognizers; 1-D HMM, 2-D MMRF HMM [18], and the proposed 2-D HMM based on the hidden layer complexity.

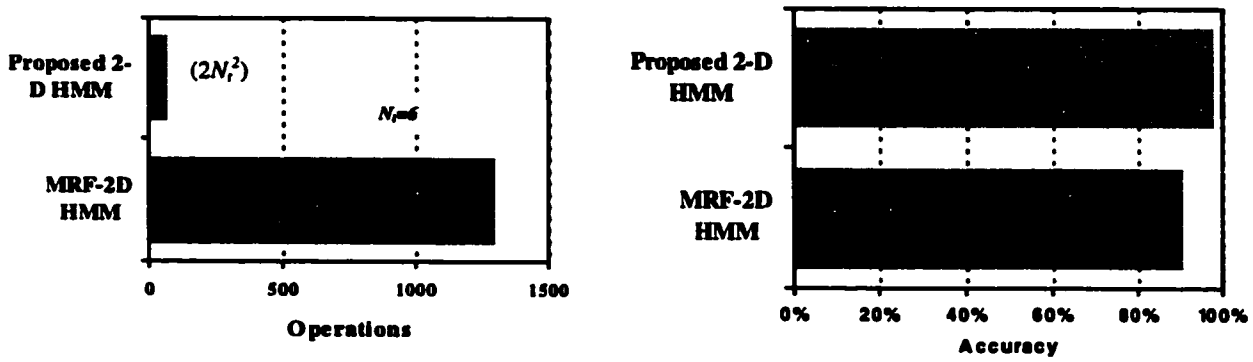


Figure 3-15. Comparison of MMRF 2-D HMM [18], and the proposed 2-D HMM Recognizers. (a) The hidden layer complexity comparison, (b) Performance comparison.

3.7 Conclusion

In this work, a low-complexity yet efficient 2-D HMM is proposed and applied to face recognition. The 2-D HMM as an extension to the 1-D HMM is known for its prohibitively high computational complexity. Over the last decade, several alternatives have been introduced to model 2-D signals on Markovian bases with affordable complexity and acceptable efficiency. The 2-D HMM alternatives include fitting the 2-D signal in a 1-D observation sequence that is modeled with a 1-D HMM, e.g. 1-D sequence of strips or blocks. Alternatively, the extracted features can be a set of parameters, e.g. parametric representation in gesture recognition and stroke parameters in hand-written OCR. Other work has been done using Pseudo 2-D HMM that consists of horizontal 1-D Markov chains placed in a vertical 1-D chain of superstates to form the Pseudo 2-D Model.

A new 2-D HMM is proposed here. The proposed model builds on a true 2-D HMM structure that exploits the conditional independence that arises among the neighboring feature blocks. Furthermore, the no-overlap strategy, combined with appropriate DCT feature block size, makes the model compatible with JPEG-compressed databases with minimal feature extraction. The different factors that contribute to reducing the overall complexity of the proposed 2-D HMM system are depicted in Figure 3-16.

Design assumptions are provided, the likelihood evaluation is derived and the Viterbi-based applied learning method used in the proposed model is described. A detailed discussion on the computational complexity of the model is also provided in this work along with performance-complexity comparison with similar HMM-based Face Recognition systems. In brief, the computational complexity of the proposed 2-D HMM

is lower than that of the Markov Mesh Random Field 2-D HMM and other 2-D Pseudo HMMs while being comparable to those of the 1-D HMM systems. The proposed system delivers recognition rate up to 100%, depending on the model size, tested on the facial database of *AT&T Laboratories Cambridge*.

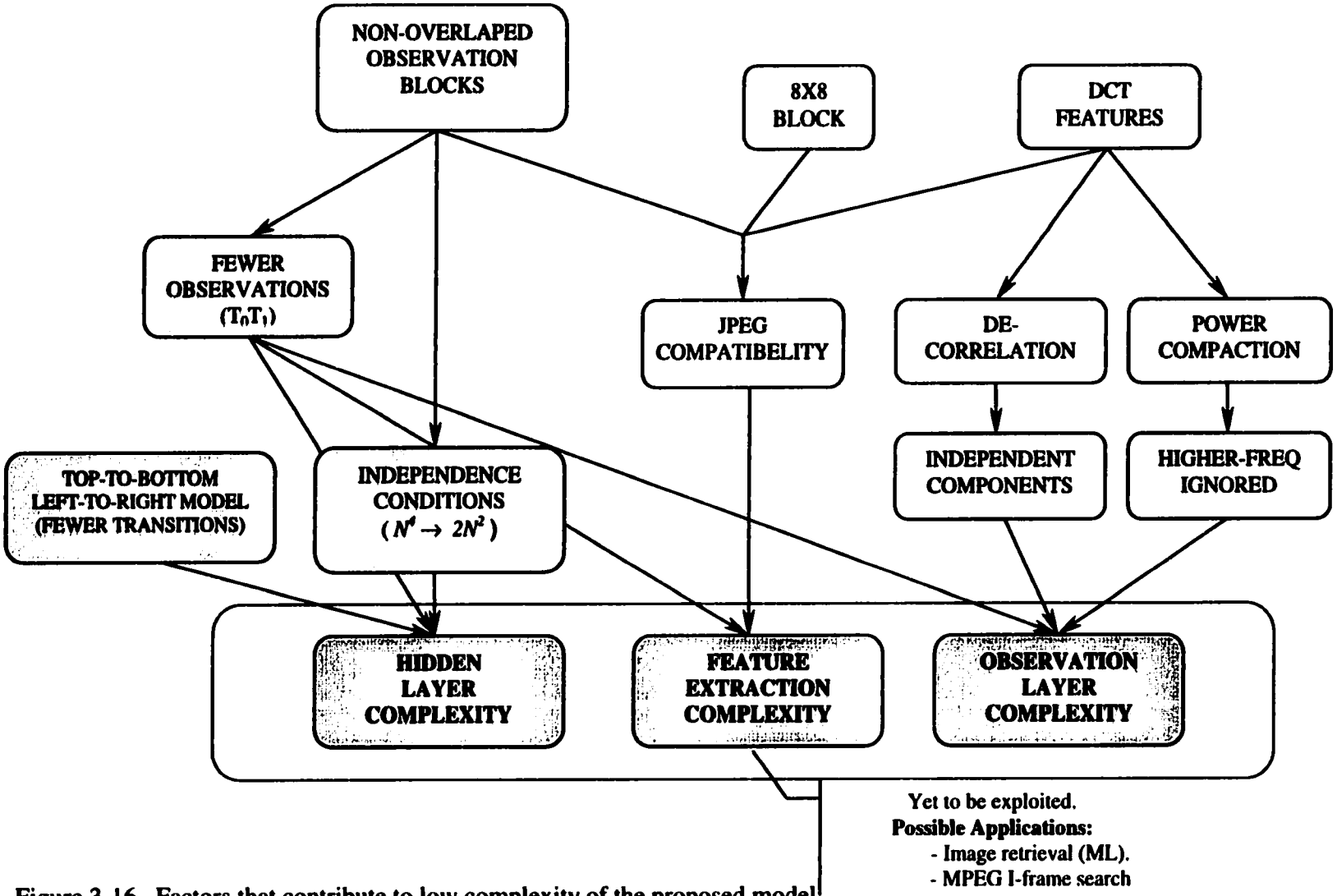


Figure 3-16. Factors that contribute to low complexity of the proposed model.

Chapter 4 A Study of Tied-Mixture Hidden Markov Model

In this chapter, we investigate the application of parameter tying to the low complexity 2-D second-order Hidden Markov Model (HMM) proposed in Chapter 3. The tied-mixture HMM system is then applied to the face recognition problem. Tying HMM parameters is a well-known solution for the problem of insufficient training data leading to non-robust estimation. It will be shown that parameter tying in HMM also enhances the resolution in the case of small models. The performance of the proposed 2-D HMM tied-mixture face recognition system is studied considering both full tying scheme and partial tying.

4.1 Introduction (Parameter Tying in HMM)

Parameter tying in HMM is a known solution for better estimation in insufficient training data conditions. The term "tying" refers to forcing two or more of the model components to share their parameters. Parameter tying was not only exploited in Speech Recognition, but also in Optical Character Recognition (OCR) [38]-[42].

Consider an HMM recognition system consisting of a number of models with the following structure:

- Each model consists of a number of states.
- Each state has a set of state transition probabilities and a PDF mixture.
- Each state PDF mixture is a weighted-sum of a number of kernel (prototype) functions, e.g. Gaussian Distributions. The set of weights in this weighted-sum is referred-to as Mixture Weight Distribution (MWD).

- Each kernel function is completely defined by a set of parameters as provided by the covariance matrix, e.g. a Gaussian Distribution is defined by its mean and its variance.

Tying is applied to HMM in different levels and at different scales. **Levels of tying** include: (starting backward from the last level)

i) Gaussian kernel tying, e.g. the covariance matrix is shared by a number of multivariate Gaussian distributions.

ii) Mixture tying, e.g. a set (codebook) of Gaussian distributions is shared by a number of states.

iii) Sub-state tying [38], where the state PDF is partially shared. In order to do this, the state PDF is expressed as a mixture of mixtures.

iv) State tying [39], where the PDF and transition probabilities of a state are shared by a number of models.

v) Model tying [40], where the whole model is shared within a semantic framework.

On the other hand, **tying scales** range from zero-tying, partial-tying to full-tying. In zero-tying, each and every parameter in the model retains its own independent value. On the contrary, in the full-tying, a model parameter is tied across all the models in the system, e.g. all the state PDF mixtures share one pool of kernels in the fully-tied-mixture HMM as depicted in Figure 4-1. In partial-tying, a model parameter is tied within group of models or model components, e.g. states, PDF mixture and kernel function. For example, states are grouped in clusters, where the states in the same cluster have their mixtures tied together. The model components, i.e. the states in the previous example, are

clustered based on either subjective criteria, e.g. phonetic, or information-theoretic criteria, e.g. Entropy-based criterion [41].

Two of the HMM popular forms, namely Continuous-density HMM (CHMM) and Semi-Continuous-density HMM (SCHMM) [42], are, in fact, a Zero-Tied-Mixture HMM (ZTM-HMM) and a Fully-Tied-Mixture HMM (FTM-HMM), respectively, from parameter tying perspective.

In the following sections, we briefly revisit the Continuous Probability and the Semi-Continuous Probability HMM and see how they can be described by one general tying framework. We then show examples of arbitrary and optimized tied-mixture HMM.

4.1.1 Continuous *vs.* Semi-Continuous HMM as They Relate to Tying

The Continuous-Density HMM (CHMM) provides high stochastic resolution in the observation space. However, it needs a huge amount of training data for robust estimation of the model parameters, which is not practically feasible in some cases. On the other hand, one can argue that the CHMM contains component redundancy since the conditional probability density functions (PDFs) associated with different states are assumed to be completely different and are expressed in terms of a weighted sum of exclusive sets of Gaussian components, as shown in Figure 4-1. This not only results in a large computational load but also more memory space. Performance-wise, the CHMM delivers high performance *provided that the training data set is sufficiently large.* Conversely, the performance degrades significantly in small training data environments where the data set is not sufficient to train the large number of free parameters.

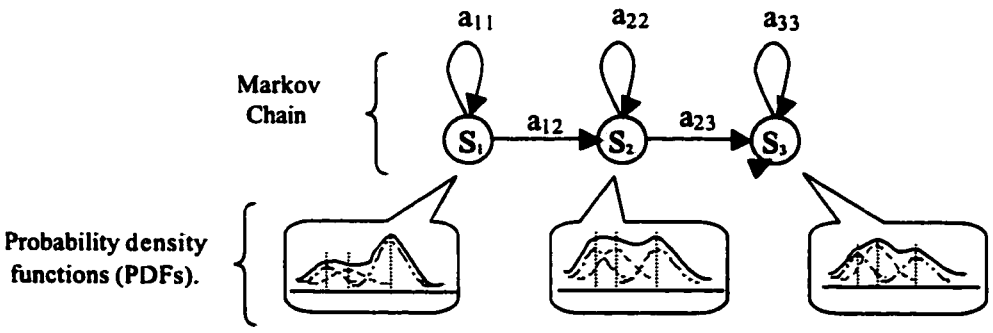


Figure 4-1. The Continuous Hidden Markov Model (CHMM).

On the other hand, all the conditional PDFs in the framework of the Semi-Continuous HMM (SCHMM) are expressed in terms of a weighted sum of one set of Gaussian components, referred to as *Codebook*, as shown in Figure 4-2. Each state maintains a mixture-weight distribution by which its own conditional PDF is constructed in terms of the codebook elements. Compared with the CHMM, the SCHMM tends to average the probabilistic features of the observation space, yet offers a compact, robust but low-resolution modeling solution for the same number of kernels per state PDF. The SCHMM is expected to work better in the case of smaller training data sets because of the lower number of parameters to be trained.

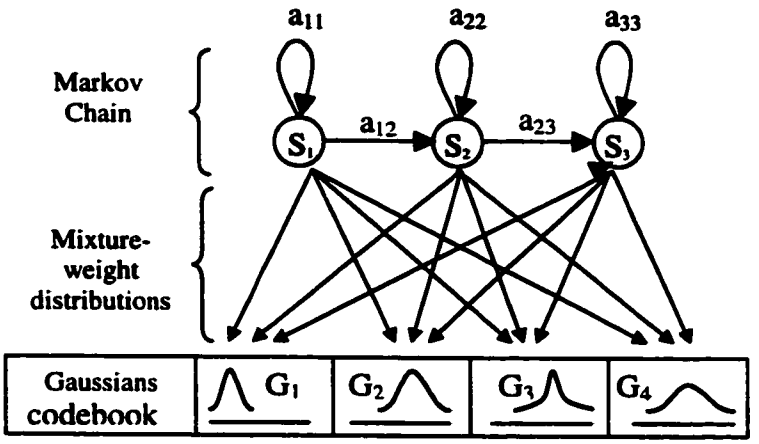


Figure 4-2. Semi-Continuous Hidden Markov Model (SCHMM)

A generalized view of both CHMM and SCHMM in the context of HMM PDF mixture tying is presented in [46]. A state is said to be tied to another state if their PDFs share the same Gaussian codebook. Generally, more than one Gaussian codebook may be available, however, in the special case of one Gaussian codebook, the whole setup of the tied-mixture HMM is equivalent to the SCHMM, i.e. a fully-tied-mixture HMM. On the other extreme, the CHMM can be viewed as a zero-tied-mixture HMM where each state has its own codebook and the number of codebooks is equal to the number of states. An arbitrary tied-mixture HMM, referred to as *genonic* HMM, can be designed to compromise the probabilistic resolution of the CHMM and the robustness of the SCHMM [46] for a given problem. The term "genonic" resolves the confusion that may arise due to the regular use of the term tied-mixture HMM to refer to the SCHMM in the literature, while the word "genone" refers to the Gaussian codebook that is shared by a set of states. Genonic HMM is, in fact, a partially-tied-mixture HMM.

4.1.2 Generalized Tied-Mixture HMM

Let **O** be the observation vector, S be a hidden state and $\mathbf{N}_\chi=\{N_\chi^{(g)}\}$ be the Gaussian codebook, i.e. the genone, of the state S. The conditional PDF of the observation vector **O** becomes:

$$p(\mathbf{O} \mid S) = \sum_{N_\chi^{(g)} \in \mathbf{N}_\chi} P(N_\chi^{(g)} \mid S) N_\chi^{(g)}(\mathbf{O}; \boldsymbol{\mu}_\chi^{(g)}, \boldsymbol{\Sigma}_\chi^{(g)}) \tag{4.1}$$

where $N_\chi^{(g)}$ is the g^{th} Gaussian component in the χ^{th} genone in the system, $\mathbf{N}_\chi$,that is associated with the state S and is given by the state-to-genone mapping γ:

$$\mathbf{N}_\chi = \gamma(S) \tag{4.2}$$

Accordingly, three main cases arise:

1- Each state uses its own exclusive Gaussian codebook, as in CHMM:

$$\mathbf{N}_{\chi_i} \cap \mathbf{N}_{\chi_j} \underset{\forall \chi_i \& \chi_j}{=} \phi \tag{4.3}$$

This case corresponds to a zero-tied-mixture HMM, i.e. CHMM.

2- All states use the same Gaussian codebook, Q, as in SCHMM:

$$\mathbf{N}_{\chi_i} \underset{\forall \chi_i}{=} \mathbf{N} \tag{4.4}$$

where **N** is the universal Gaussian kernels codebook in this case.

This case corresponds to a fully-tied-mixture HMM, i.e. SCHMM.

3- Different sets of tied-mixture states use different Gaussian codebooks, one codebook per set of states, as in genonic HMM:

$$\mathbf{N}_{\chi_i} \underset{\forall S_k \in \gamma^{-1}(\mathbf{N}_{\chi_i})}{=} \mathbf{N}_{\chi} \tag{4.5}$$

where $\gamma(S)$ is state-to-genone mapping function.

This case corresponds to a partially-tied-mixture HMM.

The degree of tying in the genonic HMM depends on the amount of available training data. The larger the training data set, the less the need for tying. At the extreme of a large training data set, the CHMM becomes the solution. The degree of tying also depends on the available computing and memory resources, i.e. a system with low degree of tying may be replaced with another that has higher degree of tying in order to obtain the best performance at certain level of complexity constraints. Last but not least, the nature of the problem usually affects the type of tying as will be briefly discussed in the following section.

4.1.3 Examples of Mixture Tying

Generally, the state parameters are tied based on a prior knowledge of correlation. However, the tying scheme can be optimized through an automated procedure to achieve the best configuration and level of tying, given the training data distribution. First we will briefly show a simple example of tying where no optimization is made then we will highlight an optimized tying technique.

Mixture tying in [47] is an example of fixed mixture tying, where the tying configuration is determined prior to the model training process and remains the same through out the training process. In that work, Mixture tying is applied to Hidden Markov Tree (HMT). HMT is a probabilistic model that is similar to the HMM except that the conditional state probability is defined across the scales of DWT [23][24] in a form of tree, as depicted in Figure 4-3, as opposed to a chain in the traditional HMM. The Markov Tree in HMT exploits the statistical dependency among the Wavelet coefficients in different wavelet scales at certain spatial location.

The local correlation among the coefficients in the same scale suggests tying their hidden states across that scale within the decomposition tree. Similarly, tying across trees is derived by the correlation of the coefficients in the same location of different trees. The estimation of the tied parameters in [47] extends the statistical averaging in the upward-downward learning process to cover the set of tied parameters including the initial and transition probabilities in addition to the PDF of the state. Note that in this work the *tied PDF* is typically used by the tied states, as opposed to the general mixture tying where each state has its *own PDF* that is obtained from the shared mixture components through its *own mixture-weight distribution*. Figure 4-3 shows tying across decomposition tree and within the decomposition tree for certain scale.

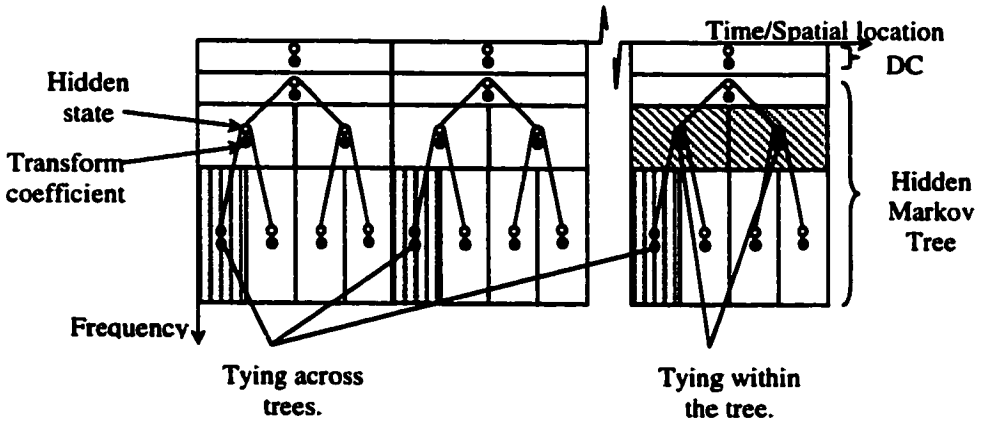


Figure 4-3. Tying across and within the decomposition tree [47].

The work in [47] shows that tied-mixture HMT has better performance denoising signals that are corrupted with zero-mean Gaussian noise compared with an independent-mixture model.

On the other hand, the Automatic Speech Recognition (ASR) system in [46] is an example of a Partially-tied-mixture HMM that is optimized for Speech Recognition, *Genonic* HMM. The process of estimating the optimum tying starts with an arbitrary tied-

mixture HMM and runs independently on the different groups of tied-mixture models. For example, the models of allophones of the same set are initially fully-tied, then this tying is optimized for each group of allophones.

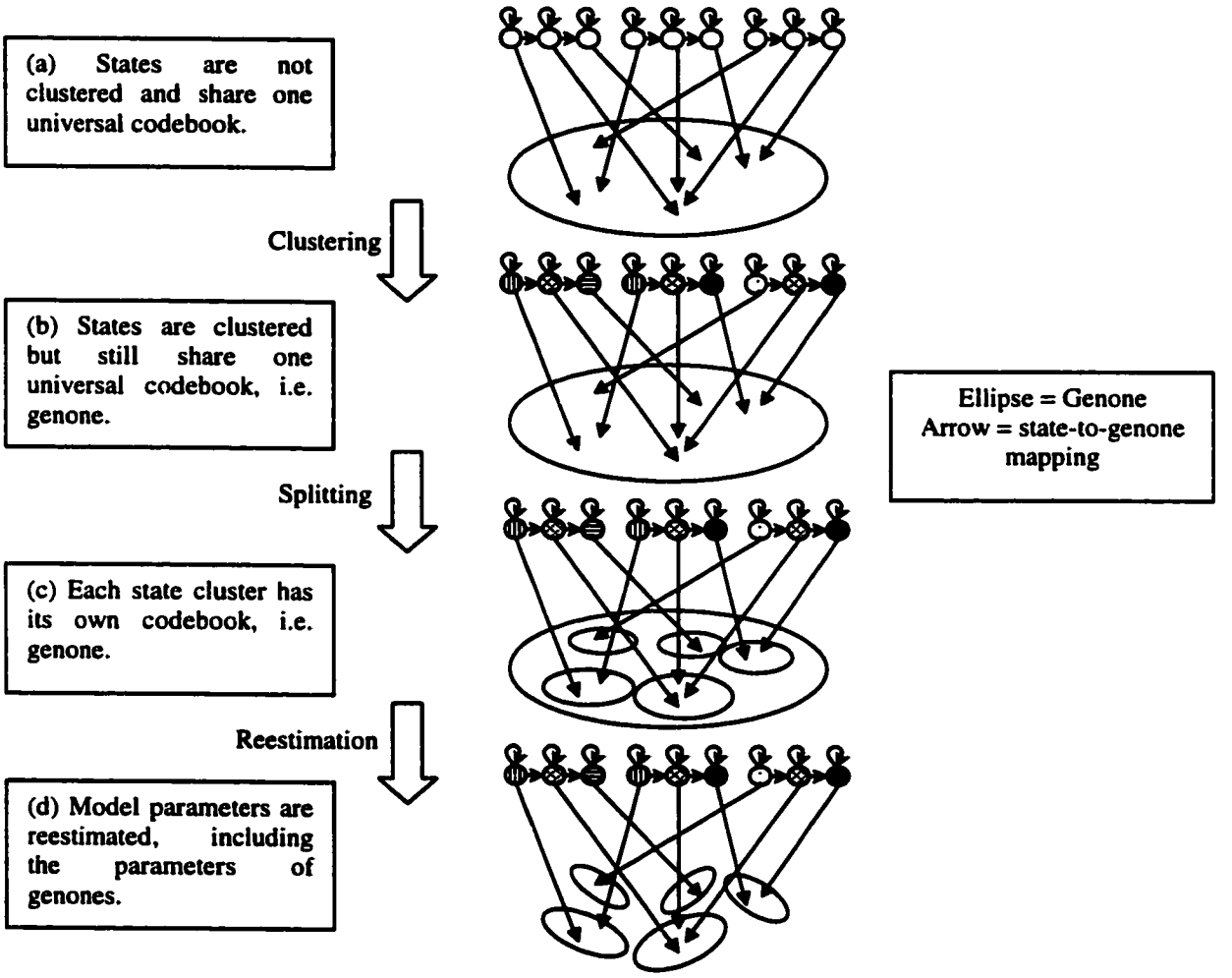


Figure 4-4. Construction of genonic Mixtures [46].

The process consists of three main parts; clustering, splitting and re-estimation as depicted in Figure 4-4 of one group of tied-mixture models:

i) Clustering: The states within the group are clustered into *n*, an empirically determined number, of exclusive clusters based on the similarity of their mixture-weight

distributions. The dissimilarity measure in that work is *the increase of entropy of the state mixture-weight distribution* due to the state merge, which is given by:

$$d(S_1,S_2)=(n_1+n_2)H(S)-n_1H(S_1)-n_2H(S_2) \tag{4.6}$$

where $d(S_1,S_2)$ is the weighted-by-count increase of entropy of the state mixture-weight distribution due merging state S_1 and state S_2 given that they are already tied. n_1 and n_2 are the number of observations that were used for estimating the mixture-weight distribution of state S_1 and state S_2, respectively, and $H(S)$, $H(S_1)$ and $H(S_2)$ are the entropy functions of the mixture-weight distributions of state S, S_1 and S_2, respectively, where $H(S)$ is defined as:

$$H(S)=-\sum_{N_\chi^{(g)}\in \mathbf{N}_\chi} P(N_\chi^{(g)}\mid S)\log(P(N_\chi^{(g)}\mid S)) \tag{4.7}$$

where $\{P(N_\chi^{(g)}|S)\}$ is the mixture-weight distribution of the state S.

The state S that results from merging the two states has a PDF that is expressed in terms of the same codebook of its mergers, i.e. $\mathbf{N}_\chi = \mathbf{N}_{\chi_1} = \mathbf{N}_{\chi_2}$ and is obtained using a new mixture-weight distribution of :

$$P(N_\chi^{(g)}\mid S)=\frac{n_1}{n_1+n_2}P(N_{\chi_1}^{(g)}\mid S_1)+\frac{n_2}{n_1+n_2}P(N_{\chi_2}^{(g)}\mid S_2) \tag{4.8}$$

ii) Splitting: In this step, n initial genones are built, one per each cluster of states. A copy of the original codebook can be used as initial genones for all the clusters in the group. Then the size of the genones is to be reduced in the next step. Alternatively, the most likely subset of components can be chosen from the original codebook to form the initial genone of the cluster. Normally, the size of the cluster genone is smaller than that of the original group since it supports fewer states.

iii) Reestimation: Given the training data and the initial values of the HMM parameters, the new Gaussian components of genones are reestimated, along with the rest of the HMM parameters. The standard Baum-Welch algorithm can be used for this purpose [46].

The results in [46] show that the genonic HMM delivers lower word recognition error rate when compared with the tied-mixture HMM and Phonetically-tied-mixture HMM for Automatic Speech Recognition task on the Wall Street Journal (WSJ) 0/1 corpus. The performance of the recognizer is also reported against different degrees of tying, which correspond to a different number of genones in the system. It can be seen that the error rate decreases with the increase of the number of genones in the system, i.e. with the decrease of the tying degree, due to the increase of the statistical resolution. However, at a certain point, the relation is reversed and the model starts to lose its robustness due to the increase in the number of free system parameters for the same training data size.

4.2 Fully-Tied-Mixture 2-D HMM Applied to Face Recognition

In this section, we will extend the concept of mixture tying to Two-Dimensional HMM Face Recognition using the model we proposed earlier in Chapter 3. We will first consider the Fully-Tied-Mixture 2-D HMM Face Recognition (FTM 2-D HMM FR), then we will introduce the Partially-Tied-Mixture in the following Section.

In FTM 2-D HMM FR, all the mixtures in the system are tied. As described in the previous Section, the PDF of the state is expressed as a weighted-sum of the kernel functions of a universal codebook. Each state retains its own set of weights that

characterize its own PDF. This set of weights constitutes the state MWD. In this work, we use a universal codebook of Gaussian distributions.

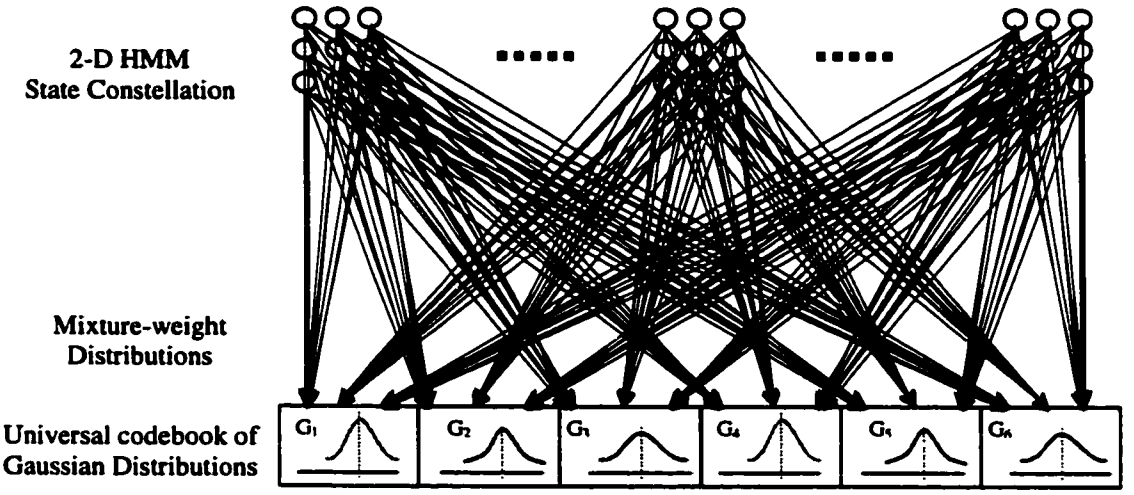


Figure 4-5. Semi-Continuous 2-D Hidden Markov Model (2-D SCHMM), i.e. Fully-Tied-Mixture 2-D HMM.
(All states access the same codebook but use different weights.)

4.2.1 Proposed Procedure of Parameters Estimation of the TM 2-D HMM

The Gaussian kernels in the universal codebook are initially obtained using the LBG algorithm [48]. The feature vectors (that represent the 8x8 pixels observation blocks) are collected from the training images, then, clustered into a number of clusters equal to the targeted number of Gaussian kernels in the universal codebook using the iterative LBG algorithm. The centroids of the clusters and the variance of the member vectors within the clusters are used to initialize the means and the variances of the Gaussian kernels, respectively.

A short training iteration is run to estimate the initial transition probabilities where a uniform segmentation is used to trigger this run. No state sequence decoding is done in this particular run. Accordingly, each state builds its own MWD to best represent the given segments (from different given training images). The transition probabilities are also estimated to best reflect the initial segmentation. Once the initial models are obtained, the iterative learning process is carried out until a threshold limit is reached. The estimation process is similar to that in Chapter 3 with a slight modification in the estimation of the means, the variances in the codebook and state MWDs as follows.

Let the Gaussian distributions codebook be:

$$\mathbf{N}_\chi = \left\{ N_\chi^{(g)}(\mathbf{O}, \boldsymbol{\mu}_\chi^{(g)}, \boldsymbol{\Sigma}_\chi^{(g)}) : 1 \le g \le G_\chi \right\} \tag{4.9}$$

where G_χ is the number of Gaussian kernels in the system, $\boldsymbol{\mu}_\chi^{(g)}$ is the mean vector and $\boldsymbol{\Sigma}_\chi^{(g)}$ is the covariance matrix.

The mean vector of the g^{th} Gaussian kernel is estimated as:

$$\overline{\boldsymbol{\mu}}_\chi^{(g)} = \frac{\sum_{\substack{\forall \text{ training} \\ \text{images}}} \left[\sum_{k=1}^{K} \sum_{l=1}^{L} I^{(g)}(\mathbf{O}_{k,l}) \cdot \mathbf{O}_{k,l} \right]}{\sum_{\substack{\forall \text{ training} \\ \text{images}}} \left[\sum_{k=1}^{K} \sum_{l=1}^{L} I^{(g)}(\mathbf{O}_{k,l}) \right]} \tag{4.10}$$

where $\mathbf{O}_{k,l}$ is the feature vector of the observation block in the k^{th} row and l^{th} column, K and L are the vertical and the horizontal number of observation blocks in the image, respectively, and the indicator function $I^{(g)}(\mathbf{O}_{k,l})$ is given by:

$$I^{(g)}(\mathbf{O}_{k,l}) = \begin{cases} 1 & , g = \underset{1 \le g' \le G_\chi}{\arg\max} \left(c_{m,n}^{(g')} N_\chi^{(g')}(\mathbf{O}_{k,l}, \boldsymbol{\mu}_\chi^{(g')}, \boldsymbol{\Sigma}_\chi^{(g')}) \right) \\ 0 & , otherwise \end{cases} \tag{4.11}$$

$$\bar{\Sigma}_{\chi}^{(g)} = \frac{\sum_{\substack{\forall \text{training} \\ \text{images}}} \left[\sum_{k=1}^{K} \sum_{l=1}^{L} I^{(g)}(\mathbf{O}_{k,l}) \cdot (\mathbf{O}_{k,l} - \boldsymbol{\mu}_{\chi}^{(g)})(\mathbf{O}_{k,l} - \boldsymbol{\mu}_{\chi}^{(g)})^{t} \right]}{\sum_{\substack{\forall \text{training} \\ \text{images}}} \left[\sum_{k=1}^{K} \sum_{l=1}^{L} I^{(g)}(\mathbf{O}_{k,l}) \right]} \tag{4.12}$$

Let $\mathbf{C}_{m,n}$, the MWD of the state $S_{m,n}$, be:

$$\mathbf{C}_{m,n} = \left\{ c_{m,n}^{(g)} : 1 \le g \le G_{\chi} \right\} \tag{4.13}$$

where $c_{m,n}^{(g)}$ is the weight coefficient of the g^{th} Gaussian in the state $S_{m,n}$ and is updated by:

$$\bar{c}_{m,n}^{(g)} = \frac{\sum_{\substack{\forall \text{training} \\ \text{images}}} \left[\sum_{k=1}^{K} \sum_{l=1}^{L} I^{(g)}(\mathbf{O}_{k,l}) I_{m,n}(q_{k,l}) \right]}{\sum_{\substack{\forall \text{training} \\ \text{images}}} \left[\sum_{k=1}^{K} \sum_{l=1}^{L} I_{m,n}(q_{k,l}) \right]} \tag{4.14}$$

where

$$I_{m,n}(q_{k,l}) = \begin{cases} 1 & , q_{k,l} = S_{m,n} \\ 0 & , otherwise \end{cases} \tag{4.15}$$

4.2.2 Performance of the Proposed 2-D FTM-HMM in Face Recognition

In order to compare the performance of the proposed 2-D FTM-HMM with the performance of the proposed 2-D HMM with no tying, a test was carried out using the same facial database, notably AT&T, that is used in Chapter 3 with the same test conditions.

States per model		# States per model	#States in the system	# kernels per state	#kernels in the system	Recogn-ition Rate
# rows	#columns					
2	2	4	160	2	320	66.0%
2	2	4	160	4	640	82.5%
2	2	4	160	8	1280	90.0%
3	2	6	240	2	480	79.0%
3	2	6	240	4	960	90.5%
3	2	6	240	8	1920	95.5%
4	2	8	320	2	640	90.0%
4	2	8	320	4	1280	94.0%
4	2	8	320	8	2560	98.5%

States-per-model = # States in rows x # States in columns
States in the system = # States-per-model x # Models in the system
Kernel in the system = # Kernels-per-state x # States in the system

Table 4-1 Performance of the 2-D Continuous-probability HMM (2-D CHMM), i.e. Zero-Tied, Face Recognition system.

States per model		# States per model	#States in the system	#kernels in the system	Recognition Rate
# rows	#columns				
2	2	4	160	256	95.0%
2	2	4	160	512	96.0%
2	2	4	160	1024	94.5%
3	2	6	240	256	94.0%
3	2	6	240	512	95.5%
3	2	6	240	1024	95.0%
4	2	8	320	512	94.5%
4	2	8	320	1024	97.0%
4	2	8	320	2048	98.0%

States-per-model = # States in rows x # States in columns
States in the system = # States-per-model x # Models in the system
Kernel in the system is emperically set

Table 4-2 Performance of the 2-D Semi-Continuous-probability HMM (2-D SCHMM), i.e. Fully-tied, Face Recognition system (The length of the state MWD is equal to the total number of kernels).

Table 4-1 shows the performance of the 2-D CHMM, i.e.2-D ZTM-HMM, while Table 4-2 shows the performance of the 2-D SCHMM, i.e. 2-D FTM-HMM. In Table 4-1, the first row, we have 4 states in the model of one face, i.e. 4x40=160 states for the

complete system (40 faces). If each state has exclusive access to 2 kernels, then we have 160x2=320 kernels in the system. Similarly, in Table 4-2, first row, we have 160 states in the system, which share the access to a pool of 256 kernels.

We use binary splitting codebook to initiate the Gaussian kernels in each state PDF in the 2-D ZTM-HMM, hence, the number of Gaussian kernels per state in Table 4-1 is a power-of-2. This also applies to the number of Gaussian kernels in Table 4-2 for the 2-D FTM-HMM system.

Comparing the performance of the two systems for the same number of states, 2-D FTM-HMM outperforms 2-D ZTM-HMM with equal or larger number of Gaussian kernels in the system. This reflects the need of the system for a shift towards more robust estimation at the expense of lower resolution. This also suggests that the different subjects (persons) share features, which benefit from mixture-tying.

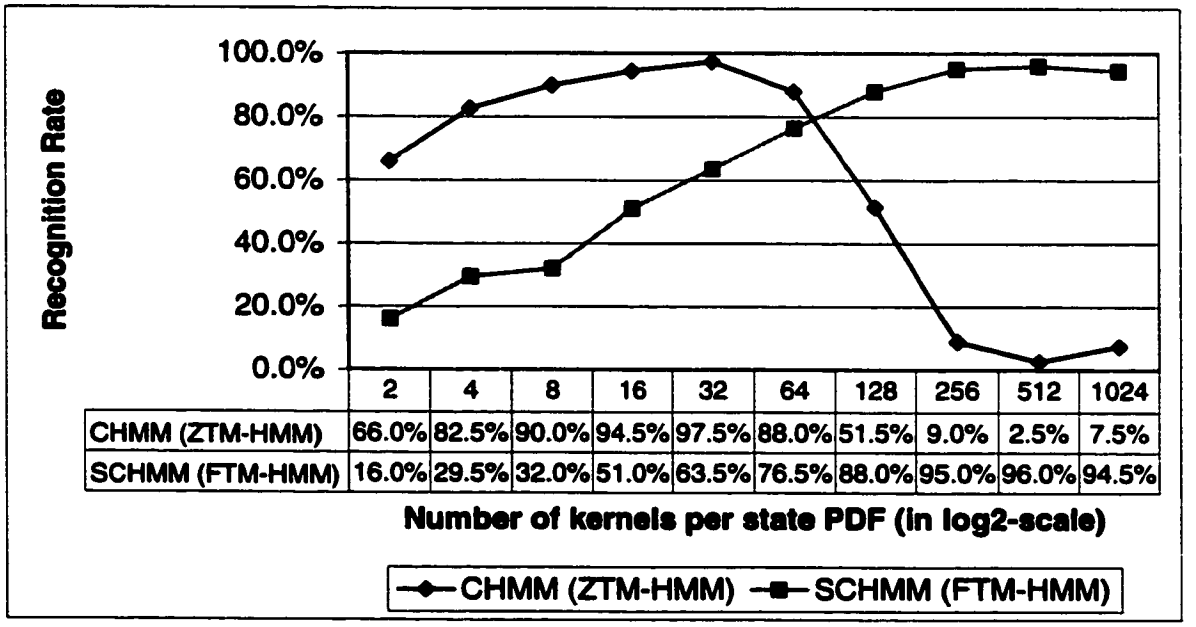
It should be noted that the model in the first row of Table 4-1 is lacking resolution due to its small size. In this case, although there are more than 256 Gaussian kernels in the system, they are not well-exploited. The resultant rate is barely 66%. If only 256 Gaussian kernels in the system are used and shared among the states, the unnecessary repetition of Gaussian kernels that represent common features is eliminated. Sharing also gives state PDFs access to a larger number of Gaussian kernels, which enhances the resolution. In large models, where the resolution of the model is sufficient, 2-D FTM-HMM is marginally better or equal in performance.

One can then argue that in small model sizes, where there is a lack of resolution, sharing of Gaussian kernels not only provides robust estimation of the model parameters due to sharing the training state of all models and states, as discussed earlier, but also

enhances the resolution of the states PDFs through widening their access to more kernels. Thus, SCHMM is recommended for low-complexity small-model systems.

This does not contradict the discussion in [46] which implies that ZTM-HMM provides better resolution while FTM-HMM provides better robustness. This discussion is true for the same number-of-kernels-per-state in all cases: FTM-HMM and ZTM-HMM, i.e. when the state MWD has the same length in the two models.

To further understand the issues affecting the performance of the system under different tying conditions, we compare the performance of the 2-D FTM-HMM FR system with that of the 2-D ZTM-HMM FR system for the same number-of-kernels-per-state on a 2x2 state model in Figure 4-6 and Figure 4-7.



	2	4	8	16	32	64	128	256	512	1024
CHMM (ZTM-HMM)	66.0%	82.5%	90.0%	94.5%	97.5%	88.0%	51.5%	9.0%	2.5%	7.5%
SCHMM (FTM-HMM)	16.0%	29.5%	32.0%	51.0%	63.5%	76.5%	88.0%	95.0%	96.0%	94.5%

Figure 4-6. Recognition rate of 2-D ZTM-HMM, i.e. CHMM, and 2-D FTM-HMM, i.e. SCHMM, versus number of kernels per state, in log2-scale.

The results in Figure 4-6 show the recognition rate of 2-D ZTM-HMM FR system and the 2-D FTM-HMM FR system versus the number of kernels per state PDF. The

number of kernels per state PDF is shown in log-scale in order to emphasize the behavior of the two systems with few kernels per state PDF.

2-D ZTM-HMM FR system performance is higher for the lower number of kernels per state PDF, i.e. 64 kernels or less. Within this region, the performance starts relatively low because of the low resolution, then improves as the number of kernels per state PDF increases up to a point where it starts to decline due to the lack of the robustness, i.e. the training data not being enough to efficiently estimate the model parameters. Starting again with the low kernels per state range, the 2-D FTM-HMM FR system performance is lower than that of the 2-D ZTM-HMM FR system because of the extremely low resolution, but it improves as the number of kernels per state PDF increases. Note that the whole system, including 40 models for the 40 different subjects, shares those kernels, which explains the inferior performance of 2-D FTM-HMM FR in this range.

As the number of kernels per state increases beyond 128, the performance of the FTM system is higher than that of the 2-D ZTM-HMM FR system. In this part of the graph, the 2-D ZTM-HMM has extremely low robustness due to the large number of parameters in the system, e.g. the case where each state in the system is exclusively using 512 kernels. On the other hand, all the states in the 2-D FTM-HMM FR system share the access to those kernels, resulting in better robustness. Since the number of kernels per state PDF is sufficiently high in this part of the graph, no resolution problem is encountered by either system.

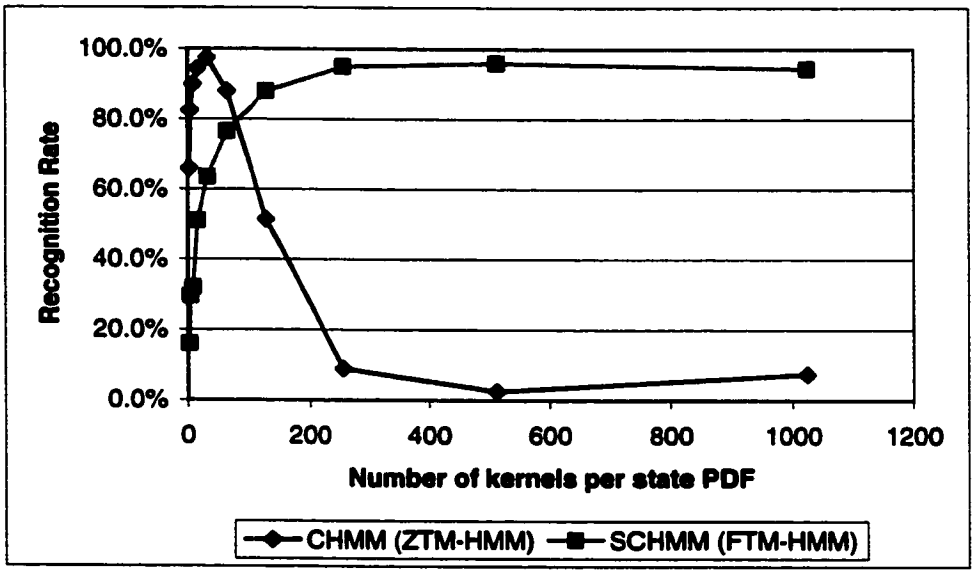


Figure 4-7. Recognition rate of 2-D ZTM-HMM, i.e. CHMM, and 2-D FTM-HMM, i.e. SCHMM, versus number of kernels per state.

The same results are plotted again in Figure 4-7 on a linear scale this time in order to emphasize another advantage of the 2-D FTM-HMM FR system, which is the insensitivity to change in number of parameters.

The performance of the 2-D ZTM-HMM FR system has a narrow peak in the low range then drops dramatically, while the performance of the 2-D FTM-HMM FR system is monotonically increasing in the low range and fairly stable in the range of 256 kernels and beyond.

This robustness to the choice of number of kernels per state PDF makes the 2-D FTM-HMM system easier to design compared with the 2-D ZTM-HMM system.

From the complexity point-of-view, the complexity of the 2-D FTM-HMM is slightly higher than the complexity of the 2-D ZTM-HMM that has the same number of kernels in the system. This slight increase in system complexity, notably in the observation layer, is due to the extra terms in the state PDF summation, (see the general form in Equation (4.1)). Fortunately, the 2-D FTM-HMM delivers better performance compared with the 2-D ZTM-HMM for the same number of kernels in the system. In other words, 2-D FTM-HMM with fewer kernels replaces a 2-D ZTM-HMM with more kernels for the same performance. This reduces the complexity of the 2-D FTM-HMM to the level of 2-D ZTM-HMM and even below.

4.3 Partially-Tied 2-D HMM Face Recognition

After having considered the two extremes of Zero-Tied and Fully-tied systems, we will now examine the range of solutions in-between these two extremes, i.e. the Partially-Tied-Mixture 2-D HMM (2-D PTM-HMM) with different tying degrees. The testing environment is the same as in the previous section, i.e. the same facial database, feature extraction, etc. First, a 2-D FTM-HMM is trained, where all the states in the system share the same kernels (universal) codebook. Then, the states are clustered into a number of state clusters that is equal to the target number of genones (each state cluster will be associated with an exclusive genone). The information-theoretic state clustering based on MWD entropy [46] is implemented. Once the state clusters in the system are determined, the genone of each state cluster is formed using a subset of the most probable kernels in the initial universal codebook given that state cluster, where the probability of a kernel $N_\chi^{(s)}$ given a state cluster χ that is associated with genone $\mathbf{N}_\chi$ is given by:

$$P(N_\chi^{(g)} \mid \chi) = \sum_{s_i \in \gamma - 1(N_\chi)} P(N_\chi^{(g)} \mid S_i) P(S_i) \tag{4.16}$$

where $\{P(N_\chi^{(g)} \mid S_i)\}$ is the mixture-weight distribution of the state S_i.

The number of kernels per genone is chosen such that the total number of kernels in the system remains unchanged across the clustering process. The experiment is carried out for 256, 512, and 1024 kernels in the system and for 2x2, 3x2, and 4x2 state models. The recognition rate is shown in Figure 4-10.

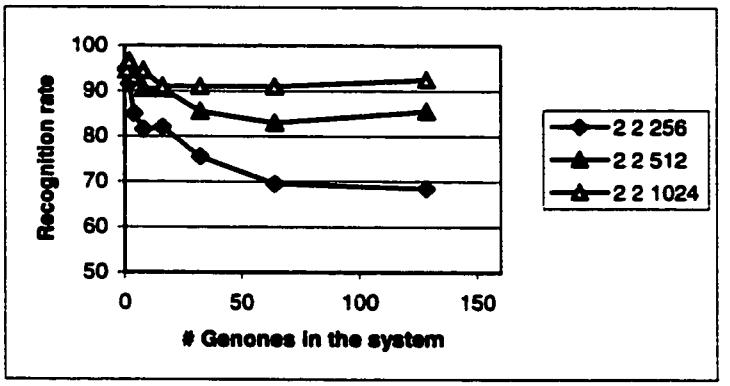


Figure 4-8. Recognition rate of the Partially-Tied-Mixture 2-D HMM (2-D PTM-HMM) against the number of genones for different number of kernels in the system. 2x2 states 2-D PTM-HMM with 256, 512, and 1024 kernels in system.

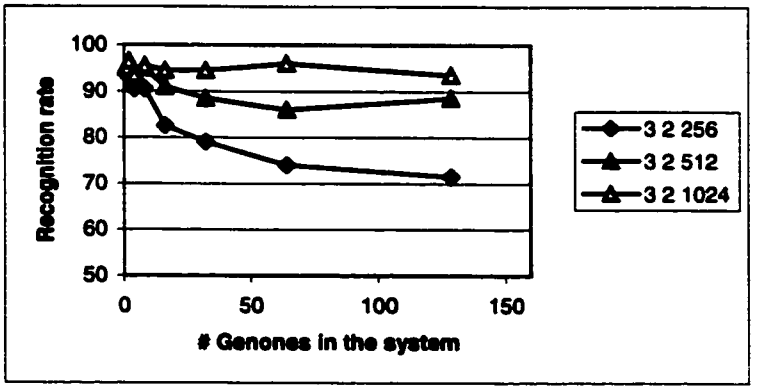


Figure 4-9. Recognition rate of the Partially-Tied-Mixture 2-D HMM (2-D PTM-HMM) against the number of genones for different number of kernels in the system. 3x2 states 2-D PTM-HMM with 256, 512, and 1024 kernels in system.

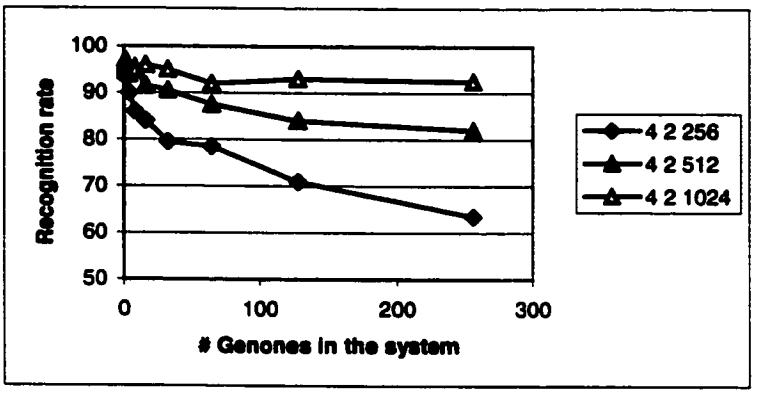


Figure 4-10. Recognition rate of the Partially-Tied-Mixture 2-D HMM (2-D PTM-HMM) against the number of genones for different number of kernels in the system. 4x2 states 2-D PTM-HMM with 256, 512, and 1024 kernels in system.

Results in Figure 4-8, Figure 4-9, and Figure 4-10 show that the recognition rate declines with the increase of the number of genones, i.e. with untying in the mixtures, when the number of kernels in the system is small, e.g. 256 kernels. This is expected because when there are only a few kernels in the system, it is better to share them to provide more robust estimation and better PDF resolution for all states. This agrees with the conclusion in the previous section that tying is better for small models. On contrary, tying does not improve much on the system performance when the number of kernels in the system is medium, e.g. 1024 kernels. These two observations are consistent across the different number of states in Figure 4-8, Figure 4-9, and Figure 4-10.

It should be noted that entropy-based state clustering is computationally costly, since a large number of computations is spent on MWD-entropy-based distance measure. This is a motivation to examine a simpler clustering scheme based on tying the mixtures of states that have the same location across all models, e.g. the states in the left-upper corner of the model are all tied together. Accordingly, the number of state clusters in the system

is equal to the number of states in the model. This topological state clustering scheme is rather empirical and does not require computational effort to determine the member states in each cluster. We will refer to this simple clustering as state-order-based clustering.

The results in Table 4-3 and Table 4-4 show the performance of the entropy-based state clustering and state-order-based clustering, respectively.

States per model		# States per model	# States in the system	# of State Clusters	# kernels in the system	Recognition Rate
# rows	#columns					
2	2	4	160	4	256	85.00%
2	2	4	160	4	512	93.00%
2	2	4	160	4	1024	93.00%
3	2	6	240	6	256	88.00%
3	2	6	240	6	512	93.00%
3	2	6	240	6	1024	97.00%
4	2	8	320	8	512	95.50%
4	2	8	320	8	1024	95.50%
4	2	8	320	8	2048	96.50%

Table 4-3 Performance of the 2-D Partially-tied HMM Face Recognition system, Entropy-Based State Clustering (The length of the state MWD is equal to the total number of kernels).

States per model		# States per model	# States in the system	# of State Clusters	Total #kernels	Recognition Rate
# rows	#columns					
2	2	4	160	4	256	85.50%
2	2	4	160	4	512	93.50%
2	2	4	160	4	1024	93.00%
3	2	6	240	6	256	88.50%
3	2	6	240	6	512	92.00%
3	2	6	240	6	1024	95.00%
4	2	8	320	8	512	94.00%
4	2	8	320	8	1024	98.00%
4	2	8	320	8	2048	98.00%

Table 4-4 Performance of the 2-D Partially-tied HMM Face Recognition system, state-order-Based State Clustering.

No significant difference of performance is observed between the two clustering approaches, except for large model cases. From the complexity point of view, the entropy-based state clustering requires more computations. However, it should be noted that state clustering is part of the training task that is performed off-line in the face recognition application. Thus, it has no implication on the recognition complexity.

4.4 Conclusion

In this chapter, the effect of mixture tying on the proposed 2-D second-order HMM has been studied when applied to face recognition. First, an introduction to parameter tying in HMM was presented. This highlighted the main advantage of parameter tying: reducing the need for large training data in the conventional HMM due to the large number of free parameters in the model. The introduction of this chapter also sheds some light on the major levels of parameter tying in HMM that range from Gaussian kernels tying up to model tying. The Continuous-density HMM and Semi-Continuous-Density HMM are approached from the parameter tying perspective as a zero-tied-mixture HMM and a fully-tied-mixture HMM, respectively. Located in an intermediate stage in generalized tying framework, partial tying does not call for sharing one codebook of PDF kernel functions, as opposed to full tying. Rather, more than one codebook in the system is available to be shared. Each codebook is shared among a set of states, i.e. *state cluster*. A state cluster can contain states from different models in the system and is formed either based on subjective knowledge or based on information-theoretic optimization. Examples from the literature are given to illustrate both state clustering methods.

Building on this introduction, we propose a fully-tied mixture face recognition system based on the 2-D HMM system presented in the previous chapter. The performance of the

proposed fully-tied-mixture 2-D HMM face recognition is shown to be better than that of the zero-tied-mixture 2-D HMM face recognition for the same or smaller number of kernels in the system. The difference in performance is highest for models with small number of kernels where the zero-tied-mixture 2-D HMM lacks resolution. This result confirms that mixture tying does not only provide robust estimation for model parameters (which it is known for), but also enhances the resolution in the parameter space in small models.

Also in this chapter, the performance of the partially-tied-mixture 2-D HMM face recognition system has been studied considering entropy-based state clustering with different tying degrees for different number of kernels in the system, which showed a steeper degradation in small model cases when moving towards lower degree of tying. On the other hand, the large computational load of entropy-based state clustering has been a motivation to try a simpler topological state clustering that requires no computation to form the state clusters in the system. The performance of the system shows no significant difference for the two clustering methods.

In conclusion, the proposed 2-D HMM benefits from mixture tying when applied to face recognition, particularly in the small model case, where both robustness and resolution are enhanced.

Chapter 5 Conclusions and Future Research

5.1 Conclusions

In this work, a low-complexity yet efficient 2-D HMM is proposed and applied to face recognition. The 2-D HMM as an extension to the 1-D HMM is known for its prohibitively high computational complexity. The proposed 2-D HMM system builds on a true 2-D HMM structure exploiting the conditional independence that arises among the neighboring feature blocks. Furthermore, the no-overlap strategy used in the proposed model, combined with appropriate DCT feature block size, makes the model compatible with JPEG-compressed databases with minimal feature extraction. The lower complexity of the system was achieved through the different factors that are depicted in Figure 3-16.

Design assumptions are provided, the likelihood evaluation and the Viterbi-based learning method used in the proposed model are described. A detailed discussion on the computational complexity of the model is also provided in this work along with performance-complexity comparison with similar HMM-based Face Recognition systems. The computational complexity of the proposed 2-D HMM is lower than that of the Markov Mesh Random Field 2-D HMM and other 2-D Pseudo HMMs while being comparable to those of the 1-D HMM systems. The proposed system delivers recognition rate up to 100%, depending on the model size, when tested on the facial database of *AT&T Laboratories Cambridge.*

Tying state mixture of the proposed 2-D second-order HMM has been studied, again as applied to face recognition. A fully-tied mixture 2-D HMM face recognition method is introduced based on the proposed model. In general, the performance of the fully-tied-

mixture 2-D HMM face recognition is shown to be better than that of the zero-tied-mixture 2-D HMM face recognition for the same or smaller number of kernels in the system, specially in small model sizes, i.e. fewer states. The difference in the performance is highest for models with small number of kernels where the zero-tied-mixture 2-D HMM lacks resolution. This confirms that mixture tying does not only provide robust estimation for model parameters (which it is known for), but also enhances the resolution in the parameters' space in small models.

Also in this work, the performance of the partially-tied-mixture 2-D HMM face recognition system has been studied considering entropy-based state clustering with different tying degrees for different number of kernels in the system, which showed a steeper degradation in small model cases when moving towards lower degree of tying. On the other hand, the large computational load of entropy-based state clustering has been a motivation to try a simpler topological state clustering that require no computation to form state clusters in the system. The performance of the system shows no significant difference for the two clustering methods.

In conclusion, this thesis proposes a low-complexity high performance 2-D HMM system. The system has been studied with varying degrees of mixture tying. The performance of the 2-D system has been shown to be very high at a level of complexity that is comparable to that of the 1-D system. It was shown that tying is very beneficial for smaller models due to sharing the system resources and thus maintaining low complexity.

5.2 Proposed Future Research Extension to Image Retrieval

5.2.1 Introduction

As the proposed model is intended for 2-D modeling in general, it is interesting to examine it in 2-D modeling applications other than face recognition. Image retrieval is an automated way to access image databases and is a strong candidate application for 2-D modeling methods. Due to the huge size of image databases nowadays, browsing the database contents is not a simple task and sometimes not even feasible. Instead of manual search, a query is supplied to the image retrieval system and the database is accordingly searched for best candidate set of images to match this query.

In the earlier days, Image Retrieval systems used to carry out a text-based type of search. The database to be searched has to be initially indexed, i.e. it contains a textual description to each image (and its contents) that is linked to the image. Text-based image retrieval is simple and fast. However, manual image annotation involves human intervention that may result in inconsistency when a large database is annotated by more than one person or images are annotated over a long period of time. For example, the newly-added images may be annotated in a different manner from that used in the already-existing images in the database. Also, the mismatch between the annotation rules that are used to describe the database contents and the textual query may lead to unexpected results. Another drawback of the text-based search appears in cases of query features that are hard to describe, e.g. searching for a similarly infected tissue in medical applications, or because the *intended* query contents were not originally anticipated and as such were not included in the time of indexing.

Alternatively, the database indexing process could be automated using pattern recognition techniques to annotate each image based on its contents with a finite set of visual elements, e.g. sky, clouds, ice, ... etc. Features such as texture, color and shape, are involved in the matching process to detect the presence of those elements in the image. A grammatical graph may be used to provide a higher level of description based on the relationship among scene elements, e.g. how trees, grass and sky would form a scene of a forest. The query can be pictorial-based, i.e. an image is first analyzed to its basic contents before the actual search is carried out, and it can be text-based as explained earlier. Although such high-level systems avoid the mismatch between the query and the database annotations, they are still subject to the vocabulary limitation of the system design.

Due to recent progress in computation devices, direct matching approaches that used to be prohibitively complex have become not only affordable but also efficient avoiding manual annotation problems. Statistical moments and histogram matching used to be a popular approach in this context as a low-level search. The problem with the low-level search is that it delivers image(s) that may be completely different from the query image in conceptual sense, yet their overall histogram and statistical moments are close. On the other hand, a medium-level search provides better overall results due to supporting locality in handling of the image features and their spatial relations. Medium-level search image retrieval systems accept man-made queries such as free-hand sketches and painted queries in addition to the regular image queries.

Clearly, HMM is a good tool for signal warping maintaining the relative spatial relations of the image features and tolerating the scale and translations differences which

suits the medium-level. 2-D PHMM was applied to the problem of color image retrieval in [12] for a pictorial-based system that performs a search through a general image database based on an image or painted query. A brief review of that work is found in Section 2.2.1. Later, Muller et al. [49] introduced an image retrieval system that searches a hand-tool image database with a sketched query. The system is based on a 1-D HMM that is modified to handle a possible 360-rotation that exists in the targeted database. The authors in [49] explained why they chose 1-D HMM over 2-D PHMM in their statement referring to the modifications they introduced, "Similar modifications might be difficult to integrate into the system presented by Lin et al. [12] due to the use of so-called pseudo-2D HMMs (P2DHMMs) which try to model two dimensional shapes but fail to model the dependency of consecutive columns if the rows are presented to the *superstates* of the P2DHMMs.". The modifications they referred to are about concatenating filler models to the original one. The filler models are similar to the original left-to-right 1-D HMM except for the extra state transitions that give access to the states of first filler model and provide exit to the states of the second filler model as shown in Figure 5-1. Only one model is trained and a duplicate is used to represent the filler model. However, later in the paper, the authors suggested a 2-D PHMM structure similar to their proposed model. Recently, Muller et al. integrated color features in their system in [50] through using multi-streams observation whose probability given the state is equal to the multiplication of the probabilities of the member streams. A 2-D PHMM extension to their work was also presented later in the paper.

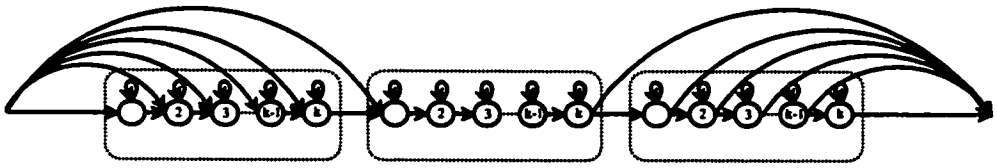


Figure 5-1. 1-D HMM for hand-tool database image retrieval [50].

The model introduced in [49] and [50] duplicates the states to three times the number of the states in the original model and uses a number of transitions that is almost five times that of the original model.

In brief, 1-D HMM is found to be useful for shape-based modeling for image retrieval, while for the texture-based modeling, there is a room of improvement over the 2-D PHMM.

5.2.2 Tied-Mixture 2-D HMM for Image Retrieval, Preliminary Work

We believe, the proposed low-complexity 2-D second-order HMM may be very useful if applied to the problem of image retrieval. As such, we did some preliminary work to investigate the applicability of the proposed model to image retrieval and to point out the expected problems that may arise due the nature of the application.

The image database used in this work is a subset from the CorelDraw collection of photos. The collection includes categories such as: *Ancient, Animals, Asia, Building, Coasts, Coins, Design, Europe, Fire, Food, Holidays, Industry, Mountain, Painting, People, Plants, Sunsets, Textures, Transportations, USA cities, Water, etc.* The image subset is extracted from this collection such that it contains 200 images from different categories. All the images are JPEG-coded and saved in JPG file format with either portrait or landscape aspects. The original size of the images is approximately 720x480

pixels for portrait images and approximately 480x720 for landscape images, which is reduced by a factor of 4 in both dimensions.

The proposed fully-tied-mixture 2-D HMM is trained by this set of images, a model per image, the same manner it is trained for face recognition. The 200 trained models share one universal Gaussian distribution codebook. For testing, the model that results in the best likelihood given the query image is selected as best match. Normally, *Feature Extraction* process is needed in training phase and in searching phase as depicted in Figure 5-2, where each image should be segmented into non-overlapped 8x8 blocks and each block is DCT-transformed into 8x8 coefficients. The HMM observation vector is formed by the first 3x3 coefficients that represents the low vertical and horizontal frequency bands of the luminance component of each block. Feature extraction here is reduced to retrieving the DCT coefficients and does not include forward or inverse transformation since images are saved in JPEG format, i.e. already transformed. That is to say, training and searching processes are carried out in the transform domain.

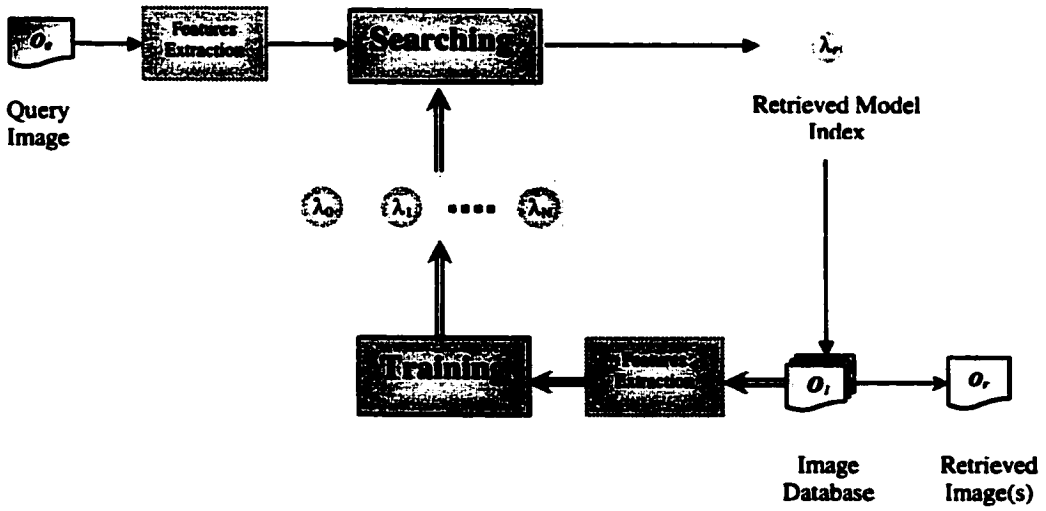


Figure 5-2. Block diagram of the proposed Image retrieval system.

Preliminary results are shown in Figure 5-3 and Figure 5-4, which obtained by 3x3 state models that share a codebook of 128 Gaussian kernel functions. The preliminary results show that the system always deliver a first best match image that is identical to the query image.

Considering the best few matches that follow the first best match, we observed that for query images with dominating low-frequency, the best few matches are acceptable as they are consistent with the query image, as depicted by the example in Figure 5-3. However, for the high-frequency dominated query image, the best few matches are not that consistent, as depicted by the example in Figure 5-4. This is obviously due to dropping the high frequency DCT coefficients from the features vector. Adding the higher frequency DCT coefficients to the features vector should fix this problem with a cost of extra complexity. We expect that incorporating the color information should enhance the system performance with complexity overhead that is less than that added with the higher frequency DCT coefficients.

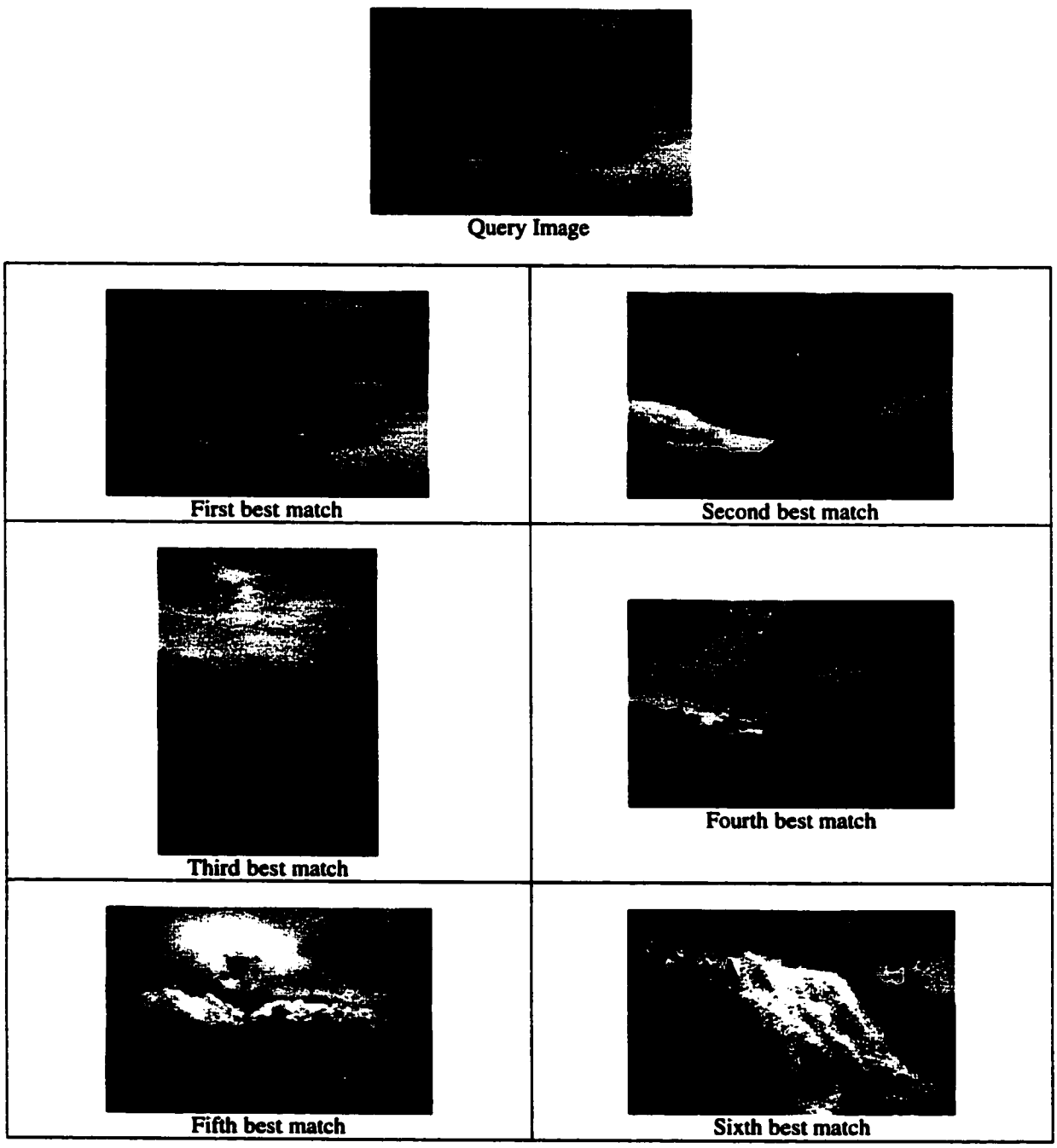
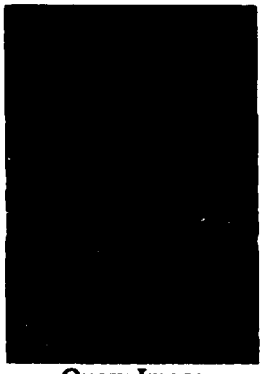
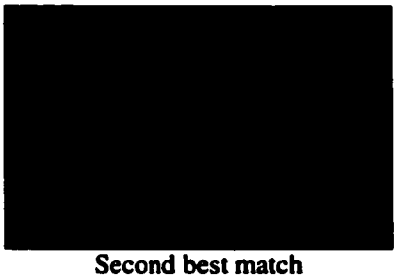


Figure 5-3. The best match set for a low-frequency query image.

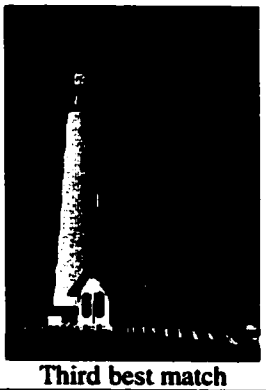


Query Image

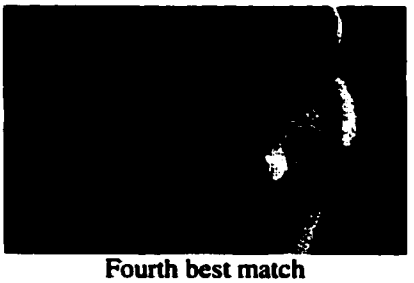
First best match



Second best match



Third best match



Fourth best match

Fifth best match

Sixth best match

Figure 5-4.The best match set for a high-frequency query image.

Future extension of this work may include:

1- Incorporating Color Information in the Observation Vector:

Considering color image databases, incorporating color information in the observation vector should enhance the performance of the image retrieval system. JPEG standards allow for three components, a luminance component Y, and two chrominance components; Cb and Cr. In the preliminarily work, only the luminance component is considered, i.e. all images are assumed grey -level form.

One problem that may arise is the dependency among the elements in the feature vector, i.e. the DCT coefficients. However, it is safe to assume that DCT coefficients of the luminance are independent since DCT nearly decorrelates normal images, with assumption of a multivariate Gaussian distribution. This results in huge saving in computations. This might not be the case when incorporating three components, possibly dependent, in the feature vector.

2- Supporting Hand-Painted Query Image:

The image retrieval system should support searching for best matches to a hand-painted image. The hand-painted image may contain don't-care areas, e.g. a user looking for images with a green tree in the middle irrespective of the background. The proposed model should be modified to handle such conditions.

3- Object Detection:

HMM has been applied to object detection in images. Examples from the literature include an Embedded HMM recognition system that examines all possible areas in the image with a model of the query object [51]. This requires significant computations and does not deeply benefit from the warping in HMM. Another example of HMM-based

object detection is found in [52] where extra surrounding states (representing the cluttered background) are added around the 2-D PHMM that originally represents of the query object. This direction seems appealing since it exploits the warping nature of the HMM, however that is not true 2-D model. Future work may include applying the proposed 2-D HMM in similar manner, i.e. outer circumference of states are added to introduce the background representation to the model of the query object. Alternatively, the proposed 2-D HMM model can be modified to support state transitions among different models in a manner that is similar to the transition among 1-D HMM model in connected speech recognition. Clearly, it will be significantly more complex in the 2-D case.

APPENDICES

Appendix A The Facial Database

The facial database used in this work is a set of facial images taken between April 1992 and April 1994 at the AT&T Laboratories, Cambridge [32], formerly known as Olivetti Research Laboratory (ORL).

There are 10 different images of each of 40 distinct subjects. For some of the subjects, the images were taken at different times and slightly varying in lighting, facial expressions (open/closed eyes, smiling/non-smiling) and facial details (glasses/no-glasses). All the images are taken against a dark homogeneous background and the subjects are in up-right, frontal position (with tolerance for some side movement). The images are 256-grey level and 112x92-pixel each. The database can be retrieved from ftp://ftp.uk.research.att.com:pub/data/att_faces.tar.Z. A preview of the database contents is provided below.

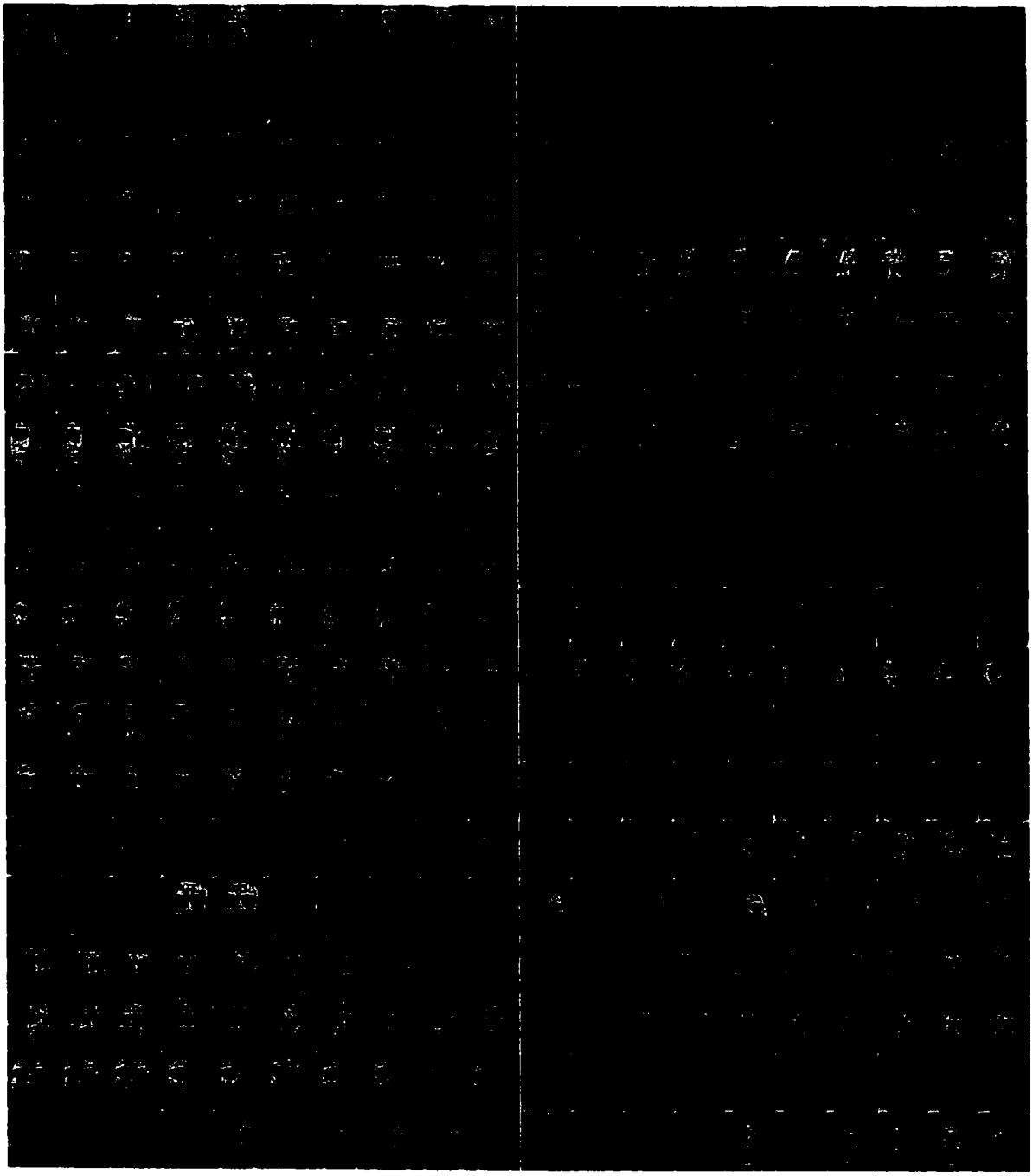


Figure A-1. Facial database of AT&T Laboratories, Cambridge [32].

Appendix B An Intuitive Understanding of the Proposed Model

B.1 Introduction

In this Appendix, we will examine the functionality of the proposed 2-D HMM model with the objective of developing an intuitive, rather than a mathematical, understanding of the model, the meaning of its parameters and how one can relate a specific image to the states and PDF representation of the model.

We will approach this intuitive understanding through using the proposed model to represent a very simple image that is composed of primitive components. This will allow us to correlate the model parameters with our intuitive description of this image.

Through varying the number of states and PDF kernels, we hope to develop some insight into how the HMM successfully represents more complex images. The model structure and its assumptions are not stated here but they are detailed earlier in Chapter 3.

In the HMM framework, different portions of the image to be modeled are assumed to be generated by different stationary sources. Each source has its own statistical properties and is represented by a "state". The statistical properties of each source (i.e. state) dictate the nature of the output observation it produces. Thus, relating the different portions of the image to the sources (states) assumed to have generated them should be feasible. The reason for referring to these sources as "states" is that they do not function independently of each other. Rather, they work together in harmony to produce the whole image. Alternatively, we can think of the image to be modeled as if it is generated by one major source. This major source is a multi-modal one whose properties alternate among a

finite number of modes. The term "state" refers to the mode of the source when a specific observation was produced.

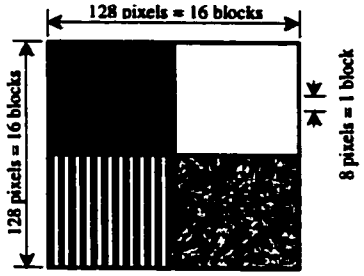


Figure B-2. The testing image.

Consider a simple image composed of four portions: black, white, strips and texture, as depicted in Figure B-2. For the sake of localization in the spatial domain, we observe the image in terms of groups of pixels, called observation blocks. We refer to the relation between the state and the observation blocks that it can produce as *state properties*. The detailed description of the statistical properties of the states is handled by the observation layer. The state statistical properties are formed in terms of either a continuous Probability Density Function (PDF) or a discrete Probability Distribution on the observation space. The continuous PDF can be approximated by a weighted sum of prototype functions, e.g. Gaussian distributions, referred to as kernels here.

The number of states, the number of kernels in each state that describe its statistical properties and the initial values of the model parameters are among the factors that control the efficiency of the model for recognition tasks.

B.2 The Effect of Model Size

As a general rule, modeling accuracy increases with the increase of the number of model parameters. For example, the accuracy of a16-state model is better than that of a 4-

state model, and so on for the rest of model parameters. However, the modeling accuracy is different from successful recognition rate, as we will explain shortly.

In pattern recognition applications where the model is used to recognize a class of observed data, the process is composed mainly of two phases: learning (training) and recognition (testing). In the training phase, some examples, i.e. the training data, are supplied. These examples, that are known beforehand, are used to teach or train the model. The ability of the model to appropriately recognize similar examples is tested with the test data in the testing phase. Normally, the training data set and the testing data set are exclusive of each other. If the model is too accurate in modeling the training data, it may fail to recognize the testing examples because of the relatively slight differences. This is known as overfitting problem where the model is too training-data-dependant to allow for differences within the target class.

Models with fewer parameters have their own limitation. They tend to average the information over the class, which results in missing some details that distinguish members of this class from members of other classes. This results in wrong recognition.

In the following section, we will discuss the practical impact of the number of states and the number of kernels on the behavior of the proposed model.

B.3 Understanding of the Roles of the HMM Parameters

The proposed HMM is used to represent the synthetic image in Figure B-2 with different number of states and kernels. At the beginning of the training session, the image is uniformly segmented into a number of patches that is equal to the number of states in the model. The state PDF is initialized using the observation blocks in the corresponding patch. The transition probabilities and the initial state probabilities are assumed initially

equal. The segmentation outcome shows how the model recognizes the image in the final training iteration. Each segment represents a portion that has been recognized as being generated by a state that is different from the state that generated the neighboring segment.

Different model sizes were used in this experiment to understand the impact of the model size. We will refer to a model that has N_0 rows and N_1 columns of states with G_s kernels for each state PDF, as $N_0xN_1xG_s$ model.

2x2x1 model

For a $N_0xN_1xG_s$ model, the image is segmented into N_0xN_1 patches and each patch is used to initialize an associated state PDF. We start with a 2x2x1 model that has four states arranged in two rows and two columns. Each state has one kernel that it can use to generate its associated observations. The image initial segmentation is given in Figure B-3. The state transition probabilities are uniformly initialized as it is shown in Table B-1 and Table B-2. Based on state transition topology defined in (3.3) and (3.4), vertical state transitions can occur from state $S_{1,1}$ to $S_{1,1}$, $S_{1,2}$, $S_{2,1}$, and $S_{2,2}$. These 4 transitions are assumed initially equally probable. Given that they should add-up to 1, the probability of each of these transitions is equal to 0.25, i.e. 1 divided by the number of possible transitions form state $S_{1,1}$. This is manifested in the first row of Table B-1 that includes probabilities of vertical state transitions from $S_{1,1}$. The same discussion applies to the second row of Table B-1 that includes probabilities of vertical state transitions from $S_{1,2}$, the first row of Table B-2 that includes probabilities of horizontal state transitions from $S_{1,1}$, and the third row of Table B-2 that includes probabilities of horizontal state transitions from $S_{2,1}$.

Again, based on state transition topology defined in (3.3) and (3.4), vertical state transitions can occur from state $S_{2,1}$ to $S_{2,1}$ and $S_{2,2}$. These 2 transitions are assumed initially equally probable. Given that they should add-up to 1, the probability of each of these transitions is equal to 0.5, i.e. 1 divided by the number of possible transition form state $S_{2,1}$. This is manifested in the third row of Table B-1 that includes probabilities of vertical state transitions from $S_{2,1}$. The same discussion applies to the fourth row of Table B-1 that includes probabilities of vertical state transitions from $S_{2,2}$, the second row of Table B-2 that includes probabilities of horizontal state transitions from $S_{1,2}$, and the fourth row of Table B-2 that includes probabilities of horizontal state transitions from $S_{2,2}$. Obviously, the sum of the elements of each row in Table B-1 and Table B-2 should be 1.

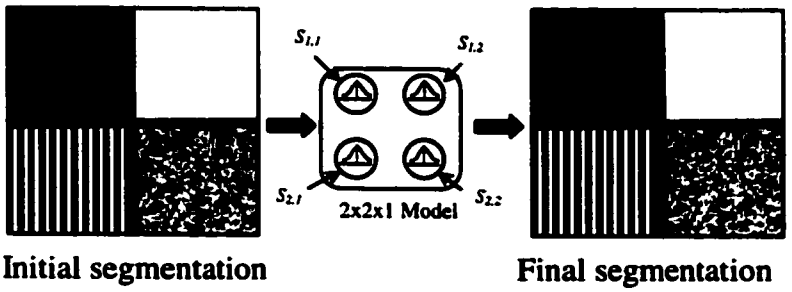


Figure B-3. The initial and final segmentation for 2x2x1 model.

From	To			
	$S_{1,1}$	$S_{1,2}$	$S_{2,1}$	$S_{2,2}$
$S_{1,1}$	0.25	0.25	0.25	0.25
$S_{1,2}$	0.25	0.25	0.25	0.25
$S_{2,1}$	0	0	0.5	0.5
$S_{2,2}$	0	0	0.5	0.5

Table B-1. Initial vertical transition probabilities (before the 2x2x1 model training).

From	To			
	$S_{1,1}$	$S_{1,2}$	$S_{2,1}$	$S_{2,2}$
$S_{1,1}$	0.25	0.25	0.25	0.25
$S_{1,2}$	0	0.5	0	0.5
$S_{2,1}$	0.25	0.25	0.25	0.25
$S_{2,2}$	0	0.5	0	0.5

Table B-2. Initial horizontal transition probabilities (before the 2x2x1 model training).

At the end of the training process, the final segmentation is the same as the initial segmentation, shown in Figure B-3. As expected, the left-upper state, $S_{1,1}$, represents the black area, the right-upper state, $S_{1,2}$, represents the white area, the left-lower state, $S_{2,1}$, represents the strips area and the right-lower state, $S_{2,2}$, represents the texture area. This is reflected in the values of the model parameters after training.

From	To			
	$S_{1,1}$	$S_{1,2}$	$S_{2,1}$	$S_{2,2}$
$S_{1,1}$	0.875	ε	0.125	ε
$S_{1,2}$	ε	.875	ε	0.125
$S_{2,1}$	ε	ε	1	ε
$S_{2,2}$	ε	ε	ε	1

(ε≅0)

Table B-3. Final vertical transition probabilities (after the 2x2x1 model training).

From	To			
	$S_{1,1}$	$S_{1,2}$	$S_{2,1}$	$S_{2,2}$
$S_{1,1}$	0.875	0.125	ε	ε
$S_{1,2}$	ε	1	ε	ε
$S_{2,1}$	ε	ε	0.875	.0125
$S_{2,2}$	ε	ε	ε	1

Table B-4. Final horizontal transition probabilities (after the 2x2x1 model training).

Although the transition probabilities were uniformly initialized, they are automatically adjusted through the training process to reflect the final segmentation as depicted in Table B-3 and Table B-4. For example, the probability of a vertical state transition from state $S_{1,1}$ to state $S_{1,1}$ being 0.875 expresses that the frequency of the vertical transitions from $S_{1,1}$ to itself is 87.5% of all transitions from this state. Examining

the image closely, state $S_{1,1}$ has been visited 64 times (to generate the 64 observation blocks in the upper-left segment of the image), consequently, state transition took place out of the state 64 times vertically and 64 times horizontally. Among those vertical transitions, there are 56(7x8) transitions from the state to itself over two vertically consecutive blocks (the top 7 rows of blocks and 8 most-left columns of blocks). The ratio 56:64 makes 0.875. On the 8^{th} row of blocks, 8 vertical transition take place from state $S_{1,1}$ and state $S_{2,1}$ along the 8 most-left columns of blocks. This makes it 8 transitions from state $S_{1,1}$ and state $S_{2,1}$ out of 64 transitions from state $S_{1,1}$. The ratio here is 0.125 which is identical to the final value of the probability of vertical transition from $S_{1,1}$ to $S_{2,1}$. It is easy to extend the previous discussion to verify the remaining trained state transition probability.

Regarding the PDF kernels, the PDF kernel in the state $S_{1,1}$ has mean values that are identical to the 2-D DCT of the all-black block (all zeros) as shown in Table B-5. Similarly, the PDF kernel in the state $S_{1,2}$ has mean values that are identical to the 2-D DCT of the all-white block (all zeros except the DC coefficient is equal to 2040), the PDF kernel in the state $S_{2,1}$ has mean values that are identical to the 2-D DCT of the strips block and the PDF kernel in the state $S_{2,2}$ has mean values that are identical to the mean values of the 2-D DCT of the texture blocks as shown in Table B-7, Table B-9 and Table B-11, respectively .

The variance of the PDF kernels is (almost) zeros for all states, except for the right-lower state, $S_{2,2}$, as depicted in Table B-6, Table B-8, Table B-10 and Table B-12. This is because the 8x8-pixel blocks in each of the four areas, except the texture area, are typically the same.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Table B-5. Mean vector of kernel 1 of state $S_{1,1}$ of the 2x2x1 model.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	ε	ε	ε	ε	ε	ε	ε	ε
2	ε	ε	ε	ε	ε	ε	ε	ε
3	ε	ε	ε	ε	ε	ε	ε	ε
4	ε	ε	ε	ε	ε	ε	ε	ε
5	ε	ε	ε	ε	ε	ε	ε	ε
6	ε	ε	ε	ε	ε	ε	ε	ε
7	ε	ε	ε	ε	ε	ε	ε	ε
8	ε	ε	ε	ε	ε	ε	ε	ε

Table B-6. Variance vector of kernel 1 of state $S_{1,1}$ of the 2x2x1 model.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	2040	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Table B-7. Mean vector of kernel 1 of state $S_{1,2}$ of the 2x2x1 model.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	ε	ε	ε	ε	ε	ε	ε	ε
2	ε	ε	ε	ε	ε	ε	ε	ε
3	ε	ε	ε	ε	ε	ε	ε	ε
4	ε	ε	ε	ε	ε	ε	ε	ε
5	ε	ε	ε	ε	ε	ε	ε	ε
6	ε	ε	ε	ε	ε	ε	ε	ε
7	ε	ε	ε	ε	ε	ε	ε	ε
8	ε	ε	ε	ε	ε	ε	ε	ε

Table B-8. Variance vector of kernel 1 of state $S_{1,2}$ of the 2x2x1 model.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	1020	-382.837	0	-783.541	0	523.546	0	76.151
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Table B-9. Mean vector of kernel 1 of state $S_{2,1}$ of the 2x2x1 model.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	ε	ε	ε	ε	ε	ε	ε	ε
2	ε	ε	ε	ε	ε	ε	ε	ε
3	ε	ε	ε	ε	ε	ε	ε	ε
4	ε	ε	ε	ε	ε	ε	ε	ε
5	ε	ε	ε	ε	ε	ε	ε	ε
6	ε	ε	ε	ε	ε	ε	ε	ε
7	ε	ε	ε	ε	ε	ε	ε	ε
8	ε	ε	ε	ε	ε	ε	ε	ε

Table B-10. Variance vector of kernel 1 of state $S_{2,1}$ of the 2x2x1 model.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	1633.63	-6.02232	2.72587	3.04164	-7.84961	-1.53541	-5.05065	-0.79796
2	6.64981	-2.14162	5.16479	-2.19145	2.09522	1.16125	5.02131	-0.466364
3	7.11987	-2.82748	-3.79109	-3.3233	0.175669	-1.59546	0.411711	3.79717
4	4.1446	-3.17571	0.200345	5.85162	0.242184	-0.451119	7.36525	-0.116071
5	1.48633	-3.5118	5.69535	2.86849	-6.56445	6.35645	1.63857	-0.212785
6	-6.01658	-1.2929	5.82259	3.30857	-5.27927	-5.14064	-3.58659	-4.87018
7	-0.342528	5.49498	-2.86173	5.71008	-2.45952	-2.64469	2.40438	-5.82351
8	-2.92635	1.12145	-2.91079	-8.07616	2.67568	-2.0481	0.865662	-0.108419

Table B-11. Mean vector of kernel 1 of state $S_{2,2}$ of the 2x2x1 model.

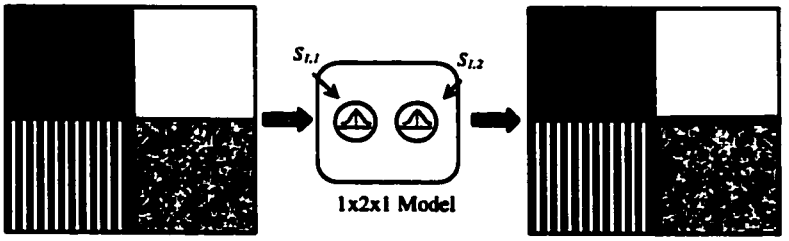
Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	7.4e+003	1.6e+003	1.4e+003	1e+003	8.6e+002	1.3e+003	6.7e+002	8.4e+002
2	2.6e+003	1.3e+003	9.6e+002	1.2e+003	8.6e+002	7.2e+002	7.5e+002	7.2e+002
3	1.4e+003	8.5e+002	7.7e+002	8.2e+002	8.2e+002	6.8e+002	8.5e+002	7.3e+002
4	1e+003	1.1e+003	1.1e+003	1.1e+003	7.6e+002	4.7e+002	5.4e+002	6.3e+002
5	8e+002	7.7e+002	9.7e+002	5.3e+002	8e+002	4.9e+002	7.5e+002	8.3e+002
6	1e+003	8.8e+002	8.5e+002	6.2e+002	5.5e+002	7.2e+002	7.8e+002	6.4e+002
7	1.1e+003	5.5e+002	5.9e+002	7.4e+002	6.1e+002	5.8e+002	5.3e+002	4.6e+002
8	7.2e+002	9.3e+002	7.2e+002	5.7e+002	4.7e+002	6.5e+002	4.3e+002	5.6e+002

Table B-12. Variance vector of kernel 1 of state $S_{2,2}$ of the 2x2x1 model.

1x2x1 Model

The previous experiment has been repeated for a 1x2x1 model. This model has two states lined in one row. Each state has a PDF with one kernel. The initial segmentation in

Figure B-4 is used to initialize the PDFs of the states while the state transition probabilities were uniformly initialized as depicted in Table B-13 and Table B-14.



Initial segmentation Final segmentation

Figure B-4. The initial and final segmentation for 1x2x1 model.

From	To	
	$S_{1,1}$	$S_{1,2}$
$S_{1,1}$	.5	.5
$S_{1,2}$	.5	.5

Table B-13. Vertical transition probabilities before the 1x2x1 model training.

From	To	
	$S_{1,1}$	$S_{1,2}$
$S_{1,1}$	.5	.5
$S_{1,2}$	0	1

Table B-14. Horizontal transition probabilities before the 1x2x1 model training.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	510	-191.418	0	-391.771	0	261.773	0	38.0755
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Table B-15. Mean vector of kernel 1 of state $S_{1,1}$ of the 1x2x1 model before training.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	2.6e+005	3.7e+004	ε	1.5e+005	ε	6.9e+004	ε	1.4e+003
2	ε	ε	ε	ε	ε	ε	ε	ε
3	ε	ε	ε	ε	ε	ε	ε	ε
4	ε	ε	ε	ε	ε	ε	ε	ε
5	ε	ε	ε	ε	ε	ε	ε	ε
6	ε	ε	ε	ε	ε	ε	ε	ε
7	ε	ε	ε	ε	ε	ε	ε	ε
8	ε	ε	ε	ε	ε	ε	ε	ε

Table B-16. Variance vector of kernel 1 of state $S_{1,1}$ of the 1x2x1 model before training.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	1836.82	-3.01116	1.36293	1.52082	-3.9248	-0.767703	-2.52532	-0.39898
2	3.3249	-1.07081	2.5824	-1.09572	1.04761	0.580625	2.51065	-0.233182
3	3.55994	-1.41374	-1.89555	-1.66165	0.0878343	-0.797729	0.205856	1.89858
4	2.0723	-1.58786	0.100173	2.92581	0.121092	-0.22556	3.68262	-0.058036
5	0.743164	-1.7559	2.84768	1.43424	-3.28223	3.17822	0.819286	-0.106393
6	-3.00829	-0.646451	2.9113	1.65428	-2.63963	-2.57032	-1.79329	-2.43509
7	-0.171264	2.74749	-1.43086	2.85504	-1.22976	-1.32234	1.20219	-2.91176
8	-1.46318	0.560723	-1.4554	-4.03808	1.33784	-1.02405	0.432831	-0.05421

Table B-17. Mean vector of kernel 1 of state $S_{1,2}$ of the 1x2x1 model before training.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	4.5e+004	8.2e+002	7e+002	5e+002	4.4e+002	6.6e+002	3.4e+002	4.2e+002
2	1.3e+003	6.3e+002	4.8e+002	5.8e+002	4.3e+002	3.6e+002	3.8e+002	3.6e+002
3	7e+002	4.3e+002	3.9e+002	4.1e+002	4.1e+002	3.4e+002	4.2e+002	3.7e+002
4	5e+002	5.7e+002	5.7e+002	5.4e+002	3.8e+002	2.3e+002	2.8e+002	3.1e+002
5	4e+002	3.9e+002	4.9e+002	2.7e+002	4.1e+002	2.6e+002	3.8e+002	4.2e+002
6	5.2e+002	4.4e+002	4.3e+002	3.1e+002	2.8e+002	3.7e+002	3.9e+002	3.2e+002
7	5.5e+002	2.8e+002	3e+002	3.8e+002	3.1e+002	2.9e+002	2.6e+002	2.4e+002
8	3.6e+002	4.6e+002	3.6e+002	3e+002	2.4e+002	3.3e+002	2.1e+002	2.8e+002

Table B-18. Variances of kernel 1 of state $S_{1,2}$ of the 1x2x1 model before training.

The final segmentation after training is depicted in Figure B-4. It is easy to verify that the values of the model parameters reflect this segmentation. For example, the left state, $S_{1,1}$, represents the black, white and strips portions of the image. Hence, the PDF kernel of the state $S_{1,1}$ has a mean vector that is the average of the 2-D DCT of the all-black, all-white and strips blocks. The model parameters after training are depicted in Table B-19 to Table B-24.

From	To	
	$S_{1,1}$	$S_{1,2}$
$S_{1,1}$	0.956522	.0434783
$S_{1,2}$	ε	1

Table B-19. Vertical transition probabilities after the 1x2x1 model training.

From	To	
	$S_{1,1}$	$S_{1,2}$
$S_{1,1}$	0.956522	.0434783
$S_{1,2}$	ε	1

Table B-20. Horizontal transition probabilities after the 1x2x1 model training.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	1020	-127.612	0	-261.18	0	174.515	0	25.3837
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Table B-21. Mean vector of kernel 1 of state $S_{1,1}$ of the 1x2x1 model after training.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	6.9e+005	3.3e+004	ε	1.4e+005	ε	6.1e+004	ε	1.3e+003
2	ε	ε	ε	ε	ε	ε	ε	ε
3	ε	ε	ε	ε	ε	ε	ε	ε
4	ε	ε	ε	ε	ε	ε	ε	ε
5	ε	ε	ε	ε	ε	ε	ε	ε
6	ε	ε	ε	ε	ε	ε	ε	ε
7	ε	ε	ε	ε	ε	ε	ε	ε
8	ε	ε	ε	ε	ε	ε	ε	ε

Table B-22. Variance vector of kernel 1 of state $S_{1,1}$ of the 1x2x1 model after training.

Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	1633.63	-6.02232	2.72587	3.04164	-7.84961	-1.53541	-5.05065	-0.79796
2	6.64981	-2.14162	5.16479	-2.19145	2.09522	1.16125	5.02131	-0.466364
3	7.11987	-2.82748	-3.79109	-3.3233	0.175669	-1.59546	0.411711	3.79717
4	4.1446	-3.17571	0.200345	5.85162	0.242184	-0.451119	7.36525	-0.116071
5	1.48633	-3.5118	5.69535	2.86849	-6.56445	6.35645	1.63857	-0.212785
6	-6.01658	-1.2929	5.82259	3.30857	-5.27927	-5.14064	-3.58659	-4.87018
7	-0.342528	5.49498	-2.86173	5.71008	-2.45952	-2.64469	2.40438	-5.82351
8	-2.92635	1.12145	-2.91079	-8.07616	2.67568	-2.0481	0.865662	-0.108419

Table B-23. Mean vector of kernel 1 of state $S_{1,2}$ of the 1x2x1 model after training.

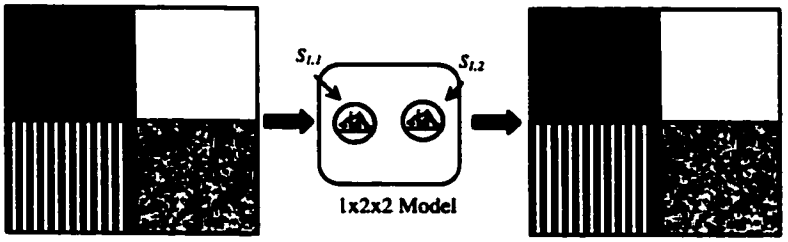
Vertical index	The horizontal index of the DCT coefficients							
	1	2	3	4	5	6	7	8
1	7.4e+003	1.6e+003	1.4e+003	1e+003	8.6e+002	1.3e+003	6.7e+002	8.4e+002
2	2.6e+003	1.3e+003	9.6e+002	1.2e+003	8.6e+002	7.2e+002	7.5e+002	7.2e+002
3	1.4e+003	8.5e+002	7.7e+002	8.2e+002	8.2e+002	6.8e+002	8.5e+002	7.3e+002
4	1e+003	1.1e+003	1.1e+003	1.1e+003	7.6e+002	4.7e+002	5.4e+002	6.3e+002
5	8e+002	7.7e+002	9.7e+002	5.3e+002	8e+002	4.9e+002	7.5e+002	8.3e+002
6	1e+003	8.8e+002	8.5e+002	6.2e+002	5.5e+002	7.2e+002	7.8e+002	6.4e+002
7	1.1e+003	5.5e+002	5.9e+002	7.4e+002	6.1e+002	5.8e+002	5.3e+002	4.6e+002
8	7.2e+002	9.3e+002	7.2e+002	5.7e+002	4.7e+002	6.5e+002	4.3e+002	5.6e+002

Table B-24. Variance vector of kernel 1 of state $S_{1,2}$ of the 1x2x1 model after training.

The initial segmentation is suggesting $S_{1,1}$ represents the black and the strips portions of the image and $S_{1,2}$ represents the white and the texture portions of the image. The state resources are limited, i.e. one kernel per state. Hence, the observation blocks represented by a state must be close to each other. The blocks in the white portion of the image prefer to be represented by state $S_{1,1}$ that represents the black and the strips portions. Note

that, the black, the white and the strips blocks are closer to each other in the transform space compared with the texture block. They have 59 coefficients (out of 64 coefficients) are identically the same, i.e. zeros. This also means zero variances for these coefficients.

For the 2-state model to be able to host the white and the texture portion by one state, the model states should be granted more resources, i.e. kernels.



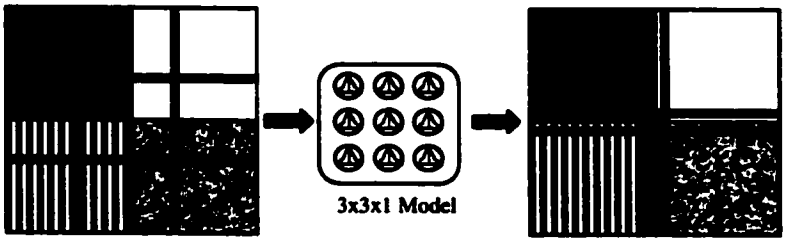
Initial segmentation Final segmentation

Figure B-5. The initial and final segmentation for 1x2x2 model.

When each state has two kernels, i.e. 1x2x2 model, the final segmentation converges after training to the initial segmentation as depicted Figure B-5. In this case, each kernel represents different portions of the image.

This is particularly useful if the state is needed to represent a portion of the image that is statistically inhomogeneous, i.e. contains different features.

3x3x1 Model



Initial segmentation Final segmentation

Figure B-6. The initial and final segmentation for 3x3x1 model.

Figure B-6 shows an interesting outcome when a 3x3x1 model is used to segment 4-portion image. Initially, the image is uniformly segmented, i.e. divided into nine equal segments. Each segment may contain different types of observation blocks, e.g. white blocks, black blocks, or white-black-border blocks, which initialize one of the states. After the model is trained, each state settles on more suitable patches, as depicted in the final segmentation. The four corner states choose to represent the pure observations, i.e. pure white, pure black, pure texture, or pure strips. Four other intermediate states choose to represent the area at the border between two different portions of the image, e.g. white-black-border area. A 9^{th} state is needed to represent the area where the four different portions of the image meet at the center of the image. This state is physically at the center of the 3x3 model.

To summarize, if the state resources are scarce, the model tends to average over the data space. Hence, the model is largely driven by the statistical features of the image contents, as opposed to the initial segmentation. As it was demonstrated in the case of 1x2x1 model that has only two states with only one kernel in each. This small model failed to follow the initial segmentation after being trained. This is because the poor single-kernel state PDFs tend to average the observation features. While in the case of 1x2x2 model where 2 kernels are granted to each state PDF, the model can easily follow a reasonable initial segmentation. Extending this discussion further, when the model is large and rich in resources, it tends to fit the observation features closely resulting in a final segmentation that close to the initial one. Equivalently, a model with a large number of parameters is expected to be trapped in a local minimum that is more likely to be near the initial solution.

References:

[1] John S. Boreczky and Lynn D. Wilcox, "A Hidden Markov Model Framework for Video Segmentation Using Audio and Image Features," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 98), vol. VI, pp. 3741-3744.

[2] Peter Morguet and Manfred Lang, "An Integral Stochastic Approach to Image Sequence Segmentation and Classification," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 98), vol. V, pp. 2705-2708.

[3] M Schuster, G. Rigoll, "Fast Online Video Image Sequence Recognition with statistical Methods," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 96), Vol. VI, pp. 3451-3454.

[4] G. Rigoll, Anderas Kosmala, "New Improved Feature Extraction Method For Real-Time Performance Image Sequence Recognition," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 97), Vol. 4, pp. 2901-2904,.

[5] Tsuhan Chen and Ram R. Rao, "Audio-Visual Interaction in Multimedia Communication," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 1997), Vol. 1, pp. 179-182.

[6] Takashi Masuko, Takao Kobayashi, Masatsune, Jun Masubuchi, and Keiichi Tokuda, "Text-to-Visual Speech Synthesis Based On Parameter Generation from HMM," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 1998), vol. VI, pp. 3745-3748.

[7] Juergen Luettin, Neil A. Thacker and Steve W. Beet, "Speachreading Using Shape and Intensity Information," Proc. 4th Int'l Conf. on Spoken Language Processing (ICSLP 1996).

[8] Juergen Luettin, Neil A. Thacker and Steve W. Beet, "Visual Speech Recognition using Active Shape Models and Hidden Markov Models," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 1996), Vol. II, pp. 817-820.

[9] Lawrence R. Rabiner, "A Tutorial On Hidden Markov Models and Selected Application in Speech Recognition," Proc. IEEE, Vol. 77, No., 2, February 1989.

[10] Ferdinando Silvestro Samaria, "Face Recognition Using Hidden Markov Model," A dissertation, Ph. D., University of Cambridge1994.

[11] Oscar E. Agazzi Shyh-shiaw Kuo, Esther Levin, and Roperto Pieraccini, "Connected and Degraded Text Recognition Using Planar Hidden Markov Models," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 1993), vol. V, pp. 113-116.

[12] Hsin-Chih Lin, Ling-Ling Wang, and Shi-Nine Yang, "Color Image Retrieval Based On Hidden Markov Model," IEEE Trans. Image Processing, Vol.6, No. 2, February 1997.

[13] S. Kuo and O. Agazzi, "Keyword Spotting in Poorly Printed Documents using Pseudo 2-D Hidden Markov Models," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 16, pp. 842-848, August 1994.

[14] Ara Nefian and Monson H. Hayes III, "An Embedded HMM for Face Detection and Recognition," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 1999), vol. VI, pp. 3553-3556.

[15] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," IEEE Trans. on PAMI, Vol. 6, No. 6, pp. 721-741, Nov. 1984.

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

[17] J. Zhang, P Fieguth, and D Wang, "Random Field Models," Handbook of Image and Video Processing, Al Povik, Academic Press, 2000.

[18] Hee-Seon Park and Seong-Whan Lee, "A truly 2-D Hidden Markov Model For Off-Line Handwritten Character Recognition," Pattern Recognition, Vol. 31, No. 12, pp.1849-1864, Dec. 1998.

[19] Rama Chellappa, Charles L. Wilson, Saad Sirohey, "Human and Machine Recognition: A Survey," Proc. IEEE Vol.83, No. 5, May 1995.

[20] Ming-Hsuan Yang, David J. Kriegman,and Narendra Ahuja, "Detecting Faces in Images: A Survey," IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol. 24, No. 1, January 2002.

[21] Anil K. Jain, "Fundamentals of Digital Signal Processing," Prentice-Hall International, Inc., 1989.

[22] Alex Pentland, Baback Moghaddam, Thad Starner, "Viwe-Based Modular Eigenspaces for Face Recognition," IEEE Conf. on Computer Vision & Pattern Recognition, 1994.

[23] Ali N. Akansu and Richard A. Haddad, "Multiresolution Signal Decomposition," Academic Press, Inc. 1992.

[24] Gilbert Strang and Troung Nguyen, "Wavelets and Filter Banks," Wellesley-Cambridge Press, 1996.

[25] International Telecommunication Union (ITU), "Information Technology-Digital Compression and Coding of Continuous-Tone Still Images-Requirements and

Guidelines," Recommendation T.81 (09/92), The International Telegraph and Telephone Consultative Committee (CCITT), Terminal Equipment and Protocols for Telematic Services.

[26] Richard V. Cox, Barry G. Haskell, Yan Lecun, Behzad Shahraray and Lawrence Rabiner, "On the Application of Multimedia Processing to Communications," Proc. IEEE, Vol. 86, No. 5, May 1998, pp. 755-824.

[27] Tourdaj Ebrahimi aand Murat Kunt, "Visual Data Compression for Multimedia Applications," Proc. IEEE, Vol. 86, No. 6, June 1998, pp. 1109-1125.

[28] Ara Nefian and Monson H. Hayes III, "Hidden Markov Models For Face Recognition," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 1998), vol. V, pp. 2721-2724.

[29] Nicolas Tsapatsoulis, Nikolaos Doulamis, Anastasios Doulamis and Stefanos Kollias, "Face Extraction From Non-Uniform Background and Recognition in Compressed Domain," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP98), vol. V, pp. 2701-2704.

[30] Afshin David, "Real Time Methods for Face Recognition," M.Sc. Thesis, University of Ottawa, 1995.

[31] Shang-Hung Lin, Sun-Yuan Kung, Long-Ji Lin, "Face Recognition/Detection by Probabilistic Decision-Based Neural Network," IEEE Trans. On Neural Network, Vol. 8, No. 1, January 1997.

[32] The Database of Faces of AT&T Laboratories Cambridge, http://www.cam-orl.co.uk/facedatabase.htm.

[33] V. V. Kohir and U. B. Desai, "Face Recognition," IEEE Int'l Symp. Circuits and Systems (ISCAS 2000), May 28-31, 2000, Geneva, Switzerland, pp. V309-V312.

[34] H. Othman and T. Aboulnasr, "Low-Complexity 2-D Hidden Markov Model Face Recognition," IEEE Int'l Symp. Circuits and Systems (ISCAS 2000), May 28-31, 2000, Geneva, Switzerland, pp. V33-V36.

[35] Javad Haddadnia, Karim Faez And Majid Ahmadi, "A Neural Based Human Face Recognition System Using An Efficient Feature Extraction Method With Pseudo Zernike Moment," Journal Of Circuits, Systems And Computers, Vol. 11, No. 3 (2002), pp. 283-304.

[36] Esther Levin, and Roperto Pieraccini, "Dyanmic Planar Warping For Optical Character Recognition," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 1992).

[37] Touraj Assefi, "Stochastic Processes, Estimation Theory, and Image Enhancement," JPL Publication, June 1978.

[38] Krishna S. Nathan, Homayoon S. M. Beigi, Jayashree Subrahmonia, Gregory J. Clary and Hiroshi Maruyama, "Real-Time On-Line Unconstrained Handwriting Recognition Using Statistical Methods". Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 95), 1995, vol. 4, pp. 2619-2622.

[39] Krishna Nathan, Jayashree Subrahmonia and Michael P. Perrone, "Parameter Tying in Writer Dependent Recognition of On-Line Handwriting," Proc. Int'l Conf. Pattern Recognition (ICPR 96).

[40] Jayashree Subrahmonia, Krishna Nathan and Michael P. Perrone, "Writer Dependent Recognition Of On-Line Unconstrained Handwriting," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 96), 1996, vol. 6, pp. 3478-3481.

[41] Krishna Nathan, Andrew Senior and Jayashree Subrahmonia, "Initialization of Hidden Markov Models for Unconstrained On-line Handwriting Recognition," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 96), 1996, vol. 6, pp. 3502-3505.

[42] A.W.Senior and K.S.Nathan, "Writer Adaptation of an HMM handwriting recognition system," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 97), 1997, Vol. 2, pp.1447-1450.

[43] L. Gu, K. Rose, "Sub-State Tying In Tied Mixture Hidden Markov Models," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 2000), 2000, vol. 2, pp. 1013-1016

[44] Akinobu Lee, Tatsuya Kawahara, Kazuya Takeda and Kiyohiro Shikano, "A New Phonetic Tied Mixture Model For Efficient Decoding," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 2000), 2000, pp. 1269-1272.

[45] Krishna S. Nathan, Jerome R. Bellegarda, David Nahamoo, Eveline J. Bellegarda, "On –Line Handwriting Recognition Using Continuous Parameter Hidden Markov Model," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 93), vol. 5, pp.121-124.

[46] VassiliosV. Digalakis, Peter Monaco, and Hy Murveit, "Genones: Genralized Mixture Tying in Continuous Hidden Markov Model-based Speech Recognizers," IEEE Trans. Speech and Audio Processing, Vol. 4, No. 4, July 1996, pp. 281-289.

[47] Matthew S. Crouse, Robert D. Nowak and Richard G. Baraniuk, "Wavelet-Based Statistical Signal Processing Using Hidden Markov Models," IEEE Trans. Signal Processing, Vol. 46, No. 4, April 1998, pp. 886-902.

[48] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," IEEE Trans. Communications, vol. 28, Jan1980, pp.84-95.

[49] Stefan Müller, Stefan Eickeler, and Gerhard Rigoll, "Image Database Retrieval of Rotated Objects by User Sketch," IEEE Workshop Content-Based Access of Image and Video Libraries in conjunction with CVPR-98, pp. 40-44, Santa Barbara, USA, June 1998.

[50] Stefan Müller and Gerhard Rigoll, "Improved Stochastic Modeling of Shapes for Content-Based Image Retrieval," IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL-99), in conjunction with CVPR-99, pp. 23-27, Fort Collins, June 1999.

[51] Ara Nefian and Monson H. Hayes III, "Maximum Likelihood Training of The Embedded HMM For Face Detection and Recognition," Proc. Int'l Conf. Image Processing (ICIP 2000), vol. 1, pp. 33-36.

[52] Stefan Muller, Stefan Eickeler, Christoph Neukirchen, and Bernd Winterstein, "Segmentation and Classification of Hand-Drawn Pictograms in Cluttered Scenes-An Integrated Approach," Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 99), Vol. 6, pp. 3489-3492.