A New Approach to Probabilistic Image Modeling with Multidimensional Hidden Markov Models

Bernard Merialdo, Joakim Jiten, Eric Galmar, Benoit Huet

Multimedia Communications Department
Institut EURECOM
BP 193, 06904 Sophia-Antipolis, FRANCE
{merialdo,jiten,galmar,huet}@eurecom.fr

Abstract. This paper presents a novel multi-dimensional hidden Markov model approach to tackle the complex issue of image modeling. We propose a set of efficient algorithms that avoids the exponential complexity of regular multi-dimensional HMMs for the most frequent algorithms (Baum-Welch and Viterbi) due to the use of a random dependency tree (DT-HMM). We provide the theoretical basis for these algorithms, and we show that their complexity remains as small as in the uni-dimensional case. A number of possible applications are given to illustrate the genericity of the approach. Experimental results are also presented in order to demonstrate the potential of the proposed DT-HMM for common image analysis tasks such as object segmentation, and tracking.

1 Introduction

In image modeling there is a fairly wide-spread agreement that objects should be presented as collections of features which appear in a given mutual position or shape (e.g. sun in the sky, sky above landscape or boat in the water etc) [2,4]. This is also relevant on a lower level; consider analyzing local features in a small region; it is sometimes difficult even for a human to tell what the image is about.

The HMM considers observations (e.g. feature vectors representing pixels) statistically dependent on neighboring observations through transitions probabilities organized in a Markov mesh [1], giving a dependency in two dimensions. The state process defined by this mesh is a special case of the Markov random field.

Hidden Markov models (HMM) have earlier become a key technology for many applications such as speech recognition [8] and language modeling. Their success is largely due to the discovery of an efficient training algorithm, the Baum-Welch algorithm [10], which allows estimating the numeric values of the model parameters from training data. HMMs have been used in such diverse applications as acoustic modeling, language modeling, language analysis, spelling correction etc. Most of the current

applications involve uni-dimensional data. In theory, HMMs can be applied as well to multi-dimensional data. However, the complexity of the algorithms grows tremendously in higher dimensions, even in two dimensions, so that the usage of plain HMM becomes prohibitive in practice [18].

In this paper, we propose a new type of multi-dimensional hidden Markov model; the dependency-tree hidden Markov model (DT-HMM). We show that for this model, most of the common algorithms keep the same linear complexity as in one dimension. We explain these algorithms and illustrate the various possible usages of the DT-HMM through a set of examples. Our contribution is mostly theoretical, to show the richness and potential of this formalism. Further research will be needed to benchmark it with existing techniques. The remainder of this paper is organized as follows: section 3 provides the theoretical basis for DT-HMM, and sections 4-6 presents the different applications and experimental results. Finally in sections 7 we give the conclusions and suggest future work.

2 Related Work

Many approaches have been proposed to overcome the complexity of 2D-HMMs [11]. Among the first ones is [5] which uses a 1D HMM to model horizontal bands of face images. A more elaborate idea is to extract 1D features out of the image or video, and model these features with one or more 1D models [15]. Another approach is to use a two-level model, called Embedded HMM or Hierarchical HMM, where a first high level model contains super-states associated to a low level HMM, which models the lines of the observed image [16]. The main disadvantage of these approaches is that they greatly reduce the vertical dependencies between states, as it is only achieved through a single super-state. Finally several attempts have been done to heuristically reduce the complexity of the HMM algorithms by making simplifying assumptions which approximate the real algorithms:

- select a subset of state configurations only [17],
- ignore correlation of distant states [6],
- approximate probabilities by turbo-decoding [9].

The main disadvantage of these approaches is that they only provide approximate computations, so that the probabilistic model is no longer theoretically sound.

3 Dependency-Tree HMM

In this section, we briefly recall the basics of 2D HMM and describe our proposed DT-HMM [7].

3.1 2D-HMM

The reader is expected to be familiar with 1D-HMM. We denote by $O=\{o_{ij}, i=1,\ldots m, j=1,\ldots,n\}$ the observation, for example each o_{ij} may be the feature vector of a block

(i,j) in the image. We denote by S = {s_{ij}, i=1,…m, j=1,…,n} the state assignment of the HMM, where the HMM is assumed to be in state s_{ij} at position (i,j) and produce the observation vector o_{ij}. If we denote by λ the parameters of the HMM, then, under the Markov assumptions, the joint likelihood of O and S given λ can be computed as:

$$P(O,S|\lambda) = P(O|S,\lambda)P(S|\lambda) = \prod_{ij} p(o_{ij}|s_{ij},\lambda)p(s_{ij}|s_{i-1,j},s_{i,j-1},\lambda) \quad (1)$$

If the set of states of the HMM is {s_1, … s_N}, then the parameters λ are:

- the output probability distributions p(o | s_i)
- the transition probability distributions p(s_i | s_j,s_k).

Depending on the type of output (discrete or continuous) the output probability distribution are discrete or continuous (typically a mixture of Gaussian distribution).

We would like to point out that there are two ways of modeling the spatial dependences between the near neighbor state variables; by a causal or non-causal Markov random field (MRF). The former is referred to as Markov mesh and has the advantage that it reduces the complexity of likelihood functions for image classification [1]. The causality also enables the derivation of an analytic iterative algorithm to estimate states with the maximum a posteriori probability, due to that the total observation is progressively built from smaller parts. The state process of DT-HMM is defined by the Markov mesh.

3.2 DT-HMM

The problem with 2D-HMM is the double dependency of $s_{i,j}$ on its two neighbors, $s_{i-1,j}$ and $s_{i,j-1}$., which does not allow the factorization of computation as in 1D, and makes the computations practically intractable.

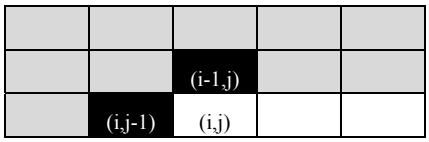

Fig. 1. 2D Neighbors

Our idea is to assume that $s_{i,j}$ depends on one neighbor at a time only. But this neighbor may be the horizontal or the vertical one, depending on a random variable t(i,j). More precisely, t(i,j) is a random variable with two possible values :

$$t(i,j) = \begin{cases} (i-1, j) & \text{with prob } 0.5 \\ (i, j-1) & \text{with prob } 0.5 \end{cases} \quad (2)$$

For the position on the first row or the first column, t(i,j) has only one value, the one which leads to a valid position inside the domain. t(0,0) is not defined.
So, our model assumes the following simplification:

$$p(s_{i,j} | s_{i-1,j}, s_{i,j-1}, t) = \begin{cases} p_V(s_{i,j} | s_{i-1,j}) & \text{if } t(i,j) = (i-1, j) \\ p_H(s_{i,j} | s_{i,j-1}) & \text{if } t(i,j) = (i, j-1) \end{cases} \quad (3)$$

If we further define a "direction" function:

$$D(t) = \begin{cases} V & \text{if } t = (i-1, j) \\ H & \text{if } t = (i, j-1) \end{cases} \quad (4)$$

then we have the simpler formulation:

$$p(s_{i,j} | s_{i-1,j}, s_{i,j-1}, t) = p_{D(t(i,j))}(s_{i,j} | s_{t(i,j)}) \quad (5)$$

Note that the vector **t** of the values t(i,j) for all (i,j) defines a tree structure over all positions, with (0,0) as the root. Figure 2 shows an example of random Dependency Tree.

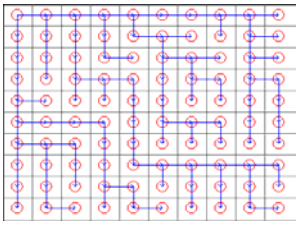

Fig. 2. Example of Random Dependency Tree.

The DT-HMM replaces the N^3 transition probabilities of the complete 2D-HMM by $2N^2$ transition probabilities. Therefore it is efficient in terms of storage. We will see that it is also efficient in terms of computation. Position (0,0) has no ancestor. In this paper, we assume for simplicity that the model starts with a predefined initial state s_I in position (0,0). It is straightforward to extend the algorithms to the case where the model starts with an initial probability distribution over all states.

3.3 Fundamental Problems

As stated in [8], three fundamental problems should be solved for using HMMs:

P1: Estimate the parameters of the model from a set of training examples,

P2: Estimate the probability of an observation to be produced by the model,
P3: Find the best state sequence in the emission of an observation.

A great advantage of HMM is that the same formalization can be used for a variety of tasks, many of which are relevant to Multimedia analysis:

- build a model from examples,
- classify an item,
- detect an object in a stream,
- analyze an object of known type, etc…

In the following sections, we will propose algorithms for the fundamental problems in the case of DT-HMM. We will show that these algorithms exhibit only moderate computation complexity, and we will provide illustrative examples of their usage in the context of Multimedia analysis.

4 Application to Image Segmentation

4.1 Viterbi Algorithm

The Viterbi algorithm is a solution for problem P3, it finds the most probable sequence of states which generates a given observation O:

$$\hat{Q} = \underset{Q}{\text{Argmax}}\ P(O,Q|t) \tag{6}$$

Let us define T(i,j) as the sub-tree with root (i,j), and define $\beta_{i,j}(s)$ as the maximum probability that the part of the observation covered by T(i,j) is generated starting from state s in position (i,j). We can compute the values of $\beta_{i,j}(s)$ recursively by enumerating the positions in the opposite of the raster order, in a backward manner:

- if (i,j) is a leaf in T(i,j):

$$\beta_{i,j}(s) = p(o_{i,j}|s) \tag{7}$$

- if (i,j) has only an horizontal successor:

$$\beta_{i,j}(s) = p(o_{i,j}|s) \max_{s'} p_H(s'|s)\beta_{i,j+1}(s') \tag{8}$$

- if (i,j) has only a vertical successor:

$$\beta_{i,j}(s) = p(o_{i,j}|s) \max_{s'} p_V(s'|s)\beta_{i+1,j}(s') \tag{9}$$

- if (i,j) has both an horizontal and a vertical successors:

$$\beta_{i,j}(s) = p(o_{i,j}|s)\left(\max_{s'} p_H(s'|s)\beta_{i,j+1}(s')\right)\left(\max_{s'} p_V(s'|s)\beta_{i+1,j}(s')\right) \quad (10)$$

Then the probability of the best state sequence for the whole image is $\beta_{0,0}(s_I)$. Note that this value may also serve as an approximation for solving problem P2.
The best state labeling is obtained by assigning $s_{0,0} = s_I$ and selecting recursively the neighbor states which accomplish the maxima in the previous formulas. Note that the complexity of the algorithm is only linear in the number of positions.

4.2 Relative Frequency estimation

Assume that we have a labeled observation, for example an image where each output block has been assigned a state of the model. (this labeled observation might have been created manually or automatically). Then, it is straightforward to estimate the transition probabilities by their relative frequency, for example:

$$p_H(s'|s,t) = \frac{N_{H,t}(s,s')}{N(s)} \quad (11)$$

where $N_{H,t}(s,s')$ is the number of times that state s' appears as a right horizontal neighbor of state s in the dependency tree t, and $N(s)$ the number of times that state s appears in the labeling. (This probability may be smoothed, for example using Lagrange smoothing).
The output probabilities may be also estimated by relative frequency in the discrete case, or using standard Multi-Gaussian estimation in the continuous case. This provides a solution for problem P1 in the case where a labeling is available, and is called Viterbi training. Each observation is assumed (with weight 1) to have resulted from the single most likely state sequence that might have caused it i.e. in the Viterbi training the state sequence with the maximum a posteriori probability P(S|O) is assumed to be the real state sequence.

4.3 Model Training

We now show the use of the previous two algorithms to train iteratively a DT-HMM on a set of images comprised of 130 consistent images depicting *beach* (see examples in Figure 3).

Fig. 3. Example of training images

During training each image is split into blocks of 16x16 pixels, and the observation vector for each block is computed as the average and variance of the LUV (CIE LUV color space) coding $\{L_\mu,U_\mu,V_\mu, L_\sigma,U_\sigma,V_\sigma\}$ combined with six quantified DCT coefficients (Discrete Cosine Transform). Thus each block is represented by a 12 dimensional vector. Every feature vector is annotated with a sub-class as described below.

4.4 States with semantic labels

In order to perform semantic segmentation we enforce semantic meaning to the states by uniquely assigning a state to one sub-class. Table 1 lists the number of states assigned to each sub-class.

Table 1. The number of states for each sub-class

Sub Class	**No. states**
Unknown	3
Sky	7
Sea	5
Sand	6
Mountain	3
Vegetation	3
Person	4
Building	3
Boat	2
8 sub-classes	36 states

Annotations was done in practice by first segmenting the training images into arbitrary shaped regions using the algorithm proposed in [20] and then manually label each region with one of the sub classes by using an application with a graphical user interface as shown in Figure. 4.

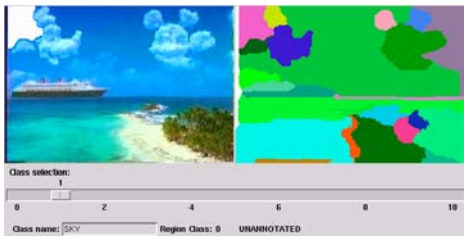

Fig. 4. Annotating an image segment as “sky”

To make an initial model for the output probabilities a GMM (Gaussian Mixture Model) is trained with observations corresponding to each sub-class. We dedicate three GMM components for every state, which gives us for instance that "sky" has 21 components and vegetation has 9 (see Table 1). Then we group the components into clusters by k-means. The number of clusters corresponds to the number of states we have dedicated to the actual sub-class. Finally each state is assigned to a cluster, which we have scaled up by a factor of two (multiplying its component weight by 2). The transition probabilities are initialized uniformly. Then, during training we iterate the following steps:

- With (11) estimate the output and transition probabilities by counting the relative frequencies (emission of an observation by a state, horizontal and vertical successors of a state) with Lagrange smoothing.
- Generate a random dependency tree and perform a Viterbi alignment to generate a new labeling of the image. The Viterbi training procedure is adapted to select the range of states that correspond to the annotated sub-class at each position, thus constraining the possible states for the observations.

4.5 Experiment Results

During training, we can observe the state assignments at each iteration as an indication of how the model fits the training data. For example, the first ten iterations on the training image to the left in figure 4 above provide the following assignments:

Fig. 5. State segmentation after 0, 2, 6 and 10 iterations

The sequence in figure 5 shows that the model has rapidly adapted each sub class to a particular set of observations. As such, the Viterbi labeling provides a relevant segmentation of the image. The graph below shows the evolution of likelihood of the training data during the training iterations. We can see that the likelihood for the model given the data has an asymptotic behavior after 10 iterations.

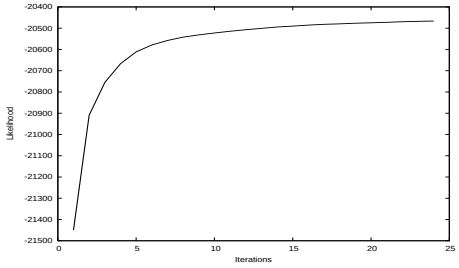

Fig. 6. Likelihood of the training data

Once the model is trained, we can apply it on new images. Below is an example of the state assignment for an image in the test set after 15 iterations, 70% of the blocks is correctly classified.

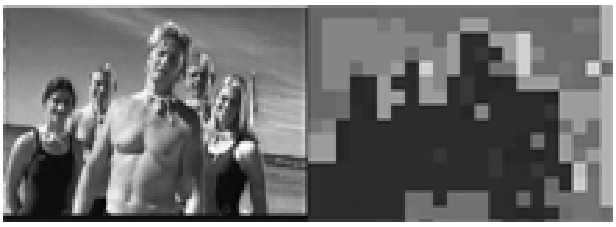

Fig. 7. State segmentation on test image

It should be emphasized that this is not just a simple segmentation of the images, but that each region is also assigned one of the 36 states (which represents one of the sub classes) of the model, and that the definition of those states has been done taking into account all training data simultaneously. We can observe that those area types are labeled with the same states during training.

5 Application to Object Tracking

In this section, we present how DT-HMMs can be applied to track an object in a video sequence. We consider a model with two types of states, object states (s^o) and background states (s^b). The general idea is to train the model on an initial image where the object has been delimited, then to use the Viterbi algorithm to find the location of the object in subsequent frames.

We use a model with 6 states:

- background states: $s^b=\{ s_1, s_6\}$,
- object states: $s^o=\{s_2, s_3, s_4, s_5 \}$.

Assuming that a bounding box has been drawn around the object on the initial frame, we set the states on the initial frame according to the pattern in figure 8.

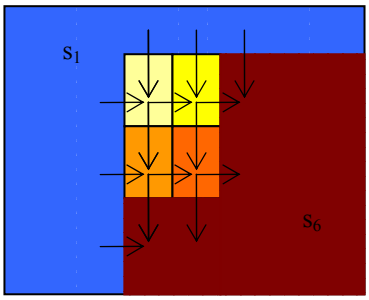

Fig. 8. Object and Background states for Object Tracking

- s_1 is the initial state of the model,
- s_6 is the final state of the model,
- The 4 object states are arranged inside a regular 2x2 grid within the bounding box.

In order to reinforce spatial constraints, we do not smooth transition probabilities, so that transitions which do not exist in the initial frame will keep a probability of zero and will remain forbidden in the subsequent frames. The output probabilities are smoothed as the color of the object may change from frame to frame.
We compared several variations of the tracking procedure:

- (b) train the model on the first frame, and use it to Viterbi align the other frames,
- (c) train a first model on the first frame, use it to Viterbi align the second frame, train a second model on this alignment, use it to Viterbi align the third frame, etc…
- (d) same as before, but train on all frames since the beginning, rather than just the current frame.

Figure 9 shows the compared results of these procedures. We can observe that, because the initial bounding box also contains background pixels, all methods have a tendency to spread outside the actual shape of the object. This is especially true for method (c), which updates the model at every frame. Method (d) improves a little, but it still not perfect.

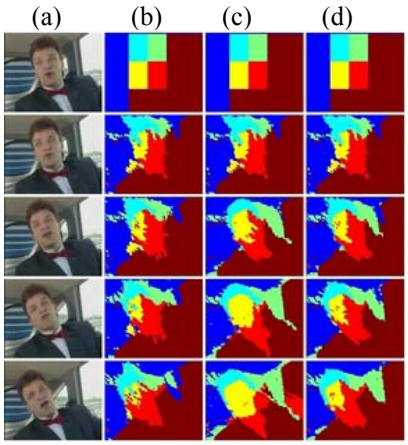

Fig. 9. Alignments at frames 70, 71, 75, 80 and 85

In order to cope with this problem, we should try to penalize background pixels which are within the object bounding box. For this purpose, we propose to modify the output probabilities of the object states with the following formula:

$$P'(o|s) \propto w_{os} * P(o|s) \tag{12}$$

$$w_{os} = \frac{E(s \to o)}{E(s \to o) + \sum_{s' \in s^b} E(s' \to o)}$$

This formula will reduce the output probability of colors which are highly probable in the background states, therefore enhancing the true object pixels only.

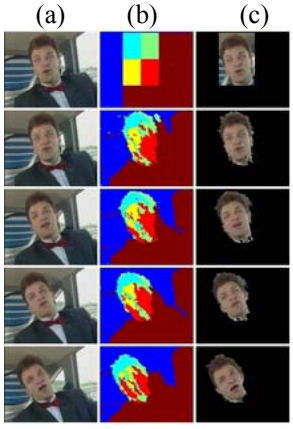

Fig. 10. Object tracking with weighted output probabilities

Figure 10 shows the result of tracking the object with method (c) using object probability weighting. It is clear that probability weighting has greatly improved the quality of object tracking.

6 Future Extensions

The DT-HMM formalism is open to a great variety of extensions and tracks. For example the algorithms that we have proposed remain valid for other ancestor functions and multidimensional Markov models.

6.1 Ancestor Dependencies

As before mentioned the state process is based on the dependencies defined by the Markov mesh, which is a special case of the Markov random field [1]. The Markov mesh defines spatial dependencies that are called "causal" because the dependent states are "past"; above and to the left of the current node. We can for example consider the following causal dependencies of a 3d and 4th order Markov mesh:

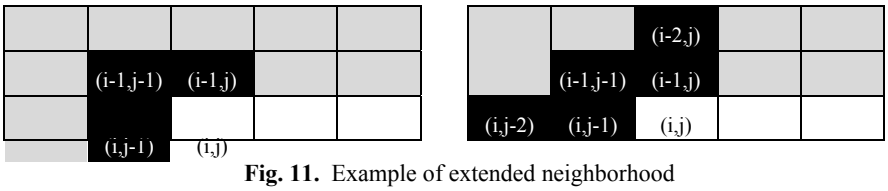

Fig. 11. Example of extended neighborhood

This only increases linearly (not exponentially) the number of transition probabilities, and therefore the complexity of the model and the algorithm.

6.2 Trees and Duals

One may notice that the value of P(O| t) depends on the specific dependency tree t that has been used in the computation. It may happen that, by chance, a given image get a more or less convenient dependency tree, and therefore a different score. The true score for an image should be:

$$P(O) = \sum_{t} P(O|\, t)P(t) \qquad (13)$$

All dependency trees are supposed to be equally likely, so that P(t) is uniform. While this sum cannot be computed easily, it may be estimated by generating several trees and averaging the conditional likelihood of the output. Of particular interest is the estimation:

$$P(O) \approx \frac{1}{2}\left(P(O|t) + P(O|\bar{t})\right) \quad (14)$$

where $\bar{t}$ is the dual tree of t, defined by replacing horizontal by vertical dependencies (and vice versa), except for boundary constraints. This formulation introduces both horizontal and vertical dependencies for all neighbor pairs in the observation. In an investigation of different estimations for (13), we demonstrated that the dual approximation is more accurate than sampling with a unique random tree [19].

6.3 Multidimensional Model

The framework can also be extended to several dimensions, for example in the case of a video. Video can be regarded as images indexed with time. Considering the continuity of consecutive frames, it is often reasonable to assume local dependencies between pixels among frames. If a position is (i,j,t), it could depend on the neighbors (i-1,j,t), (i,j-1,t), (i,j,t-1) or more.

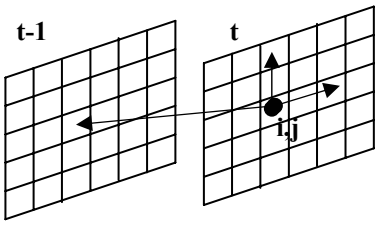

Fig. 12. Extending the block dependences to three-dimensions

7 Conclusion

In this paper, we have presented a new type of hidden Markov models based on dependency trees. We have shown that this presentation leads to very efficient algorithms for 2D observations, and we have presented two examples to show the richness of the potential application of these models.

Our research on these models is only beginning, so that the results should not be compared now with the results from more advanced techniques. More time is needed to understand how to exploit best DT-HMM. In particular, it seems obvious that efficient models should contain a large number of states (in speech, acoustic models often have several hundred states), but these states have to be constructed in a coherent manner which has yet to be defined.

Our contribution is mostly theoretical. Examples have been used to show that the DT-HMM has a great potential for applications. We have identified several issues which are potential tracks for future research. We plan to explore the properties of this model further in the future, and are confident that this type of model will be helpful for a large panel of applications.

8 Acknowledgments

The research leading to this paper was supported by the Institut Eurecom and by the European Commission under contract FP6-027026, Knowledge Space of semantic inference for automatic annotation and retrieval of multimedia content - K-Space.

References

1. Kanal, L.N.: Markov mesh models in Image Modeling. New York: Academic, 1980, pp. 239-243
2. Lim, J.H., Jin, J.S.: Semantics Discovery for Image Indexing. ECCV (1) 2004: 270-281
3. Piriou, G., Bouthemy, P., Jian-Feng Yao: Extraction of Semantic Dynamic Content from Videos with Probabilistic Motion Models. ECCV (3) 2004: 145-157
4. Moreels, P., Maire M., Perona, P.: Recognition by Probabilistic Hypothesis Construction. ECCV (1) 2004: 55-68
5. Ferdinando, S., Fallside, F.: Face Identification and Feature Extraction Using Hidden Markov Models, Image Processing: Theory and Applications, Elsevier, 1993, pp 295-298
6. Merialdo, B., Marchand-Maillet, S.; Huet, B.: Approximate Viterbi decoding for 2D-hidden Markov models, IEEE International Conference on , Acoustics, Speech, and Signal Processing, Volume 6, 5-9 June 2000 Page(s):2147 - 2150 vol.4
7. Merialdo, B: Dependency Tree Hidden Markov Models, Research Report RR-05-128, Institut Eurecom, Jan 2005
8. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition Proceedings of the IEEE, Volume 77, Issue 2, Feb. 1989 Page(s):257 – 286
9. Perronnin, F., Dugelay, J.-L.; Rose, K.: Deformable face mapping for person identification, International Conference on Image Processing, Volume 1, 14-17 Sept. 2003
10. Baum, LE.,Petrie T.: Statistical Inference for Probabilistic Functions of Finite State Markov Chains, Annual Math., Stat., 1966, Vol.37, pp. 1554-1563
11. Mohamed, M. A., Gader P.: Generalized Hidden Markov Models-Part I: Theoretical Frameworks, IEEE Transaction on Fuzzy Systems, February, 2000, Vol.8, No.1, pp. 67-81
12. Baker, J.K.:Trainable grammars for speech recognition. In Jared J.Wolf and Dennis H. Klatt, editors, Speech communication papers presented at the 97th Meeting of the Acoustical Society of America, MIT, Cambridge, MA, June 1979
13. Jelinek F., Lafferty, F. and R. L. Mercer: Basic methods of probabilistic context free grammars Technical Report RC 16374 (72684), IBM, Yorktown Heights, New York 10598. 1990
14. Jelinek, F.: Statistical Methods for Speech Recognition Cambridge, MA: MIT Press, 1997.
15. Brand, M., Oliver, N. and Pentland, A.:. Coupled hidden Markov models for complex action recognition. In Proceedings, CVPR, pages 994--999. IEEE Press, 1997
16. Fine S., Singer Y., Tishby, N.: The hierarchical hidden Markov model: Analysis and applications," Machine Learning 32(1998)
17. Li, J., Najmi, A. and Gray, R. M.: Image classification by a two-dimensional hidden markov model, IEEE Trans. Signal Processing, vol. 48, no. 2, pp. 517–533, 2000
18. Levin, E.; Pieraccini, R.: Dynamic planar warping for optical character recognition, IEEE International Conference on Acoustics, Speech, and Signal Processing, , Volume 3, 23-26 March 1992 Page(s):149 – 152
19. Jiten, J., Merialdo, B.: Probabilistic Image Modeling With Dependency-Tree Hidden Markov Models, WIAMIS 2006 : 19-21 April 2006, 7th International Workshop on Image Analysis for Multimedia Interactive Services, Hyatt Regency, Korea

20. P. F. Felzenszwalb , D. P. Huttenlocher, Image Segmentation Using Local Variation, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, p.98, June 23-25, 1998