

Multidimensional Hidden Markov Model Applied to Image and Video Analysis

Joakim Jitén Söderberg

▶ **To cite this version:**

Joakim Jitén Söderberg. Multidimensional Hidden Markov Model Applied to Image and Video Analysis. domain_other. Télécom ParisTech, 2007. English. NNT : . pastel-00002454

HAL Id: pastel-00002454
https://pastel.hal.science/pastel-00002454

Submitted on 25 Jun 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Multidimensional Hidden Markov Model
Applied to Image and Video Analysis

by

Joakim Jitén Söderberg

A Thesis Submitted to the Graduate Faculty of

Ecole Nationale Supérieure des Télécommunications Paris

in Partial Fulfillment of the

Requirements for the degree of

DOCTOR OF PHILOSOPHY

Major Subject: Signal and Images

Approved by the
Examining Committee:

_____
Bernard Merialdo, Thesis Adviser

_____
Georges Quenot, Opponent

_____
Pierre Courtellemont, Opponent

_____
Gaël Richard, Member

_____
Gerhard Rigoll, Member

_____
Philippe Joly, Member

Institute Eurecom
Sophia Antipolis
February, 2007

© Copyright 2007
by
Institute Eurecom
All Rights Reserved

Author's address:
Joakim Jitén
joakim.jiten@eurecom.fr
B.P. 193
06904 Sophia-Antipolis
France

ABSTRACT

Recent progress and prospects in cognitive vision, multimedia, human-computer interaction, communications and the Web call for, and can profit from applications of advanced image and video analysis. Adaptive robust systems are required for analysis, indexing and summarization of large amounts of audio-visual data.

Image classification is perhaps the most important part of digital image analysis. The objective is to identify and portray the visual features occurring in an image in terms of differentiated classes or themes. Applications can be found in a wide range of domains such as medical image understanding, surveillance applications, remote sensing and interactive TV.

Traditional image classification methods analyses independent blocks of an image, which results in a context-free formalism. However there is a fairly wide-spread agreement that observations 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.) [20], [21]. Consider analyzing local features in a small region of an image; it is sometimes difficult even for a human to tell what the image is about.

In this dissertation we apply a statistical machine learning approach to model context in sequential data. With a statistical model in hand, we can perform several important tasks to image analysis such as; estimation, classification and segmentation.

We employ a new efficient algorithm that models images by a two dimensional hidden Markov model (HMM). The HMM considers observations statistically dependent on neighboring observations through transition probabilities organized in a Markov mesh, giving a dependency in two dimensions. The main difficulty with applying a 2-D HMM to images is the computational complexity which grows exponentially with the number of image blocks.

The main technical contribution of this thesis is a way of estimating the parameters of a 2-D HMM in $O(whN^2)$ complexity instead of $O(wN^{2h})$, where N is the number of states in the model and (w,h) is the width respectively height of the image.

We investigate the performance of our proposed model (DT HMM), and search for its point of operation. Application to classification of TV broadcast frames reveal intrinsic weaknesses of the HMMs for which we propose remedies.

In an effort to introduce both global and local context in images, the DT HMM was extended to model multiple image resolutions. The results indicate that the earlier recorded deficiency can be conquered and that its performance can be compared with other known algorithms.

Finally we illustrate that the DT HMM formalism is open to a great variety of extensions and tracks. Since 3-D HMMs has been little studied we investigate the extension of the framework to three dimensions. We consider the case of video data, where the two dimensions are spatial, while the third dimension is temporal. To investigate the impact of the time-dimension dependency we explore the ability of the model to track objects that cross each other or pass behind another static object.

RÉSUMÉ

Les progrès récents dans les domaines de la vision cognitive, du multimédia, de l'interaction homme-machine, des communications et de l'Internet sont un apport considérable pour la recherche qui profite à l'imagerie et l'analyse vidéo. Des systèmes adaptés et fiables sont nécessaires pour l'analyse, l'indexation et le résumé de grandes quantités de données audiovisuelles.

La classification d'images constitue sans doute, la partie la plus importante de l'analyse de l'image numérique. L'objectif est d'identifier et de décrire les caractéristiques présentes dans une image afin de les répertorier par classes et par thèmes. Des applications existent dans un grand nombre de domaines, tels que l'interprétation de l'imagerie médicale, la surveillance, la photo satellite et la télévision interactive.

Les méthodes traditionnelles de classification d'images procèdent par analyse des blocs distincts d'une image, ce qui aboutit à un formalisme non contextuel des caractéristiques visuelles. Toutefois, face à l'analyse d'une parcelle d'image, l'œil humain est souvent dans l'incapacité d'identifier ce qu'il voit. Les approches récentes tendent donc de plus en plus vers une vision globale de l'image incluant sa structure et sa forme générale (ex: le soleil dans le ciel, le ciel au dessus d'un paysage ou encore un bateau sur l'eau, etc.) [20], [21].

HMM Multi dimensionnel

Cette thèse est une approche statistique issue de l'intelligence artificielle visant à une représentation séquentielle des données de l'image. Cette représentation statistique permet l'estimation, la classification, et la segmentation de l'image.

Nous utilisons un nouvel algorithme efficace représentant les images à l'aide du modèle bidimensionnel de Markov caché (HMM). Le HMM considère les observations statistiquement dépendantes d'observations voisines à travers des probabilités de transition organisées dans les mailles de Markov, ce qui induit une dépendance en deux dimensions. La principale difficulté à appliquer un 2-D HMM aux images est la

complexité algorithmique qui s'accroît de façon exponentielle avec le nombre de segments d'image.

En conséquence, la contribution technique majeure de cette thèse est d'estimer les paramètres d'un nouveau type de 2-D HMM dont la complexité sera $O(LlN^2)$ au lieu de $O(LN^{2l})$ où N est le nombre d'états dans le modèle et (L,l) sont respectivement la largeur et la longueur de l'image.

Modèle d'entraînement

L'algorithme EM est généralement employé pour trouver l'estimation des paramètres du modèle de Markov caché la plus probable d'après les vecteurs caractéristiques observés. Cet algorithme est aussi connu sous le nom d'algorithme de Baum-Welch. Nous décrivons l'ensemble complet des paramètres pour un modèle donné par $\lambda = (a_{ij}, b_s(o_t), \pi_s)$. Comme montré en [37], trois problèmes fondamentaux doivent être résolus pour l'utilisation des HMMs.

Problème 1: Estimer $P(O|\lambda)$, La probabilité de la séquence d'observation selon les paramètres du modèle

Problème 2: Trouver la séquence d'états $S = \{s_1,\ldots,s_T\}$ la plus significative.

Problème 3: Comment ajuster les paramètres du modèle λ^* pour maximiser $P(O|\lambda)$?

Comme nous verrons dans la section 3.1, il y a des méthodes établies pour travailler sur ces problèmes. Ce sont respectivement les algorithmes « forward-backward », de Viterbi, et de Baum-Welch. Dans la section suivante, Je montrerai que ces algorithmes peuvent être adaptés au nouveau modèle proposé.

Extensions 2-D nécessaires pour la classification d'images

Nous divisons une image en une grille régulière de blocs. Un bloc est désigné par sa position (i,j), et l'ensemble complet des blocs est $\aleph = \{ (i,j): 0 \le i < L, 0 \le j < l \}$ où L et l sont respectivement la largeur et la hauteur de l'image. Un vecteur caractéristique o_{ij} est calculé pour chaque bloc (i,j) et l'ensemble de vecteurs caractéristiques $O = \{o_{ij} : (i,j) \in \aleph \}$ décrivant l'image entière est appelée « *champ de vecteurs* ». D'après les hypothèses des 2-D HMM, ce champ de vecteurs est généré par les états du modèle. L'image est donc classée en accord avec ses vecteurs caractéristiques.

Un 2-D HMM est une grille de noeuds, chacun correspondant à un bloc. A Chaque nœud peut être affecté un des N états possibles {1,2,…,N}. L'état d'un bloc (i,j) est noté s_{ij}. La Figure 1 illustre les blocs d'une image ainsi que les nœuds correspondants.

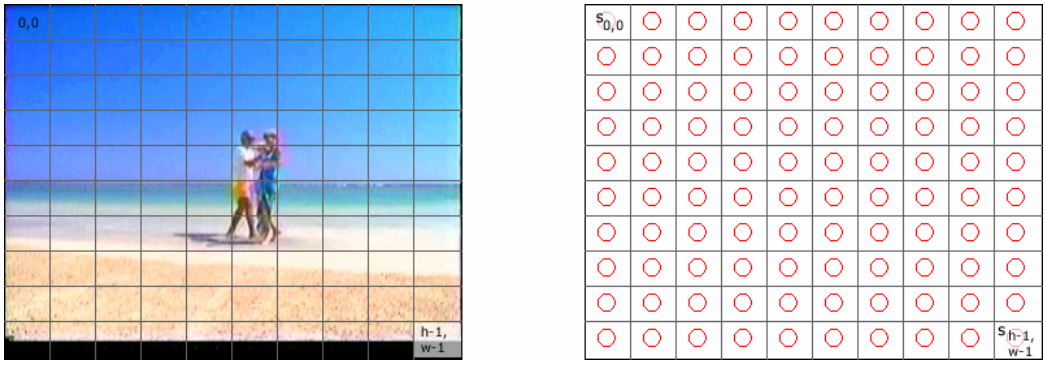


Figure 1. (a) Décomposition de l'image en blocs, (b) Etats du modèle de Markov

La formalisation du 2-D HMM est basée sur deux hypothèses. La première est :

$$P(s_{i,j} \mid s_{i',j'}, o_{i',j'} : (i',j') \in \Psi) = a_{m,n,l} \qquad (1.1)$$

$$\text{where } \Psi = \{(i',j') : (i'\,j') < (i,j)\}$$

$$\text{and } m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}$$

Cela signifie que le processus d'états est Markovien du premier ordre: La probabilité que le système entre dans un état particulier à la position (i,j) dépend uniquement des états des observations adjacentes selon les directions horizontales (i-1,j) et verticales (i,j-1). La deuxième hypothèse stipule que le vecteur caractéristique est seulement dépendant de l'état à la position (i,j), i.e. l'observation est conditionnellement indépendante des autres blocs.
Comme précédemment, nous noterons par λ les paramètres du HMM, donc selon les hypothèses de Markov, la probabilité conjointe de O et S selon λ peut être calculée d'après :

$$\begin{aligned} P(O,S|\lambda) &= P(O|S,\lambda)P(S|\lambda) \\ &= \prod_{ij} P\left(o_{ij}|s_{ij},\lambda\right)P\left(s_{ij}|s_{i-1,j},s_{i,j-1},\lambda\right) \end{aligned} \qquad (1.2)$$

Notons que la probabilité conditionnelle $P(s_{i,j}|s_{i,j-a},s_{i-1,j},\lambda)$ se réduit à $P(s_{1,j}|s_{1,j-a},\lambda)$ lorsque i=1, à $P(s_{i,1}|s_{i-1,1},\lambda)$ quand j=1, et à $P(s_{1,1}|\,\lambda)$ si i=j=1.

Dans la section suivante, nous présenterons une méthode efficace pour calculer $P(O,S|\lambda)$ basée sur l'idée d'un arbre de dépendances aléatoires.

Arbre de dépendance

Comme mentionné précédemment, le problème avec les 2-D HMM est la double dépendance entre $s_{i,j}$ et ses deux voisins $s_{i-1,j}$ et $s_{i,j-1}$, qui n'autorise pas la factorisation des composantes comme en 1-D, et rend les calculs intraitables.

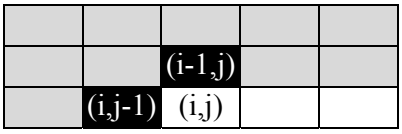


Figure 2. Voisins 2-D

Notre idée suppose que $s_{i,j}$ dépend uniquement d'un voisin à la fois. Ce voisin peut être celui à son horizontale ou à sa verticale, selon une variable aléatoire t(i,j). Plus précisément, t(i,j) est susceptible de prendre deux valeurs distinctes:

$$t(i,j) = \begin{cases} (i-1,j) & \text{with prob } 0.5 \\ (i,j-1) & \text{with prob } 0.5 \end{cases} \tag{1.3}$$

Pour la position sur la première ligne ou la première colonne, t(i,j) a seulement une seule valeur, celle qui conduit à une position valide dans le domaine. t0,0) n'est pas défini. Ce qui induit les simplifications suivantes pour notre modèle :

$$p(s_{i,j} \mid s_{i-1,j}, s_{i,j-1}, t) = \begin{cases} p_V(s_{i,j} \mid s_{i-1,j}) & \text{if } t(i,j) = (i-1,j) \\ p_H(s_{i,j} \mid s_{i,j-1}) & \text{if } t(i,j) = (i,j-1) \end{cases} \tag{1.4}$$

Si nous définissons ensuite une fonction de direction :

$$D(t) = \begin{cases} V & \text{if } t = (i-1,j) \\ H & \text{if } t = (i,j-1) \end{cases} \tag{1.5}$$

Nous obtenons alors la formulation simplifiée :

$$P(s_{i,j} \mid s_{i-1,j}, s_{i,j-1}, t) = P_{D(t(i,j))}(s_{i,j} \mid s_{t(i,j)}) \tag{1.6}$$

Notons que le vecteur **t** des valeurs t(i,j), pour tout (i,j) défini une structure d'arbre sur l'ensemble des positions, admettant (0,0) comme racine. La Figure 3 montre un exemple d'arbre de dépendances aléatoires

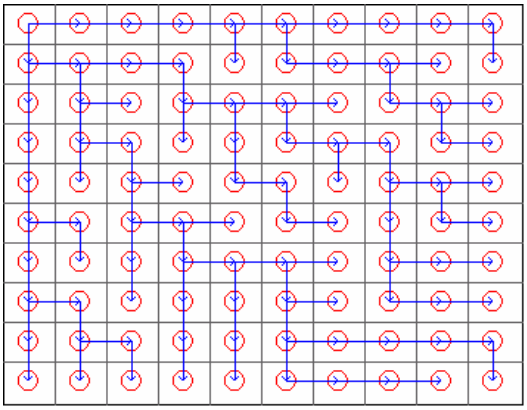


Figure 3. Exemple d'arbre de dépendances aléatoires.

Avec cette structure d'arbre, nous pouvons calculer la probabilité d'une observation produite par le modèle quelque soit la séquence d'état (tant que les états sont inconnus).

$$\boldsymbol{P}(\boldsymbol{o}) = \sum_{\boldsymbol{s}} \boldsymbol{P}(\boldsymbol{o}, \boldsymbol{s} \mid \boldsymbol{t}) \tag{1.7}$$

Et la séquence d'état la plus probable s^* qui génère ce résultat :

$$\arg\max_{\boldsymbol{s}} \boldsymbol{P}(\boldsymbol{o}, \boldsymbol{s} \mid \boldsymbol{t}) \tag{1.8}$$

Nous allons maintenant montrer comment résoudre un des trois problèmes fondamentaux cités en 2.4.9, ce qui nous permettra de mesurer le score d'une observation selon le modèle λ.

Solution au Problème 1, estimer P(O|λ)

Nous voulons calculer la probabilité de l'observation en fonction des paramètres P(O|λ) (2.1). Nous définissons la *probabilité intérieure* $\beta_{ij}(s)$ comme la probabilité que la partie de l'image couverte par le sous arbre T(i,j) de racine (i,j) soit produit par la séquence d'observation partielle et se termine à la position (i,j) (voir la portion ombrée sur la Figure 4).

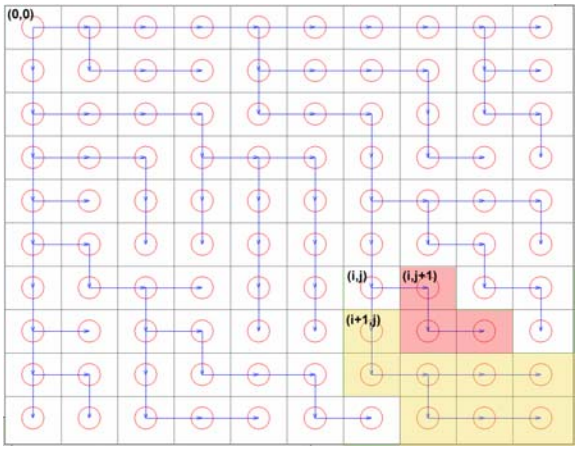


Figure 4. Les probabilités intérieures

Ces valeurs peuvent être calculées récursivement, dans l'ordre inverse (en partant de la dernière position) de leurs relations :

- Si (i,j) est une feuille de t(i,j) :

$$\beta_{i,j}(s) = p(o_{i,j}|s) \tag{1.9}$$

- Si (i,j) admet seulement un successeur horizontal :

$$\beta_{i,j}(s) = p(o_{i,j}|s)\sum_{s'} p_H(s'|s)\beta_{i,j+1}(s') \tag{1.10}$$

- Si (i,j) a seulement un successeur vertical :

$$\beta_{i,j}(s) = p(o_{i,j}|s)\sum_{s'} p_V(s'|s)\beta_{i+1,j}(s') \tag{1.11}$$

- Si (i,j) a deux successeurs, l'un horizontal et l'autre vertical :

$$\beta_{i,j}(s) = p(o_{i,j}|s)\left(\sum_{s'} p_H(s'|s)\beta_{i,j+1}(s')\right)\left(\sum_{s'} p_V(s'|s)\beta_{i+1,j}(s')\right) \tag{1.12}$$

La probabilité que l'image soit produite par le modèle est donc:

$$P(O \mid t) = \beta_{0,0}(s_i) \tag{1.13}$$

ce qui nous donne la solution au problème. La problématique 1 est applicable dans le domaine de la classification où l'on veut choisir un modèle qui satisfait le mieux une observation.

Expériences

Dans notre première expérience, nous utilisons l'arbre de dépendance HMM en temps que structure pour un classificateur d'images contextuel. Nous explorons aussi comment l'équilibre entre l'information structurelle et le contenu descriptif affectent la précision et le rappel en variant la taille des blocs.
Les algorithmes Baum-Welch modifiés (voir la section 3.4) ont été utilisés pour évaluer les paramètres des modèles dans la phase expérimentale. Pour classifier une image, ses descripteurs de bas niveau sont extraits et ensuite P (O | λ) est calculé pour chaque modèle évaluant le niveau de correspondance entre le modèle et l'observation, puis on recherche ensuite le modèle fournissant la plus haute probabilité a posteriori. On montre une illustration générale du système de classification dans la Figure 5.

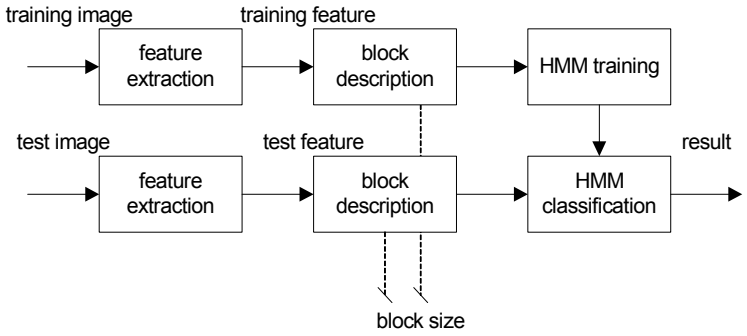


Figure 5. Schéma de Catégorisation d'Image.

Le graphique ci-dessous montre la précision de classification moyenne pour sept tailles de bloc différentes. Nous avons remarqué qu'une taille de bloc de 16x15 pixels (modèle #4) donne la précision moyenne la plus haute 0.036.

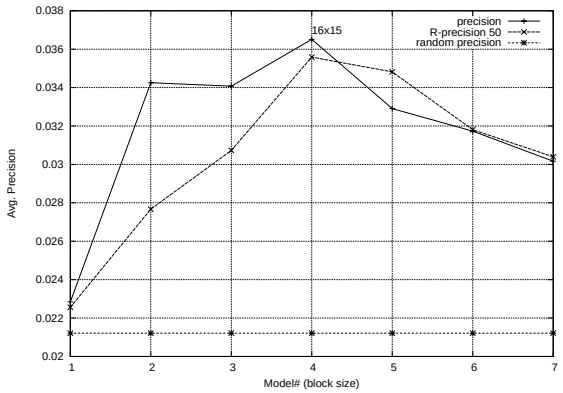


Figure 6. La précision moyenne pour tailles de bloc: (1) 176x120, (2) 88x60, (3) 44x40, (4) 16x15, (5) 8x8, (6) 4x4, (7) 2x2

Nous pouvons aussi voir que la performance diminue rapidement avec de très grands blocs. L'explication est que les moyennes des canaux de couleur ne sont pas assez descriptives et que les coefficients DCT refléteront seulement des hautes variations de fréquence puisque l'échelle sera plus haute quand la taille de bloc est augmentée.

Les résultats ne sont pas comparables avec les taux typiquement observés dans les expériences vidéo TREC [66], cependant ils nous mènent à une compréhension plus grande du modèle et ont inspiré de nouvelles expériences et raffinements qui seront discutés dans les chapitres suivants.

Nous avons noté un inconvénient connu du HMM: la probabilité de production joue un rôle plus important que la probabilité de transition. La sortie de distribution s'étend sur la plus grande dispersion que sur la probabilité de transition (cette dernière s'étend sur 16 états seulement), avec une majorité de transitions d'un état à l'autre. Cela explique pourquoi une image qui a une couleur presque uniforme a une haute probabilité d'émission.

Influence de l'Arbre de Dépendance

Le DT HMM est rendu moins complexe que le 2-D HMM, en changeant les dépendances spatiales horizontales et verticales doubles en une dépendance unidirectionnelle aléatoire, horizontale ou verticale. La question qui se pose est: quel l'impact ce choix aléatoire a-t-il sur le modèle? Nous explorons donc les différents aboutissements de l'effet de l'arbre aléatoire.

Selon le modèle, la probabilité exacte d'une observation est :

$$P(O) = \sum_{t} P(O|t)P(t) \qquad (1.14)$$

Nous pouvons *postuler* qu'il faut une équivalence de tous les arbres de dépendance, pour que la distribution P(t) soit uniforme. Étant donné qu'il y a $2^{(m-1)(n-1)}$ arbres différents pour une image de blocs m x n, le calcul complet est prohibitif. Donc, il est important de chercher des approximations de cette valeur qui sont faciles à calculer. À cette fin nous examinons trois façons différentes de faire cette évaluation par: prélèvement d'échantillon unique (P^u), moyenne d'arbre (P^a) et arbre dual (P^d). La Section 3.8 donne une étude détaillée de leurs qualités.

Segmentation sémantique d'images

Pour mieux évaluer les possibilités du modèle, nous avons introduit un champ *interprétation* aux états du DT HMM en associant une sous-classe pour partitionner les états. Affecter plusieurs états par sous-classe donne au modèle la flexibilité suffisante pour s'adapter aux sous–classes ayant des observations visuelles variées. En entraînant un nouveau modèle avec des états restreints, nous pouvons effectuer une segmentation sémantique de l'image sur des données non connues pourvu qu'elles appartiennent à une des classes.

Etats pourvus d'un label sémantique

A chaque sous-classe est assignée un ensemble d'états pour permettre une représentation flexible. Supposons qu'il y ait K sous-classes (1,…,K) et qu'un vecteur d'observation o_{ij} appartienne à une région annotée avec une sous-classe c_k. Alors, son ensemble d'états permis est {s(k)}. La table ci-dessous liste les différentes sous-classes et leur nombre d'états alloués.

Table1. Nombre d'etats pour chaque classe.

Sous Classe	No. d'état
Divers	3
Ciel	7
Mer	5
Sable	6
Montagne	3
Végétation	3
Personne	4
Batiment	3
Bateau	2
9 sous-classes	36 états

Nous utilisons une version modifiée de l'algorithme de Viterbi capable de gérer la situation lorsqu'une sous-classe visuelle est représentée par plusieurs états, et que seules les annotations des sous-classes sont disponibles. Nous avons étudié plusieurs propriétés de ce procédé.

L'entraînement fut effectué sur les archives de TRECVid [66], duquel nous avons tiré une collection hétérogène de 130 images dépeignant « plage » (voir Figure dessous).



Figure 7. Examples d'Images entraînement.

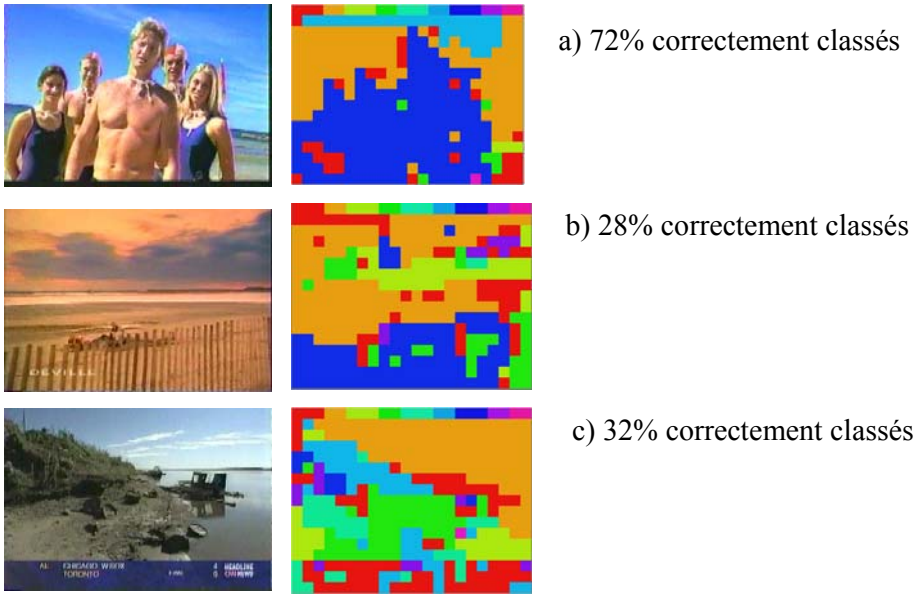
L'expérience fut conduite sur 40 images tests sémantiquement segmentées. Nous avons comparé la meilleure affectation d'état obtenue par l'algorithme de Viterbi (Cela prend en compte à la fois les probabilités de sortie et de transition) avec l'affectation où chaque vecteur caractéristique est associé à l'état de meilleure probabilité de sortie. Le taux moyen de blocs correctement labellisés est de 38% en prenant en compte les probabilités de transition, et de 32% pour les probabilités de sortie seules.

La table de confusion ci-dessous montre le nombre de blocs classés pour chaque classe. On constate que *Ciel* est parfois confondu avec *sable* (à cause des réflexions, comme dans Figure 43 b). De même, on note l'amalgame occasionnel de *mer* avec sable (à cause de leur recouvrement), et de *montagne* avec *sable* (du à leurs descripteurs similaires). Les classes végétation, bâtiment, et bateaux, sont faibles à cause du manque d'images d'entraînement.

Table 2. Table de confusion.

	Annotées								
Classées	divers	ciel	mer	sable	mont	veg	pers	bat	bateau
divers	0	0	0	0	0	0	0	0	0
ciel	199	1470	636	228	59	4	115	50	21
mer	382	136	1090	314	168	28	89	165	66
sable	245	677	573	1008	452	24	181	152	51
montagne	101	66	120	86	73	14	31	94	24
végétation	84	84	42	183	98	95	17	45	7
personne	305	260	287	601	271	145	601	196	128
bâtiment	54	11	62	90	24	7	33	57	20
bateau	2	15	2	2	0	0	1	1	2

L'exemple suivant montre quelques images segmentées, ainsi que leur nombre de blocs convenablement classés



a) 72% correctement classés

b) 28% correctement classés

c) 32% correctement classés

Figure 8. Exemple d'images segmentées.

Modèle de Markov Caché Multi résolution

Dans l'expérience précédente, nous avons démontré qu'une importante probabilité de sortie dégrade les résultats, mais aussi que le modèle contextuel échoue parfois à caractériser les sous-classes d'un concept, ce qui suggère une échelle trop petite. D'une part, une échelle élevée est nécessaire afin de distinguer les détails des objets, et d'autre part, une faible échelle permet de capturer les propriétés globales. Cela amène l'idée du développement d'un modèle multi résolution.

Le principe de l'analyse multi résolution est de capturer l'information d'une image à différentes résolutions. Nous considérons donc une combinaison linéaire de 2-D HMMs entraînés à différentes résolutions. Comparé à la modélisation hiérarchique, cette approche est plus aisée à construire: Il nous suffit d'entraîner un certain nombre de 2-D HMMs à différentes résolutions et de les combiner par interpolation ou par probabilité jointe. L'architecture d'un système multi résolution est présentée ci-dessous.

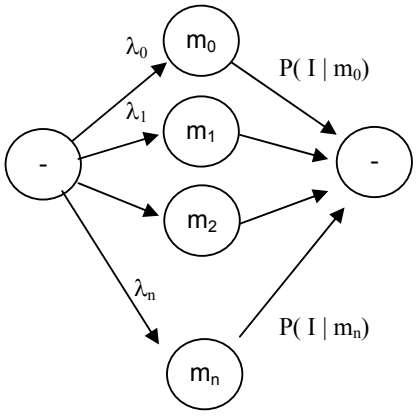


Figure 9. Combinaison linéaire de modèles multi résolution.

Expérience

Pour évaluer les performances de notre modèle, nous le comparons avec le modèle proposé par Jia Li et Al.[39], utilisant le même scénario de classification d'images que dans le chapitre précédent. Les modèles sont entraînés à l'aide d'un ensemble images annotées. La classification est ensuite effectuée sur un ensemble inconnu que le procédé convertit en une liste triée.

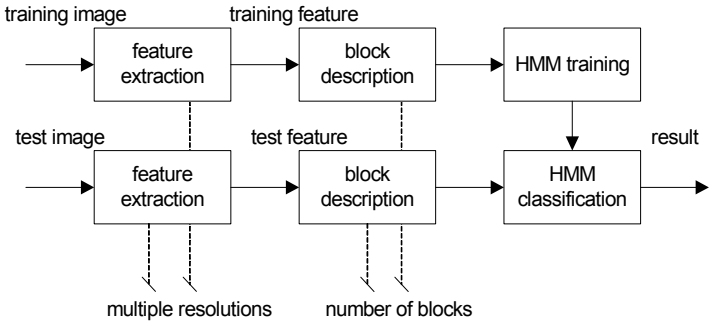


Figure 10. Schéma de classification multi résolution.

En redéfinissant itérativement la taille de l'image, tout en conservant la taille du block constante, nous obtenons les résolutions d'images de (a) 4x3, (b) 16x12 et (c) 64x48 blocks, comme illustré ci-dessous.

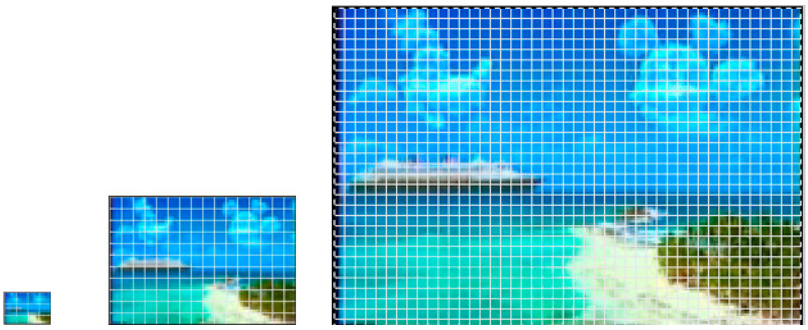


Figure 11. Trois résolutions avec des tailles de blocks constantes: (a) 4x3, (b) 16x12 et (c) 64x48 blocks.

Les modèles sont entraînés à l'aide des algorithmes adaptés de Baum-Welch, comme décrit dans la section 2.4. La Figure 12 montre l'évolution de la probabilité totale pour le DT-MHMM durant l'entraînement.

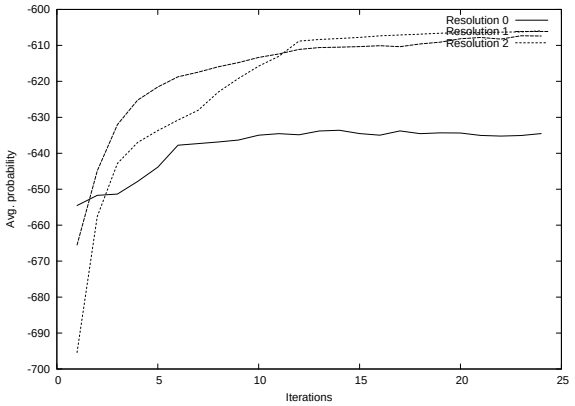


Figure 12. Probabilité totale Durant l'entraînement.

La Figure 13 montre la courbe rappel précision pour le modèle 2-D MHMM (Jia Li et al) et les deux modèles DT-MHMM.

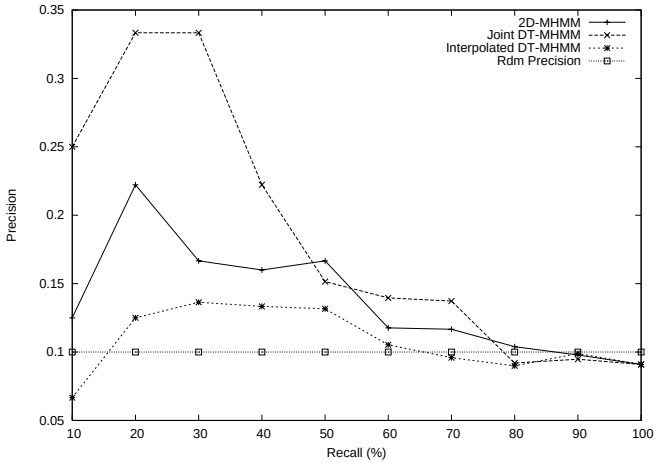


Figure 13. Rappel précision pour 2D-MHMM et DT-MHMM.

On peut observer que le modèle introduit à la plus basse résolution une information globale qui pénalise les images mono-couleurs, mais, selon le schéma de fusion, les modèles de plus hautes résolutions rétablissent parfois l'ensemble. La précision moyenne pour les différents modèles est listée en table 3.

Table 3. Précion moyenne pour DT HMM et 2-D MHMM.

Modèle	**CARTE**
DT MHMM combine jointe	0.24
2-D MHMM	0.17
DT MHMM Interpolatée	0.13

Conclusion

Dans le but d'introduire le contexte local et global, le DT HMM est utilisé comme un modèle d'image de multi résolutions. Le résultat indique que la perte enregistrée peut être diminuée et que les performances sont comparables avec celles des autres algorithmes.

Applications 3-D

Enfin nous présenterons les possibilités d'extensions de la formalisation DT HMM par une étude du 3-D HMM (qui a été rarement étudiée). Nous nous focaliserons sur les données vidéo [66], où les deux premières dimensions sont spatiales, et la troisième temporelle. Pour mieux comprendre cette dernière dimension, nous explorerons les capacités du modèle dans le domaine du suivi d'objets.

En trois dimensions, l'état $s_{i,j,k}$ du modèle dépendra de ces trois voisins $s_{i-1,j,k}$, $s_{i,j-1,k}$, $s_{i,j,k-1}$. Cette triple dépendance accroît le nombre de probabilités de transition dans le modèle, et la complexité algorithmique des algorithmes tels que Viterbi ou Baum-Welch. Cependant, l'utilisation d'un arbre de dépendance 3-D nous permet d'estimer les paramètres du modèle le long d'un chemin 3-D (voir Figure 14) qui maintient une complexité algorithmique linéaire.

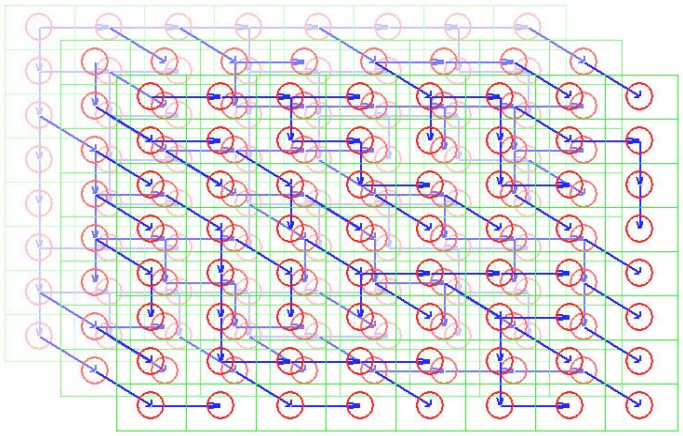


Figure 14. Arbre de dépendance 3-D aléatoire.

La fonction de direction pour l'arbre 3-D devient :

$$\boldsymbol{D}(\boldsymbol{t}) = \begin{cases} \boldsymbol{V} & \boldsymbol{if}\ \boldsymbol{t} = (\boldsymbol{i}-1, \boldsymbol{j}, \boldsymbol{k}) \\ \boldsymbol{H} & \boldsymbol{if}\ \boldsymbol{t} = (\boldsymbol{i}, \boldsymbol{j}-1, \boldsymbol{k}) \\ \boldsymbol{Z} & \boldsymbol{if}\ \boldsymbol{t} = (\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}-1) \end{cases} \tag{1.15}$$

En modélisation 3-D, les images sont représentées par des vecteurs caractéristiques sur une grille 3-D. Notons le vecteur d'observation o_{ijk} comme l'observation du bloc (i,j,k) dans une image 3-D, volume d'images issues d'une séquence 2-D. Par analogie, les variables d'états s_{ijk} du HMM représentent les états aux positions (i,j,k) produisant les vecteurs d'observation o_{ijk}. Nous pouvons donc maintenant étendre **(3.6)** à la troisième dimension:

$$p(s_{i,j,k} \mid s_{i-1,j,k}, s_{i,j-1,k}, s_{i,j,k-1} t) \tag{1.16}$$
$$= p_{D(t(i,j,k))}(s_{i,j,k} \mid s_{t(i,j,k)})$$

Le procédé de suivi se décompose principalement en deux phases : La phase d'entraînement et la phase de segmentation. Lors de la phase d'entraînement, le processus apprend les paramètres inconnus du HMM, à l'aide du système d'entraînement de Viterbi détaillé en section 5.2.1. Au cours de la phase de segmentation, le procédé effectue une segmentation spatio-temporelle en exécutant un alignement d'états 3-D de Viterbi.

Détection d'Objet

La vidéo originale contient deux skieurs passant devant des repères jaunes sur un paysage neigeux avec des ombres. Figure 15 retrace chaque seconde de la séquence.

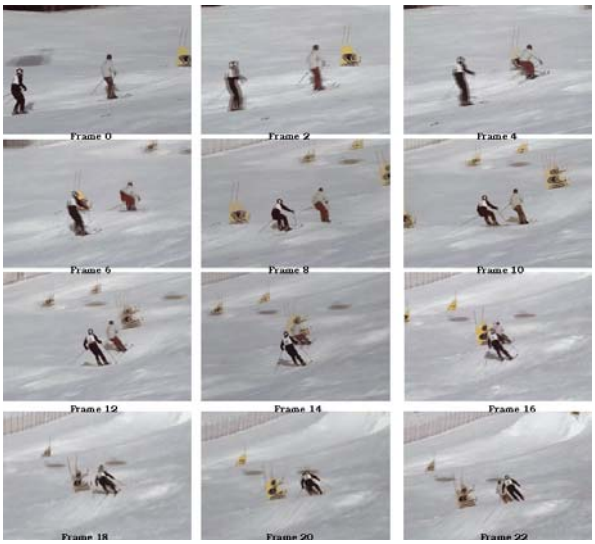


Figure 15. La séquence vidéo commence au coin supérieur gauche, suivi des autres cadres.

Les deux premières images ont été manuellement annotées et utilisées pour évaluer le modèle initial, tandis que les images suivantes constituent l'observation 3-D sur laquelle l'expérience de Viterbi a été exécutée. Puis, nous utilisons le modèle formé pour obtenir un étiquetage finalisé de l'observation 3D complète. Dans l'étiquetage final, chaque bloc d'observation est assigné à un seul état du modèle. L'étiquetage final fournit une segmentation spatio-temporelle de l'observation 3D.

Le dépistage d'objet est alors exécuté facilement, en choisissant dans chaque image les blocs que l'on étiquette en fonction de la catégorie sémantique correspondante.

Par exemple, nous pouvons facilement créer une séquence vidéo contenant seulement les skieurs en excluant le paysage - et les marques repères comme indiqué ci-dessous.

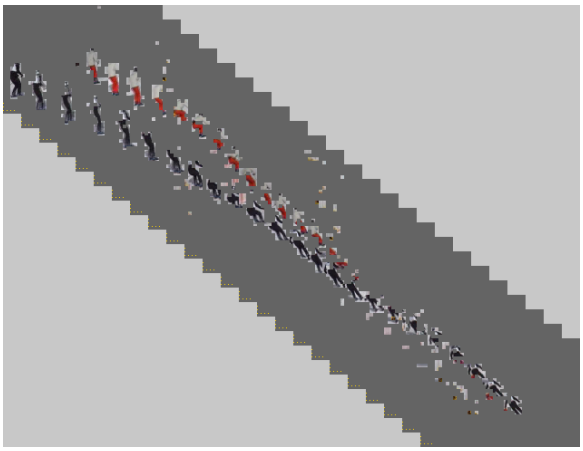


Figure 16. Détection de deux skieurs.

Nous pouvons voir dans la figure ci-dessus que certains blocs ne sont pas correctement assignés aux catégories de skieur. L'explication peut être qu'avec un seul arbre de dépendance, beaucoup de blocs à l'intérieur de la vidéo correspondent à des feuilles dans l'arbre et sont donc dispersés. Cela justifie la combinaison de plusieurs arbres de dépendance pour que la chaine d'images ne soit pas rompue.

Pour chaque arbre de dépendance, nous pouvons calculer le meilleur alignement, utiliser ensuite un vote majoritaire pour choisir l'état le plus probable pour chaque bloc. C'est une approximation pour la probabilité d'être dans cet état pour ce bloc pendant la génération de l'observation avec un arbre aléatoire inconnu. Figure 17 montre la vidéo obtenue avec cet étiquetage d'arbre multiple, en utilisant un jeu de 50 arbres aléatoirement produits.

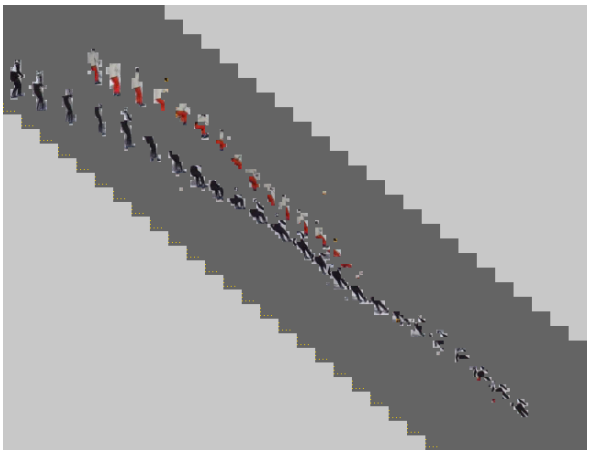


Figure 17. Détection d'objet avec lissage de 50 arbres aléatoires.

Comme le montrent ces résultats, les objets sont très clairement définis dans cette expérience et la plupart des interférences dans l'étiquetage ont disparues.

Enfin nous avons illustré l'approche DT HMM sur le problème de la segmentation vidéo et de la détection. Nous avons détaillé l'application de notre modèle sur un exemple concret. Nous avons aussi montré que quelques artefacts en raison de nos simplifications peuvent être énormément réduits par l'utilisation d'un plus grand nombre d'arbres de dépendance.

Conclusion et Travail Futur

Grace aux nouvelles capacités d'apprentissage de HMM, nous croyons que ce type de modèle sera appelé à être utilisé pour nombre d'applications. On peut envisager d'autres applications 3-D telles que la classification d'images 3-D ou la reconstruction d'image.

Au delà, puisque l'arbre de dépendance présente des discontinuités nous pouvons trouver d'autres façons de choisir un arbre. Faire par exemple un arbre optimal pour un jeu d'images ou application en analysant d'abord chaque image et faire plus de rapports (connexions) entre des régions de frontière, ou considérer d'autres arbres non-aléatoires.

Une future approche du problème de discontinuités est d'utiliser des modèles hiérarchiques, puisqu'un bloc dans une résolution plus basse peut inclure les rapports (connexions) qui n'existent pas à plus petite échelle.

Le modèle peut aussi être appliqué (étendu) aux dimensions supérieures ($n > 3$). Dans ce cas les rapports contextuels deviennent alors plus faibles et pour chaque dimension nous aurons un rapport (connexion)proche de n. Pour cette raison le choix de l'arbre deviendra encore plus important.

En conclusion nous croyons que le DT HMM est un modèle puissant qui a un véritable potentiel d'avenir pour de nombreuses applications.

ACKNOWLEDGMENT

This thesis is the result of three and a half years effort spent at the Multimedia Communications Department of institute Eurecom, which has provided me with enriching industrial and academic experiences.

First of all, I would like to thank my principal advisor, Professor Bernard Merialdo, for his great guidance, support and patience. The weekly meetings have greatly helped in developing my work. He has provided me not only with technical and mathematical knowledge but also a rigorous attitude towards research, for which I am grateful. I also express my gratitude to assistant Professor Benoit Huet, for having many discussions with me in machine learning and image processing.

Some people have given me invaluable help and scientific support during my thesis, among them I would like to express all my gratitude to Vivek Tyagi and Fabrice Souvannavong whose resourcefulness has given me the opportunity to learn and explore different subjects related to signal processing, probability theory and program construction. Special thanks go to Fabio Valente for helping me during the initial part of my PhD.

During my stay at the Eurecom institute, I came across some very special people that I consider friends rather than colleagues: Gwenael, Brian, Federico, Eric, Remi and Rachid. Thank you guys for everything!

Finally I would like to thank the staff at Eurecom for their help with administrative work, and Carole for giving me her remarkable support.

To L + N

CONTENTS

1.1 Motivation 1

1.2 Contribution and Outline 2

2.1 Image Understanding 5

2.1.1 The Meaning of the Picture and Ontologies 6

2.1.2 Hyponomic Ontology 6

2.1.3 Meronomic Ontology 7

2.1.4 Notes from Cognitive Psychology 8

2.2 Knowledge Representation and Control Strategy 8

2.3 Applications 9

2.4 Statistical Learning 12

2.4.1 Concept Learning 12

2.4.2 Classification Algorithms 13

2.4.3 Bayesian Classifier 14

2.4.4 Gaussian Mixture Model 16

2.4.5 Naive Bayesian Classifier 21

2.4.6 Dynamic Bayesian Nets 21

2.4.7 Hidden Markov Model 22

2.4.8 HMM and Image Modeling 23

2.4.9 Reestimation Formulas 24

2.5 2-D HMM 27

2.5.1 Necessary 2-D Extensions for Image Classification 28
2.5.2 Markov Random Field 29
2.5.3 Markov Mesh Random Field 31
2.5.4 Previous Work on 2-D HMM 33
3.1 Dependency Tree 40
3.2 Solution to Problem 1 (Evaluation Problem) 42
3.3 Solution to Problem 2 (Decoding Problem) 43
3.4 Solution to Problem 3 (Learning Problem) 44
3.5 Implementation Issues for HMMs 49
3.6 Experiment 50
3.6.1 Context Dependent Image Categorization 50
3.6.2 System Design 51
3.6.3 Extracted Low-Level Features 52
3.6.4 The Dataset 53
3.6.5 Results 54
3.6.6 Conclusion 57
3.7 Combination with a Global Model 58
3.7.1 Vector Quantization 58
3.7.2 Global Model 60
3.7.3 Results 60
3.7.4 Conclusion 63
3.8 Influence of the Dependency Tree 64
3.8.1 DT HMM Probability Estimation 64
3.8.2 Estimation by Average 65
3.8.3 Estimation by unique sampling 66
3.8.4 Estimation with dual tree 67
3.8.5 Conclusion 68
4.1 Semantic Image Segmentation 69

4.1.1 Related Work 70
4.1.2 Semantic Segmentation 70
4.1.3 States with semantic labels 71
4.1.4 Model Training 72
4.1.5 Experiment 74
4.1.6 Conclusion 76
4.2 Multiresolution Hidden Markov Model 78
4.2.1 Previous Work on 2-D MHMM 78
4.2.2 DT MHMM 82
4.2.3 Algorithms 83
4.2.4 Experiment 84
4.2.5 Conclusion 87
5.1 Introduction 89
5.2 3-D DT HMM 90
5.2.1 3-D Viterbi Algorithm 91
5.2.2 Relative Frequency Estimation 93
5.3 Experiment 94
5.3.1 Model Training 94
5.3.2 Object Tracking 96
5.3.3 Conclusion 100
6.1 Summary and Contributions 101
6.2 Future Work 104
Appendix A Training Data **107**
A.1 TRECVid Archive 107
A.2 Low-level Features 109
Appendix B Implementation Notes **115**
Appendix C Notation and Abbreviations **119**
Bibliography **123**

LIST OF TABLES

Table 1. Positive and negative training examples for the target concept "Dangerous dive"..... 13
Table 2. Average precision on large test set. 61
Table 3. The set of states for each sub-class. 71
Table 4. Confusion Table. 77
Table 5. Mean average precision for DT HMM and 2-D MHMM. 87
Table 6. The number of states for each sub-class. 95
Table 7. Number of ancestors for each direction. 96
Table 8. Machine learning terminology. 119
Table 9. Notation for classification algorithms. 120
Table 10. Notation for HMMs. 120
Table 11. List of abbreviations. 121

LIST OF FIGURES

Figure 1. The human brain is a pattern recognizer. 8
Figure 2. Architecture of a model-based control strategy. 9
Figure 3. An example surface of a two-dimensional Gaussian mixture PDF with three components. 19
Figure 4. The Bayesian network (BN) spells out the factorization that is a simplification of the chain rule of probability. 21
Figure 5. The one-dimensional hidden Markov model. 23
Figure 6. (a) Image decomposition into blocks, (b) states of the Markov model. 28
Figure 7. First-order neighborhood system for MRF-based models. 30
Figure 8. State dependency according to the second order Markov mesh. 32
Figure 9. The observation for the word "ul" and the pseudo 2-D HMM. 34
Figure 10. Subset of state configuration along diagonals. 35
Figure 11. Viterbi state transition diagram. 37
Figure 12. 2-D Neighbors. 40
Figure 13. Example of a random dependency tree. 41
Figure 14. The inside probabilities 42
Figure 15. The outside probabilities 45
Figure 16. Image Categorization Scheme. 52
Figure 17. Images decomposed into blocks with different sizes; (a) 44x40, (b) 16x15 and (c) 8x8 pixels 52
Figure 18. DCT coefficients of a 8 x 8 image block. 53
Figure 19. Images annotated "Waterscape_Waterfront" from the TRECVid 2005 archive 53

Figure 20. Likelihood of the training data for three different gpm's 54
Figure 21. 20 top ranked images are uni-colored. 55
Figure 22. The first true positive appears on position 122 of 9473 test images. 55
Figure 23. Avg. precision for block size: (1) 176x120, (2) 88x60, (3) 44x40, (4) 16x15, (5) 8x8, (6) 4x4, (7) 2x2 57
Figure 24. Codewords in three dimensional space. Input vectors are marked as a cross and codewords as an arrow. 59
Figure 25. Global LBG Histogram. 60
Figure 26. Recall / Precision for Beach 61
Figure 27. 20 top ranked images using a combination of the global model and DT HMM. 62
Figure 28. 20 top ranked images using the DT HMM. 62
Figure 29. The Joint probability of the observation given the model, and the tree, at position (i,j). 64
Figure 30. State alignment with two different trees. 65
Figure 31. Convergence of probability average. 66
Figure 32. Distribution of scores. 67
Figure 33. Example of images with mixed class areas. 71
Figure 34. Annotating an image segment as "sky". 72
Figure 35. Image segmentation schema. 72
Figure 36. Example of training images. 73
Figure 37. State segmentation after 0, 2, 6 and 10 iterations. 74
Figure 38. Likelihood of the training data. 74
Figure 39. State segmentation on test image. 75
Figure 40. Better performance with smaller minimum variance for GMMs. 75
Figure 41. Test images with ambiguous regions. 76
Figure 42. Labeling without/with transition probabilities. 76
Figure 43. Example of segmented images. 77
Figure 44. Quad-tree view of the parent-child dependencies of the 2-D wavelet. 80
Figure 45. Hierarchical Multiresolution Model. 81
Figure 46. Linear combination of multiple resolution models. 83
Figure 47. Multiresolution classification scheme. 84

Figure 48. Three resolutions with constant block size: (a) 4x3, (b) 16x12 and (c) 64x48 blocks. 85
Figure 49. Examples of training images for “beach”. 85
Figure 50. Total likelihood during training. 86
Figure 51. Precision recall for 2D-MHMM and DT-MHMM. 86
Figure 52. Top ranked images by the DT-MHMM. 87
Figure 53. Random 3-D Dependency Tree. 91
Figure 54. Training image and initial state configuration using annotated regions. . 95
Figure 55. Original video sequence; first frame in upper left corner, followed by every second frame. 97
Figure 56. Frame segmentation in the final labeling (a) frame 1, (b) frame 12, (c) frame 24. 97
Figure 57. Object tracking of two skiers. 98
Figure 58. Frame segmentation using complementary dual trees, (a) frame 1, (b) frame 12, (c) frame 24. 99
Figure 59. Perspective view of object tracking using complementary dual trees. 99
Figure 60. Object tracking with smoothing over 50 random trees. 100
Figure 61. DCT coefficients of an 8 x 8 image block. 110
Figure 62. Example of Gabor kernels at 4 scales and 8 orientations 111
Figure 63. Example of marked edges in an image using Canny's algorithm. 112
Figure 64. DT HMM object relations. 116
Figure 65. Low-level feature extraction objects. 117
Figure 66. DT HMM object relations. 118

Chapter 1

Introduction

1.1 Motivation

The average person with a computer will soon have access to the world's collections of digital video and images. However, unlike text that can be alphabetized or numbers that can be ordered, there is no general formalism to organize image and video. Although tools which can ``see'' and ``understand'' the content of imagery are still in their early years, they are now at the point where they can provide significant assistance to users in navigating through visual media.

The challenge in developing techniques for media semantics requires knowledge and techniques from a variety of disciplines and domains, many of them outside of traditional computer and information science. Image understanding is a kingpin thereof and is a most complex challenge of AI. To cover this complicated area of computer vision in detail it would be necessary to discuss many other branches such as; knowledge representation, semantic networks, image processing, classification algorithms learning from experience, etc. The central problem is to bridge "the semantic gap", which describes the difference between the meaning that users expect systems to associate with their queries, and low-level features that the systems actually compute. A number of researches have introduced systems that bridge the gap between low-level features and semantic classes [3], [5], [6], [8], [25], [38], [52], [68].

Semantics is meaningful only in context; it can not exist without a knowledge base to project its concept on [35]. In this work we employ a probabilistic framework, which means that the knowledge database, the meta-data of the image is stored numerically and we use a state-transition model (a hidden Markov model) for capturing the context and dynamics of images and video.

The hidden Markov models (HMM) have been successfully introduced to many important problems in image processing such as computer vision or pattern recognition. Their success is due to both their rich mathematical structure which engenders a theoretical basis for many domains, and to the Baum-Welch algorithm [38]. The Baum-Welch procedure is an efficient training algorithm that allows estimating the numeric values of the model parameters from training data.

However for images computations becomes intractable because of the statistical dependencies in two dimensions. Many approaches have been proposed to preserve a modest computation [39], [44], [49], [50]. The disadvantages of these approaches is that they either greatly reduce the vertical dependencies between states, which is then only achieved through a single super-state, or introduces simplifying assumptions and approximations so that the probabilistic model is no longer theoretically sound.

1.2 Contribution and Outline

In this thesis we present a new type of multidimensional hidden Markov model that is efficient in computational complexity and storage, while still being theoretically sound. The basic idea is to relax the joint dependencies between neighboring states by a dependency tree.

We derive the necessary expressions for the procedures associated with HMMs such as the Baum-Welch and Viterbi algorithms. The model is embedded in an image modeling framework for benchmarking and investigating the properties of the formalism. The experimental parts deal mostly with classification problems since they are easy to evaluate and due to the fact that we have access to a common annotated video database used by the TRECVid workshop [66]. The outline of the dissertation is as follows:

- Chapter 2 provides a perspective of image understanding from outside of traditional computer and information science. I review the literature on statistical learning methods and give an introduction to the discipline of Markov models.

- Chapter 3 constitutes the theoretical core of this dissertation. Here we present our new hidden Markov model based on a random dependency tree, which will later be referred to as the *dependency tree hidden Markov model* (DT HMM). The algorithms associated with HMMs are derived and we show that for this model, most of the common algorithms keep the same linear complexity as in one dimension. We provide experimental details and compare the results with those of the TRECVid workshop.

- In chapter 4 we specialize the model to the problem of image segmentation. We review the current state-of-the-art and present a solution based on the

DT HMM. Semantic regions are implemented by restricting a number of states to a sub-class. This chapter also deals with the expansion to multiple resolutions. The extension allows an image to be represented by observations in several resolutions which corresponds to local and global context. Comparisons, in an image classification scenario are made between the multiresolution DT HMM and another well known multiresolution model.

- Chapter 5 considers an extension to three dimensions in a video modeling scenario. Since video can be regarded as images indexed with time, we generate a 3-D dependency tree and compute the transition probabilities over space and time. We demonstrate the potential of the model by applying it to the problem of tracking objects in a video sequence. We explore various issues about the effect of the random tree and smoothing techniques. Experiments demonstrate the potential of the model as a tool for tracking video objects with an efficient computational cost.

Chapter 2

Image Classification

A picture can be very useful in answering questions. Like how where people dressed during the 1860's or how does a Lemur look like? However, retrieving a picture that answers a particular question can be difficult. Investigating the meaning of pictures can be compared to subject analysis of text, where the sense of the words are of concern not the words of the text, nor the bibliographic description or genre exemplified by the work.

In this chapter we present a survey of the literature on image classification. We start with a review outside computer science by touching different linguistic and cognitive aspects of image understanding. I look for answers to questions such as "what is the meaning of a picture?" and how can the relation between visual objects in an image be described? We then present important classification techniques in section 2.4.

2.1 Image Understanding

Image understanding is one of the most challenging fields of machine learning, and it depends on other independent domains such as: knowledge representation, semantic networks such as ontologies, inference, classification, segmentation, learning from experience and more.

Some successful attempts to model media semantics make use of ontologies[1] [28], [29]. The advantage of ontology learning is that its influence path is based on ontol-

[1] In this context an *ontology* denote a taxonomy with a set of inference rules.

In philosophy, ontology (from the Greek ὄν, genitive ὄντος: *of being* (part. of εἶναι: *to be*) and -λογία: *science*, *study*, *theory*) is the study of being or existence.

ogy hierarchy, which has real semantic meanings. In ontology-based learning there are two kinds of influences; boosting and constraints. Boosting is about boosting the precision of concepts by taking the results from more reliable ancestors. The constraints are used to decrease the probability of miss-classifying concepts that cannot coexist.

2.1.1 The Meaning of the Picture and Ontologies

To decide the subject of a picture, it is necessary to determine the meaning conveyed by the images within, and the relationship between this meaning and the words used to describe it. In analyzing the kinds of meaning a picture may have, and the relationship between the words used to describe it. Shatford [1] proposes a system for classifying the subjects of a picture, into *Generic Of, Specific Of* and *About*. First the meaning can be divided into the generic description *OF* the represented objects and actions, and the intrinsic meaning of the content (*ABOUT*). Since pictures are simultaneously generic and specific; a picture of a bridge is both that particular bridge and the generic bridge. She divides the *Ofness* into generic and specific, so we have:

Generic Of: Ofness, equivalent to the generic meaning of an image. Requires only "everyday familiarity with objects and actions" e.g. man, woman, child, lifting a hat. Skyscraper , Office Building.

Specific Of: Demands educational knowledge like "familiarity with specific themes and concepts", e.g. to understand that haft-lifting is a Greeting gesture", or a particular building is the Chrysler Building. There can be several specific subjects in a picture (referents) that determine the sense of the picture (c.f. *meronony* below).

Generic About: Description of a mood, identification of mythical beings that have no concrete reality, or symbol meanings abstract concepts communicated by images. Emotions; love, sorrow and concepts: truth, honor and strength, e.g. expressional like "the pity of the Crucifixion", "modern architecture".

2.1.2 Hyponomic Ontology

The crux is to determine the meaning of a picture so that we can classify it in accordance with its meaning. Only then will the user be able to search content on whatever level: generic or specific. This is particular useful since a user can only express the needs in terms of what s/he knows. A *hyponomic ontology* (hyponomy = is-kind-of relationships) will allow us to retrieve specific content by performing a generic search, i.e. the user can search on a subject without knowing its specific name. A side effect is that the user might learn specific facts. E.g. a search on "Gothic church" can retrieve Notre Dame. In this way one could say that we use an ontology to map between the specific- and the generic as mentioned above. We employ this kind of

ontology in the personalization engine for interactive TV [16], in order to match video object concepts with user preferences.

Introducing the relation ship of *Aboutness*, the description of a mood or a symbol, would extend the possibilities to search for a picture that represents strength, vanity or modern architecture. It is perhaps worthwhile mentioning that the factual meaning is easier to classify as people are more likely to agree on descriptions on objects rather than an emotion or mood.

2.1.3 Meronomic Ontology

Some attempts have been made to weight co-occurring multimodal features in order to infer the occurrence of semantic objects [3]. In a model based approach [68], Golshani et Al. measured the semantic similarity of quantized visual cue's using a correlation matrix and then mapped them to semantic labels. An ontology was integrated to further facilitate the translation of text queries into visual queries. Hence if there was no model learned for a certain key word the semantically closest will be subsumed from the ontology by finding the hyponyms. The underlying idea resembles that of LSI; to express a certain topic in text a certain collection of words will be used. The collection will be perturbed by the existence of synonyms and polysemous word.

Another approach would be to use a meronomy ontology (meronomy = part-whole relationship) to map co-occurrences of several referents to a subject. A picture is characterized by one or several objects (or *referents* in [1]). Shatford proposed a method of thresholds recommendations; only name what is whole, not necessarily an integral part of a larger whole. In our framework it would translate to use the ontology to name only larger parts and use the "has-a" relation (like in part-based object detection for example [4]).

As discussed earlier, since a picture can have several referents with its senses we can use an ontology to explain what sense the co-occurrence of some referents (visual obejcts) have. A *cityscape* consists of *sky*, *buildings*, *roads*, *rivers* etc. On a lower level of detail one might say that the presence of a large number of windows in rows formed in layers, with walls and a roof forms a skyscraper, similar to the "is-part-of " relationship.

The need of co-occurrence and spatio-temporal information is even more important in presence of blur while performing object recognition. As pointed out by the authors in [6] the sense of an image is strongly judged by the position of the objects within an image and not their shape.

The structure of an ontology can also be used in order to choose an adequate feature model for each type of image, as well as different classifiers for different problems. In a system developed by Minka and Picard [2] it is assumed that there is no single model that can capture everything what humans perceive in images, their system used a "society of models". The system internally generates several groupings of

each image's regions based on different combinations of features, then learns which combinations best represents the semantic categories given as examples by the user.

2.1.4 Notes from Cognitive Psychology

According to the theory of cognitive science, the impressive abilities demonstrated by the human brain originate mainly from one basic ability: pattern recognition. Our brain is a generalized pattern-recognition machine. For example, there are many different shapes for tables, but somehow we always implicitly recognize a table when we see one, even if we have never seen that particular one.

In some particular cases, the human brain's ability to recognize patterns is overeager, so that it recognizes patterns where there are none. This phenomenon is called pareidolia, and is sometimes used to explain phenomena's like the Loch ness monster, astrology and the man in the moon. The effect is easy to recall; try to count the black dots in the figure below.

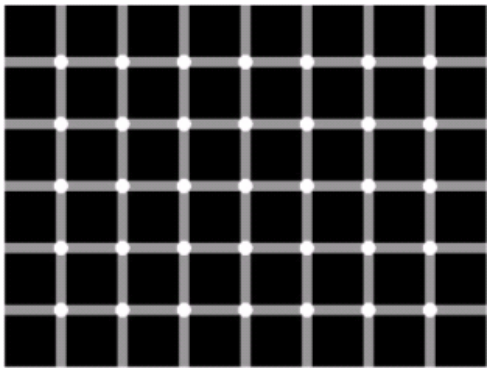


Figure 1. The human brain is a pattern recognizer.

2.2 Knowledge Representation and Control Strategy

Pictures are a kind of sensory data, and there is a lot of work going on to figure out how to index this sensory data. What kind of knowledge representation should be used for its meta-data? Currently the most important attempts to provide standards for description of content are MPEG-7 from ISO, and the semantic web from the W3C. In a paper by Ramesh Jain [31] some interesting ideas are presented on how to store and interact with spatio-temporal data.

As I mentioned in the introduction, the computed feature-based signatures by them selves infer nothing about the content, only that the content is different from surrounding clusters. To extract meaningful semantics a knowledge base is needed to project concept on. The content and representation of this knowledge is of main concern.

In this thesis we use a 2-D HMM as the backbone of the knowledge representation in conjunction with a set of sub-class definitions. The HMM is a generic model, that works in a top-down fashion which mean that an internal model is generated and is verified against a test collection Since a top-down model is only testing to retrieve images that we know something about, there is no limitation in adding prior knowledge limited to a certain domain. We can for instance add "is-part-of" knowledge (c.f. section 2.1.3) such as *sky*, *sea* and *sand* are parts of the concept *beach*. An example of this is illustrated in section 4.1.2.

The architecture of a generic top-down classification system is depicted in Figure 2. The principle of a top-down control is the construction of an internal model and its verification, meaning that the principle is *goal oriented*, like looking for your car at a parking lot.

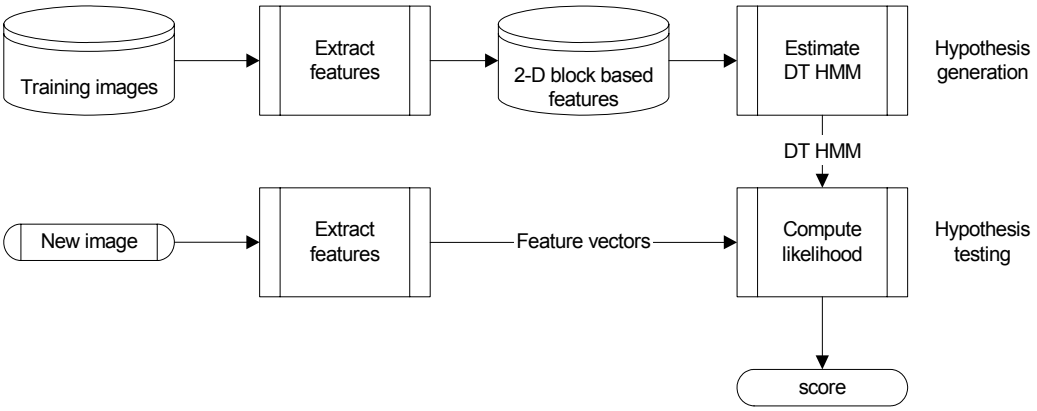


Figure 2. Architecture of a model-based control strategy.

The image understanding process consists of sequential hypothesis generation and testing. The hypothesis testing algorithms represent data by a set of points (or prototypes). A class is assigned to each prototype by majority vote on the associated class distribution of the prototype. A test feature vector is identified as the class of its closest prototype.

2.3 Applications

There are many areas of applications for image understanding: robot vision, remote sensing, surveillance applications, military applications, medical image understanding, and entertainment. In this section we shall present some examples from each category.

Robot Vision
Engineers at DaimlerChrysler are working on technology that can make cars watch out for hazards, listen to the driver or even know when he/she is distracted, in order to alert the human driver to potentially dangerous situations.

Prototype cars, equipped with stereo cameras are able to recognize hazards that the driver has overlooked – like some one running over the road or a bouncing ball. It is able to spot traffic lights and by swift camera movements is even able to check that they are still green when the car approaches [7].

Remote sensing
In remote sensing for forestry applications the main goal is to fully or partly replace the human image interpreter by a seeing computer, capable of making many decisions on its own, with a minimum of human intervention during the image processing and analysis. A review of the state-of-the-art of the research from different countries is given in Hill and Leckie [8].

Unsupervised extraction of roads eliminates the need for human operators to perform the time consuming and expensive process of mapping roads from satellite imagery. As increasing volumes of imagery become available, fully automatic methods are required to interpret the visible features such as roads, railroads, drainage, and other meaningful curvilinear structures in multi-spectral satellite imagery. The challenge of detecting curvilinear elements is also related to the problem of deriving anatomical structures in medical imaging as well as locating material defects in product quality control systems and cartographic applications [9].

Surveillance Applications
Surveillance applications often collect a large amount of video data. Currently the surveillance applications do not allow the user to quickly search the collected data for an occurrence of a particular individual. Face based browsing for surveillance applications such as ID system for the police force to detect the face of criminals, may enhance airport security.

The ultimate goal of surveillance systems is automatic detection of events and suspicious activities that triggers an alarm (detection) as well as reducing the volume of data presented to human operator (retrieval). Event detection requires interpretation of the "semantically meaningful object actions". Highway monitoring, airport surveillance, building access control are just a few of the several important applications.

Whereas most models for detecting unusual events assume that “all” unusual event events can be modeled, which requires off-line training. The research lab MERL[2] employs an unsupervised learning method that does not require definition of what is usual and what is not. They define usual as the high recurrence of events that are

[2] Mitsubishi Electric Research Laboratories

similar. As a result, unusual is the group of events that are not similar to the rest, which also allows detecting multiple unusual events [10].

The same lab has also developed a technology for automatically detecting pedestrians in video sequences. Detecting and tracking pedestrians can be used to sound an alarm if an intruder is in a restricted area or to aid in browsing hours of surveillance video by skipping to the next part of the video where a person was seen.

The problem of detecting people in low resolution surveillance video is difficult because the pedestrian may be very small in the image, making the amount of information contained in the pixels small. Furthermore, there may be background movement in the scene such as trees waving or a cloud shadow passing by which makes causes motion detectors to fail. Their approach is to encompass both the appearance and the motion of pedestrians in the model [11].

Military applications
The future battlefield is characterized by an expanding suite of sensors and sensing modalities collecting vast volumes of imagery from a mixture of ground, air, and space-borne platforms. Image understanding techniques are needed to extract the information needed by military forces from this data-rich environment.

Image understanding in military applications also provide advanced vision systems to aid intelligence image analysts, or enables an unmanned military ground vehicle to scout for enemy targets [12], [13].

Medical Image Understanding
Clinicians routinely employ a variety of imaging techniques during patient diagnosis. The volume of information, and the difficulty of interpreting it, make this area one in which advanced image understanding can make significant improvements in the detection and treatment of illness e.g. 2-D functional analysis of the heart, mammography image analysis, identify lung cancer cells and discriminate among different lung cancer types [14], [15].

Interactive Television Projects
Advanced Digital Television provides an exciting new realm for the consumer. Not only does it offer improved picture quality, it also provides a means for seamlessly blending many new services into the TV set, expanding the scope of what televisions can do. Advanced Digital Television also poses new challenges in video encoding, transmission, and reception.

MPEG2 has been successfully adopted in digital broadcasting and computer video applications. Now the coding technologies are evolving from MPEG2 to MPEG4 which standardizes algorithms and tools for flexible representation of audio-visual data in an object-oriented manner.

The object-based compression in MPEG4 enhances the user's interaction with a device or computer. This aspect has been investigated in the GMF4iTV [16] project

and a demo version of the interactive video system were presented at *4th Workshop on Personalization in Future TV* [17]. The objective of the prototype was to build an end-to-end broadcast system for providing personalization and interactivity to TV programs through active video objects [18].

Entertainment
Vision-based interfaces for computer games allow the player to move or gesture to affect the game, instead of pressing buttons. The characters in the game may imitate those motions, or respond accordingly.

Vision can be a powerful interface device for computers. There is the potential to sense body position, head orientation, direction of gaze, pointing commands, and gestures. Such free and untroubled interaction can make computers easier to use. The application of vision to computer games poses special challenges. The response time must be very fast, while the total hardware cost must be very low. An approach in using fast and simple algorithms to meet these challenges is presented in [19].

2.4 Statistical Learning

Statistical modeling methods are paramount in today's large-scale image analysis. It is critical for almost all image processing problems, such as estimation, compression and classification. This section gives a brief introduction to the theory of inductive learning, followed by a presentation of the currently most important classification algorithms.

2.4.1 Concept Learning

In the process of learning, general concepts are formulated from specific examples. Humans incrementally learn new concepts such as “butterfly”, “prime numbers”, “explosion” etc. Each such concept can be viewed as describing some subset of objects or events over a larger set (e.g. the subset of insects that constitute butterflies). A concept can be modeled as a function defined over this larger set, e.g. a function defined over all insects whose value is true for butterflies and false for all other insects.

In machine learning the desire is to induce[3] a general function (a hypothesis) that best fit a set of training examples. This is sometimes referred to *concept learning*: the problem of automatically inferring the general model of some concept, given examples labeled as members or nonmembers of the concept.

[3] Induction: the process of deriving general principles from particular facts or recurring phenomenal patterns.

Decision trees, artificial neural networks and genetic algorithms are all examples of inductive learning methods that generalize from observed training examples by identifying features that effectively discriminate positive from negative training examples.

We give an example to clarify the idea Table 1 shows a number of examples (or *instances*) of the concept "Potential dangerous dive". Each instance is represented by a set of attributes. The attribute *Dangerous* indicates whether the particular dive is considered dangerous and is also called the target concept. If the value is 'Yes' then the instance is said to be a positive example, respectively a negative example if the value is 'No'. The goal is to be able to predict this attribute for random examples.

Table 1. Positive and negative training examples for the target concept "Dangerous dive".

Example	Strong current	Depth	Night	Experience	Dangerous
Dive 1	Yes	< 10m	No	Medium	No
Dive 2	Yes	> 10m	No	Medium	Yes
Dive 3	No	> 10m	Yes	Novice	Yes
Dive 4	Yes	> 10m	No	Advanced	No

When learning the target concept, the model is presented a set of training examples, each consisting of an instance x from X (the set of examples over which the concept is defined), along with its attributes and target concept c(x) (also called class label).

If we denote by Ω the set of all possible models that might be considered to estimate 'c', then in general each $\omega \in \Omega$ represents a target function defined over X whose value is true for instances belonging to the concept and otherwise false;ω: $X \rightarrow \{0,1\}$. Usually Ω is determined by the designer's chose of representation which depends on the application as we shall se in next section.

2.4.2 Classification Algorithms

In the communities of image understanding and computer vision, classification is a very common task and it is generally not easy. The problem can also be expressed as "understanding what the data means". Some of the potential applications are, image segmentation, semantic classification, object tracking and recognition. While these tasks might be easy for humans they are very hard for computers. Even with the simplest cases there are noise and distortions affecting the results making the classification nontrivial.

Classification algorithms do not usually work well with raw data, for example where a large array of numbers represents a digital image. Data have to be preprocessed to extract few pieces of valuable information called features (see section A.2). Features are represented as a feature vector where the dimension of a vector is the number of scalar components of different features. Feature extraction is very important for achieving good classification results and is typically application-specific. In the previous section we used the general term *instance* for a concept example, from now onwards we shall instead refer to a *feature vector*. Feature vectors with class labels (earlier referred to as the target concept) can be used to estimate a model describing a concept, provided that there are enough good samples available [23].

As before mentioned some assumptions have to be made about the structure of the estimated model since totally arbitrary models are difficult to train. This is the same as saying that inductive learning methods require some form of prior assumptions, for example in Bayesian classification it can be assumed that a class can be represented in feature space with a Gaussian probability density function (PDF) (c.f. section 2.4.4). The classification of unknown samples is based on estimated class representations in a feature space.

A Classifier is an algorithm with features as input and concludes what it means based on the information that is encoded into the classifier algorithm and its parameters. The output is usually a label, but it can also be a real value.

To design the classifier, it is necessary to have knowledge about the classification task at hand. For example what type of classifier is appropriate and what is the best inner structure. For example the number of neurons and layers in a neural network, the probability density function for a Bayesian classifier, or number of states in a hidden Markov model. Classifier complexity is a tradeoff between representational power and generality. A simple classifier may not be able to learn or represent classes well which yields poor accuracy. An overly complex model can lead to over fit the training examples, which means that it exist some other model that fits the training examples less well but actually performs better over the entire distribution of instances.

2.4.3 Bayesian Classifier

The concern in machine learning is to determine the best model from some space Ω of possible solutions, given the training data $\mathbf{X}$. One way to specify what is meant by the *best* is to say the most *probable* model (lowest risk or expected cost), given the samples $\mathbf{X}$ plus any knowledge about the prior probabilities of the various models Ω.

Let us consider the general pattern classification problem where a sample $\mathbf{x}$ is to be assigned to one of a set of possible classes $\{c_i\}$. Within the Bayesian decision framework, the optimal classifier frequently referred to as the *minimum risk classifier*

employs the following decision rule: assign the observed sample $\mathbf{x}$ to the class c_i that minimizes the *conditional risk* given by:

$$\boldsymbol{R}(\boldsymbol{c}_k|\boldsymbol{x}) = \sum_l \lambda(\boldsymbol{c}_k \mid \boldsymbol{c}_l)\boldsymbol{P}(\boldsymbol{c}_l \mid \boldsymbol{x}) \tag{2.1}$$

where the *loss function* $\lambda(c_k|c_l)$ quantifies the loss incurred for selecting c_k when the true class of $\mathbf{X}$ is c_l , and where $\mathbf{P}(c_l|\ \mathbf{x})$ is the (posterior) probability of class c_l given that sample $\mathbf{x}$ was observed, which is computed from the class prior probabilities and class-conditional probabilities in (2.1).

Bayesian classification and decision making is based on probability theory and the principle of choosing the most probable model. Assume that there is a classification task to classify vectors (samples) to K different classes. A feature vector is denoted as $\mathbf{x} = [x_1, x_2,\ldots,x_D]$ where D is the dimension of a vector. The probability that a feature vector $\mathbf{x}$ belongs to a class c_k is $P(c_k|\ \mathbf{x})$ and is referred to as a posteriori probability because it represents our confidence that c_k holds after we have seen the training data $\mathbf{X}$. In this way the posterior probability reflects the influence of the training data, in contrast to the prior probability $P(c_k)$ which is independent of $\mathbf{X}$. The posterior probabilities can be computed with the Bayes' rule:

$$\boldsymbol{P}(\boldsymbol{c}_k|\boldsymbol{x}) = \frac{\boldsymbol{P}(\boldsymbol{x}|\boldsymbol{c}_k)\boldsymbol{P}(\boldsymbol{c}_k)}{\boldsymbol{P}(\boldsymbol{x})} \tag{2.2}$$

Where $p(\mathbf{x}\ |c_k)$ is the probability density function of class c_k in the feature space (which will be further discussed in section 2.4.8) and $P(c_k)$ is the a priori probability, which tells the initial probability of the class before measuring any features and may reflect any background knowledge about the model. The divisor is just a normalizing constant that ensures that the posterior is a probability, i.e., adds up to 1. The Bayes classifier is called a *generative* classifier since the distribution $P(x|\omega)$ describes how to generate random instances $\mathbf{x}$ of the target model ω.

The training

To train a model we want to find the most probable model given the training data, i.e. find the optimal model ϖ that maximizes $P(\omega_k|x)$. This maximal probability is called maximum a posteriori (MAP) solutions, because it maximizes the posterior probability given x. By using the Bayes' rule and dropping P(x) in the last step since it's a constant independent of ω we get:

$$\begin{aligned} \varpi_{MAP} &= \arg\max_{\omega\in\Omega} \boldsymbol{P}(\omega|\boldsymbol{x}) \\ &= \arg\max_{\omega\in\Omega} \frac{\boldsymbol{P}(\boldsymbol{x}|\omega)\boldsymbol{P}(\omega)}{\boldsymbol{P}(\boldsymbol{x})} \\ &= \arg\max_{\omega\in\Omega} \boldsymbol{P}(\boldsymbol{x}|\omega)\boldsymbol{P}(\omega) \end{aligned} \tag{2.3}$$

If all models ω are considered to be equally likely then P(ω) is uniform and equation (2.3) can be further simplified by dropping the term, thus:

$$\omega_{ML} = \arg\max_{\omega\in\Omega} \boldsymbol{P}(\boldsymbol{x}|\omega) \tag{2.4}$$

The result is called the *maximum likelihood* estimation (ω_{ML}). The term *likelihood* is usually used for approximations of probability density functions since it measures how well the distribution fits the observed data.

The major problem in the Bayesian classifier is the class-conditional probability density function $p(\mathbf{x}|c_k)$. The function tells the distribution of a feature vector in the feature space inside a particular class, i.e., it describes the class model. In practice it is always unknown.

2.4.4 Gaussian Mixture Model

The Gaussian probability density function is often used to approximate the class-conditional distribution. It is mathematically sound and extends easily to multiple dimensions. The assumption is that the model is truly a model of one basic class. However that is normally not the case, in a particular image class we might have several different regions emitting features of several different types. A single Gaussian approximation would describe the class with a wide range of features, including patterns that might not belong to the class at all.

Therefore a Gaussian mixture model (GMM) is often used to provide a multimodal density. The GMM is a combination of several Gaussian distributions and can therefore represent different subclasses inside on class. The idea is to fit the data in space by placing blobs of probability mass. If we believe that data lies around a number of clusters, we can model each cluster *i* by a distribution $P(x; \mu, \Sigma)$. To model the whole data set we need to describe how much probability mass to attach to each cluster using a mixture weight α_i, we get:

$$\boldsymbol{p}(\boldsymbol{x};\Theta) = \sum_{i=1}^{M} \alpha_i \boldsymbol{P}(\boldsymbol{x};\mu_i,\Sigma_i) \tag{2.5}$$

where $P(x;\mu,\Sigma)$ is the D-dimensional Gaussian probability density functions with mean μ and covariance Σ matrix, given by:

$$\boldsymbol{P}(\boldsymbol{x};\mu,\Sigma) = \frac{1}{(2\pi)^{\boldsymbol{D}/2}\sqrt{\det(\Sigma_i)}}\exp(-\frac{1}{2}(\boldsymbol{x}-\mu_i)^{\boldsymbol{T}}\Sigma_i^{-1}(\boldsymbol{x}-\mu_i)) \tag{2.6}$$

Thus the GMM density function is defined as a weighted sum of Gaussians (2.5) that is governed by its means, covariances and mixing coefficients, often denoted $\Theta = (\alpha_i, \mu_i, \Sigma_i,)$.

EM algorithm

One way of finding the Θ parameters for a set of data is by maximum-likelihood estimation. This leads however to a hard problem, for which an analytic approach is infeasible and therefore a more elaborate method is used: the expectation maximization (EM) algorithm. The EM algorithm is an iterative method for calculating maximum likelihood parameters from incomplete (or hidden) data.

Assume that each training sample contains known (or observed) data (X) and missing or unknown data (Y). The expectation step (E-step) for the EM algorithm is to evaluate:

$$\boldsymbol{Q}(\Theta,\Theta^{\boldsymbol{i}}) \equiv \boldsymbol{E}_{\boldsymbol{Y}}\left[\ln \boldsymbol{L}(\boldsymbol{X},\boldsymbol{Y}\mid\Theta)\mid \boldsymbol{X},\Theta^{\boldsymbol{i}}\right] \tag{2.7}$$

where Θ^i is the previous estimate for the distribution parameters and Θ is the variable for a new estimate describing the distribution for both the observed and hidden data. *L* is the likelihood function, which calculates the likelihood of the data, including the unknown data marginalized with respect to the current estimate of the distribution described by Θ^i. The maximization step (M-step) is to maximize $Q(\Theta, \Theta^i)$ with respect to Θ and set

$$\boldsymbol{Q}^{\boldsymbol{i}+1} \leftarrow \arg\max_{\Theta} \boldsymbol{Q}(\Theta,\Theta^{\boldsymbol{i}}) \tag{2.8}$$

The steps are repeated until a convergence criterion is met, for example:

$$\boldsymbol{Q}(\Theta^{\boldsymbol{i}+1},\Theta^{\boldsymbol{i}}) - \boldsymbol{Q}(\Theta^{\boldsymbol{i}},\Theta^{\boldsymbol{i}-1}) \le \boldsymbol{T} \tag{2.9}$$

with a suitable selected T. The EM algorithm starts from an initial guess Θ^0 for the distribution parameters and the log-likelihood is guaranteed to converge to a local or maximum. The initialization is a challenge; the selection of Θ^0 partly determines where the algorithm converges. Some solutions use a random start or a clustering algorithm such as k-Means.

The application of the EM algorithm to Gaussian mixture models is the following. The known data X is interpreted as incomplete data, and the missing part Y is the knowledge of which component produced each sample $\mathbf{x}_n$. For each $\mathbf{x}_n$ there is a binary vector $\mathbf{y}_n = \{y_{n,1},\ldots,y_{n,c}\}$, where $y_{n,c}=1$, if the sample was produced by the component c, or otherwise zero. The complete data log-likelihood becomes:

$$\ln \boldsymbol{L}(\boldsymbol{X},\boldsymbol{Y} \mid \Theta\ ) = \sum_{\boldsymbol{n}=1}^{\boldsymbol{N}} \sum_{\boldsymbol{c}=1}^{\boldsymbol{C}} \boldsymbol{y}_{\boldsymbol{n},\boldsymbol{c}} \ln(\alpha_{\boldsymbol{c}} \boldsymbol{p}(\boldsymbol{x}_{\boldsymbol{n}} \mid \boldsymbol{c},\Theta)) \tag{2.10}$$

The E-step is to compute the conditional expectation of the complete data log-likelihood (the Q-function) given the observed data X and the current estimate Θ^i. Since the complete data log-likelihood ln $L(X,Y|\Theta)$ is linear with respect to the missing Y, the conditional expectation W = E(Y|X, Θ) is computed instead and put into ln $L(X,Y|\Theta)$:

$$\boldsymbol{Q}(\Theta,\Theta^{\boldsymbol{i}}) \equiv \boldsymbol{E}\left[\ln \boldsymbol{L}(\boldsymbol{X},\boldsymbol{Y} \mid \Theta) \mid \boldsymbol{X},\Theta^{\boldsymbol{i}}\right] = \ln \boldsymbol{L}(\boldsymbol{X},\boldsymbol{W} \mid \Theta) \tag{2.11}$$

where the elements of W are defined as:

$$\mathrm{w}_{\boldsymbol{n},\boldsymbol{c}} \equiv \boldsymbol{E}\left[\boldsymbol{y}_{\boldsymbol{n},\boldsymbol{c}} \mid \boldsymbol{X},\Theta^{\boldsymbol{i}}\right] = \boldsymbol{P}(\boldsymbol{y}_{\boldsymbol{n},\boldsymbol{c}} = 1 \mid \boldsymbol{x}_{\boldsymbol{n}},\Theta^{\boldsymbol{i}}) \tag{2.12}$$

which can be written as follows by Bayes law:

$$\mathrm{w}_{\boldsymbol{n},\boldsymbol{c}} = \frac{\alpha_{\boldsymbol{c}}^{\boldsymbol{i}} \boldsymbol{p}(\boldsymbol{x}_{\boldsymbol{n}} \mid \boldsymbol{c},\Theta^{\boldsymbol{i}})}{\sum_{\boldsymbol{j}=1}^{\boldsymbol{C}} \alpha_{\boldsymbol{j}}^{\boldsymbol{i}} \boldsymbol{p}(\boldsymbol{x}_{\boldsymbol{n}} \mid \boldsymbol{j},\Theta^{\boldsymbol{i}})} \tag{2.13}$$

where α_c^i is the a priori probability of Θ^i, and $w_{n,c}$ is the a posteriori probability that $y_{n,c} = 1$ after observing x_n, i.e. $p(c|x_n)$. The probabilities $p(c|x_n)$ are also called the *responsibilities* since they measure the probability that x_n was produced by the c*th* mixture component.

Applying the M-step to the problem of estimating the distribution parameters for a C-component GMM results in the following formulas [34]:

$$\boldsymbol{\mu}_c^{i+1} = \frac{\sum_{n=1}^{N} \mathrm{w}_{n,c}\mathbf{x}_n}{\sum_{n=1}^{N} \mathrm{w}_{n,c}}$$

$$\Sigma_c^{i+1} = \frac{\sum_{n=1}^{N} \mathrm{w}_{n,c}(\boldsymbol{x}_n - \boldsymbol{\mu}_{ci})(\boldsymbol{x}_n - \boldsymbol{\mu}_c)^T}{\sum_{n=1}^{N} \mathrm{w}_{n,c}} \tag{2.14}$$

$$\alpha_c^{i+1} = \frac{1}{N}\sum_{n=1}^{N} \mathrm{w}_{n,c}$$

A potential problem with the algorithm is that it has a tendency to make very narrow Gaussians around single data points [35]. It is therefore customary to constrain the variance σ^2 to a minimum threshold, for example 10^{-3} or 10^{-6} (c.f. chapter 4.1.5). An example of this algorithm in action is shown in Figure 3, where we have trained a three component GMM on 100 sample points.

We will return to this in chapter 3.6.3 in the discussion of how to select observation features for training the GMM that represent the output probabilities.

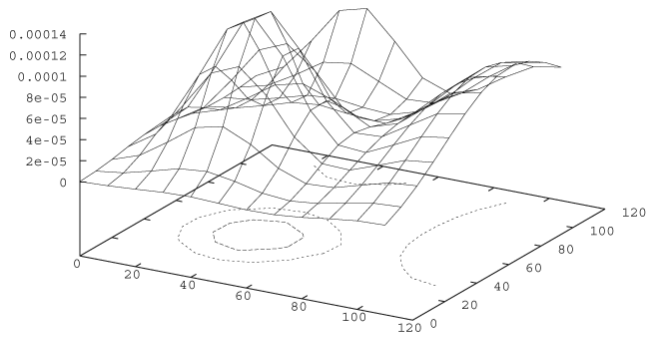


Figure 3. An example surface of a two-dimensional Gaussian mixture PDF with three components.

Estimation of the Gaussian mixture parameters for one class can be considered as unsupervised learning of the case where samples are generated by individual components of the mixture distribution and without the knowledge of which sample was generated by which component. Clustering usually tries to identify the exact components, but Gaussian mixtures can also be used as an approximation of an arbitrary distribution.

k-Means

It is common to initialize the Gaussian means using the *k-Means* algorithm, which is an example of a clustering technique (or unsupervised learning) where the training samples are not labeled but the algorithm tries to find clusters and form classes by

partitioning N data points into K disjoint subsets S_j containing N_j data points so as to minimize the sum-of-squares criterion.

$$\boldsymbol{J} = \sum_{j=1}^{K} \sum_{n \in S_j} \left| \boldsymbol{x}_n - \mu_j \right|^2 \tag{2.15}$$

where x_n is a vector representing the nth data point and μ_j is the geometric centroid of the data points in S_j. In general, the algorithm does not achieve a global minimum of J over the assignments. In fact, since the algorithm uses discrete assignment rather than a set of continuous parameters, the "minimum" it reaches cannot even be properly called a local minimum. Despite these limitations, the algorithm is used fairly frequently as a result of its ease of implementation.

The algorithm consists of a simple re-estimation procedure as follows.

1) First selected the number of sets K.
2) Assign the data points at random to the K sets, or a random point is chosen as the first centroid and then the most distant point from this is selected as the second centroid etc.
3) Then assign each of the remaining (N-k) training samples to the cluster with the nearest centroid. After each assignment, recomputed the centroid of the gaining cluster.
4) Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.
5) Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

This is a simple version of the k-means procedure. It can be viewed as a greedy algorithm for partitioning the n samples into k clusters so as to minimize the sum of the squared distances to the cluster centers. It does have some weaknesses:

- The results produced depend on the initial values for the means, and it frequently happens that suboptimal partitions are found. The standard solution is to try a number of different starting points.
- It can happen that the set of samples closest to μ_i is empty, so that μ_i cannot be updated. This is an annoyance that must be handled in an implementation, but that we shall ignore.

The results depend on the metric used to measure $\| x - \mu_i \|$. A popular solution is to normalize each variable by its standard deviation, though this is not always desirable. The results depend on the value of k.

2.4.5 Naive Bayesian Classifier

The naive Bayesian classifier assumes that the feature components are conditionally independent given the desired output value. By the definition of statistical independence we get that given the training data, the probability of observing the conjunct component values of a feature vector is just the product of the probabilities for the individual values $P(x_1,x_2,\ldots x_n|\omega_k) = \prod_i P(x_i|\omega_k)$. By substituting this into equation (2.2) we obtain:

$$\varpi = \arg\max_{\omega\in\Omega} \boldsymbol{P}(\omega)\prod_{\boldsymbol{i}} \boldsymbol{P}(\boldsymbol{x}_{\boldsymbol{i}} \mid \omega) \qquad (2.16)$$

This assumption reduces the computational complexity to the order of 2n instead of $2(2^n-1)$ where n is the number feature components. Another important difference with the naïve Bayesian classifier is that the optimal model is found without searching, but by simply counting the various data combinations with in the training examples. The number of distinct $P(x_i|\omega)$ terms that must be estimated from the training is just the number of distinct feature components (the dimensionality) times the number of distinct models.

2.4.6 Dynamic Bayesian Nets

The naïve Bayes classifier makes significant use of the assumption that the vector components are conditionally independent given the model ω. However in many cases it is too restrictive. The Bayesian belief network describes conditional independence (visualized by a directed edge) between subsets of variables (represented by nodes), i.e. independencies given a third random variable (the observed data) which is governed by a certain distribution also referred to prior knowledge. The Bayesian network in Figure 4 states that A is independent of C given B, under a given probability distribution.

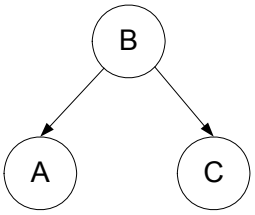


Figure 4. The Bayesian network (BN) spells out the factorization that is a simplification of the chain rule of probability.

The Bayesian network is a Graphical model where the edges are directed and point from parent to child nodes with no cycles, implicitly representing a factorization that is a simplification of the chain rule of probability:

$$\boldsymbol{p}(\boldsymbol{x}_{1:N}) = \prod_{\boldsymbol{i}} \boldsymbol{P}(\boldsymbol{x}_{\boldsymbol{i}} \,|\boldsymbol{x}_{1:\boldsymbol{i}-1}) \tag{2.17}$$

Graphical models (GMs) are a flexible statistical abstraction that has been derived from probability theory and graph theory. They provide a visual graphical formalism to depict conditional independence of natural systems and signals described by multivariate processes. They also provide procedures to reduce memory and computational demands.

2.4.7 Hidden Markov Model

Perhaps the most popular example (and simplest) of a directed graphical model (DGM) is the hidden Markov model (HMM). The one-dimensional hidden Markov model is a class of stochastic signal models which has a long history of success in various problem domains, perhaps most notably in the field of automated speech recognition (ASR).

The HMM are so named because they are composed of Markov chains that often contain hidden variables. Nodes in a graphical model can be either *observed* or *hidden*. If a variable is observed, it means that its value is known, or that data is available. In speech recognition it could correspond to utterances and for images visual observations such as color values. The observed variables are often represented by probability distributions for the feature vectors, usually modeled by a multi-component GMM (see section 2.4.4).

If a variable is hidden, it currently does not have a known value, and all that is available is the conditional distribution. Hidden variables are also called state variables and may reflect (a hypothetical) belief about the properties of the underlying process that generated the observation that is being modeled. For practical applications there is often some physical significance attached to the states; in speech they could represent a vowel or a part of a word and in images some abstract stationary property.

Figure 5 displays a one-dimensional hidden Markov model. The opaque nodes represent observed variables and the transparent nodes hidden variables.

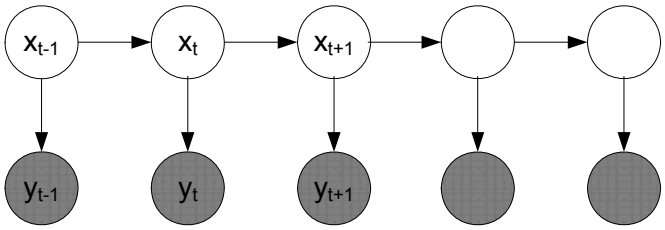


Figure 5. The one-dimensional hidden Markov model.

As mentioned, a HMM possess a Markov *chain* of hidden variables. The Markov chain has the property that, given the present, the future is conditionally independent of the past. In Figure 5 we can see that the current state variable is independent given the previous state, and that the observation is dependent only of the current state or to other observations. This is called a first-order Markov assumption: we say that the probability of a certain observation at time *t* only depends on the observation x_{t-1} at time t-1. For a second order Markovian - the current state depends on the two previous states.

2.4.8 HMM and Image Modeling

Hidden Markov models have become increasingly popular for learning purposes in such diverse applications as speech recognition [37], language modeling, language analysis, and image recognition [39]. The reason for this is that they have a rich mathematical structure and therefore can form theoretical basis for many domains. The discriminative power grows from its ability to learn the sequential evolution of the observation which can be found through the Baum-Welch's forward-backward algorithm [38] which allows estimating the numeric values of the model parameters from training data. Next we will present computationally efficient algorithms to solve the three fundamental problems of HMM design [41].

The Basic Evaluation Problem
Suppose there are N states {1,…,N} and that the probability of transition between states *i* and *j* is a_{ij}. Define o_t as the observation of the system at time *t*. According to the model this observation is governed by a probability distribution dependent only on the state at time *t* (recall Figure 5). Let $b_s(o_t)$ be the probability distribution of o_t in state *s* (also called the output probability).

If π_s is the probability of being in state *s* at time t=1 (initial state distribution), then the likelihood of observation the sequence $O = \{o_1,\ldots,o_T\}$ is obtained by summing the joint probability (the law of total probability) over all possible state sequences *s* giving the marginal:

$$P(O|\lambda) = \sum_{s_1,s_2,\ldots s_T} P(O \mid S,\lambda) P(S \mid \lambda) \qquad (2.18)$$
$$= \sum_{s_1,s_2,\ldots s_T} \pi_{s_1} b_{s_1}(o_1) a_{s_1 s_2} b_{s_2}(o_2) \ldots a_{s_{T-1},s_T} b_{s_T}(o_T)$$

where s_t is the state at position *t*.

Discrete or Continuous Observation Densities
Depending on the type observation signal the output probability distribution $b_s(o_t)$ is discrete or continuous. The observations can be characterized by discrete symbols chosen from a finite alphabet, as for instance *rain, cloudy, sunny* or the color values of a pixel, which can use a discrete probability density for each state of the model.

However many times it is necessary to use a more complex signature given by a multidimensional feature vector. Although it is possible to quantize such vector via codebooks etc (see the VQ described in 3.7.1) there might be serious degradations associated by such techniques. Hence it is desirable to be able to use continuous observation densities directly. To this end it is necessary to use a probability density function (pdf) that can be re-estimated in a consistent way, such as the mixture of Gaussian distributions (described in detail in 2.4.4):

$$\boldsymbol{b_s(O)} = \sum_{\boldsymbol{i}=1}^{\boldsymbol{M}} \alpha_i \boldsymbol{N}(\boldsymbol{x}; \mu_i, \Sigma_i) \qquad (2.19)$$

2.4.9 Reestimation Formulas

Estimation of the model parameters is usually performed with the Baum-Welch procedure which can be interpreted as an implementation of the EM algorithm [40].

The EM algorithm is usually deployed for finding the maximum-likelihood estimate of the parameters of the hidden Markov model given a set of observed feature vectors. This algorithm is also known as the Baum-Welch algorithm. We describe the complete set of HMM parameters for a given model by: $\lambda = (a_{ij}, b_s(o_t), \pi_{s)}$. As stated in [37], three fundamental problems should be solved for using HMMs:

Problem 1 - Evaluation: Estimate $P(O|\lambda)$, the probability of the observation sequence given the model parameters.

Problem 2 - Decoding: Find the state sequence $S = \{s_1,\ldots,s_T\}$ which is optimal in some meaningful sense.

Problem 3 - Learning: How to adjust the model parameters λ^* to maximize $P(O|\lambda)$?

As we will see in section 3.2 there are established methods to work out these problems. They are, in the same order; the forward-backward, Viterbi and Baum-Welch algorithms.

The calculation of the joint likelihood in (2.1) involves on the order of $2TN^T$ calculations, since at every t=1,2,…,T, there are N possible state sequences, and for each such state sequence about 2T calculations are required for each term in the sum. This computation is prohibitive in practice, even for small number of states N. To preserve a modest computational feasibility, a more efficient method is used: the Forward-Backward procedure [41]. Consider the *forward variable* $\alpha_t(i)$ defined as:

$$\alpha_t(\boldsymbol{i}) = \boldsymbol{P}(\boldsymbol{o}_1,...,\boldsymbol{o}_t,\boldsymbol{S}_t = \boldsymbol{i} \mid \lambda) \tag{2.20}$$

which is the probability of seeing the partial observation sequence $o_1,\ldots,o_t$ and ending up in state *i* at time *t*. We can effectively define $\alpha_t(i)$ recursively as follows:

1. Initialization:

$$\alpha_1(\boldsymbol{i}) = \pi_{\boldsymbol{i}}\boldsymbol{b}_{\boldsymbol{i}}(\boldsymbol{o}_1) \tag{2.21}$$

2. Recursion

$$\alpha_{t+1}(\boldsymbol{j}) = \left[\sum_{\boldsymbol{i}=1}^{N}\alpha_t(\boldsymbol{i})\boldsymbol{a}_{\boldsymbol{ij}}\right]\boldsymbol{b}_{\boldsymbol{j}}(\boldsymbol{o}_{t+1}) \quad 1\le t \le T\text{-}1 \tag{2.22}$$

3. Termination

$$\boldsymbol{P}(\boldsymbol{O} \mid \lambda) = \sum_{\boldsymbol{i}=1}^{N}\alpha_{\boldsymbol{T}}(\boldsymbol{i}) \tag{2.23}$$

The calculation of $\alpha_t(i)$ requires on the order of N^2T calculations rather than $2TN^T$.
In a similar way we can define the backward variable $\beta_t(i)$, which is the probability of the ending partial sequence $o_{t+1},\ldots,o_T$ given that we started in state *i* at time *t* [37].

With the forward-backward[4] variables we can define the probability of being in state *i* at time *t* for the state sequence *O* (also called the occupancy probability),

$$\gamma_t(\boldsymbol{i}) = \boldsymbol{P}(\boldsymbol{S}_{\boldsymbol{i}} \mid \boldsymbol{O},\lambda) = \frac{\alpha_t(\boldsymbol{i})\beta_t(\boldsymbol{i})}{\sum_{\boldsymbol{j}=1}^{N}\alpha_t(\boldsymbol{j})\beta_t(\boldsymbol{j})} \tag{2.24}$$

[4] The forward-backward variables are also amenable for the effective implementation of the Viterbi algorithm based on dynamic programming.

and the probability of being in state s_i, at time *t*, and state s_j at time t+1 given the observation sequence and model, i.e.

$$\xi_t(\boldsymbol{i},\boldsymbol{j}) = \frac{\alpha_t(\boldsymbol{i})\boldsymbol{a}_{ij}\boldsymbol{b}_j(\boldsymbol{O}_{t+1})\beta_{t+1}(\boldsymbol{j})}{\boldsymbol{P}(\boldsymbol{O}\mid\lambda)} \tag{2.25}$$

Using the above formulas we can develop the re-estimation formulas for the transition probabilities, means and covariances (a.k.a the Baum-Welch equations). The transition probabilities are computed as the ratio of the expected number of transitions from state s_i to state s_j to the expected number of times s_i is visited.

$$\boldsymbol{a}_{ij} = \frac{\sum_{t=1}^{T}\xi_t(\boldsymbol{i},\boldsymbol{j})}{\sum_{t=1}^{T}\gamma_t(\boldsymbol{i})} \tag{2.26}$$

The observation probability distribution is updated by computing the expected number of times state *i* is visited while observing *O* divided by the expected number of times in state *i*.

$$\hat{\mu}_i = \frac{\sum_{t=1}^{T}\gamma_t(\boldsymbol{i})\boldsymbol{O}_t}{\sum_{t=1}^{T}\gamma_t(\boldsymbol{i})} \tag{2.27}$$

$$\hat{\Sigma}_i = \frac{\sum_{t=1}^{T}\gamma_t(\boldsymbol{i})(\boldsymbol{O}_t-\hat{\mu}_i)(\boldsymbol{O}_t-\hat{\mu}_i)^{\boldsymbol{T}}}{\sum_{t=1}^{T}\gamma_t(\boldsymbol{i})} \tag{2.28}$$

For details see any of the references on speech recognition [42], [43].

In the next chapter we shall see that the HMM can be expanded to two dimensions and how it can be applied to image classification by *causal* reasoning or "top down" since it specifies how causes (observations) generate effects (image class labels).

2.5 2-D HMM

As we pointed out in 2.2 the HMM is a generative model, which is a great advantage since 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 [25],
- unsupervised segmentation [39],
- automatic image annotation (the ALIP system [33]),
- detect an object in a stream,
- analyze an object of known type, etc…

Statistical learning is applied to image classification in a variety of ways depending on the application. One approach is to divide the image into blocks and generate a feature vector for each block. Once the image has an adequate representation, we can deploy some classification algorithm; vector quantization, Bayes classifier, HMM, SVM or decision trees, etc. to the sequence of feature vectors, in order to generate a classifier.

However when analyzing a small region of an image, it is sometimes difficult even for a person to tell what the image is about. Hence, the drawback of context-free use of visual features is recognized up front. For most images with reasonable resolution, pixels have spatial dependencies which should be enforced during the classification. For the sake of computational simplicity, the identical independent distribution (I.I.D.) assumption is commonly used, but the relaxation comes at the cost of loosing spatial contextual constraints among pixels. In most images, if all the neighbors to a pixel belong to a class ‘A’, it is not very likely that this pixel belongs to a completely different class ‘B’ since generally images have a spatial coherence.

We can extend this argument to treat larger image regions. In an upright image depicting a *beach* for instance, one can assume that regions will appear in a predictable order: sky, sea, sand and vegetation. This natural ordering suggest the use of a top-down model, similar to the left-to-right model used in speech recognition, but here the states of the model correspond to the semantic image regions (c.f. Chapter 4) previously listed.

The HMM considers observations (e.g. feature vectors) statistically dependent on neighboring observations through transition probabilities organized in a Markov mesh, giving a dependency in two dimensions. The state process defined by this mesh is a special case of the Markov Random Field (see section 2.5.2). In the Markov random field the ordering of past, present and future in the 1-D model is replaced by spatial neighborhood.

In order to consider the neighborhood information, one approach is to estimate the joint posterior probability for the whole image's labeling configuration. To simplify the computation the Markov assumption is considered, which states that the label of a certain pixel (or block of observation) is independent to other pixel's labels given its direct neighbors.

The idea of using context information in images has given rise to algorithms based on two dimensional hidden Markov models [35], [48], [62] which are discussed. The main difficulty with applying the 2-D model to image classification is computational complexity. It has been shown that a fully connected 2-D HMM would lead to an NP-complete problem [36]. Several approximation methods have been developed to achieve computational feasibility as we will present in section 2.5.4.

2.5.1 Necessary 2-D Extensions for Image Classification

We assume that the 2-D HMM is used to predict images. An image is divided into a regular grid of blocks. A block is denoted by its position (i,j) and the complete set of blocks is $\aleph = \{ (i,j): 0 \le i < w, 0 \le j < h \}$ where *w* and *h* is the width respectively the height of the image. A feature vector o_{ij} is computed for each block (i,j) and the set of feature vectors $O = \{o_{ij} : (i,j) \in \aleph \}$ is the *vector field* describing the complete image. Under the 2-D HMM assumption this vector field is generated by the states of the model. Hence the image is classified according to the feature vectors.

A 2D-HMM is a grid of nodes, one corresponding to each block. Each node can take any of N possible states {1,2,…,N}. The state of a block (i,j) is denoted s_{ij}. Figure 6 illustrates the image blocks and the corresponding state nodes.

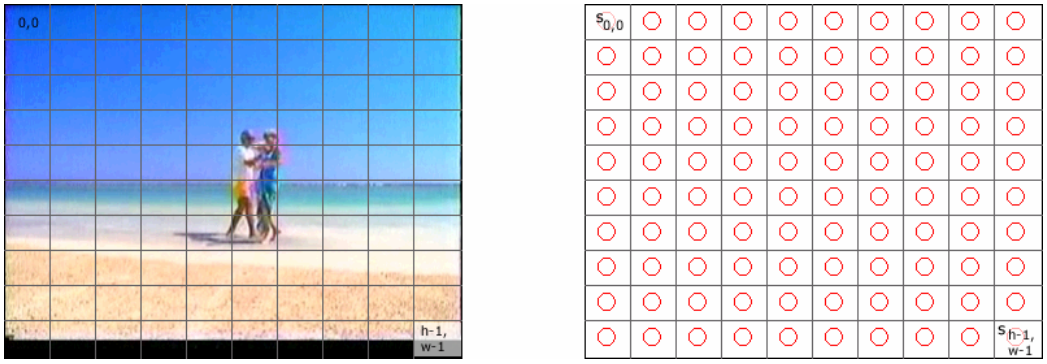


Figure 6. (a) Image decomposition into blocks, (b) states of the Markov model.

2.5.2 Markov Random Field

To extend the Markovian dependence from 1-D to 2-D a more general setting is needed. The Markov random field[5] (MRF) [55], [57], [58], [59] defines a joint probability distribution for a *neighborhood* system, which specifies nonlinear interactions between the observations of the vector field O. Given all the observations within a neighborhood of an observation, this observation is statistically independent of observations outside the neighborhood.

The basic characteristic of the chosen distribution P_x, is its decomposition as a product of factors depending on only a small set of variables. If we specify theses *local* dependencies factors we can define the joint distribution $P_x(x_1,...,x_n)$, to end up with a *global* model. With such a setup, each variable depends only directly on a few other neighboring variables. From a more global point of view all variables are mutually dependent, but only through the combination of successive local interactions.

This key concept can be formalized by the graph for which *i* and *j* are neighbors if x_i and x_j appear within a same local component of the chosen factorization. From a probabilistic point of view, this graph neatly captures Markov-type[6] conditional independencies among the random variables attached to the vertices of the graph.

The local decomposition of P_x allows us to devise an iterative algorithms, based on the common principle: at each step, consider just a few variables, all the others being "frozen". Markovian properties then imply that the computations to be done remain local, that is, they only involve neighboring variables.

However the joint probability of the non-causal MRF is not factored into the local characteristics since it is based on the conditional probability. Fortunately this can be solved by the Hamersley-Clifford theorem [60], which states that any MRF is equivalent to a GRF (Gibbs Random Field), which is defined by a joint distribution; therefore the MRF-GRF equivalence theorem allows us to represent the joint distribution X in terms of local conditional probabilities

We consider now the Gibbs distribution and its Markovian properties. Let X_i, i=1,…,n, be random variables taking values in a discrete or continuous state space Λ that forms the random vector $X = (X_1,…,X_n)^T$ with configuration set $\Omega = \Lambda^n$. Different state spaces are used depending on the application; $\Lambda = \{0,…,255\}$ for 8-bit quantized gray values and $\Lambda = \Re$ for continuous color band values. As mentioned, P_x takes a factorized form:

[5] A random field is a random object with a two dimensional index set.

[6] The Markovian property – refers to the fact that the number of previous dependencies is fixed.

$$\boldsymbol{P}_X(\boldsymbol{x}) \propto \prod_{c \in C} \boldsymbol{f}_c(\boldsymbol{x}_c) \tag{2.29}$$

where C has a partition of subsets c, the factor f_c depends only on the subset $x_c = \{x, i \in c\}$, and $\prod_c f_c$ is summable over Ω. If in addition the product is positive ($\forall x \in \Omega, P_x(x) > 0$), then (2.29) can be written in exponential form (let $V_c = -\ln f_c$):

$$\boldsymbol{P}_X(\boldsymbol{x}) = \frac{1}{\boldsymbol{Z}} \exp\{-\sum_c \boldsymbol{V}_c(\boldsymbol{x}_c)\} \tag{2.30}$$

This is, for physicists, the well known Gibbs distribution (or Boltzman) with interaction potential $\{V_c, c \in C\}$, energy function (or Hamiltonian):

$$\boldsymbol{U}(\boldsymbol{x}) = \sum_{c \in C} \boldsymbol{V}_c(\boldsymbol{x}) \tag{2.31}$$

and partition function $Z = \sum_{x \in \Omega} \exp\{-U(x)\}$. Note that the configurations of lower energies are more likely, whereas high energies correspond to low probabilities.

The dependencies induced by the factorization can be represented by an independence graph. The graph associated with $\prod_c f_c$ for example, is the **undirected** graph $G = [S,E]$, with the vertex set $S = \{1,\ldots,n\}$, and the edges (E) which are defined so that i and j are neighbors if x_i and x_j appear simultaneously in the same factor f_c. When variables are associated to the pixels of an image, the most common system of neighbors (or neighborhood system) are the regular ones where a position (not on the border) has four (se figure below) or eight neighbors.

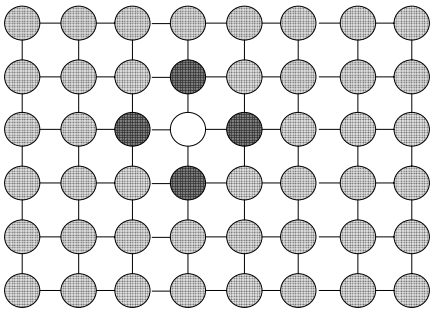


Figure 7. First-order neighborhood system for MRF-based models.

Estimating the Parameters

Learning parameters of an assumed underlying (Gibbs) distribution based on observed samples is a standard issue from statistics, and is often based on the *likelihood* (c.f. section 2.4.3) of the observed data.

We want to maximize the data likelihood $L(\theta) = \ln P^{\theta}_{Y}(y) = \ln \sum_x P^{\theta}_{XY}(x,y)$, whose derivation is intractable. But we can compute the expectation of the current fit of θ^k for the gradient of $L(\theta)$ when set to zero. In this way we can find the θ that maximizes the conditional expectation of the distribution given the data. The procedure is the Expectation-Maximization algorithm which has been introduced in section 2.4.4. The application of the plain EM-algorithm is usually intractable because of the complexity to compute the conditional expectation, and the parameter estimation of incomplete data remains in this context a tricky issue.

However, in the last few years, energy minimization approaches have had a renaissance, primarily due to new optimization algorithms such as graph cuts and loopy belief propagation (LBP) [56].

We summarize the non-causal Markov random fields as follows:

Advantages

- Isotropic behavior.
- Only local dependencies.
- Normally provide better synthesis.

Disadvantages

- Computing probability is difficult. The minimization of the energy function may be drastically time consuming, or may get stuck in local minima.
- Parameter estimation is difficult [59].
- For graphs containing cycles, loopy BP is not guaranteed to converge or be correct.

2.5.3 Markov Mesh Random Field

In this thesis we consider a special case of the Markov random field, the Markov mesh random field (MMRF). In the case of 1-D HMM the notion of past and future allowed us to develop the joint probability of states $P(S|\lambda)$ which was amenable to the forward-backward, Viterbi and Baum-Welch algorithms. The Markov mesh model reintroduces the notion of past, present and future thanks to a raster scan [57]. Let $S= \{s_{i;j}, i = 1,\ldots,w; j = 1,\ldots,h\}$ be a w x h array of states (see right panel of Figure 6) and let S_{ij} be the set of states to the left or above s_{ij}: $S_{mn} = \{s_{mn,}\ m < i$ or $n < j\}$. Then the second order Markov mesh can be defined by the following property (c.f. Figure 8).

$$\boldsymbol{P}(\boldsymbol{s}_{i,j} \mid \boldsymbol{S}_{i,j}) = \boldsymbol{P}(\boldsymbol{s}_{i,j} \mid \boldsymbol{s}_{i,j-1}, \boldsymbol{s}_{i-1,j}) \tag{2.32}$$

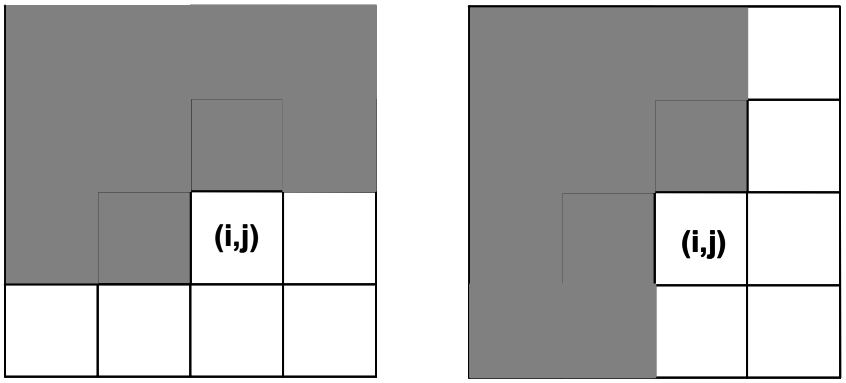


Figure 8. State dependency according to the second order Markov mesh.

Knowing the states of all the shaded blocks, we need only the states of the two adjacent blocks in the darker shade to calculate the transition probability to a next state (the order is introduced only for stating the assumptions). During the classification, blocks are not classified one by one but the algorithm tries to find the optimal combination of states jointly for all blocks.

We summarize the causal Markov mesh random fields as follows:

Advantages:

- Simple expressions for probability.
- Simple parameter estimation.
- Suited to real-time implementation.

Disadvantages:

- No natural ordering of pixels in image.
- Anisotropic model behavior.

Now we explain the causal procedure of how the image is described by the vector field O. If we know the state $s_{i,j}$ that the system occupies at each position (i,j) then the probability of observing the image can readily be computed. The 2-D HMM generates the observations in the following way:

1) the node at position 0,0 enters state $s_{0,0}$ (according to the initial probability distribution).
2) the node 0,0 emits the observation $o_{0,0}$ according to a probability distribution.
3) the system goes to node 0,1 with a probability that depends on the state chosen for node at 0,0.
4) the node at 0,1 emits the observation vector $o_{0,1}$
5) etc. for all nodes generated in raster order until node w-1, h-1.

The probability of this process is the product:

$$P(o \mid s) = P(s_{0,0}) \times P(o_{0,0} \mid s_{0,0}) \times P(s_{0,1} \mid s_{0,0}) \times P(o_{0,1} \mid s_{0,0}, s_{0,1}) \ldots \times \quad (2.33)$$
$$\times P(s_{h-1,w-1} \mid s_{0,0}, s_{0,1}, \ldots, s_{h-1,w-1}) \times P(o_{h-1,w-1} \mid s_{0,0}, s_{0,1}, \ldots s_{h-1,w-1})$$

The causality enables the extension of Viterbi and Baum-Welch algorithms to the 2-D case. However, even with the simple second-order Markovian model considered, the direct extension of these algorithms to the 2-D case is still exponential in computational complexity [61] because of the double dependency between s_{ij} and its two neighbors $s_{i-1,j}$ and $s_{i,j-1}$. Efficient approaches and approximations are therefore necessary for applications of practical value.

2.5.4 Previous Work on 2-D HMM

There's a wealth of prior related work on statistical image modeling in computer vision, image processing, and machine learning. We try to emphasize some of the work that is most related to our proposed model. The references below are to be taken as a selection, not as the complete list of work in the cited areas.

Many approaches have been proposed to extend the 1-D HMM to 2-D HMMs [44]. Among the first ones is [41] which uses a 1-D HMM to model horizontal bands of face images. A more elaborate idea is to extract 1-D features out of the image or video, and model these features by coupling one or more 1-D models [50]. The algorithm is based on projections between component HMMs and a joint HMM. Two HMMs are coupled by introducing conditional probabilities between their state variables. The resulting distribution does not satisfy the Markov property, and consequently there is no simple decomposition of the prior probability that lead to simple parameter estimation procedures. A common way to model a system with two state variables is to form a super-HMM from the product of all possible states. However this squares the number of states and training data becomes very sparse in relation to the number of states. The proposed algorithm takes the huge parameter space and embeds within it a manifold subspace which represents all possible combinations of a much smaller system of coupled HMMs (CHMM). A detailed exposition can be found in [51].

Another approach is to consider independent horizontal and vertical 1-D HMMs. For the problem of OCR Hallouli et al.[63] explored two different fusion schemes: decision fusion and data fusion. In the decision fusion scheme, the classifiers are assumed independent which enables to derive an approximation of the joint likelihood. In the data fusion scheme line and column features occurring at the same spatial index are considered correlated. 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.

PHMM (Pseudo 2-D HMM)

Agazzi et al. [52], [62] extended the 1-D HMM to a pseudo (or embedded) 2-D hidden Markov model (PHMM). The model is called "pseudo 2-D" because it is not a fully connected 2-D HMM. The assumption is that there exists a set of "super-states" that are Markovian which subsume a set of simple Markovian states. For images the superstate is decided depending on the transition probability based on the previous superstate. The superstates determine the simple Markov chain to be used by the entire row. A simple Markov chain is then used to generate observations in the row. Inherently from its structure this model is expected to perform better with structured images like documents. Since the effect of the state of a pixel on the state below is distributed across the whole row.

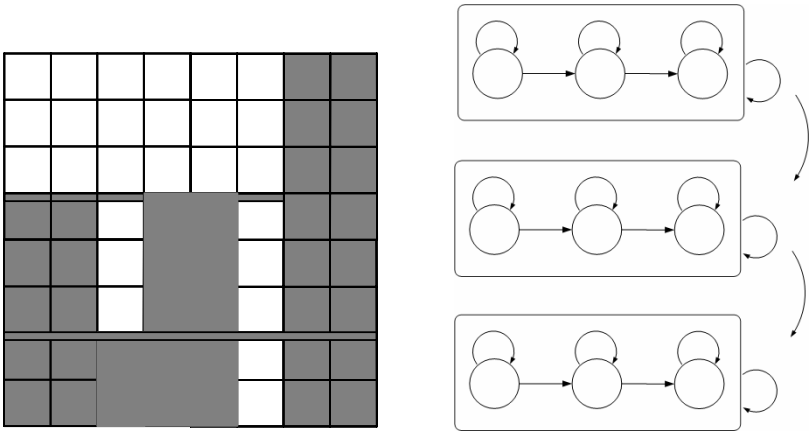


Figure 9. The observation for the word "ul" and the pseudo 2-D HMM.

Figure 9 shows an example how to represent a word by linking 1-D left-to-right models with vertical superstates. The PHMM can be specified by the parameter set: $\lambda = (N, A, \Pi, \Lambda)$ where:

1) N, is the number of superstates in the vertical direction. It is determined by the structure of the word. Consider the word "ul" in Figure 9 if we group similar rows together as indicated by the horizontal dark lines, then it is natural to assign three superstates, one for each group of lines (N=3).
2) $A = \{a_{ij} : 1 \le i,j \le N \}$ is the superstate transition probability distribution.
3) $\Pi = \{ \pi_j : 1 \le j \le N \}$, the initial superstate probability distribution.
4) $\Lambda = \{\lambda^j : 1 \le j \le N \}$, the parameters set for specifying the horizontal 1-D models within superstates. For the j^{th} superstate, λ^j consists of the number of states, the transition probabilities, observations probabilities and initial state probabilities.

The model parameters λ are estimated by using the segmental k-Means algorithm [37]. The observation sequence **O**, of each training sample is segmented into super-states sequence by determining the optimum alignment according to the current model parameters. The segmentation is achieved by a *double embedded* Viterbi

algorithm exposed in [62]. The model parameters are initialized arbitrarily and then re-estimated according to histograms of the results of the segmentation. The final step in the training loop is to test for convergence. The training procedure stops if the difference between the total accumulated likelihood of the current model and the previous falls below a threshold.

The PHMM has shown to work well with OCR problems. In applying the algorithm for machine recognition of keywords [62] an accuracy of 99% was obtained, while the conventional 1-D HMM achieved only 70% accuracy rate. Although the PHMM performs well with regular images, such as documents, since the effect of the state of a pixel on the neighboring state is distributed across the whole column, it is too constrained for classification of arbitrary images.

Path Constrained HMM

In an effort to improve image classification by context using a 2-D HMM, Li and Gray [39] proposed to use meta-states, representing state sequences on a diagonal. Thus letting a diagonal isolate the elements in the expansion of the probability of states P(s). The diagonal is chosen since it is assumed that ignoring dependencies along a diagonal intuitionally degrades performance less. The difference from the standard algorithms is that the number of possible state sequences varies depending on the length of the diagonal. In fact they grow exponentially with the number of blocks on the diagonal, so they use only the N sequences with the largest posterior probabilities. Another assumption is that the optimal sequence yields high likelihood when the blocks are treated independently.

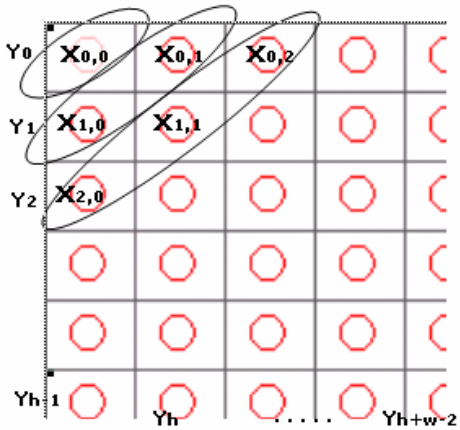


Figure 10. Subset of state configuration along diagonals.

To appreciate the idea we expose the formulas for the state probabilities here. We denote the state at each node by $X_{ij} \in \{ s_1, s_2, ..., s_n \}$ and Y_i (the meta-state) is the state sequence on diagonal *i*. Thus for the meta-states we have a 1-D Markov model:

$$\boldsymbol{Y}_o \rightarrow \boldsymbol{Y}_1 \rightarrow \boldsymbol{Y}_2 \rightarrow \ldots \tag{2.34}$$

The state variables on diagonal *i* is $\{x_{i,0}, x_{i-1,1},\ldots,x_{0,i}\}$, where (w,h) is the number of columns and rows respectively as shown in Figure 10. Now it can be shown that $P(Y_i \mid Y_{i-1},\ldots Y_0) = P(Y_i \mid Y_{i-1})$ [39]. Suppose $Y_i = \{x_{i,0}, x_{i-1,1},\ldots,x_{0,i}\}$; then $Y_{i-1} = \{x_{i-1,0}, x_{i-2,1},\ldots,x_{0,i-1}\}$ and

$$\begin{aligned} &\boldsymbol{P}(\boldsymbol{Y}_i \mid \boldsymbol{Y}_{i-1},...,\boldsymbol{Y}_0) = \boldsymbol{P}(\boldsymbol{x}_{i,0}, \boldsymbol{x}_{i-1,i},...,\boldsymbol{x}_{0,i} \mid \boldsymbol{Y}_{i-1}, \boldsymbol{Y}_{i-2},...,\boldsymbol{Y}_0) \\ &= \boldsymbol{P}(\boldsymbol{x}_{i,0} \mid \boldsymbol{Y}_{i-1},...,\boldsymbol{Y}_0) \cdot \boldsymbol{P}(\boldsymbol{x}_{i-1,1} \mid \boldsymbol{x}_{i,0}, \boldsymbol{Y}_{i-1},...,\boldsymbol{Y}_0) \\ &...\boldsymbol{P}(\boldsymbol{x}_{0,i} \mid \boldsymbol{x}_{1,i-1},...,\boldsymbol{x}_{i,0}, \boldsymbol{Y}_{i-1},...,\boldsymbol{Y}_0) \\ &= \boldsymbol{P}(\boldsymbol{x}_{i,0} \mid \boldsymbol{x}_{i-1,0}) \cdot \boldsymbol{P}(\boldsymbol{x}_{i-1,1} \mid \boldsymbol{x}_{i-2,1}\boldsymbol{x}_{i-1,0})...\boldsymbol{P}(\boldsymbol{x}_{0,i} \mid \boldsymbol{x}_{0,i-1}) \end{aligned} \tag{2.35}$$

Since all the states x_{ij} that appear in the conditions[7] are in Y_{i-1} it is concluded that

$$\boldsymbol{P}(\boldsymbol{Y}_i \mid \boldsymbol{Y}_{i-1},...,\boldsymbol{Y}_0) = \boldsymbol{P}(\boldsymbol{Y}_i \mid \boldsymbol{Y}_{i-1}) \tag{2.36}$$

Now we can expand the probability of the states $P\{s_{ij} : (i,j) \in N\}$, where $N = \{ (i,j) : 0 \le i < h, 0 \le j < w \}$ refers to all blocks in the image

$$\boldsymbol{P}\{\boldsymbol{x}_{i,j} : (\boldsymbol{i},\boldsymbol{j}) \in \boldsymbol{N}\} = \boldsymbol{P}(\boldsymbol{Y}_0) \cdot \boldsymbol{P}(\boldsymbol{Y}_1 \mid \boldsymbol{Y}_0)...\boldsymbol{P}(\boldsymbol{Y}_{\boldsymbol{h}+\boldsymbol{w}-2}\boldsymbol{Y}_{\boldsymbol{h}+\boldsymbol{w}-3}) \tag{2.37}$$

Thus we see that the sequence Y_i serves as an isolating element, which is the key for developing the algorithm.

Variable State Viterbi

Let us now see how to estimate the model parameters by a modified Viterbi training algorithm. In Viterbi training it is assumed that the single most likely state sequence accounts for practically all the likelihood of the observed data, thus searching for the sequence that maximizes $P(s \mid \mathbf{y}, \lambda)$, which is equivalent to maximizing $P\{x_{ij}, u_{ij} : (i,j) \in N\}$ (MAP rule) as we saw in section 2.4.3. By using (2.37) we can expand $P\{x_{ij}, u_{ij} : (i,j) \in N\}$ as follows:

$$\begin{aligned} &\boldsymbol{P}\{\boldsymbol{x}_{i,j}, \boldsymbol{u}_{i,j} : (\boldsymbol{i},\boldsymbol{j}) \in \boldsymbol{N}\} \\ &= \boldsymbol{P}\{\boldsymbol{x}_{i,j} : (\boldsymbol{i},\boldsymbol{j}) \in \boldsymbol{N}\} \cdot \boldsymbol{P}\{\boldsymbol{u}_{i,j} : (\boldsymbol{i},\boldsymbol{j}) \in \boldsymbol{N} \mid \boldsymbol{x}_{i,j} : (\boldsymbol{i},\boldsymbol{j}) \in \boldsymbol{N}\} \\ &= \boldsymbol{P}\{\boldsymbol{x}_{i,j} : (\boldsymbol{i},\boldsymbol{j}) \in \boldsymbol{N}\} \cdot \prod_{(\boldsymbol{i},\boldsymbol{j}) \in \boldsymbol{N}} \boldsymbol{P}(\boldsymbol{u}_{i,j} \mid \boldsymbol{x}_{i,j}) \\ &= \boldsymbol{P}(\boldsymbol{Y}_0) \cdot \boldsymbol{P}(\boldsymbol{Y}_1 \mid \boldsymbol{Y}_0) \cdot \boldsymbol{P}(\boldsymbol{Y}_2 \mid \boldsymbol{Y}_1)....\boldsymbol{P}(\boldsymbol{Y}_{\boldsymbol{h}+\boldsymbol{w}-2}\boldsymbol{Y}_{\boldsymbol{h}+\boldsymbol{w}-3}) \prod_{(\boldsymbol{i},\boldsymbol{j}) \in \boldsymbol{N}} \boldsymbol{P}(\boldsymbol{u}_{i,j} \mid \boldsymbol{x}_{i,j}) \end{aligned} \tag{2.38}$$

[7] the *past blocks* - above or to the left as described in 2.5.3.

Since Y_i isolates the elements in the expansion the Viterbi algorithm can be applied directly. The difference however from the normal Viterbi algorithm is that the number of possible state sequences at every position increases exponentially with the number of nodes on a diagonal *i*. If there are N states in the model and *v* nodes on a diagonal, then the amount of computation is in the order of N^v. This is why this version of the algorithm is called a variable-state Viterbi.

We can illustrate the number of state sequences in a state transition diagram as depicted below.

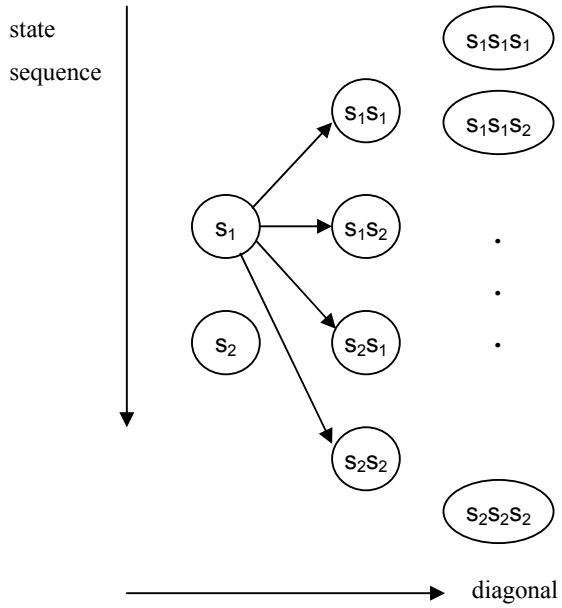


Figure 11. Viterbi state transition diagram.

To reduce computation, at each diagonal, the algorithm only uses *n* of the N^v possible state sequences. This is achieved by computing the posterior probability of a sequence of states on the diagonal as a product of the posterior probability of every block. Then the *n* sequences with the largest posterior probabilities are chosen as the *n* nodes allowed in the Viterbi transition diagram. This procedure is justified by the assumption that the optimal state sequence yields high likelihood when the blocks are treated independently. This sub-optimal version of the algorithm is referred to as the path-constrained variable-state Viterbi algorithm. (PCVS Viterbi).

We implemented the PCVS Viterbi to compare it with our model in chapter 4.2.4. A fast recursive algorithm can be developed for finding the *n* sequences with highest posterior probabilities. We don't need to compute the probabilities for all the N^v state sequences in order to find the *n* largest. Let us denote by $\gamma_{i,m}$ the log likelihood of block *i* being in state *m*, then the selection of the *n* nodes (c.f. *nodes* in the state

diagram) is to find the sequence $\{s_i: i = 1,\ldots,v\}$ with the largest $\sum_{i\in\{1,..,v\}} \gamma_{i,si}$. But we don't need to compute this sum for all sequences, we need only to find $\text{argmax}_{si}\ \gamma_{i,s_i}$ for each *i* (block) since the blocks on a diagonal are assumed to be independent.

Therefore we compute a sorted list of the highest $\gamma_{i,m}$ for each node and keep track of the *n*-sequences which gives the best solutions, and what previous state sequence that was used, for each diagonal.

To ensure that the PCVS Viterbi gives results sufficiently close to the variable-state Viterbi algorithm, the parameter *n* should be larger when there are more blocks on the diagonal. For this reason we partitioned the images into sub-images and set n to 16. See section 4.2.4 for an exposition of the application of the algorithm to multiple resolutions HMMs.

The authors of [39] claim that the algorithm perform better than other popular block-based classification algorithms such as LVQ (Learning Vector Quantization) [46] and CART (decision tree) [47].

Finally several attempts have been done to heuristically reduce the complexity of the HMM algorithms by making simplifying assumptions which approximate the real algorithms:

- ignore correlation of distant states [48],
- approximate probabilities by turbo-decoding [49],
- select a subset of state configurations only [39].

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

Chapter 3

Dependency Tree Hidden Markov Model

The 2-D HMM formalization is based on two assumptions. The first assumption made is that

$$\boldsymbol{P}(\boldsymbol{s}_{i,j} \mid \boldsymbol{s}_{i',j'}, \boldsymbol{o}_{i',j'} : (\boldsymbol{i}', \boldsymbol{j}') \in \Psi) = \boldsymbol{a}_{\boldsymbol{m},\boldsymbol{n},\boldsymbol{l}} \tag{3.1}$$

$$\text{where } \Psi = \{(\boldsymbol{i}', \boldsymbol{j}') : (\boldsymbol{i}'\, \boldsymbol{j}') < (\boldsymbol{i}, \boldsymbol{j})\}$$

$$\text{and } \boldsymbol{m} = \boldsymbol{s}_{i-1,j}, \boldsymbol{n} = \boldsymbol{s}_{i,j-1}, \boldsymbol{l} = \boldsymbol{s}_{i,j}$$

This means that the state process is first order Markovian: the probability that the system enters a particular state at position (i,j) depends only upon the states at the adjacent observations in horizontal (i-1,j) and vertical direction (i,j-1). The second assumption is that the feature vector o_{ij} is dependent only on the state at position (i,j), i.e. the observation is conditionally independent of the other blocks.

As earlier 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:

$$\begin{aligned} \boldsymbol{P}(\boldsymbol{O}, \boldsymbol{S}|\lambda) &= \boldsymbol{P}(\boldsymbol{O}|\boldsymbol{S}, \lambda)\boldsymbol{P}(\boldsymbol{S}|\lambda) \\ &= \prod_{\boldsymbol{ij}} \boldsymbol{P}\left(\boldsymbol{o}_{\boldsymbol{ij}} \middle| \boldsymbol{s}_{\boldsymbol{ij}}, \lambda\right)\boldsymbol{P}\left(\boldsymbol{s}_{\boldsymbol{ij}} \middle| \boldsymbol{s}_{\boldsymbol{i-1,j}}, \boldsymbol{s}_{\boldsymbol{i,j-1}}, \lambda\right) \end{aligned} \tag{3.2}$$

Note that the conditional probability $P(s_{i,j}|s_{i,j-a},s_{i-1,j},\lambda)$ reduces to $P(s_{1,j}|s_{1,j-a},\lambda)$ for i=1, to $P(s_{i,1}|s_{i-1,1},\lambda)$ when j=1, and to $P(s_{1,1}|\ \lambda)$ if i=j=1.

In the following section, we will present an efficient method for computing $P(O,S|\lambda)$ based on the idea of a random dependency tree.

3.1 Dependency Tree

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

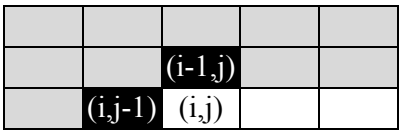


Figure 12. 2-D Neighbors.

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:

$$t(i,j) = \begin{cases} (i-1,j) & \text{with prob } 0.5 \\ (i,j-1) & \text{with prob } 0.5 \end{cases} \tag{3.3}$$

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:

$$p(s_{i,j} \mid s_{i-1,j}, s_{i,j-1}, t) = \begin{cases} p_V(s_{i,j} \mid s_{i-1,j}) & \text{if } t(i,j) = (i-1,j) \\ p_H(s_{i,j} \mid s_{i,j-1}) & \text{if } t(i,j) = (i,j-1) \end{cases} \tag{3.4}$$

If we further define a "direction" function:

$$D(t) = \begin{cases} V & \text{if } t = (i-1,j) \\ H & \text{if } t = (i,j-1) \end{cases} \tag{3.5}$$

then we have the simpler formulation:

$$P(s_{i,j} \mid s_{i-1,j}, s_{i,j-1}, t) = P_{D(t(i,j))}(s_{i,j} \mid s_{t(i,j)}) \tag{3.6}$$

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 13 shows an example of random dependency tree.

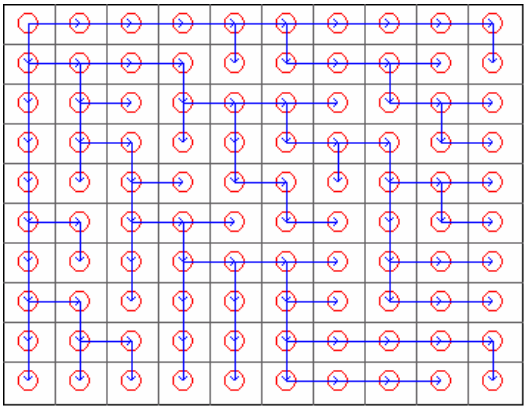


Figure 13. Example of a random dependency tree.

With this tree structure we can compute the probability of an observation produced by the model whatever the state sequence (since the states are unknown):

$$\boldsymbol{P}(\boldsymbol{o}) = \sum_{s} \boldsymbol{P}(\boldsymbol{o}, \boldsymbol{s} \mid \boldsymbol{t}) \tag{3.7}$$

and the most probable state sequence s^* that generated this output

$$\arg\max_{s} \boldsymbol{P}(\boldsymbol{o}, \boldsymbol{s} \mid \boldsymbol{t}) \tag{3.8}$$

Comments to the Anisotropy

The normal way of constructing a causal model is to impose the model constraints from left-to-right. We believe that another choice would be equivalent (although we have no proof). This was confirmed by performing the experiment in section 5 using trees with different roots, without any significant changes in the result.

Now we recall the three fundamental problems stated in 2.4.9.

3.2 Solution to Problem 1 (Evaluation Problem)

We want to compute the likelihood of the observation given the model parameters $P(O|\lambda)$ (2.1). In section 2.4.9 we introduced the forward-backward variables that enabled us to compute this joint likelihood in an efficient way 3.6.

In a similar way we define the *inside probabilities* $\beta_{ij}(s)$ as the probability that the part of the image covered by a sub-tree T(i,j) with root (i,j) is produced by the partial observation sequence and ending up in state s at position *(i,j)* (c.f. shaded portion in Figure 14).

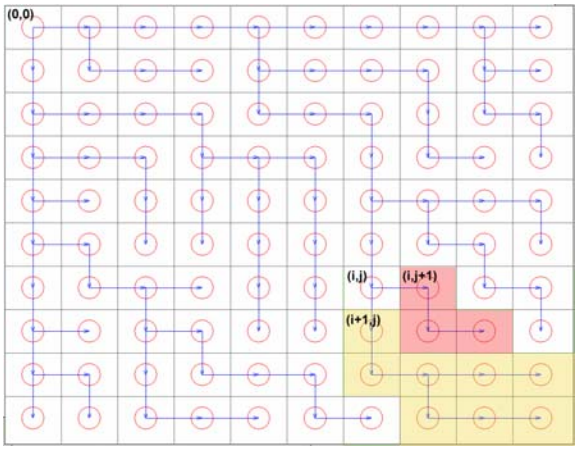


Figure 14. The inside probabilities

These values can be calculated recursively, in reverse order (starting from the last position), according to the relations:

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

$$\beta_{i,j}(s) = p(o_{i,j}|s) \tag{3.9}$$

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

$$\beta_{i,j}(s) = p(o_{i,j}|s)\sum_{s'} p_H(s'|s)\beta_{i,j+1}(s') \tag{3.10}$$

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

$$\beta_{i,j}(s) = p(o_{i,j}|s)\sum_{s'} p_V(s'|s)\beta_{i+1,j}(s') \tag{3.11}$$

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

$$\beta_{i,j}(s) = p(o_{i,j}|s)\left(\sum_{s'} p_H(s'|s)\beta_{i,j+1}(s')\right)\left(\sum_{s'} p_V(s'|s)\beta_{i+1,j}(s')\right) \tag{3.12}$$

The probability that the complete image is produced by the model is then:

$$P(O|t) = \beta_{0,0}(s_i) \tag{3.13}$$

which gives the solution to the problem. Problem 1 can be used in classification problems where we want to choose a model that best matches a test observation.

3.3 Solution to Problem 2 (Decoding Problem)

Problem 2 is the one in which we attempt to find an optimal state sequence associated with the given observation sequence. The *Viterbi algorithm* chooses the states $s_{i,j}$ that are individually most likely. This optimality criterion maximizes the expected number of correct individual states.

$$\begin{aligned} \boldsymbol{s}^* &= \arg\max_{\boldsymbol{s}} \boldsymbol{P}(\boldsymbol{s} \mid \boldsymbol{o}) \\ &= \arg\max_{\boldsymbol{s}} \frac{\boldsymbol{P}(\boldsymbol{s}, \boldsymbol{o})}{\boldsymbol{P}(\boldsymbol{o})} \\ &= \arg\max_{\boldsymbol{s}} \boldsymbol{P}(\boldsymbol{s}, \boldsymbol{o}) \end{aligned} \tag{3.14}$$

The criterion is also called *maximum a posteriori* (MAP) because it maximizes the posterior probability of s given o. The key of the algorithm, based on dynamic programming techniques, is that the object function can be computed as a sum of functions depending on the state of the previous one. This makes the amount of computation at the order of $N^2(w \times h)$ instead of $N^{(w \times h)}$ that a brute-force optimization would require.

We define T(i,j) as the sub-tree with root (i,j), and $\beta_{ij}(s)$ as the highest probability that the part of the observation covered by T(i,j) is generated starting from state s in

position (i,j). We can now compute the values of $\beta_{ij}(s)$ recursively by enumerating the positions in the reverse raster order:

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

$$\beta_{i,j}(s) = p(o_{i,j}|s) \tag{3.15}$$

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

$$\beta_{i,j}(s) = p(o_{i,j}|s) \max_{s'} p_H(s'|s)\beta_{i,j+1}(s') \tag{3.16}$$

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

$$\beta_{i,j}(s) = p(o_{i,j}|s) \max_{s'} p_V(s'|s)\beta_{i+1,j}(s') \tag{3.17}$$

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

$$\beta_{i,j}(s) = p(o_{i,j}|s) \left(\max_{s'} p_H(s'|s)\beta_{i,j+1}(s') \right) \left(\max_{s'} p_V(s'|s)\beta_{i+1,j}(s') \right) \tag{3.18}$$

To actually retrieve the state sequence, we keep track of the argument which maximized (3.14) for each position (i,j) and s. The optimal state sequence is found by backtracking over the stored best arguments. The value $\beta_{0,0}(s_i)$ is the probability of the best state sequence for the whole image, and can be used as an approximation for solving problem 1 (estimate $P(O|\lambda)$).

Note that the complexity of the algorithm is only linear in the number of positions, and that it is similar to the forward-backward algorithm, with the difference that maximization over the previous states is replaced by summations.

3.4 Solution to Problem 3 (Learning Problem)

Problem 3 is concerned with the issue of estimating the model parameters λ that maximizes $P(O|\lambda)$. In the 1-D case the Baum-Welch method (section 2.4.9) is generally used which is based on the EM algorithm. The reestimation procedure gives a local maximum solution and is called a maximum likelihood estimate of the HMM.

Here we present a variation of the Baum-Welch algorithm inspired by the inside-outside algorithm [64], [65], which is a way of re-estimating production probabilities in a probabilistic context-free grammar.

Recall that the reestimation formulas in the 1-D case (2.26), (2.27), (2.28) was defined by using the forward-backward probabilities to describe the sequence of events leading up to the probability of being in state s_i, at time t, and state s_j at time t+1. In a similar fashion we define the inside-outside probabilities to compute the joint event that the system is in state s_i at position (i,j) and state s_j at position (i,j+1) or (i+1,j). We have already introduced the inside probabilities in section 3.2.

To define the outside probabilities we denote by O(i,j) the portion of the image that is not covered by the sub-trees starting with the root at (i,j) (see Figure 15). For example, if (i,j) has two successors, O(i,j) is the portion of the image outside the sub-trees T(i+1,j) and T(i,j+1).

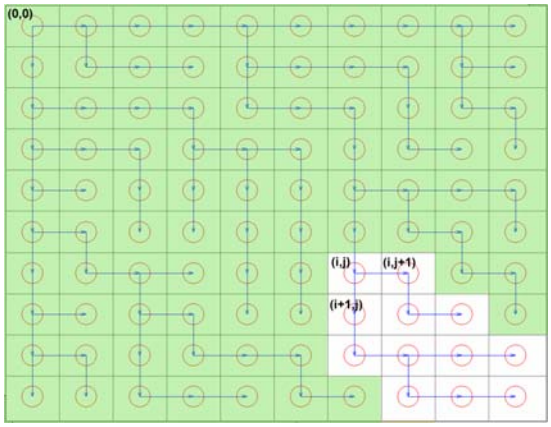


Figure 15. The outside probabilities.

If we denote by $\alpha_{i,j}(s)$ the probability of starting from position (0,0), generating the output vectors for all the positions in O(i,j), and reaching position (i,j) in state s, then the probabilities may be computed by the following recursion:

- $\alpha_{0,0}(s_i) = p(o_{0,0}|s_i)$, $\alpha_{0,0}(s) = 0$ for $s \neq s_i$,

- if (i,j+1) has an horizontal ancestor:

$$\alpha_{i,j+1}(s) = p(o_{i,j+1}|s)\sum_{s'} \alpha_{i,j}(s')p_H(s|s')\sum_{s''} p_V(s''|s')\beta_{i+1,j}(s'') \qquad (3.19)$$

(in this case, O(i,j+1) is the union of O(i,j), (i,j) and T(i+1,j)),

- if (i+1,j) has a vertical ancestor:

$$\alpha_{i+1,j}(s) = p(o_{i+1,j}|s)\sum_{s'}\alpha_{i,j}(s')p_V(s|s')\sum_{s''}p_V(s''|s')\beta_{i,j+1}(s'') \tag{3.20}$$

(in this case, O(i+1,j) is the union of O(i,j), (i,j) and T(i,j+1)). The cases where (i,j) has only one successor lead to simpler formulas:

- if (i,j+1) is the horizontal successor:

$$\alpha_{i,j+1}(s) = p(o_{i,j+1}|s)\sum_{s'}\alpha_{i,j}(s')p_H(s|s') \tag{3.21}$$

- if (i+1,j) is the vertical successor:

$$\alpha_{i+1,j}(s) = p(o_{i+1,j}|s)\sum_{s'}\alpha_{i,j}(s')p_V(s|s') \tag{3.22}$$

Thus, using the inside and outside probabilities, the probability of generating the complete image is:

$$P(O|t) = \sum_{s}\alpha_{i,j}(s)\left(\sum_{s'}p_H(s'|s)\beta_{i,j+1}(s')\right)\left(\sum_{s''}p_V(s''|s)\beta_{i+1,j}(s'')\right) \tag{3.23}$$

During the maximum likelihood training, we perform the E (expectation) step by estimating the number of times that a particular transition, or a particular emission, is used while generating the complete image. This is sometimes referred to as *occupancy probabilities* and can be written as:

$$\begin{aligned}\gamma^H_{i,j}(s) &= \sum_{s'}p_H(s'|s)\beta_{i,j+1}(s') \\ \gamma^V_{i,j}(s) &= \sum_{s''}p_V(s''|s)\beta_{i+1,j}(s'')\end{aligned} \tag{3.24}$$

Using this formulation we can write (3.23) as

$$P(O|t) = \sum_{s}\alpha_{i,j}(s)\gamma^H_{i,j}(s)\gamma^V_{i,j}(s) \tag{3.25}$$

Note that this relation is valid for all positions (i,j). Note also that if we define $\gamma^H_{i,j}(s)$ (respectively $\gamma^V_{i,j}(s)$) to be equal to 1 when (i,j) has no horizontal (respectively

vertical) successor, then the formula is valid whatever the number of successors of (i,j) is.

The probability for being in state s at position (i,j) while generating the complete image is therefore:

$$P(s_{i,j} = s|O,t) = \frac{1}{P(O|t)} \alpha_{i,j}(s)\gamma^{H}_{i,j}(s)\gamma^{V}_{i,j}(s) \tag{3.26}$$

The expected number of times that the system is in state s during the generation of the image is:

$$E(s) = \sum_{i,j} P(s_{i,j} = s|O,t) = \frac{1}{P(O|t)} \sum_{i,j} \alpha_{i,j}(s)\gamma^{H}_{i,j}(s)\gamma^{V}_{i,j}(s) \tag{3.27}$$

The probability for going from state s at position (i,j) to state s' in position (i,j+1) while generating the complete image is:

$$P(s_{i,j} = s, s_{i,j+1} = s'|O,t) = \frac{1}{P(O|t)} \alpha_{i,j}(s)\, p_H(s'|s)\, \beta_{i,j+1}(s')\gamma^{V}_{i,j}(s) \tag{3.28}$$

The expected number of times that the horizontal transition from s to s' takes place during the generation of the image is:

$$\begin{aligned} E(s \xrightarrow{H} s') &= \sum_{i,j} P(q_{i,j} = s, q_{i,j+1} = s'|O,t) \\ &= \frac{1}{P(O|t)} \sum_{i,j} \alpha_{i,j}(s) p_H(s'|s) \beta_{i,j+1}(s')\gamma^{V}_{i,j}(s) \end{aligned} \tag{3.29}$$

This provides the basis for the reestimation of the horizontal transition probabilities:

$$p'_H(s'|s) = \frac{E(s \xrightarrow{H} s')}{\sum_{s''} E(s \xrightarrow{H} s'')} \tag{3.30}$$

The vertical transitions are handled in the same way. The reestimation of the output probabilities is similar, as the probability of being in state s at (i,j) is also the probability of emitting the output vector $o_{i,j}$ from state s at position (i,j). When the output probability distribution is discrete, the re-estimate is obtained by counting:

$$p'(o \mid s) = \frac{E(s \to o)}{\sum_{o'} E(s \to o')} \tag{3.31}$$

When the distribution is continuous, the expected number of times that the emission is observed can be used to update the parameters of the distribution. In the case of a mixture of Gaussian distributions, its means, covariances and mixing coefficients, $\Theta = (\alpha_i, \mu_i, \Sigma_i)$. It can be shown [54] that the reestimation formulas for the coefficients of the mixture density take the form (example for a horizontal successor):

$$\hat{\mu} = \frac{\sum_{(i,i)\in N} \gamma^{H}_{i,j}(s) \cdot \boldsymbol{O}_{i,j}}{\sum_{(i,i)\in N} \gamma^{H}_{i,j}(s)} \tag{3.32}$$

$$\hat{\Sigma} = \frac{\sum_{(i,i)\in N} \gamma^{H}_{i,j}(s) \cdot (\boldsymbol{O}_{i,j} - \hat{\mu})(\boldsymbol{O}_{i,j} - \hat{\mu})'}{\sum_{(i,i)\in N} \gamma^{H}_{i,j}(s)} \tag{3.33}$$

In summary, the ML algorithm consists in iterating two steps over the whole set of training images:

- Compute the inside and the outside probabilities (for an image and dependency graph),
- Accumulate the expected number of occurrences for transitions and outputs.

When all images have been processed, the probabilities are re-estimated, and a new iteration on the set of images is performed. A stopping criterion is used to terminate the iterations.

Another way of estimating the model parameters is by so called *Viterbi training*. We replacing the "soft" classification reflected by $\gamma_{i,j}(s)$ by "hard" classification, $\gamma_{i,j}(s)$ = I(s^*=k), i.e., $\gamma_{i,j}(s)$ equals 1 when the optimal state sequence is in state s at (i,j), and 0 otherwise. Then we update the parameters with the same formulas as in the EM-estimation.

An approximation to the maximum likelihood training provided by the Baum-Welch algorithm is what is often termed Viterbi training, in which each observation is assumed (with weight of 1) to have resulted from the single most likely state sequence that might have caused it.

3.5 Implementation Issues for HMMs

The discussion in the previous section has primarily dealt with the theory of HMMs and its extensions to 2-D. Here we comment on some practical implementation issues including numerical problems, the choice of initial model, model size (number of states) and computational complexity.

Numerical problems
It is well known that that the implementation of the forward-backward technique obtained by a mere translation of Baum's formulas into computer programs would be distorted by underflow problems on any existing computer for all but the most trivial problems. Levinson et al. [53] have proposed a rather heuristic way to remedy the situation which consists in rescaling the forward and backward probabilities.

Recall from the definition of the forward variable

$$\alpha_t(\boldsymbol{i}) = \boldsymbol{P}(\boldsymbol{o}_1,...,\boldsymbol{o}_t, \boldsymbol{S}_t = \boldsymbol{i} \mid \lambda) \tag{3.34}$$

that each $\alpha_t(i)$ consists of a sum over a large number of terms.

$$\alpha_{t+1}(\boldsymbol{j}) = \left[\sum_{i=1}^{N} \alpha_t(\boldsymbol{i})\boldsymbol{a}_{ij}\right]\boldsymbol{b}_j(\boldsymbol{o}_{t+1}) \quad 1 \le t \le T\text{-}1 \tag{3.35}$$

Since each term a_{ij} and b_j is generally significantly less than 1, it follows that as the number of observations (t) starts to get big (e,g, 100 or more), the terms $\alpha_t(i)$ heads exponentially towards zero. For sufficiently large *t* the range of computing $\alpha_t(i)$ will exceed the precision range of virtually any computer, hence a rescaling method is needed as presented in [37], [53].

When using the Viterbi algorithm to compute the maximum likelihood state sequence we can use logarithms instead of the scaling procedure. Using the logarithms during the calculations not only solves the numerical precision problem [53] but also simplifies products to sums. Therefore we maximize the logarithm of the probability from (3.14).

$$\boldsymbol{s}^* = \arg\max_{\boldsymbol{s}} \log \boldsymbol{P}(\boldsymbol{s} \mid \boldsymbol{o}) \tag{3.36}$$

then the highest path probability $\beta_{ij}(s)$ for horizontal successor is computed as

$$\log \beta_{i,j}(\boldsymbol{s}) = \log \boldsymbol{p}(\boldsymbol{o}_{i,j} \mid \boldsymbol{s}) + \max_{\boldsymbol{s}'} \log \boldsymbol{p}_{\boldsymbol{H}}(\boldsymbol{s}' \mid \boldsymbol{s}) + \beta_{i,j+1}(\boldsymbol{s}') \tag{3.37}$$

and the corresponding for a vertical successor

$$\log \beta_{i,j}(\boldsymbol{s}) = \log \boldsymbol{p}(\boldsymbol{o}_{i,j}|\boldsymbol{s}) + \max_{\boldsymbol{s}'} \log \boldsymbol{p}_{\boldsymbol{V}}(\boldsymbol{s}'|\boldsymbol{s}) + \beta_{i+1,j}(\boldsymbol{s}') \tag{3.38}$$

This will give us log P(O|λ) instead of P(O|λ), but with less computation and no numerical problems.

How to choose initial model parameters?
The initial model depend on the choice of number of states, continuous or discrete observations, and in the case of continuous observations; the number of Gaussian mixture components. There is no theoretical way of making these choices, but they are based on the experience from working with HMMs. They depend on the nature of the physical problem, the low-level signal, the amount and nature of training data.

Computational Complexity
As we could se in section 2.4.9, the key to re-estimate the model parameters is to compute $\gamma_t(\boldsymbol{i})$ and $\xi_t(\boldsymbol{i},\boldsymbol{j})$. If we compute them directly from equation (2.24) and (2.25) we need to consider all the combination of states. Since at every position there are N possible states which can be reached and we have w x h positions, there are $N^{(w \text{ x } h)}$ state sequences. For each such state sequence there are about 2(w x h) computations for each term in the sum of (2.1), hence we have a complexity of the order $2(w \text{ x } h)N^{(w \text{ x } h)}$ calculations.

By introducing the forward-backward procedure the complexity is reduced to $o(N^2)$ (or more precise wN^{2h} calculations), which is still intensive. If we consider a modest model with 16 states and 22 x 16 observation blocks, there are on the order $22x16^{2x16} \approx 10^{39}$ calculations. The complexity of the DT HMM is linear with the number of observations as in the 1-D case $(w \text{ x } h)N^2$, since the algorithm assumes that the states depends at one neighbor at the time.

3.6 Experiment

Our first experiment was to use the dependency tree HMM as the framework for a context-dependant classifier for images. We also explored how the balance between structural information and content description affect the precision and recall by varying the size of the blocks.

3.6.1 Context Dependent Image Categorization

Conventional block-based classification is based on the labeling of individual blocks of an image, disregarding any adjacency information. When analyzing a small region

of an image, it is sometimes difficult even for a person to tell what the image is about. Therefore we are motivated to model the context between visual features. The HMM model context by considering observations statistically dependent on neighboring observations through transition probabilities organized in a Markov mesh, giving a dependency in two dimensions.

We train a set of models, each representing a semantic class using the development video archive annotated by the TRECVid 2005 participants (see section 3.6.4). To perform classification we compute the likelihood of the model $P(O|\lambda)$ to choose a model that best matches a test observation as described in section 3.2.

3.6.2 System Design

We implemented an experimental system which processes the TRECVid database to extract signature features and semantic descriptions. The results were stored in an indexed database to speed up experiments. For the classifier we implemented both a discrete and a continuous DT HMM. The discrete model uses a vector quantization (LBG-quantizer [77]) step to handle multivariate signature vectors, and the continuous model uses a GMM to describe the output probabilities. We use the modified EM algorithm to estimate the GMM and K-means to initialize the parameters. To evaluate and analyze the result we calculate a precision – recall curve and list the top 50 ranked frames. The presented results are based on the continuous model, whereas the discrete model was used for verification.

As in all block-based classification systems, an image is divided into non overlapping blocks, forming a regular grid. Feature vectors are then evaluated as statistics of the blocks, which can be regarded as a sequence of observations. The assumption in a 2-D HMM is that the observation sequence was produced by the model, i.e. $P(O \mid \lambda)$ where O is the observation sequence and λ the set of model parameters. The number of states is a fixed parameter and is set to 16 (N); each one is associated with a Gaussian mixture model to represent the continuous observation densities, which has a fixed number of components (M):

$$b_j(o) = \sum_{m=1}^{M} c_{jm} \pi\left[o, \mu_{jm}, \Sigma_{jm}\right] \quad 1 \le j \le N \tag{3.39}$$

We use the modified Baum-Welch algorithms (see section 3.4) to estimate the model parameters in the training step. To classify an image its low-level features are extracted and then $P(O \mid \lambda)$ is computed for each model giving a score on how well the model matches the observation, and then search the model with highest a posteriori probability. A general illustration of the classification system is shown in Figure 16.

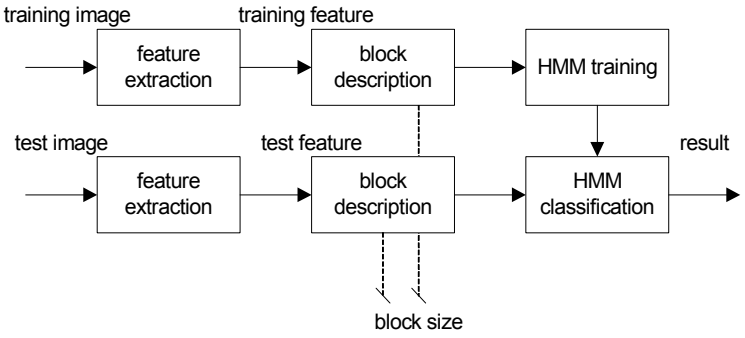


Figure 16. Image Categorization Scheme.

Since we are interested in exploring the balance between structural information and content description, we vary the size of the blocks. Using big blocks means that content description becomes more important while small blocks imply that structure becomes more prominent and content description simpler. To produce the observations for training we decompose the image into n_x by n_y non-overlapping blocks as shown in Figure 17. This gives us a vector field describing the whole picture, $O = \{o_{ij}\}$ where o_{ij} is the feature vector extracted at position (i,j). We let the block size vary in the range of 176 x 120 to 2 x 2 pixels.



Figure 17. Images decomposed into blocks with different sizes; (a) 44x40, (b) 16x15 and (c) 8x8 pixels.

3.6.3 Extracted Low-Level Features

The visual features were extracted from the decompressed key-frames in the source videos of the TRECVid 2005 development archive [66] as described below (see A.1). As aforementioned the image was split into blocks before extraction.

We designed and implemented feature extraction algorithms for color moments and DCT coefficients. In the light of the fact that we use Gaussian mixture models it was desirable to use features which are Gaussian distributed and are as much uncorrelated as possible. Further as it is well known that histogram output as features has highly skewed probability distributions and can therefore lead to ill-conditioned systems, we decided to use HSV means and variances for color descriptors and DCT coefficients for their discriminative ability of energies in the frequency domain (see A.2). Hence for each block, there are six color features and 16 DCT coefficients $\{ H_\mu, S_\mu, V_\mu, H_\sigma, S_\sigma, V_\sigma , D_{i,j \,:\, i,j} \in (0,1,2,3) \}$. The definition of the DCT coefficients is shown by

Figure 18. We pick the coefficients corresponding to low- and middle frequency response in each direction since, which are visually more discriminating.

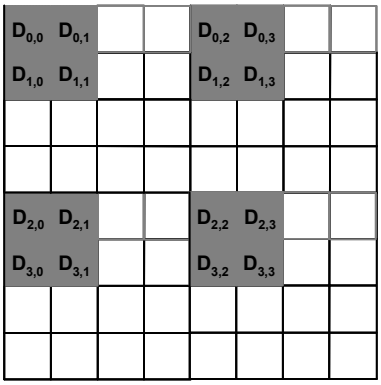


Figure 18. DCT coefficients of a 8 x 8 image block.

Since our classifier is trained over a sequence of observations; the number of dimensions is the same of that of the block (22-dimensions). With this order of dimensionality we disregard the possibility of dimension reduction.

3.6.4 The Dataset

The algorithms were tested on data from TRECVid 2004 during development, and then the experiments were carried out on TRECVid 2005. The archive consists of 133 annotated video files corresponding to 63 GB, from which we used 9473 random images. To manage the extensive amount of data and numerous files we developed a *Sample parser* (see Appendix B). Given a header file and a label, the Sample parser creates an indexed subset of frames to speed up the training and test process.

Each video has a common shot boundary reference that was provided by the TREC-Vid organization. To create the shot reference, the videos was segmented and keyframes defined according to a scheme that guarantees that a shot is at least two seconds in length. We decompress all annotated keyframes within each shot, crop it to a standard size of 352x240 pixels, and then compute image signatures on features extracted from those representative images. The classifier was jointly trained on 100 frames annotated as “Waterscape/Waterfront”. Some sample images are shown in Figure 19.



Figure 19. Images annotated “Waterscape_Waterfront” from the TRECVid 2005 archive.

During training we measure the average probability of the model for different number of Gaussians per mixture (GPM) to investigate how that affects the performance of the model. The training was performed on a Pentium 2.99 GHz PC with a LINUX operating system. The approximate time to train a model with 16 states and 22 x 16 blocks was 3 hours. The graph below shows the evolution of likelihood of the training data during the training of a model based on blocks with the size 88x60 pixels.

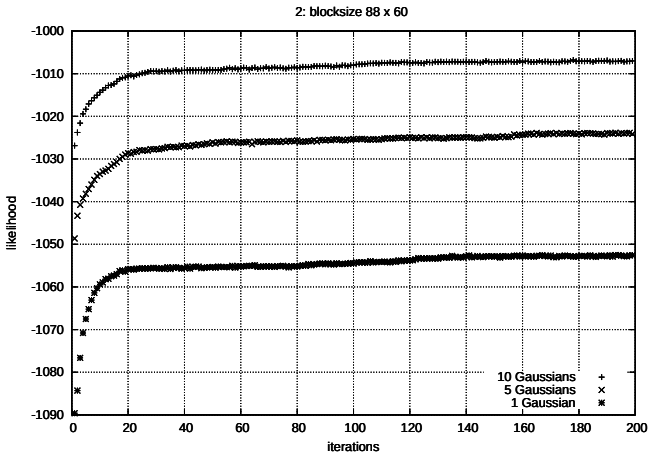


Figure 20. Likelihood of the training data for three different gpm's

We choose to use five Gaussians per mixture in our further experiments since they provide a good compromise between computational complexity and performance.

3.6.5 Results

Here we noticed the problem of one known drawback of the HMMs, that the output probability plays a more important role than the transition probability. The output distribution ranges over greater dispersion than the transition probability which range over 16 states only, with a majority of transitions from a state to itself. This explains why an image which has an almost uniform color that has a high output probability from one of the states is likely to get a very large emission probability (see Figure below).

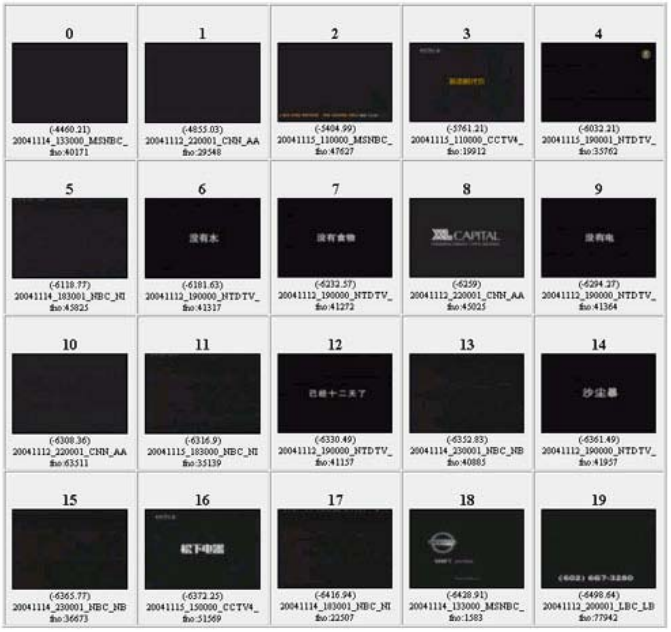


Figure 21. 20 top ranked images are uni-colored.

The first images annotated as "Waterscape/Waterfront" appears on rank 122 out of 9473 images in the test set.



Figure 22. The first true positive appears on position 122 of 9473 test images.

One possible approach to remedy this problem is to rescale the transition probabilities after the training. The *looping probabilities*[8] can be set to a predetermined maximum constant and then all the other probabilities are rescaled so that they sum up to one.

In the speech community fully connected HMM (ergodic models[9]) are rarely used since too much freedom can be difficult to train. Instead restrictions to state transitions are sometimes introduced in modeling language characteristics such as a *minimum duration constraint* [67] for example. In this experiment it appears that we have the opposite problem; instead of forcing a minimum duration of the signal we would like to minimize the maximum duration.

Instead of rescaling the transition probabilities, we suggest a modified algorithm: during the training phase, we compute the probability that a state is used in a given position p(s | (i,j)). We use this as prior probability providing some knowledge on the position of states in the image. This allows us to describe, for example, that states representing sky colors are more likely to appear in the top area of the image. We then compute the inside probabilities with prior, which are given by the formula:

$$\beta'_{i,j}(s) = p(o_{i,j}|s)p(s|(i,j))\left(\sum_{s'} p_H(s'|s)\beta'_{i,j+1}(s')\right)\left(\sum_{s'} p_V(s'|s)\beta'_{i+1,j}(s')\right) \tag{3.40}$$

This modification, to weight with prior probabilities, later shoved to increase the precision with 19%. To explore how the block-size (structural information) affects the precision, we trained seven models; each one based on observations with a different partitioning, and then the models were tested against a common test set resulting in a ranked list. The graph below shows the average precision for seven different block sizes. We noticed that a block-size of 16x15 pixels (model #4) gives the highest average precision 0.036.

[8] The probability for a state to reach itself.

[9] A model where the transition from any state to any other state is possible.

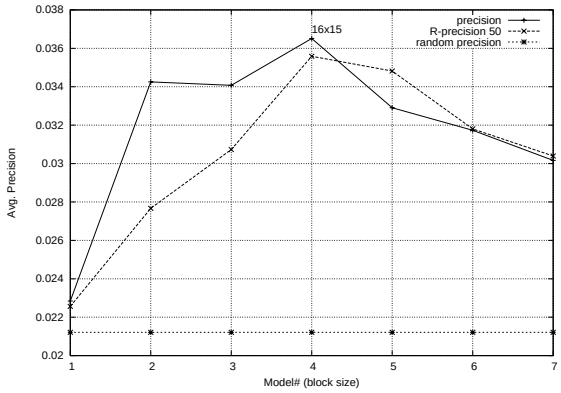


Figure 23. Avg. precision for block size: (1) 176x120, (2) 88x60, (3) 44x40, (4) 16x15, (5) 8x8, (6) 4x4, (7) 2x2

We can also see that the performance decreases rapidly with very large blocks. The explanation for this is that the averages of the color channels are not descriptive enough, and that the DCT coefficients will reflect only high frequency variations since the scale will be higher when the block-size is increased.

A solution to this problem would be to use histogram-based features, for which I can choose the number of bins (or dimensions), but then I am confronted with GMMs with spikes and very narrow variances which gives rises to singularities when training the HMMs. For this reason future experiments concerned with varying the block-size was conducted by filtering (i.e. resizing) the image first and then divide into blocks of constant size.

3.6.6 Conclusion

In an effort to demonstrate the potential of the proposed model we have applied it to the problem of image classification We have shown that the most common algorithms for solving the necessary problems associated with HMMs, can be adapted to DT HMM, which allows reestimation of the HMM parameters in the same linear complexity as in the one dimensional HMM case.

To investigate the performance of the model and to find its point of operation, we have studied the importance of number of Gaussians per mixture during training and the effect of varying the block size. The results indicate that during the training process a model with a larger number of Gaussians per mixture (GPM) needs more iteration to reach an asymptote which happens after about 30 iterations. A model trained with five GPMs achieves the best performance with a block size of 16x15 pixels.

Using the modified algorithm in (3.40), with the prior probabilities during training, the precision increased with 19%, but the results are still not comparable with the

rates typically observed in the TREC video experiments, however it lead us to a greater understanding of the model and gave inspiration to new experiments and refinements that will be discussed in the following chapters.

3.7 Combination with a Global Model

As we could see in the previous experiment, the result was somewhat *shadowed* by uni-colored images. In the case of speech recognition it can be compared with trying to recognize the word "aaaaaa". Since our classification model is a generative one, this kind of observations becomes a part of the model, and can therefore not be distinguished from the rest.

Driven by the desire to investigate the models behavior on *normal* non-synthetic images we introduce a model based on global characteristics that we combine with the DT HMM in order to sift away the uni-colored images.

The global model uses a histogram to represent the frequency of global observations over the whole training set. Since each observation is an n-dimensional vector we first *quantize* them by using a vector quantizer.

3.7.1 Vector Quantization

Vector quantization is generalization of scalar quantization to multiple dimensions. The idea dates back to Shannon [89] in his development of the information rate of a source subject to a fidelity criterion.
A vector quantizer maps n-dimensional vectors in the vector space $\Re^n$ into a finite set of vectors $Y = \{y_i: i = 1, 2, ..., n\}$. Each vector y_i is called a *code vector* or a code-word and the set of all the codewords is called a codebook. Associated with each codeword, y_i, is a nearest neighbor region called Voronoi region, and it is defined by:

$$\boldsymbol{V}_i = \left\{ \boldsymbol{x} \in \Re^n : \left\| \boldsymbol{x} - \boldsymbol{y}_i \right\| \le \left\| \boldsymbol{x} - \boldsymbol{y}_j \right\|, \forall \boldsymbol{i} \ne \boldsymbol{j} \right\} \tag{3.41}$$

As an example we take vectors in the three dimensional case without loss of generality. Figure 24 shows some vectors in space. Associated with each cluster of vectors is a representative codeword. Each codeword resides in its own Voronoi region. These regions are separated with imaginary planes in the figure for illustration. Given an input vector, the codeword that is chosen to represent it is the one in the same Voronoi region [91].

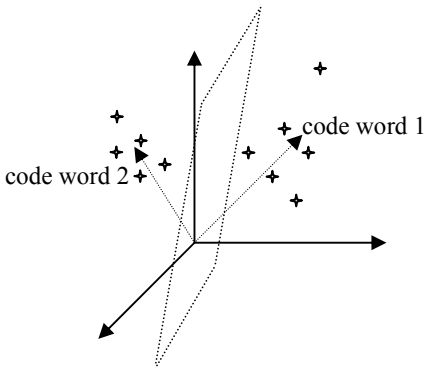


Figure 24. Codewords in three dimensional space. Input vectors are marked as a cross and codewords as an arrow.

The problem of designing a codebook that best represents a set of input vectors is NP-hard (Nondeterministic in Polynomial time). This means an exhaustive search that grows exponentially with the number of codewords, where each potential solution can be verified in polynomial time. I therefore resort to suboptimal codebook design scheme called LBG for Linde-Buzo-Gray, the authors of the method [91]. This algorithm is similar to the k-means algorithm.

The algorithm
1) Determine the number of codewords *n* (the size of the codebook).

2) Make an initial partition of the code vectors by selecting an input vector at random, and then iteratively chose the vector as far away as possible from the selected one [90].

3) Use the Euclidean distance measure to clusterize the vectors around each codevector. This is done by taking each input vector and finding the Euclidean distance between it and each codeword. The input vector belongs to the cluster of the codeword that yields the minimum distance.

4) Compute the new set of codewords. This is done by obtaining the average of each cluster. Add the component of each vector and divide by the number of vectors in the cluster

$$\boldsymbol{y}_i = \frac{1}{\boldsymbol{m}} \sum_{j=1}^{\boldsymbol{m}} \boldsymbol{x}_{ij} \tag{3.42}$$

where *i* is the component of each vector (x, y, z, ... dimension), m is the number of vectors in the cluster. Steps 2 and 3 are repeated until either the codewords don't change or the change in the codewords is below a certain threshold.

3.7.2 Global Model

We let a histogram represent the global characteristic of the training set. By computing a code cook for the training data according to the algorithm above, we can express the number of observations associated with each code word in a histogram.



Figure 25. Global LBG Histogram.

For a test image we compute its histogram based on the training-codebook by finding the closest code words for each observation vector and then forming a histogram based on the frequencies of the code word indices.

To compare the test image with the training set with need a similarity measure for their histograms. Swain and Ballard introduced a histogram matching method called Histogram Intersection [94]. Given a pair of histograms, $H(I_1)$ and $H(I_2)$, of images I_1 and I_2 respectively, each containing n bins, they define the histogram intersection of the normalized histograms as:

$$\boldsymbol{H}(\boldsymbol{I}_1) \cap \boldsymbol{H}(\boldsymbol{I}_2) = \sum_{j=1}^{n} \min(\boldsymbol{H}_j(\boldsymbol{I}_1), \boldsymbol{H}_j(\boldsymbol{I}_2)) \tag{3.43}$$

where $H_j(I)$ represents the frequency of color *j* in image I. For two images, the larger the value of the histogram intersection, the more similar the image pair is considered to be. The L_1 (Manhattan) and L_2 (Euclidean) distances measures are commonly used when comparing two feature vectors. In practice, the L_1 distance measure performs better than the L_2 distance measure because the former is statistically more robust to outliers [93]. In this experiment I use the L_1 distance measure for comparing histograms because it is simple and robust [95].

3.7.3 Results

The desire is to combine the global model with the DT HMM so that we obtain a model that takes into account both local and global properties. Several schemes of

combination and fusion can be considered to combine the models but is beyond the scope of the objective here. We resorted to a combination by taking the product of the two probabilities $P^{combined} = P^{global} * P^{DTHMM}$. To study the behavior of the different models we first applied them separately on a small test set of 100 samples. Table 2 shows the average precision for each model.

Table 2. Average precision on large test set.

	Average Precision
Global model	0,104577
DT HMM	0,137663
Combined model	0,146427

We the performed an experiment using a larger test set comprised of 4700 images from the TRECVid 2003 development archive [66]. The test set has 40 *true positives* (classified as "Beach") giving a random precision of 0.0083. Figure 26 shows the recall precision for the concept *Beach* using the combined model.

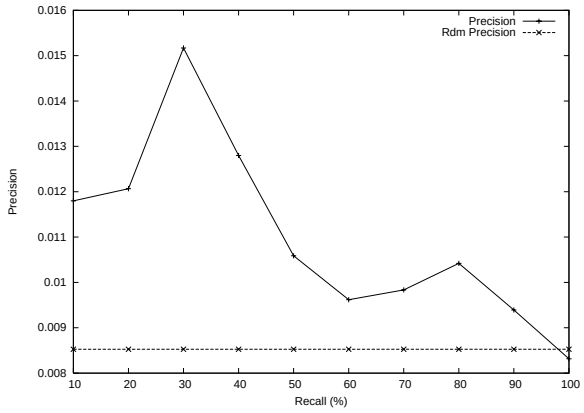


Figure 26. Recall / Precision for Beach

We can see in the graph that the precision peak is shifted to the right which can be explained by that there is still the effect of images with large uni-colored regions that get a high probability, meanwhile they also comply with the global models requirements. This is supported by the figure below where we can see a mix of images coming from the global model which match in terms of global characteristics e.g. images: 0,1,4,6,7 and images that comply with the DT HMM which meet with local characteristics: 2,3,5,8. In image 2 and 3 the model believes that the white bottle is a person, since there are a lot of light people-objects in the middle of the image in the training data.

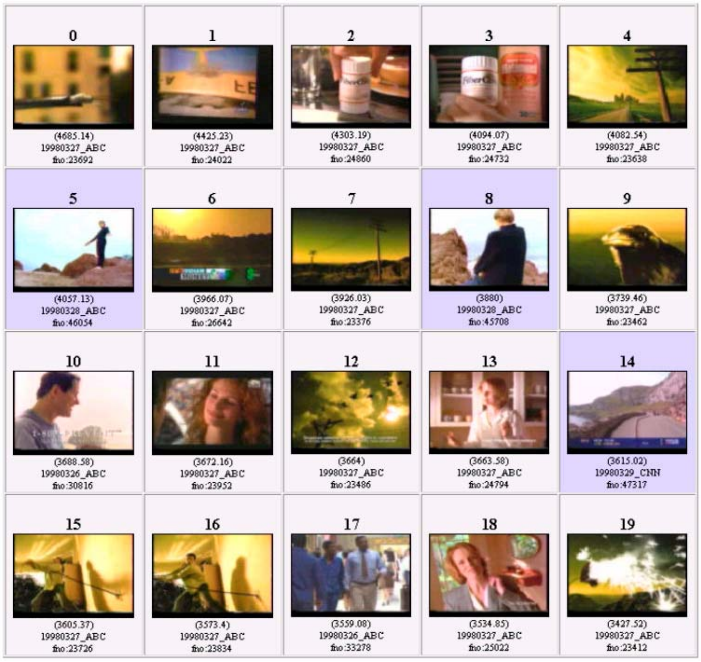


Figure 27. 20 top ranked images using a combination of the global model and DT HMM.

We give the top ranked images by using only the DT HMM for comparison:



Figure 28. 20 top ranked images using the DT HMM.

By using the combined model, the classification performance increased to 0.045, from 0,013 (average precision rate in the TREC workshop for the *beach* concept was 0.017). Showing a three fold increase indicates that global information may be used to improve the quality of the HMM model

3.7.4 Conclusion

To be able to study our contextual model we sifted out uni-colored images by using a global model. It was confirmed that big output probabilities degrade the result, but also that the contextual model sometimes fails to discriminate sub-classes with-in the concept which might suggest the scale is too low. At one hand we need high scale to distinguish fine details in objects, and low scale to capture global properties. This gives rise to the idea of introducing restrictions to the states in order to enforce sub-classes which can be given interpretations such as sky, sea, sand, person, boat etc. Naturally it also motivated us to form a multiscale model which we present in chapter 4.

3.8 Influence of the Dependency Tree

The DT HMM avoids the complexity of regular 2D HMM by changing the double horizontal and vertical spatial dependencies into a random uni-directional dependency, either horizontal or vertical. A question that arises is what impact does this random choice have on the model? We therefore explore various issues about the effect of the random tree.

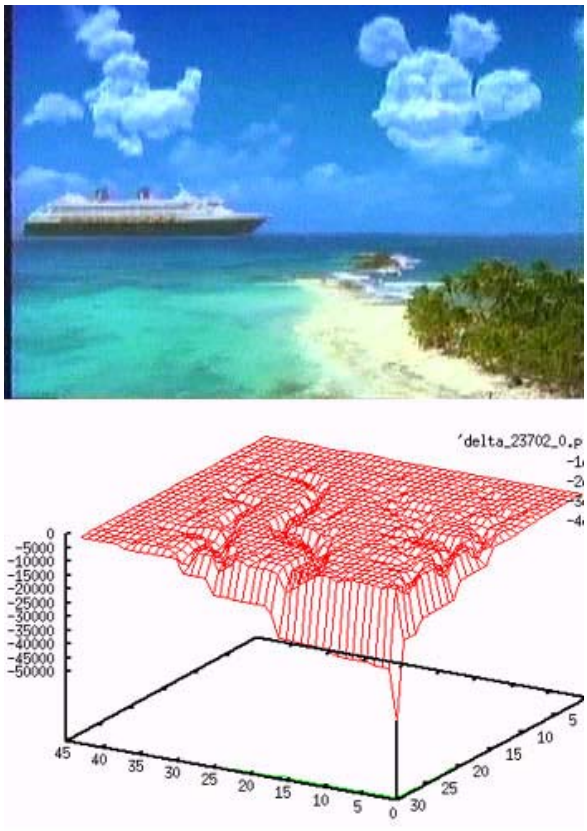


Figure 29. The Joint probability of the observation given the model, and the tree, at position (i,j).

3.8.1 DT HMM Probability Estimation

According to the model, the exact probability of an observation is:

$$P(O) = \sum_{t} P(O|t)P(t) \tag{3.44}$$

We may assume that all dependency trees are equally likely, so that the distribution P(t) is uniform. As there are $2^{(m-1)(n-1)}$ different trees for an image of m x n blocks, the complete computation is prohibitive. Therefore, it is important to search for approximations of this value which are easy to compute. In this chapter we investigate three different ways of doing this estimation by: unique sampling (P^u), tree average (P^a) and dual tree (P^d). The section below gives a detailed study of their qualities.

The experiments are conducted using a 36-state model trained on a consistent set of 130 *Beach* images annotated at the pixel level into 8 different sub classes. We divide the images in 22x16 blocks that are represented by color moments and DCT coefficients. Evaluation is done by computing P(O|t) (also called the score), using the modified Viterbi algorithm, for 100 random images, with different set of trees, to investigate the behavior of the dependency.
We give an example below to illustrate the state alignment in one image for two different trees.

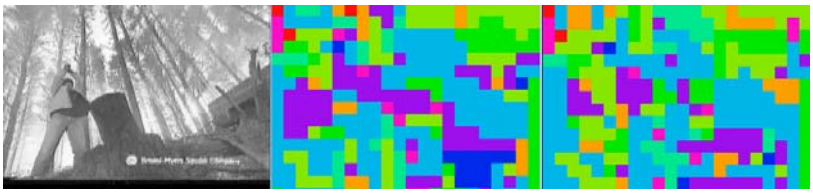


Figure 30. State alignment with two different trees.

3.8.2 Estimation by Average

The best approximation (in terms of quality) that we can do is to compute an average over as many trees as possible: consider a set of trees $T_k=\{t_1, t_2, \ldots t_k\}$ and compute the estimate as:

$$P_k^a(O) = \frac{1}{k}\sum_{i=1}^{k} P(O|t_i) \tag{3.45}$$

As k gets larger, the value of $P_k^a(O)$ should converge towards the optimal value. The graph in Figure 31 shows the magnitude of the relative error E(k) computed as the relative difference of this score to the average over the largest number of trees (here chosen to be 50), i.e.:

$$E(k) = \left| \frac{\frac{1}{50}\sum_{i=1}^{50} P(O|t_i) - \frac{1}{k}\sum_{i=1}^{k} P(O|t_i)}{\frac{1}{50}\sum_{i=1}^{50} P(O|t_i)} \right| \tag{3.46}$$

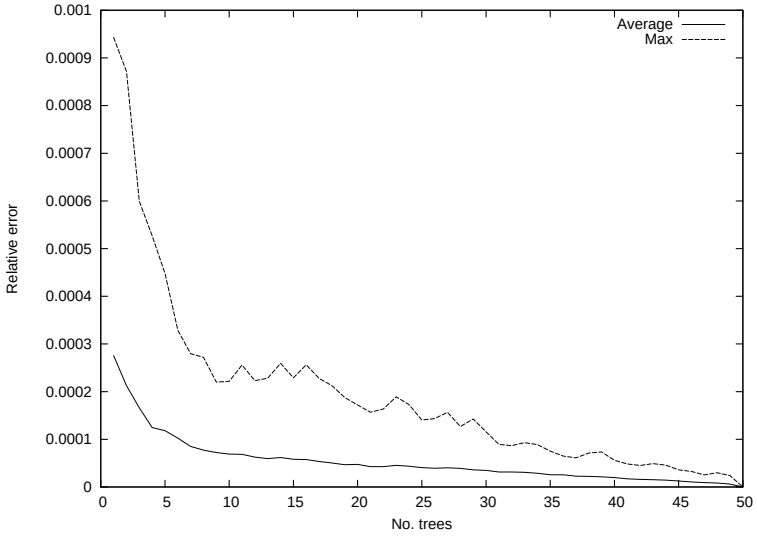


Figure 31. Convergence of probability average.

We can see that with 10 trees or more, the average relative error is below 0.01% while the maximum relative error remains below 0.03%.

3.8.3 Estimation by unique sampling

As the computation of the probability with a lot of different trees can be computationally expensive, we would like a faster approximation of this probability. The fastest way is to use a single tree,

$$P^u(O) = P(O|t) \tag{3.47}$$

To evaluate the quality of the approximation, we compute the relative difference between an estimation and the mean over all 50 trees, and we compute the histograms of these values for different estimations (P^u, P^a_{15}, P^d).

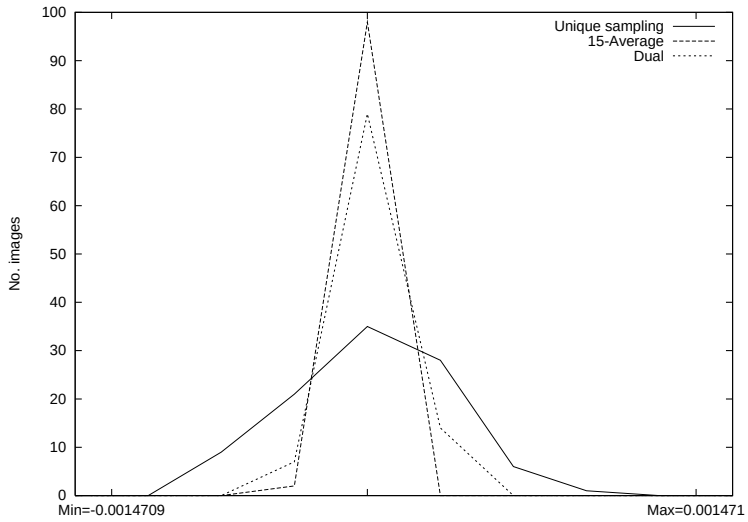


Figure 32. Distribution of scores.

P^d is the dual estimation and is explained below. We can see that 90% of the values for P_{15}^a are within a range of ± 0.015%. By comparison P^u has only 63% of its values with in the same range. For a confidence interval of 90%, P^u has a variation of ± 0.060%.

3.8.4 Estimation with dual tree

The dual t^T of a tree t is defined as the tree where the horizontal dependencies are replaced by vertical dependencies and vice versa (except for the border nodes for which the ancestor is unique).

As the couple (t, t^T) contains all the possible vertical and horizontal dependencies, it is interesting to consider the estimation by the average probability between the tree t and its dual t^T

$$P^d(O) = \frac{1}{2}\left(P(O|t) + P(O|t^T)\right) \tag{3.48}$$

where t^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.

As indicated in Figure 32 the dual estimation gives a better approximation than the unique sampling. The dual estimation has 82% of its samples in the previously mentioned interval with limits ± 0.015%. For a confidence interval of 90%, P^d has a variation of ± 0.022%.

3.8.5 Conclusion

Discontinuity
The DT HMM considers one neighbour at the time which is not sufficient to give a robust model. A theoretically sound model requires that all possible trees would be considered. Since this is not possible, we have done an experiment with many trees (yielding a multiple-tree model). The advantage of the multiple-tree model is that it has a low computational complexity while being sound modelled (without discontinuities).

We have investigated the variation of the observation probability with respect to the particular dependency tree that is used. The results show that the divergence between P^u and P^a is small, i.e. that there are great chances that a random tree will provide a value which is close to the average.

We have also compared three different approximations of the exact probability (3.44) which results may suggest the choice of the best approximation method, based on the deviation which is considered reasonable for a given application.

What are we modeling?
The DT HMM is an attempt to combine information of blocks in the image with their neighbors by imposing constraints to the side-by-side relation. We impose local constraints by considering neighboring blocks in an effort toremove ambiguous context and improve the performance.

Our objective is to model relative relationship between observations which can be part of larger semantic regions such as sky or mountain. The semantic regions depend on the concept we try to learn, the size of the blocks and the model, as we will se in the next chapter.

.

Chapter 4

Advanced 2-D Applications

To further investigate the model we introduce *interpretations* to the states of the DT HMM by associating a sub-class to partitions of states. The sub-classes model the variability of the visual features representing an image region. They are therefore amenable for abstracting the later to semantic regions. By training a new model with restricted states we can perform semantic image segmentation on unseen but in-class images

To this end we need to segment the images in the training data and manually annotate the resulting image regions. Further we need to introduce a mechanism in the training phase that restricts the possible states depending on corresponding sub-class of the image patch.

4.1 Semantic Image Segmentation

Segmentation is the operation concerned with partitioning images by determining disjoint and homogeneous regions or, equivalently, by finding edges or boundaries. The homogeneous regions or the edges are supposed to correspond to actual objects, or parts of them, within the images. It plays an important role in many areas of image processing and computer vision as the first step before applying higher level operations such as recognition and semantic interpretation [74]. These systems are also of key importance for the new content based applications like object-based image and video compression for which among others the standards MPEG-4 and MPEG-7 are developed.

Semantic segmentation can be said to emulate the cognitive task performed by the human visual system (HVS) to decide what one "sees", and relies on a priori assumptions. In this chapter, we investigate how this prior information can be modeled by

learning the local and global context in images by using our multidimensional hidden Markov model.

4.1.1 Related Work

A number of researches have introduced systems for mapping users' perception of semantic concepts to low-level feature values [68], [69]. Carson et al. at Berkeley introduced a so-called "blobworld" [89], in where the image representation provides a transformation from the raw pixel data to a small set of image regions (blobs) which are coherent in color and texture space. The "blobworld" representation is based on segmentation using the Expectation-Maximization algorithm on combined color and texture features.

In an effort to perform unsupervised texture segmentation, a multiple resolution algorithm was used in [73]. The algorithm first segments the image at coarse resolution and proceeds to progressively finer resolutions until individual pixels are classified. At each resolution a greedy algorithm is used to perform the classification, and then the result is used as an initial condition at the next finer resolution.

The probabilistic framework of multijects (multi-objects) and multinets by Naphade and Huang [25] map high level concepts to low level audiovisual features by integrating multiple modalities and infer unobservable concepts based on observable by a probabilistic network (multinet). The Stanford SIMPLIcity system [53] uses a scalable method for indexing and retrieving images based on region segmentation. A statistical classification is done to group images into rough categories, which potentially enhances retrieval by permitting semantically adaptive search methods and by narrowing down the searching range in a data-base.

Li and Gray [39] proposed a 2D-HMM for image segmentation. Their experiments showed that the algorithm performs better than other popular block-based classification algorithms such as LVQ [46] and CART [47]. An attempt in associating semantics with image features was done by Barnard and Forsyth at University of California at Berkeley [72]. In recent work by Kumar and Hebert at Carnegie Mellon University [75], a hierarchical framework is presented to exploit contextual information at several levels. The authors argue that the system encodes both short- and long-range dependencies among pixels respectively regions, and that it is general enough to be applied to different domains of labeling and object detection.

4.1.2 Semantic Segmentation

Semantic segmentation relies on domain-specific a priori knowledge about the concept at hand, which is not easy to generalize. We therefore restrict the problem to one concept; *beach* images in a top-down fashion.

4.1.3 States with semantic labels

Naturally the modeling is based on the multidimensional DT HMM. Since semantic video regions do not usually have invariant visual properties, we divide the image class into a number of sub-classes (or regions). Every sub-class is assigned a set of states to allow for a flexible representation. Suppose there are K sub-classes $\{1,\ldots,K\}$ and that observation vector o_{ij} belongs to a region annotated with sub-class c_k, then its set of permissible states is $\{s(k)\}$. The table below lists the sub-classes and their designated set of states.

Table 3. The set of states for each sub-class.

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

Due to this restriction of states we introduce a flexible but controlled representation for the observation vectors. The sub-class *sky* can for instance have states corresponding to observations with the colors: saturated blue, white, dark gray or yellow and with different texture properties such as regular, striped or grainy.

The miscellaneous (or mixed) class is used for areas that are ambiguous or contain video graphics etc. (see Figure 33). Ambiguous areas are patches which contain several sub-classes or which are difficult to interpret. Observations from this class are not a part of the model since they are considered as noise.

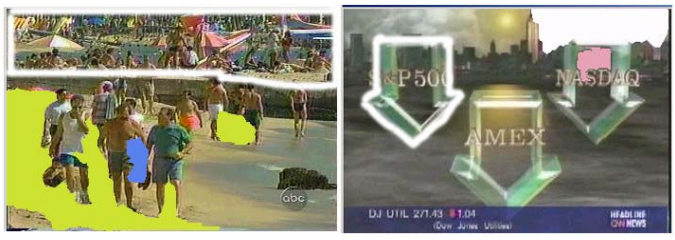


Figure 33. Example of images with mixed class areas.

Annotations was done in practice by first segmenting the training images into arbitrary shaped regions using the algorithm proposed in [86] and then manually label each region on the pixel level (352x240 pixels) with one of the sub classes by using an application with a graphical user interface as shown below.



Figure 34. Annotating an image segment as "sky".

4.1.4 Model Training

The training process is as follows; extract the image signatures and train a model using the restricted states. For the image to be segmented; extract its low-level features and compute the maximum posterior probability of the state sequence given the model as given by the Viterbi algorithm 3.3.

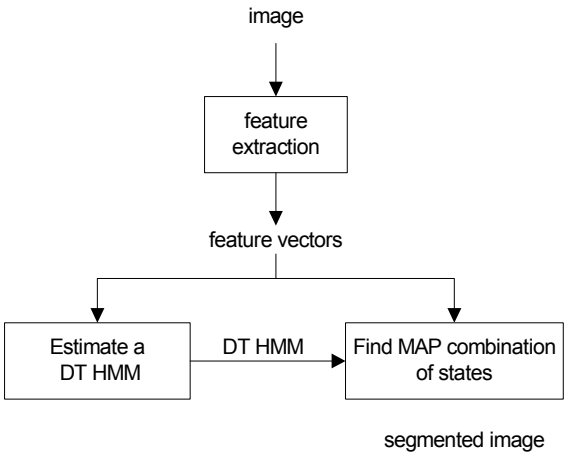


Figure 35. Image segmentation schema.

We used the algorithms presented in section 3.4 for Viterbi training to fit the model. 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, by:

$$p_H(s'|s,t) = \frac{N_{H,t}(s,s')}{N(s)} \tag{4.1}$$

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, to allow for transitions that have not been seen during training.

The output probabilities may be also estimated by relative frequency in a similar way. This provides a solution to estimate the parameters of the model from a set of labeled training examples, and is called Viterbi training as before mentioned. 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.

The training was conducted on the TRECVid archive [66], from which we selected a heterogeneous image collection of 130 images depicting "Beach" (see Figure 36).



Figure 36. 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 well-known LUV (CIE LUV color space)[10] 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. Those images have been manually segmented and annotated, so that every feature vector is annotated with a sub-class.

To define the initial output probabilities, a GMM (Gaussian Mixture Model) is trained with the feature vectors corresponding to each sub-class. We allow three GMM components for every state, so the GMM for the sub-class sky has 21 components and for vegetation (c.f. Table 3). Then we group the components into as many clusters as there are states for this sub-class (using the k-means algorithm). Finally, the GMM model for each state is built by doubling the weight of the components of the corresponding cluster in the GMM of the sub-class. The transition probabilities are initialized uniformly. Then, during training we iterate the following steps:

[10] We started to use the LUV color space since it has been shown to perform better in similar experiments [39]. It is used in calculation of small color differences, and it is often chosen because of its good perception correlation properties (see section A.2).

- We generate a random dependency tree and perform a Viterbi alignment to generate a new labeling of the image. The Viterbi training procedure is modified to consider only states $\{s(k)\}$ that correspond to the annotated sub-class (c_k) at each position, thus constraining the possible states for the observations (the manual annotation specifies the sub-class for each feature vector, but not the state).
- We re-estimate the output and transition probabilities by relative frequencies (4.1) (emission of an observation by a state, horizontal and vertical successors of a state) with Lagrange smoothing.

4.1.5 Experiment

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 36 above provide the following assignments:



Figure 37. State segmentation after 0, 2, 6 and 10 iterations.

The sequence in Figure 37 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 shape after 10 iterations.

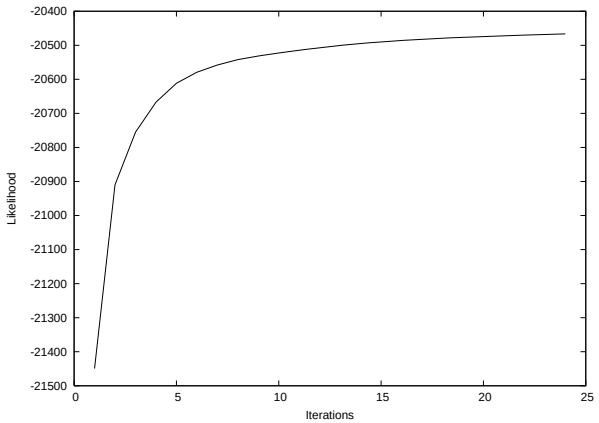


Figure 38. Likelihood of the training data.

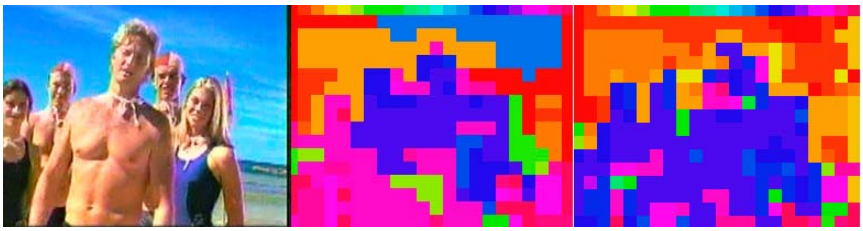
Once the model is trained, we can apply it on new images. Below is an example of the state assignment for an image in the test set; 72% of the blocks are correctly classified.



Figure 39. 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 belong to one of the 8 sub classes). The definition of those states has been generated while taking into account all training data simultaneously, and provides a model for the variability of the visual evidence of each sub-class.

During training, we impose a minimum variance for the Gaussian distributions, in order to avoid degeneracy. This minimum has an impact, as we noted that the number of correct labeled blocks in the example above increased to 72% when changing the minimum variance from 10^{-6} to 10^{-10}. An explanation for this is that if the selected minimum variance is too high, some Gaussians will be flattened out and collides with Gaussians from states representing similar observations.



(a) original image b) min. variance 10^{-6} (c) min. variance 10^{-10}

Figure 40. Better performance with smaller minimum variance for GMMs.

Sometimes the result is degraded because of visually ambiguous regions, as in the examples below (video effects or sky reflection on the sea). Because the output probabilities of model have generally a greater dynamic range than the transition probabilities, they often play the major contribution in the choice of the best state assignment.

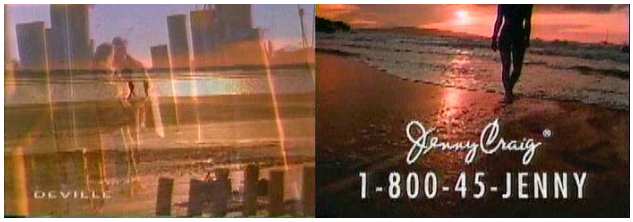


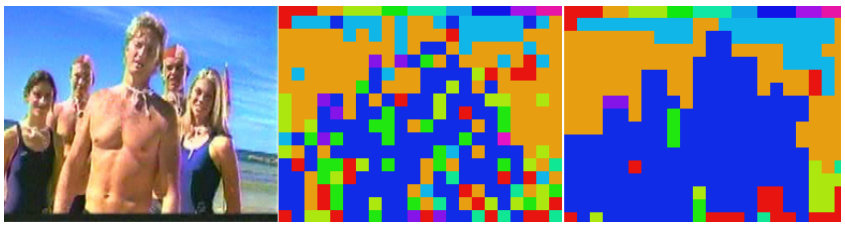
Figure 41. Test images with ambiguous regions.

Comparison with the Mixture Model

Still, to show the effect of transition probabilities, we compared the model with a Gaussian mixture model. We can view a mixture model as a special hidden Markov model with the underlying state process being i.i.d (identical independent distribution), i.e. a reduced Markov chain where the transition probabilities are uniform.

The experiment was carried out by semantically segment 40 test images. We compare the best state assignment obtained by the Viterbi algorithm (this takes into account both output and transition probabilities) with the assignment where each feature vector is assigned the state which has the highest output probability. The experiment was performed on a Pentium 2.99 GHz PC with a LINUX operating system. The approximate time to classify an image was 2 s.

The average rate of correctly labeled blocks was 38% when taking transition probabilities into account and 32% with only the output probabilities. Figure 42 shows an example, with the original example image, the sub-class assignment without transition probabilities (56% blocks correctly labeled), and the Viterbi assignment (72% correct).



(a) (b) (c)

Figure 42. Labeling without/with transition probabilities.

4.1.6 Conclusion

In this section we have demonstrated the application of the DT HMM to semantic segmentation of an image. We show how the model can be trained on manually segmented data, and used for labeling new test data. In particular, we use a modified version of the Viterbi algorithm that is able to handle the situation when a visual sub-class is represented by several states, and only the sub-class annotation (not the state annotation) is available. We investigated several properties of this process. The motivation for this approach is that it can be easily extended to a larger number of classes and sub-classes, provided that training data is available. Allowing several

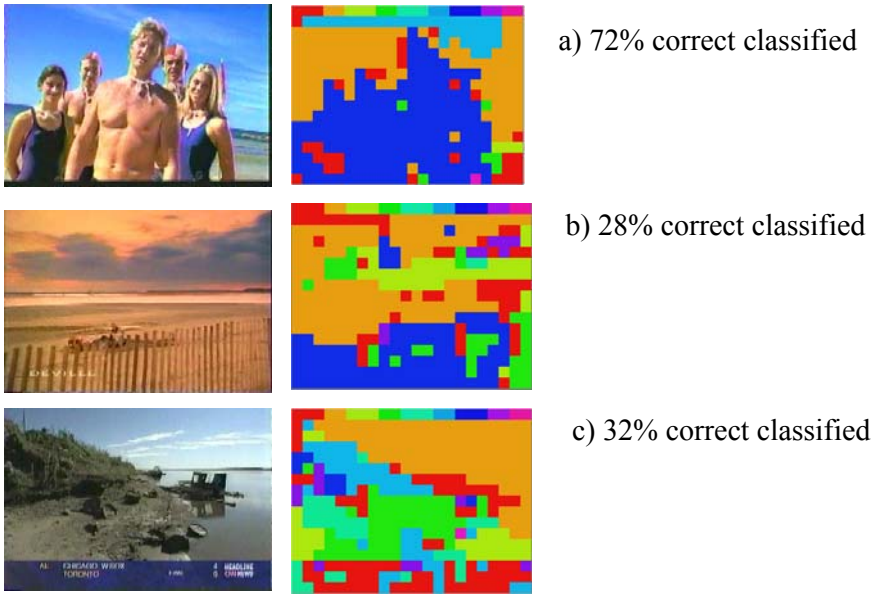
states per sub-class gives the model the flexibility to adapt to sub-classes which may have various visual observations.

The confusion table below shows the number of classified blocks for each class. The *miscellaneous* class has zero blocks since it was excluded during the test phase. We can see that *sky* is sometimes confused with *sand* (because of reflections, as in Figure 43 b), and *sea* with *sand* (because they overlap) and *mountain* with *sand* (because of similar descriptors). The class vegetation, building and boat are weak because of small training sets.

Table 4. Confusion Table.

	Annotated								
Classified	misc	sky	sea	sand	mount	veg	pers	build	boat
misc	0	0	0	0	0	0	0	0	0
sky	199	1470	636	228	59	4	115	50	21
sea	382	136	1090	314	168	28	89	165	66
sand	245	677	573	1008	452	24	181	152	51
mountain	101	66	120	86	73	14	31	94	24
vegetation	84	84	42	183	98	95	17	45	7
person	305	260	287	601	271	145	601	196	128
building	54	11	62	90	24	7	33	57	20
boat	2	15	2	2	0	0	1	1	2

We could observe that the average classification rate was degraded by non consistent images in the test set and by confusion of visually similar observations from different sub-classes (see examples below). To cater for this problem we plan to extend the model to multiple resolutions in order to introduce global context. We also investigated to what extent the transition probabilities in the hidden Markov model improves the classification rate.



a) 72% correct classified

b) 28% correct classified

c) 32% correct classified

Figure 43. Example of segmented images.

Comments on the Concept *beach*
This dataset is particularly challenging due to wide within-class variance in the appearance of the data, further the category *beach* proves to be difficult to classify. It has one of the lowest average precision rates in the TREC workshop (0.017). The explanation for this can be that the combination of a broad semantic definition and very varying lightening conditions (c.f. Figure 36) makes the category hard to discriminate. Apart from the problem of inconsistency, there were images in the training set that were consistent in the sense of subsequent sub-classes. For example *sky* followed by *sea* and then *sand*, but then the *sky* was reflected in the *sea*, which would affect the transition probabilities (c.f. Figure 41).

4.2 Multiresolution Hidden Markov Model

It has been shown that the human visual system (HVS) uses both global and local information to make decisions of what an image is depicting. Ambiguous regions are examined more closely while the major entities can be located by a brief glance; hence the effort is unevenly distributed [78].

Multiresolution analysis on images has long been an active research field. The desire is to extract important details and context at different resolutions and then combine the results to form a powerful image model.

Global context information can be introduced by a joint decision procedure for the image on several resolutions. Several classification algorithms based on multiresolution modeling have been developed [79], [80], [81], [82]. In this chapter we extend the 2-D HMM described in section 3.1 to multiple resolutions, which we shall refer to as DT MHMM. We demonstrate the application of this novel model to multiple scale analysis in the context of image classification. To investigate the quality of the model we compare it to another known block-based multiresolution model [79] (see section 2.5.4).

4.2.1 Previous Work on 2-D MHMM

Many multiscale models have been developed to represent statistical dependence among image pixels, with wide range of applications in image restoration, segmentation, classification, etc. In recent work by Kumar and Hebert at Carnegie Mellon University [75], a two-layer hierarchical framework is presented to exploit contextual information at several levels. Each layer is modeled as a *conditional random field* (CRF) [76] that allows the capture of arbitrary observations-dependent label interactions.

The two layers are coupled with directed links. A node in layer 1 may represent a single pixel or a patch while a node in layer 2 represents a larger homogeneous region or a whole object. The nodes in the two layers are connected to its neighbors through undirected links. In addition, each node in layer 2 is also connected to multiple nodes in layer 1 through directed links. In this way the model encodes both local context (e.g., pixel-wise label smoothing) as well as global interactions (e.g., relative configurations of objects or regions) in a tractable manner[11].

Bouman and Shapiro [81] proposed the multiscale random field (MSRF) models for images. Assume that an image is described by a random field X, and the pixel labels (or classes) at resolution r are $C^{(r)}$, r = 1,…,R. Then the first assumption of the MSRF is that the Markov property holds across resolutions, i.e.

$$\begin{aligned} &\boldsymbol{P}(\boldsymbol{C}^{(r)} = \boldsymbol{c}^{(r)} \mid \boldsymbol{C}^{(l)} = \boldsymbol{c}^{(l)}, \boldsymbol{l} < \boldsymbol{r}) \\ &= \boldsymbol{P}(\boldsymbol{C}^{(r)} = \boldsymbol{c}^{(r)} \mid \boldsymbol{C}^{(r-1)} = \boldsymbol{c}^{(r-1)}) \end{aligned} \tag{4.2}$$

the second assumption is that X only depends on $C^{(r)}$:

$$\begin{aligned} &\boldsymbol{P}(\boldsymbol{X} \in \boldsymbol{dx} \mid \boldsymbol{C}^{(r)} = \boldsymbol{c}^{(r)}, \boldsymbol{r} = 1,..., \boldsymbol{R}) \\ &= \boldsymbol{P}(\boldsymbol{X} \in \boldsymbol{dx} \mid \boldsymbol{C}^{(R)} = \boldsymbol{c}^{(R)}) \end{aligned} \tag{4.3}$$

During block classification, the model follows two restrictions. First the classes in $C^{(r)}$ are conditionally independent given the class in $C^{(r-1)}$. Second, every class in $C^{(r)}$ depends only on classes in a neighborhood at the coarser resolution r-1.

A difference from our approach is that features are observed only at the highest resolution, while the coarser resolutions are only involved in prior probabilities of classes.

Hidden Markov Tree

The hidden Markov tree (HMT) is a wavelet-domain hidden Markov model which connects the state variables vertically across scale. They are designed for the intrinsic properties of the wavelet transform and provide powerful signal models for denoising and prediction [83].

Transform-domain models are based on the idea that often a linear, invertible transform will restructure an image, resulting in transform coefficients whose structure is simpler to model. Most grayscale images are well characterized by their singularity (i.e. edge and ridge) structure. The wavelet transform provides a powerful tool for modeling singularity-rich images [88]. It can be interpreted as a multiscale edge

[11] The use of directed links between the two layers, instead of the undirected ones, avoids the intractability of dealing with a large partition function.

detector that represents the singularity content of an image at multiple resolutions and orientations. Wavelets covering a singularity field yield large coefficients; wavelets overlying a smooth region yield small coefficients.

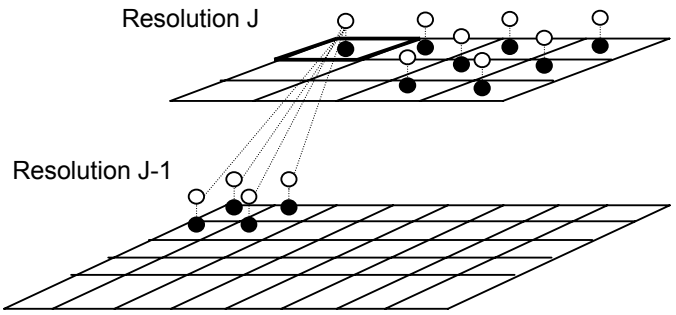


Figure 44. Quad-tree view of the parent-child dependencies of the 2-D wavelet.

Four wavelets at a given scale nest inside one at the next coarser scale, which gives rise to a quad-tree structure of wavelet coefficients (c.f. Figure 44). Image singularities will manifest themselves as cascades of large wavelet coefficients through scale along the branches of the quad-tree [88].

The HMT associates with each wavelet coefficient a hidden state variable that controls whether it is “large” or “small”. The marginal density of each coefficient is modeled as a two-component Gaussian mixture, using a large variance Gaussian for the large state and a small variance Gaussian for the small state. The HMT captures the cross-scale persistence of large/small coefficients by Markov chains between the hidden states across scale. This multiscale singularity characterization makes the HTM appropriate for modeling texture images.

The model proposed in [83] is based on the HMT. Wavelet coefficients across resolutions are assumed to be generated by one-dimensional hidden Markov models with resolution dependency instead of time as in the Markov chain. If we view wavelet coefficients as a special case of features, the model considers features observed at different resolutions. The approach is extended to general features in [84].

As we shall see in section 4.2.2, the difference between our work (together with [79], [81]) and the approaches in [83], [85], [84] is that the intra-scale dependencies are not regarded. Our model regard an image as an instance of a 2-D random process characterized by one model, which reflects the transition properties among classes/states at all resolutions as well as the dependence of feature vectors. The set of states with the maximum *a posteriori* probability is calculated according to the model.

Further the states in wavelet-domain HMMs are not related to classes as in our case, but "large" and "small" wavelet coefficient. To classify the blocks of an image, a separate HMM is trained for each class. A local region (or block) is regarded as an instance of a random process described by one of the HMMs. To decide the class of the block, the likelihood is computed for each HMM, and the class yielding the maximum likelihood is selected. Consequently the spatial dependencies between classes are not accounted for in those models.

A two dimensional multiresolution model (2-D MHMM) that incorporates both intra-scale and inter-scale dependencies was introduced by Jia Li et Al in [79]. The 2-D MHMM forms a hierarchical structure by introducing constraints to states depending on states at the lower resolution level. A block at a lower resolution, called a parent block, has a number blocks at the same spatial position at a higher resolution, which are defined as its child blocks (see Figure 45).

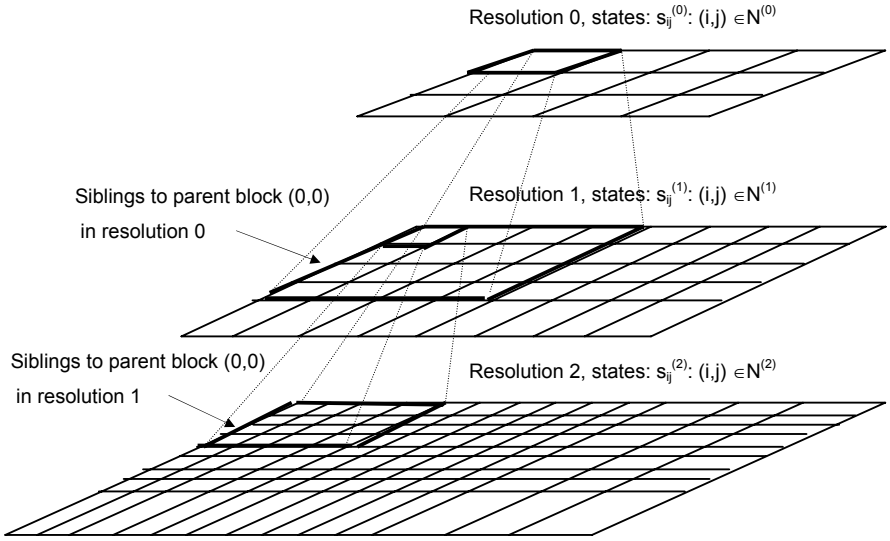


Figure 45. Hierarchical Multiresolution Model.

The inter-scale dependencies are modeled by a Markov chain. Given the states and features at the parent resolution, the states and features at the current resolution are conditionally independent of the other previous resolutions. If we denote the state of block (i,j) as s_{ij}, with the observation u_{ij} and the collection of resolutions $\Re$ = {1,…,R}, with r=R being the highest resolution we have;

$$
\begin{aligned}
&P\{s_{i,j}^{(r)}, u_{i,j}^{(r)} : r \in \Re, (i,j) \in \mathrm{N}^{(r)}\} = P\{s_{i,j}^{(1)}, u_{i,j}^{(1)} : (i,j) \in \mathrm{N}^{(1)}\} \\
&\times P\{s_{i,j}^{(2)}, u_{i,j}^{(2)} : (i,j) \in \mathrm{N}^{(2)} \mid s_{k,l}^{(1)} : (k,l) \in \mathrm{N}^{(1)}\} \\
&\times \ldots \times P\{s_{i,j}^{(R)}, u_{i,j}^{(R)} : (i,j) \in \mathrm{N}^{(R)} \mid s_{k,l}^{(R-1)} : (k,l) \in \mathrm{N}^{R-1)}\}
\end{aligned}
\tag{4.4}
$$

Intrascale dependencies are modeled by 2-D HMMs. The observations at the coarsest resolution are assumed to be generated by a single 2-D HMM, at all higher resolutions sibling blocks belonging to the same parent (see Figure 45) are assumed to be generated by a 2-D HMM. But the HMMs vary according to the state of the parent block, thus for each resolution with M states there are M HMMs in the next resolution, except for the highest resolution.

To retain a modest computational complexity the image at each resolution is divided into sub-images and a computational efficient variant of the Viterbi alignment is deployed: the *variable-state Viterbi* [39] (see also section 2.5.4). The algorithm isolates state dependencies along diagonals in the sub-image. Then only the n-best state sequences on a diagonal are considered when computing the most probable path. A fast algorithm is also proposed by relaxing the maximum a posteriori (MAP) classification rule. First at the lowest resolution

$$\{s_{i,j}^{(0)}, u_{i,j}^{(0)} : (i,j) \in N^{(0)}\} \tag{4.5}$$

is search to maximize

$$\log P\{s_{i,j}^{(0)}, u_{i,j}^{(0)} : (i,j) \in N^{(0)}\} \tag{4.6}$$

The procedure is repeated for the next higher resolution; given the states at resolution 0, the child blocks at resolution 1 are governed by a single 2-D HMM. So

$$\log P\{s_{i,j}^{(1)}, u_{i,j}^{(1)} : (i,j) \in D(k,l) \mid s_{k,l}^{(0)}\} \tag{4.7}$$

is maximized. Where D(k,l) denotes the child blocks of block (k,l) at the previous resolution.

4.2.2 DT MHMM

The purpose of multi-resolution analysis is to capture information in an image at different resolutions. Therefore we are lead to consider linear combinations of 2D-HMMs trained at different resolutions. Compared to hierarchical modeling this approach is straight forward to construct. We simply have to individually train a number of 2-D HMMs at different resolutions and then combine them by interpolation or by joint likelihood. The architecture of the multi model system is depicted in Figure 46.

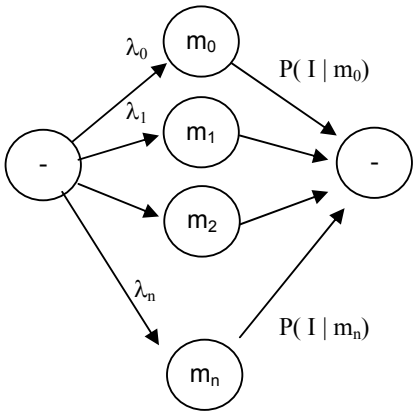


Figure 46. Linear combination of multiple resolution models.

4.2.3 Algorithms

As before mentioned we investigate two different combination schemes for our multiresolution model; interpolation and joint combination. The interpolated model is computed as a weighted sum of the probabilities from the individual models $m_j\, j \in \{1,..M\}$, where M is the number of models, given the observation:

$$P^I(I|m) = \sum_j \lambda_j P(I|m_j) \tag{4.8}$$

The weights $\lambda_j\ j \in \{1,2,\ldots M\}$ are found by expectation maximization (EM) estimation. We compute the maximum a posterior likelihood for the combined model given a held-out training set I_i: $i=\{1,2,\ldots T\}$.

$$\arg\max_{\lambda} L(\lambda) = \prod_i P(I_i|m) = \prod_i \sum_j \lambda_j P(I_i|m_j) \tag{4.9}$$

The iteration formula for λ is given by computing the expected relative contribution for each model m_j:

$$\lambda_j^{t+1} = \frac{1}{T}\sum_i^T \frac{\lambda_j^t P(I_i|m_j)}{\sum_k \lambda_k^t P(I_i|m_k)} \tag{4.10}$$

The initial value of λ_j is set uniformly to 1/M. The joint model is expressed as a product of the probability of the observation given the model for each resolution:

$$P^J(I|m) = \prod_j P(I|m_j) \tag{4.11}$$

4.2.4 Experiment

We compare the models using the same image classification scenario as in Chapter 3. An illustration of the classification scheme is shown in Figure 47. The models are trained using 130 annotated training images from the TrecVid archive, and then classification is performed with 100 arbitrary images which results in a rank list.

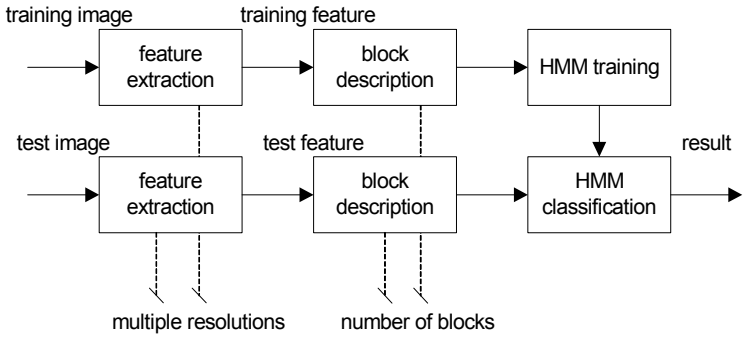


Figure 47. Multiresolution classification scheme.

In the first step we extract multi-resolution block-based feature vectors. The size of the blocks is 4x4 pixels which is the same as in the experiments presented by Jia Li et Al. [39]. As in the first experiment (3.6) we use Gaussian Mixture Models as out-put probabilities and variances and means for color descriptors, except that we use the LUV color coding instead of HSV for its good perception correlation properties (see section A.2). Again means and variances of DCT coefficients are used to describe structural properties. Hence for each block, there are six color features and 6 DCT coefficients $\{ L_\mu, U_\mu, V_\mu, L_\sigma, U_\sigma, V_\sigma , D_{i,j \,:\, i,j} \in (0,1,2,3) \}$. Since our classifier is trained over a sequence of observations; the number of dimensions is the same of that of the block (12-dimensions).

Low-level features are extracted to produce a three-level model with a 4x4 split between parent- and child blocks (c.f. Figure 45). By iteratively resizing the image while keeping the block size constant we obtain the resolutions (a) 4x3, (b) 16x12 and (c) 64x48 blocks, as illustrated below.

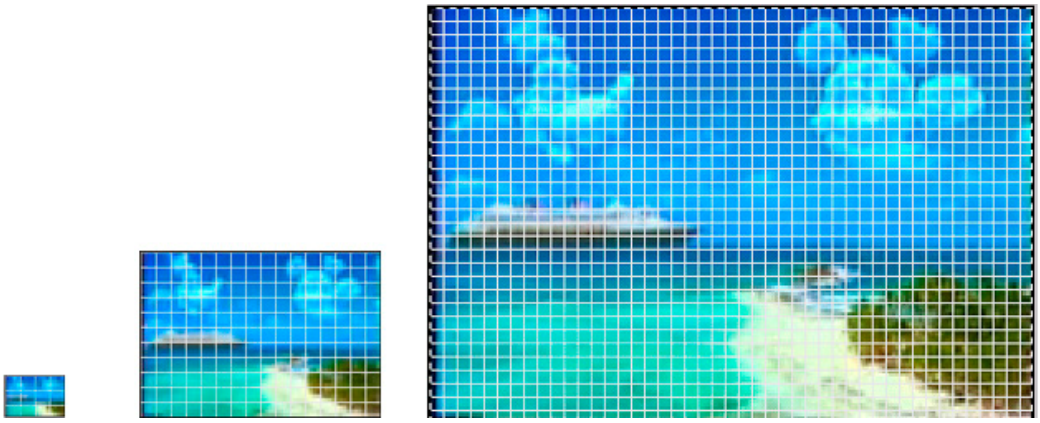


Figure 48. Three resolutions with constant block size: (a) 4x3, (b) 16x12 and (c) 64x48 blocks.

In this experiment we used a DT MHMM model with: 9, 16, 16 states for resolution: 0, 1 and 2 respectively. The HMMs are initiated with an alignment where one state covers a region of a regular 3x3 grid (or 4x4 for 16 states) over the image. Each model is trained on a consistent set of images corresponding to its resolution. As training data we use videos from the TRECVid database [66], which consists of 60hrs of annotated news broadcast. In this experiment we selected images annotated as "Beach" as shown below.



Figure 49. Examples of training images for "beach".

The models are trained using the adapted Baum-Welch algorithms as described in section 3.4. Figure 50 shows the evolution of the total likelihood for the DT-MHMM during training.

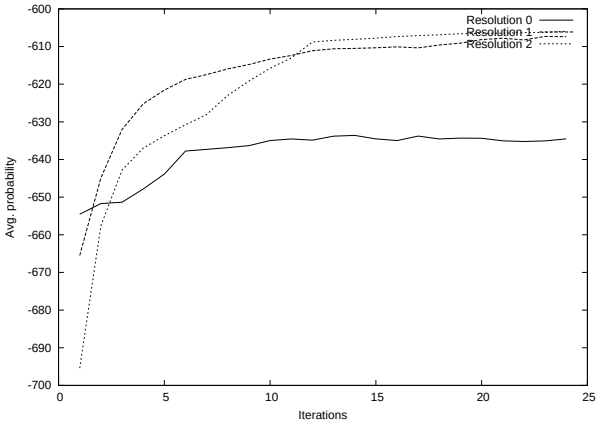


Figure 50. Total likelihood during training.

To compare with another algorithm we trained the 2-D MHMM described in section 4.2.1 on the same training set, using the same number of resolutions and states. Figure 51 shows the precision recall curve for the 2-D MHMM model and the two DT-MHMM models.

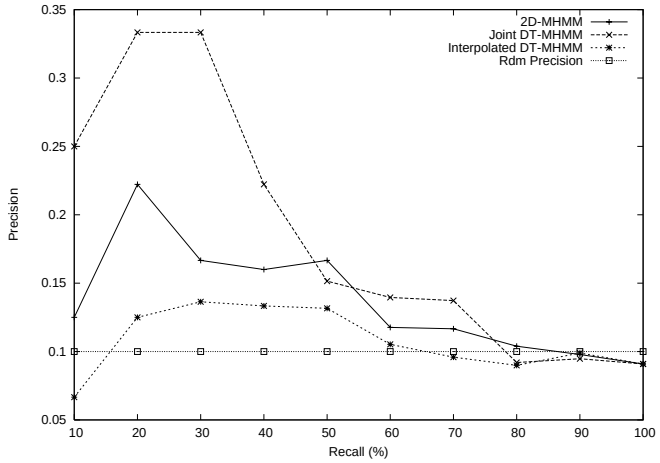


Figure 51. Precision recall for 2D-MHMM and DT-MHMM.

In our initial experiments we have already noticed the problem of one known drawback of the HMMs, that the output probability plays a more important role than the transition probability. The output distribution ranges over greater dispersion than the transition probability which range over 16 states only, with a majority of transitions from a state to itself. This explains why an image which has an almost uniform color (see the first six images in Figure 52) is likely to get a high output probability from one of its states. We can see in Figure 51 that the precision is low for the top ranked images, because the single colored images are ranked higher, than the true positives.



Figure 52. Top ranked images by the DT-MHMM.

We can observe that the model at the lowest resolution introduces global information that penalizes uni-colored images, but depending on the fusion scheme the higher resolution models sometimes takes over. The mean average precision for the different models are listed in Table 5.

Table 5. Mean average precision for DT HMM and 2-D MHMM.

Model	MAP
DT MHMM Joint combined	0.24
2-D MHMM	0.17
DT MHMM Interpolated	0.13

4.2.5 Conclusion

We have shown that the novel dependency tree HMM can be extended to multiple resolutions. The results indicate that we can improve the performance depending on the combination algorithm. By using fewer blocks, i.e. letting each block cover a larger part of the image we can improve the result since we reduce the chance of having single colored output probabilities. We must further investigate this and also what is the appropriate resolution for the lowest scale to model the global context so as to avoid the dominant output probabilities that arises from uni-colored blocks during training.

Chapter 5

Opening to 3-D Applications

5.1 Introduction

As a research direction of computer vision, object detection and tracking provides useful access to high-level information about objects obtained in an image and has attracted persistent interests.

In the past object tracking problems have been studied extensively and many different approaches have been proposed [97], [98], [99]. An approach based on color object tracking was presented in [100], where object training and tracking is modeled in the spatial domain rather than in the temporal domain.

A type of HMM for real-time tracking is proposed in [96] by using parametric shape model object contours. The authors also introduce a joint probability data association filter (JPDAF) to compute the transition probabilities with an enforced inter-relationship between neighbors.

The main obstacle to robust object tracking is that distracting features, such as clutter in the background regions, compete for the attention of the tracker and may succeed in pulling the tracker away from the foreground (target objects). To make the tracker reliable, it is common practice to discriminate the foreground pixels from the background pixels. Earlier researchers have attempted to increase the robustness of the tracker by image differentiation techniques [97].

Different domains of problems occur, the object to be tracked can be any mobile object in a scene or it can be a specific object that is already known and the desire is to follow its trajectory. In this chapter we focus on the latter case; where the object(s) can be trained in the first frame when the representation of the object is available. Once the training has been performed, object tracking consists of searching

the occurrence of the object in the successive frames of the video. This can be seen as a pattern recognition problem that can be solved by statistical machine learning tools.

A number of researches have introduced systems that employ statistical modeling techniques to video segmentation and object tracking [100], [102], [104]. In [101] the authors present a system for tracking vehicles based on a hidden Markov model. The HMM/MRF-based segmentation method is capable of classifying regions in an image into three different categories: vehicles, shadows of vehicles and background. The temporal continuity of the categories is modeled as single HMMs along the time axis independently of the neighboring regions. To incorporate spatial dependencies into the tracking process the output from the HMMs are regarded as a MRF generated through a stochastic relaxation process. This method has proven some success in discriminating foreground (vehicles) and non-foreground regions.

Give the rich resource of successful work using HMMs for 1-D and 2-D problems; it appears natural to extend 2-D HMM to three dimensions for object tracking in video. However until present 3-D HMMs have been very rarely used, and only on simplistic artificial problems [102].

In the following section we show how our proposed model is easily extended to three dimensions. Then, in section 5.3, we experiment this model on the problem of tracking objects in a video. We explain the initialization and the training of the model, and illustrate the tracking with examples of a real video.

5.2 3-D DT HMM

The DT HMM formalism is open to a great variety of extensions and tracks; for example other ancestor functions or multiple dimensions. Here we consider the extension of the framework to three dimensions. We consider the case of video data, where the two dimensions are spatial, while the third dimension is temporal. However, the model could be applied to other interpretations of the dimensions as well.

In three dimensions, the state $s_{i,j,k}$ of the model will depend on its three neighbors $s_{i-1,j,k}$, $s_{i,j-1,k}$, $s_{i,j,k-1}$. This triple dependency increases the number of transition probabilities in the model, and the computational complexity of the algorithms such as Viterbi or Baum-Welch. However the use of a 3-D Dependency Tree allows us to estimate the model parameters along a 3-D path (see Figure 53) which maintains a linear computational complexity.

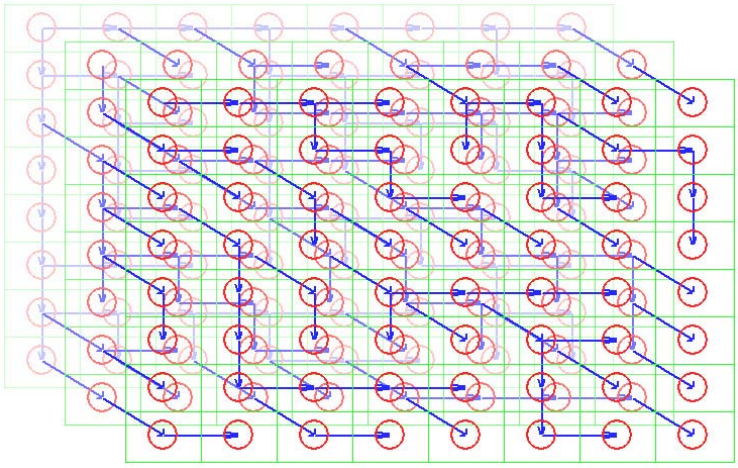


Figure 53. Random 3-D Dependency Tree.

The "direction" function for the 3-D tree becomes:

$$D(t) = \begin{cases} V & \text{if } t = (i-1, j, k) \\ H & \text{if } t = (i, j-1, k) \\ Z & \text{if } t = (i, j, k-1) \end{cases} \tag{5.1}$$

In 3-D modeling, images are represented by feature vectors on a 3-D grid. Let us denote the observation vector o_{ijk} as the observation of a block (i,j,k) in a 3-D image, volume image or sequence of 2-D images. In an analogous way the HMM state variables s_{ijk} represents the state at position (i,j,k) that produce the observation vector o_{ijk}. Thus now we can extend (3.6) to three dimensions:

$$\begin{aligned} & p(s_{i,j,k} \mid s_{i-1,j,k}, s_{i,j-1,k}, s_{i,j,k-1}, t) \\ & = p_{D(t(i,j,k))}(s_{i,j,k} \mid s_{t(i,j,k)}) \end{aligned} \tag{5.2}$$

In this chapter we use **Viterbi training** to fit our model, thus we need to iterate the search for the optimal combination of states and then re-estimate the model parameters.

5.2.1 3-D Viterbi Algorithm

The Viterbi algorithm finds the most probable sequence of states which generates a given observation O:

$$\hat{S} = \underset{S}{\operatorname{argmax}}\ P(O,S|t) \tag{5.3}$$

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

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

$$\beta_{i,j,k}(\boldsymbol{s}) = \boldsymbol{p}(\boldsymbol{o}_{i,j,k}|\boldsymbol{s}) \tag{5.4}$$

- if (i,j,k) has only an horizontal successor, by adopting equation (5.2) we get:

$$\beta_{i,j,k}(\boldsymbol{s}) = \boldsymbol{p}(\boldsymbol{o}_{i,j,k}|\boldsymbol{s}) \max_{\boldsymbol{s}'} \boldsymbol{p}_{\boldsymbol{H}}(\boldsymbol{s}'|\boldsymbol{s})\beta_{i,j+1,k}(\boldsymbol{s}') \tag{5.5}$$

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

$$\beta_{i,j,k}(\boldsymbol{s}) = \boldsymbol{p}(\boldsymbol{o}_{i,j,k}|\boldsymbol{s}) \max_{\boldsymbol{s}'} \boldsymbol{p}_{\boldsymbol{V}}(\boldsymbol{s}'|\boldsymbol{s})\beta_{i+1,j,k}(\boldsymbol{s}') \tag{5.6}$$

- if (i,j,k) has only a z-axis successor:

$$\beta_{i,j,k}(\boldsymbol{s}) = \boldsymbol{p}(\boldsymbol{o}_{i,j,k}|\boldsymbol{s}) \max_{\boldsymbol{s}'} \boldsymbol{p}_{\boldsymbol{Z}}(\boldsymbol{s}'|\boldsymbol{s})\beta_{i,j,k+1}(\boldsymbol{s}') \tag{5.7}$$

- if (i,j,k) has both an horizontal and a vertical successors (and respectively for the other two possible combinations):

$$\beta_{i,j,k}(\boldsymbol{s}) = \boldsymbol{p}(\boldsymbol{o}_{i,j,k}|\boldsymbol{s}) \left(\max_{\boldsymbol{s}'} \boldsymbol{p}_{\boldsymbol{H}}(\boldsymbol{s}'|\boldsymbol{s})\beta_{i,j+1,k}(\boldsymbol{s}')\right) \left(\max_{\boldsymbol{s}'} \boldsymbol{p}_{\boldsymbol{V}}(\boldsymbol{s}'|\boldsymbol{s})\beta_{i+1,j,k}(\boldsymbol{s}')\right) \tag{5.8}$$

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

$$\beta_{i,j,k}(\boldsymbol{s}) = \boldsymbol{p}(\boldsymbol{o}_{i,j,k}|\boldsymbol{s}) \left(\max_{\boldsymbol{s}'} \boldsymbol{p}_{\boldsymbol{H}}(\boldsymbol{s}'|\boldsymbol{s})\beta_{i,j+1,k}(\boldsymbol{s}')\right) \left(\max_{\boldsymbol{s}'} \boldsymbol{p}_{\boldsymbol{V}}(\boldsymbol{s}'|\boldsymbol{s})\beta_{i+1,j,k}(\boldsymbol{s}')\right) \left(\max_{\boldsymbol{s}'} \boldsymbol{p}_{\boldsymbol{Z}}(\boldsymbol{s}'|\boldsymbol{s})\beta_{i,j,k+1}(\boldsymbol{s}')\right) \tag{5.9}$$

Then the probability of the best state sequence for the whole image is $\beta_{0,0,0}(s_I)$. Note that this value may also serve as an approximation for the probability that the observation was produced by the model.

The best state labeling is obtained by assigning $s_{0,0,0} = S_I$ and selecting recursively the neighbor states which accomplish the maxima in the previous formulas. Note that the computational complexity of this algorithm remains low: we explore each block of the data only once, for each block we only have to consider all possible states of the model, and for each state, we have to consider at most three successors. Therefore, if the video data is of size (w, h, t), and the number of states in the model is N, the complexity of the Viterbi algorithm for 3-D DT HMMs is only O(whtN).

5.2.2 Relative Frequency Estimation

The result of the Viterbi algorithm is a labeled observation, i.e. a sequence of images where each output block has been assigned a state of the model. Then, it is straightforward to estimate the transition probabilities by their relative frequency of occurrence in the labeled observation, for example:

$$p_H(s'|s,t) = \frac{N_{H,t}(s,s')}{N(s)} \qquad (5.10)$$

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 case of discrete output probabilities, or using standard Multi-Gaussian estimation in the case of continuous output probabilities.

With these algorithms we can estimate the model parameters from a set of training data, by so called Viterbi training: starting with an initial labeling of the observation (either manual, regular or random), or an initial model, we iteratively alternate the Viterbi algorithm to generate a new labeled observation and Relative Frequency estimation to generate a new model. Although there is no theoretical proof that this training will lead to an optimal model, this procedure is often used for HMMs and has proven to lead to reasonable results.

Note that with 3D-DT HMMs, the Baum-Welch algorithm and the EM reestimation lead to a computational complexity that is similar to the Viterbi algorithm, so that a true Maximum Likelihood training is computationally feasible in this case. We have not yet implemented those algorithms, but started with the simple Viterbi and Relative Frequency for our initial experimentations on 3-D video data.

5.3 Experiment

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. Our motivation is to model these dependencies by a 3-D HMM. As described in 5.2, images are represented by feature vectors on an array of 2-D images.

To investigate the impact of the time-dimension dependency we explore the ability of the model to track objects that cross each other or pass behind another static object. To this end we have chosen a video sequence with two skiers that pass behind each other and static markers that remain fixed on the scene. The video contains 24 frames.

The method is mainly composed of two phases: the training phase and the segmentation phase. In the training phase, the process learns the unknown HMM parameters using the Viterbi training explained in section 5.2.1. In the segmentation phase, the process performs a spatio-temporal segmentation by performing a 3-D Viterbi state alignment.

5.3.1 Model Training

We consider a 3-D DT HMM with 9 states and continuous output probabilities. Our example video contains 24 frames which are divided into 44 x 30 blocks; hence the state-volume has dimension 44 x 30 x 24. For each block, we compute a feature vector as the average and variance of the 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). This constitutes the observation vector o_{ijk}.

Initial Model
The first step of the training is to build the initial model. To build initial estimates of the output probabilities, we manually annotated regions in the first frame of the video by segmenting the image into arbitrary shaped regions using the algorithm proposed in [86] and then manually associating each region with a class category. As it can be seen in the figure below, the segmentation is rather coarse, which means that parts of the background may be included in the object regions.

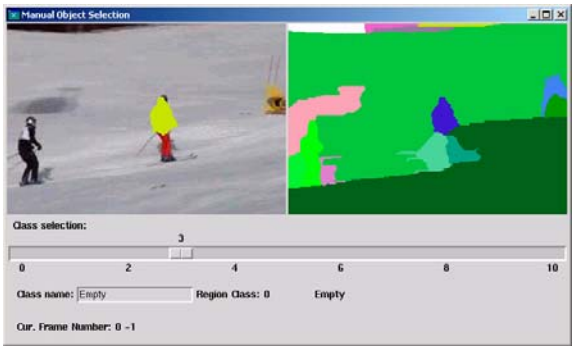


Figure 54. Training image and initial state configuration using annotated regions.

The image was labeled using four different categories: background, skier1, skier2 and marker (static object in the scene). Since semantic video regions do not usually have invariant visual properties, we assign a range of states to allow for a flexible representation for each category (or sub-class). The sub-class *background* can for instance have the colors: white or light grey with different texture properties such as regular or grainy. The table below lists the sub-classes and their associated number of states.

Table 6. The number of states for each sub-class.

Sub Class	No. states
Background	3
Skier 1	2
Skier 2	2
Marker	2
4 sub-classes	9 states

Each state has an output distribution which is represented by a GMM (Gaussian Mixture Model) with five components. To estimate these probabilities, we collect the observation vectors for each category, cluster them into the corresponding number of states, and perform a GMM estimation on each cluster. The transition probabilities are estimated by Relative Frequency on the first frame for the spatial dependencies, and by uniform distribution for the temporal dependency.

3D Dependency tree

The observation volume has size 44 x 30 x 24, and each observation is supposed to be generated by a hidden state. We build a random 3-D dependency tree (see Figure 53) by creating one node for each observation, and randomly selecting an ancestor for each node out of the three possible directions: horizontal, vertical and temporal. The border nodes have only two (if they lie on a face of the volume), one (if they lie on an edge) or zero (for the root node) possible ancestors. In our experiments, we generated a random 44 x 30 x 24 dependency tree which contains:

Table 7. Number of ancestors for each direction.

Direction	No. ancestors
Horizontal	10 604
Vertical	10 694
Temporal	10 382
Total	31 680

As previously explained, we perform Viterbi training by iteratively creating a new labeling over the observation, using Viterbi training, and generating a new model, using Relative Frequency estimation. The iterations stop when the increase in the observation probability p(O | S, t) is less than a threshold.

Data size vs. Model Complexity
According to the Bayesian information criterion (BIC) the quality of the model is proportional to the logarithm of the number of training samples and the complexity of the model [87]. Thus since our training data is sparse, we shall use a small mixture size. The necessary complexity of the GMM depends of the data class to model, which in this case is relatively uniform since the image is segmented into annotated object regions (each one represented by a number of states as shown in Table 6).

The EM-algorithm is used for training, which has a tendency to make very narrow Gaussians around sparse data points. To avoid this potential problem we construct the GMMs so that there are always a smallest number of samples in each component, and we constrain the variance to a minimum threshold.

5.3.2 Object Tracking

The original video contains two skiers passing yellow markers on a snowy background with shadows. Figure 55 depicts every second frame of the sequence.

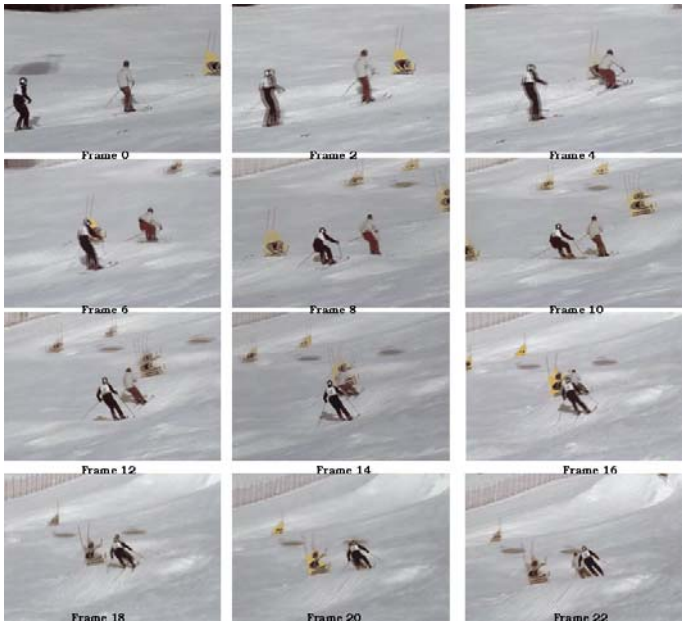


Figure 55. Original video sequence; first frame in upper left corner, followed by every second frame.

The first frame was manually annotated and used to estimate the initial model, while the following frames constitute the 3-D observation (or a sequence of 2-D observations) on which the Viterbi training was performed. Then, we use the trained model to get a final labeling of the complete 3D observation. In the final labeling, each observation block is assigned to a single state of the model. The final labeling provides a spatio-temporal segmentation of the 3D observation. The experiment was performed on a Pentium 2.99 GHz PC with a LINUX operating system. The average time to perform a 3-D Viterbi alignment was 20s.

As the states of the model correspond to semantic categories, it is possible to interpret the content of specific blocks in the video sequence. Figure 57 shows the segment classification for frame 1, 12 and 24.



Figure 56. Frame segmentation in the final labeling (a) frame 1, (b) frame 12, (c) frame 24.

Object tracking is then performed easily, by selecting in each frame the blocks which are labeled with the corresponding semantic category. For example, we can easily

create a video sequence containing only the skiers by switching off the states for the background- and marker-classes as shown in Figure 57.

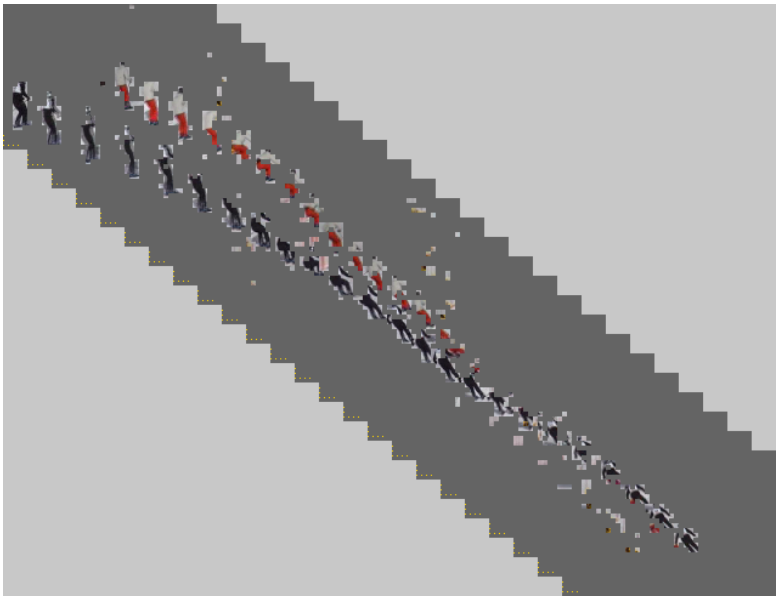


Figure 57. Object tracking of two skiers.

We can see in the figure above that some blocks are incorrectly assigned to the skier categories. An explanation for this may be that with a single dependency tree many blocks inside the video corresponds to leaves in the tree, and are therefore not constrained by any successor. This motivates the combination of several dependency trees so that no node is left without any constraints from its successors.

Complementary Dual Trees

We would like to consider dependencies in every direction for each node. Therefore, given a random dependency tree t, it is reasonable to consider its dual trees, which are trees where for each node, we select one direction among those not used in t (except when the node has a single possible ancestor). Note that in 2D, the dual tree is unique, while in 3D, there are a lot of different dual trees for a given t. However, we can select a pair of complimentary dual trees so that every possible dependency for every node appears at least once in one of the three trees. We use a majority vote to compute the best labeling for the triplet of trees. Figure 63 show the result of the segmentation on various frames.



Figure 58. Frame segmentation using complementary dual trees, (a) frame 1, (b) frame 12, (c) frame 24.

As previously, we can construct a video sequence showing only the tracked objects:

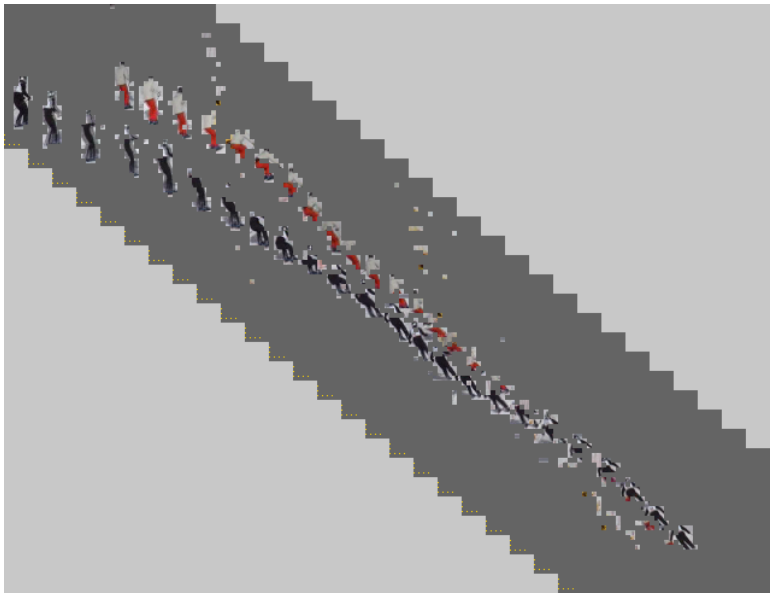


Figure 59. Perspective view of object tracking using complementary dual trees.

Unfortunately, this combination shows only minor improvement over the segmentation with a single tree.

Multiple tree labeling

Although using a triplet of tree and complimentary dual trees takes every dependency from every node into account, this is only a local constraint between neighbors, which may not be sufficient to propagate the constraint to a larger distance. Notice that, for every pair of nodes (not necessarily neighbors), there is always a dependency tree where one of the nodes will be the ancestor of the other. So, the idea now is to use a large number of trees (ideally all, but they are too numerous), so that we increase the chance of long-distance dependency between non-neighbor nodes.

For each dependency tree, we can compute the best state alignment, then use a majority vote to select the most probable state for each block. This is an approximation for the probability of being in this state for this block during the generation of the observation with an unknown random tree (a better estimate could be obtained

using the extended Baum-Welch algorithm, but we have not implemented this algorithm yet, so we just use the Viterbi algorithm here). Figure 60 shows the video obtained with this multiple tree labeling, using a set of 50 randomly generated trees.

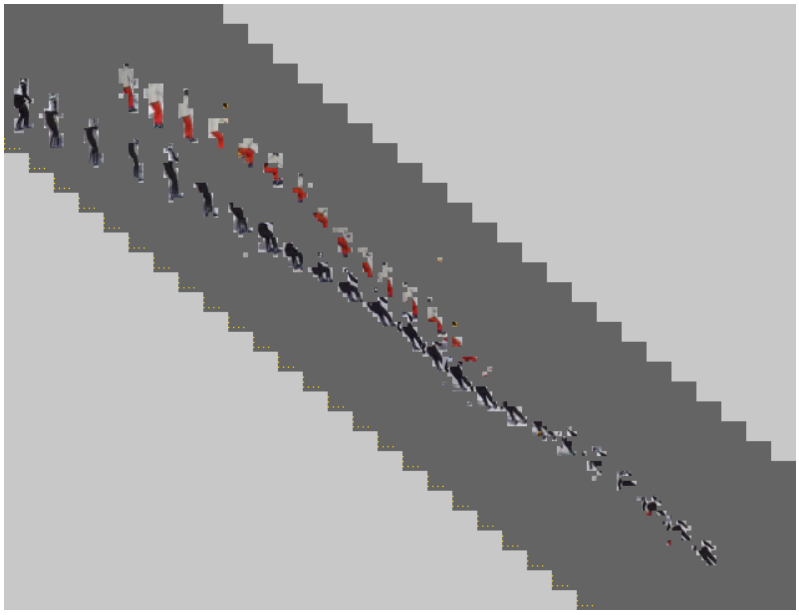


Figure 60. Object tracking with smoothing over 50 random trees.

As can be seen from these results, the objects are much clearly defined in this experiment, and most of the noise in the labeling has disappeared.

5.3.3 Conclusion

In this chapter, we have proposed a new approximation of multi-dimensional Hidden Markov Model based on the idea of Dependency Tree. We have focused on the definition and use of 3-D HMMs, a domain which has been very weakly studied up to now, because of the exponential growth of the required computations.

Our approximation leads to reasonable computation complexity (linear with every dimension). We have illustrated our approach on the problem of video segmentation and tracking. We have detailed the application of our model on a concrete example. We have also shown that some artifacts due to our simplifications can be greatly reduced by the use of a larger number of dependency trees.

In the future, we plan to explore other possibilities of 3-D HMMs, such as classification, modeling, etc… on various types of video. Because of the learning capabilities of HMMs, we believe that this type of model may find a great range of applications.

Chapter 6

Conclusions

6.1 Summary and Contributions

After presenting an out-of-the box introduction to image understanding, we give a review of current statistical machine learning algorithms in chapter 2. It was suggested that in order to improve image classification, contextual constraints among visual features should be taken into account.

The hidden Markov model has both a rich mathematical structure, and (when applied properly) work very well for modeling context in sequential data. Thus we proposed to represent images by 2-D hidden Markov models with the underlying state processes being 2^{nd} order Markov meshes. However due to the exponential computational complexity that arises for 2-D problems, we were motivated to conceive an algorithm that can make the estimation of the model parameters tractable.

Consequently the main contribution of this dissertation consists in the proposal and study of an efficient method (the DT HMM) for approximating the parameters of a multidimensional hidden Markov model based on a random dependency tree.

In chapter 3 we have discussed the theoretical framework of hidden Markov models as developed in the machine learning field in the last twenty years. Then we showed how to perform the extension to two dimensions and how to solve the three fundamental problems of HMM design [37], given our new framework.

Formulas were derived for estimating the model, based on the EM algorithm developed for maximum likelihood estimation with incomplete data. Following the idea of the forward-backward algorithm for estimating 1-D HMMs, we developed a similar

algorithm that reduces computation to O(wh) in complexity instead of $O(wN^{2h})$, where N is the number of states in the model and (w,h) is the width respectively height of the image.

We developed the components of the DT HMM, along with other involved mathematical tools such as GMMs, k-Means and Vector Quantization (see Appendix B) in order to conduct experiments on the model.

To study the influence of the dependency tree we investigated the variation of the observation probability with respect to the particular dependency tree that is used. The results showed that there are great chances that a random tree will provide a value which is close to the average. We also compared three different approximations of the exact probability (3.44) which results may suggest the choice of the best approximation method, based on the deviation which is considered reasonable for a given application.

In searching for the point of operation of the model, we encountered an intrinsic weakness of the HMH; that the output probability plays a more important role than the transition probability. This explains why images with uniform colors get a very high probability; because they have a very large emission probability.

It was also confirmed that the contextual model sometimes fails to discriminate sub-classes with-in the concept which might suggest the scale is too low. At one hand we need high scale to distinguish fine details in objects, and low scale to capture global properties. This gave rise to the ideas of introducing restrictions to the states in order to enforce sub-classes, as well as to form a multiscale model in order to combine global and local context.

Therefore in 4.1 we explored the inherent ability of the DT HMM to automatically associate image blocks to semantic sub-classes represented by the states of the Markov model. We used a modified version of the Viterbi algorithm that is able to handle the situation when a visual sub-class is represented by several states, and only the sub-class annotation (not the state annotation) is available. We investigated several properties of this process. The performance of this joint probabilistic mapping of states was evaluated on a wide within-class database to illustrate its tolerance to invariant visual evidences.

Finally we investigated the role of the transition probabilities. The experiment was carried out by letting the model label 40 in-class but unseen test images. The average rate of correct labeled blocks was 38% when taking transition probabilities into account and 32% with only the output probabilities modeled by the Gaussian mixture models.

Since we had detected the problem of high output probabilities in chapter 3, we wanted to introduce global context to overcome the effect of dominant uni-colored images. Chapter 4.2 deals with the extension to multiple resolutions. The extension allows an image to be represented by observations in several resolutions which

corresponds to local and global context. When the DT HMM is extended to multiple resolutions, it shows that it has a competitive performance in comparison to other known algorithms. We could observe that the model at the lowest resolution introduced global information that penalize uni-colored images, but depending on the fusion scheme the higher resolution models sometimes takes over. The mean average precision for the different models are listed in Table 5.

Finally in Chapter 5 we illustrated that the DT HMM formalism is open to a great variety of extensions and tracks. Since 3-D HMMs has been little studied we investigated the extension of the framework to three dimensions. We considered the case of video data, where two dimensions are spatial, and the third is temporal. The model is applied to the problem of video segmentation and tracking. The approximation lead to reasonable computation complexity and we showed that some artifacts, due to our simplifications, can be greatly reduced by the use of a larger number of dependency trees

This work resulted in the following conference and workshop articles:

1. J. Jitén, B. Merialdo, ”Video Modeling Using 3-D Hidden Markov Model”, 2nd International Conference on Computer Vision Theory and Applications, 8 - 11 March, 2007 Barcelona, Spain

2. J. Jiten, B. Mérialdo, ”Semantic image segmentation with a multidimensional Hidden Markov Model”, MMM 2007, 13th International Multi-Media Modeling Conference, January 9-12, 2007, Singapore

3. B. Mérialdo, J. Jiten, E. Galmar, B. Huet, “A new approach to probabilistic Image modeling with multidimensional hidden Markov models”, AMR 2006, 4th International Workshop on Adaptive Multimedia Retrieval , 27-28 July 2006, Geneva, Switzerland

4. J. Jiten, B. Mérialdo, B. Huet, “Multi-dimensional dependency-tree hidden Markov models”, ICASSP 2006, 31st IEEE International Conference on Acoustics, Speech, and Signal Processing, May 14-19, 2006, Toulouse, France

5. J. Jiten, B. Mérialdo, ”Probabilistic image modeling with dependency-tree hidden Markov models”, WIAMIS 2006, 7th International Workshop on Image Analysis for Multimedia Interactive Services, April 19-21, 2006, Incheon, Korea

6. J. Jiten, B. Huet, B. Mérialdo, ”Semantic feature extraction with multidimensional hidden Markov model”, SPIE Conference on Multimedia Content Analysis, Management and Retrieval 2006, January 17-19, 2006 - San Jose, USA Volume 6073 , pp 211-221

7. J. Jiten, F. Souvannavong, B. Mérialdo, B. Huet, "Eurecom at TRECVid 2005: extraction of high-level features", TRECVid 2005, TREC Video Retrieval Evaluation, November 14, 2005, USA

8. B. Cardoso, F. de Carvalho, C. Fausto; L. Cravalho, G. Fernandez, P. Gouveia, B. Huet, J. Jiten, B. Mérialdo, A. Navarro, H. Neuschmied, M. Noe, R. Salgado, G. Thallinger, "Hyperlinked video with moving objects in digital television", ICME 2005, IEEE International Conference on Multimedia & Expo, July 6-8, 2005, Amsterdam, The Netherlands

9. B. Huet, J. Jiten, B. Mérialdo, "Personalization of hyperlinked video in interactive television", ICME 2005, IEEE International Conference on Multimedia & Expo, July 6-8, 2005, Amsterdam, The Netherlands

10. B. Cardoso, F. de Carvalho; G. Fernandez, B. Huet, J. Jiten, A Lopez, B. Mérialdo, H. Neuschmied, M. Noe, D. Serras Pereira, G. Thallinge, "Personalization of interactive objects in the GMF4iTV project", TV'04, 4th Workshop on Personalization in Future TV held in conjunction with Adaptive Hypermedia 2004, August 23, 2004, Eindhoven, The Netherlands

6.2 Future Work

In the conclusion of chapter 4 and 5 we recognized the importance of the block size. By using fewer blocks, i.e. letting each block cover a larger part of the image we can improve the result since we reduce the chance of having single colored output probabilities. We can further investigate this and also what is the appropriate resolution for the lowest scale to model the global context so as to avoid the dominant output probabilities that arises from uni-colored blocks during training.

Because of the learning capabilities of HMMs, we believe that this type of model may find a great range of applications. We may consider other 3-D applications such as 3-D classification or image reconstruction.

Further since the dependency tree introduces discontinuities we may find other ways to select a tree. For example make an optimal tree for a set of images or application by first analyzing each image and make more connections between boundary regions, or consider other non-random trees.

A future way to approach the problem of discontinuities is to use hierarchical models, since a block in a lower resolution may introduce connections that does not exists in the smaller scale.

The model may also be extended to higher dimensions (n>3). In that case the contextual connections become weaker since for each dimension we will have one

connection out of *n* neighbors. For this reason the choice of the tree will become even more important.

In conclusion we believe that the DT HMM is a powerful model that hopefully will lead to powerful applications in the future.

Appendix A

Training Data

A.1 TRECVid Archive

Much of the experimental work in this thesis was accomplished by using the TREC-Vid [66] archive for training and testing. As a participant of the workshop TRECVid 2005 [108] we had access to 133 hours of annotated ABC/CNN TV newscasts, composed of news, weather, sports, financial reports, and commercial advertise-ments.

Common Feature Annotation
A common annotation effort was organized by Lin et al. at IBM [105]. The work was carried out by dividing the participants into 23 groups who used IBM software to manually annotate the development collection of over 60 hours of video content with respect to 133 semantic labels. The application facilitated the annotation process by given the shot boundaries; the corresponding key frame was presented through a graphical user interface that permitted the user to associate regions or the entire image with a semantic label.

There were significant contributions from other participants as well; Jean-Luc Gauvain of the Spoken Language Processing Group at LIMSI provided automatic speech recognition (ASR) output for the entire collection [109]. Georges Quénot of the CLIPS-IMAG [106] group provided a common set of shot boundary definitions and keyframes, which served as the predefined units of evaluation for the feature extraction (see section A.2). The data was then made available to all participants in MPEG-7 format for subsequent use such as development, training and testing.

In total, the development collection contained 133 videos in MPEG-1 format and 35,067 shots as defined by the common shot reference, together with its annotations.

The advantage of the common annotated data is primary to have access to a large quantity of high quality annotated image/video data, and to have the possibility to objectively compare the performance of different systems.

Contribution
I participated in the feature extraction task, which was to contribute with search results for evaluating the effectiveness of detection methods for semantic concepts such as “People", “Waterscape", “Explosion" etc., which occur frequently in video information.

More precise, the task was defined as follows: given a feature test collection, the common shot boundary reference for the feature extraction test collection, and the list of feature definitions (see below), participants returned for each feature the list of at most 2000 shots from the test collection, ranked according to the highest possibility of detecting the presence of the feature. Each feature was assumed to be binary, i.e., it was either present or absent in the given reference shot.

The feature extraction task has two objectives: to continue work on a benchmark for evaluating the effectiveness of detection methods for various semantic concepts, and allowing exchange of feature detection output for use in the TRECVID search test set prior to the search task results submission date.

The number of features (which was discussed and proposed online by participants) to be detected was kept small (10) so as to be manageable and the features were ones for which more than a few groups could create detectors. Another consideration was whether the features could, in theory, be used in executing searches on the video data using the topics.

1. People walking/running: segment contains video of more than one person walking or running
2. Explosion or fire: segment contains video of an explosion or fire
3. Map: segment contains video of a map
4. US flag: segment contains video of a US flag
5. Building exterior: segment contains video of the exterior of a building
6. Waterscape/waterfront: segment contains video of a waterscape or waterfront
7. Mountain: segment contains video of a mountain or mountain range with slope(s) visible
8. Prisoner: segment contains video of a captive person, e.g., imprisoned, behind bars, in jail, in handcuffs, etc.
9. Sports: segment contains video of any sport in action
10. Car: segment contains video of an automobile

A.2 Low-level Features

Images are represented by features that are extracted directly from their digital representations which are therefore often called *low-level features*. The process of extracting features is the basis for image classification, and it is decisive for the final quality of the system.

Many features (or descriptors) have been proposed; frequently used features for images are color, shape, texture, color layout, etc. A comprehensive review can be found in [115], [121]. In the following section I will introduce the features that were studied in the course of this thesis.

Color descriptors
Color features are one of the most widely used, mainly because of its robustness and invariance to image size and orientation. For humans, color is the perceptual result of visible light, which lies within the spectrum wavelength of 380 nm - 750 nm. In computer applications it is usually represented by three color channels corresponding to the amount of red (R), green (G) and blue (B). To facilitate the specification of colors in a standard way, the image processing community use *colors models*.

A color model (or color space) is a specification of a coordinate system and a sub-space where each color is represented by a single point. There are several color models oriented towards different hardware and software applications. Below I will briefly present the most common models for image classification.

HSV is a natural representation color model, which means close to the physiological perception of the human visual system (HVS). It consists of hue angle (H), color saturation (S) and brightness (V). It is good in its capacity to recognize the presence or absence of colors (hue) in an image, which is often explored since the HVS is more sensitive to changes in hue than saturation or value.

A step to gain more control over color is to use a perceptual colors model like CIE LUV, CIE Lab, or Munsell [113]. In 1976, the CIE (Commission Internationales de l'Eclairage) defined CIE LUV color space (based on CIE XYZ) to enable us to get more uniform and accurate models. LUV is used in calculation of small color differences, especially with additive colors, and it is often chosen because of its good perception correlation properties [39]. This means that the Euclidean distance between two colors in the LUV color spaces is strongly correlated with the human visual perception.

Another popular color space is YUV that is used in European TVs, and YIQ in North American TVs (NTSC). Lim et al. [111] found that the YIQ color coding performed better over other color spaces in there effort to extract semantics in home photo

collections. Further studies of color perception and color spaces can be found in [119], [120].

Texture descriptors

A *texture* can be described as the feature that captures spatial distribution of luminance variations in terms of *visual patterns*. It is an innate property of most surfaces, including clouds, vegetation, bricks, hair etc. It contains important information about the structural arrangements of surfaces and their relationships to its neighborhood.

The Discrete Cosine transform (DCT) provides the visual information of an image block with different frequencies. AC coefficients in the upper left corner reflect information of lower frequencies, whereas those AC coefficients in the lower right corner correspond to information of higher frequencies (c.f. Figure 61). The transform has been a predominant tool in signal and image processing for decades due to its computational efficiency and effectiveness in representing images.

Extensive psychological studies [112] conclude that human vision system is not of uniform discrimination to details of different frequencies: changes occur in low frequencies is far more distinguishable than those in high frequencies. Therefore, a few coefficients can be chosen to represent the horizontal and vertical intensity variations of an image as exemplified below.

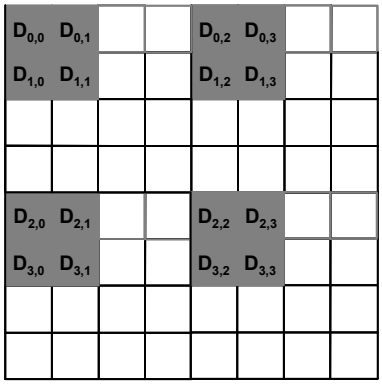


Figure 61. DCT coefficients of an 8 x 8 image block.

Gabor wavelets, which are plane waves restricted by a Gaussian envelope, have long been successfully applied to the problems of image analysis because of their biological relevance and computational properties [116][117].

Basically, Gabor filters are a group of wavelets [110], with each wavelet capturing energy at a specific frequency (ω) and a specific direction (θ). Expanding a signal using this basis provides a localized frequency description. Often a set of Gabor filters are produced for texture analysis, for example 4 scales and eight orientations as depicted below.

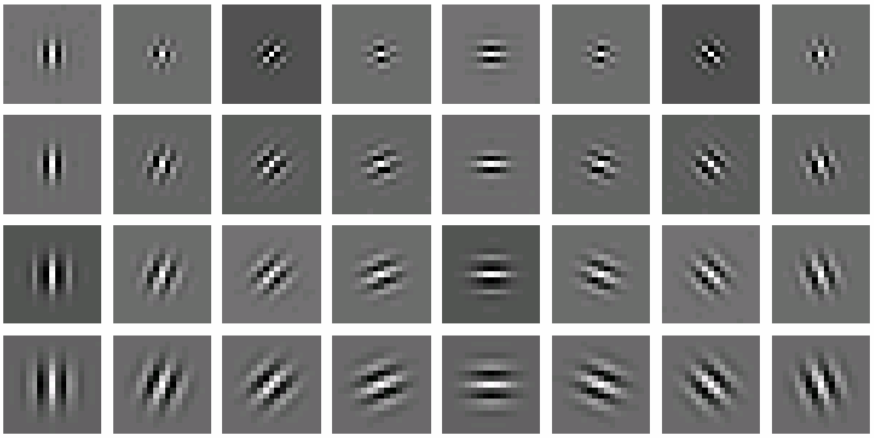


Figure 62. Example of Gabor kernels at 4 scales and 8 orientations

A low scale (high frequency ω), implies a compressed wavelet that captures rapidly changing details, and inversely a high scale (low frequency ω) captures slowly changing details. The Gabor wavelets have been found to be particularly suitable for image decomposition and representation when the goal is the derivation of local and discriminating features.

Shape Descriptors

To allow us to describe a structural content of images it is desirable to use shape or structural descriptors. Usually a shape represents a specific edge feature that is related to object contour. The Canny edge operator takes as input a gray scale image, and produces as output an image showing the positions of tracked intensity discontinuities (see Figure 63).

The operator works in several steps; first it uses linear filtering with a Gaussian kernel to smooth noise and then applies a simple 2-D first derivative operator to highlight regions with high first order spatial derivates, these edges give rise to ridges in the gradient magnitude image (called edgels). The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output.

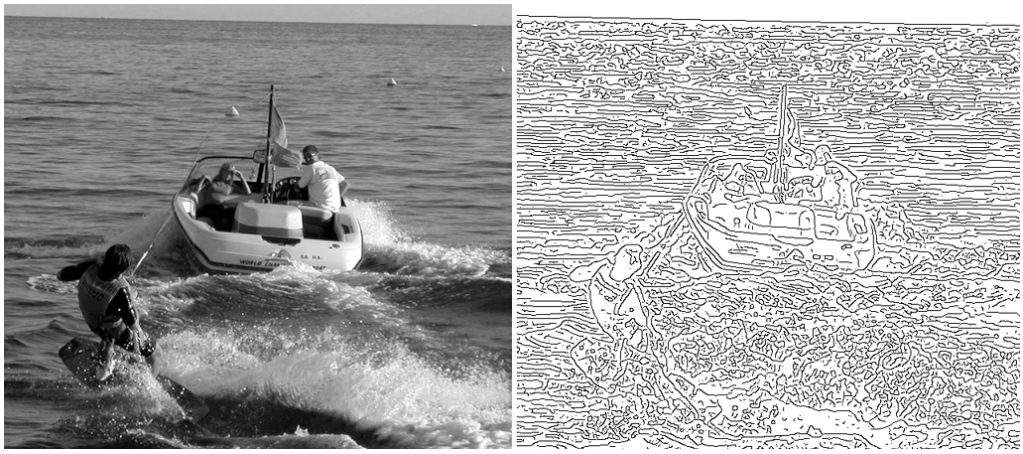


Figure 63. Example of marked edges in an image using Canny's algorithm.

Problems might occur where three ridges meet in the gradient magnitude image, the tracker will treat two of the ridges as a single line segment, and the third one as a line that approaches, but doesn't connect to.

Another simple (but effective) edge detector is the Sobel detector, which uses a simple convolution kernel to create a series of gradient magnitudes. However the Sobel algorithm is very sensitive to noise in the picture which is often the case with photographic images [118].

Summarizing
Depending on the application, features represent global or local properties in the image. Local features can be computed over arbitrary regions or non-overlapping blocks. In both cases the features are usually summarized within the region by forming a histogram. Statistically, a histogram denotes the joint probability of the attributes in the feature vector. Considering that most histograms are sparse and sensitive to noise, Stricker and Orengo used cumulated histograms and to overcome the quantization effects they proposed to use a color moments approach [122].

Since most of the information is concentrated on the low-order moments, and in the light of the fact that I use GMMs to represent the output probabilities (which are difficult to train for highly skewed probability distributions), I decided to use means and variances to represent the feature vectors in my experiments.

Dimension Reduction
When forming a signature it is common to concatenate several descriptors to one high dimensional feature vector. To make the system scalable to large size image collections, it is thus sometimes necessary to perform a dimension reduction.

Even though the dimension of the resulting feature vector is high, the *embedded* dimension can be much lower. Principal Component Analysis (PCA) can be applied

to transform the features to a reduced space by computing a new, smaller set of uncorrelated variables which best represent the original data.

Experiments show that most real data set can be considerably reduced in dimension without significant degradation in retrieval quality [123]. However in this thesis our feature vectors never exceeds 36 dimensions, so we kept the full vector.

Appendix B

Implementation Notes

The experiments are carried out by procedures developed in ANSI (GNU) C++ using the Standard Template Library (STL). I would like to say an encouraging word about this since I was used to work with object pointers; working with templates apparently has several advantages:

1. It's less error prone since by using automatic allocation (on the stack) of container objects, they take care of the explicit memory allocation (using new, delete).
2. Allows use of C++ scoping rules to avoid memory leaks.
3. The STL gives C++ a new level of abstraction. The Vector container for instance knows its size, takes care of memory allocation and knows how to serialize its self.
4. Common algorithms are already implemented such as sort, min/max and search element.

Class Design
In coding algorithms for research purposes it is important to have a flexible object structure to facilitate future modifications meanwhile retaining code reusability. We therefore broke out some common objects that were inherited for different specialized models. The figure below gives a comprehensive view of the inheritance structure used for implementing the multiresolution hidden Markov models used in 4.2.

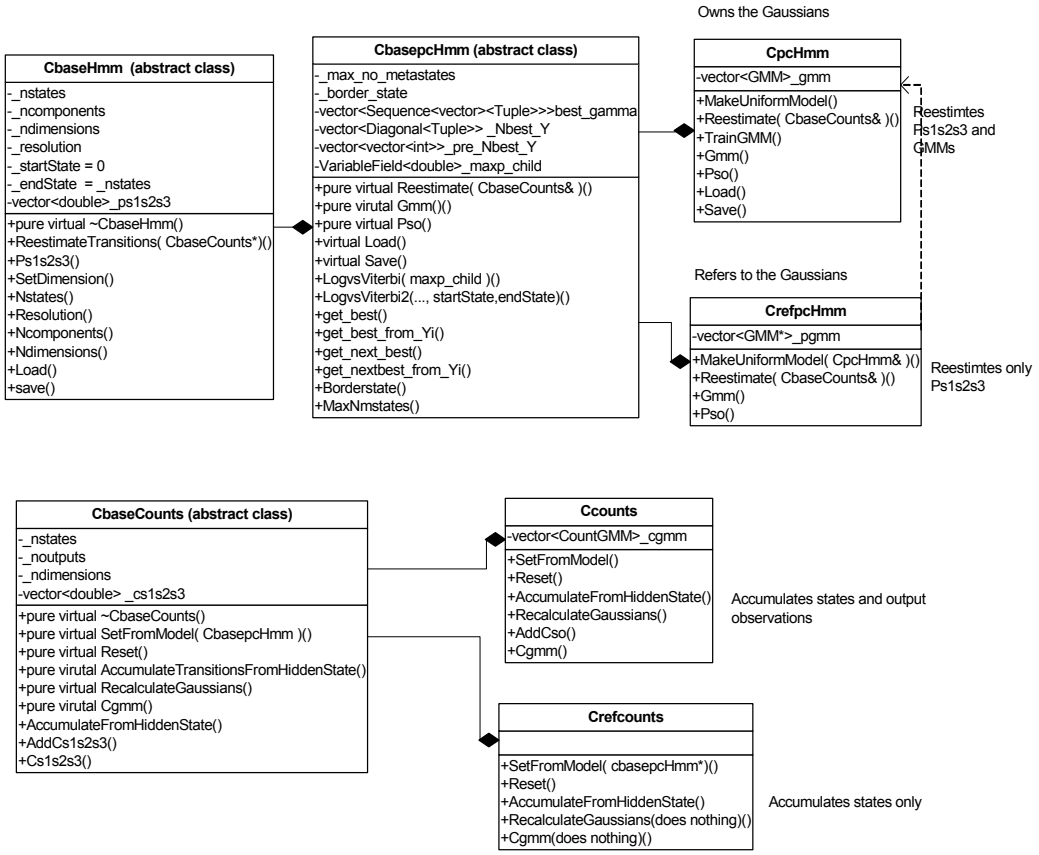


Figure 64. DT HMM object relations.

The base classes *CbaseHmm* and *CbaseCounts* are abstract classes since they contain pure virtual functions[12] and can therefore not by instantiated. To implement the multiresolution path constrained model proposed in [79], we instantiate an array of *CpcHmm* and *CrefpcHmm* which are derived from *CbasepcHmm*. Since according to the algorithm, each resolution share the same output probabilities, only *CpcHmm* instantiates a GMM, to which the *CrefpcHmm* holds a pointer. Due to the fact that the reestimation function is virtual, we can call the *Reestimate()* to all objects in the array and let the objects themselves know what action to perform. In this case: update the Gaussians or not.

[12] A pure virtual function provides no default code in the base class.

Low-level Feature Production

We used the Dali video library [125] to manage video streams and extract video frames. The library is developed at Cornell University and contains an extensive set of routines for manipulating video, audio, and image data. For image analysis and processing I used VIGRA (Vision with Generic Algorithms) [124], which is heavily based on templates and generic programming. An advantage is that it obviates tiles and *scanlines* by using *iterators*. Other benefits are:

- Supports very large images
- Supports color models
- Supports different bit depths through templating
- Modern C++
- Relatively small
- Well-documented

The figure below shows the objects involved to perform image processing and feature extraction.

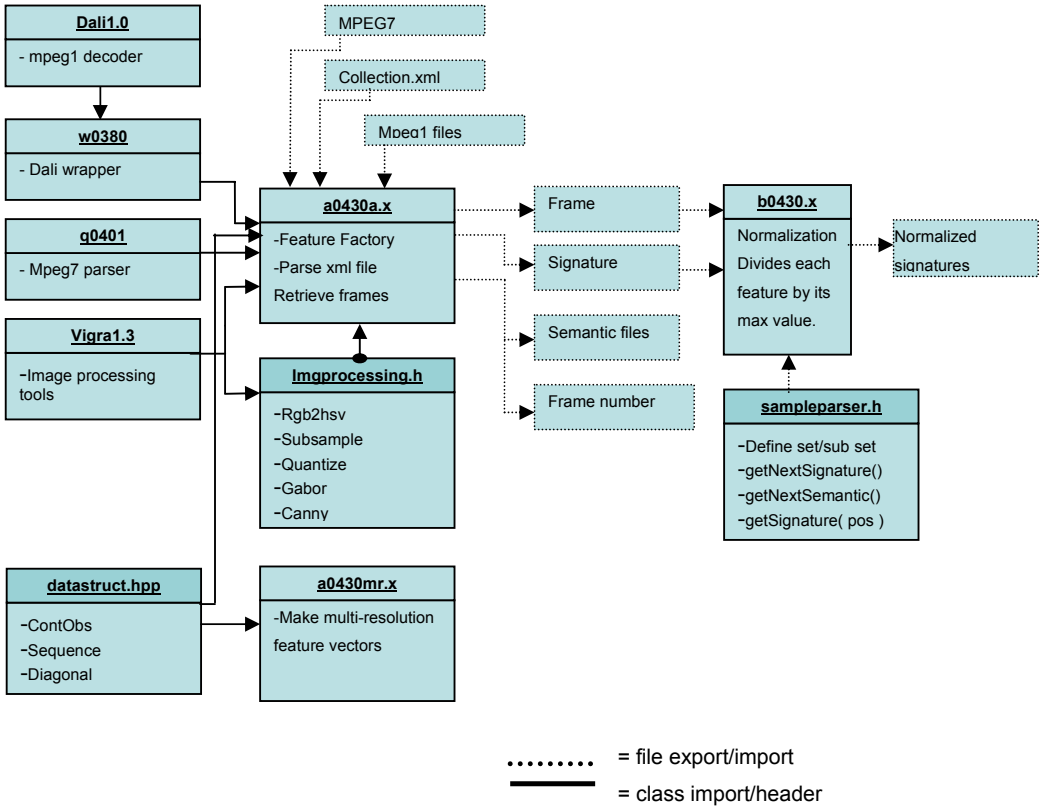


Figure 65. Low-level feature extraction objects.

The low-level features (signature files) were produced by a script file (a0430.x) that automates the procedure which is divided in the following steps: for every key-frame in the video archive, compute and store the demanded low-level features, update the file index database. To avoid ill-conditioned problems during the training process, an optional second step (b0430.x) was used to zero-mean normalize the data.

Training Algorithms

We implemented algorithms for k-Means, GMM, HMM, multi-dimensional algebra and multivariate statistics in C++. The design goal was, as earlier mentioned, to make use of the STL as much as possible. The figure shows the relations of the objects involved for training a DT HMM model.

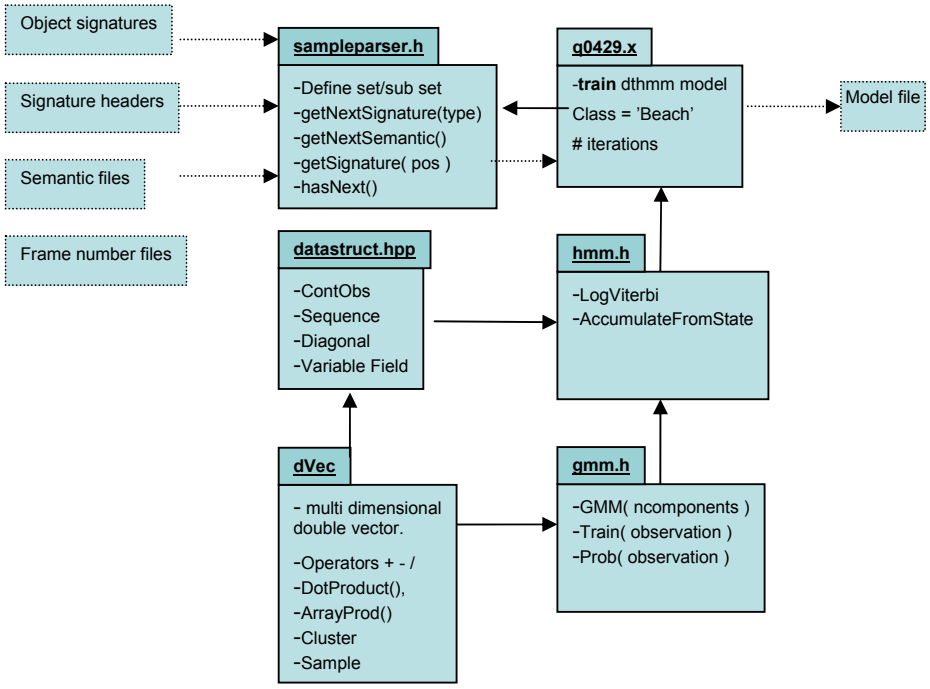


Figure 66. DT HMM object relations.

A script file (q0429.x) controls the training process. Given a concept class, it finds the corresponding feature vectors in the low-level feature archive by calling *getNextSignature* defined by the *Sampleparser*. The class feature data is fed to the classifier module which iteratively computes the statistical model.

Appendix C

Notation and Abbreviations

A summary of the most commonly used notations and abbreviations are listed in the tables below. I adopt the convention that random variables are written with capital letters, and instantiations there of (values) are written with lower-case letters. When working with linear algebra, we denote vectors as lower case and matrices with upper case.

When vectors are used in linear algebra manipulations (for instance as feature vectors) with matrices and other vectors, we will assume that they are column vectors so that strictly speaking the vector should be denoted $x = (x_0,x_1,\ldots,x_{k-1})^T$, where T denote transpose.

Through out this thesis we employ the following terminology when discussing concept learning:

Table 8. Machine learning terminology.

Term	Meaning
Concept	Comprehensive term for the concept.
Ground truth	Same as *target concept* also called training data or feature vector with class label.
Instance	Feature vector/sample; the data over which the model is defined.
Model	Mathematical construction to explain the concept.
Target concept	Class label; the concept to be learnt.

Table 9. Notation for classification algorithms.

Symbol	Meaning
ω	Estimation of the target class function to be learnt, a.k.a. hypothesis or model.
c_i	Class label
$\mathbf{x}$	Feature vector
n	No. attributes/dimensions

Table 10. Notation for HMMs.

Symbol	Meaning
$S = \{s_1, s_2, \ldots s_n]$	Individual states
N	Number of states
π	Initial probabilities
a_{ij}	Transition probabilities
o_{ij}	Observation
γ_{ij}	Occupancy probability
λ	The parameters of the HMM
α	Forward variable
β	Backward variable

Table 11. List of abbreviations.

Abbreviation	Meaning
ANN	Artificial Neural Network
ASR	Automated Speech Recognition
BN	Bayesian network
CIE	Commission Internationales de l'Eclairage
CPD	Conditional Probability Distribution
DCT	Discrete Cosine Transform
DGM	Directed Graphical Model
EM	Expectation Maximization
GMM	Gaussian Mixture Model
GM	Graphical Model
GPM	Gaussians Per Mixture
HMM	Hidden Markov Model
HMT	Hidden Markov Tree
HVS	Human Visual System
I.I.D.	Independent identically distributed
MLE	Maximum Likelihood Estimation
MRF	Markov Random Field
MMRF	Markov Mesh Random Field
MAP	Maximum A Posteriori
PCA	Principal Component Analysis
PDF	Probability Density Function
STL	Standard Template Library

Bibliography

[1] Shatford, ”Analyzing the Subject of a Picture: A Theoretical Approach”, Cataloging and Classification Quarterly, Vol 6(3), Spring 1986.

[2] R. W. Picard “A Society of Models for Video and Image Libraries”, 1996, IBM Systems Journal.

[3] Milind R. Naphade and Thomas S. Huang “Extracting Semantics From Audiovisual Content: The Final Frontier in Multimedia Retrieval”, 2002 IEEE.

[4] R.Fergus, P.Perona, and A.Zisserman. “Object class recognition by unsupervised scale-invariant learning”. In Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2003.

[5] K. Barnard and D. Forsyth. “Learning the semantics of words and pictures” In International Conference on Computer Vision, volume 2, pages 408--415, 2001.

[6] B. Moghaddam and A. Pentland. “Probabilistic visual learning for object representation” IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI19 (7):696--710, July 1997.

[7] Scientific American Frontiers Program #1502 "Cars That Think" AIRDATE: January 26, 2005.

[8] Hill, D.A., and Leckie, D.G. (eds.) 1999. “Automated interpretation of high spatial resolution digital imagery for forestry. Proceedings of International Forum”, Feb. 10-12, 1998. Pacific Forestry Centre, Victoria, British Columbia, Canada.

[9] Porikli, F.; Shao, J.; Maehara, H., "Heli-Tele: Road Extraction from Helicopter Video", IAPR Conference on Machine Vision Applicaitons (MVA), May 2005 (MVA 2005).

[10] Porikli, F.M., "Trajectory Distance Metric Using Hidden Markov Model Based Representation", European Conference on Computer Vision (ECCV), May 2004 (EVVC 2004, PETS 2004, TR2004-030).

[11] Viola, P.; Jones, M.J.; Snow, D., "Detecting Pedestrians Using Patterns of Motion and Appearance", IEEE International Conference on Computer Vision (ICCV), Vol. 2, pp. 734-741, October 2003 (IEEE Xplore, TR2003-090).

[12] Proc. of the ARPA IU Workshop, Morgan Kaufmann, San Francisco, 1994.

[13] L. Matthies, A. Kelly, and T. Litwin, “Obstacle Detection for Unmanned Ground Vehicles: A Progress Report,” Proc. Int’l Symp. Robotics Research, Int’l Federation on Robotics Research, 1995.

[14] Yu-Bin Yang, Shifu Chen, Zhi-Hua Zhou, Hui Lin, Yukun Ye: “An Intelligent Medical Image Understanding Method Using Two-Tier Neural Network Ensembles”, IEA/AIE 2005: 616-618.

[15] M. Lievin, N. Hanssen, P. Zerfass, and E. Keeve, “3D Markov random fields and region growing for interactive segmentation of MR data,” in Proc. Medical Image Computing Computer-Assisted Intervention, Utrecht, The Netherlands, Oct. 2001, pp. 1191–1192.

[16] Generic Media Framework for Interactive Television, http://www.gmf4itv.net/

[17] Cardoso, Bernardo;de Carvalho, Fausto;Fernandez, Gabriel;Huet, Benoit;Jiten, Joakim;Lopez, Alejandro;Mérialdo, Bernard;Neuschmied, Helmut;Noe, Miquel;Serras Pereira, David;Thallinge, Georg, “Personalization of interactive objects in the GMF4iTV project TV'04”, 4th Workshop on Personalization in Future TV held in conjunction with Adaptive Hypermedia 2004 , August 23, 2004, Eindhoven, The Netherlands.

[18] B. Huet, J. Jiten, B. Mérialdo, "Personalization of hyperlinked video in interactive television", ICME 2005, IEEE International Conference on Multimedia & Expo, July 6-8, 2005, Amsterdam, The Netherlands

[19] Freeman, W.T., Anderson, D.B., Beardsley, P.A., Dodge, C.N., Roth, M., Weissman, C.D., Yerazunis, W.S., Kage, H., Kyuma, K., Miyake, Y.; Tanaka, K., "Computer Vision for Interactive Computer Graphics", IEEE Computer Graphics and Applications, Vol. 18, Issue 3, pp. 42-53, May-June 1998 (IEEE Computer Graphics and Applications).

[20] Joo-Hwee Lim, Jesse S. Jin: “Semantics Discovery for Image Indexing”, ECCV (1) 2004: 270-281.

[21] Pierre Moreels, Michael Maire, Pietro Perona: Recognition by Probabilistic Hypothesis Construction. ECCV (1) 2004: 55-68

[22] Samaria, Ferdinando and Fallside, Frank, “Face Identification and Feature Extraction Using Hidden Markov Models”, Image Processing: Theory and Applications, Elsevier, 1993, pp 295-298.

[23] Mitchell. Machine Learning. McGraw-Hill, 1997

[24] A.L. Ratan, O. Maron, W.E.L. Grimson and T. Lozano-Perez. “A framework for learning query concepts in image classification”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 423-431, 1999.

[25] M.R. Naphade et al., "Probabilistic multimedia objects (multijects): A novel approach to video indexing and retrieval in multimedia systems", in Proc. of ICIP'98, 1998, vol. 3, pp. 536--540.

[26] Djeraba, C., “Content-based multimedia indexing and retrieval”, IEEE Multimedia 9 (2002) 52—60.

[27] Vernon, D. and Furlong D. 1992. "Relativistic Ontologies, Self-Organization, Autopoiesis, and Artificial Life: A Progression in the Science of the Autonomous. Part II - A Scientific Development", Proc. of the ESPRIT Workshop on Autopoiesis and Perception, Dublin City University, (22 pages).

[28] Hunter, J., "Adding Multimedia to the Semantic Web - Building an MPEG-7 Ontology", Proceedings of the First Semantic Web Working Symposium (SWWS), Stanford, USA (2001) 261-281.

[29] Wielinga B.J., Schreiber A.Th., Wielemaker J., "Semantics for Multimedia on the Web", Position paper for The Semantic Web Workshop SWWS, Stanford, USA, July 2001.

[30] Y. Rui, T. S. Huang, and S.-F. Chang, "Image retrieval: Past present and future", Journal of Visual Communication and Image Representation, 10:1--23, 1999.

[31] Ramesh Jain, "Out-of-the-Box Data Engineering - Events in Heterogeneous Data Environments," icde, p. 8, 19th International Conference on Data Engineering (ICDE'03), 2003.

[32] A. Torralba. "Contextual priming for object detection", Intl. J. Computer Vision, 53(2):153–167, 2003.

[33] J. Li and J. Z. Wang, "Automatic linguistic indexing of pictures by a statistical modeling approach", IEEE Trans. on Pattern Analysis and Machine Intelligence, 25(10), 2003.

[34] J. Bilmes, "A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian Mixture and Hidden Markov Models.," Tech. Rep., University of Berkeley, 1997.

[35] Probabilistic Modeling and Reasoning: c David Barber 2003, 2004. Course Mixture models.

[36] E. Levin and R. Pieraccini, "Dynamic planar warping for optical character recognition", Proceedings of ICASSP, 149-152, 1992.

[37] 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.

[38] LE. Baum and T. Petrie, "Statistical Inference for Probabilistic Functions of Finite State Markov Chains", Annual Math., Stat., 1966, Vol.37, pp. 1554-1563.

[39] J. Li, A. Najmi, and R. M. Gray, "Image classification by a two-dimensional hidden Markov model", IEEE Trans. Signal Processing, vol. 48, no. 2, pp. 517–533, 2000.

[40] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", Journal of the Royal Statistical Society, 39(1):1{38, 1977.

[41] Leonard E. Baum and J. A. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model

for ecology", Bulletin of the American Mathematical Society, 1976, 73:360-363.

[42] X. D. Huang, Y. Ariki, and M. A. Jack, "Hidden Markov Models for Speech Recognition", Edinburgh University Press, 1990.

[43] L. Rabiner and B. H. Juang, Fundamentals of Speech Recognition, Prentice Hall, Englewood Cliffs, NJ, 1993.

[44] M. A. Mohamed and P. Gader, "Generalized Hidden Markov Models-Part I: Theoretical Frameworks", IEEE Transaction on Fuzzy Systems, February, 2000, Vol.8, No.1, pp. 67-81.

[45] Othman, H.; Aboulnasr, T., "A simplified second-order HMM with application to face recognition", IEEE International Symposium on Circuits and Systems, Volume 2, 6-9 May 2001 Page(s):161 – 164.

[46] T. Kohonen, G. Barna, and R. Chrisley, "Statistical pattern recognition with Neural Networks: benchmarking studies", IEEE Int, Conf. Neural Networks, pp. I-61-68, July 1988.

[47] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, Classification and Regression Trees, Chapman & Hall, 1984.

[48] 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.

[49] Perronnin, F.; Dugelay, J.-L.; Rose, K., "Deformable face mapping for person identification", International Conference on Image Processing, Volume 1, 14-17 Sept. 2003 Page(s):I - 661-4.

[50] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition", In Proceedings, CVPR, pages 994--999. IEEE Press, 1997.

[51] M. Brand, "Coupled hidden markov models for modeling interacting processes", Technical Report 405, MIT Media Lab Perceptual Computing, June 1997.

[52] S. Fine, Y. Singer, N. Tishby, "The hierarchical hidden Markov model: Analysis and applications," Machine Learning 32(1998).

[53] M. S.E.Levinson, L.R.Rabiner, "An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition," Bell Syst.Tech.J., vol. 62, pp. 1035--1074, 1983.

[54] B. H. Juang, Stephen E. Levinson, and M. M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of Markov chains", IEEE Transactions on Information Theory, IT-32(2):307--309, March 1986.

[55] R. L. Dobrushin, "The description of a random field by means of conditional probabilities and conditions of its regularity," Theory Prob. Appl., vol. 13, pp. 197-224, 1968.

[56] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy

minimization methods for Markov random fields", In Ninth European Conference on Computer Vision (ECCV 2006), volume 2, pages 19-26, Graz, Austria, May 2006.

[57] K. Abend, T. J. Harley, and L. N. Kanal, "Classification of binary random patterns," IEEE Trans. Inform. Theory, vol. IT-11, no. 4, pp. 538-544, Oct. 1965.

[58] S.-Z. Li, "Markov random field models in computer vision", In Proc. of the IEEE European Conf. on Computer Vision (ECCV), volume B, pages 361-370, 1994.

[59] P. Pérez and F. Heitz, "Restriction of a Markov random field on a graph and multiresolution statistical image modeling," IEEE Trans. Inform. Theory, vol. 42, pp. 180--190, Jan. 1996.

[60] Besag, J., "Spatial interaction and the statistical analysis of lattice systems (with discussion)". Journal of the Royal Statistical Society, Series B 36, 192-236, 1974.

[61] S. Kuo and O. Agazzi., "Keyword spotting in poorly printed documents", IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 16(8):842{848, Aug 1994.

[62] O. Agazzi, S. Kuo, E. Levin, and R. Pieraccini, "Connected and degraded text recognition using planar hidden Markov models", In Proc. of the IEEE Int. Conf. on Acoustics Speech and Signal Processing (ICASSP), volume 5, pages 113-116, 1993.

[63] K. Hallouli, L. Likforman-Sulem, and M. Sigelle, "A comparative study between decision fusion and data fusion in Markovian printed character recognition", In Proc. of the IEEE Int. Conf. on Pattern Recognition (ICPR), volume 3, pages 147-150, 2002.

[64] J. Baker (1979), "Trainable grammars for speech recognition", In J. J. Wolf and D. H. Klatt, editors, Speech communication papers presented at the 97th meeting of the Acoustical Society of America, pages 547–550, Cambridge, MA, June 1979. MIT.

[65] F. Jelinek, J. D. Lafferty, and R. L. Mercer, "Basic methods of probabilistic context free grammars", Technical Report RC 16374 (72684), IBM, Yorktown Heights, New York 10598. 1990.

[66] TREC Video Retrieval Evaluation, http://www-nlpir.nist.gov/projects/trecvid/

[67] Gupta, V.N.; Lennig, M.; Mermelstein, P.; Kenny, P.; Seitz, F.; O'Shaughnessy, D.; "Using phoneme duration and energy contour information to improve large vocabulary isolated-word recognition" Acoustics, Speech, and Signal Processing, 1991. ICASSP-91.

[68] F. Golshani, Y. Park, S. Panchanathan, "A Model-Based Approach to Semantic-Based Retrieval of Visual Information", SOFSEM 2002: 149-167.

[69] M. R. Naphade, and T. S. Huang, "Extracting Semantics from Aduiovisual Content: The Final Frontier in Multimedia Retrieval", IEEE Transactions on Neural Network, Vol. 13, No. 4, 793--810, 2002.

[70] J.Z. Wang, "Integrated Region-Based Image Retrieval", Dordrecht: Kluwer Academic, 2001.

[71] C. Carson and S. Belongie and H. Greenspan and J. Malik, "Blobworld: Image Segmentation Using ExpectationMaximization and Its Application to Image Querying," IEEE Tr. on Pattern Analysis and Machine Intelligence, Vol. 24, Number 8, pp. 1026--1038, 2002.

[72] K. Barnard and D. Forsyth, "Learning The Semantics of Words and Pictures," Proc. Int'l Conf. Computer Vision, vol 2, pp. 408-415, 2001.

[73] C. Bouman and B. Liu, "Multiple resolution segmentation of textured images," IEEE Transactions on Pattern Analysis and Machine Intelligence, 13, no. 2, pp. 99--113, 1991.

[74] L. Lucchese and S. Mitra, "Color image segmentation: A state-of-the-art survey", Proc. of the Indian National Science Academy (INSA-A), 67(2):207{221, 2001.

[75] S. Kumar and M. Hebert, "A Hierarchical Field Framework for Unified Context-Based Classification," Proc. ICCV, October, 2005.

[76] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", In Proc. ICML, 2001.

[77] Allen Gersho and Robert M. Gray, "Vector Quantization and Vector Compression", Kluwer Academic Publication, 1992.

[78] Y. Meyer, "Wavelets Algorithms and Applications", Philadelphia, PA: SIAM, 1993.

[79] Jia Li, Robert M. Gray, "Image Classification Based on a Multiresolution Two Dimensional Hidden Markov Model", ICIP (1) 1999: 348-352.

[80] M. Basseville, A. Benveniste, K. C. Chou, S. A. Golden, R. Nikoukhah, and A. S. Willsky, "Modeling and estimation of multiresolution stochastic processes", IEEE Trans. Inform. Theory, vol. 38, pp. 766–784, Mar. 1992.

[81] C. A. Bouman and M. Shapiro, "A multiscale random field model for Bayesian image segmentation", IEEE Trans. Image Processing, vol. 3, pp. 162–177, Mar. 1994.

[82] W. T. Freeman and E. C. Pasztor, "Learning low-level vision. in Proc.7th Int. Conf. Computer Vision", Corfu, Greece, Sept. 1999.

[83] M. S. Crouse, R. D. Nowak, and R. G. Baraniuk, "Wavelet-based statistical signal processing using hidden Markov models", IEEE Trans. Signal Processing, vol. 46, pp. 886–902, Apr. 1998.

[84] R. Nowak, "Multiscale hidden Markov models for Bayesian image analysis," in Bayesian Inference in Wavelet Based Models (B. Vidakovic and P. Muller, eds.), Lecture Notes in Statistics 141, Springer-Verlag, 1999.

[85] H. Choi and R. G. Baraniuk, "Image segmentation using wavelet-domain classification," in Proceedings of SPIE, (Denver, CO), July 1999.

[86] P. F. Felzenszwalb , D. P. Huttenlocher, “Image Segmentation Using Local Variation”, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Rec-ognition, p.98, June 23-25, 1998.

[87] M. Nishida and T. Kawahara, “Unsupervised Speaker Indexing Using Speaker Model Selection based on Bayesian Information Criterion,” Proc. ICASSP, Vol. 1, pp. 172-175, 2003.

[88] A Wavelet Tour of Signal Processing, by Stéphane Mallat; ISBN : 0-12-466606-X; Academic Press, 1999.

[89] C. E. Shannon,"Coding theorems for a discrete source with a fidelity criterion," IRE Nat. Conv. Rec., pp. IV-142-163, March 1959.

[90] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, Angela Y. Wu, “An Efficient k-Means Clustering Algorithm: Analysis and Implementation”, IEEE Trans. Pattern Anal. Mach. Intell. 24(7): 881-892 (2002).

[91] Han, J., Kamber M.: Data mining: Concepts and Techniques. San Francisco, Morgan Kaufman Publishers, 2001, s. 335-393. ISBN 1-55860-489-8.

[92] Y. Linde, A. Buzo, and R. M. Gray, “An Algorithm for Vector Quantizer Design”, IEEE Transactions on Communications, pp. 702--710, January 1980.

[93] P.J. Rousseeuw and A.M. Leroy, “Robust regression and outlier detection”, Wiley, New York, 1987.

[94] M. J. Swain and D. H. Ballard, "Color indexing," International Journal of Computer Vision, vol. 7, no. 1, pp. 11--32, November 1991.

[95] J. Huang, S. R. Kumar, M. Mitra, W. J. Zhu, and R. Zabih, “Image indexing using color correlograms”, Proc. 16th IEEE Conf. on Computer Vision and Pattern Recognition, pp. 762–768, 1997.

[96] Y.Q. Chen, T.S. Huang, and Y. Rui, "JPDAF Based HMM for Real-Time Contour Tracking," Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition, 2001.

[97] A. Baumberg and D. Hogg, “Learning flexible models from image sequences”, In ECCV, volume 1, pages 299--308, Stockholm, Sweden, May 1994.

[98] N. Paragios and R. Deriche, “A PDE-based Level Set Approach for Detection and Tracking of Moving Objects”, Technical Report 3173, INRIA, France, May 1997.

[99] D. Koller, J. W. Weber, and J. Malik, “Robust multiple car tracking with occlusion reasoning”, in European Conference on Computer Vision, pp. 189--196, LNCS 800, springer-Verlag, May 1994.

[100] S. Lefevre, E. Bouton, T. Brouard, N. Vincent, "A new way to use hidden Markov models for object tracking in video sequences", IEEE International Conference on Image Processing (ICIP), Volume 3, page 117-120 - September 2003.

[101] Kato, J.[Jien], Watanabe, T.[Toyohide], Joga, S.[Sébastien], Liu, Y.[Ying], Hase, H., “An HMM/MRF-based stochastic framework for robust vehicle

tracking", ITS(5), No. 3, September 2004, pp. 142-154. IEEE Abstract. IEEE Top Reference. 0501.

[102] Joshi, D. Jia Li Wang, J.Z., "A computationally efficient approach to the estimation of two- and three-dimensional hidden Markov models", Image Processing, IEEE Transactions on, July 2006.

[103] Lievin, M., Hanssen, N., Zerfass, P., and Keeve, E. 2001, "3D Markov Random Fields and Region Growing for Interactive Segmentation of MR Data", In Proceedings of the 4th international Conference on Medical Image Computing and Computer-Assisted intervention (October 14 - 17, 2001). W. J. Niessen and M. A. Viergever, Eds. Lecture Notes In Computer Science, vol. 2208. Springer-Verlag, London, 1191-1192.

[104] Alexandrov, V. and Gerstein, M., "Using 3D Hidden Markov Models that explicitly represent spatial coordinates to model and compare protein structures", BMC Bioinformatics, 5:2, 2004.

[105] Ching-Yung Lin, Belle L. Tseng, and John R. Smith, "Video collaborative annotation forum: Establishing ground-truth labels on large multimedia datasets", In Proceedings of the TRECVID 2003 Workshop, 2003.

[106] Georges M. Quénot, "Trec-10 shot boundary detection task: CLIPS system description and evaluation", In The 10th Text REtrieval Conference (TREC), 2000.

[107] Fabrice Souvannavong, Bernard Merialdo, and Benoit Huet, "Latent semantic indexing for video content modeling and analysis", In The 12th Text Retrieval Conference (TREC), 2003.

[108] Jiten, Joakim; Souvannavong, Fabrice; Mérialdo, Bernard; Huet, Benoit, "Eurecom at TRECVid 2005: extraction of high-level features", TRECVid 2005, TREC Video Retrieval Evaluation, November 14, 2005, USA.

[109] J. Gauvain, L. Lamel, and G. Adda, "The limsi broadcast news transcription system". Speech Communication, 37(1-2):89--108, 2002.

[110] I. Daubechies, Ten Lectures on Wavelets, Capital City Press, 1992.

[111] Joo-Hwee Lim, Jesse S. Jin, "Unifying local and global content-based similarities for home photo retrieval", ICIP 2004: 2371-2374.

[112] W.B. Pennebaker. JPEG still image data compression standard. Van Nostrand Reinhold, 1993.

[113] Wyszecki, G., And Stiles, W. S., "Color Science: Concepts and Methods", QuantitativeData and Formulae, 2nd Edition. JohnWiley & Sons, Inc., New York, New York, 1982.

[114] H. Tamura, S. Mori, T. Yamawaki, "Texture features corresponding to visual perception", IEEE Trans. Systems Man Cybernet SMC8 (6) (1978) 00.

[115] Y. Rui and T. S. Huang, "Image Retrieval: Current Techniques, Promising, Directions and Open Issues", Journal of Visual Communication and Image Representation, Vol. 10, No. 4, April 1999.

[116] C. Liu and H. Wechsler, “Gabor feature based classification using the enhanced Fisher linear discriminant model for face recognition”, IEEE Trans. on Image Processing (IP), 11(4):467-476, Apr 2002.

[117] Zhang, D., Wong, A., 2000, “Content-based image retrieval using Gabor texture features, In: Proc. IEEE Pacifin-Rim Conf. on Multimedia, Sydney, Australia, December 13-15, pp. 392-395.

[118] Tao, H. and T. Huang, "Color Image Edge Detection using Cluster Analysis", IEEE International Conference On Image Processing, pp. 834-836, California, 1997.

[119] Wang J., Yang W.J. & Acharya R. (1997), “Color clustering techniques for color content-based image retrieval from image databases”, In Proc. IEEE Conference on Multimedia Computing and Systems, 442-449.

[120] M. Miyahara and Y. Yoshida, “Mathematical Transform of (R, G, B) Color Data to Munsell (H, V, C) Color Data,” SPIE Visual Communications and Image Processing ‘88, Vol. 1001, pp. 650-657.

[121] Y. Rui, T. S. Huang, and S.-F. Chang, “Image retrieval: Past present and future”, Journal of Visual Communication and Image Representation, 10:1--23, 1999.

[122] Markus A. Stricker and Markus Orengo, "Similarity of Color Images", SPIE Proceedings Vol. 2420, 1995.

[123] D. A. White and R. Jain, "Similarity indexing: Algorithms and performance", Proc. SPIE Vol.2670, San Diego , 1996.

[124] http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/

[125] http://www.cs.cornell.edu/dali/