Multimodal Structuring of Tennis Videos using Segment Models

Emmanouil Delakis

N° d'ordre: 3463

THÈSE

Présentée

devant l'Université de Rennes 1

pour obtenir

le grade de : Docteur de l'Université de Rennes 1
Mention Informatique

par

Emmanouil Delakis

Équipe d'accueil : TexMex - IRISA
École Doctorale : Matisse
Composante universitaire : IFSIC

Titre de la thèse :

Structuration multimodale des vidéos de tennis en utilisant des modèles segmentaux

Soutenue le 23 octobre 2006 devant la commission d'examen

M. :	Patrick	Bouthemy	Président
MM. :	Petros	Maragos	Rapporteurs
	Régine	André-Obrecht	
MM. :	Guillaume	Gravier	Examinateurs
	Christophe	Garcia	
	Patrick	Gros	

Acknowledgements

I would like to express my deep gratitude to my advisors Patrick Gros and Guillaume Gravier for all their valuable guidance in these three last years. When I needed help and fresh thinking, I always found their office open to enter in and discuss. I am particularly thankful for their large contribution during the publication stage of this work, in order to create well-structured and readable texts. Finally, I am especially grateful to them for translating in real french my ≪french≫ synthèse.

I would also like to thank the other members of the dissertation committee: Régine André-Obrecht – whose idea of using Segment Models for multimedia processing was inspiration for this work – Petros Maragos, Patrick Bouthemy and Christophe Garcia, for sparing time and for their insightful comments. In particular, many personal thanks to Christophe who, despite he left IRISA for other lands in the first year of my dissertation, did not stop being interested in.

Contents

List of Figures

List of Tables

CHAPITRE A

Synthèse en français

A.1 Introduction

L'annotation automatique de documents vidéos est un outil puissant pour gérer des grandes bases de données vidéos ou, plus récemment, pour le développement de produits grand public sophistiqués qui satisfont les besoins de haut niveau de l'utilisateur, comme l'extraction de moments clés d'une vidéo. On peut accomplir cette tâche en utilisant des modèles explicites faits à la main, et ainsi dépendants du domaine, qui donnent de bons résultats dans quelques cas [16]. Mais il a vite été clair que nous avons besoin de façons plus efficaces de rapprocher les besoins de haut niveau de l'utilisateur d'un coté et les caractéristiques vidéo de bas niveau que nous avons de l'autre coté, comme les histogrammes d'image ou l'excitation sonore. Pout atteindre ce but, un point clé est la mise au point d'une manière efficace de représenter le contenu des vidéos [95]. Les Modèles de Markov Cachés [81] (HMMs) constituent une approche statistique puissante pour modéliser le contenu d'une vidéo et peuvent être utilisés comme un analyseur syntaxique statistique de la vidéo [108], comme cela est fait dans le domaine de la reconnaissance de la parole.

Les modèles markoviens ont été utilisés pour l'analyse de structure d'émissions de tennis. Dans ce type de vidéo, les règles de jeu aussi bien que les règles de production aboutissent à un document structuré. Dans un travail précédent, des HMMs ergodiques ou hiérarchiques [49] ont été utilisés pour faire l'analyse syntaxique de cette structure et segmenter les données vidéo brutes en scènes reconnaissables par des humains. La table des matières d'une vidéo peut alors être construite automatiquement.

La fusion multimodale avec des HMMs est généralement effectuée par fusion concaténative d'observations qui fait l'hypothèse de caractéristiques homogènes et synchrones. Cependant, les deux modalités image et son sont échantillonnées à des fréquences différentes. En outre, le contenu visuel suit les règles de production tandis que le contenu sonore contient des sons bruts du court, mélangés avec le discours du commentateur. Il y a ainsi, premièrement, un certain degré d'asynchronisme entre des caractéristiques auditives et visuelles et, deuxièmement, des modèles temporels différents. Dans cette étude, nous présentons l'indexation des vidéo avec des Modèles Segmentaux [72] (SMs), comme modèles qui permettent une fusion multimodale plus efficace et nous fournissons une comparaison expérimentale avec la fusion basée sur les HMMs. Avec les SMs, les contraintes de synchronisation entre les modalités peuvent être repoussées aux frontières des scènes. Nous sommes alors libres de traiter, au sein de chaque scène, chaque modalité avec son modèle et sa fréquence d'échantillonnage natifs.

Par ailleurs, des ressources textuelles comme les indications de score et les statistiques du jeu transmettent une information importante sur l'évolution de jeu qui doit être exploitée au mieux. Les difficultés entrent du fait que ces informations peuvent apparaître longtemps après l'événement de jeu correspondant, à l'extérieur des frontières de la scène, ou peuvent être absentes. L'utilisation des indications de score comme une caractéristique supplémentaire peut être ainsi de contribution limitée. Nous avons proposé au lieu de cela une variante originale du décodage de Viterbi, appelée recherche de Viterbi guidée par le score, qui utilise les indications de score pour piloter le décodage de Viterbi. L'alignement des étiquettes aux événements de jeu correspondants est laissé comme partie du problème d'optimisation, tandis que la cohérence avec l'évolution de jeu est satisfaite lors du décodage.

Cette thèse est organisée comme suit. Le travail relatif à la fusion multimodale basée sur les HMMs est expliqué dans la section A.2. Des définitions de base sur le problème abordé sont dans la section A.3. L'extraction des caractéristiques est discutée dans la section A.4. Dans la section A.5 nous voyons comment le contenu visuel est modelisé par les HMMs et les SMs, avec une topologie ergodique ou hiérarchique. L'intégration audiovisuelle avec ces modèles est discutée dans la section A.6. On donne l'intégration de l'affichage du score dans la section A.7. On fournit des détails sur l'estimation des paramètres des modèles dans la section A.8 et les résultats expérimentaux dans la section A.9. Finalement, la section A.10 conclut cette étude.

A.2 Travail précédent

Pour une revue détaillée des aspects divers du problème de l'indexation multimodale de la vidéo, le lecteur intéressé pourra se référer aux état de l'art paru dans [16, 106, 95, 17]. Dans cette section nous nous concentrerons sur les HMMs, qui sont largement utilisés

pour exploiter l'aspect temporel des données vidéo. En effet, selon le genre de la vidéo et les règles de production, les événements vidéo arrivent avec un ordre temporel qui permettra de trouver finalement l'étiquette sémantique. Les HMMs fournissent une structure statistique puissante pour traiter des données séquentielles et sont ainsi un candidat naturel pour modéliser des dépendances temporelles dans la vidéo.

Wolf [108] a présenté les HMMs comme un analyseur statistique de la syntaxe d'un document vidéo. On considère les caractéristiques vidéo de bas niveau comme les observations d'un processus caché Markovien stochastique qui représente la syntaxe de la vidéo. L'algorithme de Viterbi est alors employé pour récupérer la syntaxe, étant donné le modèle vidéo (c'est-à-dire, les probabilités de transition et d'observation). Dans ce contexte, on peut considérer le problème d'extraction de syntaxe vidéo comme semblable à celui de la reconnaissance de la parole, où nous essayons d'extraire la transcription (la syntaxe de la vidéo, dans le cas d'indexation de vidéo) d'un texte parlé (le fichier vidéo, respectivement). Comme quelques genres de vidéos présentent une structure hiérarchique inhérente, le HMM Hiérarchique (HHMM) [28] peut être utilisé pour récupérer la hiérarchie. Les HHMMs ont été utilisé pour l'analyse des vidéos des sports [110] ou éducatives [78].

Un scénario simple d'intégration multimodale avec les HMMs est d'exécuter la classification ou l'indexation basée sur chaque modalité indépendamment et ensuite de fusionner les probabilités pour prendre la décision finale. On a proposé cet type de fusion, appelée fusion tardive, dans [42] entre autres.

Une deuxième approche de l'intégration multimodale basée sur les HMMs est la fusion précoce, i.e., la concaténation des caractéristiques de toutes les modalités rassemblées dans un super-vecteur d'observations. Ce type de fusion, étant un choix populaire dans la communauté d'indexation de vidéo en raison de sa simplicité, a été utilisé pour la segmentation vidéo [12, 7], la détection de dialogues humains [2] et la classification d'émissions [42, 106, 24, 26, 44] en utilisant des caractéristiques visuelles et sonores. La fusion précoce a aussi été largement étudiée dans le domaine de la reconnaissance audiovisuelle de la parole [79]. Une tâche nécessaire avant la fusion précoce est la conversion des fréquences d'échantillonnage, qui sont toujours dépendants des données. La concaténation des caractéristiques visuelles (d'habitude échantillonnées à 25 images par seconde) aux auditives (d'habitude échantillonnées à 100 Hz) peut être faite par interpolation [79], tandis que la conversion opposée peut se faire par moyennage [7]. Néanmoins, dans beaucoup d'approches de l'indexation vidéo, les caractéristiques de grands segments vidéo (comme un plan) sont rassemblées et puis pré-classifiées pour former des descripteurs. Ainsi, on contourne artificiellement le problème de la conversion des fréquences d'échantillonnage.

L'hypothèse sous-jacente cependant à ce type de fusion est que toutes les modalités doivent être synchrones et suivre la même topologie, ce qui n'est pas vrai en général.

Pour remédier à ce problème de synchronisation forcée entre les modalités et augmenter les capacités de modélisation, on a proposé un certain nombre de variantes HMM dans les communautés de la reconnaissance audiovisuelle de la parole et de l'indexation vidéo. Dans les HMMs multiflux [13], chaque modalité (ou flux) est modélisée par des HMMs indépendants qui sont synchronisés en des points fixes. Dans les HMMs multiflux synchrones, les états eux-mêmes sont ces points. En pratique, ces modèles ne diffèrent pas des HMMs synchrones d'état, en dehors de l'hypothèse explicite d'indépendance des observations. Ce schéma tient compte de l'inclusion d'une pondération dépendante de l'état ou des mesures de fiabilité sur chaque modalité [33].

Dans les HMMs Multiflux asynchrones, les points de synchronisation sont étendus au-delà des états cachés, comme la fin des phonèmes en reconnaissance audiovisuelle de la parole. Entre ces points, on considère les flux indépendants on les modélise chacun par des HMMs unimodaux. L'idée clef est que les HMMs unimodaux peuvent suivre une topologie différente et aussi fonctionner sur une fréquence d'échantillonnage propre. Les probabilités produites par les HMMs unimodaux sont recombinées aux points de synchronisation par des produits de probabilité ou avec une autre fonction de recombinaison. Cependant, pour des raisons de convenance, pendant le décodage, les HMMs multiflux asynchrones ont été utilisés sous la forme d'un HMM produit en reconnaissance audiovisuelle de la parole [79]. Dans la pratique, les HMMs produit peuvent gérer un asynchronisme limité entre les modalités mais l'exigence d'une topologie unique et d'une seule fréquence d'échantillonnage persiste. En outre, quand plus de deux modalités sont fusionnées, la complexité globale du modèle devient intraitable.

Le HMM asynchrone (AHMM) [8] est une architecture HMM spécialement conçue pour modéliser des paires de flux légèrement désynchronisés, échantillonnés à des fréquences différentes. Les AHMMs traitent les deux flux en laissant le plus court être dilaté temporellement pour l'aligner au mieux avec le premier flux. La séquence la plus longue est produite classiquement comme dans tous les modèles Markoviens en entrant dans un état caché, en émettant un symbole d'observation, en passant ensuite dans un nouvel état caché, etc. La nouveauté des AHMMs est qu'à certains instants, l'état caché émet deux symboles d'observation, un de la séquence longue, comme normalement, et aussi le symbole d'observation correspondant de la séquence courte. Les instants des émissions doubles, c'est-à-dire l'alignement des deux flux, sont contrôlés par une nouvelle variable cachée. L'interprétation physique de cette variable cachée est qu'elle donne à chaque instant l'index temporel du flux court. Pendant le décodage Viterbi, la recherche est exécutée non seulement par tous les chemins possibles d'état cachés, mais aussi par tous les alignements possibles entre les deux flux. Les AHMMs ont été introduits en reconnaissance audiovisuelle de la parole, afin de représenter la désynchronisation due au bruit des deux flux. Ils ont été aussi utilisés par McCowan *et al.* [63] dans une tâche d'identification des actions de groupe dans des réunions sur des caractéristiques audio-

visuelles, comme la parole, l'intensité du discours, la position des têtes, etc. Les HMMs état-synchrones ont mieux fonctionné dans les expériences conduites que d'autres variantes d'HMMs comme les HMMs produits. Les résultats étaient encore améliorés par les AHMMs, qui peuvent facilement gérer des désynchronisations naturelles qui introduisent de la confusion aux HMMs état-synchrones.

En partageant avec les HMMs multiflux l'idée de points de synchronisation et l'utilisation interne de modèles indépendants, des architectures superposées de HMMs peuvent être construites. La vidéo est segmentée selon quelques points de synchronisation fixés a priori, par exemple à chaque fin de seconde ou aux frontières des plans. Les diverses modalités de chaque portion de vidéo sont alors traitées indépendamment, ce qui permet de respecter leur fréquences d'échantillonnage natif et des topologies différentes. Les résultats des HMMs ainsi obtenus sont enchaînés et donnés en entrée au processus Markovien de la couche suivante. Des architectures multicouches, appelées HMM Multicouche (LHMM), ont été utilisées par Olivier *et al.* [70] dans une tâche d'inférence d'activité de bureau. Des caractéristiques auditives et visuelles sont traitées et préclassifiées séparément par un ensemble de HMMs. Le résultat de ces classificateurs, rassemblés chaque seconde, sont donnés en entrée, avec des caractéristiques d'activité de clavier à la seconde (et dernière) couche de la hiérarchie, où l'activité est déduite en utilisant un autre ensemble de HMMs. Les auteurs ont comparé la fusion état-synchrone concaténative des HMMs à l'approche du LHMM sur un corpus de 60 minutes d'activité de bureau, démontrant la supériorité de leur approche. En plus de la fusion de flux asynchrones, les HMMs du plus haut niveau peuvent capturer des interactions et une sémantique de plus haut niveau, d'une manière semblable aux HHMMs. Dans [117], une approche HMM à deux couches a été utilisée pour attaquer un problème d'identification d'actions de groups similaire au problème abordé dans [63]. Le rôle de la première couche de HMMs est de modéliser les actions individuelles (c'est-à-dire, spécifiques aux personnes) sur des données audiovisuelles brutes, conjointement traitées par des AHMMs audiovisuels. On donne les sorties de la première couche, avec quelques autres caractéristiques spécifiques au groupe, à la deuxième couche, qui modélise la vidéo au niveau du groupe. L'utilisation d'une deuxième couche pour capturer des dynamiques de plus haut niveau a été aussi adoptée dans [111] pour la structuration des vidéos de football.

Nous verrons dans cette étude comment les problèmes mentionnés ci-dessus peuvent être résolus en utilisant des Modèles Segmentaux [72] qui unifient les HMMs multiflux et les HMMs Multicouches dans une nouvelle structure pour l'intégration multimodale. Premièrement, un modèle de durée est ajouté aux HMMs multiflux. Les points de synchronisation entre les modalités ne sont plus fixés, mais sont maintenant des variables du problème d'optimisation. Chaque modalité peut être échantillonnée à des fréquences différentes et modélisée par sa topologie native. Deuxièmement, les observations de

toutes les modalités entre les frontières de synchronisation sont associées à un état caché ≪multimodal≫ commun. Cet état caché correspond à un plus haut niveau sémantique et modélise la vidéo comme la deuxième couche des HMMs multicouches.

A.3 Définitions de base

Un certain nombre de caractéristiques arrivent invariablement dans chaque vidéo de tennis suite aux règles de jeu et au travail du producteur, avant la diffusion. Quand une action de jeu va commencer, par exemple, la caméra passe à une vue du court. Il est extrêmement rare, bien que toujours possible, de présenter l'action de jeu par une vue différente, comme une vue de côté. D'autre part, l'inactivité de jeu correspond d'habitude aux vues de non-court. Les rediffusions sont présentées avec un effet spécial, comme un fondu enchainé. Des rediffusions multiples aboutissent à un effet de ≪sandwich≫ des successions des fondus enchainés et des rediffusions. Finalement, les temps morts (i.e., quand le jeu est en pause) correspondent à la publicité, à de présentation des statistiques de jeu ou à des interviews.

Basé sur les caractéristiques de la vidéo de tennis, quatre types de scènes sont identifiées et servent comme composantes sémantiques de base du vidéo. Ils sont définis au dessus de la segmentation vidéo en plans, i.e., chaque scène est en fait une collection de plans successifs qui partagent un contenu sémantique homogène. Les scènes sont définies comme suit :

1. *Premier service manqué et échange.* Ce type de scène commence par un plan de jeu où un service manqué s'est produit. Un certain nombre des plans de non-jeu suivent jusqu'à ce qu'un échange normal ait lieu. Finalement, un nombre de nouveaux plans de non-jeu peuvent apparaitre (facultativement), jusqu'à ce qu'une nouvelle scène commence. Il y a aussi la possibilité d'une répétition de plusieurs services manqués avant l'échange.

2. *Échange.* La scène commence par un plan de jeu correspondant à un échange, suivi par un certain nombre de plans de non-jeu qui remplissent la scène, jusqu'à ce qu'une nouvelle scène commence.

3. *Rediffusion.* Les scènes de rediffusion commencent par un plan de fondu enchaîné et contiennent ensuite un certain nombre de successions de vues différentes du court et de fondus enchainés. Ils finissent avec des plans de non-jeu qui remplissent la scène jusqu'à ce qu'une nouvelle scène commence.

4. *Pause.* Les pauses peuvent être des publicités ou un temps mort de longue durée où une interview peut avoir lieu, ou bien les statistiques du jeu sont affichées.

Les pauses commencent au plan où les joueurs quittent le court. La publicité ou les interviews ont lieu et, finalement, des plans qui remplissent la scène peuvent apparaître à la fin.

Les deux premières scènes correspondent, en pratique, au même événement de jeu de base : l'échange. On leur a assigné des étiquettes différentes pour rendre le problème plus difficile. La définition de la scène de pause exige un raisonnement humain de haut-niveau pour être proprement détectée. Néanmoins, la trace temporelle d'une pause est unique et facilement détectée : la publicité est une longue succession de plan courts tandis que les interviews et les temps d'inactivité sont une collection de plans, mais d'extrêmement longue durée.

La structuration de la vidéo est définie comme le problème de la classification de chaque plan comme appartenant à une des quatre scènes définies ci-dessus, et aussi de la détection des frontières des scènes. Chaque frontière de plan dans la vidéo peut être une frontière de scène.

L'utilisation des informations issues des modalités multiples est un atout majeur pour mener cette tâche à bien. Typiquement un observateur humain peut résoudre le problème d'étiquetage et de segmentation de la vidéo en scènes en observant juste le contenu visuel, même si cela comporte un certain degré de difficulté. Cependant, l'inférence automatique basée uniquement sur des caractéristiques visuelles peut poser des problèmes. Aussi, des caractéristiques auditives et textuelles sont utilisées pour améliorer la performance en fournissant des indices supplémentaires.

A.4 Extraction des caractéristiques

Nous avons utilisé un corpus de 6 vidéos complètes de tennis, enregistrées entre 1999 et 2001 et aimablement fournies par l'INA[1]. Tous les matchs ont lieu sur un court d'intérieur, sauf un. Les parties des émissions précédant et suivant le match de tennis ont été manuellement enlevées des vidéos. Néanmoins, les programmes contiennent toujours des publicités et des interviews qui apparaissent de temps en temps pendant le match. La durée totale des vidéos est d'environ 15 heures. Les images des vidéos ont été extraites au flux MPEG brut, dans l'espace de couleurs RGB et avec une taille de 288×352 pixels. L'audio a été sous-échantillonné à 16 kHz. La moitié des vidéos (n'incluant pas le match d'extérieur) a été réservée dans cette thèse pour tester les systèmes. Un schéma approximatif du processus d'extraction des caractéristiques est montré à la figure A.1. Les flux visuel et auditif ont été traités séparément pour l'extraction de descripteurs visuels et auditifs. Les caractéristiques textuelles ont été extraites en utilisant comme base la segmentation vidéo.

[1] Institut national de l'audiovisuel, France. http://www.ina.fr

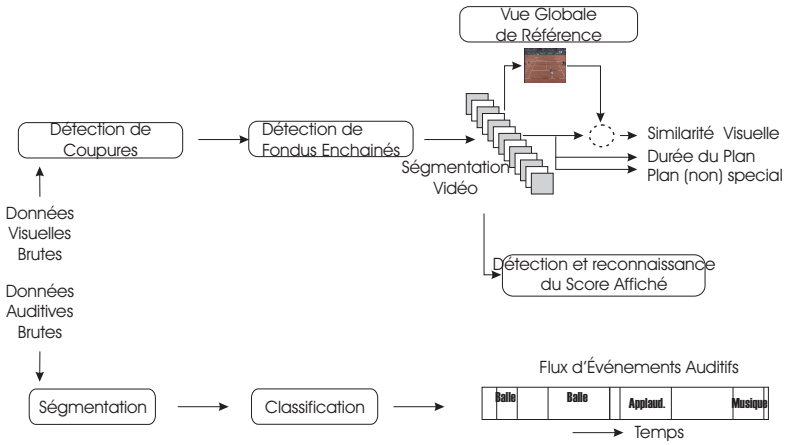


Figure A.1
Plan du processus d'extraction des caractéristiques.

A.4.1 Caractéristiques visuelles

Le flux visuel a été premièrement segmenté en plans, qui sont des grands segments vidéo de contenu visuel homogène. La segmentation vidéo implique premièrement la détection des coupures, puis les fondus enchainés sont détectées à l'intérieur de chaque plan individuellement. Les fondus enchainés sont des transitions progressives entre deux plans où le premier disparaît en fondu tandis que le deuxième apparaît progressivement. Dans les vidéos de tennis, comme dans d'autres émissions sportives, ils sont largement utilisés pour signaler le début et la fin des rediffusions. Les coupures et les fondus enchainés ont été détectés avec la méthode de [101] (méthode de choix adaptatif de seuil) en inspectant localement l'évolution des caractéristiques d'image pendant un fondu enchainé. Au final, 10775 coupures ont été détectées avec un nombre négligeable de fausses alarmes ou détections manquées. Les fondus enchainés, étant des transitions progressives, ont été laissées intactes par la détection des coupures parce qu'on utilise un seuil adaptatif. La détection des fondus enchainés a abouti à 1196 vraies détections dans les séquences d'apprentissage et de test ensemble. Sur le corps d'apprentissage (resp., de test), nous avons observé 3 (resp., 8) fausses alarmes et 10 (resp., 22) non détections. Étant donné le début et la fin des fondus enchainés, un nouveau type de plans est défini, appelés plans spéciaux.

La classe la plus intéressante de plans dans les vidéos de tennis est ceux qui montrent une vue globale du court (appelée vue globale). La plupart du temps et selon le style du producteur, la caméra est fixée sur une vue globale quand le jeu se déroule. Les vues globales sont ainsi des points de repère importants dans une vidéo de tennis. Dans [22] les vues globales sont détectées par de l'information de bord et la transformée de Hough. Une solution plus simple est d'utiliser des caractéristiques à base de couleur

car la couleur du court domine les vues globales. Un désavantage de l'utilisation de la couleur résulte du fait que cette couleur du court change de vidéo en vidéo ou peut même subir des modifications légères au sein même d'une vidéo en raison des conditions environnementales. Dans [119], une procédure semi-adaptative est suivie pour détecter les vues globales, avec la contrainte que la vue à reconnaître doit être proche des modèles initiaux.

Nous avons utilisé, par contre, une procédure entièrement automatique pour détecter les vues globales sans exiger une quelconque approximation antérieure de la couleur du court, telle que décrit dans [49]. La procédure est appliquée à chaque fichier vidéo individuellement, puisque chaque vidéo est caractérisée par sa propre couleur du court. Premièrement, chaque plan de la vidéo est représenté par une image clef, prise arbitrairement comme son image médiane. L'extraction des couleurs dominantes est alors appliquée à chaque image clef et les images dont la couleur dominante occupe moins de 70% de l'image entière sont étiquetées sans risque comme des vues non-globales. La vue globale de référence est définie comme l'image clef qui est la plus proche possible du centre d'inertie des images clefs restantes, dans un espace défini par une distance entre images (par exemple, à base d'histogrammes). Pourtant, dans l'ensemble des images clefs restantes il reste un certain nombre de vues non-globales qui ont passé l'étape de prefiltrage. En les considérant comme images aberrantes, la méthode des moindres carrés médians [84] a été utilisée pour extraire la vue globale de référence. À l'étape finale, chaque image clef de la vidéo est comparée à la vue globale de référence pour calculer le descripteur de la similarité visuelle. La distance à base d'histogramme utilisée inclut non seulement la couleur mais prend aussi les contours en compte pour être plus robuste aux légers changements de lumière/couleur dans la vidéo entière.

En plus de la similarité visuelle, l'ensemble des descripteurs visuels à base de plans est complété par la longueur de chaque plan et un indice binaire qui indique si le plan est un fondu enchainé ou non.

A.4.2 Caractéristiques sonores

Dans une vidéo de tennis, il peut être intéressant de détecter la présence des classes sonores comme les bruits de balle et les applaudissements. Les bruits de balle sont enregistrés pendant les échanges ou services et devraient compléter les caractéristiques visuelles pour une bonne discrimination entre vues globales d'action réelle de jeu et vues globales de non-jeu. Comme les applaudissements arrivent d'habitude après un échange, leur présence peut être utilisée pour détecter de tels événements. Finalement, la classe sonore ≪musique≫ peut transmettre de l'information utile puisqu'elle est caractéristique pendant des publicités. Ces classes sonores ont été automatiquement détectées dans la bande sonore comme décrit en détail dans [10]. Les Mel Frequency Cepstral Coefficients

(MFCCs) sont classiquement extraits de la bande sonore en utilisant une fenêtre glissante. Un vecteur de caractéristiques qui représente le signal est ainsi formé, comprenant des coefficients d'énergie et des dérivées du premier et seconde ordres. Le processus de détection est séparé en deux phases distinctes : segmentation de bande sonore en segments acoustiquement homogènes et classification de ces segments. Pendant la première phase, une segmentation brute avec une fenêtre de taille croissante est raffinée en fusionnant ou pas les segments avoisinants avec le Bayesian Information Criterion [100]. La détection de chaque classe sonore est effectuée indépendamment dans chaque segment et indépendamment pour chaque classe, comme on les a considérées indépendantes. Un test d'hypothèse binaire est exécuté dans chaque segment et pour chaque classe pour vérifier sa présence dans le segment. Ce test implique l'examen des probabilités produites par le modèle de la classe et celui de l'anti-classe, exprimés par des modèles de mélange de gaussiennes, appris sur des données étiquetées.

A.4.3 Caractéristiques textuelles

Les scores affichés à l'écran ont été manuellement extraits et reconnus. Néanmoins, ces données simulées sont une bonne approximation de ce qui pourrait avoir été automatiquement extrait par un outil de reconnaissance optique de caractères moderne (voir par exemple [109]). De plus, beaucoup de connaissances a priori spécifiques aux émissions de tennis peuvent être exploitées, plutôt que d'essayer de résoudre le problème dans sa généralité. Le score affiché apparaît à des positions fixes et est, d'habitude, encadré et bien distingué du fond pour être facilement visible. Les fausses alarmes peuvent ainsi être en grande partie éliminées. Finalement, il y a un ensemble fini d'étiquettes possibles, impliquant des chiffres (comme ≪0-30≫, ≪40-15≫, etc) ou quelques mots-clés prédéterminés (comme ≪break≫ ou ≪avantage≫), ce qui peut énormément faciliter l'identification des étiquettes.

A.5 Les modèles de vidéo

Nous décrivons en détails, dans cette section, comment les HMMs et les SMs sont appliqués au problème de la structuration de la vidéo. Nous considérons d'abord un scénario unimodal (flux visuel seul) pour mettre en évidence les différences conceptuelles entre ces deux modèles. L'intégration de l'audio et de l'affichage des scores affichés sont discutés dans les sections suivantes.

Des approches diverses du problème de modélisation du contenu visuel d'une émission de tennis sont présentées. Nous commençons par les transitions ergodiques entre les scènes. Avec la méthodologie des HMMs, chaque scène est modélisée à l'aide des HMMs sous-modèles, qui sont connectés pour fournir un HMM ergodique unique qui

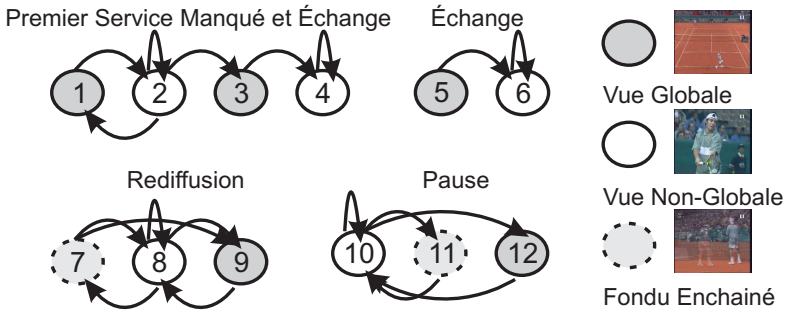


Figure A.2
Les états cachés du HMM. Les flèches dénotent les transitions principales à l'intérieur de chaque scène. Les probabilités de transition d'état sont évaluées à l'aide des données d'apprentissage.

modélise la vidéo comme une succession de plans. Avec la méthodologie des SMs, on considère chaque scène comme un segment et un SM ergodique est utilisé. Il modélise la vidéo comme une succession de scènes. Finalement, nous examinons l'incorporation d'informations structurelles sur les règles de tennis en utilisant des transitions hiérarchiques entre les scènes, tant pour le HMM que pour le SM.

A.5.1 Modèles de Markov cachés

Étant donné les définitions des scènes dans la section A.3, il est facile de déterminer le nombre d'états cachés et leur signification sémantique pour les sous-modèles HMMs. Le HMM pour la scène ≪Premier service manqué et échange≫ est illustré sur la figure A.2. L'état 1 représente la vue globale du service manqué. Quelques plans de vues non-globales suivent (état 2), jusqu'à ce que le joueur serve de nouveau. Si ce nouveau service est de nouveau manqué, une transition en arrière vers l'état 1 arrive. La transition pour l'état 3 représente qu'un échange normal a lieu. Les sous-modèles HMMs pour le trois scènes restantes sont définis de la même manière. Notez que les deux dernières scènes contiennent des vues globales sans jeu (états 9 et 12) que nous souhaitons séparer des vues globales de jeu (états 1, 3 et 5). Cela est presque impossible avec des caractéristiques visuelles seules car toutes les vues globales sont visuellement identiques. L'introduction de caractéristiques sonores dans la section suivante pourra aider à résoudre le conflit.

Finalement, les quatre sous-modèles HMMs sont connectés pour former un HMM unique avec 12 états cachés (comme représenté dans la figure A.2). Il modélise le contenu entier de la vidéo comme une succession de plans.

Le vecteur d'observations associé au HMM est simplement défini par l'ensemble des descripteurs visuels décrits dans la section A.4 :

$$O_t^{\mathrm{v}} = \left[o_t^c \; o_t^l \; o_t^d \right] \tag{A.1}$$

où o_t^c, o_t^l, o_t^d correspondent respectivement à la similarité visuelle, la durée du plan et le descripteur pour les plans spéciaux. On considère les caractéristiques comme indépendantes. La probabilité d'observation conditionnelle est alors :

$$P(O_t^{\mathrm{v}}|S_j) = P(o_t^c|S_j)P(o_t^l|S_j)P(o_t^d|S_j) \tag{A.2}$$

pour un état donné S_j, $1 \leq j \leq 12$.

Le décodage de Viterbi [81] implique la découverte de l'étiquette de l'état caché le plus probable pour chaque plan, étant donnés une séquence d'observations $O_{1:T}^{\mathrm{v}} = O_1^{\mathrm{v}} O_2^{\mathrm{v}} \dots O_T^{\mathrm{v}}$ de longueur T (c'est-à-dire, une séquence vidéo de T plans) et les paramètres du modèle. Dans le domaine logarithmique, ce problème se traduit par :

$$Q_{1:T}^* = \arg\max_{Q_{1:T}} \{ \ln P(Q_{1:T}) + \sum_{r \in \{c,l,d\}} \ln P(o_{1:T}^r|Q_{1:T}) \} \tag{A.3}$$

où $Q_{1:T}$ est la séquence des états cachés et où r se réfère aux caractéristiques de l'équation A.1. $Q_{1:T}^*$ est la séquence des états cachés optimale qui associe à chaque plan une des 12 étiquettes de la figure A.2. Il est facile ensuite de trouver les correspondances entre chaque étiquette de plan et une étiquette de scène et trouver les frontières des scènes, selon la définition du problème dans la section A.3.

A.5.2 Modèles segmentaux

La différence fondamentale entre SMs et HMMs est que, pour les SMs, un état caché est associé à une séquence complète d'observations $O_{1:l}$, appelée *segment*, au lieu d'une unique observation O_t. Chaque état caché i, dans les SMs, définit donc premièrement un modèle de durée $p(l|i)$ qui représente la longueur l du segment. De même que pour les valeurs des états cachés, l est stochastique. Deuxièmement, une distribution de probabilités d'émission sur un segment $p(O_{1:l}|l, i)$ est définie, conditionnée à la longueur du segment et à l'état caché. D'un point de vue génératif, on peut voir les SMs comme un processus markovien où un état caché émet une séquence d'observations dont la longueur est dirigée par un modèle de durée avant la transition à un autre état. La différence entre HMMs et SMs est illustrée par la figure A.3. À gauche, nous voyons ce qui se passe conceptuellement dans le cas des HMMs : à un moment donné le processus est dans un état donné et produit un symbole d'observation avant de passer ensuite à un autre état. À droite, nous voyons comment une séquence est produite par les modèles segmentaux. À un moment donné, le processus stochastique entre dans un état et reste là selon une probabilité définie par le modèle de durée de l'état. Une séquence d'observations est produite, au lieu d'une seule observation, selon une distribution conditionnée sur l'étiquette du segment. Le processus passe alors à un nouvel état avec une probabilité de transition, comme dans les HMMs, et ainsi de suite jusqu'à ce que la séquence complète des observations soit produite.

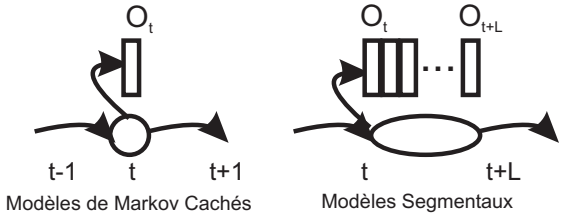


Figure A.3
La génération d'observations selon les modèles de Markov cachés (gauche) et les modèles segmentaux (droite). L'état caché dans les SMs produit L symboles d'observations, au lieu d'un seul.

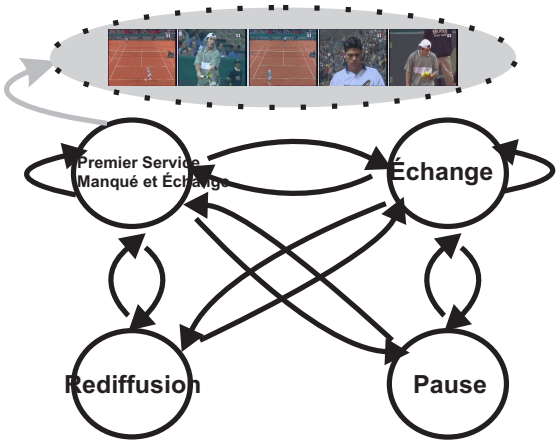


Figure A.4
Modélisation du contenu visuel comme une succession de scènes par les SMs.

Dans le cas des vidéos de tennis, nous pouvons concevoir le contenu d'une scène comme un segment. Le SM définit ainsi quatre états cachés, chacun correspondant à une des quatre scènes de la section A.3. On donne une illustration de la génération de la vidéo avec le SM sur la figure A.4. Le processus stochastique entre dans un état caché (ou, autrement dit, dans une scène) et émet plusieurs observations chacune correspondant à un plan. Il passe alors dans une nouvelle scène et ainsi de suite, jusqu'à ce que la vidéo complète soit produite. Il y a une structure ergodique complète, sauf en ce qui concerne les boucles pour les scènes de pause et la rediffusion qui ne sont pas permises. Ceci est dû au fait que, par définition, des rediffusions ou pauses répétées consécutives donnent en fait une seule rediffusion ou pause plus grande et unique. En outre, on ne permet pas de rediffusion après une pause et vice versa.

La longueur l du segment $O^{\mathrm{v}}_{1:l}$ est facilement définie comme le nombre de plans qu'il contient. Il reste à définir la distribution d'observations $P(O^{\mathrm{v}}_{1:l}|l, S_i)$ du segment, ce qui est plus compliqué qu'avec les HMMs car ce sont des séquences de caractéristiques qui

sont maintenant modélisées. Ce score de probabilité peut être fourni par un HMM λ_i, spécialisé dans la modélisation des observations de la scène i :

$$P(O^{\mathrm{v}}_{1:l}|l, S_i) \equiv P(O^{\mathrm{v}}_{1:l}|\lambda_i) = \sum_{Q_{1:l}} P(O^{\mathrm{v}}_{1:l}, Q_{1:l}|\lambda_i) \tag{A.4}$$

Le dernier terme est une somme sur tous les chemins d'états cachés du HMM et est calculé par la passe en avant de l'algorithme de Baum-Welch [81]. Le HMM λ_i fonctionne essentiellement comme un calculateur des scores d'observations et ne doit pas être confondu avec le HMM de la figure A.2.

Le décodage de Viterbi pour les SMs [72] implique la découverte des étiquettes *et* des frontières de scène les plus probables dans un problème d'optimisation étendu. Ce problème d'optimisation est défini dans le domaine logarithmique comme :

$$\begin{aligned}(L^*_{1:N^*}, Q^*_{1:N^*}) \quad = \quad & \arg\max_{L_{1:N}, Q_{1:N}} \{\ln p(Q_{1:N}) + \ln p(L_{1:N}|Q_{1:N}) \\ & + \ln p(O^{\mathrm{v}}_{1:T}|L_{1:N}, Q_{1:N})\}\end{aligned} \tag{A.5}$$

où T est le nombre total des plans de la vidéo, N^* est le nombre de scènes trouvées, $L^*_{1:N^*}$ donne la segmentation en scènes et $Q^*_{1:N^*}$ est la séquence trouvée d'états cachés la plus probable. Le terme $p(O^{\mathrm{v}}_{1:T}|L_{1:N}, Q_{1:N})$ est défini par l'équation A.4.

Ce problème est résolu via une extension directe de l'algorithme Viterbi pour les HMMs afin de prendre en compte la durée explicite des états, décrite en détail dans [81]. De manière intuitive, pendant le décodage de Viterbi pour les HMMs avec N états cachés, nous appliquons la maximisation suivante pour chaque instant t et pour chaque état i :

$$\delta_t(i) = \max_{1 \le j \le N} \delta_{t-1}(j) \mathrm{a}_{ji} \mathrm{b}_i(O_t) \tag{A.6}$$

où $\delta_i(t)$ est le score du meilleur chemin finissant à l'instant t dans l'état i, a_{ji} dénote la probabilité de transition de l'état j à i et $\mathrm{b}_i(O_t)$ dénote la probabilité d'observation. Pour les SMs, nous étendons la recherche aux instants précédents k pour prendre en compte des possibilités de segmentation multiples :

$$\delta_t(i) = \max_{1 \le j \le N, 1 \le k \le L_{\max}} \delta_{t-k}(j) p(O_{t-k+1:t}|k, i) p(k|i) \mathrm{a}_{ji} \tag{A.7}$$

où $p(k|i)$ dénote la probabilité de durée. Pour éviter du calcul non nécessaire, nous avons limité notre recherche des segmentations possibles à une fenêtre de $L_{\max}$ instants de temps (ou plans). Nous avons fixé cette valeur à 80 parce qu'aucune scène ne contient plus de 80 plans. En supposant que le calcul de $p(O_{t-k+1:t}|k, i)$ puisse être exécuté dans un temps $O(1)$, le décodage de Viterbi pour le SM donnera un coût approximatif de calcul $L_{\max}$ fois supérieur à celui des HMMs. Il augmente par conséquence d'un facteur constant. L'hypothèse, cependant, que le score du segment peut être calculé

immédiatement est généralement fausse. Pour le cas des HMMs par exemple, ce coût de ce calcul augmente avec la longueur du segment. Cependant, ce coût supplémentaire peut être évité en éliminant les calculs superflus. En effet, pendant la maximisation de l'équation A.7, les scores des séquences $O_{t:t}, O_{t-1:t} \dots O_{t-k+1:t}$ se sont évalués successivement. Le début de ces séquences est différent mais leur fin est commune. L'utilisation de la passe arrière de l'algorithme de Baum-Welch pour évaluer la probabilité de l'équation A.7, au lieu de la passe avant, permet la réutilisation du calcul du score de $O_{t-k+1:t}$ durant le calcul du score de $O_{t-k:t}$, etc. En mettant en oeuvre une telle réutilisation des calculs, l'hypothèse que les SMs sont approximativement de $L_{\max}$ fois plus coûteux que les HMMs tient toujours.

A.5.3 Transitions hiérarchiques des scènes

Les matchs de tennis possèdent une structure fortement hiérarchique avec des transitions entre les points, les jeux et les sets, structure qui peut être directement codés dans la topologie des modèles. Des états cachés internes sont ainsi introduits pour guider la structure hiérarchique du processus Markovien. D'un point de vue génératif, cette topologie est proche du HMM Hiérarchique [28], qui est généralement employé pour capturer des statistiques en multi-échelles. Dans notre cas cependant, la structure hiérarchique code la structure du match et sert comme source de contraintes.

La topologie proposée est illustrée sur la figure A.5. Au niveau supérieur, il existe l'état racine du modèle, représentant le match complet. Ce match est modélisé au niveau suivant comme succession des sets, de temps en temps interrompue par des pauses. Chaque set est encore analysé comme une succession de jeux et de pauses. Les règles du tennis proclament qu'il y a une pause après le premier jeu du set ou, plus tard, après deux jeux consécutifs de sorte que les joueurs changent leurs positions. Chaque jeu est alors analysé comme succession de points au niveau suivant, avec 4 points au minimum dans chaque jeu. Chaque point est analysé au dernier niveau des états internes comme contenant une scène ≪Premier service manqué et échange≫ ou ≪Échange≫, suivie facultativement d'une rediffusion.

Dans le cas du HMM, ces trois derniers états et les états correspondant à une pause sont les états internes (états cerclés dans la figure A.5) de la topologie hiérarchique. Ils sont analysés par les sous-modèle HMMs respectifs de la figure A.2. La topologie hiérarchique contient de ce fait 67 états internes et 141 états de production (i.e., les états feuilles déclenchent la production des observations). Dans le cas du SM, la hiérarchie a besoin d'un niveau de moins parce que les états internes mentionnés ci-dessus sont maintenant des états de production, représentant les scènes elles-mêmes. La topologie hiérarchique contient alors 20 états internes et 47 états de production.

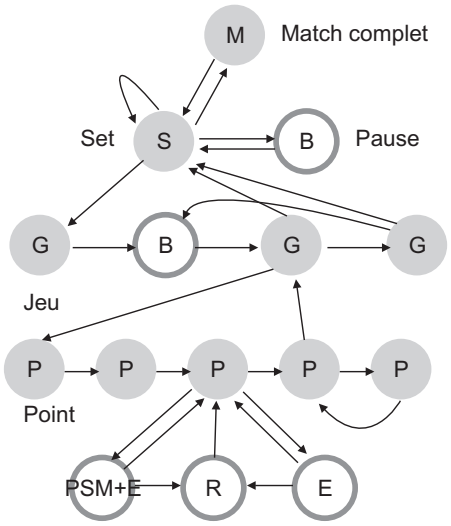


Figure A.5
Illustration de la topologie hiérarchique. Pour simplifier la représentation, seulement un noeud est déplié à chaque niveau. Les états cerclés dénotent des états de production pour le SM.

A.6 Intégration audiovisuelle

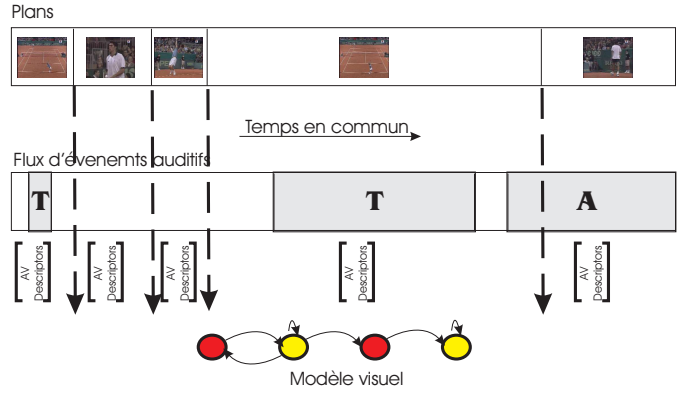
La bande sonore de la vidéo est une source importante d'information qui devrait être prise en compte dans notre modélisation. Par exemple, les états 1, 3, et 5 de la figure A.2 correspondent tous à des vues globales, qui sont visuellement très semblables aux états 9 et 12, qui correspondent eux à des vues globales sans jeu. La détection des bruits de balle dans la bande sonore peut considérablement aider dans la désambiguïsation entre les vues globales avec ou sans jeu.

A.6.1 Fusion avec les modèles de Markov cachés

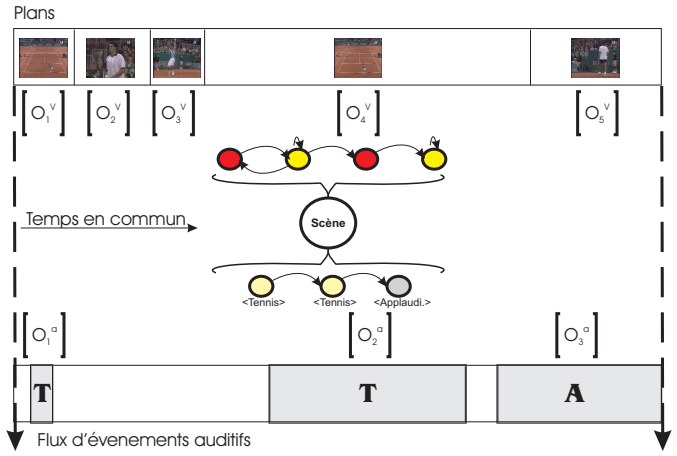
L'intégration audiovisuelle avec les HMMs n'a aucune autre option que l'approche standard de la fusion synchrone par concaténation à chaque état : les deux flux doivent être artificiellement synchronisés avant la fusion. Le procédé est illustré sur la figure A.6(a). Le flux d'événements sonores est premièrement segmenté selon les frontières des plans. Lors des étapes suivantes, trois descripteurs binaires sonores sont ajoutés au vecteur O_t^{v} des descripteurs visuels. Le vecteur des observations O_t^{v} de l'équation A.1 est ainsi redéfini comme :

$$O_t^{\mathrm{av}} = \left[o_t^c \; o_t^l \; o_t^d \; o_t^b \; o_t^a \; o_t^m \right] \tag{A.8}$$

où o_t^b, o_t^a, and o_t^m sont des descripteurs binaires qui capturent la présence ou l'absence des trois classes sonores principales (tennis, applaudissements, et musique) sur la bande sonore et à l'intérieur de la fenêtre temporelle définie par les frontières du plan respectif.



(a)



(b)

Figure A.6

Illustration de l'intégration audiovisuelle avec (a) HMMs et (b) SMs. Dans le premier modèle, un certain nombre de descripteurs binaires auditifs sont rassemblés à l'intérieur de chaque plan et concaténés au vecteur de caractéristiques visuelles. Avec les SMs, la bande sonore est échantillonnée et modélisée indépendamment et puis est fusionnée avec le contenu visuel au niveau de scène. Le 'T' représente le bruit de tennis (coups de balle) et le 'A' les applaudissements.

Les observations sont considérées ici encore comme indépendantes. Afin d'inclure les caractéristiques auditives pendant le décodage, l'équation A.3 est modifiée de sorte que $r \in \{c, l, d, b, a, m\}$.

A.6.2 Fusion avec les modèles segmentaux

L'utilisation de caractéristiques au niveau de segment, plutôt qu'à celles des plans comme cela était fait avec les HMMs, peut être bénéfique pour l'intégration audiovisuelle. L'idée principale est de séparer le contenu audiovisuel d'une scène en un segment ≪visuel≫ et un segment ≪sonore≫. Le segment sonore peut alors être échantillonné à sa fréquence native et aussi peut être modélisé par une distribution d'observations au niveau de segment différente. La contrainte de synchronisation entre les deux flux est alors repoussée aux frontières de la scène. Ceci est une hypothèse valide parce que toutes les caractéristiques visuelles et sonores relatives se trouvent à l'intérieur de la scène. Formellement, le contenu audiovisuel $O^{\mathrm{av}}_{1:l}$ d'une partie de la vidéo correspondant à l plans successifs est factorisé comme :

$$P(O^{\mathrm{av}}_{1:l}|l, S_i) = P(O^{\mathrm{v}}_{1:l}|l, S_i)P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i) \tag{A.9}$$

où l_{a} est la longueur du segment sonore, i.e., le nombre d'échantillons qu'il contient. Les SMs permettent à l_{a} d'être différent de l. Cette factorisation et la fusion des deux flux asynchrones est illustrée sur la figure A.6(b). Dans le restant de cette section, différentes possibilités de modélisation pour $P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i)$ sont explorées.

Caractéristiques sonores au niveau de la scène

Ayant une série de caractéristiques sonore rassemblées au niveau de la scène, la manière de faire la plus simple est juste de capturer la présence de chaque événement sonore à l'intérieur d'une scène. C'est un prolongement direct des descripteurs sonores basés sur les plans utilisés dans les HMMs. Cependant, la différence principale est qu'on permet maintenant aux caractéristiques sonores d'être *asynchrones* avec les visuelles. Les scores de segments sont simplement définis comme :

$$P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i) = \prod_{k=1}^{l_{\mathrm{a}}} P(O^{\mathrm{a}}_k|S_i) \tag{A.10}$$

où $P(O^{\mathrm{a}}_k|S_i)$ est la probabilité de la présence dans la scène des trois événements audio. La longueur l_{a} du segment sonore est définie par le nombre d'événements auditifs que la scène contient.

Succession d'événements sonores

La simple détection de la présence des événements dans les scènes est peut-être insuffisante. En effet, le contenu audio de la scène ≪Premier service manqué et échange≫

est normalement une succession {tennis, tennis, applaudissement} alors celui de la scène ≪Échange≫ est {tennis, applaudissement}. La présence des événements sonores résultera à un modèle identique pour les deux scènes. Pour remédier à ce problème, des probabilités pour capturer la succession des événements audio peuvent être employées :

$$P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i) = \prod_{k=2}^{l_{\mathrm{a}}} P(O^{\mathrm{a}}_k|O^{\mathrm{a}}_{k-1}, S_i) \tag{A.11}$$

Sans compter les trois événements audio de base, deux événements plus spéciaux sont introduits ici : le ≪début≫ et ≪fin≫ du segment. Le contenu sonore de la figure A.6(b), par exemple, sera {début, tennis, tennis, applaudissement, fin}.

Caractéristiques sonores cepstrales

Jusqu'ici, le flux sonore est intégré sous forme d'événements audio qui sont pré-détectés et pré-classifiés. Dans une approche tout à fait différente, les segments sonores sont modélisés directement sur les coefficients cepstraux. La raison de faire ceci est double : premièrement, la pré-segmentation probablement incorrecte du flux sonore dans des segments homogènes est évitée. Il est difficile de détecter les frontières de ces segments qui sont généralement plus vagues que les coupures du flux visuel. Deuxièmement, le contenu de ces segments peut contenir plus d'une classe sonore comme des bruits de balle superposés à de la parole. Ceci peut transformer la pré-classification en classes sonores incorrectes, comme par ailleurs l'extraction des descripteurs sonores à partir de ces segments.

Afin de modéliser le contenu sonore à partir des caractéristiques cepstrales génériques, des HMMs avec des densités continues sont utilisés. Leurs scores probabilistes sont calculés par la passe avant, comme pour les HMMs qui modélisent les segments visuels (équation A.4). Le flux audio est échantillonné maintenant à la fréquence de 100 trames par seconde. La longueur l_{a} du segment sonore est maintenant égale au nombre de trames audio de la scène entière.

Intégration audiovisuelle synchrone

Enfin, rien n'interdit l'utilisation d'un schéma de fusion par concaténation, exactement comme avec les HMMs standards. Le vecteur de caractéristiques visuelles du HMM est augmenté avec des descripteurs sonores (synchronisés), comme montré sur la figure A.6(a). Le modèle sonore $P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i)$ est alors rejeté.

A.7 Integration du score affiché

Dans cette section, nous discutons de l'intégration de l'information donnée par le score affiché avec les HMMs et les SMs.

On peut penser à deux manières possibles d'intégrer l'information du score affiché : intégration a posteriori et intégration au niveau des caractéristiques. Avec l'intégration a posteriori, un décodage N-best de Viterbi est premièrement exécuté et puis la solution la plus conforme aux scores affichés est choisie. Un nombre très grand de candidats, cependant, devrait être stocké pendant le décodage N-best afin de trouver une solution qui est réellement conforme aux scores affichés. Ceci aurait pour conséquence des contraintes de stockage et de durées de décodage prohibitives.

L'intégration au niveau des caractéristiques traite le score affiché comme un descripteur supplémentaire à ajouter aux vecteurs de caractéristiques qui sont associés aux plans. Comme alternative à l'utilisation de la caractéristique du score au niveau des plans, les SMs offrent également la possibilité d'utiliser cette caractéristique au niveau des scènes. Ceci tient compte d'un certain degré d'asynchronisme entre l'événement relatif de jeu et l'affichage du score correspondant. Généralement, l'affichage du score peut être une caractéristique utile et peut améliorer ainsi la performance du système. Le contenu sémantique du score (i.e., le nombre d'événements de jeu existants dans la vidéo), au contraire, ne peut pas être explicitement exploité.

Un troisième type d'intégration, la recherche de Viterbi guidée par le score, est présenté dans cette étude. Elle se situe entre les deux approches ci-dessus puisqu'elle exploite l'affichage précis du score et aussi son contenu sémantique pour piloter le décodage de Viterbi et pour garantir que la solution obtenue est conforme à l'évolution réelle du match. L'intégration au niveau des caractéristiques et la recherche de Viterbi guidée par le score sont le sujet du reste de cette section.

A.7.1 Les scores affichés comme caractéristique

Dans le cadre des HMMs, le vecteur des descripteurs audiovisuels basés sur les plans de l'équation A.8 est redéfini comme :

$$O_t^{\text{avs}} = \left[o_t^c \; o_t^l \; o_t^d \; o_t^b \; o_t^a \; o_t^m \; o_t^s \right] \tag{A.12}$$

où o_t^s est binaire et fixé à 1 pour les plans où un affichage du score est apparu à l'écran et à 0 autrement. Les caractéristiques sont de nouveau considérées comme indépendantes. L'hypothèse fondamentale sous-jacente à cette manière d'intégration est qu'après un échange s'est produit, le producteur de la diffusion le reconnaît en montrant une étiquette et cela à l'intérieur même du plan. La caractéristique supplémentaire introduite est ainsi employée pour marquer les plans d'échange, i.e., les états 3 et 5 de la figure A.2.

Un décalage limité entre l'événement de jeu et sa reconnaissance par le producteur peut être supporté dans le cadre des SMs. La présence d'un score affiché peut être capturé au niveau de scène, au lieu du niveau de plan. Le score du segment de l'équation A.9 est alors défini comme :

$$P(O_{1:l}|l, S_i) = P(O^{\mathrm{v}}_{1:l}|l, S_i)P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i)P^l(o^{\mathrm{s}}|l, S_i) \tag{A.13}$$

où $P(o^{\mathrm{s}}|l, S_i)$ dénote la probabilité de la présence d'un score affiché à l'intérieur de la scène S_i. Ce terme est élevé à la puissance l afin de fournir des scores qui montent exponentiellement avec la longueur de segment l, comme les deux autres termes de l'équation A.13. L'hypothèse fondamentale est maintenant que quand un échange s'est produit, le producteur en rend compte en affichant le score quelques plans après, et dans la même scène. La caractéristique de score est alors employée pour marquer les scènes ≪Premier service manqué et échange≫ et ≪Échange≫.

Le décalage entre l'événement de jeu et l'affichage du score peut être prolongé cependant au delà des frontières de scène. Par exemple, l'affichage peut apparaître pendant une scène de rediffusion qui se situe de temps en temps après une scène d'échange. En outre, il peut exister des événements de jeu qui ne sont pas pris en compte par le producteur. Nous avons noté, dans nos séquences vidéos, que 9% à 32% des échanges, selon les choix du producteur, n'ont pas été pris en compte par un affichage. Dans une des vidéos en particulier, 4 échanges consécutifs ont été laissés sans affichage. Une utilisation efficace de l'affichage du score ne devrait faire aucune supposition sur le style du producteur et devrait tolérer un décalage important et aussi des événements non pris en compte. Ceci peut être réalisé avec la recherche de Viterbi guidée par le score.

A.7.2 La recherche de Viterbi guidée par le score

Avant de procéder à la description de l'algorithme lui même, on rappelle que l'affichage du score apparait *après* l'événement de jeu correspondant et également *avant* le prochain événement de jeu. Un schéma typique est montré sur la figure A.7 où nous voyons trois étiquettes et leurs échanges correspondants. Il est clair que l'événement de jeu reconnu par l'étiquette dans le plan t_2 se situe dans l'intervalle $t \in [t_1, t_2]$. La scène complète qui contient cet échange, quant à elle, finira quelque part dans l'intervalle $t \in [t_1, t_3]$.

L'idée principale est d'exécuter une passe vers l'avant de Viterbi entre t_1 et t_3 avec un décodage N-best. Tous les chemins de cet intervalle qui ne sont pas conformes aux indications du score sont alors pénalisés. Dans le schéma de la figure A.7, un événement de point de jeu a dû être gagné entre les étiquettes ≪15-0≫ et ≪15-15≫. Les chemins ainsi de l'intervalle $t \in [t_1, t_3]$ qui ne contiennent aucun événement de point gagné ou qui en contiennent plus d'un seront pénalisés. L'action de la passe avant locale de Viterbi et la pénalisation des chemins locaux non conformes est désignée sous le nom du *recherche*

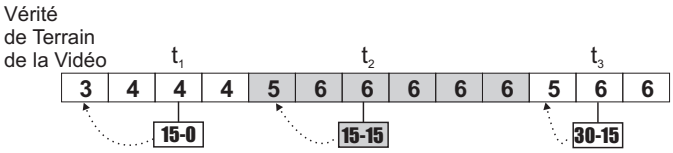


Figure A.7
Événements de jeu et apparition des étiquettes du score. La vérité de terrain de la vidéo se réfère aux conventions de la figure A.2

locale dans ce texte. Après la recherche locale pour l'étiquette «15-15», l'algorithme procède de la même manière pour l'étiquette «30-15». La nouvelle recherche locale partage cependant avec la précédente l'ensemble des chemins possibles dans l'espace $[t_2, t_3]$. L'algorithme procède ainsi en une cascade des recherches locales où les chemins restants à une étape sont encore développés jusqu'à la fin de la vidéo. La passe arrière du décodage de Viterbi est exécutée à la fin et la solution obtenue ainsi est conforme à tous les scores affichés. L'algorithme garantit l'optimalité parce que l'ensemble des chemins établis par les recherches locales ne contient pas les N meilleurs chemins, mais le meilleur chemin pour chaque nombre possible de points marqués entre deux étiquettes. Le nombre maximum de points marqués entre deux affichages a été fixé à 5, puisque le nombre maximum d'échanges consécutifs laissés sans affichage était de 4 dans le corpus utilisé dans cette étude.

Définition de l'algorithme

L'algorithme s'applique tant aux HMMs qu'aux SMs. Aucun paramètre supplémentaire n'est nécessaire pour les modèles, seul le décodage change. L'algorithme est décrit ici dans le cas des HMMs dont le décodage est plus simple. L'extension pour les SMs est facile et est ici omise.

Étant donnée une séquence vidéo $O_{1:T}$ de T plans, M étiquettes de score $l_1, l_2 \dots l_M$, apparaissant dans les plans $t(l_1), t(l_2) \dots t(l_M)$, sont détectées et identifiées. Le nombre de points marqués entre une étiquette et celle qui la précède (par exemple quand l'étiquette est '30-30' et la précédente est '15-15', deux points ont alors été marqués) sont donnés comme $s(l_1), s(l_2) \dots s(l_M)$. Pour chaque instant $1 \dots T$ et pour chaque état caché $1 \dots N$, deux files $Q_{t,i}$ et $Q'_{t,i}$ sont définies. La taille de chacune d'elles est $S = 5$, qui est le nombre maximum des points permis entre deux étiquettes. La première file contient les résultats de la recherche locale courante, alors que la seconde stocke les résultats de la recherche locale précédente et agit comme un pont entre les recherches locales successives. Chaque file stocke d'abord la vraisemblance $\delta_{t,s}(i)$ du meilleur chemin aboutissant à l'état caché i au temps t et avec un nombre de points s. Cette dernière quantité est simplement le nombre d'instances des états cachés 3 et 5 dans le chemin. La file Q définit

$\delta_{t,s,Q}(i)$ et la file Q' définit $\delta_{t,s,Q'}(i)$. Chaque file doit également stocker l'information convenable pour la passe arrière de Viterbi qui, dans notre cas, est l'étiquette de l'état caché précédent dans le chemin et également sa position correspondante dans la file.

L'algorithme procède en 3 étapes :

1) Initialisation. Pour chaque $t \in [1, T]$, $i \in [1, N]$, $s \in [1, S]$, $q \in Q$: $\delta_{t,s,q}(i) = -\infty$

2) Enchainement de recherches locales. Pour chaque étiquette l_k, $k \in [1, M]$, trois étapes sont exécutées (voir figure A.8 pour une illustration) :

- une passe avant de Viterbi est exécuté dans l'espace $t \in [t(l_{k-1}), t(l_{k+1})]$:

$$\delta_{t,s,Q}(i) = \max_{j,q,\sigma | s=\sigma(q)+n_{ji}} \delta_{t-1,\sigma,q}(j) a_{ji} b_i(O_t) \tag{A.14}$$

  où $q \in Q \cup Q'$ quand $t \in [t(l_{k-1}), t(l_k)]$ et $q \in Q$ quand $t \in [t(l_k), t(l_{k+1})]$, où $j \in [1, N]$ et $\sigma \in [1, S]$. Les résultats de la recherche locale sont stockés dans la file Q. Puisque chaque élément dans la file contient le meilleur chemin qui aboutit à un score donné s, les valeurs de j, q, et σ sont contraintes de sorte que le chemin obtenu corresponde à s points marqués. Si $q \in Q'$, puis $\sigma(q) = 0$ car des points développés précédemment ne sont pas pris en compte par l'étiquette courante de points. Si $q \in Q$, $\sigma(q) = \sigma$, comme prévu. Le terme n_{ji} est un indicateur binaire dénotant la transition qui correspond à un échange. Il est 1 quand i est 3 ou 5 et zéro autrement. Enfin, les termes a_{ji} et $b_i(O_t)$ dénotent des probabilités de transition et d'observation, comme dans le décodage standard de Viterbi.

- Tous les chemins qui aboutissent un score différent de $s(l_k)$ sont pénalisés :

$$\delta_{t,s,Q}(i) = \delta_{t,s,Q}(i) + P(s, s(l_k)) \tag{A.15}$$

  où $t \in [t(l_{k-1}), t(l_{k+1})]$, $s \in [1, S]$, $P(s, s(l_k)) = -\infty$ si $s \neq s(l_k)$ ou est égal à zéro, autrement.

- Le contenu de la file Q est transféré à Q' pour les besoins de la prochaine recherche locale. Le contenu de Q alors est remis à zéro comme dans l'étape d'initialisation.

Il est à noter que l'information nécessaire pour la passe arrière de Viterbi doit être convenablement gardée, particulièrement quand le contenu de la file Q est transféré.

3) La passe arrière. Ayant exécuté les recherches locales pour toutes les score affichés, la passe arrière est alors exécutée comme dans le décodage standard de Viterbi. La séquence des états cachés récupérée $Q^*_{1:T}$ est la plus probable selon les paramètres du HMM et aussi entièrement compatible avec les scores affichés.

Il est assez intéressant de noter que quand $P(s, s(l_k)) = 0$ pour chaque s (c'est-à-dire, quand aucun chemin n'est pénalisé), la recherche de Viterbi guidée par le score fournit

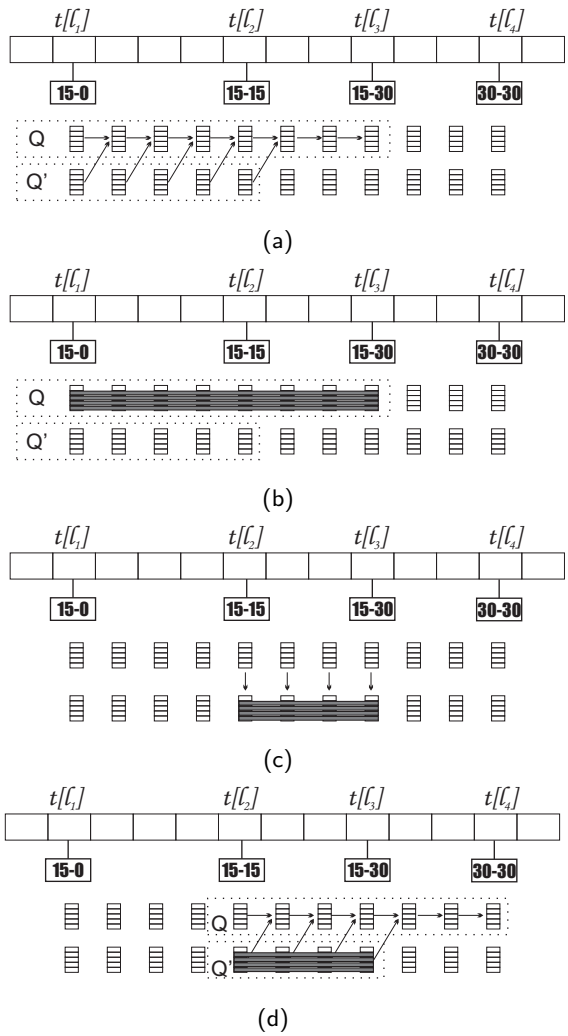


Figure A.8

Les étapes de l'exécution d'une recherche locale. En (a), une passe avant de Viterbi est exécutée en utilisant les files Q et le Q' pour rechercher des résultats et les stocker dans Q. Des chemins non conformes sont pénalisés en (b). En (c), les résultats de la recherche locale sont transférés à Q' pour les besoins de la prochaine recherche locale, exécutée en (d).

exactement la même solution que la version standard du décodage de Viterbi. Le coût de la recherche de Viterbi guidée par le score pour les HMMs est estimé comme suit. Le coût de calcul du décodage N-best avec des files de taille de $2S$ est $O(2SN^2T)$ (le coût de la passe arrière est négligeable). Puisque l'algorithme parcourt les intervalles entre chaque paire d'étiquettes deux fois, son coût global sera simplement $O(4SN^2T)$, qui est à comparer au $O(N^2T)$ du décodage standard de Viterbi.

Prise en compte de l'incertitude

Le nombre de points marqués entre deux étiquettes l_{k-1} et l_k a été précédemment considéré comme bien défini, étant donné que les listes ont été extraites manuellement. Cependant, les particularités du décompte des points au tennis (qui emploie des marques comme ≪break≫, ≪avantage≫, etc.) et encore le non-affichage de certains scores provoquent un degré d'incertitude sur le nombre de points marqué entre deux étiquettes. En raison de cela, nous avons employé une pénalité $P(s, l_{k-1}, l_k)$ variable, au lieu de la constante utilisée à l'équation A.15. Elle est estimée par des données de sorte que les transitions peu fréquentes obtiennent des pénalités plus élevées. Formellement, on la définit en fonction de la probabilité que la transition de l'étiquette l_1 à l_2 contienne s points marqués :

$$P(s, l_1, l_2) = A\Big(1 - \frac{N(s, l_1, l_2)}{\sum_{s=1}^{S} N(s, l_1, l_2)}\Big) \tag{A.16}$$

où $N(s, l_1, l_2)$ est le nombre de fois que s points sont marqués entre les étiquettes l_1 et l_2. Cette estimation est réalisée sur l'ensemble d'apprentissage. La valeur de la constante A, qui prend généralement de grandes valeurs négatives, a été fixée à -10 après expérimentation.

Utilisation de topologies hiérarchiques

La recherche de Viterbi guidée par le score possède d'autres avantages lorsqu'elle est utilisée avec une topologie hiérarchique. En effet, la topologie hiérarchique garantit une solution conforme aux règles de tennis, mais pas nécessairement au nombre de set ou de jeux par set réellement marqués. La recherche de Viterbi guidée par le score peut alors être facilement modifiée pour incorporer la topologie hiérarchique en ajoutant deux variables supplémentaires dans les files Q et Q' pour tenir en compte le nombre de sets ou jeux traversés. Selon les transitions de la topologie hiérarchique nous mettons à jour ces deux variables à chaque instant. Quand des informations sur l'évolution du match sont fournies par une étiquette de points (elles apparaissent habituellement à la fin de chaque jeu), les chemins non conformes peuvent être sans risque supprimés. De cette façon, nous pouvons obtenir une solution qui convient non seulement au nombre de points marqués, mais également à la structure réelle du jeu.

A.8 Estimation des paramètres

Nous avons manuellement étiqueté les séquences vidéo, une fois la segmentation en plans automatique réalisée, avec les étiquettes des états cachés de la figure A.2. Les séquences d'apprentissage contiennent 807 scènes au total et celles de test en contiennent 979. Afin d'obtenir des distributions discrètes des observations, les caractéristiques visuelles de similarité visuelle et de longueur de plan ont été discrétisées de manière homogène en dix intervalles, choisis après expérimentation. Il est alors simple d'estimer tous les paramètres du modèle HMM (i.e., probabilités de transition et d'observation) comme la fréquence relative de l'occurence des événements correspondant. Le même procédé convient aussi pour les probabilités de transition et de durée pour le SM. En particulier, le modèle de durée du SM, a été mesuré par la durée absolue de la scène en secondes, plutôt par le nombre des plans que la scène contient. Nous avons constaté que la durée absolue fournit une mesure plus fiable et plus robuste.

Les paramètres des HMMs visuels et audiovisuels qui servent à calculer les vraisemblances d'observation du SM ont été estimées par l'algorithme de Baum-Welch. Les probabilités de transition ont été initialisées à partir des transitions observées sur la figure A.2 pour les scènes ≪Premier service manqué et échange≫ et ≪Échange≫, alors que l'initialisation aléatoire a été utilisée pour les scènes restantes (les résultats finaux ont été peu affectés par ce schéma d'initialisation). Un schéma simple de back-off a été employé pour l'évaluation des probabilités audio de bigrammes afin d'éviter des probabilités nulles pour les séquences inconnues.

Pour estimer les paramètres des CDHMMs (composants des GMMs et probabilités de transition), l'outil HTK[2] a été utilisé. Parmi les diverses topologies examinées, les meilleurs résultats ont été obtenus avec des HMMs de gauche à droite sans un saut de plus de deux états, i.e., avec les probabilités $a_{ij} = 0$ de transition quand il est $j > i+1$. Le nombre des états cachés pour les quatre CDHMMs a été fixé à 20 et chacun d'eux définit 32 composantes gaussiennes, initialisées de façon uniforme. Les HMMs ont été entraînés avec l'algorithme de Baum-Welch jusqu'à convergence. Les trames audio se composent de 12 coefficients cepstraux et de l'énergie, plus les dérivées du premier ordre. Les CDHMMs produisent en fait des vraisemblances et non des probabilités, dont l'intervalle des valeurs possibles est beaucoup plus grand que celui obtenus avec des HMMs discrets. Pour moduler leur impact pendant le décodage, les vraisemblances ont été transformées linéairement dans un intervalle approprié des valeurs.

Concernant les topologies hiérarchiques, l'évaluation des probabilités de transition entre les états internes mènerait à un sur-apprentissage étant donné la taille limitée de notre corpus d'apprentissage. Par conséquent, ces probabilités ont été arbitrairement

[2]HTK toolkit : http://htk.eng.cam.ac.uk

fixées à 1 dans les expériences rapportées ici.

A.9 Résultats expérimentaux

Tout d'abord, on rappelle que la moitié des vidéos a été réservée strictement pour les tests. Puisque la vérité terrain des vidéos a été établie à partir de la segmentation automatique du flux visuel, les erreurs de coupures ou de fondus enchaînés ne sont pas prises en considération dans cette analyse.

Les mesures de performance comportent premièrement le pourcentage C des plans classifiés avec l'étiquette de scène correcte, moyenné sur l'ensemble des séquences de test. Elle est une mesure de la qualité de la classification. En outre, nous avons besoin d'une mesure concernant la qualité de la segmentation en scènes. Par exemple, si deux ou plusieurs scènes successives de la même étiquette sont correctement classifiées mais si leurs frontières ne sont pas correctes, la mesure C indique une bonne performance de manière non réelle. La qualité de segmentation peut être mesurée par les taux de rappel R et de précision P sur la détection des frontières des scènes, pris en moyenne sur les séquences de test. Afin de faciliter la comparaison entre deux approches, au lieu de comparer 3 statistiques, une seule mesure de performance est définie comme :

$$\hat{F} = \frac{3CPR}{C+P+R} \tag{A.17}$$

qui ressemble à la F-mesure standard, connue en statistique. Évidement, l'utilisation de $\hat{F}$ est purement suggestive et tous les taux C, P, R sont indiqués.

Nous présentons en premier les résultats sur la fusion audiovisuelle avec les modèles ergodiques, puisque les modèles hiérarchiques ne changent pas la manière dont l'audio est intégré. Ensuite, nous analysons l'intégration du score affiché, avec ou sans les modèles hiérarchiques.

A.9.1 Fusion audiovisuelle

Modèles de Markov cachés

Les résultats sur l'ensemble de test sont indiqués dans le tableau A.1 pour les HMMs et les SMs et pour des hypothèses et des caractéristiques variables. Nous voyons dans les premières lignes la performance des HMMs avec des caractéristiques visuelles (≪HMMs-V≫) et audiovisuelles (≪HMMs-VA≫). Tandis que la majeure partie de la contribution vient des caractéristiques visuelles, une amélioration claire de la performance est remarquée avec l'addition des descripteurs sonores aux vecteurs de caractéristiques visuelles, calculés au niveau des plans.

Table A.1
Comparaison des performances des HMMs et des SMs pour des ensembles de caractéristiques et des hypothèses de modélisation variés. Les pourcentages moyens C, P, et R sont calculés sur les trois vidéos de test et puis $\hat{F}$ est calculé.

	C	P	R	$\hat{F}$
Modèles de Markov cachés				
HMMs-V	76.30	81.99	73.44	59.48
HMMs-VA	80.23	84.69	79.70	66.42
Modèles segmentaux				
SMs-Vhmm	79.69	83.54	74.82	62.78
SMs-VhmmA1gram	80.07	83.86	75.27	63.39
SMs-VhmmA2gram	81.77	84.10	79.45	66.81
SMs-VhmmAcep	79.86	84.64	75.20	63.62
SMs-(AV)hmm	84.39	86.25	79.32	69.29
SMs-VhmmAhmm	81.52	87.83	77.55	67.47
SMs-(LD)hmmChmmAhmm	78.96	86.33	75.02	63.84
SMs-(AV)hmmA2gram	84.73	84.13	81.67	69.71

Modèles segmentaux

La performance des SMs est donnée sur les lignes suivants de la table A.1, en commençant avec l'utilisation des caractéristiques visuelles seulement (≪SMs-Vhmm≫). La comparaison avec les HMMs visuels indique quelques premiers résultats intéressants : le SM donne une meilleure performance par rapport au HMM (62.78% contre 59.48%, pour les scores $\hat{F}$), bien qu'ils fonctionnent dans un espace augmenté de recherche des chemins et des segmentations possibles. Dans le scénario unimodal des caractéristiques visuelles, la différence entre SMs et HMMs s'explique principalement par l'inclusion dans le SM du modèle de durée au niveau de la scène. Ce modèle de durée s'avère ainsi être utile au SM, alors qu'il est impossible de l'ajouter au HMM.

Sur les trois lignes suivantes, la performance des SMs audiovisuels avec des caractéristiques auditives asynchrones est indiquée. Les scores des événements sonores au niveau de scène (≪SMs-VhmmA1gram≫) peuvent améliorer la performance, mais à un niveau peu satisfaisant. La raison en est simple : ce modèle capture juste la présence des bruits de tennis dans la scène, ce qui finalement ne peut pas aider dans la discrimination des scènes ≪Premier Service Manqué et Échange≫ et ≪Échange≫. La différence entre les deux scènes est liée à la bande sonore : la première scène contient une succession de deux bruits de tennis (le service manqué et puis l'échange) contrairement à la seconde. Ceci peut être plus efficacement capturé en utilisant les modèles bigrammes (≪SMs-

VhmmA2gram≫). En effet, un gain clair de performance est noté (+4.03% comparés à +0.61% obtenu par le ≪SMs-VhmmA1gram≫, pour les scores $\hat{F}$) avec l'intégration de l'audio selon ce modèle.

La performance obtenue avec l'intégration de l'audio sous forme de caractéristiques cepstrales est donnée à la ligne suivante (≪SMs-VhmmAcep≫), où un gain de performance peu important est noté. Il est clair que les HMMs à densités continues apportent une mauvaise performance, à cause des trop nombreux degrés de liberté qu'a maintenant le problème. En effet, ils sont chargés de modéliser la bande sonore d'une scène entière sans a priori et ils reçoivent alors en entrée un grand nombre de vecteurs de caractéristiques de grande dimension. Dans les approches ≪SMs-VhmmA1gram≫ et ≪SMs-VhmmA2gram≫ par contre, beaucoup de connaissance a priori est intégrée, puisque les événements sonores intéressants de la bande sonore sont définis a priori. Néanmoins, cette approche démontre que les SMs fournissent beaucoup de liberté pour la fusion multimodale. Dans ce scénario spécifique, l'information visuelle échantillonnée en plans est fusionnée avec des catactéristiques auditives échantillonnées à 100 fps, ne faisant aucune hypothèse forte de synchronisation.

La performance des HMMs audiovisuels synchronisés est indiquée à la ligne ≪SMs-(AV)hmm≫ de la table A.1. Les résultats obtenus avec ce modèle sont meilleurs par rapport aux modèles de fusion asynchrone mentionnés ci-dessus. Le gain de performance obtenu après avoir pris en compte de l'information sonore est, sans surprise, de +6.51%, niveau proche de celle obtenue avec les HMMs : +6.94%.

Integration audiovisuelle précoce ou tardive

En comparant les modèles ≪SMs-VhmmA2gram≫ et ≪SMs-(AV)hmm≫, il s'avère que la fusion asynchrone d'information sonore provoque une légère dégradation de performance. Ceci semble étrange puisque les deux modèles utilisent la même information (visuelle et sonore), seule la méthode d'intégration change. Dans un premier temps, on peut supposer que le modèle bigramme (dans ≪SMs-VhmmA2gram≫) est un modèle pauvre et, au résultat final, les descripteurs sonores au niveau des plans pourraient être plus instructifs. Pour vérifier ceci, le calcul des scores pour le HMM audiovisuel du ≪SMs-(AV)hmm≫ a été scindé en deux HMMs indépendants avec le même nombre d'états cachés. Le premier HMM reçoit les caractéristiques visuelles et le deuxième les descripteurs sonores au niveau des plans[3]. Cette approche utilise exactement le même ensemble de caractéristiques au niveau des plans que le ≪SMs-(AV)hmm≫ mais elle modélise le segment sonore et visuel indépendamment. Elle remplace ainsi le modèle bigramme ≪SMs-VhmmA2gram≫ par les descripteurs au niveau des plans sonores utilisés dans le ≪SMs-(AV)hmm≫. Sa performance est indiquée comme ≪SMs-VhmmAhmm≫

[3]Ceci donne un modèle semblable aux HMMs multibandes, proposés en reconnaissance de la parole.

dans la table A.1. Une performance très proche de celle du ≪SMs-VhmmA2gram≫ est obtenue, ce qui implique que les modèles bigrammes sont à peu près aussi informatifs que les descripteurs sonores au niveau des plans.

Les chiffres ci-dessus montrent que, en modélisant les segments auditifs et visuels indépendamment, la performance chute. Ceci est bien plus clair encore avec le ≪SMs-(LD)hmmChmmAhmm≫, où l'ensemble des caractéristiques du segment visuel est scindé en deux, similarité visuelle d'un coté et longueur et descripteur des plans spéciaux, de l'autre. Ce dernier modèle utilise trois calculs des scores HMM indépendants, un pour le segment auditif et deux pour le segment visuel. La performance chute maintenant beaucoup plus. On peut en conclure que des corrélations importantes entre les différentes modalités ne sont pas prises en compte quand on commence par calculer les scores des segments des HMMs avant de les fusionner, alors que les ≪SMs-(AV)hmm≫ procèdent d'abord à l'intégration des modalités avant de calculer un score unique pour chaque segment.

Quand des caractéristiques sont associées directement à la sémantique et l'intégration est effectuée ensuite, nous parlons de fusion tardive, alors que quand l'intégration précède la décision sur la sémantique, nous parlons de fusion précoce. L'analyse ci-dessus suggère ainsi que, pour le problème actuel, une intégration précoce est préférable. Cependant, d'une manière générale pour l'indexation audiovisuelle, la question de savoir si une fusion tardive ou précoce est la meilleure option reste entière. Deux études [42, 96], par exemple, ont conclu favorablement pour la fusion tardive avec cependant peu de différence entre les performances. Dans la première, Huang *et al.* comparent la fusion concaténative avec des HMMs et les produits des vraisemblances des HMMs, dans un schéma semblable à celui de cette étude (en ce qui concerne l'intégration). Dans la seconde, Snoek *et al.* comparent l'intégration tardive et précoce avec des classifieurs SVMs.

Une solution possible pour surmonter le problème de la fusion tardive est d'intégrer l'audio par fusions tardive et précoce, simultanément. Cette approche utilise pour les segments visuels et sonores leurs fréquences d'échantillonnage et topologies natives et, en même temps, fournissent des astuces en ce qui se passe dans les autres modalités avant l'intégration tardive. Ce double schéma de fusion est donné comme ≪SMs-(AV)hmm-A2gram≫ dans la table A.1, où une légère amélioration est notée par rapport au ≪SMs-(AV)hmm≫.

Matrices de confusion

Les matrices de confusion concernant la classification des plans sont données par la table A.2 pour les ≪SMs-Vhmm≫ et ≪SMs-(AV)hmm≫. Il est clair que la majeure partie de la confusion vient entre les deux premières scènes (≪Premier service manqué et échange≫ at ≪Échange≫, marqués en tant que 1 et 2 dans la table), tandis que les rediffusions et

Table A.2
Les matrices de confusion pour la classification des plans avec les étiquettes de scène. Chaque ligne 1-4 se réfère à une scène et chaque colonne donne le pourcentage des plans qui sont classifiés avec l'étiquette respective 1-4 de classe.

SMs-Vhmm				
	1	2	3	4
1	77.1	20.3	2.4	0.2
2	22.6	75.5	1.3	0.6
3	10.9	4.4	84.7	0.0
4	2.9	7.8	6.9	82.4

SMs-(AV)hmm				
	1	2	3	4
1	86.1	11.8	1.7	0.4
2	18.8	79.2	1.3	0.7
3	7.5	4.5	87.4	0.5
4	2.8	7.7	5.3	84.2

les temps morts sont bien détectés (marqués en tant que 3 et 4 dans la table). L'addition des caractéristiques auditives enlève une partie de la confusion, particulièrement pour les deux premières scènes.

A.9.2 Intégration du score affiché

Systèmes de base

Les résultats sont indiqués dans la table A.3. Sur la première ligne de la table, les performances du ≪HMMs-VA≫ et du ≪SMs-VhmmA2gram≫ sont recopiées de la table A.1. Elles servent de référence pour la comparaison avec les systèmes de cette section. Des résultats pour les topologies hiérarchiques sont indiqués dans la deuxième ligne de la table. Une légère dégradation de performance est notée pour les HMMs et SMs en passant des topologies ergodiques aux hiérarchiques. Une explication possible pour ceci est que les probabilités des transitions hiérarchiques permises ont été manuellement fixées à de valeurs arbitraires. Les probabilités intra-scènes sont ainsi perdues, alors que les modèles ergodiques les apprennent à partir des données d'apprentissage.

Sur la troisième ligne de la table A.3, nous voyons la performance des systèmes ergodiques quand les étiquettes de score sont employées en tant que caractéristiques additionnelles. Une légère amélioration de performance est remarquée, principalement pour les HMMs. Les désynchronisations prolongées qui existent entre l'événement de jeu et l'étiquette de score et l'absence de l'affichage pour quelques événements sont la cause d'une distribution presqu'aléatoire de la caractéristique de score entre les plans (pour les HMMs) ou les scènes (pour les SMs). Ceci a pour conséquence des distributions presque uniformes de la caractéristique de score et sa contribution pendant le décodage est ainsi neutralisée.

Table A.3

Les résultats expérimentaux pour les topologies hiérarchiques et l'intégration des scores affichés. Pour chaque approche, la performance du HMM et du SM est indiquée. Les trois nombres à la ligne supérieure correspondent aux mesures C, P, R et celui à la ligne inférieure à $\hat{F}$.

	HMM	SM
Systèmes de base		
Ergodique	80.23 84.69 79.70 66.42	81.77 84.10 79.45 66.81
Hiérarchique	79.28 84.96 77.96 65.04	81.25 85.63 77.24 66.04
Score comme caractéristique	80.81 85.75 80.37 67.66	81.96 84.19 79.70 67.11
Recherche de Viterbi guidée par le score		
Ergodique	82.17 83.40 82.39 68.31	85.97 84.90 83.43 71.84
Hiérarchique	82.67 84.30 80.54 68.03	85.80 85.15 82.89 71.57
Niveau d'incertitude		
Pas d'Incertitude	82.03 83.63 82.65 68.50	85.99 85.19 83.30 71.94
90% de bruit simulé	81.59 83.88 82.22 68.16	85.60 85.53 82.75 71.59
50% de bruit simulé	81.15 84.37 80.67 67.31	84.15 87.02 80.16 70.07

Recherche de Viterbi guidée par le score

Les résultats pour l'algorithme proposé de la recherche de Viterbi guidée par le score sont indiqués sur les deux lignes suivants de la table A.3. Clairement, dans les HMMs et les SMs et avec des topologies ergodiques ou hiérarchiques, les contraintes apportées par les scores affichés ont aidé le système et ont amélioré la performance. La topologie hiérarchique une fois de plus n'arrive pas à surpasser la topologie ergodique. Mais la différence entre les deux est clairement réduite, grace aux contraintes additionnelles sur la structure du match imposées par les scores affichés. On peut également noter que le gain pour les SMs est beaucoup plus fort que celui pour les HMMs (+5.03% contre +1.89% dans les mesures $\hat{F}$). Ceci peut être expliqué par le fait que les instants même d'affichage indiquent approximativement les frontières des scènes, fournissant de l'information valable supplémentaire pour le décodage Viterbi pour les SMs. Quand aucune pénalité n'est employée, la recherche de Viterbi guidée par le score donne comme résultat $C = 81.71$, $P = 85.15$, $R = 79.16$, $\hat{F} = 67.16$ (tandis que pour les HMMs elle ne change pas, comme expliqué dans la description de l'algorithme).

Variation du niveau d'incertitude

Jusqu'ici, les pénalités de la recherche de Viterbi guidée par le score ont été calculées selon l'équation A.16 et sur des étiquettes extraites et identifiées manuellement. La seule source d'incertitude est due au schéma de décompte du tennis, combiné avec le non-affichage de certains scores. Sur les trois dernières lignes, nous avons indiqué la performance de la recherche de Viterbi guidée par le score avec les modèles ergodiques et avec un niveau varié d'incertitude pour le calcul des pénalités. Tout d'abord, nous avons indiqué, à titre de référence, les performances obtenues lorsqu'aucune incertitude n'existe. Dans ce scénario, le nombre de points marqués entre deux affichages de score est manuellement fourni par la vérité terrain. Les pénalités sont ainsi $-\infty$ ou 0. Le décodage contraint de Viterbi fourni donc un chemin qui contient exactement le même nombre d'échanges que la vérité de terrain et pour toutes les vidéos. Quand il y a incertitude, le nombre d'échanges récupéré est toujours proche de ce qui s'est vraiment produit sur le court. Plus précisément, les vidéos de test contiennent au total 731 échanges. La recherche de Viterbi guidée par le score (avec de l'incertitude) en a récupéré 727, alors que le Viterbi standard (≪SMs-VhmmA2gram≫) en découvrait 794, un bien moins bon résultat.

Sur les deux dernières lignes de la table A.3, nous voyons la dégradation de performance de la recherche de Viterbi guidée par le score (structure ergodique) quand un pourcentage indiqué des scores affichés est artificiellement mal-reconnu (substitué avec une autre marque au hasard) et après avoir ré-estimé les pénalités de l'équation A.16. Le bruit simulé a pour conséquence des pénalités plus uniformes et finalement les per-

formances se dégradent et s'approchent à celle du décodage de Viterbi standard. Le système démontre ainsi un excellent degré de robustesse aux erreurs de reconnaissance des scores affichés.

Notes sur les performances

Les performances avec le ≪SMs-VhmmA2gram≫ ergodique, décodé avec la recherche de Viterbi guidée par le score, est la meilleure obtenue sur l'ensemble de caractéristiques extraites automatiquement. Quand l'approche du ≪SMs-(AV)hmmA2gram≫ (qui a mené aux meilleures performances dans les expériences pour l'intégration audiovisuelle) est employée à sa place, la performance respective est $\hat{F} = 71.57$, ce qui est très proche. Nous concluons qu'une saturation de performance est rencontrée dans ces expériences, qui rend les deux methodes équivalents sur le plan de leurs performances.

Enfin, on peut trouver étrange que même lorsque la solution de Viterbi est entièrement conforme au nombre d'échanges effectuées et à l'évolution réelle de jeu, les mesures de performance continuent toujours à enregistrer une quantité considérable d'erreur. Ceci est expliqué par le fait que les scores affichés ne peuvent résoudre aucun des cas de confusion entre les scènes ≪Premier service manqué et échange≫ et ≪Échange≫, car toutes les deux contiennent un échange et sont alors équivalents en termes de score.

A.10 Conclusions

Dans cette étude, le cadre des SMs a été introduit pour l'indexation des vidéos dans le but d'effectuer une intégration audiovisuelle avec des contraintes synchronisation moins fortes. Les systèmes HMMs de base souffrent du fait que la contrainte de synchronisation s'applique à chaque état et donc pour toutes les observations. L'utilisation de caractéristiques au niveau des segments peut par contre prolonger les points de synchronisation entre les modalités aux frontières des segments. En modélisant chaque modalité à l'intérieur de son propre segment, des fréquences d'échantillonnage et des topologies natives peuvent être employés. Les SMs ont été appliqués sur une tâche de segmentation de retransmissions de tennis en scènes reconnaissables par les humains, chaque scène étant un segment. Sur des données visuelles seulement ou avec une fusion audiovisuelle synchrone au niveau des plans, les SMs ont démontré une amélioration de performance par rapport aux HMMs, avec un coût de calcul supplémentaire négligeable. Des possibilités de fusion asynchrone des modèles sonores avec les SMs ont été également examinées mais les performances n'ont pas été améliorées.

La deuxième contribution de cette étude est un algorithme spécialisé de décodage de Viterbi, la recherche de Viterbi guidée par le score. L'algorithme découvre le chemin le plus probable qui est conforme aux scores affichés, avec un calcul d'un coût peu

supérieur à celui du décodage de Viterbi standard. La fusion de l'information due au score affiché en tant que tel, par opposition à la fusion par caractéristique supplémentaire, a apporté une nette amélioration de performance des HMMs et SMs et avec des topologies ergodiques ou hiérarchiques. L'algorithme a également montré une tolérance naturelle aux caractéristiques bruitées et se rapproche avec élégance du Viterbi standard quand le niveau de bruit augmente significativement.

Les perspectives ouvertes par ces travaux comportent premièrement l'addition d'informations utiles dans la liste des caractéristiques, comme les résultats d'un procédé de suivi des joueurs. Les règles de tennis dictent que les joueurs doivent changer de position entre des échanges successifs. Le suivi des joueurs entre les échanges successifs peut ainsi fournir un indice utile. Nous planifions également de prolonger le cadre des SMs à d'autres genres de vidéos où l'analyse de structure est exigée. Les journaux télévisés en sont un exemple. On peut définir un segment comme une unité de nouvelles. Les sources d'informations par des modalités multiples, comme l'image, le son et le texte peuvent être asynchrones entre eux mais, par définition, ils sont synchrones à l'intérieur des frontières de l'unité de nouvelles. Enfin, la recherche de Viterbi guidée par le score pourrait être employée dans d'autres genres de vidéo où les événements de jeu se produisent plutôt fréquemment et sont comptés, comme dans le basket-ball, par exemple. L'algorithme peut garantir que la solution obtenue par le décodage de Viterbi en termes de points marqués est conforme à l'évolution réelle du jeu.

CHAPTER 1

Introduction

Human observers can watch, understand and interact with video documents effortlessly, making these tasks part of their every day life. They pay attention to important game events in a sport broadcast, gather useful information from news, decide to store interesting or amusing videos, and, more generally, they interpret the video content according to their needs. In the last few years, with the advances in the technology of communication, capturing and storage devices, video data collections have become extremely voluminous. There exists thus an increasing need for automated processing of video that will replace or aid humans in the time consuming task of searching and browsing through these collections. Modern pattern recognition provides us with a variety of tools for extracting low level image, audio and text features from raw video data. This study is about jointly exploiting all these possible information sources with temporal stochastic models, the Segment Models. Taking as case study tennis broadcasts, the table of contents of the video is automatically constructed, providing thus human-meaningful interpretation of the video.

1.1 The Video Indexing Problem

Video indexing is formally defined as the problem of "[automatically] attaching content-based labels to video" [16]. The role of these labels or indexes is to capture the semantic meaning of the video and to provide descriptors that meet the needs of the documentalist or, more generally, of the end-user. Once the video is annotated with these labels, it can be stored and then easily and quickly accessed. The question of what to index

is strongly dependent not only on the target application, but also on the video genre. Indexing soccer games usually involves the detection of goals and other highlights by filtering out less interesting game action. For news broadcasts, on the contrary, we often need to segment it into topics and then to index all of them with appropriate keywords.

Given a target application, the first task in video indexing is usually to automatically detect and extract salient features from raw video data. Fortunately, modern video and audio processing techniques provide a great deal of tools for automatic feature extraction. Considering, at first, video frames as still images, we can build visual indices based on color, texture, shape, etc. On top of these features, we can detect image similarities with one of the numerous techniques developed in the field of Image Indexing [93]. Furthermore, face detection or optical character recognition tools can extract some higher-level information from the still images. Moving from still images to video frames, we can perform motion analysis, detect hard cuts or other relative image sequence tasks. Regarding the audio track, there are automatic techniques for music, speech or other prominent sound classes detection. Audio information at a higher level can be extracted with word spotting or speech recognition techniques [61]. Finally, superimposed text or closed captions provide useful textual features. The particular choices of features is, once again, genre-dependent. For example, speech transcription may be important for news broadcast analysis but, at the same time, redundant for sports broadcasts.

In some simplified scenarios, one can attempt a direct mapping of the above low-level features to human-meaningful labels. But generally speaking, there is a distance in how humans perceive video and what we can actually extract from it with computational methods. For example, detected faces in news broadcast can have different semantic meaning as they can be the anchor person, a journalist or an interviewed person. Crowd excitation in soccer video may appear after a goal but also after a serious fault. For humans, these two events are not the same. This problem is widely referred to in the relative literature as the *semantic gap*, defined as "the lack of coincidence between the information that one can extract from the [audio]visual data and the interpretation that the same data have for a user in a given situation" [93]. This problem is largely unsolved in image/audio indexing and, thus, naturally inherited to video indexing.

Bridging the semantic gap requires a high-level reasoning that only humans can do and thus this problem is in practice unsolvable with the existing audio and visual pattern recognition methods. There are however ways to effectively narrow it in video indexing. The use of *video content* can be decisive at this point. The concept of content analysis is not new: it has been used in image indexing exactly for the same reason but in video indexing it can play a more prominent role. Indeed, as video is not just still images, we can use multiple concurrent or successive features in order to extract high-level semantics. In addition, and with the exception of surveillance video, audiovisual raw data are usually perplexed before transmission with production effects and rules, like

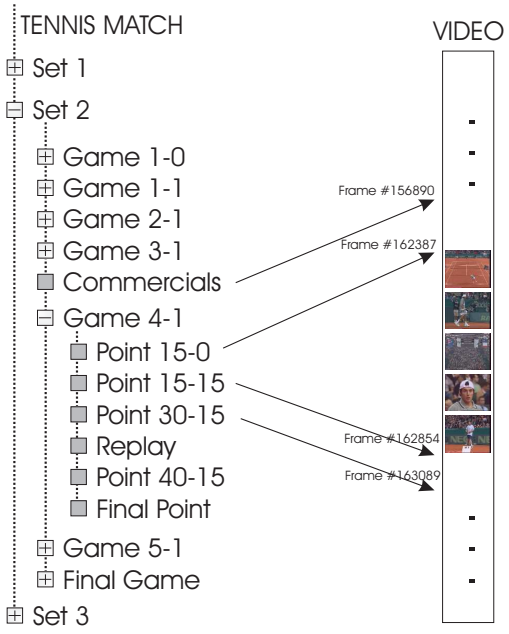


Figure 1.1
The table of contents of a tennis broadcast.

the superimposed text, that can always give useful hints. In a generalized view, video[1] can be conceived as the result of an authoring process that gives a certain content and layout [95]. Video analysis attempts to reverse this process by recovering the underlying content and layout.

1.1.1 Tennis Broadcasts Structure Analysis

This study considers the problem of video structure parsing, which is defined as the problem of segmenting the video into individual scenes that contain a unique semantic or narrative entity [91]. Structure parsing usually operates on top of the video shot or sound boundaries detection. For these two last problems, local signal inspection is performed in order to detect homogeneous segments at the signal level. Scene segmentation, on the contrary, requires the examination of the video content at a higher level in order to group multiple shots or sound segments under the same semantic label. Having detected and classified the scenes of a video, we can construct its table of contents. This process has an analogy in text document analysis, where we use its physical blocks, which are the words and the sentences, to detect thematic units and finally construct its table of contents.

Tennis broadcasts are examined as a case study. There is an underlying structure in

[1] With the exception of surveillance video.

this type of video as a result of the production rules and the game rules as well. The detection of this structure will allow the segmentation of raw video data into human-meaningful scenes that constitute its semantic building blocks. In tennis broadcasts we are interested in detecting and recognizing the scenes of game exchanges, replays, and commercials (for filtering them out). On top of these building blocks, we are also interested in recovering the hierarchy in games and sets that tennis videos naturally contain. It is then straightforward to construct the table of content of the video, as it is illustrated in Fig. 1.1. The user can unfold the hierarchy in order to reach the leaf nodes that point to the corresponding timestamps of the video. Instant access is thereby provided, instead of having to patiently fast-forward and search until the desired point is found out.

1.2 Motivation

As video documents are inherently multimodal, an efficient content representation scheme should take into consideration all the modalities of the video, namely the visual, auditory, and textual modalities [95]. There are three main reasons that call for multimodal representations:

- *Multimodal semantics.* Some semantics are based naturally on multiple media. Unimodal representations provide thus a partial view and are naturally incapable of carrying out the task. The detection of a goal in soccer video, for instance, requires joint processing from multiple information sources like soundtrack, motion fields, etc. Unimodal processing would lead to a lot of false alarms.

- *Richer semantics.* High-level labeling of raw audiovisual material can be achieved when using suitable modalities like text. Superimposed text in news video, for instance, can provide high-level keywords on the news category or on the depicted personality. This is the key idea of the Name-It project [90].

- *Robustness and efficiency.* Features automatically extracted from one modality are often erroneous. To enhance robustness, further support can be found from features from the other modalities. For example, it is hard to tell if a person is a male or a female subject using image-only data. The addition of the soundtrack can provide further evidence.

Regarding tennis video, multimodal processing can play an important role as features extracted from various modalities may be erroneous and also provide a partial picture of what actually happens in the court. For example, we can detect court views using color information but some of them may not contain game action at all. The presence of ball hits sounds in the soundtrack can further support the detection of the interesting

court views. The fusion of audio and visual features is a challenging task as sampling rates are different. In addition, the visual content follows the production rules (court views, close-ups, etc) while the audio content captures raw sounds from the court, interlaced with commentary speech. There is thus, firstly, a certain degree of asynchrony between auditory and visual features and, secondly, they follow different temporal models. Finally, textual resources like score indications and game statistics convey important information on the game evolution that has to be fully exploited. Difficulties rise from the fact that they may appear long after the relative game event or can be missing.

There are numerous approaches to multimodal fusion in the relative literature, reviewed in [106, 95]. Hidden Markov Models [81] (HMMs) is a powerful statistical approach that can model temporal patterns and are widely used as statistical parser of a video sequence [108], sharing notions from the field of speech recognition. A first solution for multimodal integration in video indexing is to process each modality separately and then to a-posteriori combine outputs of unimodal HMMs (e.g., [42]). Another approach, referred to in the literature as *early integration*, is to concatenate features from all the modalities into a super-vector of observations and to use a single HMM to model the content (e.g., [42, 48]). However, explicit state synchrony between the modalities is thus assumed and, in addition, different modalities are forced to follow the same HMM topology. The problem of state asynchrony has been addressed in audiovisual speech recognition by the use of product HMMs [79] or Asynchronous HMMs [8], but still the auditory and visual streams are generated by the same stochastic process (the speech production) and there exists thus only a limited asynchrony caused by natural noise. Layered HMMs [70] use the outputs of HMMs operating at low levels to feed HMMs of the next level in a cascade fashion. They provide a number of advantages like fusion at different frame rates and with independent models for each modality, but they require synchronization at a-priori fixed time intervals.

1.3 Approach

The framework of Segment Models (SMs) is proposed in this study in order to perform audiovisual integration with relaxed synchrony constraints. SMs have been introduced by Ostendorf *et al.* [72] in the speech recognition literature as a generalization of HMMs to account for a more accurate modeling of the speech production process. In SMs, each hidden state is associated to a sequence of observations, called a *segment*, instead of using a single feature vector as in HMMs. Therefore, each hidden state in SMs defines a duration model that accounts for the segment length and an emission probability distribution of a sequence. The extension from the frame-based features of HMMs to segmental ones can be beneficial for audiovisual integration by processing each modality in its own segment. The synchrony constraint between the modalities is thus relaxed

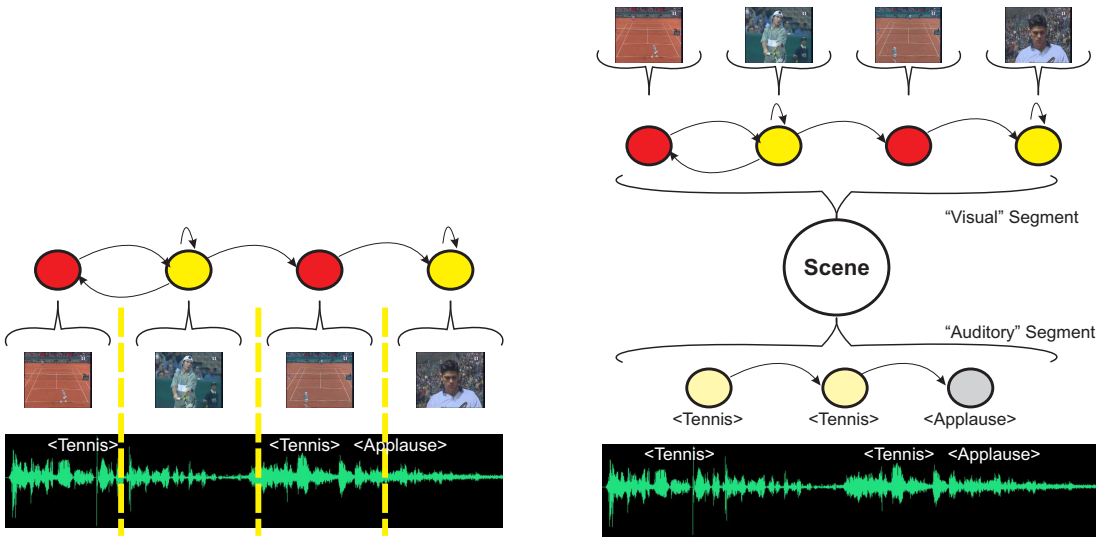


Figure 1.2
Audiovisual integration with (a) Hidden Markov Models and (b) Segment Models.

to the segment boundaries, while different sampling rates and models can be used. The Viterbi decoding for SMs involves the most likely classification of the hidden states, as in HMMs, and also the detection of the segment boundaries.

In tennis video, the content of a complete scene is represented by two distinctive segments, a visual and an auditory one. The difference between HMMs and SMs is illustrated in Fig. 1.2. In the HMM framework, a reference modality is chosen and modeled on top of its native segmentation. This role is given to the visual modality in Fig. 1.2(a) and it is modeled according to the production rules. Regarding the audiotrack, tennis sounds and applause are detected. These audio features have to be artificially aligned to the model of the reference modality. In SMs (Fig. 1.2(b)), on the contrary, individual models can be built for each modality using their native segmentation. The asynchrony is extended to the scene boundaries, which is the elementary semantic unit of the video.

Textual resources present further difficulties as their asynchrony may be extended beyond the scene boundaries. In addition, they convey important semantic information on the game evolution that the final Viterbi solution should be consistent with. An a-posteriori rescoring of the best paths with techniques like N-best decoding would require infinite storage capabilities. Instead, a novel search algorithm is proposed, called *Score-Oriented Viterbi Search*, that uses the score indications to pilot Viterbi decoding. The alignment of the labels to the corresponding game events is left as part of the optimization problem, while the consistency with the game evolution is satisfied on the fly, during decoding.

1.4 Methodology

The video track is segmented with automatic hard cut and dissolve transition detection techniques. On top of this segmentation, four scenes are identified: first missed serve and exchange, exchange, rediffusion and break. From each shot, three features are extracted. The first one is a color-based distance to a reference court view, which is unique for every video and is automatically selected with some statistical optimality criteria. The remaining two shot-based features are the length of the shot and a binary indicator of presence/absence of a dissolve. The processing of the soundtrack involves the detection of key sound classes, which are ball hits, applause and music. They are detected automatically using Gaussian Mixtures Models that are built on manually labeled data. Finally, the soundtrack is converted to a stream of audio events.

Following the production rules, the video content of each scene is modeled by an HMM. The baseline HMM system is formed by interconnecting these HMMs into a large ergodic HMM, modeling the game as a succession of shots. This HMM may have also a hierarchical structure that reflects the tennis match structure. Audiovisual integration is performed by concatenative fusion of binary descriptors of the presence/absence of the respective sound classes in the shot. Regarding SMs, the visual segment is modeled again with HMMs that serve as probabilistic scorers of a segment. The audio content is modeled with bigrams models on succession of audio events. As SMs offer a great deal of freedom in segmental modeling, two more possibilities are examined. Firstly, continuous density HMMs are employed to model the audio content of a scene directly on top of cepstral features and without any pre-segmentation and pre-classification to sound classes. Secondly, the use of a recurrent neural network architecture, called Long Short-Term Memory [39], is also examined as a segmental scorer, leading to a Segment Model-Recurrent Neural Network hybrid. Both ergodic and hierarchic structures are considered for SMs, too. The enhanced optimization problem for SMs is solved via a straightforward extension of the Viterbi decoding for HMMs with explicit state duration [81].

Three tennis videos were used for parameter estimation and three were reserved for testing purposes. Model parameters were estimated straightforwardly using the ground truth of the videos. The parameters of the HMM scorers were estimated with the Baum-Welch algorithm. Performance measurements consider the quality of the classification and the quality of the segmentation into scenes, as well.

For the needs of the Score-Oriented Viterbi Search, score and statistics labels were manually extracted and recognized. The algorithm, however, can take into account feature uncertainty due to erroneous label recognition. Its robustness to noise is demonstrated experimentally by introducing artificial noise to the features.

1.5 Outline

This study is organized as follows: chapter 2 provides a literature survey on multimodal integration. Emphasis is given to statistical approaches and to HMMs variants. Feature extraction is discussed in chapter 3. Approaches to tennis video processing (whether multimodal or not) are also given at the end of this chapter. Video parsing and audiovisual integration with both HMMs and SMs is detailed in chapter 4. Hierarchical topologies and the integration of the information conveyed by the score announcements is discussed in chapter 5. In chapter 6, the use of Long-Short Term Memory for video structure analysis is presented. Finally, chapter 7 concludes this study and gives possible directions for further work.

CHAPTER 2

Literature Survey

This chapter provides a brief survey on the integration of the three (visual, auditory, and textual) video modalities. Typically, audiovisual raw data are firstly pre-processed in order to extract low or mid-level features such as image histograms or motion vectors, which in turn are integrated into a multimodal framework for the extraction of the needed semantics. The focus of this chapter is on the integration stage. The interested reader is referred to [16] for technical information regarding video feature extraction, and to the image indexing [93] and audio indexing [61] literature as well.

Reviews on multimodal video indexing can be found in [106, 82, 1, 17, 54], while a balanced overall picture is given in [95]. In this last study, multimodal techniques are divided according to the way content segmentation of the three modalities is performed (simultaneous or ordered processing), the processing cycle of the use of context (incremental or not), and finally by the classification method (statistical or knowledge-based). As the authors remark, most of the approaches use simultaneous and non incremental processing.

Generally speaking, it is common sense to classify multimodal integration methods as *decision fusion* (or late fusion) or as *early fusion*. In the former category, independent unimodal hard decisions are taken and then fused in a second time. Incremental and ordered processing may be used, while the fusion can be performed by knowledge-based or probabilistic approaches. In early fusion, multiple low or mid-level features are extracted with soft decisions and fused in a unique multimodal space where the solution is sought usually with statistical methods. Incremental and ordered processing is, by definition, not possible. Audiovisual speech recognition is probably the most prominent example

of early fusion, where raw low-level features are directly fused. A number of indexing methods, on the other hand, use actually high-level features (like detected faces) which are fused nevertheless in a multimodal space, before any decision on the target semantics is taken.

As the focus of this study is on HMMs, they are treated separately in section 2.3. HMMs are an early fusion approach that exploits also the temporal dimension of a video.

2.1 Decision Fusion

In this approach, video is considered as a pool of features that are fused with explicit reasoning about the domain at hand. While some features like faces or transcribed speech are hard to extract and require sophisticated statistical methods, little or no machine learning is involved at the fusion stage. The question of conversion of the modalities is not answered in a uniform manner, being always data driven.

2.1.1 Knowledge-based

Of the first and straightforward attempts for multimodal integration is to combine features with hand-crafted domain-specific rules derived from careful inspection of the video.

Nepal *et al.* [69] detect scoring video segments in basketball broadcasts based on evidence from multiple modalities. They use as features audience excitation, motion analysis and text score appearance, which are fused with rules like "loud cheer should appear within 3 seconds after the goal" or "scoreboard should appear within 10 seconds after the goal". In [29], classification of video segments as commercials, sports, newscasts, etc., is performed on top of a series of auditory and visual low and mid-level features, like motion analysis and waveform statistics. The features are combined by rules that exploit domain knowledge and heuristics. For instance, commercials usually start and end with monochrome frames of short duration. The authors of [65] use concurrent analysis of both video and audio features for detection of violent scenes in movies. Guided by domain knowledge and carefull video inspection, they use features like fast motion, blood, gunfire, that coincide with interesting features in the soundtrack, like abrupt energy changes.

Successive Analysis

Many approaches use *successive analysis* of the video content, where interesting segments are detected in one modality and then the other modalities are successively analyzed to refine the results or to enrich the semantics. As there is no need to examine the whole video content across all modalities, successive analysis is a straightforward and fast way of integration. But there is always the risk of false rejection at an early stage or

propagation of errors at later stages. In [43], the authors segment and classify video as news or commercials by using audio features at a first level. Video features are then used to refine the segmentation. Finally, text features (transcribed speech or closed captions) are integrated to further refine the segmentation and attach a semantic label. Sports video can provide further ground for successive analysis, as game events are usually followed by cheering sounds, which are quickly and easily detected in the soundtrack. Results are further refined by processing information from the videotrack. High-level game semantics are then recovered, for instance in football [19] or in basketball [50]. Reversing the ordering, visual features are firstly processed by a Controlled Markov Chain in [53] and then the hypotheses are rescored with the help of audio loudness detection.

Finally, successive analysis is considered as *intermodal collaboration* in [5]. The authors search for pre-defined keywords corresponding to selected events in the closed caption stream of american football video. To detect the corresponding video shot that actually contains the game event, a search is performed in a time window around the appearance of the closed caption. Color-based features were used in order to calculate the distance of a given shot to a model distribution throughout this search. In [4], the same philosophy is used to detect and name events using the superimposed score indications and their transitions. In [6], audio information is added.

Clustering

One can easily notice in some kinds of video like news, that the scenes consist of a succession of shots of similar content. For example, in a dialog of two persons, we usually see the succession of two camera plans depicting these two persons. When entering into a new scene, the new thematic content is not usually related to that of the previous scene. In light of this remark, we can segment a video into scenes by detecting shots of similar content and grouping them into clusters that represent the scenes. The disadvantage of this approach is that it cannot be used in some domains where all the scenes share more or less the same content, like sports.

The use of audio or text information can provide robustness, compared to visual-only approaches [115, 86, 36]. In [102] for instance, dialog scenes in news can be identified by the frequent alternation of faces in the videotrack and of the same speakers in the soundtrack. In [59], major anchor persons are extracted from video. Face images are extracted and tracked in the videotrack and speech segment detection and speaker segmentation is performed in the soundtrack. Faces and speech segments are firstly grouped and then associated based on temporal correlation, or they are grouped based on audio-visual features. In [80], the authors parse in a successive manner and categorize news broadcasts using audio, video and text features. Video shots are firstly grouped based

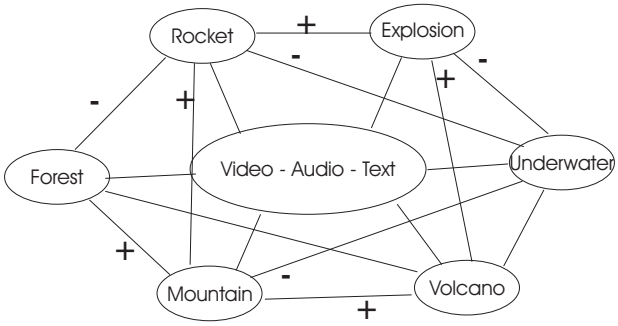


Figure 2.1
Illustration of a multinet [66]. Positive or negative signs indicate positive or negative interaction, respectively.

on color information. This grouping is further refined to form news stories with audio information (speaker change). Finally, textual resources from superimposed text or the closed captions is used to attach high-level labels to the stories. The exact label is given by an SVM classifier on top of natural text data. Ausiovisual clustering can be used also in movies video [89, 77], where a series of visual and audio features are extracted from each shot and then knowledge-based distance measurements are used to decide if two shots belong to the same scene.

2.1.2 Bayesian Networks

Baysian Networks offer the possibility to perform decision fusion without using any explicit rules regarding the context but, instead, to *infer* them through learning. Furthermore, prior knowledge is easily infused, making them a convenient tool for multimodal content representation.

Naphade and Huang [66] extract semantic concepts from movies using a novel Bayesian-based multimodal framework. They define the multiject, a probabilistic multimedia object that represents low and mid-level features like “sky”, “beach”, “human face”, etc. More complex semantics are represented in a Bayesian probabilistic network, the multinet, which contains a series of multijects connected according to their semantic relations (Fig. 2.1). The multinet encodes thereby in a straightforward way the relative context of the semantics. Apart from modeling complex semantics, the multinet can also help the inference of the detection of certain multijects through the relations of the graph, making it context-dependent. A question is raised for the conversion into the graph of multiple video features, as they are generally heterogeneous. Image-based features were collected in a frame basis, while temporal features such as motion and tracking in a shot basis. The soundtrack can be processed separately or the detection of certain sound classes can be further supported by the connections of the multinet,

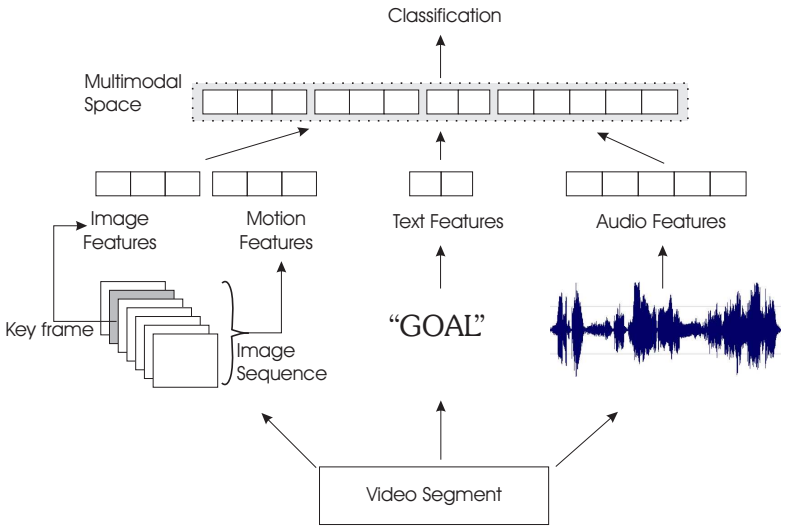


Figure 2.2

Example of an early fusion scheme. Given a video segment (corresponding to an audio clip or video shot for instance), data from the three modalities are optionally pre-processed and/or pre-classified independently. The features are then gathered in a multimodal space where the final classification is performed.

exploiting the relative context.

Information from video, audio and text is jointly modeled with a Bayesian Network of three layers, according to the semantic level of the features in [45]. Explicit synchronization is assumed at the video frame level. The system firstly filters out commercials and then classifies video segments as financial news or talk shows. Bayesian Belief Networks were also used in [27] for video classification on top of image and motion features, which are preprocessed with HMMs.

2.2 Early Fusion

In early fusion, a multimodal space is created by pre-processing data from various modalities independently and then concatenating features, as illustrated in Fig. 2.2. Supposing that the dimensionality of the visual (image and motion-based), textual and auditory features is V, T, and A, respectively, the dimensionality of the created multimodal space will simply be $V + T + A$. A first problem one has to face in early fusion is the conversion of these features into a common time scale due to the fact that the three modalities are generally sampled at very different rates. There can also be a limited or more extended asynchrony between the modalities. These two problems are particularly true for schemes that fuse low-level features as in audiovisual speech recognition. On the contrary, the problem of conversion can be bypassed when the features are also pre-classified

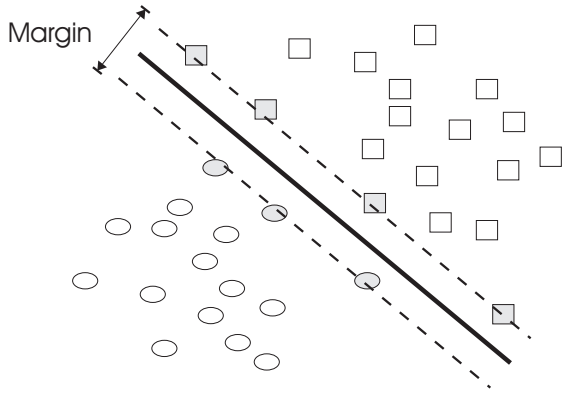


Figure 2.3
Illustration of the separation hyperplane in SVMs for the classification between two classes, marked with squares and circles. The support vectors are the examples that lie close to the boundaries of the two class. In SVMs, an optimal hyperplane is sought, in the sense of maximizing the separation margin.

to yield a series of unimodal but synchronized and homogenized descriptors, collected in a video shot basis, for instance. The drawback is then that this pre-classification stage may take actually hard decisions that affect the final classification.

Statistical methods are largely employed in the classification stage. Depending on the nature of data, decision trees [20] can be used, or more advanced techniques like Support Vector Machines or the Maximum Entropy Model. These techniques are used to automatically detect from a series of multimodal features the relevant ones to the target class.

2.2.1 Support Vector Machines

Support Vector Machines [103] (SVMs) are a popular choice among other classification methods like Multilayer Perceptrons, Bayesian methods, etc. First of all, SVMs provide an appealing theoretical framework where the optimal separation hyperplane is constructed using the most discriminating examples (the support vectors), as illustrated in Fig. 2.3. This guarantees optimal generalization especially when few training patterns are used, which is essential in video indexing as video data collection and annotation are always expensive. Furthermore, when few training data exist, the optimization problem of the SVMs becomes faster and more tractable to solve, compared to large scale problems like face detection [73].

In [58], early fusion is performed for detection of news segments in broadcasts. Auditory features at the clip level are further assisted by visual information (color and motion), to form a supervector of observations. Statistical techniques including GMMs

and SVMs were used for the classification. The addition of visual information improved the results in all cases, while SVMs performed better. In [55], news segments are classified into 5 categories like politics, social or sports. Auditory, visual and textual features are treated separately, on a news story basis. SVMs are used to pre-classify features from each modality. Then, the results of pre-classification are concatenated to a supervector and a Bayesian fusion rule was used for the final classification. SVMs were also used in [57] for the detection of weather reports inside news broadcasts.

In [88] the authors address the problem of event detection in field sports (like soccer, american football, etc) in its generality and without making game-specific assumptions. From a carefull examination of several sport videos from various genres, they found a series of (more or less) common features that occur after a major game event, like crowd excitation, close-up, on-screen graphics appearance, etc. After the extraction of these features (which is the main focus of the paper) in a shot-based manner, they classify them as interesting or not event using SVMs.

In [96], a comparison is provided of an early and a late fusion scheme of multimodal integration for learning semantic concepts such as "people walking" or "beach" in video. Visual features include descriptors of concepts like sand, sky, etc. Textual features obtained from the transcribed speech are fused in a shot-based way. In the early integration approach, the features are integrated in a multimodal space and then classification is performed with SVMs. For late fusion, unimodal SVMs pre-classify the features directly to the target semantic concepts, followed by a next-level SVM that arbitrates the final decision. For most of the concepts, late fusion performed better, but as the performances were close, it is difficult to draw safe conclusions.

2.2.2 Maximum Entropy Model

Another classification method that is widely employed is the Maximum Entropy Model (MEM) [9]. Given a set of descriptors $f_i(x)$ for the video data x, the likelihood of class ω is given by an exponential model:

$$p(\omega|x) = \frac{1}{Z(x)} \exp(\sum_i \lambda_i f_i(x)) \tag{2.1}$$

where $Z(x)$ is a normalization term and λ_i are the free parameters of the model, usually estimated via the Generalized Iterative Scaling algorithm [23]. The parameters λ_i weigh the contribution of each descriptor, allowing thus for human explanation of the solution provided by the algorithm.

The authors of [34, 35] used the Maximum Entropy Model (MEM) to detect and classify seven predefined categories of highlights in baseball games. Firstly, the video was segmented into shots. Then, features from the image (color and edge distributions,

motion, and player detection), sound (detection of applause, speech, etc.), and text (detection of words in closed captions) modalitities were collected to form a super-vector of features of dimensionality of 186. In addition, as features are generally asynchronous, all the features of 4 consecutive shots were fused into a 744-dimensional vector. As a result of the training process, the 30 most informative features were selected and, regarding the performance of the system, 70% and 60% average recall and precision rates were achieved, respectively. As training corpus, 10 games were collected of total duration of 32 hours and 3 games were reserved for testing purposes. The system demonstrated significant performance degradation when only image features were used. The authors also compared favorably the MEM framework to HMMs, having them operating in small sets of consecutive shots and with input features that of a single shot (186-dimensional).

2.2.3 Advanced Feature Wrapping

Some studies also address the problem of conversion of multimodal features to account for feature asynchrony and different sampling rates.

In [40], a large series of auditory, visual and textual features are fused to one super-vector using a specialized *feature wrapper*. During this conversion, features are processed taking into account feature derivatives, multiple thresholds for binarization and multiple time windows. This results into a high-dimensional multimodal feature vector, which is classified by a statistical classifier. To this end, the Maximum Entopy Model, Boosting approaches, and SVMs are compared in [41]. SVMs performed slightly better. The system was used to classify video segments as news or non-news.

In [94], the problem of context dependencies and the lack of synchronization between the different video modalities was addressed in a statistical framework to detect and classify soccer and news broadcast events. The authors started from the remark that a given video event often consists of a series of low-level indices from each modality, defining thus with their relations in time a suitable context. For example, a soccer goal event could start with a camera motion (image modality) which is followed by speaker excitement (audio modality) and finally, a few seconds after, by a goal keyword (textual modality). They proposed the use of the TIME framework to capture this context, where 13 relationships of type of precedes, overlaps, meets, etc and their inverses are showed to be sufficient enough to capture any relation and contextual dependency between two features. The image modality was chosen as a reference modality, it was segmented into shots and then, for each shot, all the relevant TIME relations between the features from various modalities were gathered into a super-vector with binary elements. Standard statistical techniques like C4.5, Maximum Entropy, and SVMs were compared to a soccer video benchmark, where the last one clearly outperformed.

2.3 Hidden Markov Models

An important aspect of video data is their temporal dimension. Indeed, depending on the video genre and the production rules, video events occur with a temporal order that will finally give the semantic label. In the previous approaches, a unique video segment like a video shot was analyzed individually by collecting features from it and, optionally, from its neighboring segments. The temporal dimension of the video was thus ignored or modeled explicitly by hand-crafted rules. Hidden Markov Models (HMMs) provide a powerful statistical framework for handling temporal data and it is thus a natural candidate for learning temporal dependencies in video.

HMMs were originally developed some decades ago and still are the dominant technology in the field of speech recognition [81]. They have found application in many other fields too, including recognition of the human sign language [97], gesture recognition [25], video surveillance [46], DNA analysis [74], character recognition [62], and face detection and recognition [67].

2.3.1 Video Indexing with HMMs

W. Wolf [108] introduced HMMs as a *statistical parser* of the syntax of a video document. Low-level video features are considered as the observations $O_{1:T}$ of a hidden Markovian stochastic process $s_{1:T}$ of length T that represents the video syntax. The Viterbi algorithm is then employed to recover the syntax, i.e., the most likely hidden state sequence S^*:

$$S^* = \arg\max_{s_{1:T}} p(O_{1:T}|s_{1:T})p(s_{1:T}) \tag{2.2}$$

given the observation sequence $O_{1:T}$ and the model parameters, which are the hidden state transition probabilities and the state-conditional observation probabilities $p(O_t|s)$. In this context, the problem of video syntax extraction can be considered as similar to the one of speech recognition, where we attempt to extract the transcription (video syntax, in the case of video indexing) of a spoken text (video file, respectively). The use of HMMs for video segmentation into shots [12, 7] follows the same philosophy. The difference here is that the hidden states represent low-level video syntax, like being in a hard cut, dissolve, etc.

Some video genres exhibit in their syntax an inherent *hierarchical structure* that it is also meaningful to be recovered along with their base (flat) syntax. In a lot of sports video for instance, the game is naturally divided into sets, points, etc. This structure is easily encoded in the transition probabilities of the HMM resulting to a Hierarchical Hidden Markov Model (HHMM) [28]. The hidden states are now divided into internal states that guide the hierarchical Markovian process and into emitting states, which are associated with the observations as in flat HMMs. The Viterbi optimization is always

provided by eq. 2.2 with some minor modifications to account for non-emitting states. HHMMs have been used for sports [110, 113] or educational [78] video analysis.

In addition to structure parsing, HMMs can also be used to *classify* video segments to one or more classes, considering video indexing as a problem of temporal pattern recognition. Indeed, given an HMM λ_i that models class i, one can compute the likelihood $p(O_{1:T}|\lambda_i)$ that the model has generated the sequence $O_{1:T}$. The class label ω^* of the video segment is then simply given by the model that maximizes this likelihood:

$$\omega^* = \arg \max_{i=1...M} p(O_{1:T}|\lambda_i) \tag{2.3}$$

among M candidate classes. This approach was followed for baseball [18] or soccer [3] highlight detection and classification, for sport genres classification [32] or TV programs classification [60] on top of video-only features.

2.3.2 State Synchronous Multimodal Fusion

The first and direct approach to multimodal fusion with HMMs is to simply concatenate features collected from all the modalitities in a multimodal space, much the same way as in Fig. 2.2 for the early fusion approaches. A *state synchrony* is then assumed, forcing the modalities to follow the same sampling rate and also the same model topology. The exact way the sampling rate conversion if performed is always task-dependent. When image-based features (sampled usually at 25 fps) are to be converted to the audio frame rate (usually 100 fps), like in audiovisual speech recognition [79], interpolation can be used. For the opposite conversion, audio features can be averaged [7]. Nevertheless, in many video indexing approaches, features from large video segments (like a video shot) are collected and then pre-classified to yield descriptors. In so doing, the problem of the sampling rate conversion is artificially bypassed.

In a formal definition, the state-conditional observation probabilities for two converted and synchronized modalities $O^{(1)}$ and $O^{(2)}$ is given by:

$$p(O^{(1)}, O^{(2)}|s) = p(\mathbf{O}|s) \tag{2.4}$$

where $\mathbf{O} = [O^{(1)}O^{(2)}]$ represents multimodal features after concatenative fusion. An example of a concatenative HMM is given in Fig. 2.4.

This fusion scheme has become a popular choice in the video indexing community due to its simplicity. Of the first studies, Boreczky *et al.* [12] use image and motion-based histograms sampled at the video frame rate, which are further supported by concatenative fusion of audio-based features. Considering the task of video shot segmentation, this pure statistical HMM-based method was proposed in contrast to the tedious manual thresholding, yielding good experimental results. A similar approach was followed in [7], where separate HMMs were used to represent each shot transition type.

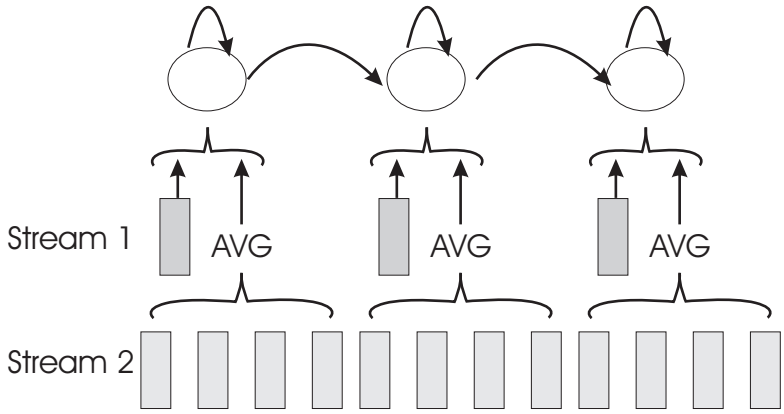


Figure 2.4
Fusion of two streams sampled at different rates with state synchronous HMMs. Observations from the second stream are suitably averaged and then synchronized with the ones of the first stream.

In [42, 106], HMMs are used to classify different video sequences as belonging to a predefined class like commercials, news broadcasts, basketball games, etc. For the integration of audiovisual information, the concatenative fusion was compared to some other fusion schemes of unimodal HMMs, namely product of HMM likelihoods, successive HMM-based analysis and MLP-based fusion. The addition of video features to the audio ones generally improved the performance, while the direct concatenation and the product of HMM likelihoods performed better, regarding the fusion scheme.

In [2], an HMM-based framework for human dialog detection is proposed. The video track is segmented into shots and location change and face presence/absence information is extracted. In parallel, the authors used as audio features sound classes like speech, silence, etc that are collected according to the video track segmentation. The authors modeled the temporal evolution of these features using HMMs. Different HMM topologies were compared on simulation data.

In [26, 44], the authors try to segment TV news broadcasts into topics using HMMs. A number of image features is extracted from each video frame. On top of these features, the authors build an HMM that models the broadcast scene transitions and editing effects. The segmentation provided by this model is further refined by some rules. This pure video-only method is compared to an audio-only segmentation, based on BIC. Finally, the authors proposed an audiovisual approach, where the existence or not of an audio cut is added to the image features of the HMM. This last approach yielded the best result in a set of 9 broadcasts of 2:15 hours duration.

In [24], HMMs are used to classify TV broadcasts as news, commercials, sitcom, and soap. Superimposed text and faces are detected and tracked in the video. Each shot is then pre-classified into one of 15 classes, such as “Anchor person with text”. The

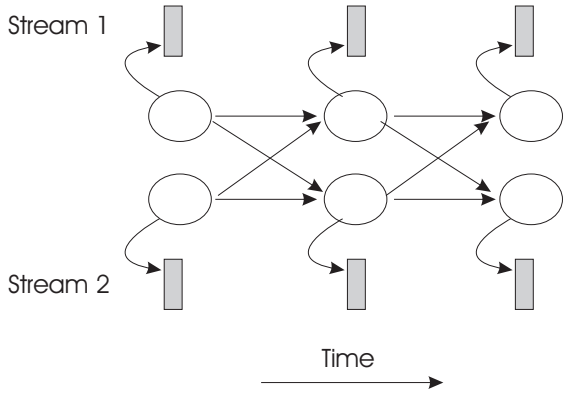


Figure 2.5
The Coupled HMM models concurrently two streams 1 and 2. Arrows denote dependencies through time.

evolution of these features is modeled with four discrete HMMs and TV segments are classified according to the produced likelihoods.

Coupled HMMs

Apart from concatenative fusion of the observations to jointly model two streams, another approach consists of using two (or more) *concurrent* unimodal HMMs. An illustration of this model, called Coupled HMM [15], is given in Fig. 2.5. In this model, the transition probabilities are conditioned not only on the previous state of the same stream, but also on the previous state of the other stream. Formally, supposing that the hidden state variables of the two HMMs are s_t and q_t, then the transition distributions are expressed as $p(s_t = i|s_{t-1} = k, q_{t-1} = j)$ and $p(q_t = i|s_{t-1} = k, q_{t-1} = j)$. The two streams $O^{(1)}$ and $O^{(2)}$ are again supposed to be synchronous and the observation distribution are modeled independently as $p(O_t^{(1)} = o|s_t = i)$ and $p(O_t^{(2)} = o|q_t = i)$. Coupled HMMs have been used in audiovisual speech recognition [79] or for sports highlight detection [112].

2.3.3 Asynchronous Multimodal Fusion

The underlying assumption of the HMM-based multimodal fusion that all the modalities are synchronous does not hold generally. In order to remedy for the forced synchrony between the modalities and to increase modeling capabilities, HMM variants have been proposed in the audiovisual speech recognition and video indexing communities. They are the subject of this section.

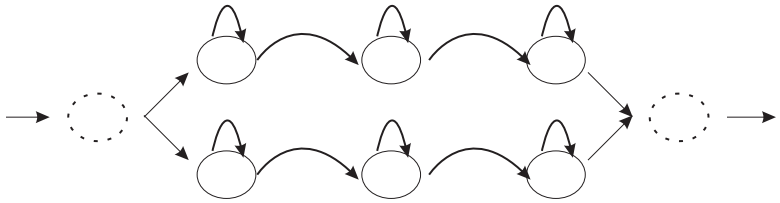


Figure 2.6
Example of a Multistream HMM with fixed synchronization points, marked with dotted circles. The two unimodal HMMs share the same number of states, although they could have different topologies. Different sampling rates are allowed too.

Multistream HMMs

In Multistream HMMs [13], each modality (or stream) is modeled by independent HMMs which are forced to synchronize in some fixed points. In *synchronous* Multistream HMMs, the states themselves are these points. In practice, these models do not differ with the state synchronous HMMs, except for the explicit assumption of independence of conditional observations, optionally fused with a weighting scheme:

$$p(O^{(1)}, O^{(2)}|s) = p(O^{(1)}|s)^{w_1} p(O^{(2)}|s)^{w_2} \tag{2.5}$$

where usually (but not necessarily) $w_1 + w_2 = 1$. Instead of using fixed global weights, state-dependent weights or reliability measurements on each modality can also incorporated in this fusion scheme [33].

In *asynchronous* Multistream HMMs, the synchronization points are extended beyond the hidden states, like the end of phones in audiovisual speech recognition. Between these points, the streams are considered independent and are modeled by individual unimodal HMMs as shown in Fig. 2.6. The key idea is that the unimodal HMMs can follow different topologies and also operate at the native sampling rate of their modality. The likelihoods (or scores) produced by the unimodal HMMs are recombined at the synchronization points with probability products as in eq. 2.5 or with any other combination function like Neural Networks.

For convenience reasons during decoding, asynchronous Multistream HMMs have been used with the form of a *product* HMM in audiovisual speech recognition. In this model, the unimodal HMMs of Fig. 2.6 follow the same number of states and sampling rates. Then, product states can be introduced that encode every possible combination between the unimodal states, as depicted in Fig. 2.7. In this way, the product HMM is in fact an equivalent HMM to the asynchronous Multistream model, but where the streams are (product) state-synchronous. The observation distributions follow the scheme eq. 2.5, with the only difference that the hidden states are the product ones. As the product HMM contains much more hidden states compared to the unimodal HMMs, parameter

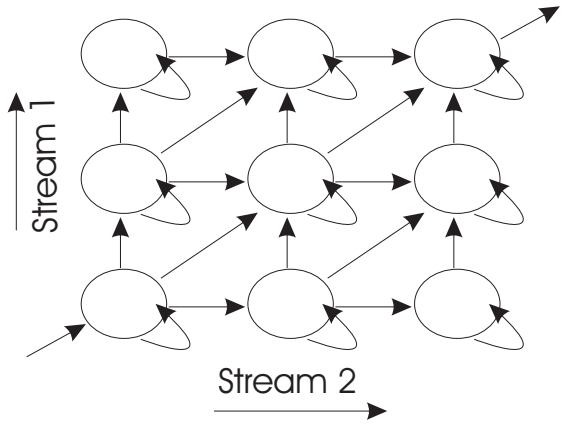


Figure 2.7
The resulting product HMM of Fig. 2.6.

tying is largely used to drastically reduce the number of free parameters. More precisely, the vertical states of Fig. 2.7 share the same distributions with the unimodal HMM of stream 1, as do the horizontal ones with the HMM of stream 2. The extra parameters to be estimated in the product HMM are thus the product transition probabilities. The Viterbi decoding (eq. 2.2) for the product HMMs is formally defined as:

$$S^* = \arg \max_{s_{1:T}^{(1)}, s_{1:T}^{(2)}} p(O_{1:T}^{(1)}|s_{1:T}^{(1)})^{w_1} p(O_{1:T}^{(2)}|s_{1:T}^{(2)})^{w_2} p(s_{1:T}^{(1)}, s_{1:T}^{(2)}) \tag{2.6}$$

Coupled HMMs (section 2.3.2) are also usually transformed to an equivalent product HMM.

The Asynchronous HMM

The Asynchronous Hidden Markov Model (AHMM) [8] is a special HMM architecture designed to jointly model pairs of lightly de-synchronized sequences, sampled at different frame rates. We want to jointly model two streams $O_{1:T}^{(1)}$ and $O_{1:S}^{(2)}$ of length T and S respectively, with $T > S$. AHMMs process the two streams by letting the smaller one be stretched in time in order to meet a better match with the first stream. The longer sequence is generated classically as in all Markovian models by entering into a hidden state, emitting an observation symbol, then switching into a new hidden state, and so on. The novelty in AHMMs is that at some time instants, we allow the hidden state to emit *two* observation symbols, one from the longer sequence, as normally, and also the next observation symbol of the shorter sequence. The time instants of the doubled emissions, i.e., the *alignment* of the two streams is expressed via a new hidden variable $\tau_{1:T}$. The physical interpretation of this hidden variable is that it gives at each time instant the corresponding (matched) time index of the shorter stream, as depicted

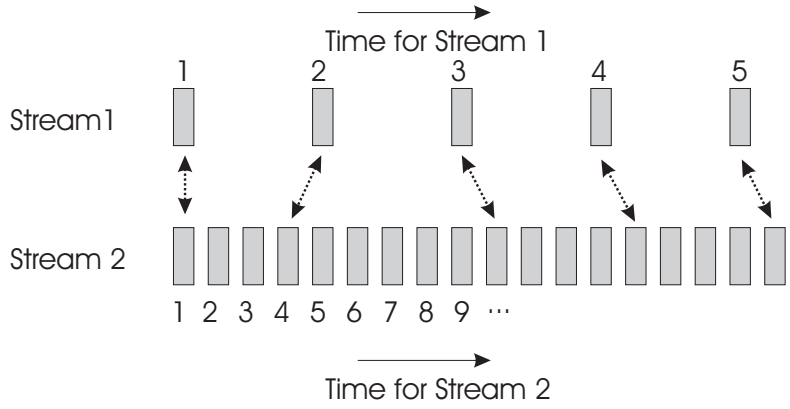


Figure 2.8

Illustration of the alignment of two streams offered by the AHMM. The dotted lines give the matched observations of stream 1 to stream 2. The corresponding sequence of the hidden variable $\tau_{1:T}$ is (1 1 1 2 2 2 2 2 2 3 3 3 3 4 . . .).

in Fig. 2.8. The distributions to be modeled are firstly the transition and emission $p(O_t^{(1)}|q_t = i)$ distributions, as in normal HMMs. Furthermore, AHMMs define the joint emission distribution $p(O_t^{(1)}, O_s^{(2)}|q_t = i)$ of the two streams being in time instants t ans s, respectively. Finally, the alignment distribution $\epsilon(i, t)$ is defined. It provides the probability of emitting the symbol s of $O^{(2)}$, while being at state i and time instant t in stream $O^{(1)}$. During Viterbi decoding, the search is performed not only through all the possible hidden state paths but also through all the possible alignments between the two streams. The solution of the maximization problem will provide also the most likely alignment:

$$(s, \tau)^* = \arg \max_{s_{1:T}, \tau_{1:T}} p(O_{1:T}^{(1)}, O_{1:S}^{(2)}|s_{1:T}, \tau_{1:T}) p(\tau_{1:T}|s_{1:T}) p(s_{1:T}) \tag{2.7}$$

where $p(O_{1:T}^{(1)}, O_{1:S}^{(2)}|s_{1:T}, \tau_{1:T})$ translates to joint or not emission probabilities according to $\tau_{1:T}$. AHHMs were introduced in audiovisual speech recognition, in order to account for noisy desynchronization of the two streams.

In [63], AHMMs were used and compared to other HMM-based alternatives in a task of analysis of group actions in meetings. The goal is to segment and classify audiovisual material from group meetings in actions like monologue, presentation, etc. Visual features like detection of head/hands blobs were extracted from the visual stream at a frame rate of 5 Hz. Audio features like 'speech activity' at different locations/microphones, pitch, etc, were extracted from the audio track and finally downsampled to 5 Hz, as with the video features. For the case of AHMMs, the audio track was sampled at 10 Hz as AHMMs naturally operate on streams sampled at different rates. Various baseline HMM approaches were tested like state-synchronous HMMs, coupled HMMs, and

product HMMs. The last model allows for action-synchronous modeling, similar to phoneme-synchronous modeling with product HMMs in audiovisual speech recognition. State-synchrony performed better than product HMMs. Results were further improved by AHHMs, as they can easily handle natural desynchronizations that introduce confusion to state-synchronous HMMs.

Cascade of HMMs

Sharing with Multistream HMMs the idea of synchronization points and the use of independent models inside them, architectures of cascade of HMMs can be built (Fig. 2.9). A video is segmented according to some a-priori fixed synchronization points, like at every 1 second or at the shot boundaries. The respective video portion of each modality is then processed independently, which allows for native sampling rate and varying HMM topologies. The outcomes from the HMMs thus obtained are concatenated and given as input to a Markovian process of the next layer. This scheme has the advantage of firstly fusing asynchronous unimodal HMMs, and secondly, the higher-level HMMs can capture interaction and semantics of a higher order, in a way similar to HHMMs. The HMMs of each layer are trained independently, which provides an interesting degree of modularity in the modeling.

Oliver *et al.* [70] presented an interesting approach to infer office activity based on a cascade of HMMs, called Layered HMM (LHMM). A portion of the input sequence is sampled at fixed time intervals and is classified as belonging to one class. This classification is performed with a bank of HMMs, each of them modeling one class, by choosing the class label corresponding to the higher HMM output. This portion of input sequence will provide one observation symbol for the HMMs of the next layer, and so on until the highest layer is reached. This topology was used for inferring office activity like "Phone conversation", "Face to Face conversation", etc from low-level features like Zero Crossing Rate (audio), Motion Density (images), recorded keyboard activity, etc. At the first layer, HMMs are trained to classify the audio features as "Human speech", "Music", etc, and the visual ones as "Nobody present", etc. The input signals (both video and audio) was processed at fixed manually-chosen time intervals of 1 second. The output of these classifiers are given as input features along with keyboard activity features to the second (and last) layer of the hierarchy, where the activity is inferred by using another bank of HMMs. The authors compared the standard state-synchronous concatenative HMM to the LHMM approach in a corpus of 60 minutes of office activity, demonstrating the superiority of the latter.

In [117], a two-layer HMM approach was used to attack the same problem of group action recognition with AHMMs, as in [63]. The role of the first layer HMMs is to model individual actions (i.e., person-specific) from the raw audiovisual data. The out-

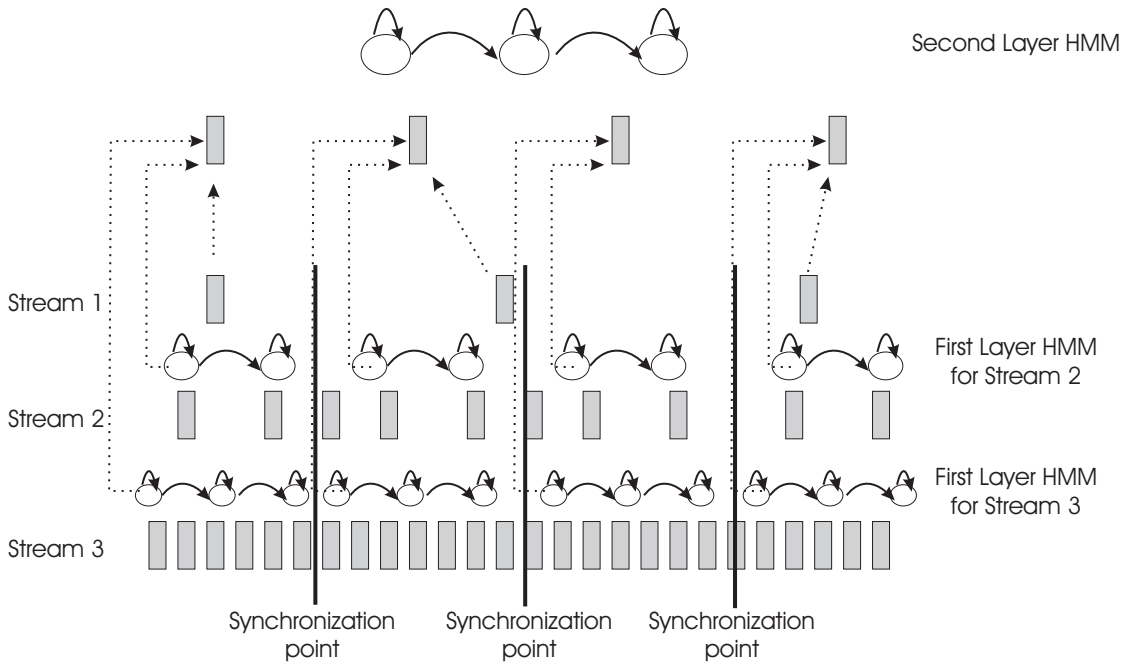


Figure 2.9

Illustration of a cascaded structure of HMMs. The HMMs for the streams 2 and 3 operate at constant time steps, using their own topologies and sampling rates. Their results are then fused to feed the HMM of the second layer. The features of stream 1 are fused directly with the outputs of the other streams. For instance, stream 1 could be a modality (like text) that captures higher-order semantics and with its sampling rate being rather sparse compared to the other streams.

puts of these HMMs are given as input to the second layer, as long as with some other group-specific features extracted from raw data. The role of the second layer is to model interaction between individuals and finally to give the desired structure analysis of the video. The first layer performs thus a low-level audiovisual fusion, while the second layer integrates some higher-order information. The state-synchronous early integration approach, the late fusion of unimodal HMMs and the AHMMs were used in the first layer. As input to the next layer the authors provided directly the likelihoods of the HMMs of the first layer (soft decision) or they performed a preclassification (hard decision) before feeding it. Regarding audiovisual integration, the AHMM and the early integration performed the best, verifying the conclusions of [63]. An observable performance improvement was noticed when using the two-layer HMM as opposed to the flat (single-layer) one. Finally, soft decisions performed slightly better than hard decisions.

Some form of a two-layered HMM was also used in [111] for structure analysis of soccer video using image and motion features. The second layer was used here to capture high-order dynamics, rather than performing multimodal integration.

2.3.4 Dynamic Bayesian Networks

In section 2.1.2, Bayesian Networks were presented as directed graphical models aiming to capture dependencies between random variables. The presence (absence) of an edge denotes dependency (independency) between the nodes of the graph that represent the random variables. Each edge in the graph thus defines a conditional distribution probability, to be estimated from data. Dynamic Bayesian Networks (DBNs) extend this framework in order to model stochastic processes that extend through time. Each node in the graph represents the realization of the random variables at each time instant, while the edges can represent dependencies not only through variables but also through time. DBNs offer in fact a generalization of HMMs in the sense that the Markovian assumption and state-conditional observations can be a small subset of all possible dependencies in the graphical model. A DBN representation of an HMM is depicted in Fig. 2.10. DBNs offer the possibility to discover and model dependencies like $p(Q_t|O_{t-1})$ that is impossible with HMMs.

A first challenge in DBNs is to provide efficient algorithms to automatically infer the graph topology through learning from data. Alternatively, the topology can be determined manually. Given the topology of the DBN θ, the inference $p(O_{1:T}|\theta)$ for a sequence $O_{1:T}$ is factored as:

$$p(O_{1:T}|\theta) = \prod_i P(X_i|\Phi_i, O_{1:T}) \tag{2.8}$$

where X_i is a node in the graph and Φ_i denotes its parents. The exploration of fast and efficient algorithms to estimate this quantity (analog to Forward propagation in

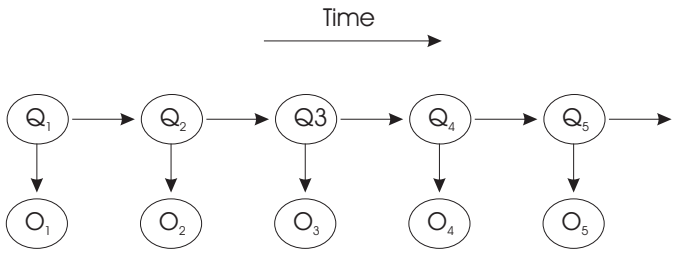


Figure 2.10
A DBN representation of an HMM. Arrows denote dependencies between the random variables $Q_1, Q_2, Q_3 \ldots$ (hidden states) and $O, O_2, O_3 \ldots$ (observations).

HMMs) constitutes the second challenge in DBNs. The interested reader is referred to [64] for algorithms for topology and fast likelihood inference with DBNs. The DBN representation of numerous HMM variants is also presented.

Regarding multimodal integration, DBNs offer the possibility to represent complex dependencies between the modalities or to discover new ones through learning. In [75], audio, video and text features are sampled at a unique frame rate, as in state-synchronous HMMs. DBNs are then used to explore time synchronous or asynchronous dependencies between the features and concept classes. In [71], Oliver *et al.* compare DBNs and HMMs in the framework of Layered HMMs analyzed in section 2.3.3. The first layer was left intact and the comparison was held in the second layer, where DBNs replaced HMMs. DBNs performed better by exploring relations between hidden states that HMMs were unable to capture.

DBNs have been also used as generic framework from which can be derived variants of HMMs, like product HMMs [68] for audiovisual speech recognition or Hierarchical product HMMs [104] for sports highlight extraction.

2.4 Discussion

Decision fusion is a simple and straightforward approach to multimodal integration. By hard-wiring domain knowledge, one can accomplish a given task easily and effectively, but the drawback is that handling large-scale problems can become quickly intractable. In addition, experience acquired in one task is not transferrable to other tasks or environments as the domain rules may completely change.

On the other hand, the use of statistical approaches is more promising for solving large-scale and complex problems, for both unimodal and multimodal scenarios. In the last case, a unique multimodal space is created where features from various modalities are projected and treated homogeneously. As we have seen in section 2.2, concatenative fusion of the unimodal features is widely used. The issues of good generalization in the

high-dimensional multimodal space and of the human-interpretable feature selection is efficiently adressed by using SVMs or MEM as the statistical classifier. On the other hand, questions regarding sampling at different frame rates or feature asynchrony are not addressed in a uniform manner. Most of the studies use mid-level descriptors, acquired after a pre-classification stage. The unimodal features are thus artificially converted and synchronized. The use of advanced feature wrappers [40, 94] has been proposed as an alternative, where features at all possible time scales or time correlations are fused, in order to select through learning the most appropriate ones.

HMMs provide a powerful framework for capturing the temporal dynamics of a video. This is important because in a lot of tasks the ordering of the video events (i.e., the video syntax) alone can reveal semantics. Most of the HMM-based approaches use again concatenative fusion of unimodal features, assuming state-synchrony of the observations. This assumption imposes however a lot of constraints during modeling: features should be sampled at (or converted to) the same frame rate, observation should be synchronous, and finally, all modalities should follow the same model topology. AHMMs [8] can address the first two constraints by attempting to re-synchronize streams that are sampled at different frames rates and exhibit a certain degree of asynchrony. Multistream HMMs can further allow for different topologies for every modality. For practical reasons however, they usually take the form of product HMMs, where the same topologies and frame rates are used in order to reduce the model complexity. An additional problem with product HMMs, which holds for AHMMs too, is that fusing more than two modalities will explode the complexity of the models and of the Viterbi decoding. In the framework of Layered HMMs [70], different topologies and frame rates can be used for every modality, but hard synchronization points, fixed a-priori, are assumed.

We will see in this study how the above-mentioned problems can be solved using Segment Models [72], which unify Multistream and Layered HMMs into a novel framework for multimodal integration. Firstly, a duration model is added to Multistream HMMs. The synchronization points between the modalities are not fixed but are now left as part of the optimization problem. Each modality can be sampled at different frame rates and modeled by its native topology. Second, observations from all the modalities and within the synchronization boundaries are assigned to a common "multimodal" hidden state. This hidden state belongs to a higher semantic level and models the video like the second layer of a Layered HMM does.

CHAPTER 3

Feature Extraction

The automatic extraction of the relative video features is described in detail in this chapter, along with a brief literature survey on tennis video processing in section 3.6. The videotrack and the soundtrack are processed separately, as shown in Fig. 3.1. The video is segmented into shots by hard cut detection. Dissolves are then detected in each shot. Given the starting and ending points of the dissolves, a new type of shots is defined, called dissolve shots. A reference court view, unique for every video, is automatically extracted using color and edge information. The key frame of each shot (defined as the middle one) is then compared to it, to provide a visual similarity measurement. This descriptor, along with the shot duration and a binary indicator for dissolves shots form the set of visual features. The audiotrack is segmented into homogeneous segments and then three key sound classes are detected, namely ball hits, applause and music. The soundtrack is thus transformed into a stream of binary indicators, one for each sound class.

3.1 Description of the Corpus

Before proceeding to feature extraction details, a brief description of the corpus used in this study is given in this section. 6 tennis broadcasts were recorded and kindly provided by INA[1]. They correspond to the following tennis matches:

- Agassi - Safin, Open Paris 1999, 7-6, 6-2, 4-6, 6-4

[1]Institut National de l'Audiovisuel, France. http://www.ina.fr

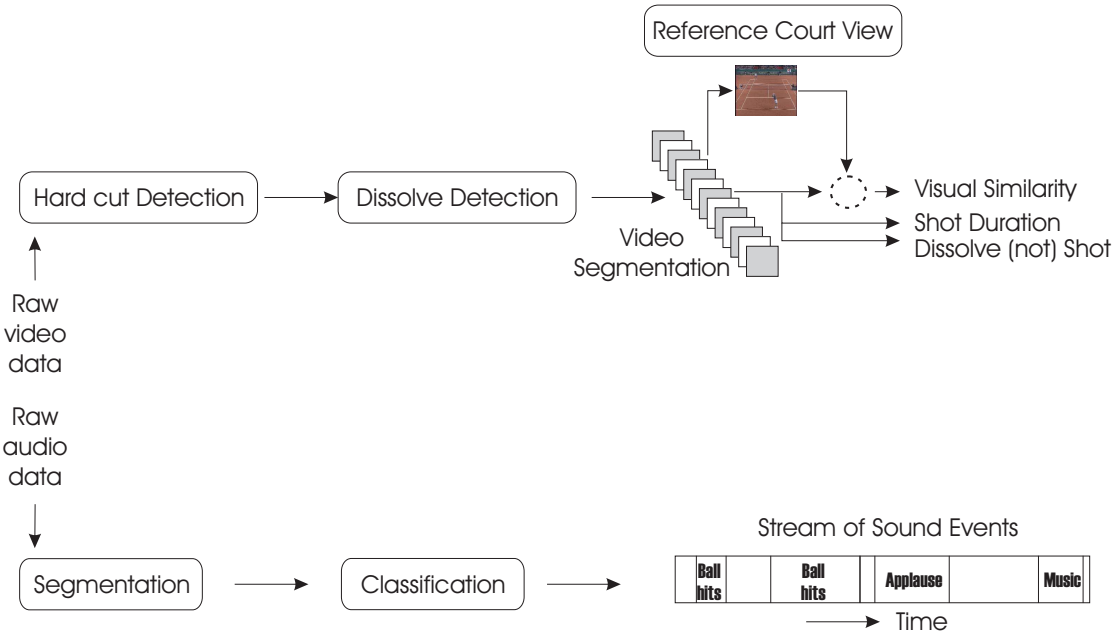


Figure 3.1
Outline of the feature extraction process.

- Pioline - Hewitt, Davis Cup 1999, 7-6, 7-6, 7-5
- Pioline - Philippoussis, Davis Cup 1999, 3-6, 7-5, 1-6, 2-6
- Sampras - Rusedski, Open Paris 1998, 4-6, 6-7, 3-6
- Grosjean - Philippoussis, Davis Cup 1999, 4-6, 2-6, 4-6
- Capriati - Clijsters, Roland Garros 2001, 1-6, 6-4, 12-10

The last tennis match was held outdoors. The parts of the broadcasts before and after the actual tennis match were manually removed from the videos. Nevertheless, the programs still contain commercials and interviews that occasionally appear during the match. Video frames were extracted from raw MPEG data in the RGB colorspace and in the size of 288×352 pixels. The audiotrack was sampled at 46 kHz, except for the last match which was downsampled at 41 kHz.

The first three broadcasts were reserved for testing purposes throughout this chapther and the subsequent ones.

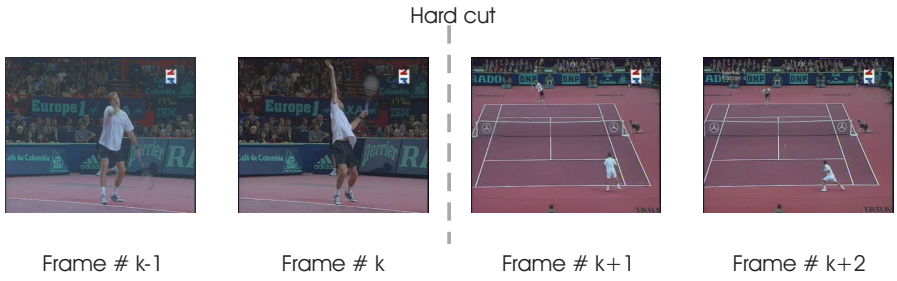


Figure 3.2
An example of a video hard cut. A visual discontinuity appears between two successive frames.

3.2 Video Hard Cut Detection

Problem Definition and Relative Literature

Video hard cuts are defined as points where an abrupt change occurs in the visual content of the video frames (Fig. 3.2). Typically, they are results of the editing effect of switching camera view in order to stress the viewer's attention to a new storyboard or to present another (visual) aspect of the displayed scene. Hard cuts segment the video into *shots*, which are video segments of homogeneous visual content. As it is hard to imagine any produced video without hard cuts, their detection has attracted the interest of the video indexing community and numerous approaches have appeared in the relative literature [56, 30]. Hard cuts are detected with pure signal processing techniques, in contrast to scene (thematic units) cuts, which require high-level reasoning. Due to this reason, shots usually form the basis of the vast majority of video abstraction techniques.

The detection of hard cuts involves the examination of the visual (dis)continuity between frame pairs or inside local neighbors in the video track. One has firstly to make a choice regarding the image features to be used. Many techniques (e.g., [114]) rely on compressed-domain video data like the MPEG DCT coefficients, if fast processing is required. In the uncompressed domain, histograms collected from the pixel intensities have been used, like color [30] or edge histograms [116], for instance. Motion features have also been proposed (e.g., [14, 30]), where usually a large motion discontinuity is sought. As in this study performance issues are not adressed, feautures were collected from uncompressed video data. Grayscale histograms were used as the simplest and yet effective choice.

A second choice regards the detection method. In many approaches, image pairs are examined and when the two histograms are found to be distant enough, a hard cut is detected. A number of distance metrics between image histograms have been borrowed from the field of image indexing, like the Euclidean distance or the χ^2 test [85]. Having decided about the metric to be used, a global threshold T is needed to be set by manual

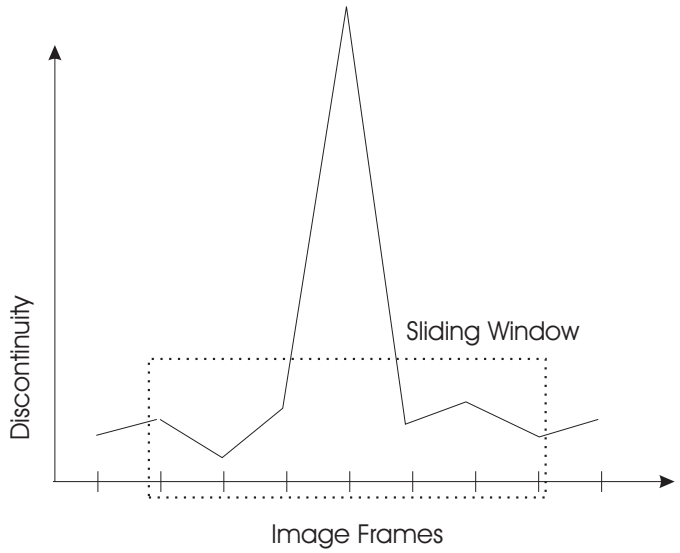


Figure 3.3
The use of a sliding window for hard cut detection. The discontinuity in the middle of this window far exceeds the remaning ones.

inspection or by some statistical optimality criterion. However, this use of a global threshold was found to be sensitive to video noise and also to the changing of statistics from video to video. The appearance of a fast camera motion, for instance, would require high thresholds, which in turn can result in a lot of misses in the same or other videos.

This shortcoming of the global threshold is efficiently addressed if using a time-varying threshold $T(t)$ [114], which is adaptive to the local variations of the histogram statistics. The key idea is to use a *sliding window*, where the last N histogram discontinuities are computed (Fig. 3.3). A hard cut is detected if a much larger discontinuity appears in the middle of this window, compared to the remaining ones. In this way, the threshold $T(t)$ is in fact a function of the local discontinuities appearing in the window. The presence of strong motion fields will result in strong but uniform discontinuities, so it will leave the hard cut detection unaffected.

Description of the Method

The adaptive threshold selection framework of [101] was used in this study. Given the grayscale image histogram H_t of the frame t and a window of size $2w + 1$, the hard cut detection proceeds as follows:

1. The histogram distance $D(H_k, H_{k-1})$ of the frames k and $k - 1$ is computed:

$$D(H_k, H_{k-1}) = \frac{1}{L} \sum_{i=1}^{M} |H_k[i] - H_{k-1}[i]| \tag{3.1}$$

where $k = 1 \dots 2w + 1$, L is the total number of pixels of the frames, and M is the number of bins of the histograms.

2. In true hard cut, the distances attain their maximum value at the middle frame $w + 1$ of the window, i.e., $D(H_{w+1}, H_w) \geq D(H_j, H_{j-1})$, $j = 2 \dots 2w + 1$.

3. The distance $D(H_{w+1}, H_w)$ should be greater by a factor of α to the mean of the distances within the window, excluding $D(H_{w+1}, H_w)$. To handle situations where the distances are very close to zero yielding thus noisy close-to-zero estimations of the adaptive threshold, an additive term c is added to every distance $D(H_j, H_{j-1})$. Finally, a hard cut is detected at the frame $w + 1$ if:

$$D(H_{w+1}, H_w) + c \geq \frac{\alpha}{2w} \sum_{j=2, j \neq w+1}^{2w+1} (D(j, j-1) + c) \tag{3.2}$$

The values of the parameters were set to $M = 256$, $w = 3$, $\alpha = 1.7$ and $c = 0.25$. The values of the last two parameters were determined after manual inspection of the training sequences. An example of the evolution of the values of the adaptive threshold against the ones of the histogram discontinuity is given in Fig. 3.4.

A very good performance was noticed in both training and test videos. In total, 10,775 hard cuts were detected with a negligible number of false detections and misses.

3.3 Dissolve Detection

Problem Definition and Relative Literature

Dissolves are defined as progressive transitions between shots where the first one fades out while the second one fades in [56]. An example of such a transition is given in Fig. 3.5. In tennis video, as in other sports broadcasts, they are widely used to signal the start and the end of replays and highlights. Indeed, it was noticed in the tennis corpus of this study that almost all replays start and end with dissolves and with no other production effect. In addition, they separate shots inside (multiple) replays, in a way that a replay shot is always surrounded by two dissolves.

As it was generally reported in the relative literature [56, 30], the detection of progressive transition is far more challenging compared to hard cut detection. Indeed, when comparing two successive frames, the discontinuity will take small values inside a shot due to the visual homogeneity, while it will form some large picks upon hard cuts. But when comparing two frames that are distant in time, large discontinuity values are expected whether a dissolve intervenes or not. For example, there could be also a progressive appearance of an object or a light motion field that would drastically change the statistics of the histograms in a cumulative fashion.

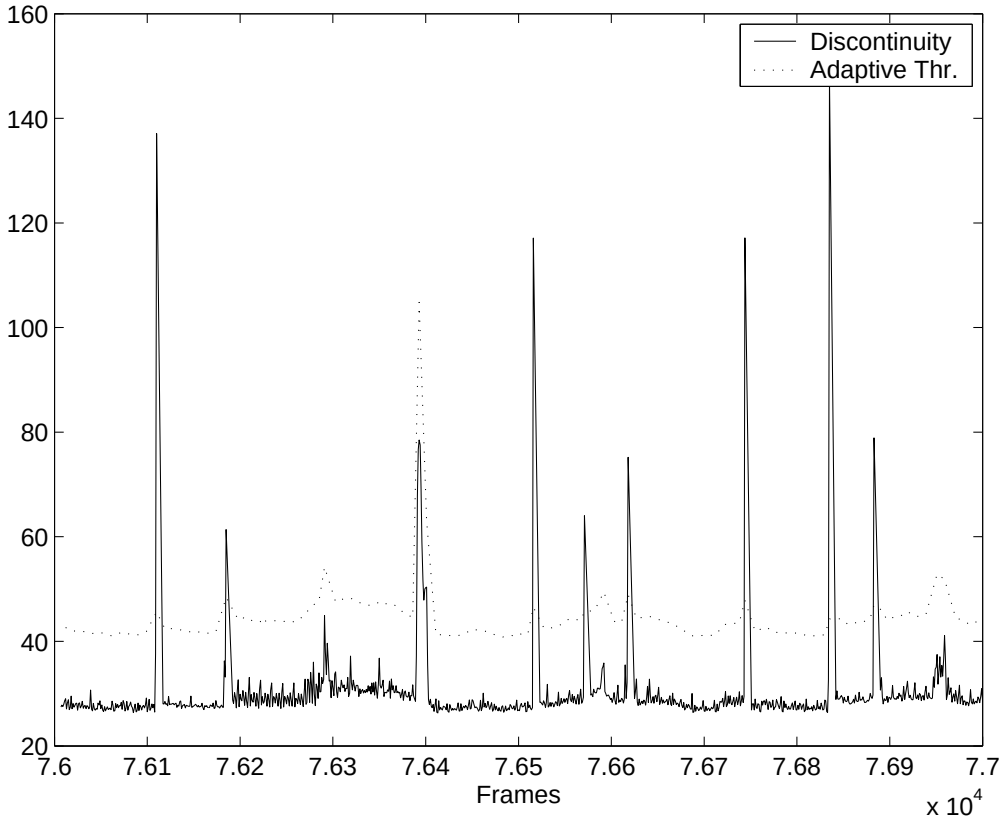


Figure 3.4

The evolution of the adaptive threshold (right side of eq. 3.2) and of the image discontinuity (eq. 3.1) in a video portion of 1000 frames. In frames around 76400, the discontinuity takes high values which are rejected by the adaptive threshold, which takes a higher pick. These frames correspond actually to a progressive smoothed transition (dissolve). On the contrary, in frames around 76600, some smaller discontinuity values are not rejected, corresponding to true hard cuts. The use of a constant and acceptive (lower than the value of the false pick, around 80) threshold in this setting would produce a false alarm. An over-rejective (greater than 80) threshold, on the other hand, would result in 4 misses.

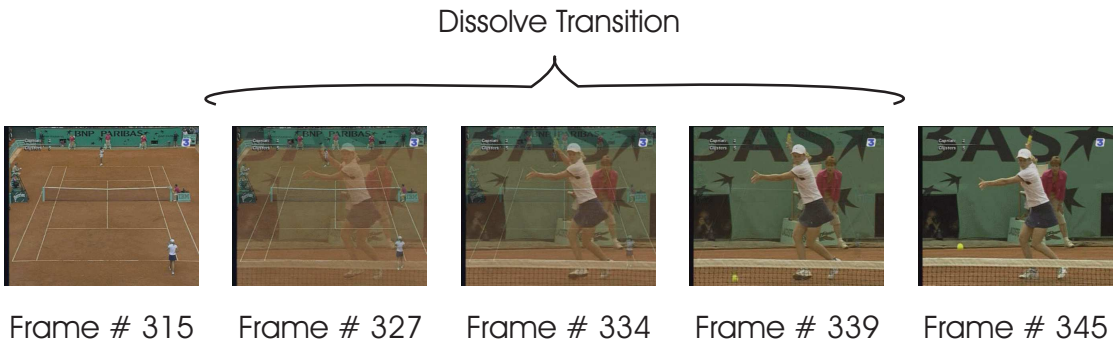


Figure 3.5
Example of a dissolve transition. The dissolve lasts for a number of frames where the first shot progressively disappears, while the second one fades in.

Of the first attempts to detect dissolves, Zhang *et al.* [118] proposed the method of the twin comparison to exploit the fact that, during a dissolve, the visual discontinuity takes some consecutive moderate values. To detect this behavior, they used multiple thresholds, while motion characteristics should be modeled and detected for false alarms removal. Other approaches include the modeling of the evolution of some image features during a dissolve [101]. The intensity variance, for instance, is expected to yield some local parabolic-like minima at the frames of a dissolve, as they are essentially a linear combination of two images. Nevertheless, complicated and error-prone image features have to be used to detect this shape, like temporal derivatives of the variance.

Proposed Method

A novel dissolve detection scheme is proposed in this study. It is based on the coupling of two key facts that are expected to happen in a dissolve:

- As it was noticed above, during a dissolve a large cumulative histogram distance is expected, as a result of a progressive transition between two shots.
- During a dissolve, the pixel-wise distances between consecutive frames are expected to take small or moderate values due to the *spatiotemporal smoothness* that appears. Cases like strong motion fields normally will explode the values of pixel-wise distances.

One may object that both conditions can be violated in cases of global or local gradual changing of luminance. To handle such situations, the histogram distances should be computed after lighting compensation. Nevertheless, this effect was not noticed in the tennis sequences used and was not taken into consideration.

The algorithm for dissolve detection operates after hard cut detection has taken place. This does not raise any problem because the adaptive threshold is insensitive to progressive transitions (Fig. 3.4). Each video segment between hard cuts is futher analyzed as follows.

1. For each frame $i = 1 \dots S$ between two hard cuts, the following quantities are computed:

   - The cumulative distance of the current frame to the preceding K ones:

     $$M_c(i) = \frac{1}{K} \sum_{j=i-K}^{i} D(\frac{H_i + H_{i-1}}{2}, H_j) \tag{3.3}$$

     where D is as defined in eq. 3.1.

   - Given the histogram $H^P_{i,i-1}$ of the absolute pixel-wise differences between the frames i and $i-1$, the metric:

     $$M_p(i) = \sum_{k=\tau+1}^{M} \Big(H^P_{i,i-1}[k] - \tau \Big)^2 \tag{3.4}$$

     will provide a measurement of the spatiotemporal smoothness between the two frames. M is the number of bins of $H^P_{i,i-1}$ and is equal to 256, as in hard cut detection. The threshold τ rejects small and noisy differences. In was manually set to 70. The rationale for this metric is simple: during a dissolve, most edges are faded out, resulting to small $M_p(i)$. Image motion, on the contrary, will displace (some of) the edges of the image and will result in high $M_p(i)$.

   - Immediately after a dissolve there are a number of frozen frames where $M_c(i)$ still takes high values and $M_p(i)$ small ones. In order thus to signify in a better way the end of the dissolve, the histogram distance is also taken into account:

     $$M_h(i) = D(H_i, H_{i-1}) \tag{3.5}$$

   - Finally, some oversmoothed frames may confuse $M_p(i)$. Generally, the absence of texture in the frames makes unreliable the detection of dissolves. To account for such situations, the frame i is passed through the simple edge detection filter $[-1 \;\; 1]$ in both columns and rows, and then the result is summed up to a new metric $M_e(i)$.

2. The quantities $M_c(i)$, $M_p(i)$, $M_h(i)$ are smoothed over time. An example of the temporal evolution of these quantities is given in Fig. 3.6. Regions where the

activity is far distant than the expected one are filtered out by using some soft approximate thresholds. For the remaining regions, the three above quantities along with $M_e(i)$ are averaged in order to form candidates for further examination.

3. A multilayer Perceptron[2], which is trained on positive and negative examples of candidates, gives the final answer if a candidate region is a dissolve or not. Both positive and negative examples were collected from the training sequences. Post-processing involves the rejection of very small regions (less than 4 frames).

During early experimentation, it was found that game statistics that appear and disappear progressively at the bottom half of the images introduce a lot of false alarms. To reject them, the quantities $M_c(i)$, $M_p(i)$, $M_h(i)$ were calculated at the top and bottom halves of the image frames separately. Then, in step 2, both halves must follow the same statistics for not being filtered out. Step 3 remains unchanged. At the expense of an increased false negative risk, the vast majority of the false alarms due to the statistics were rejected.

As a final result, 1,196 true dissolves were detected in both training and test sequences. The algorithm also yielded 3 false alarms and 10 losses in the training and 18/22, respectively, in the test sequences. This evaluation does not consider dissolves that occasionally appear inside commercials, as a lot of them are special effects that are hard to label as actually dissolves or not. Furthermore, the detection of these dissolves does not affect the performance of the models of the next chapters.

3.4 Visual Similarity

The most interesting class of shots in tennis videos is the ones that depict a global view of the court (Fig. 3.7). According to the production rules, when the actual game takes place, then the camera view is switched to the court view so that we are able to watch the game action in a convenient way. Court views are thus important landmarks in a tennis video. Fortunately, their automatic detection is extremely facilitated by the fact that they are dominated by the court color. However, a global color model for every tennis broadcast cannot be built as it significantly differs from tournament to tournament. A unique for every individual broadcast color model is thus needed.

In this study, the framework of [48, 47] was followed to this end, with some minor implementation optimizations regarding the features and the metrics used. A reference court view frame is automatically extracted from every video that is optimal in some statistical sense. Each video shot is then compared to this reference court view with standard histogram-based distances to yield a similarity measurement.

[2]See section 6.1.1.

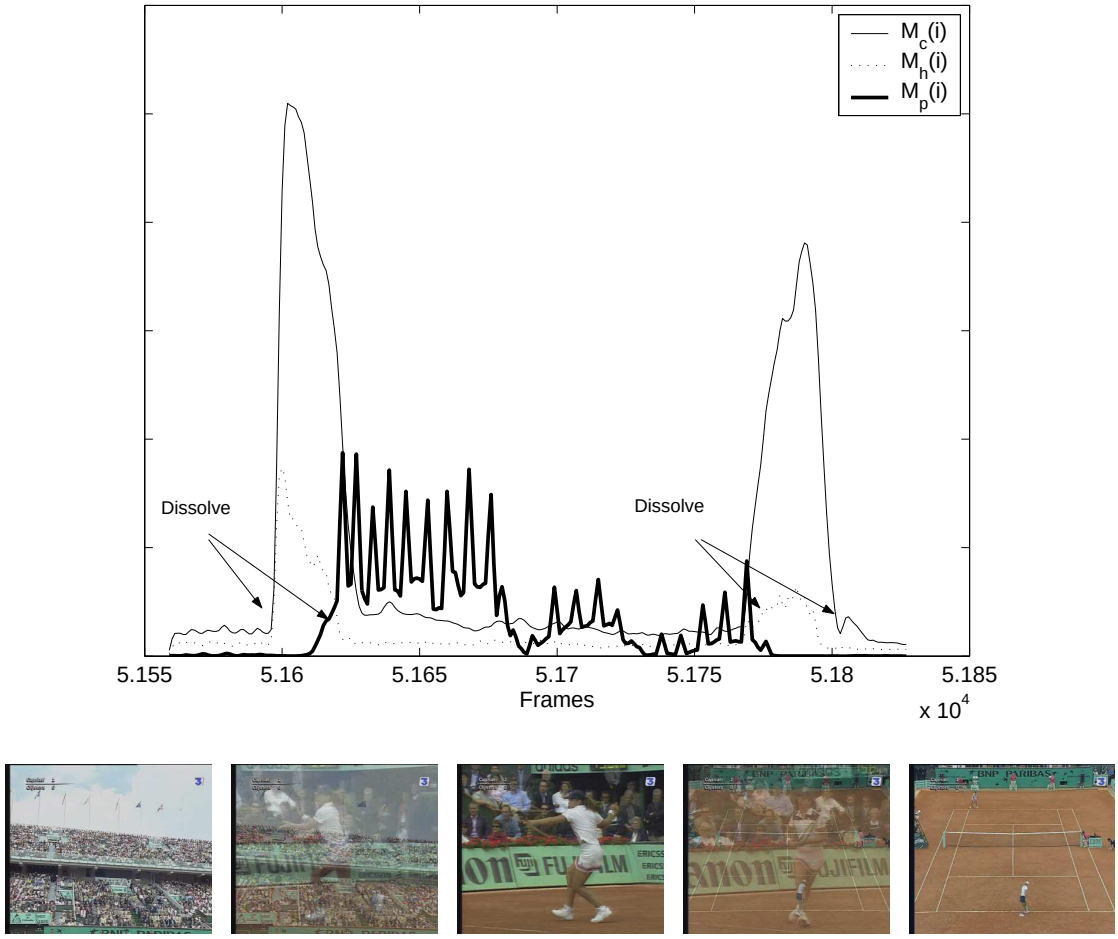


Figure 3.6

Example of the temporal evolution of the metrics $M_c(i)$, $M_p(i)$, $M_h(i)$. Two dissolves that surround a replay as seen in the images are detected. $M_c(i)$ takes high values during the dissolves, which are coupled with small values of $M_p(i)$. $M_h(i)$ decreases immediately after the dissolve. Finally, it is noticable that $M_p(i)$ follows an "up and down" curve inside the rediffusion. This is explained by the transformed and slowed-down rates that image frames are displayed in rediffusions. The strong motion fields (zoom that follows the motion of the player) at the beginning of the rediffusion resulted in high $M_p(i)$ values.

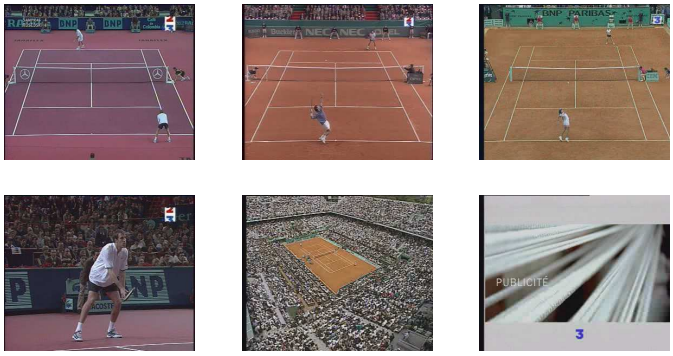


Figure 3.7

Some typical court (first row) and non court (second row) views. Court views are mainly characterized by the domination of the court view color, which is however different for every broadcast. Typical non court views one can see in a tennis video include close ups, crowd views, or commercials. Examples of them are given in the second row.

Given the segmentation of the video by the hard cuts and the dissolves, the middle frame of each shot is arbitrarily selected as the key frame of the shot, i.e., the frame that represents the best the visual content of the entire shot. This choice is sound regarding the court view shots as their visual content changes only slightly during the exchange. For the non-court view shots, the middle frame may not be the most representative one, but still the interest in these shots is to be characterized as non-court views. To this end, every frame of the shot could be used, as long as all of them are visually distant to the court view.

Dominant Colors Extraction and Prefiltering

Before proceeding to the actual algorithm for reference court view detection, extraction of the dominant colors of each key frame is performed. The majority of the non-court view frames are then easily filtered out as they are not dominated by a single color, as is the case of the court view frames. Dominant color extraction was performed by clustering to a small number of clusters the distributions of the pixel intensities. A straightforward implementation of the K-means algorithm was used as clustering technique, with uniform initial distributions and random presentation of the patterns. The number of the wanted clusters was set to 4 after experimentation. Clustering was performed in the LUV colorspace, after transformation[3] of the pixel intensities from the

[3]The interested reader is referred to
http://staff.science.uva.nl/ horus/dox/horus1.1/refcpp/html/HxColConvert_8h.html
and to
http://www.neuro.sfc.keio.ac.jp/ aly/polygon/info/color-space-faq.html
for more details on color conversion.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412452 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

$$L = \begin{cases} 116\left(\frac{Y}{Y_n}\right)^{1/3} - 16 & \text{if } \left(\frac{Y}{Y_n}\right)^{1/3} > 0.008856 \\ 903.3\left(\frac{Y}{Y_n}\right)^{1/3} & \text{if } \left(\frac{Y}{Y_n}\right)^{1/3} \leq 0.008856 \end{cases}$$

$$\begin{array}{ll} u' = \frac{4X}{X+15Y+3Z} & u'_n = \frac{4X_n}{X_n+15Y_n+3Z_n} \\ v' = \frac{9Y}{X+15Y+3Z} & v'_n = \frac{9Y_n}{X_n+15Y_n+3Z_n} \end{array}$$

$$U = 13L(u' - u'_n)$$

$$V = 13L(v' - v'_n)$$

Table 3.1

Color conversion from RGB to LUV colorspaces. The $(R\ G\ B)$ triplet is firstly converted to the XYZ colorspace to finally provide the $(L\ U\ V)$ values. $(X_n\ Y_n\ Z_n)$ is a reference white color, taken simply by converting the RGB white color in XYZ.

Figure 3.8

Result of the dominant color extraction for two images. The court in the second image contains in fact two colors, not very distant to human perception. Clustering in the RGB colorspace yielded in a lot of cases two dominant colors for the court. In the LUV colorspace, court is characterized by a single dominant color, as expected.

RGB colorspace, as given in Table 3.1. The LUV colorspace approximates human perception and is considered to be optimal for color-based comparisons [85]. Indeed, the quality of the clusters was significantly improved when switching from RGB to LUV colorspaces and also a much more stable behavior of the K-means clustering was noticed. Some examples of the dominant colors extraction are given in Fig. 3.8.

Finally, all the key frames whose dominant color occupies less the 70% of the pixels were a-priori labeled as not being a reference court view and were excluded from further analysis. Around 60% to 80% of the key frames were thus discarded, depending on the video.

Automatic Extraction of a Reference Court View

The reference court view is defined as the key frame image K_{ref} that is the nearest possible to the mass center of the court view key frames, in a space defined by the image distance $d(K_i, K_j)$ between two key frames K_i and K_j:

$$K_{\text{ref}} = \arg\min_{j \in I} \sum_{i \in I} d^2(K_i, K_j) \tag{3.6}$$

where I denotes the set of the (true) court view key frames. This problem could be solved by standard least-squares optimization in I. However, this set is not known a-priori as still, after the pre-rejection stage described above, a number of the remaining key frames do not correspond to court views. They can be considered as outliers, which can be discarded by a least-squares technique with outliers rejection capabilities. To this end, the *Least Median of Squares* method [84] was used.

The method proceeds as follows:

1. Random selection of p key frames $K_j \in \Omega$, where Ω is the set of all candidate key frames, including outliers.

2. Selection of the key frame K_m that satisfies:

$$K_m = \arg\min_{j \in [1 \ldots p]} \operatorname{MEDIAN}_{i \in \Omega} d^2(K_i, K_j) \tag{3.7}$$

   The use of the median operator, instead of summation, allows for outlier rejection.

3. Selection of the key frames that satisfy:

$$d^2(K_j, K_m) < \operatorname{MEDIAN}_{i \in \Omega} d^2(K_i, K_m) \tag{3.8}$$

   where the right part is simply the minimum median found in eq. 3.7. Supposing that the key frames K_j form a set I', then it is expected that $I' \subseteq I$, i.e., I' does not contain any outliers. If I' is empty, then simply $K_{\text{ref}} = K_m$.

4. K_{ref} is provided by a final optimization in the set I':

$$K_{\text{ref}} = \arg\min_{j \in I'} \operatorname{MEDIAN}_{i \in I'} d^2(K_i, K_j) \tag{3.9}$$

The pre-selection of candidates in step 1 allows for a speed-up of the computation. Given that p is sufficiently large while the percentage of outliers is small, the probability of selecting an outlier as K_m is close to zero [84]. After experimentation, p was set to 30, as the safest and smallest value allowed. Note also that, whatever the initial selection in step 1, K_{ref} will take its unique optimal value in step 4.

The distance $d(K_i, K_j)$ was calculated as a weighted sum of edge and color histogram-based distances:

$$d^2(K_i, K_j) = 0.9D(H_i^e, H_j^e) + 0.1D(H_i^c, H_j^c) \tag{3.10}$$

where H_i^c is the histogram of the color distibution of the frame i as computed in the LUV space, and H_i^e is the one for the distribution of the edges. Edge features were introduced as they were found to be more stable through time. In some videos, small luminance variations were noticed due to the large extend in time (each video is at least 2 hours). Furthermore, in the Roland Garros match that was held in open air, weather conditions greatly affected the court luminance at some points.

Computation of the Visual Similarity

The computation of the visual similarity D_{VS} of every key frame K_j of a video to its reference court view K_{ref} is given simply by:

$$D_{\text{VS}}(K_j) = d^2(K_j, K_{\text{ref}}) \tag{3.11}$$

where $d^2(K_j, K_{\text{ref}})$ is as defined in eq. 3.10. It was noticed that the range of values of D_{VS} is generally different for every match, influenced by the individual video characteristics. To compensate for this and yield homogenized statistics for every video, the similiraty measurements were passed through whitening:

$$D'_{\text{VS}}(K_j) = \frac{D_{\text{VS}}(K_j) - \mu}{\sigma} \tag{3.12}$$

where μ and σ are the mean and standard deviation of D_{VS}, computed in every video individually.

To provide an experimental analysis, firstly the court views of all videos were manually labeled. The training videos contain 901 in total and the test ones 1194. The histogram of the values of $D'_{\text{VS}}(K_j)$ for the two classes, court views or not, is given in Fig. 3.9. We notice a good behavior as, in their vast majority, court views yield a visual similarity of around -1.5, while most of the non-court views correspond to values of around 0. Selecting a threshold of $T_{\text{VS}} = -0.95$, for instance, the two classes are well separated. Precision and recall rates on the detection of court views are defined as:

$$\begin{aligned} \text{precision} &= 100\frac{\#\text{correct}}{\#\text{correct} + \#\text{false}} \\ \text{recall} &= 100\frac{\#\text{correct}}{\#\text{correct} + \#\text{lost}} \end{aligned} \tag{3.13}$$

where #correct is the number of true positives, #false that of false positives and #lost that of false negatives. The above value of T_{VS} resulted in 94.03% precision and 96.12% recall rates on the training sequences, while in the test ones 93.46% and 98.07%, respectively.

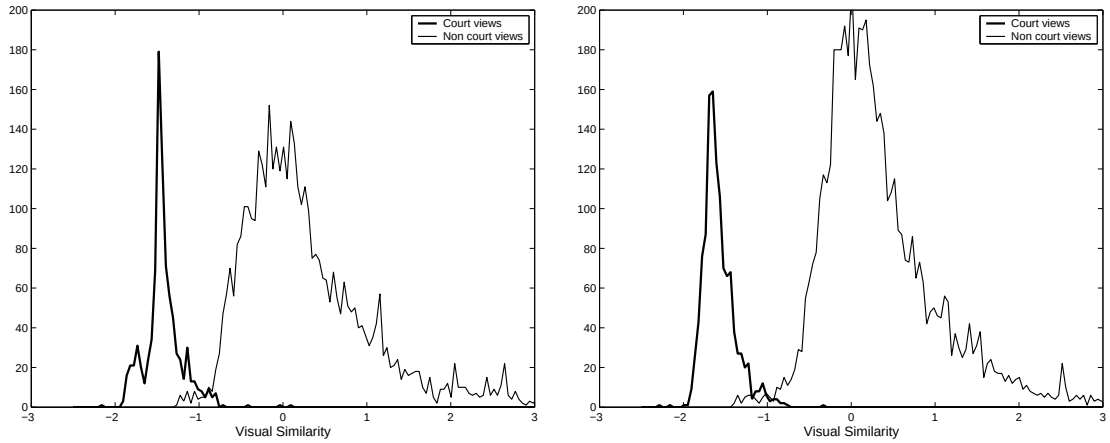


Figure 3.9
The histograms of the responses for the two classes in (a) test sequences and (b) training ones. In both cases, the two classes are nicely separated.

3.5 Auditory Features

In tennis video, there is the need to track the presence of the *ball hits* and *applause* sound classes. Ball hits are recorded during exchanges or serves and are expected to support the visual features at the discrimination between court views of actual game action and idle court views. As applause usually occurs after an exchange, its presence can be used to spot such events. Finally, the *music* sound class can convey useful information as it occurs on commercials. On the contrary, the sound class of speech that dominates the soundtrack cannot be informative and its presence was ignored. Indeed, the superimposed speech of the commentator may occur invariantly at every time instant on the video, including commercials.

Outcomes of the automatic detection of these sound classes were kindly provided by the research group METISS[4] of IRISA. Methodology is described in detail in [11, 10]. A summary is provided in this section.

Mel Frequency Cepstral Coefficients (MFCCs) are classically extracted from the soundtrack using a sliding window. A feature vector that represents the signal is thus formed, along with energy coefficients and first and second order derivatives. The process of the detection is separated into two distinct steps: soundtrack segmentation into acoustically homogeneous segments and segment classification. At the first step, a rough segmentation with a growing window was further refined with the Bayesian Information

[4]http://www.irisa.fr/metiss/

Criterion (BIC) [100]. It expresses, for a given segment y, a compromise between the produced log-likelihood of a model and its number of parameters $|\Lambda|$ and segment length T:

$$\mathrm{BIC}(\Lambda) = \ln(p(y|\Lambda)) - \frac{1}{2}\lambda|\Lambda| \ln T \tag{3.14}$$

where λ is a weighting factor, experimentally set. For every pair of adjacent segments, it is decided if it is more optimal in terms of BIC to fuse the adjacent segments or not. Let us suppose that the two segments are modeled by two Gaussian distributions Λ_1 and Λ_2 respectively and the fused segment by Λ_0. If then $\mathrm{BIC}(\Lambda_0) > \mathrm{BIC}(\Lambda_1) + \mathrm{BIC}(\Lambda_2)$, the two segments are fused.

The detection of each key sound class is carried out independently in each segment. Firstly, note that when two (or more) sound events C_i and C_j occur simultaneously in a segment y, they are considered as independent:

$$p(C_i, C_j|y) = p(C_i|y)p(C_j|y) \tag{3.15}$$

A binary hypothesis test is performed in every segment and for every class C_i to verify its presence in the segment. It is based on the MAP criterion which dictates that the class C_i (presence) or its anti-class $\overline{C}_i$ (absence) should be chosen according to the ratio:

$$\frac{1}{T}\sum_{t=1}^{T} \ln \frac{p(C_i|y_t)}{p(\overline{C}_i|y_t)} = \frac{1}{T}\sum_{t=1}^{T} \ln \frac{p(C_i)p(y_t|C_i)}{p(\overline{C}_i)p(y_t|\overline{C}_i)} \tag{3.16}$$

When it is greater than 1, class C_i is chosen. The distributions $p(y_t|C_i)$ and $p(y_t|\overline{C}_i)$ are modeled by Gaussian mixture models. Their parameters, along with the class priors $P(C_i)$ and $P(\overline{C}_i)$ were estimated on manually annotated data. The use of class priors in [10] yielded a performance improvement compared to uniform priors [11]. This two-step approach also slightly outperformed a Viterbi-based system where segmentation and classification are jointly carried out.

3.6 Relative Literature on Tennis Video Processing

To complete the picture of what has been accomplished in terms of automatic processing and salient feature extraction from tennis video, a review of the relative studies that appeared, to the author's knowledge, in the video indexing literature is given here. While none of these studies attempts to provide a dense structure analysis, it is usefull to see the state of the art for the feature extraction stage itself. For this reason, the review of these studies was reserved in this chapter instead of presenting them in the relative literature of chapter 2.

The first, probably, study on tennis games appeared in [98]. The authors attempt to extract high-level information from tennis clips fully exploiting domain knowledge.

Firstly, tennis court clips are detected from raw footage using the tennis court color and some heuristics. Prototype court colors were manually hard coded by the authors. Then, using projective geometry and line detection algorithms, they reconstructed the tennis court. In this new geometry space, player tracking is carried out using some standard frame-by-frame matching technique and heuristics. Finally, using the relative positions and hand-crafted rules, they annotate the video clips with labels like "baseline rallies" etc. No quantitave experimental analysis is provided.

In [119] an automatic unimodal adaptive procedure was proposed to detect serve shots in tennis video based on domain-specific rules. The video is first segmented into shots by hard cut detection. Color filtering is firstly applied to each key frame in order to detect court views. This is performed by using a training set of color histograms of serves where K-means is performed to form a number of histogram models. It is assumed that every new (not seen) color histogram should be close enough to any of these models. The actual color filtering proceeds by selecting the first L frames of the test video, which are matched with the models. The frames of the test video of the winning model are then used to build a new model, which is specific to the test video in hand. The color filtering is then followed by some domain-specific rules for further processing, i.e., false alarms removal. The key frames are segmented into homogeneous regions with respect to color and motion. Frames that do not contain a large enough region (the court view) are discarded. These regions will hopefully provide the players' positions. If a player is not detected in the lower half part of the court, this frame is considered again as false alarm. Finally, using the Hough transform, two vertical and horizontal lines should be detected. The system was tested in a 1-hour tennis video. As it operates on compressed raw video data, real-time performance is achieved.

In [22] tennis exchanges between the two players were detected by a combination of image and sound (ball hits) information as none of them alone can sufficiently characterize tennis exchanges. Image features used were based on edge information and the Hough transform. Ball hits in the soundtrack were detected by a PCA-based distance. Having segmented the video into shots, an average likelihood over all the image and sound frames were computed to finally form the classical product of probabilities as the resulting likelihood. The system was tested on a small corpus containing 18 shots and 5 exchanges.

In [21], a method for tennis video abstraction is proposed on a large corpus of 7 tennis programs. After segmentation of the video track into shots, motion-based features are extracted. A method for player detection is also proposed, where residual motion of small objects (the players) is detected and then vertically projected to form horizontal cumulative histograms. Using K-means clustering, the video track is thus divided to non-interesting and interesting segments, which are further analyzed with audio information. The presence of ball hits and applause sound classes is tracked in

the soundtrack. According to their duration percentages on the preselected segments, tennis abstraction classes are formed, such as aces, rallies, etc.

Some studies attempt to extract from tennis clips high-level information like game tactics. In [105] tennis exchanges are classified into two classes, namely "Net Game" and "Baseline Game". The authors extract motion features that characterize the players' trajectories, after projecting them to the true tennis court space. The evolution of these features over time is then used as input to HMMs that, finally, provide the wanted class labels.

In [76] the authors attempt to recognize different types of tennis strokes (servings) using HMMs. Given a tennis court view (provided rather manually), the authors detect the player at the lower part of the tennis court using color-based morphological operations. They also construct a court 3D model to aid the player segmentation. The second step is to extract a number of 16 points around the player. The temporal evolution of these features through 45 frames will be classified by HMMs as a given type of stroke. The authors used 120 clips for training and 240 clips for testing the system. They provide some experimentation as to which feature points are the most informative ones.

CHAPTER 4

Structure Parsing and Audiovisual Integration

This chapter details video structure parsing and audiovisual integration with Hidden Markov Models and Segment Models on top of the features presented in the previous chapter. First, basic concepts of these two modeling alternatives are reviewed. Next, the tennis scenes are defined. With the HMM framework, each scene is modeled by sub-model HMMs, which are interconnected to provide a unique ergodic HMM that models the video as succession of shots. The state-synchronous concatenative fusion is used for audiovisual integration. With the SM framework, each scene is considered as a segment and an ergodic SM is used to model the video as succession of scenes. Various possibilities of audiovisual integration are presented. Finally, a comparative exprimental analysis is provided.

4.1 Basic Concepts of Stochastic Temporal Models

This section summarizes notions and basic definitions for the HMMs and SMs. For a more detailed analysis, see the classical tutorial of Rabiner [81] on HMMs and Ostendorf *et al.* [72] for SMs.

4.1.1 Hidden Markov Models

Definitions

HMMs model a stochastic process that evolves in discrete time steps. At each time instant t, the process is assumed to being in a state q_t, chosen between a finite set of N states $S_1 S_2 \dots S_N$. The process evolves by switching between the states with a given probability. In a first order Markov chain model, this probability is considered to be constant through time and dependent to only the previous state q_{t-1}:

$$P(q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \dots) = P(q_t = S_j | q_{t-1} = S_i) \tag{4.1}$$

The term $P(q_t = S_j | q_{t-1} = S_i)$, often abbreviated as a_{ij}, denotes the probability from switching from state S_i to state S_j. The model is thus defined by the row-stochastic state transition matrix $A = \{\mathrm{a}_{ij}\}$, $1 \leq i, j \leq N$. In addition, the initial state distribution $\pi = \{\pi_i\}$, $1 \leq i \leq N$, defines the probabilities according to which the states are drawn at the first time instant, i.e.:

$$\pi_i = P(q_1 = S_i) \tag{4.2}$$

In this simple Markov chain model, the states are *observable*, i.e., the state sequence itself is the result of the stochastic process. HMMs generalize this model by introducing an observation sequence $O_{1:T} = O_1 O_2 \dots O_T$, which is generated according to a probabilistic function from the state sequence. The observations are thus the actual output of the stochastic process and usually correspond to physical events, a set of features, or more generally, to something measurable. The states of HMMs are not measurable and have to be inferred, hence the term *hidden* states. They correspond to the underlying dynamics that guide the stochastic process. In the classical example of speech production, the hidden states correspond to the phonemes uttered, while the observations to the produced auditory signal.

Given a set of M observation symbols $v_1 v_2 \dots v_M$, the probability of emitting a symbol $O_t = v_k$ while in hidden state j is given as:

$$\mathrm{b}_j(k) = P(O_t = v_k | q_t = S_j) \qquad 1 \leq j \leq N, 1 \leq k \leq M \tag{4.3}$$

As with transition probabilities, $\mathrm{b}_j(k)$ are independent of time t. Note also that as $\mathrm{b}_j(k)$ is conditioned only to the hidden state label, a conditional independence of observations is assumed. HMMs define thus also the row-stochastic $N \times M$ matrix $B = \{\mathrm{b}_j(k)\}$. To handle cases of non-discrete observations, Continuous Density HMMs (CDHMMs) use Gaussian Mixture Models to model observation densities. They provide an observation likelihood instead of a probability. Overall, the complete parameter set of an HMM is defined by the triplet:

$$\lambda = (A, B, \pi) \tag{4.4}$$

As classically described in [81], three problems related to HMMs have to be solved:

- *Inference.* How one can efficiently compute the probability $P(O_{1:T}|\lambda)$ that the HMM λ has produced the sequence $O_{1:T}$?
- *Decoding.* Given an observation sequence $O_{1:T}$ and an HMM, which is the corresponding most likely hidden state sequence $Q_{1:T} = q_0 q_1 \dots q_T$? When the hidden states have a physical meaning, this sequence provides an "explanation" of $O_{1:T}$.
- *Parameter Estimation.* Given one or multiple observation sequences $O_{1:T}$, how one can estimate the model parameters $\lambda = (A, B, \pi)$ in order to maximize $P(O_{1:T}|\lambda)$? The Baum-Welch algorithm [81] is usually employed to this end. This problem will not be analyzed further here.

Inference

To solve the first problem, let us consider a fixed hidden state sequence $Q_{1:T}$. The probability of the observations given $Q_{1:T}$ and the HMM λ is factored as:

$$P(O_{1:T}|Q_{1:T}, \lambda) = \prod_{t=1}^{T} \mathrm{b}_{q_t}(O_t) \tag{4.5}$$

due to the conditional independence of the observations. In addition, the probability for drawing $Q_{1:T}$ is simply given by:

$$P(Q_{1:T}|\lambda) = \pi_{q_1} \prod_{t=2}^{T} \mathrm{a}_{q_{t-1}q_t} \tag{4.6}$$

while the joint probability of generating $P(O_{1:T}|Q_{1:T}, \lambda)$ and $P(Q_{1:T}|\lambda)$ is:

$$P(O_{1:T}, Q_{1:T}|\lambda) = P(O_{1:T}|Q_{1:T}, \lambda) P(Q_{1:T}|\lambda) \tag{4.7}$$

Finally, the wanted probability $P(O_{1:T}|\lambda)$ is given as a sum of the joint probabilities over all possible hidden state sequences:

$$P(O_{1:T}|\lambda) = \sum_{Q_{1:T}} P(O_{1:T}|Q_{1:T}, \lambda) P(Q_{1:T}|\lambda) \tag{4.8}$$

$$= \sum_{Q_{1:T}} \pi_{q_1} \mathrm{b}_{q_1}(O_1) \prod_{t=2}^{T} \mathrm{a}_{q_{t-1}q_t} \mathrm{b}_{q_t}(O_t) \tag{4.9}$$

As there exist N^T possible hidden state sequences and, for each of them, $2T$ computations are required, the straightforward computation of eq. 4.8 is of order $O(2TN^T)$, which is clearly prohibitive. For this reason, the *forward-backward procedure* [81] can be used instead, whose computation cost is $O(TN^2)$.

Decoding

Decoding involves finding the most likely *single* hidden state sequence $Q^*_{1:T}$. In terms of the Maximum A-posteriori Criterion (MAP), this translates to:

$$\begin{aligned} Q^*_{1:T} &= \arg\max_{Q_{1:T}} P(Q_{1:T}|O_{1:T}) && (4.10)\\ &= \arg\max_{Q_{1:T}} \frac{P(O_{1:T}|Q_{1:T})P(Q_{1:T})}{P(O_{1:T})} && (4.11)\\ &= \arg\max_{Q_{1:T}} P(O_{1:T}|Q_{1:T})P(Q_{1:T}) && (4.12) \end{aligned}$$

The denominator $P(O_{1:T})$ of eq. 4.11 is constant for every hidden state sequence and thus discarded. In pattern recognition terminology and intuitively speaking, this problem is equivalent to classification of a series of T features $O_{1:T}$ to a set of N classes corresponding to the hidden state labels. When considering only the term $P(O_{1:T}|Q_{1:T})$, then eq. 4.12 reduces to a series of MAP classifications of the patterns, *individually*. The addition of the term $P(Q_{1:T})$ adds the temporal dimension to the problem and dictates that the order of the class labels obtained should satisfy some optimality with respect to the state transition matrix A of the HMM. This term can help in cases of noisy features by adding built-in robustness or of ambiguity between the classes. When for example, in speech recognition, the model provides two possible transcriptions of an utterance of more or less the same probability, then the one that matches an actual word is selected.

Decoding is solved via the *Viterbi algorithm*, a general dynamic programming technique that finds out the most likely hidden state sequence (or path) in a trellis of nodes. The algorithm defines the probability score $\delta_t(i)$ of the best path ending at time t in hidden state i and proceeds as follows:

1. *Initialization.*
$$\delta_1(i) = \pi_i \mathrm{b}_i(O_1) \qquad 1 \le i \le N \tag{4.13}$$

2. *Recursion* for $1 \le i \le N$ and $2 \le t \le T$, where T is the sequence length.
$$\begin{aligned} \delta_t(i) &= \max_j \delta_{t-1}(j)\mathrm{a}_{ji}\mathrm{b}_i(O_t) && (4.14)\\ \psi_t(i) &= \arg\max_j \delta_{t-1}(j)\mathrm{a}_{ji}\mathrm{b}_i(O_t) \qquad 1 \le j \le N && (4.15) \end{aligned}$$
The variable $\psi_t(i)$ (to be used in step 3) keeps track of the paths obtained.

3. *Backtracking.*
$$\begin{aligned} q^*_T &= \arg\max_j \delta_T(j) && (4.16)\\ q^*_t &= \psi_{t+1}(q^*_{t+1}) \qquad t = T-1, T-2 \ldots 1 && (4.17) \end{aligned}$$

The computational cost of the Viterbi algoritm is $O(TN^2)$.

4.1.2 Limitations of HMMs

In the speech recognition literature a number of the HMM modeling assumptions like the conditional independence of the obervations have been found too strong leading thus to poor modeling of the speech production process. Various alternatives to standard HMMs have been proposed in order to build more accurate models. As concisely summarized by Ostendorf *et al.* [72], the limitations and their possible solutions are:

- *Weak duration modeling.* In standard HMMs, the probability of remaining at the same hidden state S_i for d consecutive time instants (in other words, after making $d-1$ self-transitions) is:

$$p_i(d) = \underbrace{\mathrm{a}_{ii}\mathrm{a}_{ii}\ldots\mathrm{a}_{ii}}_{d-1 \text{ times}}(1-\mathrm{a}_{ii}) = \mathrm{a}_{ii}^{d-1}(1-\mathrm{a}_{ii}) \tag{4.18}$$

  where $(1-\mathrm{a}_{ii})$ is the probability of leaving S_i. This expression is equivalent to a geometric duration model, which is a modeling assumption that may not hold always. The Explicit State Duration HMMs [81] (or semi-Markov models) have been introduced as an alternative. In these models, self-transition probabilities a_{ii} are set to zero while an arbitrary distribution $p_i(d)$ is used in their place to model the state duration. Explicit duration models require however an extra computational cost during Viterbi decoding as all the possible segmentation hypotheses have to be evaluated.

- *Conditional independence of observations.* This assumption implies that there is no correlation between successive observations. This may not be true, especially when simulating continuous-time media like speech waveforms. Features from adjacent observations can be used for more effective observation distributions. In speech recognition, feature derivatives are widely used to capture the time-varying spectral dynamics of speech.

- *Frame-based observations.* Extending the skepticism on the above-mentioned assumption, one may be tempted to use segmental features (i.e., sequences of features) rather than frame-based observations. This greatly advances the freedom for modeling inter-segmental and local in nature correlations between them, leaving the intra-segmental ones to be modeled by the HMM itself. The use of segmental features in speech recognition can be beneficial also during the sampling stage, allowing for adapted sampling strategies [72]. Segmental HMMs have been proposed to this end.

Gathering all these modeling alternatives into a unified framework, Ostendorf *et al.* [72] defined Segment Models.

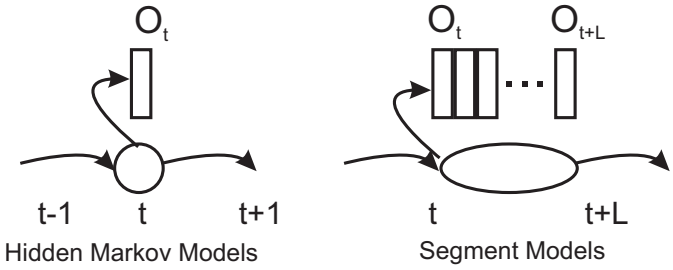


Figure 4.1
The generation of observations according to Hidden Markov Models (left) and to Segment Models (right). The hidden state in SMs generates L observation symbols, instead of one.

4.1.3 Segment Models

The fundamental difference between SMs and HMMs is that in SMs a hidden state is associated to a complete sequence of observations $O_{1:l}$, called *segment*, instead to a unique feature vector O_t. Therefore, each hidden state i in SMs defines firstly a duration model $p(l|i)$ that accounts for the segment length l. Like the hidden state labels, l is stochastic. Secondly, an emission probability distribution over a segment $p(O_{1:l}|l, i)$ is defined, conditioned on the segment length and the hidden state. A SM is thus defined by the parameter set:

$$\lambda = (A, B, D, \pi) \tag{4.19}$$

where D denotes the state duration distributions.

From a generative point of view, SMs can be seen as a Markovian process where a hidden state emits a sequence of observations whose length is governed by a duration model before transiting to another state. The difference between HMMs and SMs is illustrated in Fig. 4.1. On the left, we see what happens conceptually in the case of HMMs: at a given time instant the process is in a given state and generates one observation symbol and then transits to another state. On the right, we see how a sequence is generated according to Segment Models. At a given time instant, the stochastic process enters into a state and remains there according to a probability given by the state duration model. A sequence of observations is generated, instead of a single one, according to a distribution conditioned on the segment label. Then the process transits to a new state with a transition probability, as in HMMs, and so on until the complete sequence of observations is generated.

SMs provide a generalized framework that can take several HMM variants as a special case. Indeed, conventional HMMs is simply the special case of a SM whose duration model is fixed and equal to 1. An Explicit State Duration HMM, as another example,

is a SM where the conditional independence of observation inside a segment still holds:

$$p(O_{1:l}|l,i) = \prod_{k=1}^{l} b_i(O_k) \tag{4.20}$$

In [72], the EM-based estimation of the parameters for SMs is formulated under various segmental distributions assumptions and models specific to speech recognition. The derivation of these formulas is beyond the scope of this study.

Viterbi decoding for SMs, quite similar to that for Explicit State Duration HMMs, involves finding not only the most likely hidden state sequence, but also the most likely *segmentation*, or, in other words, the most likely duration of each segment:

$$(L^*_{1:N^*}, Q^*_{1:N^*}) = \arg \max_{L_{1:N}, Q_{1:N}} p(O_{1:T}|L_{1:N}, Q_{1:N}) p(L_{1:N}|Q_{1:N}) p(Q_{1:N}) \tag{4.21}$$

where N^* is the number of segments found, $L^*_{1:N^*}$ is the most likely segmentation, $Q^*_{1:N^*}$ is the most likely hidden state sequence, and T is the length of the observation sequence. The term $L_{1:N}$ denotes both a segmentation hypothesis of N segments and their respective hypothesized durations. It is expected generally that $N < T$ (N is fixed to T in the case of standard HMMs).

In a straightforward extension of the Viterbi decoding described in section 4.1.1, the term $\delta_t(i)$ denotes now the best segmentation and hidden state sequence ending at time t in state i and the modified algorithm proceeds as follows:

1. *Initialization.*

$$\delta_1(i) = \pi_i p(O_1|1,i) \qquad 1 \le i \le N \tag{4.22}$$

2. *Recursion* for $1 \le i \le N$ and $2 \le t \le T$, where T is the sequence length.

$$\delta_t(i) = \max_{j,k} \delta_{t-k}(j) p(O_{t-k+1:t}|k,i) p(k|i) \mathrm{a}_{ji} \tag{4.23}$$

$$(\psi_t(i), \psi^l_t(i)) = \arg \max_{j,k} \delta_{t-k}(j) p(O_{t-k+1:t}|k,i) p(k|i) \mathrm{a}_{ji} \tag{4.24}$$

$$1 \le j \le N, \quad 1 \le k \le L_{\max}$$

The added variable $\psi^l_t(i)$ keeps track of the best segmentation obtained. $L_{\max}$ puts a constraint on the maximum allowed length of the segment to keep the computation tractable.

3. *Backtracking.*

$$q^*_{N^*} = \arg \max_j \delta_T(j) \tag{4.25}$$

Set $t' = T$ and iterate until $t' > 0$

$$(q^*_n, l^*_n) = (\psi_{t'}(q^*_{n+1}), \psi^l_{t'}(q^*_{n+1})) \tag{4.26}$$

$$t' = t' - l^*_n \tag{4.27}$$

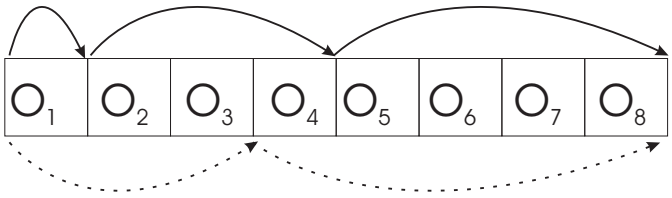


Figure 4.2

The effect of searching through multiple segmentation hypotheses for a sequence $O_{1:8}$. The probability scores of the two displayed paths are
$\delta_8(x) = p(O_1|1,x)p(1|x)p(O_{2:4}|3,x)p(3|x)p(O_{5:8}|4,x)p(4|x)$ and
$\delta_8(x) = p(O_{1:3}|3,x)p(3|x)p(O_{2:8}|5,x)p(5|x)$, respectively. For sake of simplicity, initial distributions and transition probabilities terms are omitted, while x is a generic state label.

An illustration of searching through different segmentation hypotheses is given in Fig. 4.2. Compared to standard Viterbi decoding, the version for SMs is clearly more computationally expensive due to the enhanced search through segmentations and hidden states of eq. 4.23. Supposing that the segmental scores $p(O_{1:l}|l,i)$ can be evaluated in $O(1)$ time, the cost is now of order $O(TN^2L_{\max})$, i.e., $L_{\max}$ times more costly compared to HMMs. This is a satisfying result as it does not explode with time. Of course, the assumption of the instant evaluation of the segmental scores does not hold generally, being in the best scenarios of order of $O(l)$. Still, a lot of segmental scorers can support *caching* of the outcomes during successive evaluations so that the overall cost approaches again $O(TN^2L_{\max})$.

4.2 Tennis Scenes and Problem Definition

4.2.1 Tennis Video Characteristics

Tennis video exhibits a certain type of characteristics as result of the tennis rules and the work of the producer before broadcasting it, as well. Some interesting tennis rules or commonalities that occur in the tennis court are[1]:

- When a player serves and during the exchange, absolute silence is required from the crowd. This is a strict tennis rule, guaranteed by the referee. The player cannot serve if there is no silence and an exchange is annulled in case of unexpected noises.
- After a missed serve, the player repeats the serve in a short time period.
- After an exchange, the crowd applauds. Of course, nothing can ensure this but it is true in the vast majority of the exchanges. False applauds, on the other hand, also appear some times after a missed serve.

[1] A complete description of the tennis rules are outside the scope of this study.

The production rules (or commonalities) in a tennis broadcast are:

- When game action occurs, then the camera is switched to the court view. It is extremely rare, although still possible, to present game action by a non-court view, like a side view. On the other hand, game idleness usually corresponds to non-court views.
- After a missed serve, a number of non-court views interfere before the repeated serve.
- Rediffusions are surrounded by dissolve transitions. Multiple rediffusions result in a "sandwich" effect of successions of dissolve transitions and rediffusions.
- Dead time (i.e., when the game is paused) corresponds to commercials or presentation of game statistics or interviews.

4.2.2 Scenes Definition

Based on the tennis video characteristics, four types of scenes are identified and serve as the basic semantic building blocks of the video. They are defined on top of the video segmentation into shots, i.e., each scene is in fact a collection of successive shots that share a unique semantic content. The scenes are defined as follows:

1. *First Missed Serve and Exchange.* This type of scene starts with a game action shot where a missed serve occurs. A number of non-game action shots follow until a normal exchange takes place. Finally, a number again of non-game action shots (optionally) appears, until a new scene begins. There is also the possibility of repetitive missed serves before the exchange.
2. *Exchange.* The scene starts with a game action of an exchange, followed by a number of filling-up non-game action shots, until a new scene starts.
3. *Rediffusion.* Rediffusion scenes start with a dissolve shot and then contain a number of successions of non-court views and dissolve shots. They finish with filling-up non-game action shots until a new scene begins.
4. *Break.* Breaks can be either commercials or dead time of long duration where an interview may take place or game statistics are displayed. Breaks start at the shot where the players leave the court. The commercials or interviews take place and, finally, filling-up non-game action shots may appear at the end.

The first two scenes correspond in practice to the same basic game event: the exchange. They are assigned different labels to make the problem more challenging. Note also that

the term "game action shot" is used, instead of court view. This is because some court views can be idle in terms of game action. It is up to the system to decide which court views actually contain game action. Regarding the break scenes, their exact starting point can be precisely defined by a human observer, which is able to figure out when the players leave the court and sit on the desks. In terms of the extracted audiovisual features, however, it is troublesome to imagine a feature (or feature combination) that is generally able to signal the start of a break. Nevertheless, their temporal trace is easily detected, as commercials are a long succession of short shots that may contain music while interviews and idle time are a collection of a few but extremely lengthy shots.

4.2.3 Problem Definition

Video structure parsing is defined as the problem of the classification of each shot as belonging to one of the four scenes defined above, plus the detection of the scene boundaries. Every shot boundary in the video should be examined as a potential scene boundary.

The frameworks of HMMs and SMs fit perfectly well to this problem, as its temporal aspect is evident by definition. For example, the decision if an exchange belongs to the first or to the second scene can be taken only after inspection of the recent past in order to detect the presence (absence) of missed serves. Given a video sequence and the model parameters, Viterbi decoding will provide the wanted semantic labels and the segmentation of the scenes.

Audiovisual integration can greatly benefit carrying out the task at hand. Typically, a human observer can solve the problem of labeling and segmenting the video into scenes by just watching the videotrack, even with a certain degree of difficulty. The inference however on top of the automatically extracted visual features can be troublesome and thus auditory features are introduced to boost the performance by providing further evidence.

4.2.4 Ground Truth

For the needs of evaluation, each shot of the video was manually labeled in order to provide the *ground truth*. Errors from the automatic hard cut and dissolve detection (though rare) were treated as follows: insertions or deletions of hard cuts do not alter the definition of the scenes; they were thus ignored. Inserted dissolves were treated as noisy segmentation/features and marked with their real scene label. On the contrary, the entire rediffusion was ignored if one of its two dissolves is missed. The reason for doing this is three-fold: firstly, in terms of audiovisual features, there is no way to recover this error of the low-level videotrack segmentation. Second, in some cases the producer himself omits to surround a rediffusion with two dissolves. Thirdly, they do not affect comparative results between different models.

Training Sequences		
	% of scenes	% of shots
1	30.48	34.04
2	44.11	29.59
3	19.70	14.20
4	5.70	22.17

Test Sequences		
	% of scenes	% of shots
1	36.98	42.23
2	37.39	24.00
3	20.22	15.07
4	5.41	18.70

Table 4.1
Ground truth statistics for both training and test sequences. The scene labels 1 to 4 are as defined in the text. The second column of each table gives the percentage of occurrence and the third one the percentage of shots occupied.

The training sequences contain 807 scenes in total and the test ones 979. The percentage of occurrence of each scene is given in table 4.1 along with the percentage of shots that each scene occupies. Especially for the distributions of shots, they are more or less evenly distributed among the scenes and thus no bias towards a specific scene is present (i.e., a classifier biased to prefer the majority class cannot achieve good performance).

4.3 Content Modeling with Hidden Markov Models

With the HMM framework, the observations O_t are modeled *at the shot level* and the feature vectors correspond to the extracted shot-based audiovisual measurements and descriptors. The visual content of the four scenes is analyzed firstly by four sub-model HMMs, which are finally interconnected to a unique HMM that models the complete video as a succession of shots. Audiovisual integration is then discussed and, finally, parameter estimation is detailed.

4.3.1 Visual Content Modeling

Given the definitions of the scenes in section 4.2.2, it is straightforward to determine the number of hidden states and their semantic meaning for the sub-model HMMs. The HMM for the 'First Missed Serve and Exchange' scene is depicted in Fig. 4.3(a). State 1 represents the court view of the missed serve. Some non-court view shots follow (state 2), until the player serves again. If this new serve is again missed, then a transition back to state 1 occurs. The transition to state 3 represents that a normal exchange takes place. Finally, state 4 represents the trailing non-court view shots after the exchange. The definition of the 'Exchange' scene sub-model, given in Fig. 4.3(b) is straightforward, being a part of the above-mentioned sub-model.

The two remaining scenes require 3 hidden states each, as shown in Fig. 4.3(b). Regarding rediffusion, state 8 represents both a rediffusion shot and the trailing non-

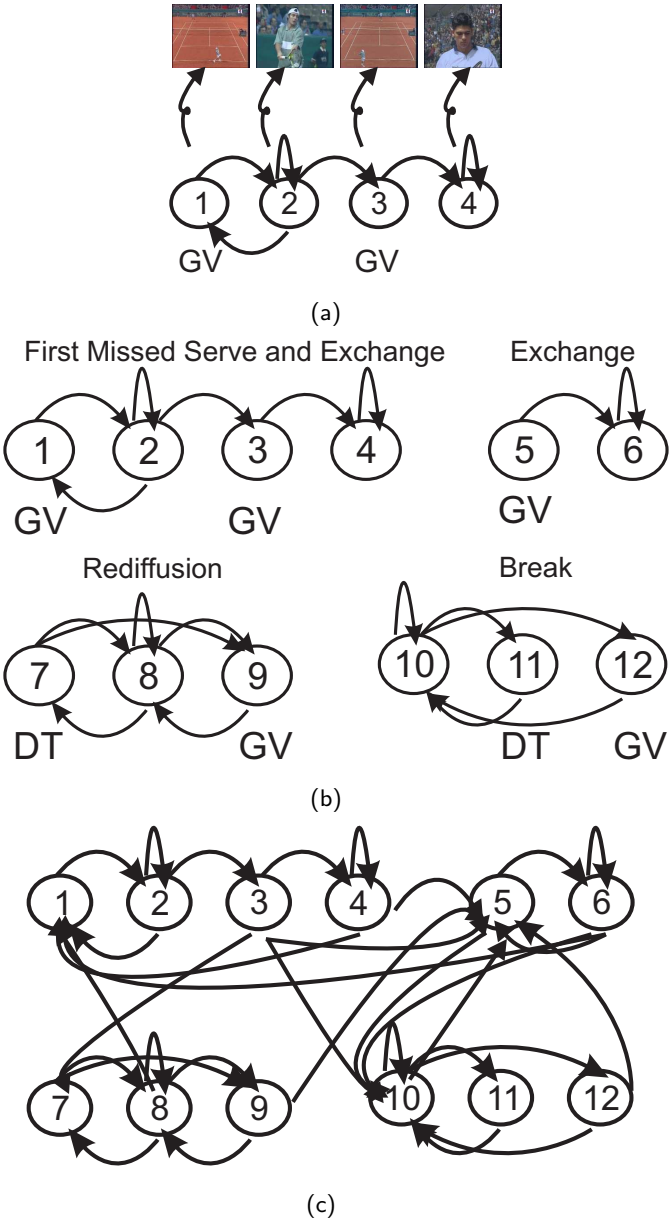


Figure 4.3

The sub-model HMM for the first missed serve and exchange scene (a), the four sub-model HMMs (b), and the HMM that models the complete video (c). "GV" stands for global view and "DT" for dissolve transition. The semantic meaning of each hidden state is easily deduced. Note that arrows represent actually major transitions and are given for illustrative purposes. The HMMs are in fact ergodic. The interconnection arrows in (c) are not exhaustive.

court views after the rediffusion. The distinction between the two cases after decoding is easily done given a hidden state sequence as rediffusion shots are always surrounded by the hidden state 7 (i.e., dissolve transitions). Hidden state 9 represents global views of non game action that may occur between the trailing non-court views, as does the hidden state 12 for the break scene. Such idle global views were found to be far more rare between the trailing non-court views of the first two scenes and thus they were not assigned a special hidden state. Finally, state 11 represents dissolve transitions corresponding to special effects that signal the start or the end of the commercials or other special effects inside them.

As illustrated in Fig. 4.3(c), the four sub-model HMMs are fused into a unique HMM that models the entire video content and whose hidden states set is simply the union of the respective sets of Fig. 4.3(b).

The feature set extracted from a shot includes the descriptors of the visual similarity, shot length and the absence/presence of a dissolve, as described in chapter 3. The shot-based feature vector is thus defined as follows:

$$O_t^{\mathrm{v}} = \begin{bmatrix} \text{visual similarity} \\ \text{shot length} \\ \text{dissolve shot} \end{bmatrix} = \begin{bmatrix} o_t^c \\ o_t^l \\ o_t^d \end{bmatrix} \tag{4.28}$$

where O_t^{v} denotes explicitly visual features. To make parameter estimation and subsequent computations easier, the features were discretized. More precisely, the values of o_t^c and o_t^l were homogeneously quantized into 10 bins each[2]. o_t^d is naturally discrete being a binary descriptor. Furthermore, features were considered to be independent so that the conditional observation probability is:

$$P(O_t^{\mathrm{v}}|S_j) = P(o_t^c|S_j)P(o_t^l|S_j)P(o_t^d|S_j) \tag{4.29}$$

for a given state S_j.

Viterbi decoding involves finding the hidden state label of each shot, given an observation sequence $O_{1:T}^{\mathrm{v}}$ and the model parameters. In light of eq. 4.29 and in the log domain, eq. 4.12 translates to:

$$Q_{1:T}^* = \arg\max_{Q_{1:T}} \{ \ln P(Q_{1:T}) + \sum_{r\in\{c,l,d\}} \ln P(o_{1:T}^r|Q_{1:T}) \} \tag{4.30}$$

As show in Fig. 4.4, decoding classifies each shot with one of the 12 class labels of Fig. 4.3(c). It is straightforward then to map each shot label to a scene label and find the scene boundaries in order to meet the problem definition of section 4.2.3.

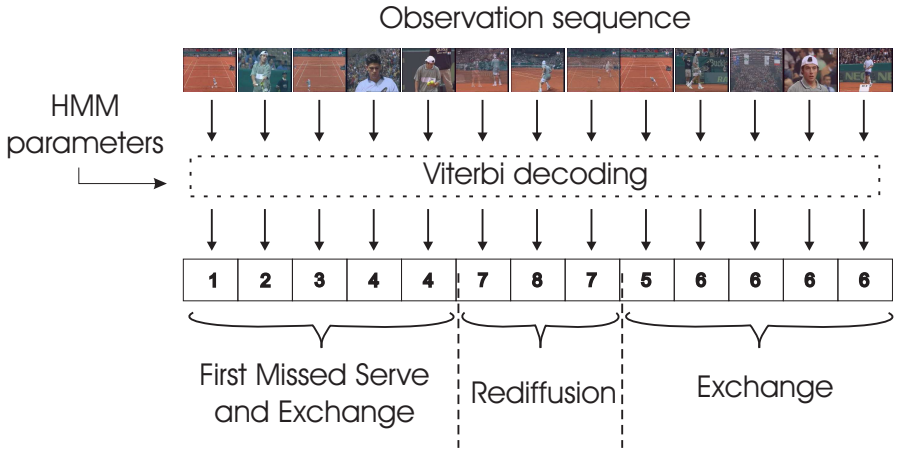


Figure 4.4

Viterbi-based classification of the shots and the final map to scene labels and scene boundary determination. HMM parameter estimation is discussed later in this chapter.

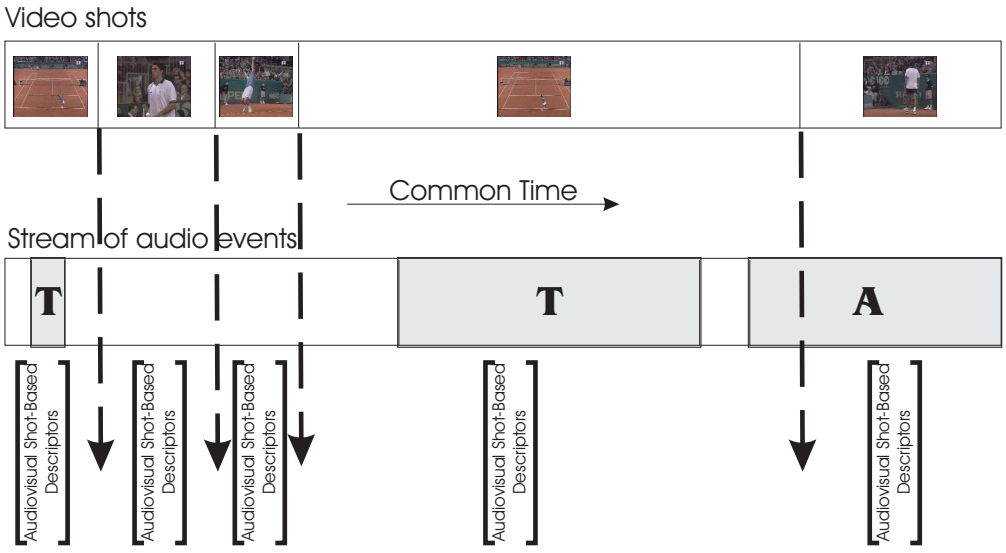


Figure 4.5

Audiovisual integration with HMMs. The stream of audio events has to be synchronized firstly to the visual one. It is segmented thus using the time stamps of the video shot boundaries. Inside each shot, a number of auditory binary descriptors are collected and concatenated to the visual feature vector. 'T' stands for tennis sound (ball hits) and 'A' for applause.

4.3.2 Audiovisual Integration

Besides the visual features, which are represented as a stream of visual shot-based descriptors, there are also features from the soundtrack that should be used. They come in the form of a stream of audio events, as described in section 3.5. The two streams are thus sampled at different (and variable) frame rates exhibiting a natural asynchrony. However, audiovisual integration with HMMs has no other option than the standard approach of state synchronous concatenative fusion (section 2.3.2): the two stream have to be artificially synchronized before fusion. The procedure is illustrated in Fig. 4.5. The stream of audio events is firstly segmented according to the shot boundaries of the video track. As a second and final step, three auditory binary descriptors are added to the vector O_t^{v} of the visual ones. They capture the presence/absence of the three key sound classes (tennis, applause, and music) on the soundtrack and inside the time window defined by the respective shot boundaries. As it is captured just the presence, multiple instances or different orderings will yield the same set of auditory descriptors.

The vector of observations O_t^{v} of eq. 4.28 is thus redefined as:

$$O_t^{\mathrm{av}} = \begin{bmatrix} \text{visual similarity} \\ \text{shot length} \\ \text{dissolve shot} \\ \text{tennis sound} \\ \text{applause sound} \\ \text{music sound} \end{bmatrix} = \begin{bmatrix} o_t^c \\ o_t^l \\ o_t^d \\ o_t^b \\ o_t^a \\ o_t^m \end{bmatrix} \tag{4.31}$$

The observations are considered again as independent. Viterbi decoding changes only in the inclusion of the new features. Eq. 4.30 is thus modified so as $r \in \{c, l, d, b, a, m\}$.

4.3.3 Parameter Estimation

The parameters of HMMs were estimated from labeled training sequences instead of using the Baum-Welch algorithm. Although this algorithm can provide a good solution, especially when a good initial condition is given a-priori, the estimation of the parameters via labeled sequences is straightforward and results to the true, and thus optimal, parameter set as the hidden state sequence is manually uncovered.

Each shot of the training sequences was labeled according to Fig. 4.3(b). The parameters (A, B, π) are learned as *relative frequency of occurrence*:

$$\pi_i = \frac{\text{number of occurrences of } S_i \text{ at time } t = 1}{\text{number of sequences}} \tag{4.32}$$

$$\mathrm{a}_{ij} = \frac{\text{number of transitions from } S_i \text{ to } S_j}{\text{number of occurrences of } S_i} \tag{4.33}$$

[2]Set after experimentation.

$$P(o^r|S_i) = \frac{\text{number of occurrences of } S_i \text{ with } o_t^r = o^r}{\text{number of occurrences of } S_i} \quad (4.34)$$
$$1 \leq i \leq 12, \quad 1 \leq j \leq 12, \quad r \in \{c, l, d, b, a, m\}$$

The observation distributions $P(o^r|S_i)$ are estimated thus as probability histograms. Especially for the distributions of visual similarity and the shot length features, smoothing of the histograms with a mask [1 1 1] was applied after estimation to avoid overfitting to the training data.

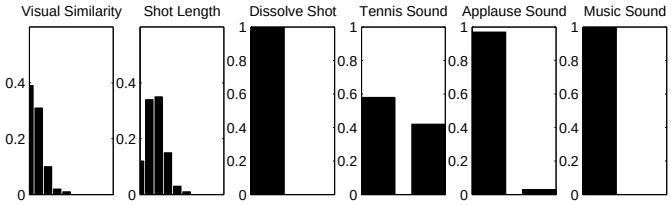
Qualitative Analysis of the Features

In order to provide some qualitative analysis of the features, the observation probability histograms are given in Fig. 4.6 and for the states 1, 3, 4, 9, and 10 (missed serve, exchange, shot after exchange, idle court view and non-court view shot of commercials or dead time, respectively). For the first state, we notice a high probability for low visual similarity (i.e., low distance to the reference court view) and low shot length, as expected for a missed serve shot. The probability of capturing a tennis sound is however mitigated. The reason is that tennis sound is difficult to detect and in missed serves only one ball hit occurs (the serve), which makes its detection even harder. Note also that tennis sounds are detected with some small probability in all the last 3 states in the figure, which do not contain game action. The difference between states 1 and 3 lies mainly on the fact that states 3 can be more lengthy being normal exchanges and also the detection of the tennis sounds is more reliable. In Fig. 4.6(c), we see that state 4 produces greater distances of visual similarity and also a great deal of applause is detected. State 9 corresponds to idle court views. Compared to states 1 and 3, the absence of ball hits and the presence of applause is more probable. Finally, in state 10 music is detected in contrast to the above 5 states in the figure, where music occurs with zero probability. Furthermore, we notice shots lengths of the full spectrum. Most of them are short commercials shots, but there exist also some number of lengthy dead time shots.

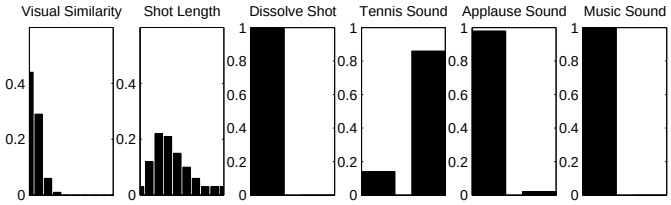
Overall, visual similarity seems to provide good discriminative capabilities, in accordance to Fig. 3.9. The importance of the dissolve transition features is evident, while shot duration provides also its contribution. Regarding the auditory features, music and applause are detected reliably, but tennis sounds are clearly harder to detect, producing a lot of misses or false alarms.

4.4 Content Modeling with Segment Models

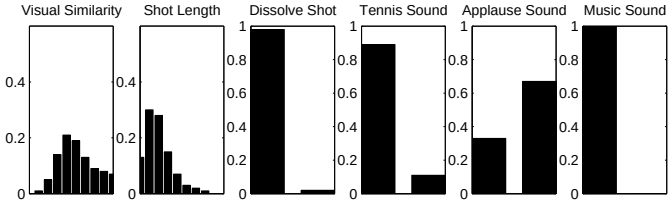
SMs extend the shot-based features of HMMs to *scene-based* ones. The observation distributions thus model the entire audiovisual content of a scene. The video is modeled



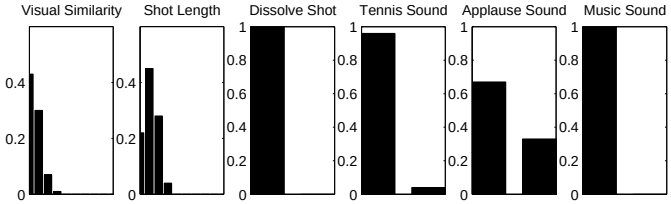
(a) state 1



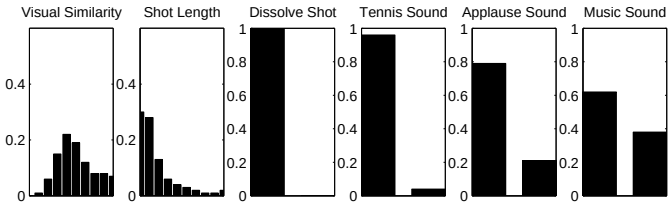
(b) state 3



(c) state 4



(d) state 9



(e) state 10

Figure 4.6

The observation probability histograms for the states 1, 3, 4, 9, and 10. For each state and for the last 4 histograms, which correspond to binary descriptors, the first bar gives the probability of absence of the feature and the second one the probability of presence.

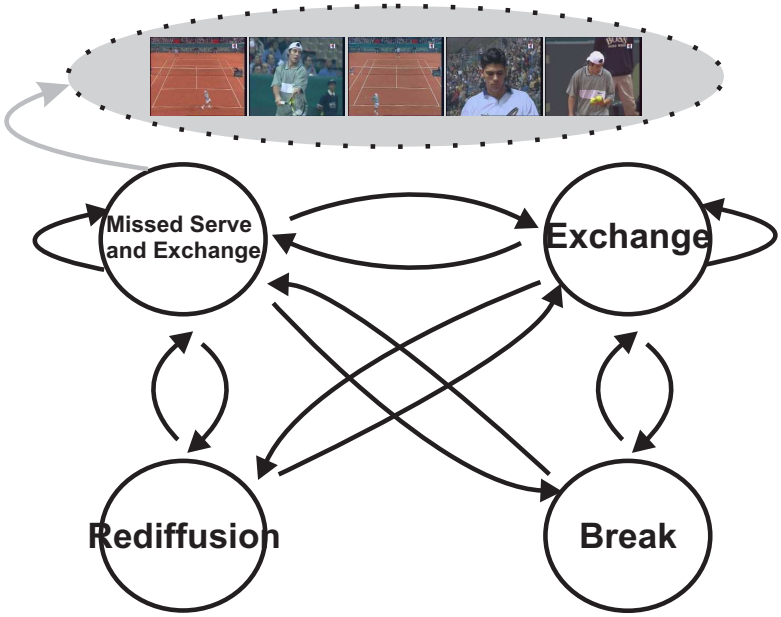


Figure 4.7
Modeling of the visual content as a succession of scenes with SMs.

then by an ergodic SM as a succession of scenes (as opposed to a succession of shots, with HMMs). A formal description of the SM is firstly given based only on the videotrack. The integration of the audiotrack and the various possibilities that SMs offer for segmental feature modeling are then discussed. The section concludes with parameter estimation details.

4.4.1 Modeling of the Visual Content

The SM will contain 4 hidden states, each one corresponding to the scenes defined in section 4.2.2. An illustration of the generation of the video with the SM is given in Fig. 4.7. The stochastic process enters into a hidden state (or, in other words, into a scene) and emits several observations in the form of shot-based observations. It then switches into a new scene and so on, until the complete video is generated. There is a full ergodic structure, except the non-allowed self-transitions for the rediffusion and break scenes. This is because, by definition, multiple repeated rediffusions or breaks result into a larger and unique rediffusion or break, respectively. Furthermore, it is not allowed a rediffusion after a break and vice versa.

The segment length l normally would be defined as the number of shots that the scene contains. It was found though in early experimentation that it is better to be defined as the real time duration of the scenes. The rational behind this is that, real time duration

captures what really happens in the tennis court. On the other side, the number of shots depends on the producer's choices: if the producer chooses frequent alternations of non-court views after an exchange or, on the contrary, some few lengthy shots, this results in great variability of the number of shots for a scene and for a given real time duration. In addition, commercials and dead time last approximately the same real time (defined by the duration of a game pause) but commercials contain a great number of (short) shots and dead time just a few (long ones). The term l in the expression of the duration distribution $P(l|i)$ will refer thus to the real time duration of the scene i. Outside this context, it continues to represent the length of a segment $O^{\mathrm{v}}_{1:l}$, i.e., the total number of shots it contains.

It remains to define the segmental observation distribution $P(O^{\mathrm{v}}_{1:l}|l, S_i)$, which is more complicated compared to HMMs because sequences of features are now modeled. The simplest approach is to assume the observations of a segment as independent (eq. 4.20), reducing the SM to an Explicit State Duration HMM. Of course, in this way the inter-segmental dependecies are discarded, which would result to a poor modeling as there exists a strong temporal context inside each scene. A more efficient way to compute a probability score over a sequence is given by an HMM λ_i, specialized to model the observations of the scene i:

$$P(O^{\mathrm{v}}_{1:l}|l, S_i) \equiv P(O^{\mathrm{v}}_{1:l}|\lambda_i) = \sum_{Q_{1:l}} P(O^{\mathrm{v}}_{1:l}, Q_{1:l}|\lambda_i) \tag{4.35}$$

The last term is a sum over all the hidden state paths of the HMM and is calculated by the forward pass (see section 4.1.1). The HMM λ_i operates essentially as an *observation scorer* and should not be confused with the HMMs of Fig. 4.3. For instance, the observation scorer HMMs can have different topologies to the ones of Fig. 4.3(b). Strictly speaking, the SM thus defined differs to the full-model HMM of Fig. 4.3(c) firstly by the inclusion of the duration model. Secondly the segmental scores are calculated over all hidden state paths $Q_{1:l}$, while in HMMs over the best ones $Q^*_{1:l}$ as part of the Viterbi decoding. Finally, the probabilities of the scene transitions are spread over all the hidden states of the full-model HMM and are calculated as probabilities of shots transitions. In the SM, they are the true scene transition probabilities.

Viterbi decoding for SMs, as explained in section 4.1.3, involves finding the most likely scene labels *and* the scene boundaries in an enhanced optimization problem. In the tennis video context, equation 4.21 (in the log domain) translates to:

$$(L^*_{1:N^*}, Q^*_{1:N^*}) = \arg \max_{L_{1:N}, Q_{1:N}} \{\ln p(Q_{1:N}) + \ln p(L_{1:N}|Q_{1:N}) + \ln p(O^{\mathrm{v}}_{1:T}|L_{1:N}, Q_{1:N})\} \tag{4.36}$$

where T is the total number of shots of the video, $L^*_{1:N^*}$ gives the segmentation into scenes and $Q^*_{1:N^*}$ is the most likely hidden state sequence found. The term $p(O^{\mathrm{v}}_{1:T}|L_{1:N}, Q_{1:N})$

is defined by eq. 4.35. An illustration of the decoding procedure is given Fig. 4.8. It provides at once the most likely scene label for each shot and the most likely segmentation into scenes.

4.4.2 Audiovisual Integration

Transferring the discussion of section 4.1.2 on the limitations of HMMs to the video indexing context, the use of segmental features, rather than frame-based ones, can be beneficial for the audiovisual integration. The key idea is to separate the audiovisual content of a scene into a "visual" segment and an "auditory" one. The auditory segment can then be sampled at its native sampling rate and can be modeled by a different segmental observation distribution. The synchrony constraint between the two streams is relaxed to the scene boundaries. This is a valid assumption because, according to the definition of section 4.2.2, all the relative visual and auditory features lie inside the scene.

Formally, the audiovisual content $O^{\mathrm{av}}_{1:l}$ of a video portion corresponding to l successive shots is factored as:

$$P(O^{\mathrm{av}}_{1:l}|l, S_i) = P(O^{\mathrm{v}}_{1:l}|l, S_i)P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i) \tag{4.37}$$

where l_{a} is the length of the auditory segment, i.e., the number of samples it contains. SMs allow l_{a} to be different than l. This factorization and the fusion of two asynchronous streams is depicted in Fig. 4.9. In the remaining of this section, different modeling possibilities for $P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i)$ are explored.

Scene-Based Auditory Features

Having a series of auditory features collected at the scene level, the simplest case is just to capture the presence of each sound event inside a scene. It is a direct extension of the shot-based auditory descriptors used in HMMs to scene-based ones. However, the key difference is that the auditory features are allowed now to be *asynchronous* to the visual ones. The segmental score is merely defined as:

$$P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i) = \prod_{k=1}^{l_{\mathrm{a}}} P(O^{\mathrm{a}}_k|S_i) \tag{4.38}$$

where $P(O^{\mathrm{a}}_k|S_i)$ is the probability of presence in the scene of the three audio events. The length l_{a} of the auditory segment is defined by the number of audio events that the scene contains.

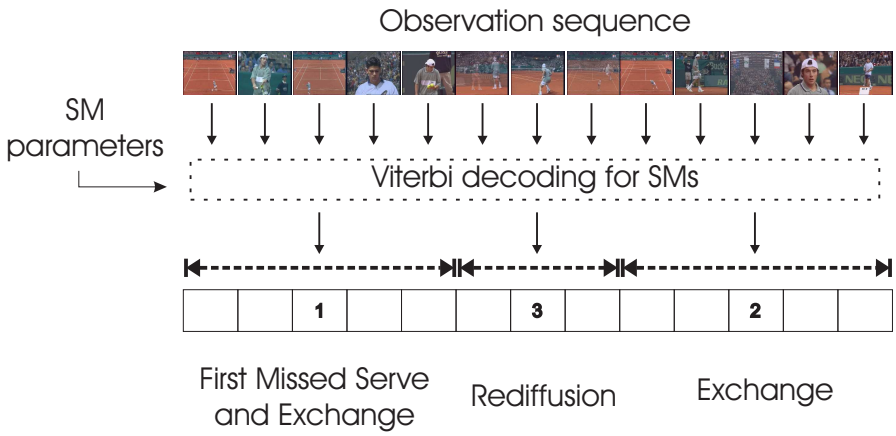


Figure 4.8

Viterbi decoding with SMs for finding the most likely segmentation into scenes and the most likely scene labels. In contrast to Fig. 4.4, these two problems are solved at once.

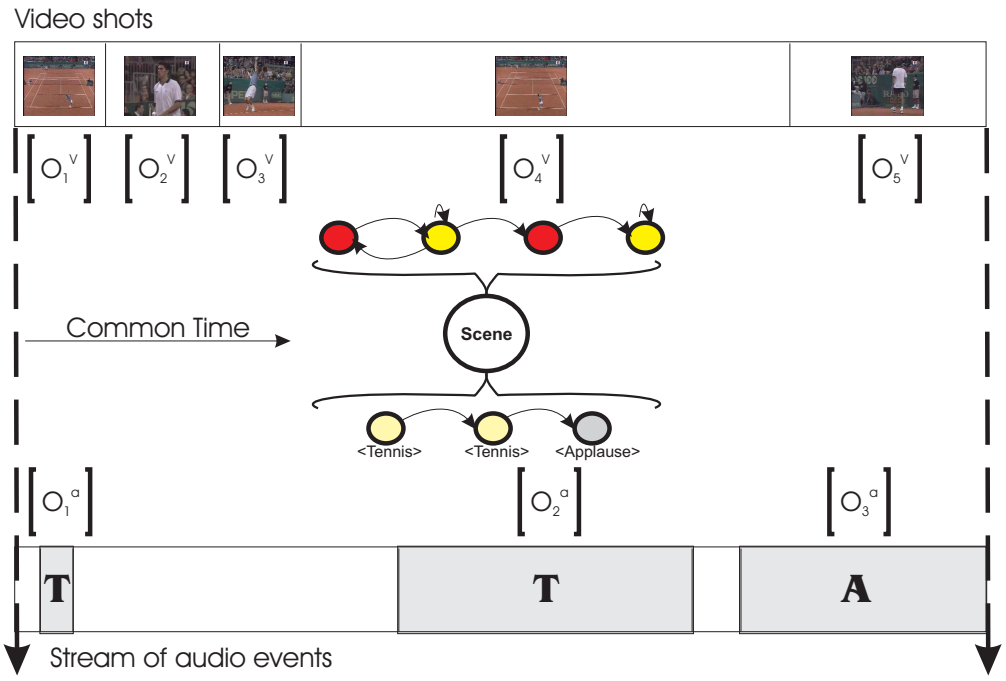


Figure 4.9

Asynchronous audiovisual integration at the scene level with SMs. The visual stream is sampled at the rate of shots while the auditory one at the rate of sound events. Different models are employed for each stream. The segmental distributions are fused at the scene level.

Succession of Sound Events

Capturing just the presence of the events in the scenes could be insufficient. Indeed, the audio content of the 'First Missed Serve and Exchange' scene is normally a succession {tennis, tennis, applause}, while for the 'Exchange' scene is {tennis, applause}. The presence of audio events will result to an identical model for the two scenes. To account for this inconveniency, capturing probabilities of succession of audio events can be used instead:

$$P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i) = \prod_{k=2}^{l_{\mathrm{a}}} P(O^{\mathrm{a}}_k|O^{\mathrm{a}}_{k-1}, S_i) \tag{4.39}$$

Besides the three basic sound events, two more special events are introduced here: the "start" and "end" of segment. The audio content of Fig. 4.9, for instance, will be {start, tennis, tennis, applause, end}.

HMM Scorers with Cepstral Features

So far, the audio stream is integrated in the form of pre-detected and pre-classified audio events. In a quite different approach, auditory segments are modeled directly on top of the audio cepstral coefficients. The reason for doing this is twofold: firstly, the erroneous pre-segmentation of the audio track into homogeneous segments is avoided. The bounds of these segments are hard to detect and generally more vague that the hard cuts of the video track. Secondly, the content of these segments can contain more than one audio class like ball hits superimposed by speech. This may make the pre-classification into sound classes and the extraction of audio descriptors from these segments erroneous.

In order to model the audio content on top of generic cepstral features, Continuous Density HMMs are employed. Their probabilistic score is calculated via the forward pass, as for the HMMs that model the visual segments (eq. 4.35). The audio stream is sampled now at the rate of 100 frames per second. The length l_{a} of the auditory segment is now equal to the total number of audio frames of the scene.

State-Synchronous Audiovisual HMM Scorers

Finally, nothing prohibits the use of a concatenative shot-based fusion scheme, exactly as with standard HMMs. The feature vector of the visual HMM scorer is enhanced with the (synchronized) auditory descriptors, as shown in Fig. 4.5, while the audio model $P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i)$ is discarded.

4.4.3 Computational Cost

At the end of section 4.1.3, the computational cost of Viterbi decoding for SMs was estimated as $O(L_{\max}TN^2)$, where T is the sequence length (i.e., the number of shots of

a video, typically around 2000, heavily depending on the game duration) and N is the number of hidden states of the SM ($N = 4$, in our case). The term $L_{\max}$ is the maximum segment length, i.e., the maximum number of shots that a scene can contain. It was set to 80, after inspection of both test and training sequences (a greater value can be safely used as it does not alter the results).

However, these estimations are done with the premise that the computation of a segmental score $p(O_{1:l})$ can be performed in $O(1)$ time, which is generally false. Indeed, for all of the models previously described, their computational cost scales linearly with the segment length. Especially for the Continuous Density HMMs (operating at 100 fps), the overall computational cost explodes. This extra overhead can be avoided by eliminating redundant computation. Indeed, at the maximization step of eq. 4.23, the scores for the sequences $O_{t:t}, O_{t-1:t} \dots O_{t-k+1:t}$ are needed to be evaluated *successively*. These sequences start at different points but their end is common. The use of the backward pass to evaluate the likelihood of eq. 4.23, instead of the forward pass, will permit to reuse the computation of the score for $O_{t-k+1:t}$ when computing the score for $O_{t-k:t}$, and so on. Using such a computation caching, the argument that segment models are approximately $L_{\max}$ times slower than HMMs still holds.

The implementation of Viterbi decoding for standard HMMs yielded a computational cost of less than a second for a full broadcast. Viterbi decoding for SMs with discrete observation scorers require around half a minute. The use of CDHMMs without caching explodes decoding time to several hours. This is reduced to some minutes when caching is employed. Note also that, in addition to score caching between successive segmental evaluations, the GMM likelihoods of the continuous densities were in fact pre-computed for all the video before decoding. The disadvantage of this scheme is that extensive memories capabilities are required (of around 500 MB), which is nevertheless inside the capabilities of a modern workstation.

4.4.4 Parameter Estimation

Having the ground truth of the sequences, the transition probabilities of the model were estimated as relative frequency, as in eq. 4.33 for HMMs. The initial distributions π_i were set uniform. The duration models $P(l|S_i)$ were estimated as probability histograms, after their quantization into 30 bins each[3]. Finally, the histograms were smoothed with a [1 1 1] mask. The four histograms are given in Fig. 4.10. We notice distinctive behaviors for the four scenes, especially for the breaks that are much more lengthy than the other scenes. Rediffusions are most of the times the shortest scene, as expected.

Regarding the audio unigram and bigram models, the probabilities were again estimated based on the relative frequency. For the bigram model especially, a backing-off

[3]Set after experimentation.

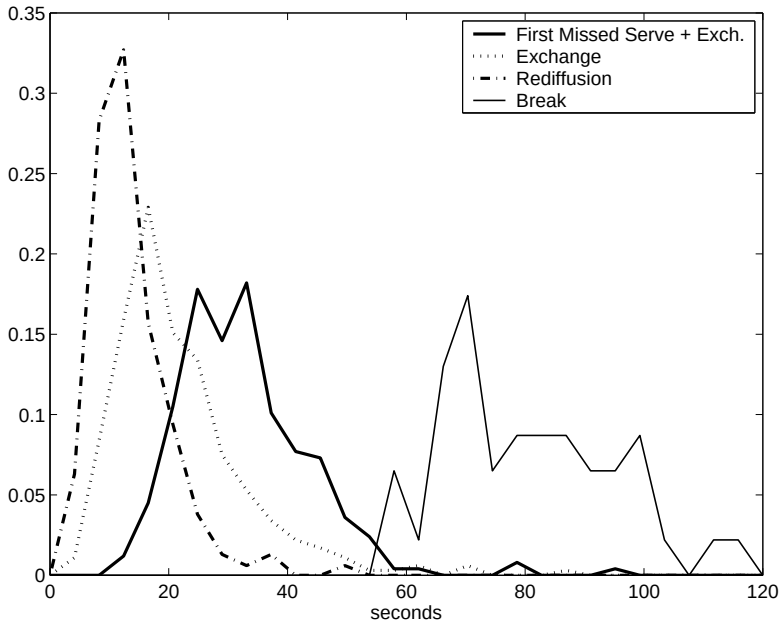


Figure 4.10
Probability histograms for the four duration models.

scheme was used to smooth the distributions and thus to improve generalization. More precisely, the probability of the succession of the sound events i to j was calculated as:

$$P(j|i) = w\frac{\mathcal{C}_{ij}}{\mathcal{C}_i} + (1-w)\frac{\mathcal{C}_j}{\sum_k \mathcal{C}_k} \tag{4.40}$$

where $\mathcal{C}_{ij}$ is the number of times that the transition i to j occurs, $\mathcal{C}_i$ is the number of times the event i occurs, and $w = 0.75$, set after experimentation on the training sequences. This formula is essentially a ponderation between the estimated probability of succession and the probability of presence of the event j.

The parameters of the discrete HMM probability scorers were estimated via the Baum-Welch algorithm. The topologies of the HMMs for the scenes "First Missed Serve and Exchange" and "Exchange" were defined as in Fig. 4.3(b). In the initialization phase of the algorithm, the non-allowed transitions in these figures were set to zero, letting a small and appropriate set of the transition probabilities be learned. The quality of the solution thus obtained is better. The HMMs for the rediffusion and break scenes contain 3 and 2 hidden states, respectively. No special initialization was used for these two scenes as it was not found to improve performance. As a final note, the final solution obtained by the EM-algorithm is generally affected by the random initialization of the parameters. Multiple runs of the algorithm were tested, giving a similar performance on the experiments and without affecting the comparative study that follows in the next

section.

To estimate the parameters of the CDHMM scorers (GMM components and transition probabilities), the HTK toolkit[4] was employed. Among various topologies tested, the best results were obtained with left-to-right HMMs with no jump of more than two states, i.e., with transition probabilities $a_{ij} = 0$ when $j > i + 1$. The number of hidden states for all the four HMM scorers was fixed to 20 and each of them defined 32 Gaussian components, initialized in a uniform manner. The HMMs were trained with the Baum-Welch algorithm until convergence was noticed. Regarding the cesptral features, they were extracted from the audio waveforms with the Spro toolkit[5] and with standard choices for the extraction parameters. The audio frames consist of 12 cepstral coefficients and the energy, plus the first order derivatives.

The CDHMMs produce in fact likelihoods and not probability scores, whose range of values is much greater than the ones of discrete HMMs. To balance their impact during decoding, the likelihoods were linearly rescaled in an appropriate range of values.

4.5 Experimental Results

Firstly, let be reminded that half of the videos were reserved for testing purposes. Also, as the ground truth of the videos was collected on top of the automatic videotrack segmentation, errors of the hard cut and dissolve detection are not taken into account in this analysis (see section 4.2.4 for more details).

Performance measurements include firstly the percentage C of shots classified with the correct scene label, averaged over the test sequences. This is a measurement for the quality of the classification. In addition, we need a measurement regarding the quality of the segmentation into scenes. For example, if two or more successive scenes of the same label are correctly classified but the retrieved boundaries are not correct, the measurement C will give a (false) good performance. Segmentation quality can be quantified by the recall R and precision P rates on the scene boundary detection (see eq. 3.13 for the exact definition of these measurements), averaged over the test sequences. To make the comparison between two approaches easier than having to compare 3 statistics, a single performance measurement is defined as:

$$\hat{F} = \frac{3CPR}{C + P + R} \tag{4.41}$$

which is reminiscent of the standard F-measurement[6], widely used in statistics. Of course, the use of $\hat{F}$ is purely suggestive and all of the C, P, R rates are provided.

[4]HTK toolkit Homepage: http://htk.eng.cam.ac.uk

[5]Spro toolkit Homepage: http://www.irisa.fr/metiss/gravier/spro/index.html

[6]The F-measurement on the precision and recall rates is $F = \frac{2PR}{P+R}$.

Results on the test set are reported in table 4.2 for HMMs and SMs and with varying assumptions and feature sets.

4.5.1 Hidden Markov Models

We see in the first rows the performance of HMMs. In the first row ('HMMs-V/A=1'), visual features are used only and the hidden state sequence probabilities were set uniform ($p(Q_{1:T}) = 1$ in eq. 4.12). This results into classifying each shot individually, completely ignoring the temporal dimension of the problem. As expected, the performance is very poor. Video syntax thus plays an important role and must be taken into consideration. Next we see the performance with video-only ('HMMs-V') and audiovisual ('HMMs-VA') features. While most of the contribution comes from the visual features, a clear performance improvement is noticed with the addition of the auditory descriptors to shot-based feature vectors.

Let recall from Fig. 4.6 that tennis sound appears hard to detect and thus the tennis sound feature is a considerable source of error. At the same time, it is a crucial auditory feature as it helps to discriminate between court views that are idle or contain game action. To verify the impact of this feature, it was manually set to its perfect values, i.e., 1 for the shots of the hidden states 1, 3, and 5 and 0 otherwise. The remaining two auditory descriptors keep their automatically-obtained values. After retraining the HMM, the obtained results are given in the fourth column ('HMMs-VA*'). We notice so that a perfect tennis sound detection would greatly improve performance.

4.5.2 Segment Models

Video-Only Segment Models

The performance of SMs is given at the next rows of table 4.2. Using video only features modeled by an HMM scorer and firstly setting the transition probabilities and durations scores as uniform ('SMs-Vhmm/A=1, D=1'), a poor performance is achieved. However, it is better compared to 'HMMs-V/A=1' as the HMM scores still contain the inter-scene transition probabilities. At the next two rows, the transition probabilities scores ('SMs-Vhmm/D=1') and the duration scores are added, to give the performance of the full SM on video only data ('SMs-Vhmm'). We see thus that the addition of the duration model to SMs results finally to a better performance compared to HMMs (compare 'HMM-V' and 'SMs-Vhmm'). This is a first interesting outcome: SMs perform better than HMMs, despite the fact that they operate in an enhanced search space of possible paths and segmentations.

	C	P	R	$\hat{F}$
Hidden Markov Models				
HMMs-V/A=1	41.54	89.92	18.51	13.10
HMMs-V	76.30	81.99	73.44	59.48
HMMs-VA	80.23	84.69	79.70	66.42
HMMs-VA*	84.75	92.20	88.78	78.32
Segment Models				
SMs-Vhmm/A=1, D=1	51.52	96.88	32.25	26.73
SMs-Vhmm/D=1	64.48	88.90	64.62	50.98
SMs-Vhmm	79.69	83.54	74.82	62.78
SMs-VhmmA1gram	80.07	83.86	75.27	63.39
SMs-VhmmA2gram	81.77	84.10	79.45	66.81
SMs-VhmmAcep	79.86	84.64	75.20	63.62
SMs-VhmmA*2gram	82.58	89.15	80.56	70.53
SMs-(AV)hmm	84.39	86.25	79.32	69.29
SMs-(A*V)hmm	89.60	93.07	89.19	82.07
SMs-VhmmAhmm	81.52	87.83	77.55	67.47
SMs-(LD)hmmChmmAhmm	78.96	86.33	75.02	63.84
SMs-(AV)hmmA2gram	84.73	84.13	81.67	69.71

Table 4.2

Performance comparison of HMMs and SMs with varying feature sets and modeling assumptions. The C, P, and R rates are averaged over the three test videos and then $\hat{F}$ is calculated.

Asynchronous Audiovisual Integration

At the next three rows, the performance of SMs with added asynchronous auditory features is reported. Scene-based probabilities of sound events ('SMs-VhmmA1gram') can improve performance, though not in a satisfactory level. The reason is simple: this model captures for instance the presence of tennis sounds in the scene, which finally cannot help in the discrimination of the scenes 'First Missed Serve and Exchange' and 'Exchange'. The difference between them and with respect to the soundtrack is that the first one contains a succession of two tennis sounds (the missed serve and then the exchange). This can be more efficiently captured using the bigram models ('SMs-VhmmA2gram'). Indeed, a clear performance gain is noticed (+4.03% compared to +0.61% of 'SMs-VhmmA1gram', in the $\hat{F}$ scores) with the integration of audio with this model. However, the performance gain of the HMM-based audiovisual integration (+6.94%, moving from 'HMMs-V' to 'HMMs-VA') is still larger. An analysis why this happens is given later in this section, after reporting the performance of state-synchronous audiovisual HMM scorers.

The performance of the integration of audio in the form of cepstral features is given at the next row ('SMs-VhmmAcep'), where again a poor performance gain is noticed. It is clear that the Continuous Density HMM scorers yield a poor performance, affected by the many degrees of freedom that the problem has. Indeed, they are charged to model a target semantic (i.e., a *scene*) from scratch and receive as input a large number of high-dimensional feature vectors. In the 'SMs-VhmmA1gram' and 'SMs-VhmmA2gram' approaches instead, a great deal of prior knowledge exists as the interesting sound events that the soundtrack contains are defined a-priori. Nevertheless, this approach demonstrates that SMs provide a great deal of freedom for multimodal fusion. In this specific scenario, visual information sampled at the rate of shots is fused with auditory features sampled at 100 fps, making no hard synchrony assumptions.

Finally, we see in row 'SMs-VhmmA*2gram' what could be achieved with a perfect tennis sound detection. Again, the asynchronous fusion seems not to perform well, even with perfect tennis sound features.

Synchronous Fusion

The performance of the state-synchronous audiovisual HMM scorers is reported at the row 'SMs-(AV)hmm' of table 4.2. The results obtained with this model are better compared to the above-mentioned asynchronous fusion models. Not surprisingly, the performance gain of adding audio information in this way is +6.51%, almost the same to the +6,94% obtained with HMMs. At the next row ('SMs-(A*V)hmm'), we see the performance of this model when the tennis sound descriptors are manually set to their optimal values, as in the case of 'HMMs-VA*'. It is again an indication of what could

be achieved with a perfect tennis sound detection method.

Early Versus Late Audiovisual Integration

By comparing the models 'SMs-VhmmA2gram' and 'SMs-(AV)hmm', it appears that the asynchronous fusion of auditory information results to a small performance degradation. This seems to be awkward as the two models use the same information (visual and auditory), only the integration method changes. In a first look, one may assume that the bigram model (in 'SMs-VhmmA2gram') is a poor model and, at the bottom line, the shot-based auditory descriptors could be more informative. To verify this, the audiovisual HMM scorer of 'SMs-(AV)hmm' was split into two independent HMMs with the same number of hidden states. The first HMM receives the visual features and the second one the shot-based auditory features[7]. This approach uses exactly the same shot-based feature set of 'SMs-(AV)hmm' but models the auditory and the visual segments independently. It replaces thus the bigram model of 'SMs-VhmmA2gram' with the auditory shot-based descriptors used in 'SMs-(AV)hmm'. Its performance is given as 'SMs-VhmmAhmm' in table 4.2. A close performance is noticed with 'SMs-VhmmA2gram', which implies that the bigram models are more or less as informative as the auditory shot-based descriptors.

What is noticable in the above figures is that, when modeling the auditory and the visual segments independently, the performance drops. This is even more clear with the 'SMs-(LD)hmmChmmAhmm', where the feature set of the visual segment is splitted to the visual similarity feature and to the shot length and dissolve features. This last model uses three independent HMM scorers, one for the auditory and two for the visual segment. The performance now drops much more. It appears thus that important correlations between the multimodal features are not efficiently captured when the HMM scorers first compute the segmental scores and then integrate them, as opposed to 'SMs-(AV)hmm' which firstly integrates and then computes the segmental score.

Let us recall from chapter 2 that, when features are mapped directly to the semantics and then integration is performed, we talk about late fusion, while when integration precedes the decision on the semantics, we talk about early fusion. The above analysis thus suggests that, for the problem at hand, an early integration is preferable. However, generally speaking of video indexing, the question whether a late or an early fusion is best remains open. Two studies [42, 96] for instance have concluded favorably to late fusion, though with little difference in performance. In the first one, Huang *et al.* compare concatenative fusion with HMMs and product of HMM likelihoods, in a similar setting to this study (with respect to the integration). In the latter one, Snoek *et al.* compare late and early integration with SVM classifiers.

[7] This results into something similar to the multi-band HMMs, proposed in speech recognition.

SMs-Vhmm				
	1	2	3	4
1	77.1	20.3	2.4	0.2
2	22.6	75.5	1.3	0.6
3	10.9	4.4	84.7	0.0
4	2.9	7.8	6.9	82.4

SMs-(AV)hmm				
	1	2	3	4
1	86.1	11.8	1.7	0.4
2	18.8	79.2	1.3	0.7
3	7.5	4.5	87.4	0.5
4	2.8	7.7	5.3	84.2

SMs-(A*V)hmm				
	1	2	3	4
1	83.1	14.3	2.2	0.5
2	7.2	91.2	1.3	0.3
3	0.2	0.4	98.4	1.0
4	2.0	4.0	2.6	91.4

Table 4.3
Confusion matrices on the classification of shots with scene labels. Each row 1-4 refers to a scene and each column gives the percentage of the shots that are classified with the respective class label 1-4.

A possible solution to overcome the problem of late fusion could be to integrate audio with both late and early fusion. Techically, this approach uses for both the visual and the auditory segments their native sampling rates and topologies and, at the same time, give hints on what happens in the other modalities before the late integration. This double fusion scheme is given as 'SMs-(AV)hmmA2gram' in the table 4.2, where a slight performance improvement is noticed over 'SMs-(AV)hmm'.

4.5.3 Confusion Matrices

The confusion matrices regarding the classification of the shots are given in table 4.3 for the 'SMs-Vhmm', 'SMs-(AV)hmm', and 'SMs-(A*V)hmm' models. It is clear that most of the confusion comes between the first two scenes ('First Missed Serve and Exchange' and 'Exchange', labeled as 1 and 2 in the table), while rediffusions and breaks are well-detected (labeled as 3 and 4 in the table). The addition of auditory features and of perfect tennis sound detection gradually removes the confusion, especially for the first two scenes.

CHAPTER 5

Hierarchical Topologies and Integration of the Score Labels

The succession of the scenes followed a flat and ergodic structure in the previous chapter. A tennis match however contains an inherent hierarchical structure as a result of the match organization and rules. The incorporation of this structure to the model topologies, for both HMMs and SMs, is discussed in this chapter. The use of hierarchically structured topologies allows the detection and labeling of higher-level structures in the video.

Besides the hierarchical topologies, the integration of the score labels is also discussed in detail in this chapter. The score labels, which can be viewed as the textual modality of a tennis video, possess important high-level information on the game events that is useful to be incorporated during decoding. A novel decoding scheme was developed, called Score-Oriented Viterbi Search, which uses the positions and the contents of the labels to pilot the Viterbi algorithm and to ensures that the solution obtained is consistent with the actual game evolution.

The common point between hierarchical topologies and Score-Oriented Viterbi Search is that they both impose a set of meaningful *constraints* that reduce the initial Viterbi search space.

5.1 Hierarchical Topologies

We will briefly review the hierarchical decomposition of a tennis match and then present the Hierarchical HMM, which provide usefull notions for modeling hierarchical processes. The application of the Hierarchical HMM in tennis will result to models where the ergodic scenes transitions are replaced by hierarchical ones. Some implementation details of decoding with this complicated models are provided at the end.

5.1.1 Tennis Match Organization and Rules

We know that each tennis match contains a hierarchical structure of sets, games and points that reflects the match organization and rules[1]. The elementary unit of this structure is a *point*, which is the outcome of an exchange. In the rather awkward tennis naming scheme, the first three point are named as '15', '30', and '40'. When a player has reached the third point and has also scored two more points than the opponent, then he or she wins a *game*. In their turn, games are organized into *sets*. Usually, the first player that wins 6 games takes the set. A tennis match can contain at maximum 3 or 5 sets, depending on the tournament. The match stops when a player has taken the sufficient majority of the sets (i.e., 3 sets in a match of 5 sets or 2 sets in a match of 3 sets). Finally, we know that after the first game of the set and then after two successive games, a break occurs. An example of this hierarchical decomposition of a tennis match is given in Fig. 5.1.

5.1.2 Hierarchical Hidden Markov Models

The Hierarchical Hidden Markov Model (HHMM), originally introduced by Fine *et al.* in [28], models a Markovian process that is analyzed hierarchically in a multi-level structure. The hidden states of this model are split into two categories, the internal and the production states. The internal states specify the hierarchy of the model and do not emit observation symbols. In a recursive definition, each internal state serves as the root node of a sub-HHMM. The recursion ends to the production states, which are the emitting states of the model and share all the properties of the hidden states of standard HMMs. An internal state is formally defined by the transition probabilities A between its child nodes and their initial state distribution π.

From a generative point of view, the sequence of the observation symbols is generated as follows. The root internal state is activated and then it activates according to its initial state distribution one of its child internal states. This repeats recursively until a production state is reached and an observation symbol is emitted according to its

[1]Tennis rules change over time and it is out of scope of this study to fully comply with them. For more details, see the Homepage of International Tennis Federation, http://www.itftennis.com

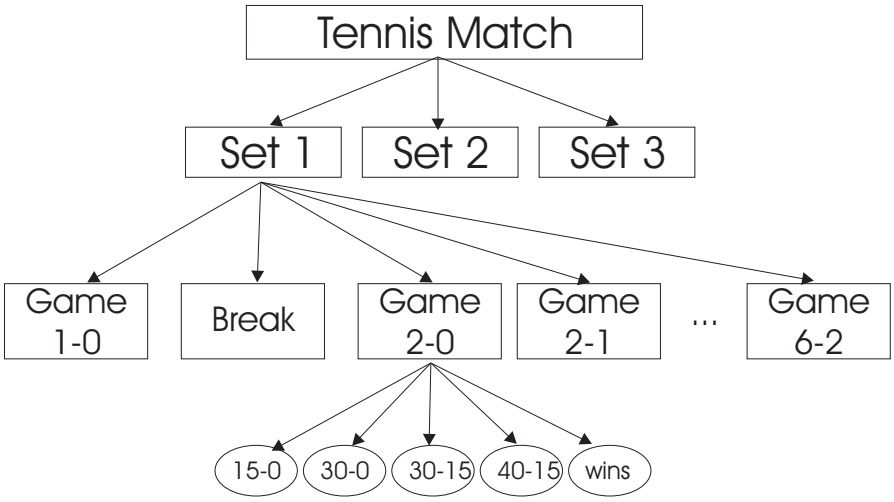


Figure 5.1

The hierarchical structure of an example tennis match into sets, games and points. For the convenience of the presentation, all the nodes are not expanded. The match consists of 3 sets. The first one is decomposed to 8 games. Finally, the second game contains 5 points. A break appears between the first and second game.

observation distribution. The stochastic process can then remain at the same level and transit according to the transition probabilities of the last internal state or it can jump one level up, meaning that an internal state has completed its activation. These jumps are triggered when a special internal state is reached, the final state. The generative process continues in this recursive manner until the root node of the HHMM is reached again.

The adapted versions of the EM algorithm and of the Viterbi decoding for HHMMs is formulated in [28]. HHMMs were used by Fine *et al.* to capture the inherent multi-level statistics that appears in handwritten text. In this study, however, HHMMs serve in fact, firstly, as a source of constraints that the Markovian process follows the hierarchical decomposition of the tennis match, and secondly, to recover during decoding high-level game semantics. Therefore, in order not to abuse the original definition of HHMMs, the term Hierarchical Topology is used instead.

5.1.3 Hierarchical Scene Transitions

The hierarchical structure of a tennis match can be easily incorporated in the models by extending the flat and ergodic scene transitions of Fig. 4.7 to hierarchical ones, sharing notions with HHMMs. The new model topology (applied to both the HMMs and SMs of the previous chapter) is given in Fig. 5.2.

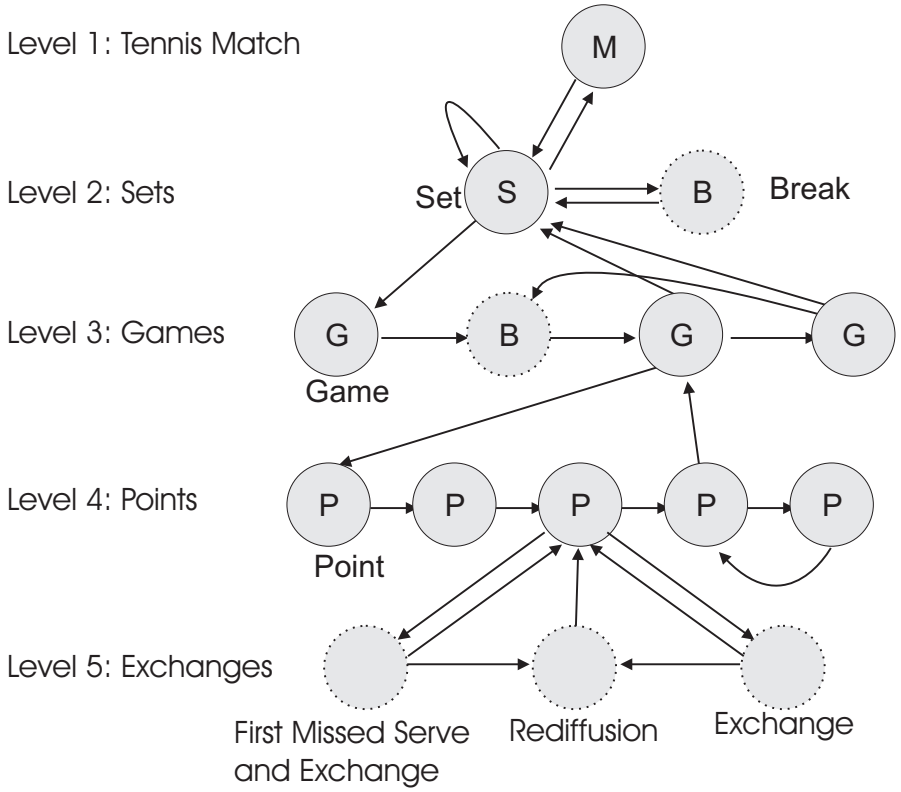


Figure 5.2

The hierarcical topology that encodes tennis game organization and rules. At each level, only one node is fully expanded. The arrows depict allowed transitions according to tennis rules. The dotted internal states are production states for the SM. For the HMM, they are further analyzed as in Fig. 4.3(b).

- *Level 1: Tennis Match.* At the first level of the hierarchy we see the root internal state, which represents the whole tennis match. It has two child internal states, one representing a tennis set and one representing a break that occurs between two sets. Normally, it should contain 2 or more sets, according to the minimum number of sets the match can contain. But this number changes with the tournament and thereby a new topology should be constructed for every new match. Furthermore, it was found that during decoding, there is no performance change if two or more internal states of sets are defined. And, needless to say, adding more internal states multiplies model complexity and decoding time, especially when these internal states are close to the root.

- *Level 2: Sets.* The internal hidden state representing a set is analyzed as a succession of games and breaks, as shown in Fig. 5.2. Strictly applying tennis rules, it should contain at least 6 game internal states but they were rejected as redundant like in the case of extra sets explained above. The transition matrix of this state encodes the rule that after the first game and after two successive games, a break appears. The hidden state that represents a break is in SMs a production state and corresponds to a break scene. For the HMMs, it is still an internal state and is further analyzed as in Fig. 4.3(b).

- *Level 3: Games.* The internal state of games is analyzed as a series of points. Its transition matrix and its final state distribution encode the fact that a game must contain at least 4 points and that a player must have two more points than the opponent to win the game.

- *Level 4: Points.* Each point is analyzed as containing an exchange (that is, either 'First Missed Serve and Exchange' or 'Exchange'), optionally followed by a rediffusion. Its child nodes are production states in SMs or internal states in HMMs, in which case they are analyzed as in Fig. 4.3(b).

For the case of SMs, this topology contains in total 20 internal states (15 in level 4, 3 in level 3, and 1 in levels 2 and 1) and 47 production states (45 in level 5, and 1 in levels 2 and 3). As the production states of SMs are still internal ones in the HMM model, the topology then contains 67 internal states and also a number of 141 production states. Note that each internal state defines a $N \times N$ transition matrix for its children, a $N \times 1$ initial state distribution and a $N \times 1$ final state distribution[2], where N is the number of its children.

[2] A final state distribution, similar to the initial one, is used instead of defining a final internal hidden state as in the original definition of HHMMs.

5.1.4 Parameter Estimation

Regarding firstly the internal states, their transition, initial, and final state distribution probabilities were manually set according to Fig. 5.2. More precisely, non-allowed transitions were penalized with zero probability and then the remaining ones were arbitrarily set to 1. The option to estimate them through labeled data was rejected as their number is still large, leading to situations of overfitting.

Regarding the observation distribution of the production states, their parameters were *tied* according to which type of leaf node they belong to. So, the frame-based (for the HMM) and the segmental distributions (for the SM) of sections 4.3.3 and 4.4.4, respectively, are shared across the production states of the hierarchical topology and no further learning is required. Finally, and especially for the HMM, the inter-scene transition probabilities were also tied and shared across the dotted internal states of Fig. 5.2.

5.1.5 Decoding

Viterbi decoding with the hierarchical topology involves labeling each shot (in the case of HMMs) or each scene (in the case of SMs) with the correct production state label. Having these labels, it is straightforward to fully reconstruct the hierarchical structure of the tennis match. The decoding algorithm itself does not change but it can become intractable as the number of hidden paths to explore is much larger than with the flat models. A number of implementation precautions have to be taken in order to keep the computational cost in a acceptable level.

Starting with the HMM, a great deal of the N^2T possible paths (with $N = 141$ the number of the production states, much greater to the number of states of the flat model) can be a-priori rejected as most of the transition probabilities of the hierarchical topology are in fact zero. To do so and using the graph of Fig. 5.2, all the possible incoming transitions for each production state are detected, as illustrated in Fig. 5.3. This results to only a small number of production states, on average 4.54. This number is clearly negligible compared to 141. Decoding thus can be achieved in $O(NT)$ time.

The same technique of a-priori pruning is applied for the hierarchical SM topology. Furthermore, as the segmental distributions are largely tied, caching techniques can be used to avoid redundant computation. Once, for instance, a segmental score is computed for a production state of type break, then all the others production states of the same type can share this score.

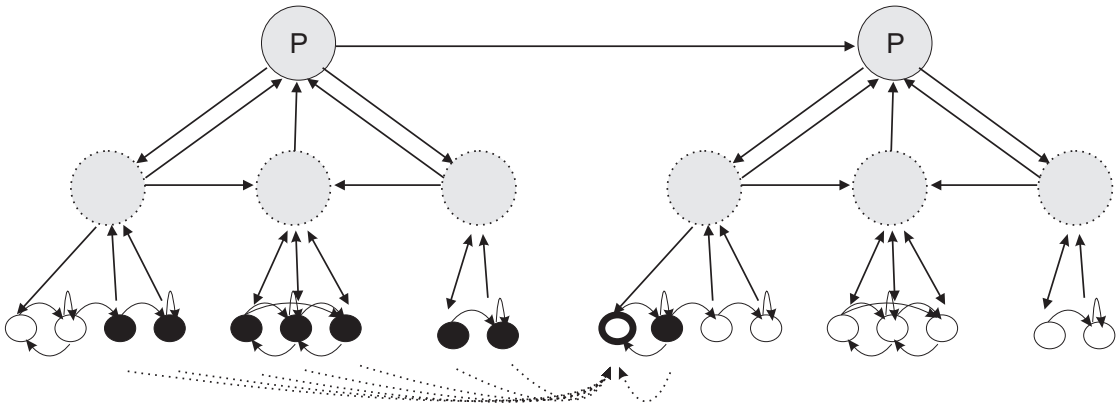


Figure 5.3

Detail of the topology graph for the HMM. The black circles denote the hidden production states from which a transition comes to the boldly circled production state. The detected incoming transitions are summarized with the dotted arrows. They are detected by traversing the graph from each leaf node until a new leaf node is reached. Their probability score is simply the product of the probability scores of each transition traversed. Note that certain traversing rules have to be applied for the transitions to be valid in the sense of HHMMs. Moving from a child to its parent and then again to any of its children is not permitted, unless the parent has a self-transition edge. When reaching a parent not from its children, on the other hand, only its children have to be traversed.

5.2 Integration of the Score Labels

In a tennis broadcast the producer usually announces the changes in the tennis score after an exchange has taken place or a game or set has finished. These announcements take the form of superimposed text on the image frames, as shown in Fig. 5.4. Undoubtedly, these score labels carry on important high-level information that it is interesting to integrate to the models. Firstly, it is a quite reliable source of information on the actual game evolution. When a label appears then it is certain[3] that a game event has occurred. In fact, this source of information is equivalent to consulting a human expert on what happens in the court. Secondly, they provide high-level labeling of the scenes that it is useful to be integrated in the Table of Contents of the video. An end-user, for instance, could be interested in viewing games or sets where a player loses at the beginning and finally succeeds to take the game or the set. This information cannot be extracted from the audiovisual information of the video unless an automatic high-level reasoning module on which player wins each exchange is built. This is extremely difficult with the current computer vision techniques.

[3]The probability of error of the producer is really small. Such an error was not noticed in any of the sequences used in this study.

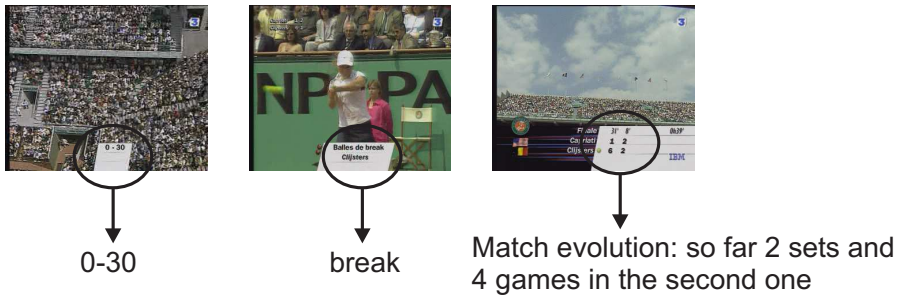


Figure 5.4

Examples of score announcements in a tennis broadcast. Except the score changes after an exchange ('0-30' and 'break' in the first two images), the scoring over games and sets is also provided after the end of a game. These last announcements give important information on the overall match evolution, up to the appearance of the announcement.

The score labels were manually extracted and recognized from the video sequences. Nevertheless, these simulated data are a good approximation of what could have been automatically extracted by a modern Optical Character Recognition tool. Indeed, the relative literature on image/video text extraction is mature enough and there exist numerous approaches with a very good performance (see e.g., [109]). In addition, a great deal of prior knowledge specific to tennis broadcasts can be exploited, rather than attempting to solve the problem in its generality. The score labels appear at fixed positions and are usually framed and well-distinguished from the background to be explicitly visible. False alarms can thus be largely eliminated. Finally, there is a finite set of possible labels, involving digits (like '0-30', '40-15', etc) or some predefined keywords (like 'break' or 'advantage'), which can greatly facilitate the recognition of the labels.

One can think of two possible ways to integrate score label information: posterior integration and integration at the feature level. In a simplified scenario of posterior integration, the score labels are integrated after decoding has taken place and for just labeling the game events. A scene is then labeled according to the score label that is detected inside the scene boundaries. Using a hierarchical topology, the bounds between games or sets can be detected and labeled with the match evolution label that may appear inside these bounds. The problem with this approach is that score labels refer to what actually happened in the tennis court, while the Viterbi solution may (and usually does) fail to recover all the game events with perfect accuracy. This could introduce a certain degree of inconsistency between the labels and the Viterbi solution that would make posterior labeling appearing as illogical to the end-user. In an attempt to overcome this problem of inconsistency, another way of posterior integration could be to perform N-best decoding and then to select the solution that agrees with the score labels. Still, a very large number of candidates should be stored during N-best decoding in order to

find a solution that is actually consistent with the score labels. This would result in prohibiting storage requirements and decoding time.

The integration at the feature level considers the appearance of a score label as an extra descriptor to be added to the feature vectors associated with the shots. As an alternative to the shot-based sampling of this feature, the framework of SMs offers also the possibility of collecting this feature at the scene level. This allows for a certain degree of asynchrony between the relative game event and the actual appearance of the score label. Generally, the score labels can be a useful feature and can improve thus the performance of the system. The semantic content of the labels (i.e., number of game events existing in the video), on the contrary, cannot be explicitly exploited.

A third way of integration, the Score-Oriented Viterbi Search is introduced in this study. It lies in the middle of the above two approaches as it exploits both the appearance of the labels and their semantic content to pilot Viterbi decoding and to ensure that the solution obtained is consistent with the actual match evolution.

Integration at the feature level and the Score-Oriented Viterbi Search are the subject of the remainder of this section.

5.2.1 Score Labels as a Feature

With the HMM framework, the integration of the score labels is straightforward. The vector of the audio-visual shot based descriptors of eq. 4.31 is redefined as:

$$O_t = \begin{bmatrix} \text{visual similarity} \\ \text{shot length} \\ \text{dissolve shot} \\ \text{tennis sound} \\ \text{applause sound} \\ \text{music sound} \\ \text{score appear.} \end{bmatrix} \tag{5.1}$$

where the last descriptor is binary and set to 1 for shots where a score label appeared and to 0 otherwise. Note that a single descriptor is added instead of adding one binary descriptor for each possible label. This last would result to a multitude of descriptors and to noisy measurements. The underlying assumption of this mode of integration is that after an exchange has happened, the producer acknowledges it by displaying a score label and inside the same game action shot. The added feature is thus used to spot exchange shots, i.e., the states 3 and 5 of Fig. 4.3.

A limited asynchrony between the game event and its acknowledgement by the producer can be handled with the SM framework. The presence of a score label can be captured at the scene level, instead of at the shot level. The segmental score of eq. 4.37

is extended as:

$$P(O_{1:l}|l, S_i) = P(O^{\mathrm{v}}_{1:l}|l, S_i)P(O^{\mathrm{a}}_{1:l_{\mathrm{a}}}|l, S_i)P^{l}(o^{\mathrm{score}}|l, S_i) \tag{5.2}$$

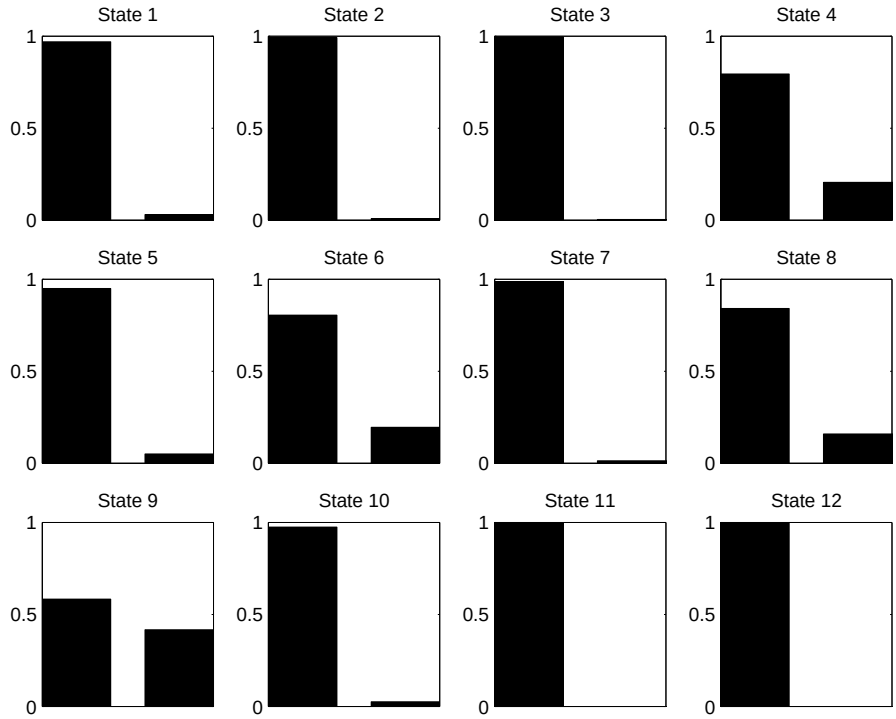
where $P(o^{\mathrm{score}}|l, S_i)$ denotes the probability of presence of a score label inside the scene S_i. This term is raised on the l power in order to provide a segmental score that scales exponentially with the segment length l, as do the other two terms of eq. 5.2. The underlying assumption is now that when an exchange occurs, the producer acknowledges it by displaying a score label some shots after, and within the same scene. The score feature is thus used to spot the 'First Missed Serve and Exchange' and 'Exchange' scenes.

Analysis of the Score Label Feature

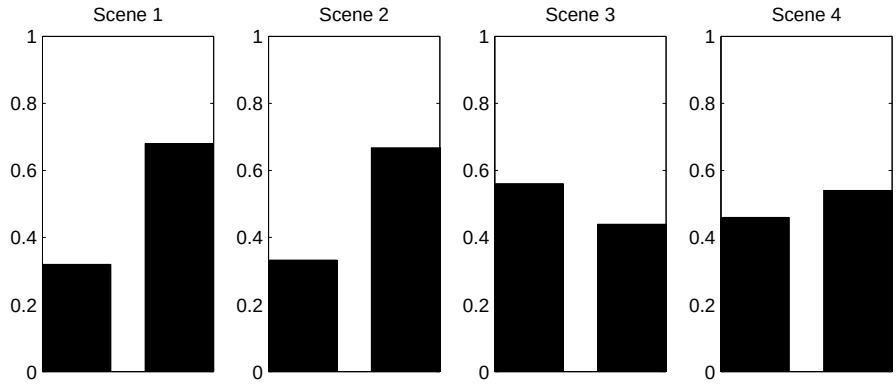
The probability histograms for the extra feature and for each hidden state of the HMM are given in Fig. 5.5(a). One firstly notice that it is rather rare for a score label to appear within a shot that contains an exchange (states 3 and 5). The producer thus acknowledges a game event some shots after it has taken place. Still, these histograms convey useful information as long as they are not uniform across the states. Indeed, the states 4 and 6 (representing shots that immediatelly follow an exchange) can be somehow spotted with this feature as a score label usually appears in them, even with a small probability. The same holds for states 8 and 9 that lie inside a rediffusion scene. This is logical as score labels appear after the relative exchange and before the next one begins. If a rediffusion occurs in this space, then the score label will finally appear inside the non-court views of the rediffusion.

The corresponding probability histograms for SMs, calculated at the scene level this time, are given in Fig. 5.5(b). The appearance of a score label in the first two scenes is visibly more probable than its absence, as expected. These two scenes can thus be spotted by this feature. Score labels, however, also appear in the next two scenes, with more or less equal probability of absence or presence. Note that the histograms of states 8 and 9 in Fig. 5.5(a) are not uniform while the histogram of scene 3 in Fig. 5.5(b) is. The reason for this is simple: a score label appears (or not) in a rediffusion with uniform probability but it is distributed in the shots that lie inside not uniformly. A score label never appears in a dissolve shot (state 7).

Overall, what teach us the histograms of Fig. 5.5(b), is that the producer may display the score labels much after the relative exchange. This asynchrony can be extended beyond the scene boundaries. Another fact about the producer style, which cannot be easily seen in the histograms, is that there exist game events which are not acknowledged at all by the producer. In table 5.1 we see how many exchanges occurred in each video and how many score labels appeared. Depending on the producer's choices, they cover from 68% to 91% of the exchanges. In the match Agassi - Safin, in particular, 4 exchanges in a row were not acknowledged.



(a)



(b)

Figure 5.5

The distributions of appearances of score labels according to (a) shots and (b) scenes, as estimated in the training set. The first bar in each histogram gives the probability (as calculated in the training sequences) of absence of a score label, and the second one the probablity of presence. The state numbers refer to Fig. 4.3 and the scene numbers are as defined in section 4.2.2.

Video	# of exchanges	# of labels	percentage
Agassi - Safin	248	168	67.74%
Pioline - Hewitt	270	247	91.48%
Pioline - Philippoussis	213	192	90.14%
Grosjean - Philippoussis	169	150	88.76%
Sampras - Rusedski	178	145	81.46%
Capriati - Clijsters	259	229	88.42%

Table 5.1

Scoring statistics for each video. The number of exchanges refers to the total number of scoring scenes (i.e., 'First Missed Serve and Exchange' and 'Exchange') of the video. In the next two columns, the total number of score labels appeared and the percentage of the exchanges they cover are reported.

An effective use of the score labels should not make any assumptions on the producer's style and should tolerate extensive asynchrony and undisplayed labels.

5.2.2 Score-Oriented Viterbi Search

The optimal use of the score labels is to constrain the Viterbi search space to the paths that are consistent with the labels and thus with the actual game evolution. Instead of posterior use of the labels after an N-best decoding, Score-Oriented Viterbi Search uses score information during decoding.

Game Events and Score Labels Ordering

Before proceeding to the description of the algorithm, it is reminded that a score label appears after the corresponding game event has happened and also before the next game event. A typical setting is given in Fig. 5.6, where we see three score labels and their corresponding exchanges. The score label at the shot t_2 acknowledges an exchange that occurred at a time $t \leq t_2$. This exchange however cannot precede also the previous score label at time t_1 because this would imply that the score label at t_2 refers to no game event. At the very limits, the exchange may occur at time t_1, having in the same shot firstly the appearance of a label and then a new exchange. The relative exchange of the score label '15-15' must lie thus inside the space $t \in [t_1, t_2]$. The complete scene that contains this exchange, in its turn, will end somewhere in the space $t \in [t_1, t_3]$. Note that it can finish before t_2 as a rediffusion or a break may appear before the next exchange.

The key idea is to perform a local Viterbi forward pass between t_1 and t_3 with an N-best-like scheme. Then all the paths in this space that are inconsistent with the score indication are penalized and will not be preferred by the final Viterbi solution. In the setting of Fig. 5.6, one scoring event must have happened between the labels '15-0' and

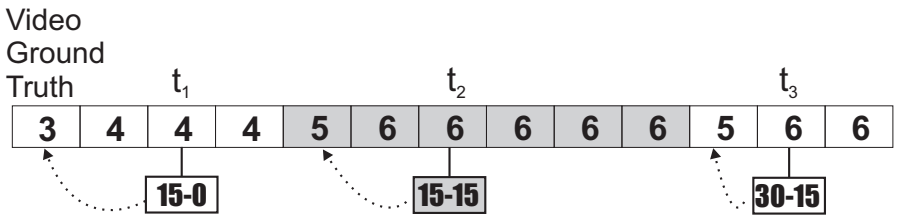


Figure 5.6

Game events and appearance of the score labels. The scoring game events are the exchanges, which correspond to hidden states 3 and 5 for the HMM or to the scenes 'First Missed Serve and Exchange' and 'Exchange' for the SM. The scene to which label '15-15' refers generally starts after the previous label and ends before the next one.

'15-15'. The paths thus in the space $t \in [t_1, t_3]$ that contain no scoring event or more than one will be penalized. The action of the local forward pass and the penalization of the inconsistent local paths will be referred to as a *local search* througout this text. After the local search for the label '15-15', the algorithm proceeds similarly for the next label '30-15'. The new local search shares however the queues of the previous local search in the space $[t_2, t_3]$. The algorithm proceeds thus as a *pipeline* of local searches where remaining paths are further developed until the end of the video. The backtracking of the Viterbi decoding is then performed and the solution thus obtained is consistent with all of the score labels.

The algorithm guaranties optimality as the queues of the local searches do not store in fact the N-best paths but the best path for each possible number of scored points between two score labels. The maximum number of scored points was set to 5 as in the corpus of this study a maximum number of 4 consecutive exchanges was left unacknowledged.

Algorithm Definition

The algorithm applies to both HMMs and SMs. No extra parameters for the models are needed, only decoding changes. The algorithm is firstly described for the case of HMMs whose decoding is simpler and then differences for SMs are discussed.

Given a video sequence $O_{1:T}$ of T shots, M score labels $l_1, l_2 \dots l_M$ appearing at shots $t(l_1), t(l_2) \dots t(l_M)$ are extracted and recognized. The number of points scored between a label and its predecessor (for example when the label is '30-30' and the previous one is '15-15', then two points have been scored) are given as $s(l_1), s(l_2) \dots s(l_M)$. For each time step $1 \dots T$ and for each hidden state $1 \dots N$, two queues $Q_{t,i}$ and $Q'_{t,i}$ are defined. The size of each of them is $S = 5$, the maximum number of points allowed between two labels. The first queue holds results of the current local search, while the second one stores results of the previous local search and acts like a bridge between successive local searches. Each queue stores at first the likelihood $\delta_{t,s}(i)$ of the best path ending in

hidden state i at time t and with number of points s. This last quantity is simply given as the number of instances of the hidden states 3 and 5 in the path. The queue Q defines $\delta_{t,s,Q}(i)$ and the queue Q' defines $\delta_{t,s,Q'}(i)$. Each queue needs also to store backtracking information, which in our case is the label of the previous hidden state in the path and also its corresponding queue entry.

The algorithm proceeds as follows:

1. *Initialization.* For each $t \in [1, T]$, $i \in [1, N]$, $s \in [1, S]$, $q \in Q \cup Q'$:

$$\delta_{t,s,q}(i) = -\infty \tag{5.3}$$

2. *Pipeline of local searches.* For each label l_k, $k \in [1, M]$, three steps are executed (see Fig. 5.7 for an illustration):

   - a Viterbi forward pass is performed in the space $t \in [t(l_{k-1}), t(l_{k+1})]$:

$$\delta_{t,s,Q}(i) = \max_{j,q,\sigma | s=\sigma(q)+n_{ji}} \delta_{t-1,\sigma,q}(j) a_{ji} b_i(O_t) \tag{5.4}$$

     where $q \in Q \cup Q'$ when $t \in [t(l_{k-1}), t(l_k)]$ and $q \in Q$ when $t \in [t(l_k), t(l_{k+1})]$, $j \in [1, N]$ and $\sigma \in [1, S]$. The results of the local search are stored in the queue Q. As each entry in the queue holds the best path that results to a given score s, the values of j, q, and σ are constrained so that the path obtained will result to the score s. If $q \in Q'$, then $\sigma(q) = 0$ as scores developed previously are not taken into consideration by the current score label. If $q \in Q$, then $\sigma(q) = \sigma$, as expected. The term n_{ji} is a binary indicator denoting transition that corresponds to an exchange. It is 1 when i is 3 or 5 and zero otherwise. Finally, the terms a_{ji} and $b_i(O_t)$ denote transition and observation probabilities, as in standard Viterbi decoding.

   - All the paths that result to a score which is not equal to $s(l_k)$ are penalized:

$$\delta_{t,s,Q}(i) = \delta_{t,s,Q}(i) + P(s, s(l_k)), \quad t \in [t(l_{k-1}), t(l_{k+1})], \quad s \in [1, S] \tag{5.5}$$

     where $P(s, s(l_k)) = -\infty$ if $s \neq s(l_k)$ and zero otherwise.

   - The contents of the queue Q are transfered to Q' for the needs of the upcoming local search:

$$\delta_{t,s,Q'}(i) = \delta_{t,s,Q}(i), \quad t \in [t(l_k), t(l_{k+1})], \quad s \in [1, S] \tag{5.6}$$

     The contents of Q for the above time instants are reset as in the initialization stage.

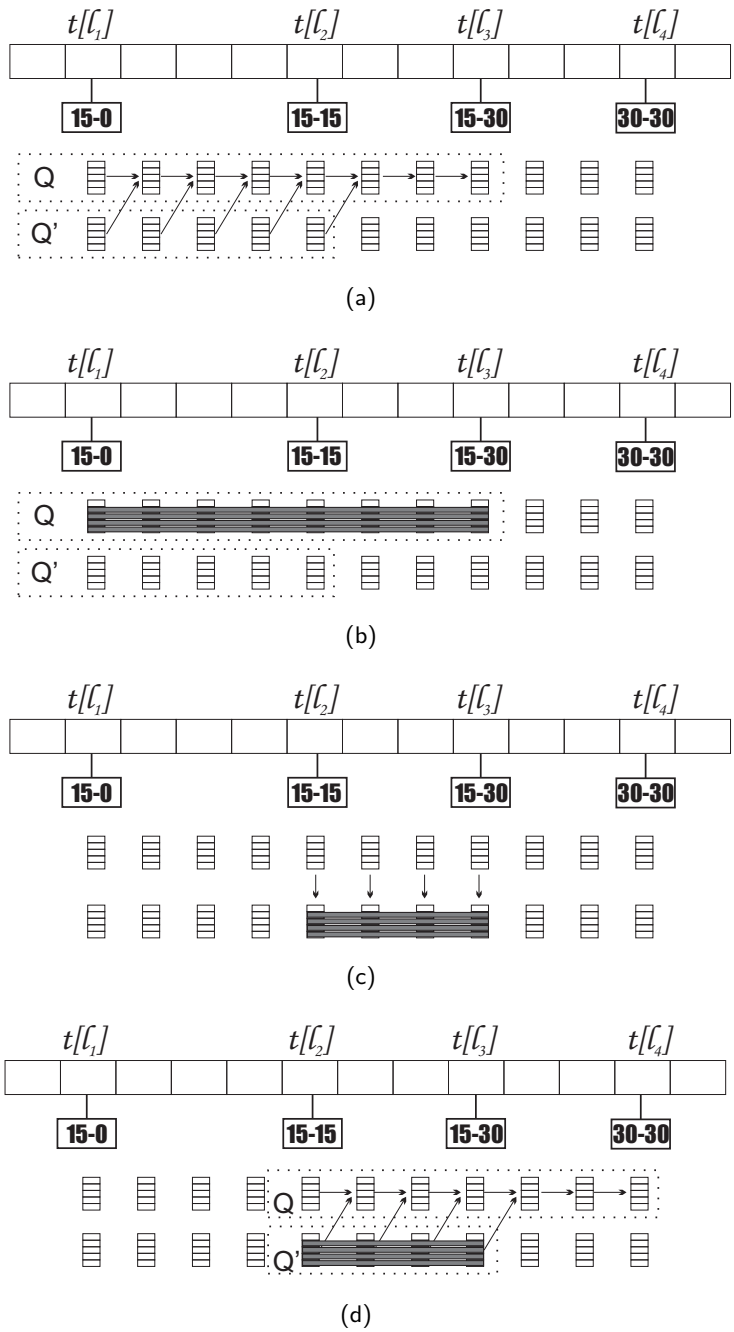


Figure 5.7

Execution steps involved in a local search. In (a), a Viterbi forward pass is performed using both queues Q and Q' for looking up and storing results in Q. Inconsistent paths are penalized in (b). In (c), the results of the local search are transferred to Q' for the needs of the next local search, performed in (d).

Note that backtracking information has to be appropriately kept, especially when queue contents are transfered.

3. *Backtracking.* Having performed the local searches for all the score labels, backtracking is then performed much like in standard Viterbi (eq. 4.16 and 4.17). The recovered hidden state sequence $Q^*_{1:T}$ is the most likely one according to the HMM parameters and also fully consistent with the score labels. In addition to recovering the most likely hidden state sequence one may also need to find out the correspondences between the hidden states and the labels. These correspondences can then be used in the construction of the table of contents of the video. They are simply provided by the backtracked position of the hidden state in the queues: if it is found to belong in the queue Q, then it corresponds to the following score label in the video. If it is found to be in the queue Q', then it corresponds to the previous in time score label.

Interestingly enough, when $P(s, s(l_k)) = 0$ for every s (i.e., when no path is penalized), then the Score-Oriented Viterbi Search will provide exactly the same solution as the standard version of Viterbi decoding.

The computational cost of the Score-Oriented Viterbi Search for HMMs is estimated as follows. The computational cost of a local search (similar to a Viterbi operating with queues of $2S$ size) is roughly $(2SN) \times (N) \times (T_{l_k}) = 2SN^2T_{l_k}$, where $T_{l_k} = t(l_{k+1}) - t(l_{k-1})$ is the length of the time window of the local search for the label l_k. The total computational cost is the sum over all labels:

$$\sum_{k=1}^{M} 2SN^2T_{l_k} = 2SN^2 \sum_{k=1}^{M} T_{l_k} = 4SN^2T \tag{5.7}$$

where the total length of all time windows is simply two times the length T (in shots) of the video. The cost is thus of order of $O(4SN^2T)$, which means that it scales the cost of standard Viterbi decoding by a constant factor $4S$. As generally S receives moderate values, the Score-Oriented Viterbi Search does not require a prohibitive computational load.

The scheme of the algorithm is the same for SMs. As Viterbi decoding for SMs involves searching through different segmentation hypotheses, the local searches and the backtracking have to be modified accordingly, as explained in the discussion for SMs (section 4.1.3). More precisely, the local search of eq. 5.4 changes to:

$$\delta_{t,s,Q}(i) = \max_{j,\tau,q,\sigma | s=\sigma(q)+n_{ji}} \delta_{t-\tau,\sigma,q}(j) a_{ji} p(O_{t-\tau+1:t}|\tau, i) p(\tau|i) \tag{5.8}$$

$$t(l_{k-1}) \leq t \leq t(l_{k+1}) \tag{5.9}$$

$$t(l_{k-1}) \leq \tau \leq t - 1 \tag{5.10}$$

and n_{ji} now detects score changes between scenes (and not shots, as in HMMs). It is set to 1 if i equals to 1 or 2 and zero otherwise. Looking at the range of values that τ can take in eq. 5.10, we notice an interesting side effect of the Score-Oriented Viterbi Search on SMs: the set of possible segmentation hypotheses is drastically reduced. In the standard Viterbi decoding for SMs, every possible scene with at most $L_{\max} = 80$ shots must be evaluated. With this new decoding algorithm, every segmentation inside the time windows T_{l_k} must be evaluated, instead. So, the Score-Oriented Viterbi Search may provide a different solution to the one of the standard Viterbi due to the fact that the seach space is modified, even when no path is penalized. It is expected to be already a better solution as the score labels give in fact some better approximations on the scene boundaries than blind search.

The computational cost of Score-Oriented Viterbi Search for SMs is approximated as follows. Supposing that the average length of each time window is $\hat{T}_l$, the cost of each local search is $O(2S\hat{T}_l N^2 T_l)$, analogous to the cost of the standard Viterbi $O(L_{\max} N^2 T)$, but with using queues of size $2S$. The overall cost will thus be $O(4S\hat{T}_l N^2 T)$. $\hat{T}_l$ generally differs from video to video, being smaller the more score labels the video contains. In the videos used in this study, its value is around 20.

Handling of Uncertainty

The number of scored points between two labels l_{k-1} and l_k was previously considered to be well-defined given the content of the labels. This can be true when the score labels use an incremental scoring scheme, like when switching from '15-0' to '15-15' (1 point) or from '15-0' to '40-30' (3 points). This scoring scheme however changes by using some labels like 'break', 'advantage', or 'ball of set' to follow the peculiarities of the tennis terminology. This scoring scheme in combination with the fact that some labels may not be displayed results in some cases to uncertainty on the number of scored points between two labels. When switching for example from the label 'break' again to label 'break', this could imply 2 points (with one label 'advantage' left undisplayed), 4 points (with 3 labels 'advantage', 'break', 'advantage' left undisplayed in a row), 6 points, and so on. An odd number of points however can never appear. The idea is thus to learn the possible number of points scored for each label transition and to use a varying penalty $P(s, l_{k-1}, l_k)$ defined accordingly or learned from data.

Another source of uncertainty is errors that may occur during the recognition of the score labels. The use of a penalty learned from data can also handle this situation. When a lot of errors occur, the penalties will then learn to be more uniform between the values of s and not penalize with high penalties. In the extreme case, when all the labels are wrongly recognized, the penalties will be perfectly uniform so that the Score-Oriented Viterbi Search is reduced to the standard Viterbi decoding. Errors during the extraction

(or detection) itself of the labels were not taken into consideration as score labels appear in fixed positions and are well-framed that a false detection is practically impossible.

In order to account for uncertainty, the penalties $P(s, l_1, l_2)$ were estimated as a function of the probability that the label transition l_1 to l_2 contains s scored points:

$$P(s, l_1, l_2) = A\Big(1 - \frac{N(s, l_1, l_2)}{\sum_{s=1}^{S} N(s, l_1, l_2)}\Big) \tag{5.11}$$

where $N(s, l_1, l_2)$ is the number of times that s points are scored between the labels l_1 and l_2, estimated on the training set. The value of the constant A, which should generally take large negative values, was experimentally set to -10. According thus to eq. 5.11, the penalty $P(s, l_1, l_2)$ will be close to zero when the labels l_1 and l_2 usually correspond to s points, while it will take large negative values on the contrary. If the recognition of the labels introduce a lot of errors then $N(s, l_1, l_2)$ will tend to take the same value for all s and in the end the penalties will take values around $A(1 - 1/S)$, independent of s.

Using Hierarchical Topologies

So far the labels that convey the overall match score in sets and games (third image in Fig. 5.4) were considered as a generic label 'end of game', leaving in fact unexploited the information on the number of games and sets. The reason is that the flat models used so far can keep track of the number of exchanges traversed but not of the number of games or sets. To do so, the hierarchical topology discussed in section 5.1 should be used. In fact, the Score-Oriented Viterbi Search comes as a further and decisive support for the kinds of constraints that the hierarchical topology imposes to Viterbi decoding. This topology ensures that the solution obtained is consistent with the tennis rules on the hierarchical decomposition of the match but quite likely will not provide a solution which agrees on the number of sets or games per set *actually* scored. Combining the Score-Oriented Viterbi Search with a hierarchical topology has the unique advantage of finding a solution that is consistent with both the tennis rules and the actual game evolution.

The Score-Oriented Viterbi Search changes only when a match score label l_k appears. Firstly, what was described for flat models applies exactly the same also with the hierarchical topology. The match score label continues to be considered as an 'end of game' label and the paths are penalized with respect to the number of scored points they contain. In addition to this, all the paths in the space $[t(l_{k-1}), t(l_{k+1})]$ are penalized according to the number of games and sets they contain in total. These are calculated as decoding evolves by using the incoming transitions of each production state (section 5.1.5). These connections can carry information whether switching from a given production state to another one is equivalent of advancing one game or one set. So, for each

path, in addition to its cumulative likelihood and backtracking information, the number of games and sets that it contains are stored.

5.3 Experimental Results

For the experimental evaluation of the various models and algorithms discussed in this chapter, the same measurements C, P, R, and $\hat{F}$ as in section 4.5 were used. Results are reported in table 5.2.

In the first line of the table, the performance of the systems 'HMMs-VA' and 'SMs-VhmmA2gram' is copied from table 4.2. They serve as a frame of reference for the comparison with the systems of this chapter. Results for the hierarchic topologies are reported in the second row of the table. A light performance degradation is noticed for both HMMs and SMs when switching from flat to hierarchic topologies. A possible explanation for this is that the probability scores of the allowed hierarchic transitions were manually set to arbitrary values. The intra-scene probability scores are thus lost, while the flat models learn them from data. Consider for simplicity the case of SMs, where the hierarchy ends to the scenes. When removing the inter-scene transition probabilities during Viterbi decoding (but keeping the duration ones), the results are then: $C = 81.55$, $P = 85.54$, $R = 75.90$, $\hat{F} = 65.37$. They are thus a little inferior to what the hierarchic SM yields.

In the third entry of table 5.2, we see the performance of the ergodic systems when the score labels are used as additional features. HMMs demonstrate a small performance improvement, mainly due to spotting of some hidden states where a score label is likely to appear. A performance improvement is also noticed for SMs, although being marginal. This can be explained by the fact that the probability histograms of the score feature, given in Fig. 5.5, are more uniform at the scene level than at the shot level. The extended asynchrony that exists between the game event and the score label causes a more or less random distribution of the score feature across the scenes and thus it is harder to exploit it in terms of performance.

Results for the proposed Score-Oriented Viterbi Search algorithm are reported in the next two rows of table 5.2. Clearly, in both HMMs and SMS and with both ergodic or hierarchical topologies, the constraints of the scores helped the system and improved the performance. The hierarchical topology still cannot outperform the ergodic topology. But the gap between them is clearly reduced due to the additional game structure constraints imposed by the score labels. It can also be noted that SMs gain a lot more than HMMs (+5.03% versus +1.89% in $\hat{F}$ measurements). This may be explained by the fact that the positions themselves of the score labels provide some rough approximations of the scene boundaries, giving some extra valuable information for the Viterbi decoding in SMs. When no penalties are used, the Score-Oriented Viterbi Search gives

	HMM	SM
Baseline systems		
Ergodic	80.23 84.69 79.70 66.42	81.77 84.10 79.45 66.81
Hierarchic	79.28 84.96 77.96 65.04	81.25 85.63 77.24 66.04
Score label as feature	80.81 85.75 80.37 67.66	81.96 84.19 79.70 67.11
Score-Oriented Viterbi Search		
Ergodic	82.17 83.40 82.39 68.31	85.97 84.90 83.43 71.84
Hierarchic	82.67 84.30 80.54 68.03	85.80 85.15 82.89 71.57
Level of Uncertainty		
No uncertainty	82.03 83.63 82.65 68.50	85.99 85.19 83.30 71.94
90% simulated noise	81.59 83.88 82.22 68.16	85.60 85.53 82.75 71.59
50% simulated noise	81.15 84.37 80.67 67.31	84.15 87.02 80.16 70.07

Table 5.2

Experimental results for hierarchical topologies and score label integration. For each approach, the performances of the HMM and of the SM are reported. The three numbers at the top row correspond to the measurements C, P, R, and the one at the bottom row is $\hat{F}$.

Video	SMs-VhmmA2gram	Score-Oriented	No uncertainty
Agassi - Safin	231	245	248
Pioline - Hewitt	323	269	270
Pioline - Philippoussis	240	213	213
Grosjean - Philippoussis	181	169	169
Sampras - Rusedski	163	176	178
Capriati - Clijsters	250	256	259

Table 5.3

The number of exchanges that exist in the hidden state sequence after decoding each video sequence. The second column reports outcomes of the standard Viterbi decoding and for the 'SMs-VhmmA2gram'. The next two columns report results of the Score-Oriented Viterbi Search, with or without uncertainty due to the tennis scoring scheme. The difference between these last two is negligible, while the last column is in total conistency with table 5.1.

a performance of $C = 81.71$, $P = 85.15$, $R = 79.16$, $\hat{F} = 67.16$ (while for HMMs does not change, as explained in the description of the algorithm).

So far, the penalties of the Score-Oriented Viterbi Search were calculated according to eq. 5.11 and on manually extracted and recognized labels. The only source of uncertainty is thus due to the scoring scheme of tennis combined with undisplayed labels. In the last three rows, we see the performance of Score-Oriented Viterbi Search with ergodic models and with varying level of uncertainty for the calculation of the penalties. Firstly we see the performance when no uncertainty exists for purely demonstrating purposes. In this scenario the number of scored points between each score label is manually provided by the ground truth. The penalties are thus either $-\infty$ or 0. The constrained Viterbi decoding resulted into a hidden state sequence that contains exactly the same number of exchanges as the ground truth and for all the videos. When the ground truth is not used and there exists uncertainty, the number of exchanges recovered is still close to what the ground truth reports, however. The number of recovered exchanges for every video is summarized in table 5.3. Generally, a small difference in the number of recovered exchanges and also in terms of performance is noticed when no uncertainty is simulated. The system thus demonstrates an excellent degree of robustness to undisplayed labels.

In the last two rows of 5.2, we see the performance degradation of the Score-Oriented Viterbi Search (ergodic structure) when a given percentage of the score indications is artificially misrecognized and after re-estimating the penalties of eq. 5.11. Simulated noise will thereby result into more uniform penalties and finally the performance nicely degrades to the one of the standard Viterbi decoding.

The performance obtained with the ergodic 'SMs-VhmmA2gram', decoded with the Score-Oriented Viterbi Search, is the best achieved on automatically extracted feature set. When the 'SMs-(AV)hmmA2gram' approach (which yielded the best performance in

SMs-VhmmA2gram				
	1	2	3	4
1	81.8	15.5	2.5	0.2
2	23.0	75.0	1.0	1.1
3	8.0	5.6	85.9	0.4
4	2.2	5.6	4.9	87.2

Score-Oriented				
	1	2	3	4
1	86.2	10.1	2.7	1.0
2	21.1	77.4	1.0	0.5
3	7.1	1.2	91.3	0.4
4	2.5	2.2	3.3	92.1

Table 5.4
Confusion matrices on the classification of shots with/without the Score-Oriented Viterbi Search.

section 4.5) is used instead, the respective performance results in $\hat{F} = 71.95$ ($C = 88.62$, $P = 85.01$, $R = 82.90$), which is very close.

Finally, one may find awkward that even when the Viterbi solution is fully consistent with the number of exchanges scored and the actual game evolution, still the performance measurements continue to record a considerable amount of error. This is explained by the fact that the score labels cannot resolve any confusion between the scenes 'First Missed Serve and Exchange' and 'Exchange', as both of them contain an exchange and are equivalent in terms of tennis score. This is clear in table 5.4 where the confusion matrices with and without the Score-Oriented Viterbi Search are reported. We see that the confusion between the above mentioned scenes (scenes 1 and 2) is not reduced as much as in table 4.3. There, the inclusion of the auditory features (and especially of the tennis sound) helped a lot in disctriminating between these two scenes.

CHAPTER 6

A Segment Model-Recurrent Neural Network Hybrid

In chapter 4 the segmental scores were computed by HMMs or bigram models. In this chapter, the idea of using Recurrent Neural Networks (RNNs) to compute a segmental score is explored. The idea of fusing HMMs and neural networks, resulting to HMM-NN hybrids, is not new. Neural networks are discriminative models and are expected to provide better performance compared to Gaussian Mixtures or discrete densities that are widely used in HMMs. The SM-RNN hybrid that we will examine in this chapter is the direct extension of the HMM-NN hybrid on segmental features.

The disadvantage of RNNs is that the Back-Propagation Through Time (BPTT) and Real-Time Recurrent Learning algorithms that are developed to train them cannot efficiently propagate the gradient through long time lags and thus cannot process effectively input sequences of moderate or large size. A newly created recurrent connectionist model, called Long Short-Term Memory, which is especially designed to overcome this problem is explored and compared to BPTT.

The focus of this chapter is the comparison between different approaches and hybrids and not the multimodal integration itself. As feature set, the synchronous audiovisual shot-based descriptors is used. Before proceeding to the details of the hybrids, basic concepts of multilayer Perceptrons, of the BPTT algorithm and of the Long Short-Term Memory are provided.

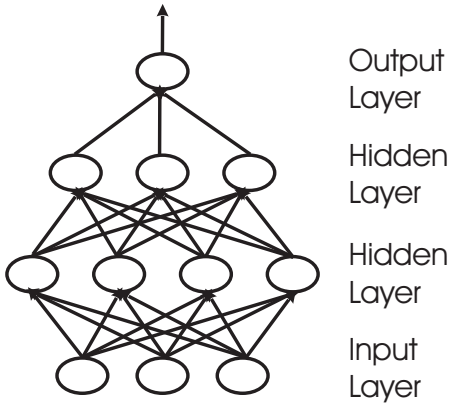


Figure 6.1
A typical multilayer feed-forward neural topology.

6.1 Feedforward and Recurrent Multilayer Perceptrons

6.1.1 Backpropagation on Feedforward Networks

The error back-propagation algorithm (or simply *backpropagation*) is a gradient-based learning rule that operates on multilayer feed-forward neural topologies. An example of such a topology is given in Fig. 6.1. The network is organised in multiple layers, each of them having a certain number of nodes or *neurons*. The input of the network is provided by the input layer. Then, the signal is propagated through the connections of the network and processed by the hidden layers until reaching the output layer, where the actual response of the network to the input is calculated. There is no recursion in the network, i.e., the signal is propagated strictly from bottom to top. Having a set of pairs of input-output mappings, we train the network in a supervised manner in order to *interpolate* between these input-output mappings. The training procedure consists of adjusting the free parameters of the network to produce the desired behavior. These free parameters are the *weights* of the network, which are scalars that characterize the strength of every connection of it.

The backpropagation algorithm is an extension of the Perceptron Rule [83], developed to train single-layer perceptrons, to multilayer topologies (hence the alternative name "MultiLayer Perceptrons", or MLPs). An MLP can learn arbitrarily complex class separation boundaries due to the computational power of its hidden layers, provided that a sufficient number of training patterns is used. Backpropagation was originally introduced by Rumelhart *et al.* [87] and was also discovered independently and from a different perspective by LeCun [52]. For a more detailed and comprehensive coverage of the algorithm, the reader is referred to [37].

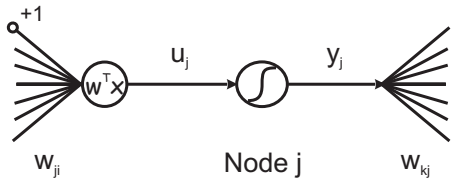


Figure 6.2
A typical node in an MLP network. After the calculation of the dot product of its weights with the input, the signal is passed through a sigmoid and finally is propagated to the nodes of the next layer.

The operation of the backpropagation algorithm is seperated into two phases, the *forward* and the *backward* pass. In the former, the signal is propagated bottom to top to produce the output of the network. It is equivalent to a pure stimulation of the network. The operation of a neuron j that receives an input vector $\mathbf{x}$ from the previous layer or from the input layer is described by the following equations (see Fig. 6.2):

$$u_j = \sum_i w_{ji} x_i \tag{6.1}$$

$$y_j = f(u_j) \tag{6.2}$$

$$f(x) = \begin{cases} (1+\exp(-\alpha x))^{-1} \\ \text{or} \\ \tanh(-\alpha x) \end{cases} \tag{6.3}$$

where w_{ji} notates the connection between the i node of the previous layer and the j node, $x_o = +1$ constant and w_{jo} is the bias of this neuron. Having computed the dot product between the input and weight vector, the outcome is passed through a sigmoid function f, usually being one of the two alternatives of equation 6.3. The parameter α controls their slope. Finally, the output y_j is propagated to the next layer and the procedure goes on until reaching the output layer.

The operation of the backward pass is to adjust the weights of the network according to the gradient descent formula:

$$\Delta w = -\rho \frac{\partial E}{\partial w} \tag{6.4}$$

where ρ is the learning rate and E is the error function over the training set. The equations that describe the backward phase are defined as follows:

$$E = \frac{1}{2}\sum_o (d_o - y_o)^2 \tag{6.5}$$

$$e_o = d_o - y_o \tag{6.6}$$

$$\delta_o = -\frac{\partial E}{\partial u_o} = f'(u_o) e_o \tag{6.7}$$

$$\Delta w_{oi} = \rho \delta_o y_i \tag{6.8}$$

$$\delta_j = -\frac{\partial E}{\partial u_j} = f'(u_j)\sum_k \delta_k w_{kj} \tag{6.9}$$

$$\Delta w_{ji} = \rho \delta_j y_i \tag{6.10}$$

where the index o denotes output node, d_o is the desired response of this output node and the w_{kj} is the connection strength between the hidden node j and the node k of the following layer. Following [37], the quantity δ_k will be called *local gradient* for the neuron k. The algorithm proceeds with the presentation of every training pattern to the network, followed by the adjustment Δw until the error E is minimized to a satisfactory level, i.e., convergence is achieved. A necessary condition for backpropagation to work is the random initialization of all the weights of the network. This creates asymetry in the hidden layer that, as the algorithm goes on, will finally allow for the extraction of different and useful features from the input.

6.1.2 Backpropagation Through Time

The backpropagation algorithm as described previously raises the question of how can we process data of sequential nature with an MLP. The input then would be a complete sequence of vectors $\mathbf{x}_{1:T}$ instead of a single one $\mathbf{x}$, as it was supposed in the previous section. Of the first approaches to apply MLPs to such problems was the use of time delays at the input layer [92, 51]: the input window of the MLP receives more than one consecutive input vectors at once. The network is thus enhanced with some short-term memory capabilities, defined by the (fixed) length of the window. The advantage of this approach is that the MLP structure and learning algorithm do not change, making its application straightforward. When however the relative features in the input sequence do not lie inside the input window of the MLP or, in other words, long-term memory capabilities are required, then time-delaying the input will clearly fail.

The idea or recurrent topologies is to use time delays inside the network. A node then receives as input not only the output of the previous layers of the network as in feedforward topologies, but also the output of the same or above layers, delayed in time as shown in Fig. 6.3. Through these recurrent connections the network has to learn to extract and store relative features of the input sequence and to release them in a later time, when the output nodes need them. To account for the complicated dynamics of recurrent MLPs, two algorithms have been proposed in the relative literature, namely the BackPropagation Through Time (BPTT) and Real-Time Recurrent Learning (RTRL) [107, 37]. The two algorithms have been experimentally proved to be equivalent in terms of learning capacities and in this study we will focus on BPTT due to its simplicity.

Let us suppose that an input sequence $\mathbf{x}_{1:T}$ is to be processed by a recurrent network. The operation of BPTT starts by unfolding the network through time in order to construct a multilayer topology of T layers with the connections between them defined

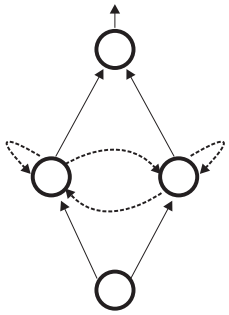


Figure 6.3

A simple recurrent topology. Dotted lines represent time delayed connections. The exact nodes and layers that are connected with recurrent connections are left as an engineering choice, as much the topology of the recurrent MLP. Recurrent connections may be present between different layers too, for instance.

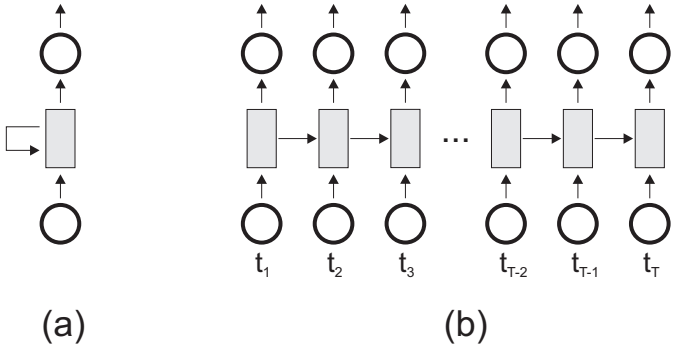


Figure 6.4

The operation of unfolding through time a recurrent MLP. In (a), a recurrent network is depicted with one or more hidden layers represented by the square and with its input and output nodes represented by two circles. The activation of this network with the input sequence is equivalent to the activation of the unfolded network in (b).

by the recurrent connections of the initial network. The operation of network unfolding is depicted in Fig. 6.4. The unfolded network is then a legitimate feedforward MLP that contains exactly the same weights as the recurrent network and, in effect, shares them across its T layers. During the activation of the unfolded network, incoming inputs and activation outputs for each node and for each time instant have to be stored.

The objective function to be minimized by BPTT is the sum of squared differences between the target response and the network's output

$$E^{\text{total}} = \sum_{t=1}^{T} \frac{1}{2} \sum_{o} (d_o(t) - y_o(t))^2 \tag{6.11}$$

over all time steps $1 \ldots T$ or whenever a desired response is defined. The backward pass

of BPTT starts at time $t = T$ and ends at $t = 1$, just as in feedforward MLPs the backward pass starts from the output until the input layer. In fact, BPTT follows literally the equations of backpropagation; only some special care must be taken regarding the unfolded structure of the network. The local gradients $\delta_j(t)$ (given in eq. 6.9 for standard MLPs) of all the nodes j and for each time instant t are calculated recursively as:

$$\delta_j(t) = f'(u_j(t)) \left(\sum_{k \in \mathcal{F}_j} \delta_k(t) w_{kj} + \sum_{k \in \mathcal{R}_j} \delta_k(t+1) w_{kj} \right) \tag{6.12}$$

where $\mathcal{F}_j$ denotes feedforward connections from node j to any node k and $\mathcal{R}_j$ denotes recurrent connections from node j to any node k, with k defined by the network topology. Finally, after the end of the backward pass, the weights are adjusted in a batch mode:

$$\Delta w_{ji} = \rho \sum_{t=1}^{T} \delta_j(t) x_i(t) \tag{6.13}$$

where $x_i(t) = y_i(t)$ is the activation of the node i at time t if the weight w_{ji} belongs to a feedforward connection, or $x_i(t) = y_i(t-1)$ if the connection is delayed (included recursion). This formula for Δw_{ji} makes explicit the fact that the weight w_{ji} is shared across multiple instances through time of the same recurrent MLP.

Limitations of Backpropagation Through Time

Let us suppose that an error signal is inflicted at the end of sequence in Fig. 6.4 and that the relative feature to be extracted from the input sequence lies at the beginning. It is easily noticed that the strength of the propagated signal will decay exponentially before reaching the beginning of the sequence due to the multiplication with the weights and the sigmoid derivatives in eq. 6.12. This exponential decay can be avoided only by using weights initialized with values greater than 1.0, which would, however, easily lead the algorithm to instability.

The fact of the exponential decay of the gradient means that it is extremely difficult to learn long-term dependencies with BPTT. In fact, BPTT is in practice equivalent to time delayed MLPs in terms of learning capacity. For a more concise analysis of this intuitive outcome, the interested reader is referred to [38].

6.2 Long Short-Term Memory

The Long Short-Term Memory (LSTM) recurrent MLP architecture was introduced by Schmidhuber *et al.* [39, 31] in order to provide remedies for the problem of error signal decay of BPTT. Key architectural ideas of LSTM are firstly discussed and then a formal description of the method is given.

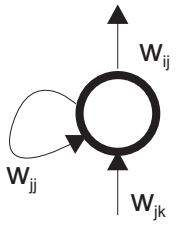


Figure 6.5
A simple node j connected to a node k from the previous layer, to a node i from the above layer and recurrently to itself.

6.2.1 Key Ideas

Consider a node j in a recurrent MLP that, for reasons of simplicity, receives input from just one node k and has only one recurrent connection w_{jj}, as illustreated in Fig. 6.5. The activation of j is then:

$$u_j(t) = w_{jj} f_j(u_j(t-1)) + w_{jk} f_k(u_k(t)) \tag{6.14}$$

where the index in f_j and f_k gives the freedom to use different activation functions for each node. According to eq. 6.12 and in order of the propagated error signal not to decay or to explode through time, the following must hold:

$$f'_j(u_j(t)) w_{jj} = 1 \tag{6.15}$$

Solving this equation with respect to f_j, we have:

$$f_j(u_j(t)) = u_j(t)/w_{jj} \tag{6.16}$$

i.e., f_j must be linear. By setting $w_{jj} = 1$, the activation of this node is summarized as:

$$u_j(t) = u_j(t-1) + w_{jk} f_k(u_k(t)) \tag{6.17}$$

This equation defines the *Constant Error Carousel* (CEC), a special node that allows for constant error signal propagation through time. This is the first key architectural idea of LSTM.

The second one is the use of *gates* to protect the contents of the CEC. As the activation of the CEC follows an additive law, there is the need to protect the CEC from irrelevant or noisy activations of node k. Furthermore, looking back to eq. 6.13 we see that if $\delta_j(t)$ is constant through time (like in a CEC), then the final weight update is guided by the input $x_i(t)$ and thus can be conflicting as $x_i(t)$ varies. To solve these two problems, a multiplicative *input gate* can be used to scale appropriately the term $w_{jk} f_k(u_k(t))$ of eq. 6.17. Following the same thinking, an *output gate* can be used to

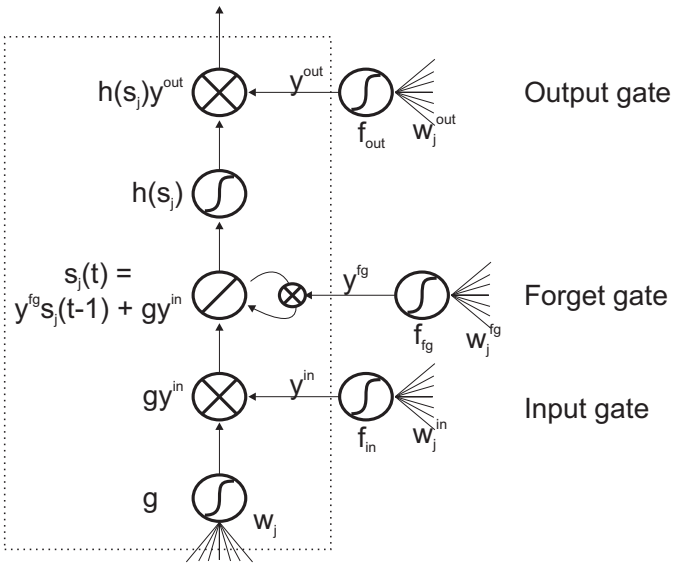


Figure 6.6
A typical Memory Cell j with one CEC $s_j(t)$.

protect the CEC from conflicting overwritings of $\delta_j(t)$ arriving from nodes k of the above hidden layer (or the output layer). This is particularly useful when error is inflicted at every time step, causing the nodes k to send an error signal back to the CEC each time. The operation of the output gate is to scale the CEC activation before it reaches another node. Finally, in [31], a *forget gate* is introduced to account for problems arising in tasks of continuous prediction or classification. In these tasks, the sequence length can be infinitely large leading the CEC into saturation. In addition, the CEC must be explicitly reset when the past history of the sequence is not useful anymore or can be conflicting. The role of the forget gate it to scale the term $u_j(t-1)$ before its addition at each time instant.

Intuitively speaking, the CEC gathers relative features from the whole input sequence and presents them to the output nodes. In this way, the temporal dimension of the problem is somehow removed. The gates have to break symmetries in the input sequence and to distribute the credit assignement problem throughout the sequence, like the random initialization of an MLP breaks symmetries in the hidden layers.

6.2.2 Definition of the LSTM

In a LSTM network, the elementary units are the *Memory Cells*, as in an MLP are the neurons. A memory cell contains one or more CECs in its heart and is controlled by an input, an output and a forget gate, as illustrated in Fig. 6.6. Each gate is in fact a sigmoid neuron whose outcomes are fused in a multiplicative fashion in the cell.

As with BPTT, the operations of the LSTM network are divided to the forward pass (stimulation) and to the backward pass (learning).

Forward Pass

The operation of the input gate is:

$$u_j^{\text{in}}(t) = \sum_k w_{jk}^{\text{in}} x_k(t) \tag{6.18}$$

$$y_j^{\text{in}}(t) = f_{\text{in}}(u_j^{\text{in}}(t)) \tag{6.19}$$

where index j refers to the cell the gate belongs to, k refers to the nodes that are connected to the input gate and $x_k(t) = y_k(t)$ if the connection is feedforward or $x_k(t) = y_k(t-1)$ is the connection is delayed. Similarly, the operations of the output and the forget gates are respectively:

$$u_j^{\text{out}}(t) = \sum_k w_{jk}^{\text{out}} x_k(t) \tag{6.20}$$

$$y_j^{\text{out}}(t) = f_{\text{out}}(u_j^{\text{out}}(t)) \tag{6.21}$$

$$u_j^{\text{fg}}(t) = \sum_k w_{jk}^{\text{fg}} x_k(t) \tag{6.22}$$

$$y_j^{\text{fg}}(t) = f_{\text{fg}}(u_j^{\text{fg}}(t)) \tag{6.23}$$

As there may exist more than one CECs inside each cell, we will use the index c for each of them. The operation for each CEC inside the cell j are as follows:

$$u_j^c(t) = \sum_k w_{jk}^c x_k(t) \tag{6.24}$$

$$s_j^c(t) = y_j^{\text{fg}}(t) s_j^c(t-1) + y_j^{\text{in}}(t) g(u_j^c(t)) \tag{6.25}$$

$$y_j^c(t) = y_j^{\text{out}}(t) h(s_j^c(t)) \tag{6.26}$$

The outputs of the cell $y_j^c(t)$ can be used as the output of the network or they can be used to feed standard sigmoid neurons that serve as the output nodes of the network. The cell outputs can also be used to feed recursively the cells themselves. The inputs of the gates can be identical to that of the CECs or less connectivity may be used. Generally, full connectivity is allowed and it is up to the engineer to decide if some of the connections are redundant. A typical LSTM network is given in Fig. 6.7 with two memory cells having 2 CECs each. The output nodes of the network are sigmoid. Standard neurons can also be used in the hidden layer of the network, along with the memory cells. They should not however carry any recurrent connection.

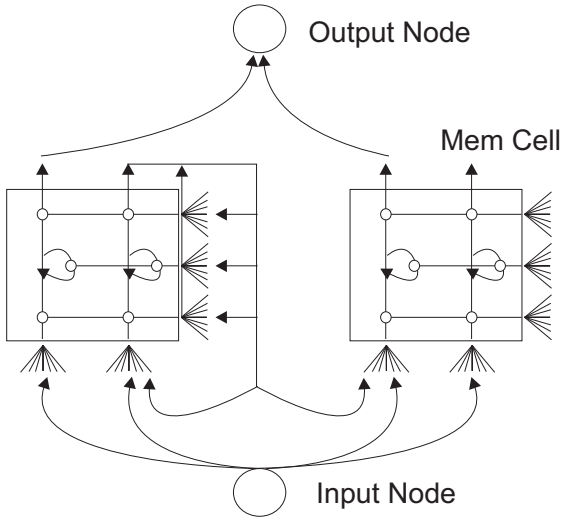


Figure 6.7

A typical LSTM network with 2×2 memory cells. Full connectivity is used between the hidden layer and the input and output layers. The hidden layer is also recurrently connected to itself. The incoming connections of the hidden layer are defined by the inputs of the CECs w_j^c, and the of the gates w_j^{in}, w_j^{fg}, w_j^{out}. The outgoing connections include the memory cell outputs y_j^c and the outputs of the gates.

Finally, following [39, 31], the functions used in a memory cell are defined as:

$$g(x) = \frac{4}{1+\exp(-x)} - 2 \tag{6.27}$$

$$h(x) = \frac{2}{1+\exp(-x)} - 1 \tag{6.28}$$

while the gate unit functions f_{in}, f_{out} and f_{fg} are the standard logistic sigmoid with range $[0, 1]$ (eq. 6.3 with $\alpha = 1$).

Backward Pass

When an error signal is inflicted at the output gates of an LSTM network, standard backpropagation is used for its propagation until it reaches a memory cell. The weights of the output gates or of other sigmoid neurons in the hidden layer are updated according to the standard backpropagation formula as they are not allowed to participate to recurrent connections.

When an error E is inflicted at the output of a cell, the weights of the cell are updated with the classical gradient-based formula $\Delta w = -\rho \partial E / \partial w$ where w belongs to the cell's and gates' weights. Errors are not inflicted directly to the gates of the cell, even when they are connected to other nodes of the network except the cell they belong to.

The calculation of the partial derivatives is done with almost standard backpropagation, mixed with some notions of RTRL learning.

According to eq. 6.26 and 6.9, the local gradient for the output gate will be:

$$\delta_j^{\text{out}}(t) = f'_{\text{out}}(u_j^{\text{out}}(t)) \sum_c \left(h(s_j^c(t)) \sum_k \delta_k(t) w_{k s_j^c} \right) \tag{6.29}$$

where c denotes summation over all the CECs of the memory cell, k summation over all the nodes of the above layer, and $w_{k s_j^c}$ is the weight of the connection between the node k and the c output of the memory cell j. The weight update for the output gate is then simply:

$$\Delta w_{jk}^{\text{out}}(t) = \rho \delta_j^{\text{out}}(t) x_k(t) \tag{6.30}$$

Let us define the internal local gradient for every CEC of the memory cell as:

$$e_{s_j^c}(t) = -\frac{\partial E(t)}{\partial s_j^c(t)} = y_j^{\text{out}}(t) h'(s_j^c(t)) \sum_k \delta_k(t) w_{k s_j^c} \tag{6.31}$$

The partial derivative of $E(t)$ with respect to the weights w_{jk}^c of the CEC is:

$$-\frac{\partial E(t)}{\partial w_{jk}^c} = e_{s_j^c}(t) \frac{\partial s_j^c(t)}{\partial w_{jk}^c} \tag{6.32}$$

The last derivative of the above equation is defined *recursively* by differentiating eq. 6.25:

$$\frac{\partial s_j^c(t)}{\partial w_{jk}^c} = y_j^{\text{fg}}(t) \frac{\partial s_j^c(t-1)}{\partial w_{jk}^c} + y_j^{\text{in}}(t) g'(u_j^c(t)) x_k(t) \tag{6.33}$$

The derivative is initialized abritrarily $\frac{\partial s_j^c(t=0)}{\partial w_{jk}^c} = 0$. Finally, the weight update is:

$$\Delta w_{jk}^c(t) = \rho e_{s_j^c}(t) \frac{\partial s_j^c(t)}{\partial w_{jk}^c} \tag{6.34}$$

The formulas for updating the weights of the input and forget gates are derived similarly. The partial derivatives of $E(t)$ with respect to a weight l of the input or forget gate is now:

$$-\frac{\partial E(t)}{\partial w_{lk}} = \sum_c e_{s_j^c}(t) \frac{\partial s_j^c(t)}{\partial w_{lk}} \tag{6.35}$$

The last derivative is defined for the input and forget gate respectively as:

$$\frac{\partial s_j^c(t)}{\partial w_{jk}^{\text{in}}} = y_j^{\text{fg}}(t) \frac{\partial s_j^c(t-1)}{\partial w_{jk}^{\text{in}}} + g(u_j^c(t)) f'_{\text{in}}(u_j^{\text{in}}(t)) x_k(t) \tag{6.36}$$

$$\frac{\partial s_j^c(t)}{\partial w_{jk}^{\text{fg}}} = y_j^{\text{fg}}(t) \frac{\partial s_j^c(t-1)}{\partial w_{jk}^{\text{fg}}} + s_j^c(t-1) f'_{\text{fg}}(u_j^{\text{fg}}(t)) x_k(t) \tag{6.37}$$

The update for the weight l is:

$$\Delta w_{lk}(t) = \rho \sum_{c} e_{s_j^c}(t) \frac{\partial s_j^c(t)}{\partial w_{lk}} \tag{6.38}$$

Following [39] and for reasons of simplicity, the propagation of the error signal stops to the incoming connections of the memory cell.

To sum up, the learning phase for a memory cell starts by stimulating it with a complete sequence $O_{1:T}$ via the equations 6.18 to 6.26. In parallel, the partial derivatives of equations 6.33, 6.36 and 6.37 are calculated recursively. When an error is inflicted (this can be at every time instant when we have a prediction problem or at the end of the sequence when we have a classification problem), the equations 6.29, 6.30, 6.31, 6.34 and 6.38 are executed.

6.3 Application to Tennis Video Analysis

Connectionist approaches can be used to model observation at the state (i.e., shot) level or at the segmental (i.e., scene) level, leading to HMM-NN or SM-LSTM hybrids, respectively. In addition, LSTM can be used by its own to detect the scene boundaries and label the scenes, without employing any of the HMM or SM models and without performing Viterbi decoding at all. The problem of video structure parsing is then translated to a problem of continuous classification of an input sequence by the LSTM.

6.3.1 HMM-NN Hybrid

There are numerous approaches to fusing neuronal and hidden Markov models, including using Learning Vector Quantization for a discriminative quantization of the input or the use of hybrid backpropagation and EM-based training algorithms. A review for various HMM-NN hybrids applied to speech recognition can be found in [99].

Perhaps, the most straightforward approach is to use the output of an MLP as a score that can replace the observation probability $P(O_t|S)$. Supposing that we can assign to the neural network output (being positive and less than 1, as it is when logistic sigmoids are used) a probabilistic interpretation, it approximates then the state posterior $P(S|O_t)$ that a given observation O_t belongs to the class S. Observation probabilities can be obtained by dividing the state posterior by the state priors $P(S)$. In an other way of viewing HMM/NN hybrids, hypotheses generated by the MLPs are refined in the global optimization framework of Viterbi decoding. The path that cumulates the best score from the MLP outputs and the best score according to the HMM transition probabilities is the one selected. In this case, the probability $P(O_t|S)$ is replaced be the score:

$$\log P(O_t|S) \equiv \log f_S(O_t) \tag{6.39}$$

where $f_S(O_t)$ is the activation outcome to the input O_t of an MLP trained to respond to class S.

In both scenarios, an MLP was trained to map each observation vector O_t to one of the 12 classes of Fig. 4.3(b). It has 1 hidden layer with 10 nodes[1] and an output layer with 12 nodes. Each node of the network uses the logistic activation function. The input nodes of the network receive the 6 audiovisual descriptors of eq. 4.31. Descriptors that receive real values were linearly scaled in $[0, 1]$ while the binary ones took the values 0 or 1. The network receives input from 5 successive vectors of the input sequence, instead of 1, in an attempt to add more context. It has thus 30 input nodes. The coding scheme for the output nodes is all zero except the node that corresponds to the target class, which is set to 1.

6.3.2 SM-RNN Hybrid

An SM-RNN hybrid can be built by using a recurrent MLP network as a segmental scorer in the same sense as using an MLP as observation scorer in HMM-NN hybrid. The segmental score is then defined as:

$$\log P(O_{1:l}|l, S) \equiv l \log f_S(O_{1:l}) \tag{6.40}$$

where f_S is an LSTM or a BPTT network trained to respond to the class (scene) S. The multiplication with l of the log-output of the network is necessary for Viterbi decoding to work as it requires scores that scale log-linearly with the segment length. The fundamental difference between the HMM-NN hybrid and the SM-RNN one is that in the second case the neuronal scorer is fed with the complete context of a scene and is able to model inter-scene dependencies. In the HMM-NN hybrid, it is difficult to capture local context as the MLP is fed with frame-wise input. The local context (as long as the global context of the inter-scene dependencies) is left to the HMM transition probabilities.

Four recurrent networks, each for the four scenes, are used as scorers. Their input is encoded as with the MLP scorer in the HMM-NN hybrid (but the input sequence is sampled with a window of length 1 this time). The networks have one output node, trained to turn on when the input segment belongs to the class.

6.3.3 Continuous Classification with LSTM

In this approach, a single LSTM network is fed with the complete observation sequence of the video and is trained to classify in a continuous manner all the shots as belonging to one of the 4 scenes. It contains 5 output nodes, 4 for the scenes plus one, which is

[1]Set after experimentation. Generally speaking, the network topologies, learning rates, etc., used in this chapter were set after experimentation.

trained to turn on whenever a scene boundary appears[2]. The shots are labeled according to the output node that exhibits the highest activation. It was found however in early experimentation that when a new scene begins it is difficult for the network to give the right answer at the first shots of the scenes. For example, both the 'First Missed Serve and Exchange' and 'Exchange' scenes start with a court view and thus it is difficult to discriminate between them based only on the first shot. The appearance of more court views (or not) in the upcoming shots will disambiguate the situation. For this reason, the shots are labeled as follows: when the scene boundary node is turned on, the node with the highest activation at this time instant labels all the shots down to the previous activation of the scene boundary node.

6.4 Experimental Results

Performance analysis of the LSTM and the various hybrids is provided in this section. Firstly, we will examine training and performance of the segmental scorers. A comparison between the LSTM and BPTT scorers with respect to their learning behavior is given. Finally, the hybrids and the continuous classification with the LSTM are compared to the models of chapter 4.

6.4.1 The Connectionist Segmental Scorers

Training an RNN-based segmental scorer has the particularity that, besides the positive examples, negative examples of the other scenes are needed and also of wrong segmentations. As the number of all possible wrong segmentation is quite large, the complete set of negative examples will then be too large to be used to train network due to the limited storage capacities. In order to collect a small and, at the same time, informative set of negative examples, a *bootstrapping* procedure is followed: the network is firstly trained with an initial set of randomly chosen negative examples. Strong false alarms are then gathered by running the network on the training sequences. The newly produced false alarms are added to the set of negative examples and learning goes on. When the number of false alarms produced of the network is drastically reduced or stability is noticed, the bootstrapping stops. In the experiments described below, false alarms are collected when being higher to a threshold which begins with 0.8 and linearly reduces to 0.5 in each bootstrapping iteration, and then remains constant.

Regarding the set of positive examples, it was enhanced with newly created examples by adding white noise to the original ones as a means to avoid overfitting. For each original training example, 100 artificial examples were added to the training set. In each

[2]A scene boundary may appear between two scenes of the same label. In this case, it is impossible to tell the presence of a boundary without using a fifth output node charged with this task.

training epoch, an equal number of randomly chosen positive and negative examples is presented to the network.

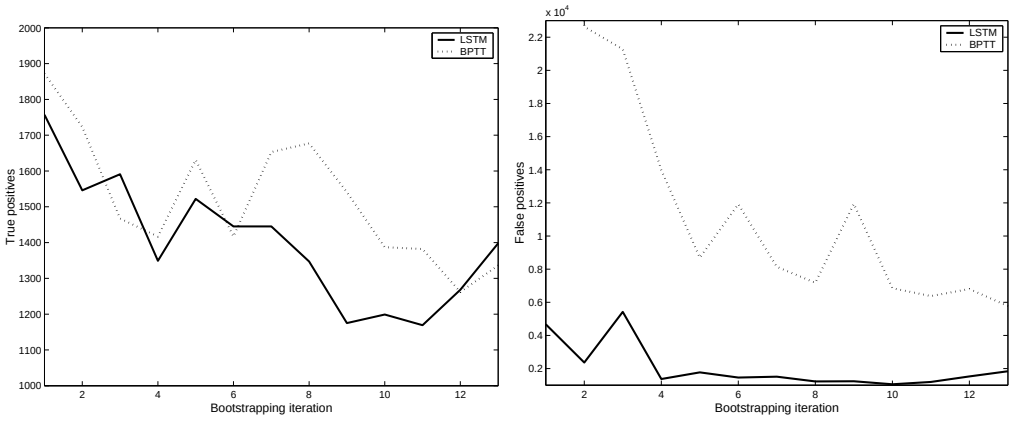
The details of the BPTT networks (identical topologies were used for each scene) are as follows. The hidden layer contains 5 neurons. Each neuron is fully connected to the input layer with no delay and, recurrently, to all the neurons of the hidden layer. The recurrent connections were doubled to include time delays of two time steps beside the usual one time step delay. This doubled recurrent connection scheme was found to improve performance during experimentation. The output nodes are fully connected to the hidden layer with no delay. This topology carries 91 adjustable weights. The weights were randomly initialized with maximum absolute value unity divided by the number of the weights of the neuron. Learning rates were kept constant and set to 0.0001. Errors are inflicted only once, at the end of the sequence.

The LSTM networks (having identical topologies for each scene) were built as follows. First of all, the forget gates were deactivated because the problem at hand is not a continuous prediction or classification problem and there is no need to "forget" the sequence. Each LSTM network has 4×2 memory cells in its hidden layer (i.e., 4 memory cells with 2 CECs each). Each memory cell exports as output the outputs of the CECs but not the ones of the gates. The gates and the CECs receive exactly the same input, originating from the input nodes without delay and, recurrently, from the memory cell outputs with delay 1. The output node is fully connected to the memory cells. This LSTM topology contains 249 weights in total. The weights were randomly initialized with maximum absolute value 0.2. The learning rate was fixed to 0.01. Errors are inflicted only once, at the end of the sequence.

Results of the bootstrapped training of the networks are reported in Fig. 6.8 and for the scene 'First Missed Serve and Exchange', which is the more challenging to model and detect. We see the produced true and false positives after running the network in the test set. A true positive is defined as an activation greater than 0.5 on an input that is a valid 'First Missed Serve and Exchange' scene[3]. If the input cannot be characterized as such, then a false positive is encountered. While BPTT produce slightly more true positives, LSTM quickly learns not to produce a lot of false alarms. Indeed, as we can see in Fig. 6.8(b), BPTT produce in fact 5 to 10 times more false alarms. This behavior for both of LSTM and BPTT was verified in several runs of the bootstrapping with different initializations of the networks.

Generally, it appears that BPTT is not able to learn the class. It has to remember, for instance, that at the first input vector of the sequence must be a global view of a missed serve. This requires a memory of something like 5 time steps or more, which seems to be difficult for BPTT. LSTM on the contrary demonstrated some success on

[3]Due to the trailing non-court views at the end of the scene, a unique scene in the test set can generate multiple true positives.



(a) (b)

Figure 6.8
Learning Behavior of LSTM and BPTT during bootstrapping. Results are collected in the test set and for the scene 'First Missed Serve and Exchange'.

carrying out the task and cleary performed better than BPTT. For this reason BPTT is excluded from any further analysis in the subsequent of this section.

6.4.2 Tennis Structure Analysis

Before proceeding to reporting the results, let us remind that in training with backpropagation one usually uses a validation set to determine the time instant where learning should stop. Unfortunately, in our case there are not a lot of data that would permit to reserve from the training set a sufficient number of examples to build a validation set. To account for this lack of validation set, the experimental protocol is defined as follows. The networks are trained until the best generalization is noticed in the test set. This training is performed just to determine an approximate number of training epochs, as a hint to achieve good generalization, i.e., for neither undertraining nor overtraining the network. The networks are then trained again from scratch in multiple runs, with fixed learning rates but also with varying random initialization of the weights. Learning is stopped strictly after reaching the above-mentioned number of training epochs. As the initialization of the network is changed, this stopping criterion does not generally results into the best performance in the test set but still it gives a reliable hint of what could be achieved with a legitimate validation set. The results reported below are the averages of the multiple runs.

Results are reported in table 6.1. The performance measurement C, P, R, and $\hat{F}$ are as defined in section 4.5. The feature set used in all approaches is the audiovisual shot-based descriptors.

	C	P	R	$\hat{F}$
HMMs				
HMM-NN	80.99	84.53	77.96	65.76
HMM	80.23	84.69	79.70	66.42
SMs				
SM-LSTM/A=1, D=1	77.13	83.85	72.85	60.45
SM-LSTM	81.31	84.14	77.07	65.23
SM/A=1, D=1	61.16	95.73	39.69	35.47
SM	84.39	86.25	79.32	69.29
SM-no init	80.17	81.80	76.93	63.35
LSTM				
	78.87	85.72	80.61	66.68

Table 6.1
Experimental results of various approaches.

In the first line, we see the performance of the HMM-NN hybrid. The reported results were obtained after division of the NN scores with the state priors. If not doing so, the results lie in the same range but with worse average. Comparing the HMM-NN hybrid with the HMM model of chapter 4, we see that more or less they perform the same. The inclusion of a neuronal scorer thereby does not improve significantly the results.

In the next two lines of the table 6.1, results regarding the SM-LSTM hybrid are given. Firstly ('SM-LSTM/A=1, D=1') we see its performance discarding the transition and duration probabilities of the SM. In fact, this is not a "hybrid": it uses only the LSTM scorers in a Viterbi optimization for decoding. A relatively good performance is achieved, which is improved with the addition of the transition and duration terms that the hybrid adds ('SM-LSTM'). In the next two lines, the performance of the SM of chapter 4 is given for comparison. In the line 'SM/A=1, D=1', the transition and duration terms of the SM are removed. We notice by comparing this to 'SM-LSTM/A=1, D=1' that the LSTM scorers alone can perform much better than the HMM scorers alone. However, this image is reversed when comparing the full SM ('SM') with the full SM-LSTM hybrid. The last one cannot outperform the SM and, in fact, cannot outperform even HMMs. A possible explanation to this is that the HMM scorers compute in fact a smoothed distance of a sequence to a model. This can be easily refined by the transition and duration scores. The LSTM scorers, on the other hand, are trained to give a 0 output even when the input sequence differs slightly to the prototype class. Most of the time thus their scores are 0 and little space of improvement is left to the transition and duration scores.

Finally, let us remind that the HMM scorers are especially initialized according to Fig. 4.3(b). This is equivalent to infusing prior knowledge to them, while the LSTM scorers were built from scratch. In line 'SM-no init', we see the performance of the SM when the HMM scorers are initialized randomly. This comparison reveals that the prior knowledge infused in the HMM scorers is really usefull to their performance. Without it, the SM-RNN hybrid can outperform the SM with HMM scorers.

Finally, in the last line of the table 6.1 we see the performance of the LSTM network of continuous classification, trained to label and segment an input sequence without employing Viterbi decoding at all. The network is trained with just the training sequences (without the need of using positive/negative examples), artificially corrupted with white noise to avoid overfitting. It has 10×2 memory cells in its hidden layer and receives input as the LSTM segmental scorers. It contains 1455 weights in total, estimated with a fixed learning rate of 0.001.

This approach does not use any form of prior knowledge, except of course from the fact that the addition of white noise does not alter the labeling of the sequences. Despite this straightforward use of LSTM to the problem, a good performance is achieved, in comparison to the hybrids analyzed above. Finally, this approach has the disadvantage that no Viterbi-based specialities can be applied to it, like the Score-Oriented Viterbi Search.

CHAPTER 7

Conclusions

7.1 Review of the Findings

In this study, the framework of SMs has been introduced in video indexing as a means of performing audiovisual integration with relaxed synchrony constraints. Baseline HMM systems suffer from the state-synchrony constraint imposed by the frame-based observations. The use of segmental features instead can extend the synchronization points between the modalities at the segment boundaries. By modeling each modality inside its own segment, native sampling rates and model topologies can be used. SMs were applied in a task of segmenting tennis broadcasts into human-meaningful scenes, each scene being a segment. Their experimental analysis and their comparison to HMMs, as summarized in table 4.2, revealed the following.

First of all, using video-only data (i.e., only one modality), SMs demonstrated a performance improvement over HMMs. The inclusion of an appropriate scene-based duration model, which was impossible with HMMs, gave finally to SMs a performance superiority. And this despite the fact that Viterbi decoding for SMs operates in an enhanced search space of possible state sequence and segmentation hypotheses. Regarding computational cost, SMs add an extra overhead to Viterbi decoding, exactly due to this enhanced search space. But, as analyzed in section 4.4.3, decoding time scales linearly to the maximum number of shots that a scene can contain (or more generally, to the maximum number of samples that a segment can contain). This number is defined a priori and was set to 80 in this study. Overall, SMs offer more modeling power than HMMs and can thus provide better performance while keeping computational cost at

reasonable levels.

Second, synchronous audiovisual integration with the form of shot-based audiovisual descriptors yielded a clear performance improvement in both HMMs and SMs. The auditory features helped the system mainly to discriminate between idle court views and court views of actual game exchanges, where tennis sound is expected to be detected in the soundtrack. SMs continue to perform better than HMMs.

The results regarding asynchronous audiovisual integration with SMs are however less optimistic. The problem is not that the auditory models that are asynchronous to the visual ones are less powerful and thus mitigate the results. But in order to fuse information from asynchronous models, they have to be assumed independent. This independency assumption in our case causes a loss of important (as it seems to be) correlations between auditory and visual features. The segmental scorers are based thus in less information and this makes their task harder. The problem detected here brings the discussion to whether a late fusion (decision on the semantics and then fusion) or an early fusion scheme (fusion before the decision on the semantics) is preferable, which is still open in the video indexing literature. Finally, nothing prohibits the simultaneous synchronous and asynchronous fusion in SMs, as a handy engineering for performance improvement.

Beside the audiovisual integration, this study was also concerned in chapter 5 with the fusion of high-level information coming from the game structure and rules and from the score announcements. The first dictates that the solution provided by Viterbi decoding should be consistent with the tennis rules and the second one that it must be consistent with the actual game evolution.

The incorporation of the tennis rules constraints is easily performed by extending the flat topologies of both HMMs and SMs to hierarchical ones, in order to follow the hierarchical decomposition of a tennis game. Hierarchical topologies also allow for the detection of high-level segments in the video, like a set or a game. Unfortunately, the number of transition probabilities in a hierarchical topology is very high and thus they have been manually set to arbitrary values. For this reason, the hierarchical topologies did not outperform the flat ones.

Regarding the score announcements, their fusion as an extra feature (or observation) at the shot level (HMMs) or at the scene level (SMs) was firstly considered. In both case, a marginal performance improvement was noticed. The reason is that score announcement may appear a lot delayed or may not appear at all after a game event. The probability distributions of this feature become thus almost uniform, i.e., they carry no useful information.

The use of the Score-Oriented Viterbi Search was proposed instead in order to fully exploit the semantic content of the score announcements and their positions as well. The algorithm finds out the most likely path that is consistent with the score announce-

ments in the expense of a computation overhead a little superior to the standard Viterbi decoding for both HMMs and SMs. The fusion of the score announcements as such yielded a clear performance improvement in both HMMs and SMs and with both flat or hierarchical topologies. The algorithm also naturally exhibits tolerance to noisy features and gracefully degrades to the standard Viterbi when a lot of noise exists.

Finally, a SM-RNN hybrid was presented in chapter 6. As connectionist segmental scorer, the newly introduced LSTM topology was favorably compared to BPTT-trained RNNs. The hybrid performed visibly inferior to the standard SM but still with a promising performance. In fact, what makes the difference is that the HMM-based segmental scorers can use prior knowledge on the task directly into their topology, while the LSTM scorers can be built only from scratch. An LSTM network was also proposed to solve on its own the problem of video structure analysis, viewing it as a problem of continuous classification. This straightforward application of the LSTM, which does not use prior knowledge on the task at all, yielded some promising results but still cannot outperform SMs.

7.2 Domains of Application

The framework developed in this study for tennis structure analysis can find practical application to the automatic management of large video databases. The typical scenario is that the documentalist stores a tennis video and then uses an automatic tool for the generation of the table of contents of the video. This can be done in a few minutes while the manual generation would require a considerable amount of time, which can overcome the total duration of the video.

The emergence of sophisticated consumer products like personal video recorders[1] (PVRs) gives to automatic video structure analysis another interesting area of application. These devices can store broadcasted video and then process it according to the user needs. Simple features currently supported like detection of commercials or of explicit content can be generalized to the automatic construction of the table of contents of a broadcast like a tennis match. Consumers will be able to personalize what they want to watch in a much more advanced way than patiently wait. This kind of automatic interaction with video can be achieved only by devices that are equipped with intelligent video understanding capabilities.

[1] See for example http://www.tivo.com

7.3 Limitations

A possible source of limitation of the system in its practical application could be the production style. It is not expected however to see in the future broadcasts that do not display a court view during an exchange because this is the easiest and most standard way for humans to watch the game action. In addition, rediffusions are always signaled with some special effect, otherwise they can be confusing to the persons that watch the broadcast. What can change is the special effect itself. A special image could be used, for instance, instead of a dissolve transition. In that case, feature extractor needs to be changed, but not the model. Generally, the feature extractors may need to be adapted to new conditions or new and improved methods may appear in the future and thus the input of the models can change. What might be needed then, in the worst case, is simply to re-estimate the parameters of the models.

It was found that the production style of displaying score announcements varies significantly from broadcast to broadcast. Fortunately, Score-Oriented Viterbi Search takes into consideration the worst case scenario, that is undisplayed announcements and extended asynchrony. More optimistic scenarios of instant (synchronous) and secured announcement of the score after a game event are easily handled.

7.4 Future Work and Extensions

The performance of the system can always be improved by adding useful information in the feature set. A kind of information that was not exploited in this study is the position of the players: players have to change position between successive exchanges as tennis rules dictate. Hence, after a missed serve the player must serve from the same position. Tracking of the player's changing position can help thus in the detection of missed serves. In some early experimentation and using some basic image processing operations, the detection of a changed position was found to be relatively easy and stable. Unfortunately, this tracking presumes that the court views with game action of the video should be already detected and without any mistakes. An idle court view for instance is equivalent to a mistake in the tracking, as in terms of changing position produces the same effect with a missed serve. A possible solution to this problem could be tracking the player's movement inside each court view shot. It is expected that the behavior of the player during an exchange or during an idle court view would be much different.

Extensions of the dissolve detection method described in chapter 3 could be interesting too. A key point of this method was that a single metric or feature alone cannot tackle the problem efficiently. An idea thus could be to use more information, like exploiting motion fields along with the pure image-based metrics. Information can be fused

using statistical methods, trained on a large set of examples to achieve a good degree of generalization. The framework of SMs can perfectly fit to this task, and it would be interesting a comparison with the pure HMM-based approaches of [12, 7]. Indeed, SMs offer at first a more advanced duration model. In addition, and more important, each feature has its own dynamics and behavior during a dissolve. With SMs, different models can be used for every feature stream, which is impossible with HMMs.

A possible future extension of the SM framework as applied to tennis structure analysis could be the inclusion of weight terms for each modality, like in eq. 2.5. These weights will balance the contribution of the respective modality so that a noisy modality will not affect the overall performance of the system. The value of the weights can be adaptive according to some reliability measurements. As there exist some studies on this topic for HMMs (e.g., [33]), it could be interesting to extend them for SMs. Unfortunately, it is believed that a much larger training corpus than the one used in this study should be used to avoid any overfitting.

SMs could also be applied to other genres of video where structure analysis is required, like news broadcasts. The segment could be defined as a news story unit, comprising a number of successive shots. The detection of the boundaries between the stories units is then left as part of the Viterbi optimization problem. Information sources from multiple modalities, like image, sound and text could be asynchronous between them but, by definition, they are synchronous inside the story unit boundaries.

Apart from video indexing, SMs could find application to audiovisual speech recognition as a generalization of the product and other HMM variants. A phoneme could be defined as a segment, which contains both auditory (speech waveform) and visual (face image) data.

A major drawback of the asynchronous fusion is that it implies a more or less late integration scheme. In this study, unimodal models made decisions directly on the semantics (the scenes) that are fused during Viterbi for SMs, following a pure late integration scheme. This resulted into light performance degradation. A question that naturally arises is how one can perform asynchronous and early fusion simultaneously. DBNs, which are in rapid development, may offer such solutions in the future. Their use as a segmental scorer in a SM-DBN hybrid can simplify their modeling/decoding complexity as modeling segments rather than whole data sequences may be easier. Another solution for asynchronous early fusion may be sought in the direction of LSTM segmental scorers. A multistream architecture may be built where each stream is sampled at its own frame rate. The nodes from different streams can be connected, resulting with this straightforward way into an asynchronous early fusion.

Finally, Score-Oriented Viterbi Search could be used in other video genres where game events occur rather frequently, like in basketball. Let us suppose that an automatic Viterbi-based system attempts to segment the video into attacks and label them as

scoring or not. The proposed algorithm can ensure that the solution obtained is in consistency with the actual game score.

Bibliography

[1] N. Adami, R. Leonardi, and P. Migliorati. An overview of multi-modal techniques for the characterization of sport programmes. In *Proc. of SPIE-VCIP'03*, pages 1296–1306, 2003.

[2] A.A. Alatan, A.N. Akansu, and W. Wolf. Multi-modal dialog scene detection using hidden Markov models for content-based multimedia indexing. *Multimedia Tools and Applications*, 14(2):137–151, 2001.

[3] J. Assfalg, M. Bertini, A. Del Bimbo, W. Nunziati, and P. Pala. Soccer highlights detection and recognition using HMMs. In *IEEE International Conference on Multimedia and Expo (ICME02)*, pages I: 825–828, 2002.

[4] N. Babaguchi, Y. Kawai, and T. Kitahashi. Event based video indexing by intermodal collaboration. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS'99)*, pages 782–786, 1999.

[5] N. Babaguchi, Y. Kawai, and T. Kitahashi. Event based indexing of broadcasted sports video by intermodal collaboration. *IEEE Transactions on Multimedia*, 4(1):68–75, 2002.

[6] N. Babaguchi and N. Nitta. Intermodal collaboration: a strategy for semantic content analysis for broadcasted sports video. In *International Conference on Image Processing (ICIP 2003)*, volume 1, pages 13–16, 2003.

[7] T.M. Bae, S.H. Jin, and Y.M. Ro. Video segmentation using hidden Markov model with multimodal features. In *Proceedings of the International Conference on Image and Video Retrieval*, pages 401–409, 2004.

[8] S. Bengio. An asynchronous hidden Markov model for audio-visual speech recognition. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems, NIPS 15*, pages 1237–1244. MIT Press, 2003.

[9] A. Berger, S. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[10] M. Betser and G. Gravier. Multiple events tracking in sound tracks. In *Intl. Conf. on Multimedia and Exhibition*, 2004.

[11] M. Betser, G. Gravier, and R. Gribomval. Extraction of information from video sound tracks - Can we detect simultaneous events ? In *Proc. of Conference on Content-Based Multimedia Indexing*, pages 71–78, 2003.

[12] J. Boreczky and L. Wilcox. A hidden Markov model framework for video segmentation using audio and image features. In *Proceedings of ICASSP*, pages 3741–3744, 1998.

[13] H. Bourlard and S. Dupont. A new ASR approach based on independent processing and recombination of partial frequency bands. In *Proc. ICSLP '96*, volume 1, pages 426–429, Philadelphia, PA, 1996.

[14] P. Bouthemy, M. Gelgon, and F. Ganansia. A unified approach to shot change detection and camera motion characterization. *IEEE Trans. on Circuits and Systems for Video Technology*, 9(7):1030–1044, October 1999.

[15] M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. In *Proceedings of IEEE CVPR*, pages 994–999, 1997.

[16] R. Brunelli, O. Mich, and C. M. Modena. A survey on the automatic indexing of video data. *Journal of Visual Communication and Image Representation*, 10(2):78–112, 1999.

[17] J. Calic, N. Campbell, S. Dasiopoulou, and Y. Kompatsiaris. An overview of multimodal video representation for semantic analysis. In *Proceedings of the European Workshop on the Integration of Knowledge, Semantics and Digital Media Technologies (EWIMT 2005)*. IEE, December 2005.

[18] P. Chang, M. Han, and Y. Gong. Extract highlights from baseball game video with Hidden Markov Models. In *Proceedings of the International Conference on Image Processing (ICIP'02)*, pages 609–612, 2002.

[19] Y.-L. Chang, W. Zeng, I. Kamel, and R. Alonso. Integrated image and speech analysis for content-based video indexing. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 306–313, 1996.

[20] S.-C. Chen, M.-L. Shyu, M. Chen, and C. Zhang. A decision tree-based multimodal data mining framework for soccer goal detection. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 265–268, 2004.

[21] F. Coldefy, P. Bouthemy, M. Betser, and G. Gravier. Tennis video abstraction from audio and visual cues. In *Proc. IEEE Int. Workshop on Multimedia Signal Processing, MMSP'2004*, Siena, Italy, September 2004.

[22] R. Dahyot, A. Kokaram, N. Rea, and H. Denman. Joint audio visual retrieval for tennis broadcasts. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 561–564, 2003.

[23] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[24] N. Dimitrova, L. Agnihorti, and G. Wei. Video classification based on HMM using text and faces. In *Proceedings of the European Signal Processing Conference*, 2000.

[25] S. Eickeler, A. Kosmala, and G. Rigoll. Hidden Markov model based continuous online gesture recognition. In *Int. Conference on Pattern Recognition (ICPR)*, pages 1206–1208, Brisbane, 1998.

[26] S. Eickeler and S. Muller. Content-based video indexing of TV broadcast news using Hidden Markov Models. In *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2997–3000, 1999.

[27] A. M. Ferman and A. M. Tekalp. Probabilistic analysis and extraction of video content. In *Proceedings of the IEEE International Conference on Image Processing (ICIP 99)*, 1999.

[28] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32(1):41–62, 1998.

[29] S. Fischer, R. Lienhart, and W. Effelsberg. Automatic recognition of film genres. In *MULTIMEDIA '95: Proceedings of the third ACM international conference on Multimedia*, pages 295–304, New York, NY, USA, 1995. ACM Press.

[30] U. Gargi, R. Kasturi, and S. Strayer. Performance characterization of video-shot-change detection methods. *IEEE Trans. on Circuits and Systems for Video Technology*, 10(1):1–13, February 2000.

[31] F. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.

[32] X. Gibert, L. Huipingand, and D. Doermann. Sports video classification using HMMs. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '03)*, 2003.

[33] H. Glotin, D. Vergyri, C. Neti, G. Potamianos, and J. Luettin. Weighting schemes for audio-visual fusion in speech recognition. In *Proceedings of Int. Conf. Acoust. Speech Signal Process*, 2001.

[34] Y. Gong, M. Han, W. Hua, and W. Xu. Maximum entropy model-based baseball highlight detection and classification. *Computer Vision and Image Understanding*, 96(2):181–199, 2004.

[35] M. Han, W. Hua, W. Xu, and Y. Gong. An integrated baseball digest system using maximum entropy method. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 347–350, 2002.

[36] A. Hanjalic, R. L. Lagendijk, and J. Biemond. Automated high-level movie segmentation for advanced video-retrieval systems. *IEEE Trans. on Circuits and Systems for Video Technology*, 9(4):580–588, 1999.

[37] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[38] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

[39] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[40] W. Hsu, S. Chang, C. Huangi, L. Kennedy, C. Lin, and G. Iyengar. Discovery and fusion of salient multi-modal features towards news story segmentation. In *IS&T/SPIE Symposium on Electronic Imaging: Science and Technology - SPIE Storage and Retrieval of Image/Video Database*, San Jose, CA, January 2004.

[41] W. H.-M. Hsu and S.-F. Chang. Generative, discriminative, and ensemble learning on multi-modal perceptual fusion toward news video story segmentation. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1091–1094, 2004.

[42] J. Huang, Z. Liu, Y. Wang, Y. Chen, and E.K. Wong. Integration of multimodal features for video classification based on HMM. In *Proceedings of IEEE Signal Processing Society Workshop on Multimedia Signal Processing*, pages 53–58, 1999.

[43] Q. Huang, Z. Liu, A. Rosenberg, D. Gibbon, and B. Shahraray. Automated generation of news content hierarchy by integrating audio, video, and text information. In *Proceedings of the IEEE International Conference On acoustics, speech, and signal processing*, volume 6, pages 3025–3028, 1999.

[44] U. Iurgel, R. Meermeier, S. Eickeler, and G. Rigoll. New approaches to audio-visual segmentation of TV news for automatic topic retrieval. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1397–1400, 2001.

[45] R. Jasinschi, N. Dimitrova, T. McGee, L. Agnihotri, J. Zimmerman, and D. Li. Integrated multimedia processing for topic segmentation and classification. In *Proceedings of the International Conference on Image Processing*, 2001.

[46] V. Kettnaker. Time-dependent HMMs for visual intrusion detection. In *Proceedings of the 2003 Conference on Computer Vision and Pattern Recognition Workshop*, pages 1–8, 2003.

[47] E. Kijak. *Structuration Multimodal des Vidéos de Sports par Modèles Stochastiques*. PhD thesis, University of Rennes 1, France, 2003.

[48] E. Kijak, G. Gravier, P. Gros, L. Oisel, and F. Bimbot. HMM based structuring of tennis videos using visual and audio cues. In *Proc. of the ICME*, volume 3, pages 309–312, 2003.

[49] E. Kijak, G. Gravier, L. Oisel, and P. Gros. Audiovisual integration for tennis broadcast structuring. *Multimedia Tools and Applications*, 30(3):289–311, 2006.

[50] K. Kim, J. Choi, N. Kim, and P.K. Kim. Extracting semantic information from basketball video based on audio-visual features. In *CIVR '02: Proceedings of the International Conference on Image and Video Retrieval*, pages 278–288, London, UK, 2002. Springer-Verlag.

[51] K. Lang, A. Waibel, and G. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):23–43, 1990.

[52] Y. LeCun. A theoretical framework for back-propagation. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 21–28, CMU, Pittsburgh, Pa, 1988. Morgan Kaufmann.

[53] R. Leonardi, P. Migliorati, and M. Prandini. Semantic indexing of soccer audio-visual sequences: a multimodal approach based on controlled Markov chains. *IEEE Trans. Circuits Syst. Video Techn.*, 14(5):634–643, 2004.

[54] M. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communication, and Applications*, 2(1):1–19, 2006.

[55] W.N. Lie and C.K. Su. News video classification based on multi-modal information fusion. In *Proceedings of the International Conference on Image Processing*, pages I: 1213–1216, 2005.

[56] R. Lienhart. Reliable transition detection in videos: A survey and practitioner's guide. *International Journal of Image and Graphics*, 1(3):469–486, 2001.

[57] W.-H. Lin and A. Hauptmann. News video classification using SVM-based multimodal classifiers and combination strategies. In *ACM International Conference on Multimedia*, pages 323–326, 2002.

[58] Z. Liu and Q. Huang. Detecting news reporting using audio/visual information. In *Proceedings of the International Conference on Image Processing*, pages III:324–328, 1999.

[59] Z. Liu and Y. Wang. Major cast detection in video using both audio and visual information. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, 2001.

[60] C. Lu, M. Drew, and J. Au. Classification of summarized videos using hidden Markov models on compressed chromaticity signatures. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 479–482, New York, NY, USA, 2001. ACM Press.

[61] J. Makhoul, F. Kubala, T. Leek, D. Liu, L. Nguyen, R. Schwartz, and A. Srivastava. Speech and language technologies for audio indexing and retrieval. *Proceedings of the IEEE*, 88(8):1338–1353, 2000.

[62] J. Makhoul, T. Starner, R. Schwartz, and G. Chou. On-line cursive handwriting recognition using hidden Markov models and statistical grammars. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 432–436, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[63] I. McCowan, D. Gatica-Perez, S. Bengio, G. Lathoud, M. Barnard, and D. Zhang. Automatic analysis of multimodal group actions in meetings. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(3):305–317, 2005.

[64] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, UC Berkeley, 2002.

[65] J. Nam, M. Alghoniemy, and A. Tewfik. Audio-Visual content-based violent scene characterization. In *Proceedings of the IEEE International Conference on Image Processing*, volume 1, pages 353–357, 1998.

[66] M. Naphade and T. S. Huang. A probabilistic framework for semantic video indexing, filtering, and retrieval. *IEEE Transactions on Multimedia*, 3(1):141–151, 2001.

[67] A. Nefian. *A Hidden Markov Model-Based Approach for Face Detection and Recognition.* PhD thesis, Georgia Institute of Technology, Atlanta, GA., 1999.

[68] A. Nefian, L.H. Liang, X. Pi, X.X. Liu, and K. Murphy. Dynamic Bayesian networks for audio-visual speech recognition. *EURASIP Journal on Applied Signal Processing*, 11:1274–1288, November 2002.

[69] S. Nepal, U. Srinivasan, and G. Reynolds. Automatic detection of 'goal' segments in basketball videos. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 261–269, New York, NY, USA, 2001. ACM Press.

[70] N. Oliver, A. Garg, and E. Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and Image Understanding*, 96(2):163–180, 2004.

[71] N. Oliver and E. Horvitz. A comparison of hmms and dynamic Bayesian networks for recognizing office activities. In *Proceedings of User Modeling*, pages 199–209, 2005.

[72] M. Ostendorf, V. Digalakis, and O. Kimball. From HMMs to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(5):360–378, 1996.

[73] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[74] L. Peshkin and G. Mikhail. Segmentation of yeast DNA using hidden Markov models. *Bioinformatics*, 15(12):960–966, 2000.

[75] M. Petkovic, V. Mihajlovic, W. Jonker, and S. Djordjevic-Kajan. Multi-modal extraction of highlights from TV Formula 1 programs. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 817–820, 2002.

[76] M. Petkovic, Z. Zivkovic, and W. Jonker. Recognizing strokes in tennis videos using hidden Markov models. In *Proceedings of the IASTED International Conference Visualization, Imaging and Image Processing, Marbella, Spain*, 2001.

[77] S. Pfeiffer, R. Lienhart, and W. Effelsberg. Scene determination based on video and audio features. *Multimedia Tools and Applications*, 15(1):59–81, 2001.

[78] D. Q. Phung, T. V. Duong, S. Venkatesh, and H. H. Bui. Topic transition detection using hierarchical hidden Markov and semi-Markov models. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 11–20. ACM Press, 2005.

[79] G. Potamianos, C. Neti, G. Gravier, A. Garg, and A.W.Senior. Recent advances in the automatic recognition of audiovisual speech. *Proceedings of the IEEE*, 91(9):1306–1326, 2003.

[80] W. Qi, L. Gu, H. Jiang, X.R. Chen, and H. Zhang. Integrating visual, audio and text analysis for news video. In *Proceedings of the International Conference on Image Processing*, pages Vol III: 520–523, 2000.

[81] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[82] M. Roach, J. Mason, L. Xu, and F. Stentiford. Recent trends in video analysis : a taxonomy of video classification problems. In *Proceedings of the International Conference on Internet and Multimedia Systems and Applications*, 2002.

[83] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[84] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, 1987.

[85] Y. Rubner, J. Puzicha, C. Tomasi, and J. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. *Computer Vision and Image Understanding*, 84(1):25–43, 2001.

[86] Y. Rui, T.S. Huang, and S. Mehrotra. Constructing table-of-content for videos. *Multimedia Systems*, 7(5):359–368, 1999.

[87] D. Rumelhart, G. Hinton, and R. Williams. Learning representations of back-propagation errors. *Nature*, 323:533–536, 1986.

[88] D.A. Sadlier and N.E. O'Connor. Event detection in field sports video using audio-visual features and a Support Vector Machine. *IEEE Trans. Circuits Syst. Video Techn.*, 15(10):1225–1233, October 2005.

[89] C. Saraceno and R. Leonardi. Identification of story units in audio-visual sequences by joint audio and video processing. In *Proceedings of the IEEE International Conference on Image Processing*, 1998.

[90] S. Satoh, Y. Nakamura, and T. Kanade. Name-it: naming and detecting faces in news videos. *IEEE Multimedia*, 6(1):22–35, January 1999.

[91] N. Sebe, M. Lew, and A. Smeulders. Video retrieval and summarization. *Computer Vision Image Understanding*, 92(2-3):141–145, 2003.

[92] T. Sejnowski and C. Rosenberg. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168, 1987.

[93] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[94] C. Snoek and M. Worring. Multimedia event-based video indexing using time intervals. *IEEE Transactions on Multimedia*, 7(4):638–647, 2005.

[95] C. Snoek and M. Worring. Multimodal video indexing: A review of the state-of-the-art. *Multimedia Tools and Applications*, 25(1):5–35, 2005.

[96] C. Snoek, M. Worring, and A. Smeulders. Early versus late fusion in semantic video analysis. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402, New York, NY, USA, 2005. ACM Press.

[97] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden Markov models. In *ISCV '95: Proceedings of the International Symposium on Computer Vision*, page 265, Washington, DC, USA, 1995. IEEE Computer Society.

[98] G. Sudhir, John C. M. Lee, and Anil K. Jain. Automatic classification of tennis video for high-level content-based retrieval. In *CAIVD '98: Proceedings of the 1998 International Workshop on Content-Based Access of Image and Video Databases (CAIVD '98)*, pages 81–90. IEEE Computer Society, 1998.

[99] E. Trentin and M. Gori. A survey of hybrid ANN/HMM models for automatic speech recognition. *Neurocomputing*, 37(1-4):91–126, 2001.

[100] A. Tritschler. *A segmentation-enabled speech recognition application using the BIC criterion*. PhD thesis, Institut EURECOM, France, 1998.

[101] B.T. Truong, C. Dorai, and S. Venkatesh. New enhancements to cut, fade, and dissolve detection processes in video segmentation. In *Proc. ACM on Multimedia*, pages 219–227, 2000.

[102] S. Tsekeridou and I. Pitas. Content-based video parsing and indexing based on audio-visual interaction. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(4):522–535, 2001.

[103] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[104] F. Wang, Y.-F. Ma, H.-J. Zhang, and J.-T. Li. A generic framework for semantic sports video analysis using dynamic Bayesian networks. In *Proceedings of the 11th International Conference on Multimedia Modeling (MMM 2005)*, pages 115–122, 2005.

[105] P. Wang, R. Cai, and S.-Q. Yang. Tennis video analysis based on transformed motion vectors. In *Proceedings of the Third International Conference on Image and Video Retrieval (CIVR'04)*, pages 79–87, 2004.

[106] Y. Wang, Z. Liu, and J. Huang. Multimedia content analysis. *IEEE Signal Processing Magazine*, 17(6):12–36, 2000.

[107] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In Y. Chauvin and D. E. Rumelhart, editors, *Back-propagation: Theory, Architectures and Applications*, pages 433–486. Lawrence Erlbaum Publishers, Hillsdale, N.J., 1995.

[108] W. Wolf. Hidden Markov model parsing of video programs. In *Proceedings of ICASSP*, pages 2609–2611, 1997.

[109] J. Xi, X.-S. Hua, X.-R. Chen, L. Wenyin, and H.-J. Zhang. A video text detection and recognition system. In *Proceedings of ICME*, pages 1080–1083, 2001.

[110] L. Xie, S. Chang, A. Divakaran, and H. Sun. Unsupervised discovery of multilevel statistical video structures using hierarchical hidden Markov models. In *Proceedings of the IEEE Intl. Conf. Multimedia and Expo (ICME)*, Baltimore, MD, July 2003.

[111] L. Xie, P. Xu, S.-F. Chang, A. Divakaran, and H. Sun. Structure analysis of soccer video with domain knowledge and hidden Markov models. *Pattern Recognition Letters*, 25(7):767–775, 2004.

[112] Z. Xiong. Audio-visual sports highlights extraction using coupled Hidden Markov Models. *Pattern Anal. Appl.*, 8(1):62–71, 2005.

[113] G. Xu, Y.F. Ma, H.J. Zhang, and S.Q. Yang. An HMM-based framework for video semantic analysis. *IEEE Trans. Circuits Syst. Video Techn.*, 15(11):1422–1433, November 2005.

[114] B. L. Yeo and B. Liu. Rapid scene analysis on compressed video. *IEEE Trans. on Circuits and Systems for Video Technology*, 5(6):533–544, December 1995.

[115] M. Yeung, B.-L. Yeo, and B. Liu. Extracting story units from long programs for video browsing and navigation. In *Proceedings of International Conference on Multimedia Computing and Systems*, 1996.

[116] R. Zabih, J. Miller, and K. Mai. A feature-based algorithm for detecting and classifying scene breaks. In *MULTIMEDIA '95: Proceedings of the third ACM international conference on Multimedia*, pages 189–200, New York, NY, USA, 1995. ACM Press.

[117] D. Zhang, D. Gatica-Perez, S. Bengio, I. McCowan, and G. Lathoud. Modeling individual and group actions in meetings: A two-layer HMM framework. In *IEEE Workshop on Event Mining at the Conference on Computer Vision and Pattern Recognition, CVPR*, volume 7, pages 117–124, 2004.

[118] H. J. Zhang, A. Kankanhalli, and S. Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, 1(1):10–28, 1993.

[119] D. Zhong and S.-F. Chang. Structure analysis of sports video using domain models. In *IEEE International Conference on Multimedia and Expo*, 2001.

Automatic video content analysis is an emerging research subject with numerous practical applications to large video databases or personal video recording systems. The focus of this study is the automatic construction of the table of contents of a tennis broadcast using Markovian models and dynamic programming. Motivated by the need for more efficient multimodal representations, the use of segmental features in the framework of Segment Models is proposed, instead of the frame-based features of Hidden Markov Models. Considering each scene of the video as a segment, the synchronization points between different modalities are extended to the scene boundaries, which is the basic thematic unit of the video. Visual features coming from the produced broadcasted video and auditory features recorded in the court are processed before fusion in their own segments, with their own sampling rates and models. Various techniques for modeling the segments are examined, including discrete or continuous density Hidden Markov Models, bigram models or connectionist scorers, operating on automatically extracted audiovisual features. Segment Models and Hidden Markov Models, with hierarchical or ergodic topologies, are built and compared in a corpus of 15 hours tennis video. The model parameters are estimated on labeled data. Depending on the segmental scorer employed, asynchronous fusion with Segment Models can achieve the same level of performance as Hidden Markov Models. The fusion of the textual resources of the video, namely the score announcements, is also considered. To fully exploit their semantic content on the actual game evolution and to account for unacknowledged game events, a novel Viterbi decoding scheme is developed. It produces solutions that are consistent with the score announcements and thus yields a clear performance improvement of the system.

Keywords: Hidden Markov models, Segment models, Video summarization, Multimodal fusion

L'analyse automatique du contenu de la vidéo est un sujet de recherche émergent avec de nombreuses applications pratiques sur de grandes bases de données de vidéo ou sur les enregistreurs vidéo personnels. Le centre de cette étude est la construction automatique de la table des matières d'une émission de tennis en utilisant les modèles markoviens et la programmation dynamique. Motivés par le besoin de représentations multimodales plus efficaces, nous proposons l'utilisation des caractéristiques segmentales dans le cadre des modèles de segment, au lieu des caractéristiques en trames des modèles de Markov cachés. En considérant chaque scène de la vidéo comme un segment, les points de synchronisation entre différentes modalités sont étendus aux frontières de la scène, qui est l'unité thématique de base de la vidéo. Les caractéristiques visuelles venant de la vidéo diffusée et les caractéristiques auditives enregistrées dans le court sont traitées avant fusion dans leurs propres segments, avec leurs propres modèles et fréquences d'échantillonnage. Diverses techniques pour modéliser les segments sont examinées, y compris les modèles de Markov cachés de densité discrète ou continue, les modèles bigrames ou des approches connexionnistes, fonctionnant sur les caractéristiques audiovisuelles automatiquement extraites. Des modèles de segments et des modèles de Markov cachés, avec des topologies hiérarchiques ou ergodiques, sont établis et comparés sur un corpus de 15 heures de vidéos de tennis. Les paramètres des modèles sont estimés sur des données étiquetées. Selon le modèle segmentale utilisé, la fusion asynchrone avec des modèles de segments peut atteindre le même niveau de performance que les modèles de Markov cachés. La fusion des ressources textuelles de la vidéo, c'est-à-dire les annonces de points, est également considérée. Pour exploiter entièrement leur contenu sémantique sur l'évolution réelle du jeu et tenir compte des événements non reconnus, un arrangement original du décodage de Viterbi a été développé. Il produit des solutions qui sont conformes aux annonces de points et apporte ainsi une nette amélioration de la performance du système.

Mots-clés: Modèles de Markov cachés, Modèles segmentaux, Structuration des vidéos, Fusion multimodale