ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2013-ENST-0027

EDITE - ED 130

# Doctorat ParisTech

# T H È S E

**pour obtenir le grade de docteur délivré par**

# TELECOM ParisTech

## Spécialité « Signal et Images »

*présentée et soutenue publiquement par*

**Alina Elma ABDURAMAN**

le 21 Mai 2013

# Structuration intra-programme de contenus TV

préparée à EURECOM - Departament Multimedia Communications
et à Orange Labs - France Télécom R&D, Rennes

Directeur de thèse : **Bernard MERIALDO**
Co-encadrement de la thèse : **Sid-Ahmed BERRANI**

**Jury**

| | |
|---|---|
| **M. Bernard MERIALDO**, Professeur, EURECOM | Directeur de thèse |
| **M. Philippe JOLY**, Professeur, Université Paul Sabatier | Rapporteur |
| **M. Guillaume GRAVIER**, Chargé de recherche CNRS HDR, Irisa | Rapporteur |
| **M. Matthieu CORD**, Professeur, LIP6/UPMC | Examinateur |
| **M. Jean CARRIVE**, Chef de projet recherche, INA SUP | Examinateur |
| **M. Sid-Ahmed BERRANI**, Chercheur, Orange Labs - France Télécom R&D | Examinateur |

THÈSE

**TELECOM ParisTech**
école de l'Institut Télécom - membre de ParisTech

# Acknowledgments

En m'assumant les risques de fautes d'orthographe (et pas seulement), je préfère adresser ces remerciements en français - preuve que cette thèse a représenté pour moi pas seulement une formation dans le domaine passionnant de l'audio-visuel mais aussi dans la culture et langue françaises.

Je commence par remercier les membres de jury d'avoir accepté la charge d'évaluer mon travail, pour leur intérêt et leurs remarques pertinentes.

Je tiens particulièrement à remercier Sid-Ahmed Berrani qui m'a guidé pendant les 3 ans de thèse à Orange Labs avec beaucoup de patience et des conseils précieux. Ceux-ci m'ont aidée à gagner de nombreuses connaissances et m'ont orientée dès mes premiers pas en tant qu'aspirante au grade de docteur jusqu'à ce jour où j'ai gagné tout le "bagage" nécessaire pour obtenir ce titre.

Je remercie également Bernard Merialdo, mon directeur de thèse, qui, même de loin, a été toujours présent et dont l'expérience du domaine et les remarques judicieuses m'ont beaucoup apporté et orientée dans les démarches suivies pendant ces 3 ans.

Dans le même ordre, je remercie tout le monde qui m'a considérablement aidée dans mon travail et que j'ai embêté pas mal de fois. N'est-ce pas JB? Je remercie donc Jean Bernard, Patrick, Franck, Olivier et j'espère que vous n'allez pas vous ennuyer sans mes mille questions.

Je poursuis en remerciant tous mes compagnons et voisins du bureau pour l'ambiance et les moments très agréables passés ensemble. Je remercie en particulier Philippe pour tous les cours gratuits de français et pour tous les goodies (roumains ou bretons) qu'on a partagés ensemble. Je te souhaite que tu gagnes au moins une fois un prix substantiel à un des jeux qu'on jouait assez souvent sur l'intranet. Et n'oublie pas qu'une fois tu as été roi et moi la reine. Garry, j'espère que j'ai finalement réussi à te montrer une autre face de la Roumaine à part celle que tu connaissais déjà très bien, même mieux que moi (celle des Roms). De toute façon, ton "s'il vous plaît, messieurs" était parfait. Haykel, je te souhaite bon courage pour la suite de ta thèse et n'oublie pas: la 1 ère année - ça va, la 2ème année - la plus belle, la 3ème année - la plus dure. Frédérique, merci pour le stress de chaque semaine de la 3ème année quand tu me demandais si j'avançais avec la rédaction. :)

Je salue également en passant et en désordre Foued, Vincent, Patrice, Jean Philippe, Nicolas, Philippe, Julien, Laurent, Benoit, David avec qui on a partagé les locaux, la can-

tine, le café, autour des conversations toujours intéressantes, amusantes et que...j'avais du mal à comprendre lors de mes 2 premiers mois quand je ne savais pas toutes les subtilités de la langue française. Je salue aussi Delphine et Géraldine même si on n'a pas réussi à avoir plus d'échanges à cause des locations différentes.

Sans les avoir oubliés à aucun moment, je remercie tous le thésards que j'ai connus pendant ma thèse et avec qui on a partagé les moments difficiles mais aussi les moments de joie de nos thèses. Je me réfère à Khaoula, Moez, Nahla, avec qui on a commencé et on a fini ensemble, mais aussi à la génération précédente: Sinda, Akl, 2xAli, Mohamed, Moussa, Lounes.

J'ai laissé en dernier les personnes qui sont les plus importantes pour moi, c'est-à-dire ma famille, sans le soutien de laquelle je n'aurais jamais réussi à commencer et finir cette thèse. Je préfère leurs adresser mes remerciements en roumain.

Am lasat la sfarsit persoanele care pentru mine sunt cele mai importante, si anume familia mea, fara a caror sustinere nu as fi inceput sau terminat niciodata aceasta teza de doctorat. Adi, mami, tati va multumesc pentru tot ce ati facut pentru mine din momentul in care am decis sa raman in Franta si pe toata durata acestor 3 ani, pentru ajutorul si sprijinul pe care mi l-ati acordat chiar si de departe. Lilu, Maia mai aveti un doctor in familie. Andu acum e randul tau. Remus asta a fost "maratonul meu" si de data asta tu m-ai urmarit si incurajat de pe margine - multumesc. Ralu si Deiu multumesc pentru ca ati fost si sunteti a doua mea familie. Stef, raman datoare cu o cinste.

# Contents

# Introduction

Due to the high number of television channels, the amount of broadcasted TV programs has significantly grown. Consequently, data organization and tools to efficiently manipulate and manage TV programs are needed. The TV channels broadcast audio-visual streams, in a linear way. This linearity generates temporal constraints in consulting the content. Services like Catch-up TV and PVRs (Personal Video Recorder) remove these constraints and allow users to watch previously broadcasted TV program. From the TV streams, a large number of TV Programs can be extracted, stored, indexed and prepared for later use. An additional indexing step is however still required. Indeed, after choosing a program, the user might want to get an overview of the program before watching it. He/she might also want to directly access a specific part of the program, to find a certain moment of interest or to skip a part of the program and go to the following one or even directly to the final one. These features represent an alternative for the basic fast forward/backward options. The value of these features was explored in [LGS+00], where users were provided with such capabilities for a wide variety of video content. The results showed that the users found the ability to browse video content very useful for the reasons of time saving and the feeling of control over what they watched.

To enable these features, after being extracted, each TV program has to be structured, that is, its original structure has to be recovered and all possible moments of interest have to be precisely tagged. A similar option exists on DVDs and offers the user a summary and the possibility to view a particular moment in the movie. But in order to obtain such a representation, a human observer is required to watch the entire video and locate the important boundaries. Obviously this could be done in the case of TV programs also, but these manual pre-processing steps are costly and highly time consuming, especially when dealing with large amount of TV programs, as is the case in real-world services. One challenge is hence to develop content-based automatic tools for TV program structuring. These tools will allow the easy production of information that will give to the users the capability to benefit of the structure of programs and watch just the parts of the programs they are interested in.

In this context, program structuring becomes essential in order to provide users with novel and useful browsing features. Basically, the objective of program structuring is to recover the original structure of the program. In other terms, the aim of structuring is to detect the start and end times of each part of the program content. When watching, this enables users to directly access the desired parts of the program or to skip the current part and directly go to the next one or to any other part he/she might find more interesting. It thus provides an advanced non-linear access functionality that could be an alternative to the basic fast forward/backward functions. In addition to advanced browsing features, the structure could also be used for summarization. Recovering the structure allows to build

a balanced summary where each part of the program is represented with respect to its importance. Indexing and querying, archiving, intra-program audience measurement are also possible applications. Interactive services could also benefit, for instance, by providing specific informations or features depending on the part currently being watched.

In time, a lot of approaches for structuring have been proposed. Some of them are more specific to the types of programs they analyze (i.e. the structuring of news or sport programs). These are supervised and generally require the use of prior knowledge of the domain. Others try to evolve towards generic approaches, that use as little or no prior knowledge if possible, as the approaches based on the use of recurrence or the scene segmentation methods. The notion of scene is however very subjective and depends on the human understanding of its meaning. This makes it hard to compare the performance of the existing approaches. The use of recurrences is however more promising. Generally approaches use the recurrence to segment news by detecting recurrences of the anchor person or to monitor stories in news, to extract meaningful informations or for topic segmentation in radio broadcasts, to discover repeating words, for music summarization, indexing and retrieval, for macrosegmentation or in event mining approaches.

This thesis focuses on TV programs and addresses the problem of TV program structuring. We propose to develop an approach that implements a largely unsupervised method for the structuring of a wide range of TV programs. As it is difficult to find a general method that covers all the types of existing TV programs, we focused, on "recurrent" TV programs.

A recurrent TV program is a program composed of several "episodes" that are periodically broadcasted (e.g. daily, weekly, monthly...). Examples of this type of programs are entertainments, game shows, magazines and news. We mainly explain our choice for this type of programs by the applicative interest this type of programs have. They represent an important part of the total number of broadcasted programs of the French generalist TV channels (as explained later in the following section). Moreover they have important properties that make the task feasible, meaning they generally have a clear structure with well defined main parts. This is done on purpose by content producers in order to allow viewers to easily follow an episode of the program even if they do not watch it from the beginning without interruption. The different parts are generally delimited by what we call "separators". Separators are short video sequences that are inserted between different parts of a program and that can be repeated between and/or within the episodes of the same program. Our goal is to detect the separators by analyzing one or several episodes of the same recurrent TV program. Once we have these separators, the different parts of the TV program can be delimited. The method does not need any other prior knowledge on the structure of a program or on the number of parts the program has.

## Contributions of this thesis

The main contributions of this thesis are:

• First of all, we propose an original approach for structuring TV programs, that is not specific to a certain type of program and that addresses a large category of programs like the recurrent TV programs.

• Second, the approach we propose for structuring is largely unsupervised. There is no need of any prior knowledge about the programs or about the number of parts the program has.

• Third, the proposed approach exploits the visual and the audio content of TV programs. It is based on classification techniques that filter and select *separators*, among a set of detected visual and audio recurrences.

• Finally, experiments are computed on real TV broadcasts in order to validate the ideas on which the proposed approach is based on and to test the proposed classification algorithms.

## Importance of recurrent TV programs

In order to give an idea about the importance of recurrent TV programs on a generalist TV channel we made an quantitative analysis on three different French TV channels (TF1, France2, M6). We used for classification the concepts defined in [Pol07], issued from the manual used by the librarians at INA(National Institute of Audiovisual). The main program classes are illustrated in the first two columns of the Table 1. Among them, a first class of TV Fiction refers to Films and TV Films, TV Series and Soap Operas. A second one, of Informative Programs, is composed of news, magazines and service programs (weather forecast, car traffic, lottery). The class of Entertainment Programs includes TV games, reality shows, talk shows, clip-based programs and varieties. Another class refers to sport programs and contains magazines and sport broadcasts. As we analyzed generalist TV channels, we only considered the sport magazines and we included them in the magazines category. We have also added a new class of "One time show" which represents the programs made and broadcasted only once, on the occasion of a special event.

Statistics have been made regarding the structure of the TV content provided by different TV channels (see [Pol07], the statistics published by CNC (Centre National du Cinema et de l'Image Animee), etc.). However, as our focus is only on some of the existing classes of TV programs, we have computed our own study, on one week of 3 different French TV Channels. Each day of the week was analyzed from 6:00 a.m. to midnight, as this represents the most important time interval for TV consumers. The rest of the day is generally composed of reruns of previous programs. These were not considered in this study.

We conducted the study on a per program and per duration basis. For the former case we have counted the number of programs from each category, during each day of the week. For the latter case, the duration of all the programs from each category, during each day of the week, was computed. The results are presented in Table 1 and represent the percentages for an entire week of broadcast.

As previously said, our interest is to automatically structure recurrent TV programs. It is important not to confuse a recurrent program with the rerun of a program. A recurrent

TABLE 1: TV Content Statistics based on the broadcasted programs types (percentages over a week).

| Program Type | | TF1 | | France2 | | M6 | |
|---|---|---|---|---|---|---|---|
| | | program based | duration based | program based | duration based | program based | duration based |
| TV Fiction | Films | 3,01 | 1,06 | 0,45 | 1,54 | 1,14 | 3,42 |
| | TV Films | 6,38 | 16,03 | 0,45 | 1,35 | 6,28 | 15,16 |
| | TV Series | 15,95 | 20,68 | 5,45 | 6,03 | 24 | 24,32 |
| | Soap Operas | 2,12 | 3,14 | 4,54 | 2,90 | 0 | 0 |
| Total | | 27,46 | 40,91 | 10,89 | 11,82 | 31,42 | 42,9 |
| Informative Programs | News* | 7,44 | 6,28 | 9,09 | 6,86 | 6,85 | 2,83 |
| | Magazines* | 2,65 | 3,79 | 15 | 28,95 | 8 | 12,85 |
| | Service Programs | 30,85 | 4,84 | 35,90 | 5,12 | 22,85 | 2,16 |
| Total | | 40,94 | 14,91 | 59,99 | 40,93 | 37,70 | 17,84 |
| Entertainment Programs | TV Games* | 11,70 | 14,13 | 15,45 | 17,56 | 1,14 | 0,92 |
| | Reality Shows* | 2,12 | 3,73 | 3,18 | 5,71 | 8,57 | 14,91 |
| | Talk Shows | 0 | 0 | 4,09 | 9,74 | 0 | 0 |
| | Variety* | 0,53 | 2,02 | 4,54 | 9,80 | 1,14 | 2,30 |
| | Clip-based | 0 | 0 | 0 | 0 | 6,28 | 5,47 |
| Total | | 14,35 | 19,88 | 27,26 | 42,81 | 17,13 | 23,6 |
| Documentaries | | 0,53 | 1,11 | 0,90 | 1,87 | 0,57 | 1,05 |
| Youth Programs | | 14,89 | 15,05 | 0 | 0 | 9,71 | 9,09 |
| Teleshopping | | 2,65 | 3,14 | 0 | 0 | 3,42 | 5,47 |
| One time show | | 1,06 | 3,01 | 0,90 | 2,51 | 0 | 0 |

program is composed of several different episodes broadcasted periodically. The rerun represents the retransmission of a certain episode that has already been broadcasted.

Regarding the classification we presented, most of these programs (excepting films, TV films, documentaries and some talk shows) are recurrent programs. However, the recurrent programs included in the TV Fiction category do not have a structure so there would be useless to try structuring them.

The categories that interest us most are marked with "*" in Table 1. The percentage that these programs have during an entire week of broadcast, when considering also the service programs, is presented, for each channel apart, in Table 2(a).

TABLE 2: Percentage of recurrent programs during one week of broadcast.

| | /program | /duration |
|---|---|---|
| TF1 | 24.46 | 29.97 |
| France2 | 47.27 | 68.90 |
| M6 | 25.71 | 33.82 |

(a)(+Service Programs)

| | /program | /duration |
|---|---|---|
| TF1 | 35.38 | 31.49 |
| France2 | 73.75 | 72.62 |
| M6 | 33.33 | 34.57 |

(b)(-Service Programs)

The interest of structuring programs is natural for longer programs, consequently the focus is on programs that have more than 10 minutes. Service programs (and short magazines) will thus be out of our scope. When not considering these among the programs

listed in Table 1, the percentage of the programs represented by the categories we are mostly interested in, increases as illustrated in Table 2(b). As the results in tables show, the percentage of recurrent programs and the time occupied in the TV timetables is quite important. Within 30 up to 70% of the programs broadcasted during an entire week are recurrent programs. Moreover, these programs have a clear structure with well defined main parts that could benefit of services providing browsing features.

## Description of the vocabulary employed in this thesis

Before detailing the proposed approach we explain the vocabulary used in this thesis. This section is very important as the lexicon used is quite varied and may lead to confusions among the terms used to describe the proposed approach.

In order to avoid any possible confusions and to facilitate the reading of this thesis we present in this section, in more detail, the lexicon employed for the description of the subject. The reader is invited to refer to this section whenever needed during the reading of the manuscript.

One of the terms frequently encountered is that of ***recurrent TV program***. A recurrent TV program is a program composed of several "***episodes***" that are periodically broadcasted (e.g. daily, weekly, monthly...). Examples of this type of programs are entertainments, game shows, magazines, news.

The next three denominations are often hired in the description of the proposed approach and can be easily confused. One of them refers to the notion of ***repeated/recurrent sequence***. A repeated sequence is a video sequence that repeats between different episodes or within an episode of a TV program.

Such a repeated sequence contains several ***occurrences*** that represent actually the different appearances of this video sequence within the current episode or other episodes of the same TV program.

Finally, we call ***recurrences*** all the occurrences of all the repeated sequences.

An illustrative example is in Figure 1. The colored boxes represent video sequences that repeat within 3 episodes of a TV program. Each repeated sequence is represented by a color. For example, sequence 3, represented in blue, is repeated in each of the 3 episodes. Consequently we have 3 occurrences of this sequence denoted by R13 in Episode 1, R23 in episode 2 and R32 in episode 3. All the occurrences together (meaning R11, R12, R13, R21, R22, R23, R24, R31, R32, R33, R34) can be referred to as recurrences.

The so-called ***separators*** are also very often mentioned in this thesis as these represent one of the basis of the proposed approach. Separators are video sequences that repeat within or between the episodes of a TV program and that separate the main parts composing a program (see Figure 2). If for example, the analyzed program is a TV game, the separators would delimit the different stages of the game.

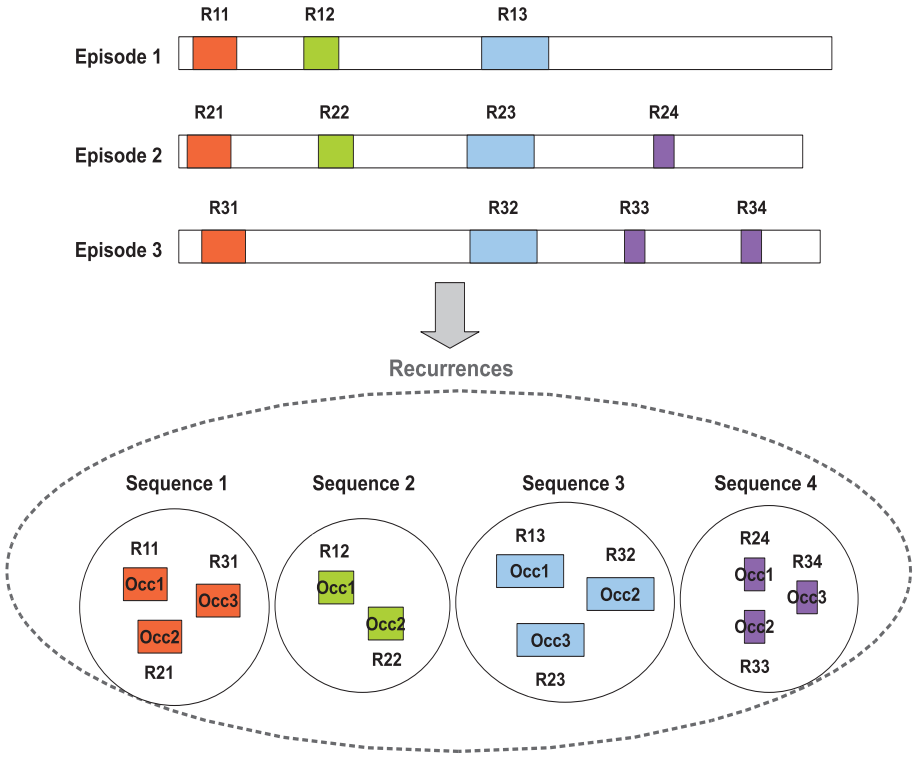



FIGURE 1: Representation of repeated sequences, occurrences and recurrences.

## Overview of the proposed approach

As previously explained, our solution focuses on detecting the structure of recurrent TV programs. We recall these are mainly programs that are broadcasted periodically like entertainment programs, TV shows or TV magazines. One of the main properties of this kind of programs is their clear and steady structure. The different parts of each episode are generally delimited by short video sequences, that we named "separators".

In Figure 2, examples of separators are illustrated on an episode of a French game show. The boxes represent the separators that delimit the main parts of the program. The 3 images are extracted from three separators of the game show.

The main idea of our approach is to detect these separators as they exhibit characteristics that make this task possible. The parts of each episode are then indirectly identified using the separators boundaries, e.g. the end of a separator is the start of a new part. The resulting structure for an episode is hence composed of a set of times. These times refer to the start and end of each part.

One of the most important property the separators have is that they are recurrent. In Figure 3 the main steps of the proposed approach are illustrated. In a first step, a set of episodes of a recurrent program are analyzed, in order to detect the audio and visual recurrences separately. The number of episodes that have to be analyzed is a key parameter that has an impact on the quality of the final obtained structuring results. It will be empirically studied.

The recurrence may be reflected inside the same episode (intra-episode recurrences),

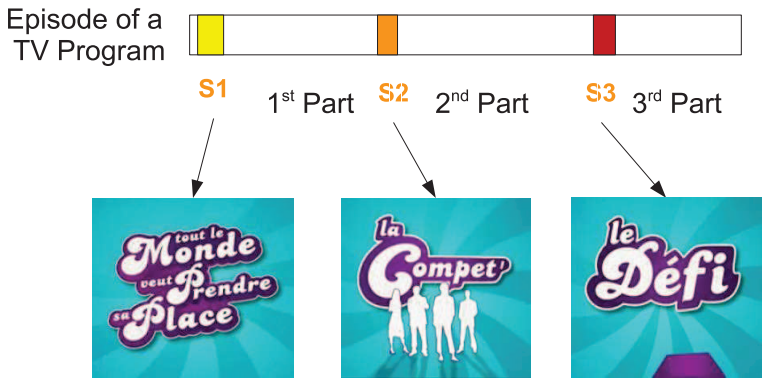



FIGURE 2: Example of video separators.

or/and between different episodes of the same TV program (inter-episode recurrences). In Figure 3, $R_{V12}$ and $R_{V14}$ are examples of intra-episode recurrences and $R_{V25}$ and $R_{Vn4}$ are examples of inter-episode recurrences.

Among the detected recurrences some are separators but not all the recurrences are necessarily separators. Consequently in a second step, a filtering process based on the temporal and spatial distribution of the recurrences is applied.

Up to here, the audio and video recurrences will be treated separately. However, we will show that each of the two approaches has its limitations. The video approach is limited to only identical recurrences and the audio one provides far too many recurrences that are not necessarily separators. Moreover, the use of audio and video recurrences together may improve the structuring results.

Therefore, in order to combine the two modalities and recover the maximum number of separators, in a third step all the recurrences resulted after the filtering are passed through a classification module. As indicated also in Figure 3, two different classifiers are considered: decision trees and SVMs (Support Vector Machines). The detected separators, are finally used to structure the input episodes. These can also be stored in a database and used later for the structuring of new coming episodes.

## Outline of the thesis

This thesis describes the different techniques that have been proposed and studied for the unsupervised structuring of recurrent TV programs. It is organized in four main chapters among which, three reflect the stages of the proposed algorithm and describe the gradual transition from the detection of recurrences to separators and program structuring.

The **first chapter** discusses the problem of TV program structuring, its applications in real-world services and provides a panorama of existing techniques. It describes in particular the different approaches that were proposed in the literature and it outlines their strengths and weaknesses.

The **second chapter** presents our method for detecting inter or/and intra-episode video recurrences. It also proceeds with the filtering of these recurrences and proposes two methods for structuring either the input episodes or new coming episodes using a database of separators. This chapter also shows the results obtained when evaluating this method.

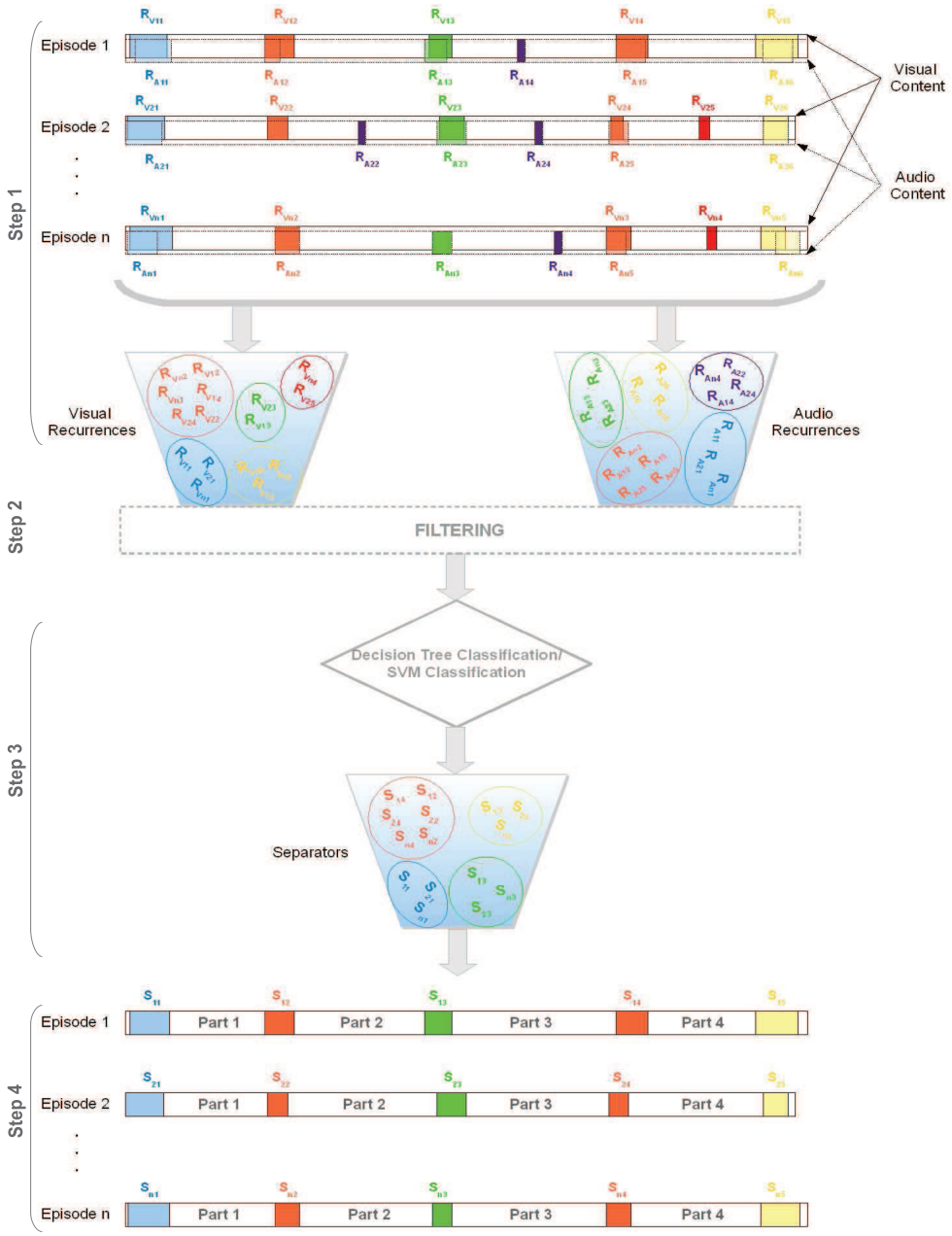



FIGURE 3: The proposed approach.

The **third chapter** describes our method for detecting inter or/and intra-episode audio recurrences. The principle is similar to the one previously presented. Three different descriptors are tested for the description of the audio content. The filtering step is also discussed for the case of audio recurrences. The method is evaluated and the obtained results are presented in the last part of the chapter.

The **fourth chapter** provides an effective classification and selection method, based on decision trees that allows the detection of as many "separator" as possible among

recurrences. All the recurrences obtained with the approaches described in the previous two chapters are passed through the classification module that decides whether a recurrence is a separator or not. The decision trees are built based on attributes issued from techniques that perform applause detection, scenes segmentation, face detection and clustering and speaker diarization and clustering. This chapter describes each of these techniques and provides also the obtained classification results. SVMs are also experimented in this chapter and used for comparison.

Finally, we conclude this thesis with a summary of the work presented and a review of some short or long term research perspectives.

# Chapter 1

# Existing methods for TV Program structuring

**Summary**



THE OBJECTIVE of the proposed chapter is to present the problem of TV program structuring, its applications in real-world services, and to provide a panorama of existing techniques.

Due to the significant growth in the amount of broadcasted TV programs, new services have emerged. Each of these services aims at making the audio-video content of TV broadcasts more available to users. Nowadays, users are no longer constrained by the linear nature of TV broadcasts. Catch-up TV and PVRs services allow them to watch most of previously broadcasted TV program. From the TV broadacats, a large number of TV Programs are extracted, stored, indexed and prepared for a later usage. An additional indexing step is however still required. After choosing a program, a user might want to get an overview of the program before watching it. He/she may also would like to have the possibility to browse inside a program and find the moments that interest him most.

Consequently, to enable these features, after being extracted, each TV program has to be structured, that is, its original structure has to be recovered and all possible moments

of interest have to be precisely tagged. In addition to advanced browsing features, TV program structuring can also be used for summarization, indexing and querying, commercial skipping, archiving, etc.

As previously stated, the first step is to extract the programs from the TV stream. This corresponds to the notion of macro-segmentation of the TV stream [BML08, Pol08, NGG06, PAO04]. Macro-segmentation algorithms generally rely on detecting "inter-programs", which include commercials, trailers, jingles and credits. These are easier to detect as they have a set of common properties related to their duration, visual and audio content. The idea is to detect these inter-programs and to deduce the long programs as the rest of the stream. Metadata (like EIT (Event Information Table) or EPG (Electronic Program Guide)) can be used afterwards to annotate the programs.

Second, to allow non-linear TV program browsing, summary generation [OSA08, OSA07, TV06], and other applications listed previously, the original structure of TV programs has to be recovered and all possible moments of interest have to be precisely tagged.

A video can be analyzed at different granularity levels. The elementary level is the image/frame, generally used to extract features like color, texture, shape. The next level is represented by shots, basic video units showing a continuous take from a camera. The shots are separated by editing effects called transitions. A transition can be abrupt, namely *cut*, and groups directly successive shots, or *gradual*, and groups together successive shots by different editing effects like dissolve, wipe, fade etc. Shot boundary detection, is the process of identifying the transitions, abrupt (cut) or gradual, between the adjacent shots. Shot boundary detection techniques were intensively studied and a lot of methods have been proposed [YWX+07]. The shot however is not a relevant level to represent pertinent parts of a program as it usually lasts few seconds and has low semantic content. A video may have hundreds or thousands of shots, which is not practical for human navigation. Therefore, high-level techniques are required to group video shots into a more descriptive segment of the video sequence. These techniques can be classified into two wide categories: *specific methods* and *generic methods*.

**Specific approaches** exploit the prior knowledge of the domain in order to construct a structured model of the analyzed video. They can be applied only to very specific types of programs like news, sports programs, series, advertisements, etc.

Authors who propose specific methods consider that a universal solution for high-level video analysis is very difficult, if not impossible, to achieve [GT02]. Low level features generally used for indexing video content, are not sufficient to provide a semantically meaningful information. In order to achieve sufficient performance, the approaches have to be specifically adapted to the application. Bertini et al. [BBP01] consider that, in order to ease the automatical extraction of high-level features, the specific knowledge of the different types of programs must be considered. Thus, many researches have focused on very specific types of programs like sports programs, movies, advertisements and news broadcasts. The methods they propose are specific. They make use of prior knowledge of the type of the analyzed TV program in order to extract relevant data and construct its structure model. They are supervised as they generally require the prior creation and manual annotation of a training set used to learn the structure. A class of TV programs that are often analyzed by specific methods is sports programs. These have a very well defined structure. The rules of the game provide prior knowledge that can be used to provide constraints on

the appearance of events or the succession of those events. These constraints are very helpful to improve the accuracy of the event detection and categorization. Another class of programs that is very appropriate for this kind of systems are news programs, as they have also a very clear structure. They are produced using almost the same production rules. They consist generally in a succession of reports and anchorperson shots. Specific methods analyze the temporal (the set, reports, advertising, forecast, etc.) and spatial structures (images with the anchorperson, logos, etc.). Most of the work relies on finding the anchorperson shots and then deduce the sequences of shots representing the reports.

We focus on these two categories of TV programs (sport and news) for the specific approaches as they are the most representative for their class and discuss them in more detail in the next subsections.

On the other hand, **generic approaches** try to find a universal approach for the structuring of videos, independently of their type and based only on their content features. These try to structure the video in an unsupervised way, without using any prior knowledge. Due to the fact that they do not rely on a specific model they are applicable to a large category of videos.

In this category, literature often reflects the importance of "*scenes*" as structural elements of a video and portrays corresponding techniques. As already mentioned earlier, the first step into building a structured description of a video is to segment it into elementary shots. A shot is the basic unit of a video document showing a continuous take from a camera. But shots are too small and too numerous to assure an efficient and relevant structure of a video. Therefore, in order to segment the video into semantically richer entities, shots need to be grouped into higher level segments, namely *scenes*. A scene is generally composed of a small number of shots all related to the same subject, ongoing event or theme [WDL+08, ZS06]. However, the definition of a scene is very ambiguous and depends on the subjective human understanding of its meaning. In the literature they are also named "video paragraphs" [HS95], "video segments" [ZS06, VRB00], "story units" [SMK+09, YY96, YL95] or "chapters" [TA09]. Existing scene segmentation techniques can be classified in two categories. The ones using only visual features and others using multimodal features. Both compute the scene segmentation either by clustering the shots into scenes based on their similarities, or by emphasizing the differences between the scenes. Even though the goal is the same, the differences appear in the choice of the parameters and their thresholds. The challenge lies thus in finding the appropriate set of features that would lead to a correct identification of the scenes in a video. An objective evaluation of these methods assumes the existence of a ground-truth at scene level. But this ground-truth is human generated so it is hard to make a reliable comparison of the performance of different approaches based on subjective judgments.

Within generic approaches, one commonly used feature in most of the recent works in the field is "*the recurrence*" [ABM11, BG11, Her06, Jac06]. Indeed, it is very frequent for programs to be composed of recurrent segments that act as anchor points in programs. Examples of such recurrences are anchor person shots in news, audio jingles that announce the passing from a topic to another in a TV/radio magazine or the passing to another stage of a TV game show. These recurrences are introduced on purpose in order to allow viewers/listeners to easily follow the program and identify its structure even if they do not watch it from the beginning.

In the rest of this chapter we discuss in more detail the different techniques that were proposed in the literature for the implementation of each of the two types of approaches, their strengths and weaknesses.

## 1.1 Approaches for sports programs structuring

Within the category of specific approaches, there are most of the sport program structuring techniques. Sports videos have been actively studied since the 1980s. Due to the fact that sports events attract a large audience and have important commercial applications, they represent an important domain of study. Sports videos are generally broadcasted for several hours. People who miss the live broadcast are often interested in the strongest moments only. Consequently, the relevant parts of the video have to be automatically selected and annotated. Sports videos are characterized by a defined structure, specific domain rules and knowledge of the visual environment. All these make possible the extraction of the video structure, allowing non-linear browsing. We distinguish two levels in the sports video analysis: segment and event. In the first case, the objective is to segment the video into narrative segments like *play* and *break* through a low-level analysis of the video. The second one, assumes a higher level analysis of the video and its objective is to identify the interesting moments (highlight events) of the sports video. Both of them are predetermined by the type of sport. For example, an event in the case of soccer might be the detection of the goal while for tennis this will be the match points. Therefore, in order to accomplish these objectives the use of prior knowledge is needed. This prior knowledge may be related to the type of the analyzed sport (i.e. game surface, number of players, game rules) but also to the production rules for the video program (i.e. slowmotion replay, camera location and coverage, camera motion, superimposed text [WDL+08]).

### 1.1.1 Low level analysis of sports videos

A sports video is composed of continuous successions of play and break sequences. As already said, the first level of semantic segmentation in sports videos is thus to identify the *play* and *break* segments. For the class of field ball games, a game is in play when the ball is in the field and the game is going on. Out of play is the complement set, when the action has been stopped: ball outside the field, score, audience, coach, play stopped by the referee, etc. Generally any sports game can be segmented into these two mutually exclusive states of the game. The interest in obtaining such a segmentation is that it allows play-by-play browsing and editing. It also facilitates the further high-level analysis and detection of more semantically meaningful units as events or highlights. It is claimed that highlights are mainly contained in a play segment [TC10]. The occurrence of audio-visual features in play-break segments show remarkable pattern for different events. For example, a goal event usually happens during a play segment and it is immediately followed by a break. This break is used by the producers to emphasize the event and to show one or more replays for a better visual experience. As a result, the goal event can be described by a pattern which can be defined using the play and break structure. Another benefit of using play-break segmentation is that it significantly reduces the video data (no more than 60% of the video corresponds to a play), as stated in [XXC+04]. Two major factors influence the sports video syntax: the producer and the game itself. A sport video benefits

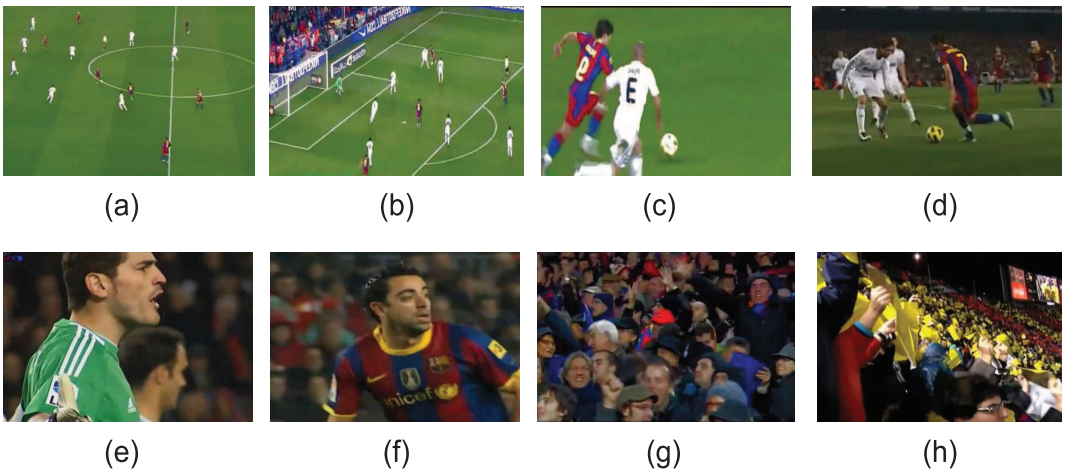



FIGURE 1.1: View types in soccer: (a,b) Long view, (c,d) in-field medium view, (e,f) close-up view, and (g,h) out of field view.

of typical production rules. Most of the broadcasted sports videos use a variety of camera view shots and additional editing effects such as replay and on screen captions to describe the sports video content. The different outputs of camera views have successfully been used for play-break segmentation. An example is in [ETM03] where shots are classified in three classes: long shot, in-field shot and close-up or out of field shot (see Figure 1.1). A long shot displays the global view of the field hence it serves for localization of the events in the field. An in-field medium shot is a zoomed-in view of a specific part of the field and it contains usually a whole human body. A single isolated medium length shot between two long shots corresponds to a play while a group of nearby medium shots corresponds to a break in the game. The replays are also considered medium shots. A close-up shot shows the view of a person and in general indicates a break in the game. Also, out of field shots that show the audience, the coach, etc., indicate a break in the game. The classification is made automatically based on the ratio of grass color pixels (G) for the close-up and out of field shots which have a small value of G. Because the limit between medium or long shots is questionable for the frames with high value of G, a cinematographic algorithm (Golden Section Composition) is used in order to classify them as long view or in field medium shots. An analysis of frame to frame change dynamics is performed in order to detect the replays. The same approach is used in [TC10] where play-break sequences are segmented using the output from camera view classification and replay detection. The camera view classification is performed on each frame using the playing field (or dominant) color ratio which measures the amount of grass pixels in the frame. As replays are often recognized as play shots because of the global view, even though they should be included as breaks (as they are played during breaks), replay detection is applied to locate additional breaks and to obtain more accurate results. The algorithm is tested on soccer, basketball and Australian football videos.

In [DXC+03], Duan*et al.*, use a mid-level representation, between low level audiovisual processing and high level semantic analysis. First, low level visual and audio features are extracted and a motion vector model, a color tracking model and a shot pace model (that describes the production rules effect on the shot length) are developed. These models are

then used in a supervised learning algorithm in order to classify shots into predefined classes, e.g. Player Close-up, Field View, Audience. SVM is also used to classify game specific sounds e.g. applause, whistling. The semantic classes are specific to the nature of the analyzed sport and are further used to construct a temporal model for representing the production rules of the sport. The sports video shot sequences are partitioned into two logical segments, namely, *in play segments* (IPS) and *out of play segments* (OPS), that occur in successive turns. Based on the temporal model and the semantic shot classes, the in play and out of play segments are determined by checking the shot transition pattern. The approach has been tested on five field ball type sports. In Xie *et al.* [XXC+04] a stochastic method based on HMMs is proposed for play-break sequence detection. As in previous approaches, dominant color ratio is used, this time along with motion intensity in order to identify the view type of each frame of a soccer video. Motion intensity gives an estimate of the gross motion in the whole frame including object and camera motion. A wide shot with high motion intensity often results from a play, while static wide shot usually occurs during the break in a game. Six HMM models are created for the respective game sequences. Using a sliding window, the likehood for each of the pre-trained (play and break) models is retained. A dynamic programming algorithm is used for the temporal segmentation depending on the transition probabilities between the play-break and the likehood of each segment to belong to one of the two classes (play/break). The experimental evaluations showed improvement of the performance of the HMM based algorithm compared to the discrete rule based one studied in a previous work of the author [XXC+01]. In the last cited one, heuristic rules were used in detecting the view type of the sequences in order to obtain play-break segmentation. The rules took in consideration the time duration of each view and their relative position in time.

HMMs can also be modeled in a multistream approach [XMZY05] to classify shots of soccer or volleyball videos. A multistream is obtained by combining multiple single stream HMMs and introducing the weight for each stream. For soccer videos, the first stream is the motion feature vector and the second is the color feature vector of each frame. For volleyball videos the processing is similar excepting the used features.

### 1.1.2 High level analysis of sports videos

Highlight events in sports videos are composed of the most interesting moments in the video that usually capture the users attention. They represent the exciting scenes in a game, i.e. the goal in a soccer match, the tennis match points or the pitches in baseball. A lot of research activities have focused on the detection of interesting events in sports videos, most of it employing rule-based annotation and machine learning or statistics based annotation. The modalities used to acquire the semantics are either individual or multiple modalities. The main sources of information used for the analysis are the video track and the audio one (because crowd cheers, announcer's excited speech, ball hits are highly correlated with the exciting segments). Also, editing effects such as slow motion replay, or close caption information that accompanies the sports program and that provides information of the game status, are used for detecting important events. Within each of these, the features used for the program segmentation may be general (i.e. shot boundary detection) or domain specific (i.e. the center line on a soccer field is vertical or almost vertical in frame [YLL09]). As the highlights are specific to each type of sport, most of the

approaches make use of sport specific features. The features used for semantic analysis of sports videos can also be considered as cinematic or object-based features. The cinematic ones are referred to those that result from the production rules , such as camera views and replays. Sport video production is characterized by the use of a limited number of cameras on fixed positions. The different type of views are very closely related to the events in the video and for each type of sport a pattern can be identified. As an example, during a rally in a tennis video, a global view of the camera, filming the entire court is selected. Right after the rally, a close-up of the player that just scored a point is captured [Kij03]. In the soccer videos a goal is indicated by a close-up on the player who just carried out the important scene, followed by a view of the audience and a slow motion replay. Moreover, starting from close-up shots, the player's identity can be automatically annotated using face recognition techniques and based on textual cues automatically read from the player's jersey or from text caption [BBN06].

For the case of object-based features, they are used for high-level analysis of sports videos. Objects are described by their color, texture, shape and motion information. Using even more extensive prior knowledge, they are localized in different frames and their motion trajectories are identified and used for the detection of important events. As example in diving videos, player body shape segmentation and transition is used to represent the action [LTW+10].

As a result, high-level analysis in sports videos can be done based on the occurrences of specific audio and visual information that can be automatically extracted from the sports video.

### 1.1.3 Rule-based approaches

Many of the existing works on event detection use rule-based approaches to extract a predictable pattern and construct, manually or automatically, a set of rules by analyzing the combination of specific features. In order to analyze different sports, Duan *et al.* [DXC+03] used a mid level representation, between low level audiovisual processing and high level semantic analysis. As described in the previous subsection, predefined semantic shot classes are used to facilitate high level analysis. Based on production rules, a general temporal model is developed and coarse event (in play)/non event (out of play) segments are identified. Richer events are classified in a hierarchy of *regular events*, which can be found in the transitions between in play and out of play segments, and *tactic events* that usually occur inside the in play segments. To identify the regular events, the semantic shot classes and audio keywords are exploited in order to derive heuristic rules according to game specific rules. For example, a soccer goal occurs if there are many *close-up* shots, *persistent excited commentator speech* and *excited audience*, and long duration within the OPS segment. For tennis, events like serve, return, score, etc. are detected by analyzing audio keywords like *hitting ball* and *applause* to compute the ball hitting times and the intervals between two ball hits, and the applause sound at the end of court view shots. Because of the use of audio keywords, the precision in detecting some events (i.e. goal for soccer) can be low due to the confusion from the loud environmental audience sound. Tactic events are strongly dependent on the game specific knowledge and are detected using object features. To identify *take the net* and *rally* in tennis games, the player tracking is employed. Player tracking and its movement over time is also used by Kim *et al.* [KGS+10] to detect the locations

where the play evolution in soccer games will proceed, e.g. where interesting events will occur. A ground level motion of players at each step is extracted from individual players movement using multiple views and a dense motion field is generated. Using this field the locations where the motion converges are detected. This could represent an important application for automated live broadcasting. Player localization is viewed as a K partite graph problem by Hamid *et al.* [HKG+10]. Nodes in each partite of the graph are blobs of the detected players in different cameras. The edges of the graph are a function of pair wise similarity between blobs observed in camera pairs and their corresponding ground plane distances. The correspondence between a player's blob observed in different cameras is treated as a K-length cycle in the graph. Yu *et al.* [YLL09] use players motion intensity, along with other object features and low level features to rapidly detect the boundary of interesting events indicated by the gamelog. The instant semantics acquisition and fast detection of event boundaries is important for the two interactive broadcast services for live soccer video they propose. The first one is a *live event alert* service that informs mobile viewers of important events of a live soccer game, by providing a video clip of the event with a time lag between 30s and 1.5 min. The second one is *the on-th-fly language selection* and allows users to choose their preferred contents and preferred language. To improve the accuracy of event boundaries detection, object features are used. The center line of a soccer field is detected based on the knowledge that this line is vertical or almost vertical in every frame. The goal mouth is also detected based on the prior knowledge : the two posts are almost vertical, the goal posts and goal bar are bold line segments and compared with the center line and side lines and the goal posts are shorter line segments. Gueziec [Gué02] tracks also specific objects, as the ball during pitches in baseball games, to show as replays if the strike and ball decisions were correct. The implemented system was launched in 2001 during *ESPN's Sunday Night Baseball* and it tracked pitches with an extremely low failure rate. The real time tracking involves the use of extensive prior knowledge about the game and system setup (i.e. camera locations and coverage). Considering that object based features are computationally costly, Ekin *et al.* [ETM03] try to use more cinematic features to detect certain events in soccer videos and employ the object based ones only when needed, to increase the accuracy. Therefore, after classifying shots and segmenting the video into play/break segments(see previous subsection), events like goal, red-yellow cards and penaltys are detected. For goal detection, a pattern is computed using only cinematic features, resulted from common rules used by producers after goal events. The red-yellow cards are indirectly identified by locating the referee by its distinguishable colored uniform. The penalty box is detected based on the *three-parallel-line rule* that defines the penalty box area in the soccer field. All these approaches use an extensive prior knowledge of the game and production rules. Most of them are based on manual observation and heuristic knowledge. In order to reduce the amount of knowledge used for sports video analysis and make the framework more flexible for different sports, Tjondronegoro *et al.* proposed in [TC10] a knowledge-discounted event detection approach in sports videos. The highlights contained in each play/break (P/B) sequence are classified using a set of statistical features (e.g. sequence duration, break ratio, play ratio, replay duration, near goal ratio, excitement ratio, close-up view ratio) calculated from the P/B sequence. During a training phase, each of the predefined events were characterized using the set of statistics and the heuristic rules were constructed. To classify which highlight is contained in a P/B segment, a score is used for each statistical feature. The value of each

calculated feature is compared to the trained one, and if the value falls within the trained statistics of the highlight, the corresponding score is incremented. The highest score will indicate the most likely highlight contained in the P/B segment.

### 1.1.4 Machine learning based approaches

Other approaches that try to use a modest quantity of prior knowledge, only needed for selecting the features that match the most with the event, are approaches based on machine learning [CC08, BPPC12]. For example Hidden Markov Models, that have been proven to be effective for sequential pattern analysis, can be used to model tennis units like *missed first serve*, *rally*, *break* and *replay*, based on the temporal relationship between shots and with respect to editing and game rules [KLP03]. Visual features are used to characterize the type of view of each shot. HMMs are also utilized to recognize the action and index highlights in diving and jump game videos [LTW$^+$10]. The highlight in diving videos is described by the entire diving action. As the action occurs many times in the video and the distance between two action clips is regular, these are detected by motion segmentation and a hierarchical agglomerative clustering. Then the action is recognized based on body shape segmentation and shape transitions modeled by continuous HMMs. The Baum-Welch algorithm is used to train the HMMs and the Viterbi algorithm is used to calculate each model's output probability. The model with the maximum probability is the recognition result. The HMMs can also be modeled in a multilayered approach to detect events like *offence at left/right court*, *fast break at left/right court*, etc., in basketball videos [XMZY05]. Instead of detecting and tracking objects, the statistical learning approach is used to build semantic models based on a set of motion features. HMMs are also used to model audio sequences. In [PWY04] the plopping sound is modeled with HMMs in order to identify the highlights in diving sports videos. As it has been proven that browsing highlight with contextual cues together is preferred, slow motion replay and game statistics information in superimposed captions are extracted. It might happen that announcer's exciting speech and audience applause to exceed the plopping sound, making it hard to take a correct decision. Special sound effects like announcer's excited speech (detected using learning machines) and ballhits (detected using directional templates) have also been employed to detect highlights in baseball programs [RGA00]. A probabilistic framework is used to combine the two sources of information. If an audio segment has a high probability to be an excited speech segment and occurs right after a frame that has a high probability to contain a baseball hit, than it is very likely for the segment to be an exciting (highlight) segment. Audio and video markers together are used in [XZT$^+$06] to detect highlights in sports videos like soccer, baseball and golf. A Gaussian Mixture Model has been generated for each predefined audio class (applause, cheering, commentator's excited speech, music) and used to identify the audio markers. For the visual ones, an object detection algorithm such as Viola and Jones has been used to detect the pattern in which the catcher squats waiting for the pitcher to pitch the ball in baseball games, the players bending to hit the golf ball in golf sports and appearance of the goal post in soccer games. Other approaches for event/highlight detection in sports videos propose the use of neural networks to classify shots into predefined classes and utilize them for further processing [ABCB02], or AND-OR graphs to learn a storyline model from weakly labeled data using linguistic cues annotations and visual data [GSSD09].

## 1.2 Approaches for news programs structuring

News programs structuring techniques also rely on specific approaches. Compared to the sports programs, the work related to news programs is much more extensive and dates for a long time. This might be partially due to their regular structure which makes the analysis much easier. Indeed, most news videos exhibit a similar and well defined structure (Figure 1.2). They usually start with a general view of the set during which the anchor-

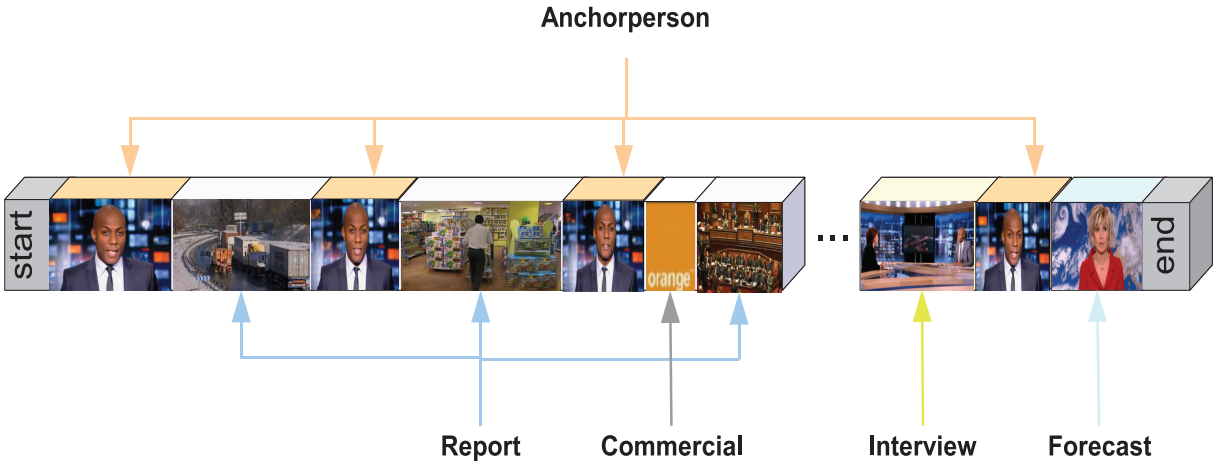



FIGURE 1.2: The structure of a TV news program.

person announces the main titles that will be developed later (the highlights). The rest of the program is organized as a succession of TV reports (stories) and segments where the anchorperson announces the next topic. Each story generally begins with an anchorperson segment that provides the general description of the event and continues with a more detailed report and sometimes interview segments. At the end of a story an anchor segment may reappear to give a summary or conclusion. Most news broadcasts end with reports on sport, weather and finance. Commercials may also appear during the broadcast.

### 1.2.1 Shots classification

The fundamental step in recovering the structure of a news program is thus to classify the shots into different classes (anchorperson, report, weather forecast, interview etc.). Then, based on these classes, video segmentation into story units can be performed.

Detecting anchorperson shots plays an important role in most of the news video segmentation approaches. The anchorperson shots exhibit a lot of characteristics that make their detection easier. Early approaches used properties like the fact that the same anchorperson appears during the same news program and the background remains unchanged during the entire shot. They also considered that anchorperson shots are usually filmed with a static camera so the anchorperson will always be situated in the same place of the image. Furthermore, each TV channel has representative semantic objects, like logos or captions with the anchorperson name, that are displayed only during news program. All of these, made possible the detection of anchorperson shots by **template matching techniques**, based on weighted similarity measures and model images.

Zhang *et al.* [ZGST94] proposed a spatial structure model for the anchorperson frame, based on a composition of distinctive regions: the shape of the anchorperson, the caption of reporters names and the logo of the broadcast channel that appears in the top right part of the image. Recognizing an anchorperson shot involves testing every frame over a frame model which in turn means testing each frame against the set of region models. However, constructing a set of models images is difficult due to variations from one TV station to another and the matching algorithm is time consuming.

Gunsel *et al.* [GFT98] proposed also semantic object detection and tracking (i.e. faces and logo) using a region matching scheme, where the region of interest is defined by the boundary of the object. The detection of anchorperson shots lies here on a face detection technique, highly used in this purpose. The possible regions where anchorpersons might be situated are extracted using skin detection (color classification) and histograms intersection. These regions are then compared to templates stored in the application database. The prior knowledge of the spatial configuration of the objects included in these shots is used to limit the region of interest during skin detection. The system has some limitations, as it is difficult to track occluded or transparent semantic objects. Also, the interview shots can be confused with anchor shots if the interviewed person has the same spatial position as an anchorperson. In [KX08], once faces have been detected, procedures including face position analysis, size filter and dress comparison are used to reduce the possibilities of erroneous identification. In [ATK00] a robust face detector approach is implemented by means of color segmentation, skin color matching and shape processing. Heuristic rules are then used to detect anchorperson, report/interview and outdoor shots (e.g. one or two face close-ups with a static background are classified as single or double anchor). Uncontrolled illumination conditions can interfere with skin-color model, making the classification rates smaller for report/interview shots.

In order to avoid using a model for the anchorperson shot recognition, another property of the anchorperson shots is exploited by researchers, for detecting these shots: this is their recurrent appearance during an entire news broadcast. Based on the fact that shots of the anchorperson are repeated at intervals of variable length and that their content is very similar, Bertini *et al.* [BBP01] propose a **statistical approach**. They compute for each shot the *shot lifetime*, as "the shortest temporal interval that includes all the occurrences of a shot with similar visual content, within a video". The shots that have the lifetime greater than a threshold are classified as anchorperson shots. Based on the assumption that both the camera and anchorperson are almost motionless in anchor shots, the statistical approach is refined by the use of motion quantity in each shot. A subclassifications of anchorperson shots (like *weather forecast*) is obtained considering the speech content of anchorperson shots. High level information about the shots being broadcasted is also extracted from close captions text. The frequency of occurrence of anchorperson shots and their similarity is also used in [Pol07] where a clustering method is used to group similar key frames in two groups. The smallest group will represent the anchorperson group. The neighboring shots that have key frames with the same label are merged together.

A different approach based on frame statistics, models frames with HMMs [EWIR01]. Feature vectors deriving from color histogram and motion variations across the frames are computed and used to train the HMMs. A HMM is built for each of the predefined six content classes (i.e. newscaster, begin, end, interview, weather forecast, report) and four editing effect classes. Frames are then classified by evaluating the optimal path resulted

from the Viterbi algorithm. The system has real time capabilities as it works three times faster than real time.

Other approaches use SVM to identify anchorperson shots. In [MHG+10] the first anchorperson shot in the video is detected using key frames frequency and similarity. For each shot, a region of three other neighboring shots on both sides is taken into account for features computation. An SVM is then trained to identify other anchor shots, using features like distance from anchorperson template, semantic text similarity, shot length distance, average visual dissimilarity and minimum visual dissimilarity between the shots of left and right region. Chaisorn *et al.* [CC06, CCL03, CCL02] defines thirteen categories (e.g. anchor, 2anchor, interview, sport, finance, weather, commercial) to cover all essential types of shots in typical news videos. First, commercials are eliminated using a heuristic approach based on black frames, still frames,cut rate, and/or audio silence. Weather and Finance are then identified using a histogram matching algorithm. Finally, a Decision Tree is employed to perform the classification of the rest of the shots using a learning based approach and temporal (audio type, motion activity, shot duration) and high level features (human faces and video text) extracted from video, audio and text. The same principle is used in [FZF06]. As anchorperson is one of the most important shot categories, they add a similarity measurement module that uses the background similarity detection to reduce the errors that might appear when identifying the anchorperson shots (anchor shots can easily be confused with speech/interview as they all have similar face features). Thresholds need to be defined for each tested broadcast channel. Gao *et al.* [GT02] classify video shots into anchorperson shots and news footage shots by a graph-theoretical cluster (GTC) analysis. Since anchorperson key frames with identical model have the same background and anchorperson, they thus have similar color histogram and spatial content. Based on this similarity the anchorperson key frames of each model will be grouped together into subtrees of the graph and distinguished from the individually distributed news footage shots.

### 1.2.2 TV news story segmentation

Once the shots are classified, a more challenging task is to segment the broadcast into coherent news stories. This means finding the boundaries of every story that succeeds in the video stream. Using the prior knowledge on the structure of a news broadcast and the classified shots, Zhang *et al.* [ZLCS04] and Gao *et al.* [GT02] use a temporal structure model to identify the different stories in the broadcast. Ko *et al.* [KX08] extracts boundaries of anchorperson and non-anchorperson shots and use them next to metadata from the video structure to classify the news segments into six predefined categories (e.g. opening animation, headlines, preview, anchorperson greeting, news stories, weather forecast, closing scene). As TV channels have different programming and scheduling formats for the news videos, before performing the categorization, metadata from every channel need to be determined by separately analyzing each structure. In [GFT98] consecutive anchorperson-news footage shots are combined into *news units* according to a set of predefined rules. For example, *the anchorperson shot(s) followed by a commercial are combined with the last news unit, the news footage shot(s) following a commercial are considered a part of the next news unit*, etc. In [NK97] Dynamic Programming is used to detect *relevant video segments (important situations)* by associating image data and language

data. For each of the two clues, several categories are introduced. Inter-modal coincidence between the two clues indicates important situations. A considerable number of association failures are due to detection errors and time lag between close caption and actual speech. Other approaches use Finite State Automatons [KCtK$^+$02, MMM97] or HMMs to model shot sequences and identify story boundaries. Chaisorn *et al.* [CCL03, CCL02] represent each shot by a feature vector based on the shot category, scene/location change and speaker change and models it with ergodic HMMs. A similar approach is used in [FZF06]. In addition, a pre-segmentation module based on heuristic rules is added to join together intro/highlights shots or weather shots that can be seen as a single story logical unit. Because in the output of the analysis using HMMs in [CCL03, CCL02] there were embedded pattern rules, in [CC06] a global rule induction technique is used instead of HMMs. Pattern rules are learned, in a form similar to if-then-else syntax, from training data. The results are slightly lower compared to the use of HMMs but this method has reduced computational cost and complexity. Text is also highly used for story boundaries detection in news videos. An example is in [PWR03] where a fully automatic television news summarization and extraction system (ANSES) is built. Shots are merged into story units based on the similarity between text keywords , extracted from the teletext subtitles that come along with the broadcast. The assumption made is that story boundaries always coincide with shot boundaries. Keywords are extracted from the subtitles and tagged with their part of speech (noun, verb, etc.). Each type of word has a score. A similarity measure is computed and each text segment is compared to each of its five neighbors on either side. When the similarity exceeds a certain threshold, the segments and their corresponding video shots are merged. In [MHG$^+$10] closed captions text along with video stream is used to segment TV news into stories. For text, Latent Dirichlet Allocation (LDA) model is used to estimate the coherence of a segment and thus provide its boundaries. The assumption made is that *each document is represented by a specific topic distribution and each topic has an underlying word distribution.* In this manner, a coherent segment (containing only one story) will have only a few active topics, while a non coherent segment (that contains more than one story) will have a comparatively higher number of active topics. Using the same vocabulary as for the training corpus and a fixed number of topics, the likehood of a segment is computed and used as a score for performing the text segmentation task. For the video based segmentation, anchorperson shots are identified (as described in the previous subsection) and story boundaries are identified based on the assumption that a story always begin with an anchorperson shot. The two detected boundaries from both approaches are fused using the "or" operator in order to improve the accuracy of story segmentation. The main drawback of the approach is that words that did not appear during training are dropped and whenever an anchorperson shot has been missed, the story boundary will also be missed.

## 1.3 Approaches for scene segmentation

Within generic approaches, a lot of works has developed on the use of scenes as structuring elements.

### 1.3.1 Visual Similarity-Based Scene Segmentation

A first group of scene segmentation approaches is based on the visual similarity between the shots of the video document. In this manner, Yeung et al. [YL95] use two metrics of similarity based on color and luminance information, to match video shots and cluster them into scenes. From these metrics, a dissimilarity index is derived and based on it, a proximity matrix for the video shots is built. The algorithm first groups the pairs of shots that are the most similar and then proceeds to group the other shots by their proximity values (dissimilarity indices). The main drawback of this approach is that it may happen that two shots belonging to different scenes are found visually similar and thus grouped into the same cluster (e.g. several scenes can take place in the same room, or several shots show the same person but were taken far apart in time). Therefore, the visual similarity of shots alone is not sufficient to differentiate the context of a shot and to produce a good structure of the video. In order to overcome this difficulty, time constrained clustering was proposed. The general idea is to group successive shots into meaningful clusters along the temporal dimension so that all the shots in a scene are relevant to a common event [WDL+08]. Different approaches have been proposed in the literature, most of them are based on the principle of Scene Transition Graph (STG). Yeung et al. [YY96] uses a STG in order to segment videos into story units. The nodes are clusters of visually similar shots and the edges indicate the temporal flow of the story. A fixed temporal threshold is used to delimitate distant shots. In this way only shots that fall within the time window can be clustered together. An edge exists between two nodes only if there is a shot represented by the first node that immediately precedes a shot represented by the second node. The *cut edges* are used in order to partition the graph into disjoint subgraphs. Each subgraph represents a story unit. The variation of clustering parameters (e.g. time window parameter) is also discussed. Rasheed et al. [RS05] exploits the same idea of a STG. They construct a weighted undirected graph called shot similarity graph (SSG). Each node represents a shot. Edges between the shots are weighted by their color and motion similarity and also with the temporal distance. Scene boundaries are detected by splitting this graph into subgraphs, so as to maximize the intra-subgraph similarities and minimize the inter-subgraph similarities. The choice of the values for the visual dissimilarity threshold and temporal parameter can generate over/under segmentation depending on the length and type of video. This problem is discussed in [ZWW+07] where a similar approach is implemented but moreover, the temporal parameter is estimated as depending on the number of shots in the video. When the shot number is large, the temporal parameter should increase so to avoid the over-segmentation and vice versa.

In [ZS06] another approach for scene segmentation is proposed, based on Markov Chain Monte Carlo (MCMC) algorithm. Based on the visual similarity between all pairs of shots in the video, each shot is assumed to have a likehood of being declared a scene boundary. The scenes boundaries are first initialized randomly and then automatically updated based on two types of updates: diffusion (shifting of boundaries) and jumps (merging or splitting

two adjacent shots). Finally, the shots with the highest likehood in their neighborhoods are declared as scene boundary locations. The method does not require the use of any fixed threshold (resolving the problem of over/undersegmentation).

As montage and cinematic rules are widely used by producers to put shots together into coherent stories, Tvanapong *et al.* [TZ04] introduces a more strict definition of the scene based on continuity editing techniques in film literature. First, visual features are extracted from two predetermined regions of the keyframes. These regions were carefully chosen to capture the essential area of frames according to the scene definition. A "background" region will be used to identify shots in the same setting and a "two corners" region to detect events like the traveling event. For each keyframe a feature vector is computed. Guided by the strict definition and predefined editing rules (background criterion, upper corner criterion, lower corner criterion), the extracted features are compared in order to cluster together shots of the same scene.

### 1.3.2 Multimodal Similarity-Based Scene Segmentation

A second category of methods use the combination of features extracted from video, audio and/or text, in order to segment videos into scenes. Thereby, in [PLE01] a scheme for identifying scenes by clustering shots according to detected dialogs, similar settings and similar audio was developed. Shots are recovered from the video and values for each semantic feature are calculated: background and cuts are identified for audio, frontal face detection is used for recovering the shot/reverse shot pattern for dialog determination, color and orientation for settings determination. Shots are merged into scenes based on computed distances with respect to each feature. This results in different types of scenes depending on the underlying feature. The scenes of different types are also combined to construct better setting scenes, by merging the clusters that overlap into clusters of maximum size. Difficulties were encountered in audio sequence detection based on speech. The experiments showed also that the merging of all the features does not significantly improve the performance.

In [CMPP08] a different approach, based on video and audio attention is proposed to automatically detect scenes in videos. Attentive video features are extracted and used to segment the video into shots. Scene changes are detected based on the Euclidean distance among attentive audio features of two neighboring shots. The shots whose audio distance is lower than a threshold are merged into a scene.

In [SMK+09] two multi-modal automatic scene segmentation techniques are proposed (audio and video), both building upon the scene transition graph. First, a visual STG is created as in [YY96] and the cut edges are identified as the set of scene boundaries. Second, audio analysis is employed and a similar audio based STG is constructed, in parallel to the video STG. The technique proposed for combining the results of the two graphs involves the creation of multiple video and audio graphs using different construction parameters each time. A measure of confidence is computed for each boundary between shots, which was identified as scene boundary over the total number of generated video STGs and separately on the total number of audio STGs. These confidence values are linearly combined and all shot boundaries for which the resulted confidence value exceeds a threshold will form the set of scene boundaries. The approach was tested over only three documentary films and research still needs to be done for the optimization of the weights

controlling the combination of the audio and visual graphs. In [GWND07] audio and visual features are extracted for every visual shot. Using these features and a Support Vector Machine (SVM) each shot boundary is classified as scene change/non-scene change. The drawback of this method it that it requires the availability of sufficient training data.

Based on the fact that scenes in videos are constructed by producers based on some cinematic rules, Wang *et al.* [WC03] use the concept of continuity to model these rules and extract the scene boundaries. After segmenting the video into shots, the framework successively (from lower level to higher level features) applies the concept of visual, position, camera focal distance, motion, audio and semantic continuity to group into scenes the shots that exhibit some form of continuity. An example of relation between cinematic rules and continuity might be that "visual continuity exists between shots with similar background" but "similar background models the scenes that happen at the same time and in the same location". For the case of audio continuity, "the same scene should possess the similar environment sound and dialog by the same speakers, especially for dialog scenes". The framework is tested using the first three levels of continuity to extract the scenes defined using most common cinematic rules.

Yamamoto *et al.* [TA09] propose another approach for semantic segmentation of TV programs based on the detection and classification of corner subtitles. These are considered as indicating the structure of a program based on the fact that they stay on the screen and switch at semantic scenes changes. Corner subtitles with similar features are grouped as relative subtitles, based on their color, location on the screen and segment location on the time axis. Chapter points are then detected according to the distribution of the relative subtitles. Three patterns of the program are manually defined according to broadcasted TV programs.

## 1.4 Approaches based on recurrence detection

More generic tend to be the approaches based on the "**recurrence**" of certain audio/visual sequences. Indeed, it is very frequent for programs to be composed of recurrent segments that act as anchor points in programs. Examples of such recurrences are anchor person shots in news, audio jingles that announce the passing from a topic to another in a TV/radio magazine or the passing to another stage of a TV game show. These recurrences are introduced on purpose in order to allow viewers/listeners to easily follow the program and identify its structure even if they do not watch it from the beginning.

In this sense, in [Jac06] the video self-similarity is exploited to identify the frequent patterns occurring in the video signal. The approach focuses on the detection of anchor shots in news and magazines. A self similarity matrix is computed based on the similarity measure between pairs of time points in the video. The similarity considers the color distribution and the presence of faces in the analyzed frames. A K-means clustering algorithm is used to find patterns in the matrix. The approach needs however as input the number of expected patterns.

Short video repeats are also identified in [YTX07] and used to discover and model syntactic structure of news videos. Unknown video repeats and known video clips of arbitrary lengths are detected using two detectors in a cascade structure. The first detector performs similarity analysis on abstracted form of video to detect repeats with a reasonable size of errors, while the second detector performs more accurate classification on candidate repeat

clips based on their full frames. A reinforcement learning approach is adopted to efficiently maximize detection accuracy. Unknown repeats are detected from one input video through self-similarity analysis, while when searching for known video clips cross-similarity analysis is performed between two videos. LSH (locality-sensitive hashing [GCP12]) is used to reduce repeat searching complexity.

In [PGGM04] repeated sequences of video clips are identified in real-time. The collections of detected clips are then merged into stories and monitored in subsequent news broadcasts. First, the input video is digitized at full frame rates and color moments are computed. The color moments information is then used to partition the video signal into video clips using a temporal segmentation algorithm. To retrieve potentially similar clips, frame level and clip level hashing is performed. These are then filtered and compared in detail to identify matching video clips. Information about repeated video sequences is then stored and used in topic tracking.

Repeated audio-visual sequences are detected and used for TV broadcast macrosegmentation in [BML08]. The detection of repeated sequences allows the identification of inter-programs like commercials, jingles, credits. These are then used to segment TV broadcasts and extract useful programs. The proposed method is unsupervised and relies on a micro-clustering technique that groups similar visual descriptors. The clusters are then analyzed and the repeated sequences are detected.

Audio recurrences are also detected in radio broadcast streams. In [BCR07] for example, a new type of fingerprint dedicated to audio identification and based on sinusoidal modeling is used to identify the jingles of a French news radio. A number of 243 jingle occurrences are tested, among which 33 correspond to jingles used to separate different topics. The obtained recall, in terms of occurrence, varies between 83% and 97%, when altering with different kinds of distorsions, or not, the corpus. Audio jingles are also detected in [PAO04] but the idea is not to do a topic segmentation but to propose an audio macrosegmentation by finding the temporal structure of broadcast program. The possible candidates are detected using an Euclidian distance and are afterwards validated with heuristic rules. However, reference paterns of the searched jingles are needed.

Repetitions that have the similar melody in music and songs are detected and used for structure analysis in [LWZ04]. Possible applications are music summarization, indexing and retrieval. Constant Q transform is used to extract features and a novel distance measure that emphasis more on melody similarity is proposed. From a computed similarity matrix, all significant repeating patterns are extracted using an adaptive method. The musical structure is further analyzed based on some heuristic rules.

Music summarization is also performed in [LC00] where "*Key Phrases*" are detected and used in the summary. Mel-cepstral features are used to parametrize the song. These features are then used to discover the structure of the song. The idea is to label each frame of the song such that frames that are similar have the same label. Two approaches have been used to determine the labels: top-down clustering based on the modifies KL distance and unsupervised learning of Hidden Markov Models. Assuming that the most interesting parts of a song are those that occur most frequently, the sequences labeled with the most frequently occurring labels were considered for the summary. Chai [Cha06] also performs segmentation and summarization of music based on recurrent structural analysis. Feature vectors and a distance matrix are computed to find matches between utterances. A matching function is defined and a repetition of a segment will correspond

to a local minimum in this function. The obtained segments are processed and labeled starting from the most frequent label. Different methods are presented for the choice of the most representative segments to be included in the summary.

In [PG08] repeating patterns but within speech signal are detected. Similar words or short phrases are discovered in an unsupervised manner: first, similar segments are retrieved using a segmental dynamic time wrapping technique. Second, the detected segments are grouped into homogeneous clusters each representing a pattern/lexical entity.

In [MGB09] frequently audio patterns are also detected in radio broadcasts in order to extract meaningful information that could serve for audio summarization or to speed up the access to relevant parts of the data. A dynamic time wrapping technique is used to discover repeating words. A library of repeated motifs is incrementally built and updated as the incoming stream is received. Based on the assumption that in real streams frequently occurring patterns are likely to repeat in a relatively short time span, short term occurrences are first discovered in the close future and the long term ones by library search. Besides the desired occurrences, different kind of shortcomings or errors are detected as the breathings between words, silences, similar but not identical words. The approach is also strongly speaker dependent. The principle is inspired from [Her06] where audio repetitions in audio streams are detected by time correlating low-dimension audio representations. In this case, video repetitions are used to validate the audio ones. An object repeats only if it has the same audio and visual content. As explained for the previous approach, once a repeating object (RO, as the author calls the audio repetitions) found, it is added to a library. When searching for ROs the library is checked first and then the reminder of a considered buffer, only if no match was found in the library. The boundaries of ROs are detected by aligning the waveforms of the streams for the detected copies. The points where the two streams diverge are the endpoints of the object.

Audio and video consistency of repeated segments is used in [BG11] where an event mining technique is proposed. First, a clustering method is applied for each modality separately. Then, a measure of audiovisual consistency between the clusters resulted from the two modalities is used to select the candidate events. Finally a trained SVM classifies each candidate event as corresponding to the considered event or not. Knowledge about the targeted event needs to be used in order to select relevant events for structure analysis, using an SVM classifier.

## 1.5 Conclusion

As stated in the introduction, TV program structuring is very important for a better understanding and an efficient organization of the video content. It facilitates tasks like video indexing, video classification and summarization and provides fast and nonlinear access to relevant parts of the programs (nonlinear video browsing). Its domain of applicability is very vast: in sports videos for the detection of moments of interest (e.g. goal in a soccer match, pitches in baseball) and also for kinematic analysis for sports professionals' training purposes [LTW+10]; in news broadcast programs to identify the different stories and/or build personalized TV news programs [MLLR99]; in entertainment TV shows to recover their structure and identify the main parts allowing the browsing inside such programs; in films to provide the chapters of the film; in home videos to permit the organization of the videos related to certain events. All the methods presented in the previous sections try

to solve one ore more of these requirements. Both, specific and generic approaches have strengths and weaknesses. For the case of specific methods, although they give promising results, they can be applied to only very specific types of programs. They are characterized by a lack of generality due to the use of rules, templates or learning algorithms based on a previous analysis of the specific videos. Also when dealing with videos more complex than news or sports videos the computation cost could increase and an important knowledge in the domain will be needed.

On the other hand, generic methods try to structure a video without using any prior knowledge in the domain. Regarding the scene segmentation, the definition of a scene is very ambiguous and depends on the human understanding of its meaning. It is difficult to find an objective one, that would cover all users interests. This is why it is hard to compare the performance of the existing approaches and develop a better one. An objective evaluation would assume the existence of a ground truth (GT), whereas this GT is human generated so it depends on human judgment.

The use of recurrence however seems to be a promising approach with a lot of possible applications. We decided thus to use it as a starting point for the approach that we propose in this thesis.

# Chapter 2

# Visual analysis of recurrent TV programs

**Summary**



THE OBJECTIVE of TV program structuring, as already stated in the introduction part of this thesis, is to automatically recover the main parts of a program. The solution we propose is based on the detection of *separators*[1], which are video sequences that repeat within or between the episodes of a TV program and that separate the main parts of a program. Once the separators are detected, the main parts of the program can be identified using the separators boundaries. More precisely, the end of a separator is the start of a new part. The resulting structure is hence composed of a set of time codes that refer to the start and end of each part. In this thesis we analyze the characteristics these separators have and determine whether these make possible the proposed task.

[1]Separators have been introduced and exemplified in the *Introduction* Chapter, Section *Overview of the proposed approach*.

In this chapter we deal with the visual information of the separators. As previously stated the separators are recurrent so their visual information is similar among the different occurrences of a recurrent sequence. Consequently, to detect the visual separators we make use of two of their main properties meaning:

- **Repeatability**: Different episodes of the same recurrent TV program share the same structure and their separators are almost identical. This repeatability of separators can be found, as said, between episodes of the same recurrent TV program (inter-episode repeatability) but also inside a single episode (intra-episode repeatability).
- **Temporal stability**: As the different parts of a program have generally a similar duration, a separator can be found at approximately the same time-offset, for different episodes of the same recurrent TV program.

The first of these properties allows us to detect the separators as recurrences between the episodes of the same TV program or inside the same episode of the TV program. However, not all the detected recurrences are separators. It might happen that within the analyzed episodes there are sequences that are replayed or that are very similar. These of course are not separators. For instance, shots showing the moderator in the same position but at different times of the episode might be detected as occurrences of a repeated sequence. They are not separators but their content is very similar. Other examples are short sequences of the show presented in the summary as introduction of what is going to be presented during the current episode. These sequences are recurrent but are not separators. A filtering step is thus required to filter out the false alarms and distinguish the true separators. To do so, the second property of separators will be employed.

The general working scheme of the proposed method for the detection of visual separators proceeds in several steps as follows:

- It detects video sequences that are repeated (recurrences) in the input episodes (the episodes that are going to be structured). The occurrences of the repeated sequences are possible separators.
- It removes the occurrences of the repeated sequences that are not real separators (filtering and detection steps).
- It deduces the structure of input episodes.

These steps are described in the following subsections.

## 2.1 Visual recurrence detection algorithm

This part of the system performs the description of the visual content of the episodes that are going to be structured. It also proceeds with detecting the repeated sequences using the approach described in [BML08]. The first step is shot segmentation that is performed based on the color similarity of each two consecutive frames of a sliding window. For each shot, few keyframes are chosen following the method described in [BML08]. The chosen keyframe is representative for each homogeneous part of the shot. Its detection is based on the Page-Hinkley test [Hin71] that determine important changes in the signal. The keyframe is chosen in the middle of each two important changes of the signal.

A two level description of the frames is used. First, a DCT-based 64-bits Basic Visual Descriptor (BVD) is calculated for each frame. Its role is only to delimit the borders of the recurrences and needs to be invariant only to small variations due to compression for instance. The BVDs are easily compared using a Hamming distance.

The second level focuses on the keyframes and associates to each keyframe a more sophisticated and robust descriptor (KVD). It is a 30-dimensional descriptor and it is also DCT-based. Its purpose is to match near identical images and the metric used to compare the KVDs is the euclidian distance.

KVDs are clustered using a micro-clustering technique in order to group together the similar shots. It is an iterative technique that builds spherical clusters. A KVD is introduced into the cluster as long as the clusters' radius remains below a certain threshold. For more details about the clustering method, readers can refer to [BAG03].

The number of KVDs per cluster corresponds to the number of times a sequence is repeated. For example, a sequence repeated 2 times, each of the occurrences containing 3 keyframes, will correspond to a number of 3 clusters having each 2 KVDs (see Figure 2.1). A KVD is associated with a frame of a recurrence but does not provide information on the sequence boundary. This is where BVDs are used, in order to precisely determine the boundaries by matching corresponding frames in all occurrences of the repeated sequence.

The obtained clusters are analyzed, and based on the temporal diversity of KVDs within a cluster and on the inter-cluster relationships, the set of recurrences is created. The temporal diversity condition is used to remove the clusters that have no chance of generating recurrences. We refer to clusters containing only KVDs coming from the same shot or neighboring shots. The inter-cluster relationships are then used to rebuild the recurrences. First of all the temporal order of the keyframes is verified. For two clusters to generate a recurrence, their keyframes have to be alternating (blue flashes in Figure 2.1). Second, the temporal distance between each couple of keyframes of the analyzed clusters must be nearly constant. In Figure 2.1 this would mean that $|t_{11} - t_{12}|$ should be similar to $|t_{21} - t_{22}|$, $|t_{12} - t_{13}|$ should be similar to $|t_{22} - t_{23}|$ and finally $|t_{11} - t_{13}|$ should be similar to $|t_{21} - t_{23}|$.

This condition assures the inter-cluster similarity. The chance for two clusters to generate a repeated sequence is thus related to the constancy of the temporal distance between their keyframes. A similarity score is thus defined based on the standard deviation of the temporal distances between the couples of keyframes of the analyzed clusters. If this score is below a certain threshold and the temporal order of the keyframes is also verified (first condition), the clusters are considered similar and will be used to generate the recurrences.

The obtained recurrences are possible separators. For more details concerning this

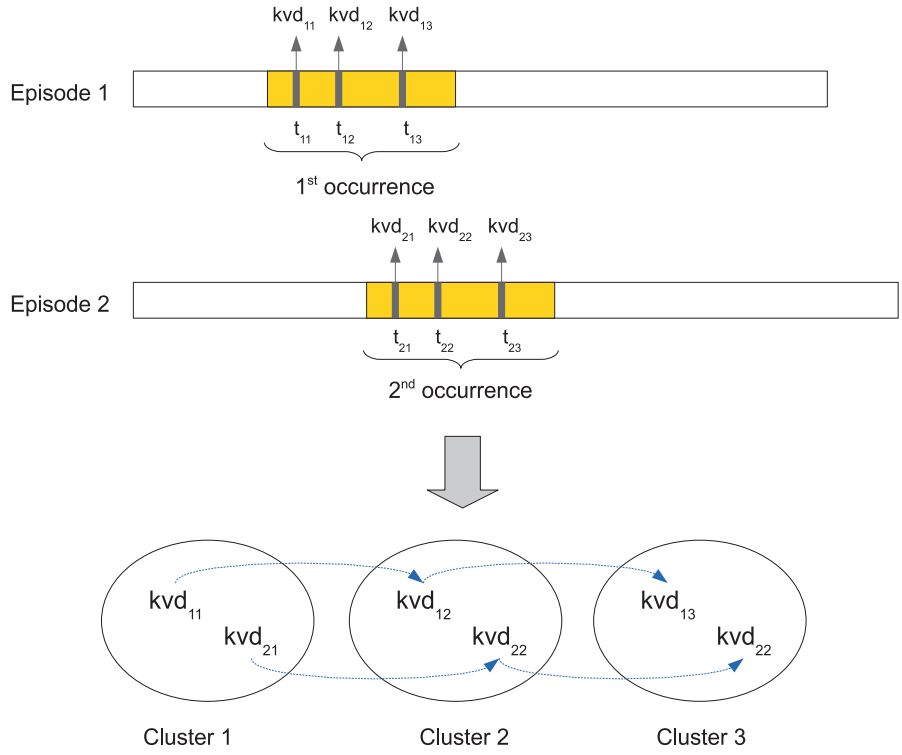



FIGURE 2.1: Example of keyframes clustering and recurrence detection.

approach for the detection of recurrences, the reader can refer to [BML08].

## 2.2 Separators detection

All the recurrences, detected during the previous step, are not necessarily separators. It might happen that within the analyzed episodes there are sequences that are replayed or that are very similar. These of course are not separators. For instance, shots showing the moderator in the same position but at different times of the episode might be detected as occurrences of a repeated sequence. They are not separators but their content is very similar. We call these occurrences "*false alarms*". In order to remove them, a post-processing filtering step is used.

### 2.2.1 Recurrence Prior filtering

The recurrences detected using the procedure described in Section 2.1 are first passed through a prior filter that removes a repeated sequence that has all its occurrences only coming from the same episode. Even if a separator can be repeated within the same episode, it has to be also repeated over at least two episodes in order to be valid. It is very unlikely to have a separator created specifically for a single episode.

### 2.2.2 Recurrence Temporal filtering

As previously explained, in the case of inter-episode recurrences, false alarms may also appear. However the separators have a property (i.e. temporal stability) that allows their identification among the recurrences. Generally episodes of a same TV program have a similar temporal length (duration) and structure. Meaning that they are generally allocated a similar temporal interval and the different parts composing the show remain the same from one episode to another. This means that the separators that delimit the different parts can be found at approximately the same time-offset, for different episodes of the same recurrent TV program. We use thus the temporal stability of separators in order to filter out the recurrences that are false alarms. To achieve that, a study of the temporal density of the occurrences of detected repeated sequences from the input episodes is performed. All the occurrences from different episodes are projected on the same temporal axis. From this projection, a histogram is computed by counting the number of occurrences during each 40ms window (each frame). A kernel-based density estimation is then performed [Sil86]. In this study, a Gaussian kernel has been used:

$$f_i = \sum_{j=i-3\sigma}^{i+3\sigma} h_j e^{-\frac{(j-i)^2}{2\sigma^2}}, \tag{2.1}$$

where $f_i$ represents the filtering result for frame $i$ and $h(j)$ represents the number of occurrences computed from the histogram, corresponding to frame $j$.

The idea is to find the areas with high concentrations. These areas are likely to be times when a separator is broadcasted. Isolated occurrences are likely to be false alarms as they appear quasi-randomly without any temporal stability.

The result of the temporal density analysis is a distribution curve where a maximum represents an area of high concentration of the separators. A threshold is then empirically determined as a fraction of the mean of all the maximum values. Section 2.4.4.3 discusses the choice of values for the computation of the threshold. The separators that have a density under the threshold are rejected. An illustrative example is given in Figure 2.2. The occurrences in blue and red are isolated occurrences that correspond to false alarms. Consequently these will be filtered out.

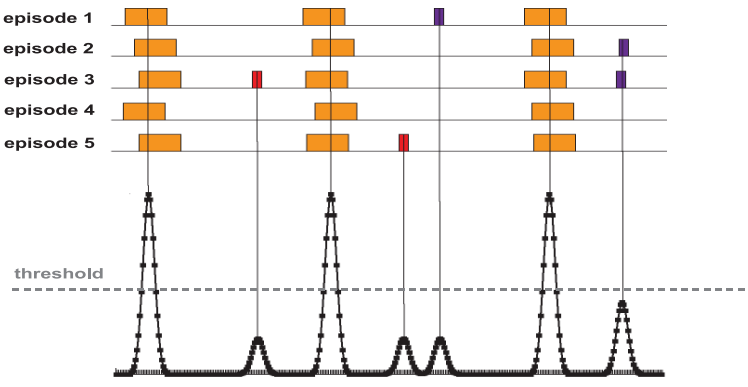



FIGURE 2.2: Temporal density of detected recurrences.

## 2.3 Program structuring

Once the separators have been detected they can be used for structuring. There are 2 possible situations:

### 2.3.1 Standard episode structuring

A first approach is when the detected separators are used to structure the input episodes. This means that the separators resulted after the application of the algorithm previously described, are directly used to structure the input/analyzed episodes. The different parts of each episode are identified using the separators boundaries, e.g. the end of a separator is the start of a new part.

In this context thus, in order to detect the separators and structure each new episode, the recurrence detection algorithm has to be applied on the current episode and a history of several previously broadcasted episodes. The number of episodes in the history is studied in Chapter 3, Section 3.6.

### 2.3.2 Episode structuring using a separators database

In a second approach, the detected separators are stored in a database and used later for structuring any future episode. In this context, the structuring of each new episode is performed using separators that have been previously detected and stored in a database. The idea is to use a content-based video matching technique in order to detect separators in the episode to structure using the database of separators.

The first step is to build the database. To do so, separators are detected on a set of episodes, as previously described. After applying the recurrence detection algorithm, for each detected repeated sequence, the longest occurrence is selected. The sequence id, the BVDs for each frame in this sequence and the offset of each frame to the beginning of the sequence (frame id), are stored in the database. Statistics about each of the detected recurrent sequences (i.e. minimal/maximal duration, duration of the chosen occurrence, mean offset (frame id), etc) can be stored for later usage and information.

Once the database is computed, the separators of each new episode to structure can be detected using a video matching technique. The algorithm we used is presented in Figure 2.3 and described hereafter.

The database contains the BVDs for each frame of the previously detected separators. For the new episode to structure the BVDs are also computed. A sliding window is used to search the perfect matching between the frames of the new episode to structure and those from the database. Frame matching is made using the BVDs. The number of frames in the sliding window is experimented in Section 2.4.6. Each of the BVDs in the sliding window is searched for in the database. If for 60% of the frames in the sliding window a match is found, the sequence id is identified as the dominant id and their temporal coherence is verified using the Viterbi algorithm. For each tested frame, there might be found several matching frames in the database. The idea is to build a best path by computing for each tested frame a cost that represents the number of previous matching frames in the sequence. At each iteration, a frame is added to the path only if it minimizes the cost. The frame ids of the matching sequence (from the database) have to be in ascending order. We do however allow an offset of 2 frames still in temporal ascending order. We work on

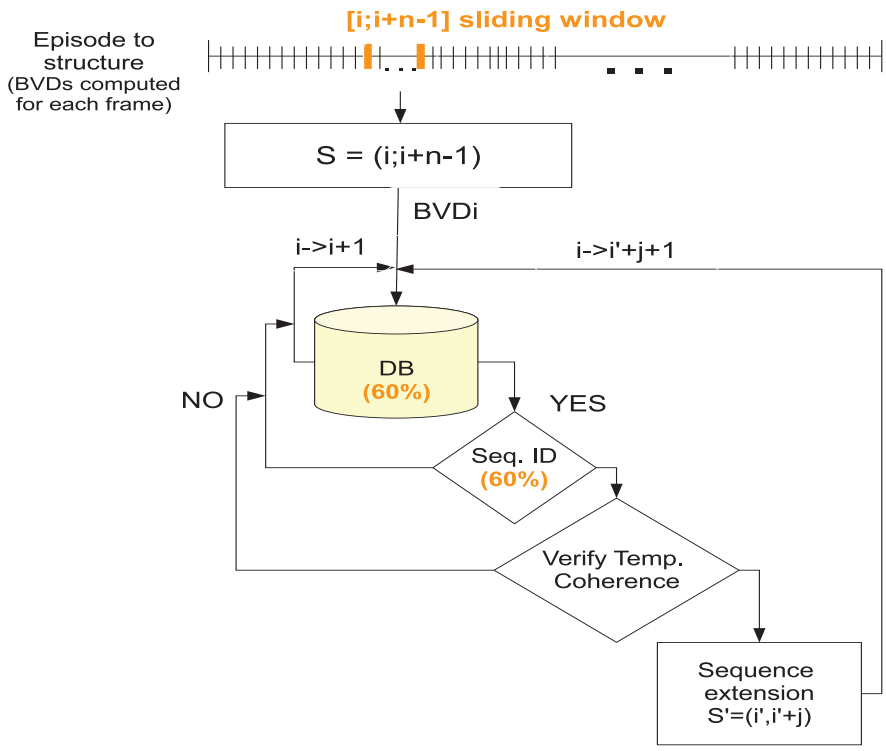



FIGURE 2.3: Episode structuring using a database of separators.

groups of 3 frames from the sequence identified in the sliding window. The Algorithm is detailed hereafter (Algorithm 1)[2].

Besides the Viterbi algorithm we have also tested another approach that considers at each iteration all the possible sequences of matching frames and finally chooses the longest one. The difference is that at each iteration we do not consider only the path with the highest cost but we compute all the possible paths. Only after treating all the matching frames corresponding to the frames in the sliding window, the best path is chosen as being the longest one (or the one with the higher cost) among all the possible detected paths.

Once the coherence is verified, the matching is validated and the algorithm proceeds with the extension of the detected sequence.

The extension procedure is illustrated in Figure 2.4 and consists of computing the similarity between the frames immediately at the left and right side of the detected sequence, with the corresponding ones in the database. We worked on a window of 5 frames and used the Hamming distance to compute their similarity. If their similarity is under a predefined threshold, the extension is made and the procedure is resumed for the next neighboring frames. This is done until there are no corresponding frames in the sequence from the DB or until the similarity measure no longer verifies the condition.

[2] <> = array

**input** : The sequence of identified frames (<seq_f> = <$f_1$,$f_2$,...,$f_n$>). Their corresponding matching frames from the database (< $seq\_f\_db$> = << $f\_db_1$ >,< $f\_db_2$ >,...,< $f\_db_n$ >>); where < $f\_db_i$ > = <$f\_db_{i1}$,$f\_db_{i2}$,...,$f\_db_{im}$>) with i = (1,n) and m = the number of possible matching frames for frame $f_i$/the length of < $f\_db_i$ >
$n$, the number of frames in the sliding window;
For each matching frame in the database ($f\_db_{ij}$) with i = (1,n) and j = (1,m), $id(f\_db_{ij})$ is the frame id registered in the database, $cost(f\_db_{ij})$ is the cost obtained when adding this frame to the current path and < $prev(f\_db_{ij})$ > is an array containing the frames ids of all the previous selected frames in the current path.
**output**: The temporal coherence and best possible path for the frames composing the matched sequence

```
begin
    // Build the tree node by node
     for i ← 0 to n − 3 do
        paths ← BuildPaths(< f_db_i >, < f_db_{i+1} >);
        paths ← BuildPaths(< f_db_i >, < f_db_{i+2} >);
    end
    paths ← BuildPaths(< f_db_{n−2} >, < f_db_{n−1} >);
    BestPath ← ChooseBestPath(paths)
end
```

**Algorithm 1:** Viterbi algorithm used to verify the order and choose the best sequence of frames in the matched visual sequence

```
k ← length(< f_db_A >);
l ← length(< f_db_B >);
for i ← 0 to k − 1 do
    for j ← 0 to l − 1 do
        if id(f_db_B^j) − id(f_db_A^i) <= 2 then
            new_cost = cost(f_db_A^i);
            if new_cost > cost(f_db_B^j) then
                cost(f_db_B^j) = new_cost; push_back(prev(f_db_B^j), prev(f_db_A^i));
                push_back(prev(f_db_B^j), id(f_db_A^i));
            end
        end
    end
end
```

**Procedure** Description of procedure "BuildPaths(< $f\_db_A$ >,< $f\_db_B$ >)"

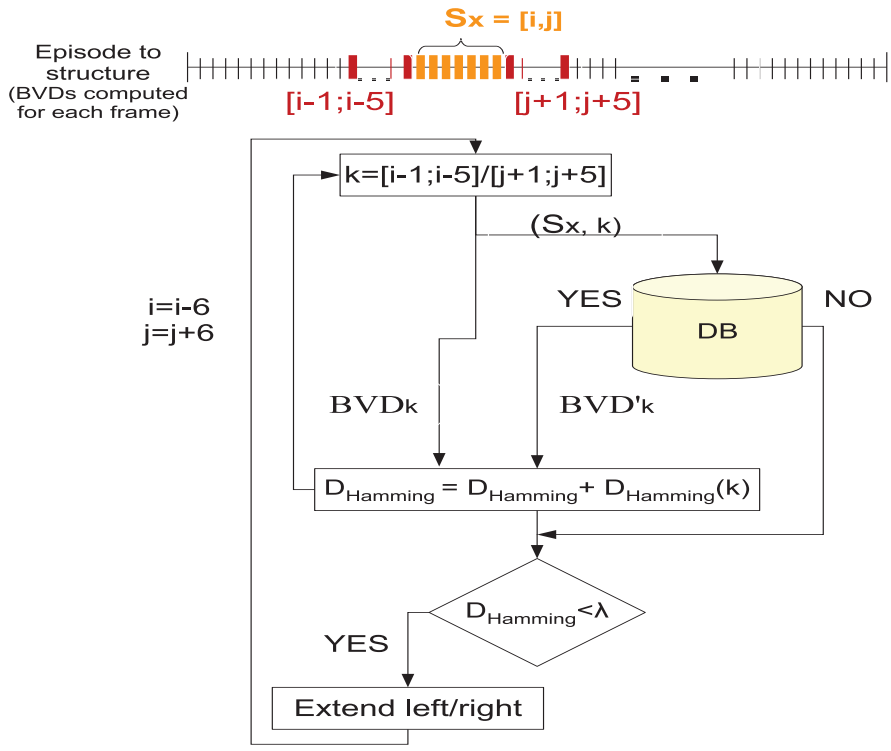



FIGURE 2.4: Extension of the matched sequence.

## 2.4 Experiments

To evaluate the method we proposed for the detection of separators in TV programs, we performed experiments on real TV broadcasts. In a first experiment we study the ability of detecting separators as repeated sequences in recurrent TV programs. The objective is to validate the main idea behind our approach, meaning that separators can be detected as recurrences among the episodes of a TV program. The second experiment evaluates the effectiveness of filtering "false alarms" from the set of detected recurrences. In a third experiment we propose and study a new filter based on the length of separators. Finally the last experiment treats the possibility of using a database of separators for structuring each new arriving episode of a TV program.

We present hereafter, the experimental context of our evaluation, the evaluation protocol used to analyze the obtained results and finally the experiments and obtained results.

### 2.4.1 Experimental context

The dataset used for experiments, is composed of real TV broadcasts. It contains about 112 hours of videos, corresponding to 169 episodes of 11 French TV programs, among which TV games, magazines and a news program. These are described in detail in Annex A and also presented in the table below (Table 2.1).

For all of the episodes composing the programs in the experimental dataset, we have manually annotated the separators. Each separator has been precisely determined, indi-

| Program Type | Program Name | Program Id | Nb. of episodes | Nb. of separators/episode |
|---|---|---|---|---|
| Games | Les Z'amours | $G_1$ | 28 | 6 |
| | Mot de passe | $G_2$ | 5 | 4 |
| | Motus | $G_3$ | 21 | 4 |
| | Tlmvpsp | $G_4$ | 26 | 5 |
| | Les 12 coups de midi | $G_5$ | 20 | 8 |
| Magazines News | Comment ca va bien | $M_1$ | 24 | 8/14 |
| | 10h Le Mag | $M_2$ | 5 | 20/23 |
| | Cine, Series et cie | $M_3$ | 7 | 15/17 |
| | 50mn Inside | $M_4$ | 14 | 12/18 |
| | 7 A 8 | $M_5$ | 10 | 5/7 |
| | 19h45 | $M_6$ | 9 | 20/27 |

TABLE 2.1: Dataset description.

cating its start time and end time. This represents the ground-truth and will be used for evaluating the proposed methods. We insist on the fact that we did not annotate all the existing recurrences in the episodes which would be a very difficult, even impossible, task. We use the ground-truth to evaluate the method proposed for the detection of separators. These separators can be found among the detected recurrences. We do not evaluate the methods' capacity to detect the recurrences, but its capacity to detect the separators, which are the scope of our recurrence detection method.

Furthermore, we conducted the experiments on sets of four episodes (the current episode plus three episodes in the history) in order to detect the separators for the fourth episode. Only these last separators will be considered for evaluation. The study in Chapter 3, Section 3.6 gives more details about the number of episodes considered in the history.

The results will be presented for each program separately. A global measure, showing the results of the evaluation for the entire dataset, is also computed.

All the algorithms have been implemented in C++ and the integrated development environment that has been used is Microsoft Visual Studio 2008.

### 2.4.2 Evaluation protocol

To evaluate the proposed methods, we used the precision and recall measures. The precision (P) is the number of relevant elements detected by a retrieval system (true positive) divided by the total number of retrieved elements (true positive + false positive). In our case, it corresponds to the number of detected separators divided by the number of all the detected recurrences.

The recall (R) represents the fraction of relevant elements retrieved by the system (true positive) from the total number of existing relevant elements (true positive + false negative). In our case, this corresponds to the number of detected separators divided by the total number of separators existing in the ground truth.

The relevance of a detected element is given by the ground truth.

$$P = \frac{relevant \cap retrieved}{retrieved} \qquad R = \frac{relevant \cap retrieved}{relevant}$$

For the moment, there is no study that could relate the type of errors to the end-user satisfaction. Consequently, we cannot decide whether when browsing inside a TV show, it is more annoying to have some missing separators (related to false negative errors) or to have to skip some recurrences that are not separators (related to false positives errors). Likewise, we do not know the number of recurrences that are not separators that could be tolerated by an user. Therefore, we use for evaluation the precision (P), recall (R) and also the harmonic mean (F).

The harmonic mean, also named F-score, combines the precision and recall into one measure. It measures the test accuracy, by computing a mean of precision and recall which are evenly weighted. It gives information about the proportion of true results in the entire test dataset.

$$F = \frac{2 * P * R}{P + R}$$

Nevertheless, if we focus on the browsing use-case, separators can be used as anchors that allow users to skip a part and to go directly to the next one, or for instance, to go back to the beginning of the current part. In this case, recovering as many as possible of the separators from a program becomes very important. Consequently, when computing the precision and recall, a separator is considered as being correctly identified if it overlaps even with a minimal number of frames with its correspondent in the ground truth. We consider that it is more important to keep and use all the detected markers that indicate the separation of 2 parts in a program than to reject them due to their small length. Examples of such cases appear generally for the audio recurrences and will be described in more details in Chapter 3.

In order to avoid extreme situations like the ones described in Figure 2.5 we have however imposed some constraints when computing the evaluation measures. In Figure 2.5(a) we would have a recall and precision of 1 even though it would be impossible to distinguish the different parts of the program. In Figure 2.5(b) the recall would also be higher than it should be. To solve such cases, the detected recurrences should superpose with only one separator from the ground truth.

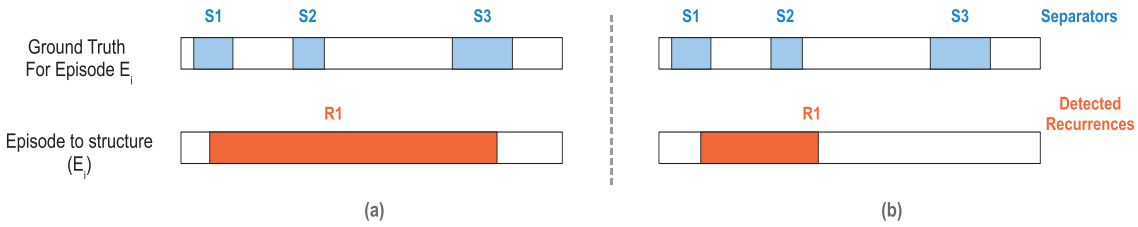



FIGURE 2.5: Precision and Recall Computation - Special Case 1.

Still, this would disadvantage the cases when 2 separators are very close as in Figure 2.6. We run into similar cases during experiments. An example is for programs where an advertising was placed between these 2 separators but it has been removed for processing. To clear out this case too, we imposed a threshold for the distance between the separators in the ground-truth. Consequently a recurrence that has multiple superpositions with separators from the ground truth is considered a separator, as long as the distance between these last ones is very small (less than 1 second).

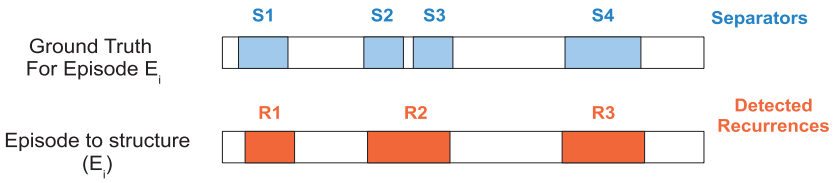



FIGURE 2.6: Precision and Recall Computation - Special Case 2.

When evaluating separately the performance for each dataset, precision and recall are computed for all the recurrences detected for all the episodes of that specific program. On the other hand, when computing the global measures of precision and recall, all the recurrences from all the episodes of all the TV programs in the dataset are considered.

### 2.4.3 Exp.1: Analysis of visual recurrence detection results

The aim of this first experiment is to evaluate the effectiveness of the recurrence detection algorithm for the detection of separators. The starting point of our study is the detection of recurrences. The main assumption of our approach is that separators are repeated and can be detected as recurrences. In order to validate our assumption we evaluated in this first experiment the proportion of separators detected as recurrences. To do so we applied the visual recurrence detection method presented in Section 2.1. We recall that experiments were computed on set of four episodes (the current episode plus three episodes in the history) in order to detect the separators for the fourth episode. The number of episodes in the history is discussed in Chapter 3, Section 3.6.

Table 2.2 shows the obtained results in terms of precision, recall and F-score.

| Dataset | Precision | Recall | F |
|---|---|---|---|
| G1 | 0.778 | 0.968 | 0.863 |
| G2 | 0.173 | 0.950 | 0.293 |
| G3 | 0.224 | 0.952 | 0.362 |
| G4 | 0.556 | 0.785 | 0.651 |
| G5 | 0.033 | 0.013 | 0.018 |
| M1 | 0.259 | 0.183 | 0.214 |
| M2 | 0.791 | 0.955 | 0.865 |
| M3 | 0.515 | 0.120 | 0.194 |
| M4 | 0.702 | 0.550 | 0.617 |
| M5 | 0.638 | 0.910 | 0.750 |
| M6 | 0.479 | 0.702 | 0.569 |
| Global | 0.502 | 0.563 | 0.531 |

TABLE 2.2: Exp1: Visual recurrences analysis.

The third column of the Table 2.2 presents the recall obtained for the visual separators. In the case of TV game shows, the results show a good detection of separators with high recall values (G1, G2, G3). This is explained by the fact that in the case of games, the

separators are identical sequences of synthetic images that can be easily recognized by our algorithm. Some of the games contain sequences that present an object that will be won. This object may or may not be different from one show to another. Hence, the recurrence based algorithm will or will not be able to detect the corresponding sequence. Including them or not in the ground-truth may change the results. For instance, for the case of dataset G3, when not including these sequences in the ground-truth, the recall increases to 1 while the precision decreases to 0.184. A similar problem appears also for datasets M4 where the recall can increase to 0.846 and the precision decreases to 0.496. The separators in dataset G5 are not appropriated for the method we used for the detection of visual recurrences. The separators of this game present the logo (of the game or of the stage of the game) having on the background images with the set, the anchor or the audience. The logo is highly animated and the background is also mobile as the camera moves through the set from the audience to the competitors and anchor. Even if the logo remains the same the background will generally be different. This separators can not be handled by our algorithm since our focus is on near identical repeated sequences. Some examples of images extracted from such separators are illustrated in Figure 2.7.


FIGURE 2.7: Example of separators for G5 dataset.

Similar cases of separators are often found in the case of magazines and news. This is proven by the smaller values of the recall. This specific type of separators, composed of a natural image with the anchor person (in different positions and with different backgrounds) or with the set, on which a logo is superposed, is not fitted for our approach. Examples of such separators from datasets M2, M3 and M4 are shown in Figure 2.8


FIGURE 2.8: Example of undetected separators for M2, M3 and M4 datasets.

For dataset M2, some of the recurrences that are not separators and that are found within the main parts of the program could be used for structuring the parts of program they belong to. However this is out of our purpose. Our scope is to structure episodes in their main parts and we are not interested in a higher level of structuring. These recurrences are thus considered false alarms and decrease the precision with about 4%.

For the case of the datasets M1 and M3, a lot of the separators are animated by different editing effects. For M3 a fold out of a frame marks the passing to the next subject of the news program (see Figure 2.9). For M1 the images composing the separators are rotated

FIGURE 2.9: Separator with fold out effect from M3.

in different directions or zoomed in/out (see Figure 2.10).

Our approach is focused on the detection of nearly identical sequences of images. Most of the specific separators previously described are not appropriated and thus, can not be handled.

FIGURE 2.10: Separators with rotation and zoom in from M1.

Regarding the precision, for most of the datasets, the results are generally lower comparing to those obtained for the recall. The false alarms generally correspond to recurrences of frontal views of the anchors, images of the set or even parts of the show that are visually very similar or that are replayed from one episode to another. These may come from inter or intra episode recurrences. Examples of such false alarms are shown in Annex F. For the case of magazines, often false alarms are generated by the fact that at the beginning of each episode, there is a brief overview of the reports that will be presented in that episode. The short sequences presented in the overview will also be found during the program, when the subject is described in detail as a report. These correspond generally to intra episode recurrences. Sometimes, this overview is also segmented with a lot of short recurrences (i.e. M5) that were not included in the ground-truth as these segment the summary (one of the main parts of the program), and not the summary from the other parts of the program. Consequently these are considered false alarms and can be referred as inter and intra episode recurrences.

We also noticed that there are episodes that contain sequences from other episodes. For example after each important part of a program, a sequence with the audience that applause is showed. These sequences are sometimes reedited and introduced into other episodes too. Another example is for the programs that present news. If a certain information has a very important significance or has a continuity over several days/weeks, it, or parts of it are presented in several consecutive episodes. Other false alarms are also sequences of black or white frames that can repeat inter or intra episodes. Examples of images coming from false alarms are illustrated in Annex F.

### 2.4.4 Exp.2: From recurrences to separators

As previously presented a lot of false alarms may appear among the detected visual recurrences. These can be issued from intra episode recurrences or from inter episode recurrences. The next two subsections present the filters we proposed to remove these false alarms.

#### 2.4.4.1 Exp.2.1: Analysis of visual recurrence results after prior filtering

A first filter, that we named "*prior filter*" targets the intra episode false alarms. The idea is to filter out all the recurrent sequences containing only occurrences from the same episode. It is reasonable to consider that separators repeat over several episodes of the same TV program and it is hardly possible to have a separator created specifically for a single episode.

The results in terms of precision(P), recall(R) and F-score (F), obtained after using this prior filter are presented in Table 2.3(b).

| Dataset | $P_{i-1}$ | $R_{i-1}$ | $F_{i-1}$ |
|---|---|---|---|
| G1 | 0.778 | 0.968 | 0.863 |
| G2 | 0.173 | 0.950 | 0.293 |
| G3 | 0.224 | 0.952 | 0.362 |
| G4 | 0.556 | 0.785 | 0.651 |
| G5 | 0.033 | 0.013 | 0.018 |
| M1 | 0.259 | 0.183 | 0.214 |
| M2 | 0.791 | 0.955 | 0.865 |
| M3 | 0.515 | 0.120 | 0.194 |
| M4 | 0.702 | 0.550 | 0.617 |
| M5 | 0.638 | 0.910 | 0.750 |
| M6 | 0.479 | 0.702 | 0.569 |
| Global | 0.502 | 0.563 | 0.531 |

(a) (before prior filter - step (i-1))

| Dataset | $P_i$ | $R_i$ | $F_i$ |
|---|---|---|---|
| G1 | 0.826 | 0.968 | 0.891 |
| G2 | 0.840 | 0.950 | 0.892 |
| G3 | 0.953 | 0.952 | 0.953 |
| G4 | 1.000 | 0.785 | 0.879 |
| G5 | 0.069 | 0.013 | 0.021 |
| M1 | 0.735 | 0.174 | 0.281 |
| M2 | 0.905 | 0.955 | 0.929 |
| M3 | 0.571 | 0.091 | 0.157 |
| M4 | 0.939 | 0.379 | 0.540 |
| M5 | 0.721 | 0.838 | 0.775 |
| M6 | 0.789 | 0.688 | 0.735 |
| Global | 0.830 | 0.539 | 0.653 |

(b) (after prior filter - step i)

TABLE 2.3: Exp2.1: Performance of prior filter on visual recurrences.

From the data in Table 2.3 it is evident that globally, there is a major increase of the precision, with 33%, with a small decrease, of 2%, of the recall, which increases the F measure with 12%.

Noticeable is that the slight decrease of the recall comes from the magazines. This is explained by the fact that for magazines generally most of the separators are inter and intra episode recurrent sequences. More precisely, the program has several separators that are repeated throughout an entire episode and within all the episodes belonging to that program. It might happen for separators within an episode to contain a small specific and overlaid logo for commercials that does not appear in the separators of other episodes. In this case, the clustering algorithm that detects identical visual sequences (see Section 2.1), will cluster together all the separators having the logo (and implicitly those belonging to only that specific episode), and separately the separators that do not contain the logo. A repeated sequence, corresponding to separators and containing occurrences only from the

same episode will thus be created. The prior filter removes it resulting thus in a decrease of the recall. Two examples of images extracted from such separators are presented in Figure 2.11. The red flash points to the logo inserted in one of the episodes of the program.

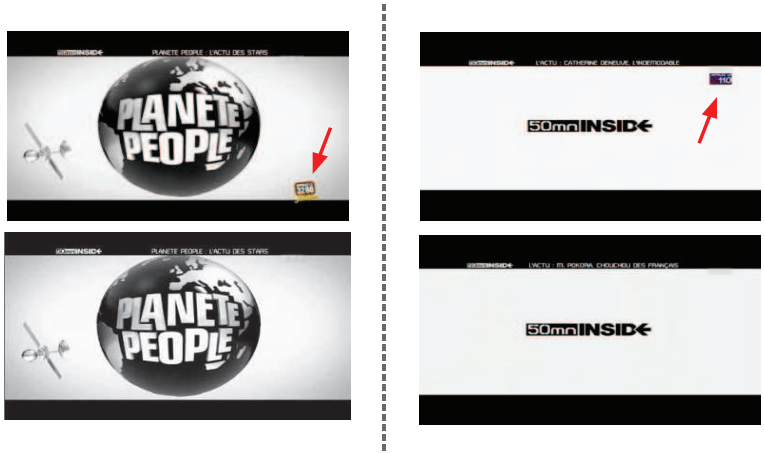



FIGURE 2.11: Differences among separators from different episodes of a same TV program.

Another case corresponds to separators that are generally hard to detect by the algorithm we used. Examples of such separators were described in Subsection 2.4.3 and Figure 2.8. These separators contain sometimes some frames that are identical among them. Starting with these frames the sequences will be extended as the descriptors used for the extension are not so precise and thus will consider similar even images similar to those presented in Figure 2.8. Important is however to find these few red frames that sometimes may miss from separators as these are not played till their end or, as it often happens, their descriptors are not similar enough to generate recurrences among different episodes (due to compression artifacts as block effects).

Other examples that explain the filtering of real separators by the prior filter is when a recurrence contains a part of a separator together with a sequence presenting the anchor. These sequences were obtained from recurrences of the anchor extended so as to cover the separators preceding it. Generally such recurrences with the anchor are detected inside the same episode as the background remains the same during one episode and the anchor has the same position and the same clothes. The prior filter, filters out these sequences and consequently the separators that were contained within the sequences too.

Even though we have described in detail the cases we encountered, all these are however not very frequent. This is proven by the fact that these represent only 2% of the detected separators. If we consider separately the magazines, the recall decreases of 6%, with an increase of the precision and F score of 21% and respectively 3%. Moreover, the filtering of separators is not due to the fact that the condition we imposed (that separators have to be repeated in at least 2 episodes) is erroneous, but to the noise induced by the recurrence detection algorithm or other types of noise from the videos (as the one induced by compression or the presence of logos). Therefore, we decided useful the prior filter and we will consider it in all the experiments that follow this section.

#### 2.4.4.2 Exp.2.2: Analysis of visual recurrence results after temporal filtering

In this experiment, we focus on the temporal filtering step and assess its ability to improve the precision of the detected separators. Based on the temporal stability of the separators, their distribution over a temporal axis is calculated. All the peaks that have their maximum value under a certain threshold will be considered as false alarms time regions. This means that if an occurrence of a repeated sequence is located in these time regions, it is filtered out. Moreover, a sequence that has one of its occurrences that has been identified as a false alarm is completely rejected. The value chosen for the threshold has been empirically deduced and is discussed in more detail in Section 2.4.4.3.

The obtained results are illustrated in Table 2.4(b).

| Dataset | $P_i$ | $R_i$ | $F_i$ |
|---|---|---|---|
| G1 | 0.826 | 0.968 | 0.891 |
| G2 | 0.840 | 0.950 | 0.892 |
| G3 | 0.953 | 0.952 | 0.953 |
| G4 | 1.000 | 0.785 | 0.879 |
| G5 | 0.069 | 0.013 | 0.021 |
| M1 | 0.735 | 0.174 | 0.281 |
| M2 | 0.905 | 0.955 | 0.929 |
| M3 | 0.571 | 0.091 | 0.157 |
| M4 | 0.939 | 0.379 | 0.540 |
| M5 | 0.721 | 0.838 | 0.775 |
| M6 | 0.789 | 0.688 | 0.735 |
| Global | 0.830 | 0.539 | 0.653 |

(a) (before temporal filter - step (i))

| Dataset | $P_{i+1}$ | $R_{i+1}$ | $F_{i+1}$ |
|---|---|---|---|
| G1 | 0.905 | 0.968 | 0.935 |
| G2 | 0.913 | 0.950 | 0.931 |
| G3 | 0.988 | 0.952 | 0.970 |
| G4 | 1.000 | 0.785 | 0.879 |
| G5 | 0.083 | 0.013 | 0.022 |
| M1 | 0.800 | 0.165 | 0.274 |
| M2 | 0.911 | 0.955 | 0.932 |
| M3 | 0.632 | 0.077 | 0.137 |
| M4 | 0.964 | 0.379 | 0.545 |
| M5 | 0.745 | 0.712 | 0.728 |
| M6 | 0.893 | 0.368 | 0.521 |
| Global | 0.878 | 0.511 | 0.646 |

(b) (after temporal filter - step i+1)

TABLE 2.4: Exp2.2: Performance of temporal filter.

Globally, Table 2.4 shows an increase of the precision but with a decrease for the recall. Some separators are thus filtered out by this filter. This means that some of the programs do not respect the temporal stability condition that we assumed. If we take a closer look in the table we notice that actually only the magazines and news decrease in recall. For most of these programs the temporal filter does not bring any benefit considering that the precision increases, the recall decreases and the F measure suffers also a small decrease. For the entire set of magazines and news, we confirm that the precision increases with 4% and the recall decreases with 4,3%. Consequently the temporal filter has no contribution. This can however be explained by the fact that the news and magazines are generally composed of reports. Even though the length of the episodes in such programs is generally the same, the number of reports and their length is different. It depends of the subject discussed in the report and the number of reports that have to be broadcasted in a predefined time period. For these programs, the assumption of temporal stability that we made is more likely to be erroneous for quite a large number of such programs. We decided thus to use the temporal filter only for the games where the temporal stability is respected and there is a contribution brought by the filter.

#### 2.4.4.3 Parameters determination for temporal filters

In Section 2.2.2 we have presented the use of a temporal filter needed to filter out the recurrences that are not separators. We assumed that the separators are temporally stable and computed their distribution on a temporal axis.

To do so, all the recurrences resulted after applying the recurrence detection algorithm and the prior filter, are projected on the same temporal axis. From this projection, a histogram is computed by counting the number of recurrences during each 40ms window (each frame). To determine the areas of high concentrations of separators, a kernel-based density estimation is then performed using a Gaussian kernel. The Gaussian is centered on each frame and multiplied by the corresponding value in the histogram. All the superposing parts of the resulted Gaussian graphs are then summed up. The result is a curve with the higher picks indicating the temporal areas when the separators are broadcasted.

Two parameters have to be considered.

The first is imposed by the use of the Gaussian kernel and it refers to the **standard deviation** ($\sigma$) or the **variance** ($\sigma^2$). This controls the width of the graph returned by the Gaussian function and consequently also the influence of the neighboring recurrences. To find the optimal value we computed such graphs for different values of $\sigma$ and for different TV programs. Some illustrative results are shown in Figures 2.12 and 2.13. The bars of different colors and heights in the figures represent the areas where separators from different episodes are broadcasted. The color and the height are specific to each episode. For example the green ones, having the same height, represent separators from the first episode in history while the blue ones from the second episode. The peaks that do not superpose to any bar represent areas with false alarms. An example is in Figure 2.12 where the false alarms are indicated with red flashes on the figure.

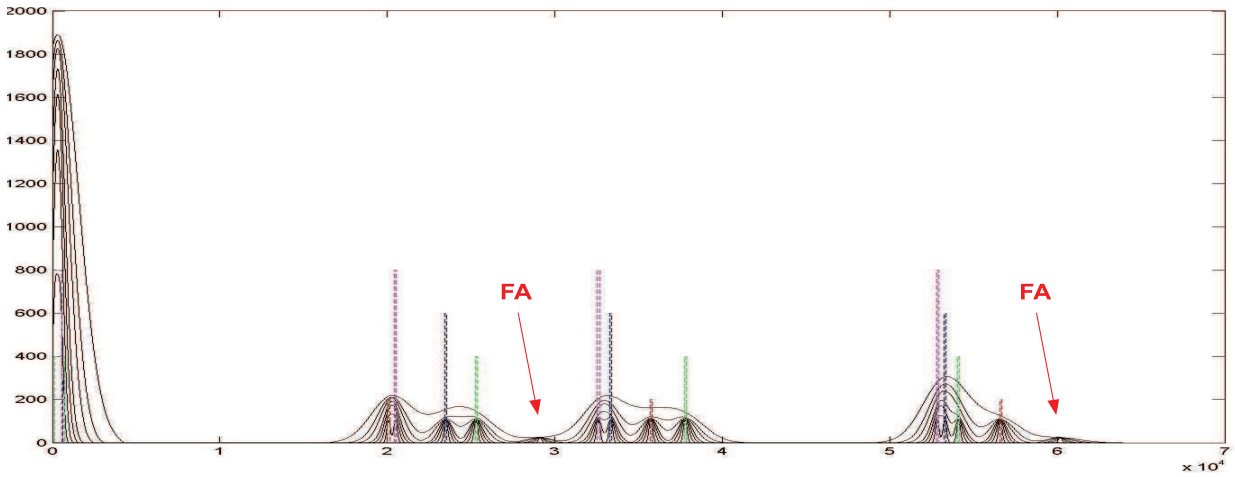



FIGURE 2.12: Example of density estimation from dataset G2.

However, false alarms may often appear close to separators or even superpose to separators in other episodes, so it is important to consider a function not too large in width. On the other hand, separators are not found at exactly the same time instances from one episode to another so a distance/gap has to be considered. After analyzing the ob-

tained results we have empirically decided that $\sigma = 207$ frames is an optimal value for our purpose.

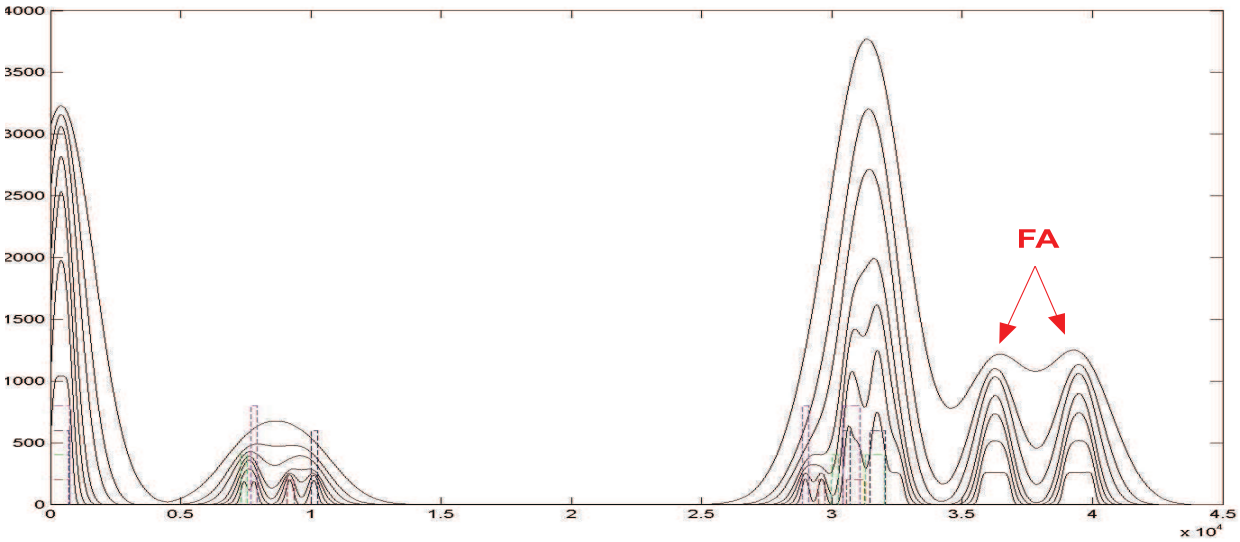



FIGURE 2.13: Example of density estimation from dataset G3.

When analyzing these graphs, we have noticed that very important is also the length of the recurrences in the sense that a long false alarm will return a high peak that would influence the threshold computed as previously described. An example is in Figure 2.13. To filter such false alarms a very high threshold would be needed, but this will also filter the separators. Consequently, we consider that a study on the length of recurrences and a possible length filter would be interesting to analyze. Section 2.4.5 approaches this problem in more details.

In Section 2.4.4.2 we have concluded that in the case of magazines and news, the temporal stability is not really satisfied. To further confirm this statement, we computed the density estimation for a magazine as well. The result is illustrated in Figure 2.14. Visibly, the separators from the analyzed episodes, are spread around the entire timeline and the areas of high concentrations are hard to identify. Moreover, often, false alarms are superposed to separators from other episodes so these cannot be identified as isolated recurrences.

The next step after computing the distribution curve is to eliminate the small peaks, meaning the regions with isolated recurrences. These are most probably false alarms as these are not temporally stable and repeat quasi randomly. To do so, a second parameter is needed, meaning a **minimal threshold**. All the peaks that are found under this threshold are rejected and consequently the corresponding recurrences too. In order to adapt this threshold to each particular case of program, we considered it as a fraction of the mean of all the maximum values of the peaks, computed for the analyzed episodes. To precisely detect which part of the mean should be considered we computed experiments by varying this parameter. The dataset used for this experimental part is composed of 3 TV games (G1, G3 and G4) and 3 magazines and news (M3, M4 and M5), all of them are parts of

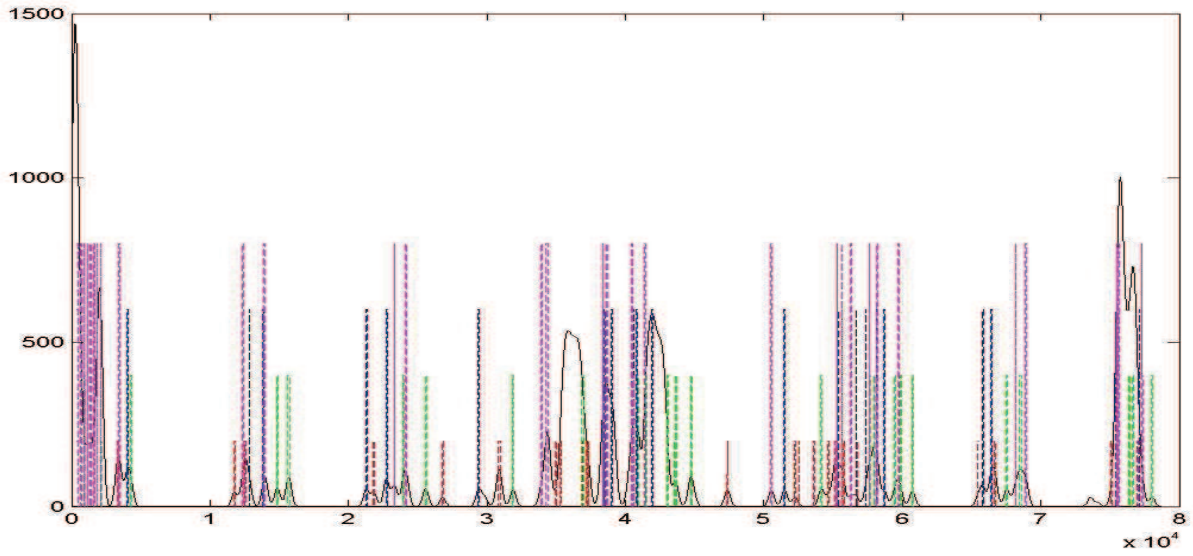



FIGURE 2.14: Example of density estimation from dataset M5.

the dataset presented in Annex A.

The results obtained on the entire used dataset are shown in Figure 2.15.

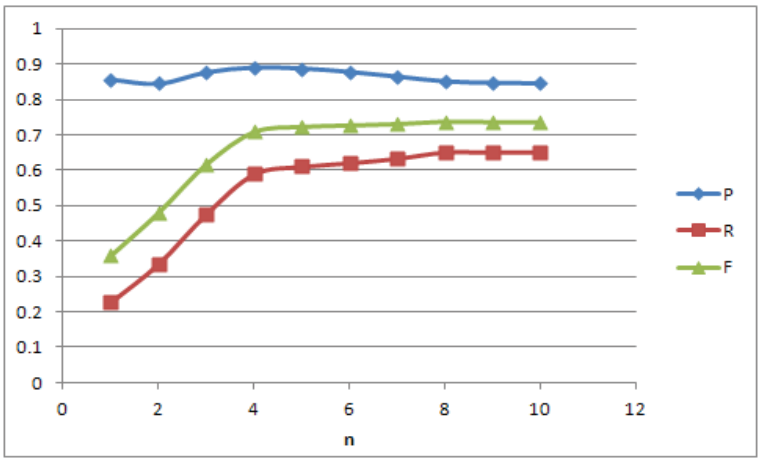



FIGURE 2.15: Precision, Recall and F measure for different thresholds of the temporal filter. Datasets: G1, G3, G4, M3, M4 and M5.

On the vertical axis, the obtained values for the precision, recall and F measure are illustrated. On the horizontal axis, the different values corresponding to the parameter $n$ are shown, where n is used to compute the threshold as : $threshold = mean/n$.

Obviously, when increasing $n$, the threshold decreases and the number of recurrences filtered by the temporal filter is smaller. A high threshold, which corresponds to small values of n, filters not only false alarms but separators too resulting in a decrease of the recall. These results were computed on the entire dataset meaning games, magazines and

news. However we showed in Section 2.4.4.2 that the temporal filter is not useful in the case of magazines and news. If we consider only the TV games, the obtained results are shown in Figure 2.16.

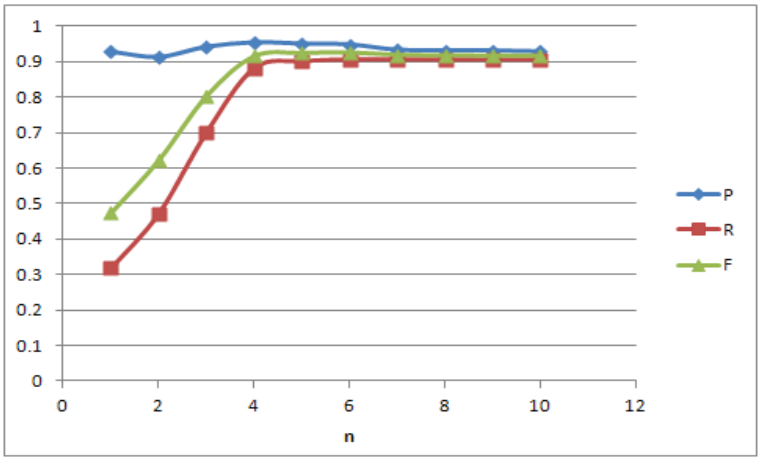



FIGURE 2.16: Precision, Recall and F measure for different thresholds of the temporal filter - Games.

For the games, we notice that starting from n = 5, the recall stabilizes to a maximum value. The precision increases also, up to a maximum value, for n = 5 or 6, and then starts to slightly decrease with the decrease of the threshold which allows more false alarms to pass. The best trade-off seems to be around n = 5. Table 2.5(a) correspond to the results represented in Figure 2.16. The last line in the table provides the results before the temporal filtering. We notice that for n = 5, the recall is equal to the one before the temporal filtering while the precision registers an increase of 5%.

In Table 2.5(b) we provided also the results for magazines and news to prove, once again, that the temporal filtering does not bring anything for this kind of programs.

| n | Precision | Recall | F |
|---|---|---|---|
| 1 | 0.928 | 0.318 | 0.473 |
| 2 | 0.914 | 0.471 | 0.622 |
| 3 | 0.941 | 0.700 | 0.803 |
| 4 | 0.953 | 0.881 | 0.916 |
| 5 | 0.947 | 0.906 | 0.926 |
| 6 | 0.947 | 0.906 | 0.926 |
| 7 | 0.933 | 0.906 | 0.919 |
| 8 | 0.931 | 0.906 | 0.918 |
| 9 | 0.931 | 0.906 | 0.917 |
| 10 | 0.929 | 0.906 | 0.917 |
| no filter | 0.900 | 0.906 | 0.903 |

(a) (Games)

| n | Precision | Recall | F |
|---|---|---|---|
| 1 | 0.731 | 0.134 | 0.227 |
| 2 | 0.722 | 0.196 | 0.308 |
| 3 | 0.736 | 0.245 | 0.367 |
| 4 | 0.739 | 0.286 | 0.413 |
| 5 | 0.744 | 0.309 | 0.437 |
| 6 | 0.728 | 0.325 | 0.449 |
| 7 | 0.727 | 0.351 | 0.473 |
| 8 | 0.710 | 0.387 | 0.501 |
| 9 | 0.700 | 0.387 | 0.498 |
| 10 | 0.700 | 0.387 | 0.498 |
| no filter | 0.706 | 0.410 | 0.519 |

(b) (Mag&News)

TABLE 2.5: Precision, Recall and F measure for different thresholds of the temporal filter.

Consequently, from now on, we will use the temporal filters only for TV games, and with a minimal threshold of $mean/5$.

### 2.4.5 Exp.3: Visual Recurrence Length filtering

In the previous subsection, we have seen that the detected visual recurrences might sometimes be longer than we might have expected. This influences the temporal filter as long recurrences give very high peaks and thus increase the threshold computed for filtering. Generally, these do not correspond to separators so a pertinent idea would be to use a length filter in order to filter them out. For this, we first need to know the length of separators so that a threshold for the length could be computed. This threshold should evidently be higher than the maximum length of the separators to avoid filtering these ones too.

The dataset is thus split into two. A first subset is used for training while the second one for testing. Table 2.6(a) shows some statistics made on the length of the separators in the training dataset. The data refers to separators excluding the opening and closing credits which generally are among the longest separators in a TV show. The maximal length of separators is of 720 frames. If we use it as threshold ($\Delta_{max}$) and filter all the recurrences that have a length longer than the threshold, the obtained results are presented in Table 2.7 column "$\Delta_{max} = 720$ f". The column "Initial" of the table corresponds to the initial evaluation of the recurrences. When comparing the two, we notice that globally while increasing the precision the recall slightly decreases. Even if the changes are very small, they indicate that there are separators longer than 720 frames that were filtered out. It is thus deductible that the separators and the false alarms have similar lengths. This could mean that there are differences in length between the separators in the ground truth and the detected visual recurrences. This is illustrated in Table 2.6(b) where statistics on the length of all the detected visual recurrences are shown.

| Training Datasets | **Separators** | | |
|---|---|---|---|
| | **Min** | **Max** | **Mean** |
| **G1** | 47 | 122 | 92.68 |
| **G2** | 109 | 118 | 114.33 |
| **G3** | 193 | 720 | 390.8 |
| **G4** | 59 | 243 | 139.69 |
| **G5** | 100 | 164 | 125.30 |
| **M1** | 10 | 122 | 51.97 |
| **M2** | 44 | 157 | 83.2 |
| **M3** | 12 | 204 | 29.12 |
| **M4** | 59 | 342 | 113.64 |
| **M5** | 29 | 126 | 66.97 |
| **M6** | 13 | 133 | 47.8 |

(a) (Stats on Separators)

| Training Datasets | **Recurrences** | | |
|---|---|---|---|
| | **Min** | **Max** | **Mean** |
| **G1** | 14 | 282 | 124.90 |
| **G2** | 24 | 114 | 95 |
| **G3** | 32 | 684 | 387.55 |
| **G4** | 22 | 447 | 207.84 |
| **G5** | 22 | 398 | 227.70 |
| **M1** | 28 | 507 | 96.88 |
| **M2** | 45 | 155 | 63.29 |
| **M3** | 14 | 228 | 89.88 |
| **M4** | 19 | 966 | 175.13 |
| **M5** | 12 | 1523 | 119.32 |
| **M6** | 13 | 658 | 72.09 |

(b) (Stats on Visual Recurrences)

TABLE 2.6: Separators/Detected Recurrences Length Statistics (in frames) - Training Dataset.

Indeed, generally, if we consider the mean length, the detected recurrences are longer than the separators. The results obtained after filtering do not really match/agree to the data presented in Table 2.6(b), since the training for finding a threshold is made on the training dataset while the filtering is applied on the test dataset. When considering the lengths presented in Table 2.6(b) we should have had changes in precision and recall (Table 2.7 column "$\Delta_{max}$ = 720 f") for M4 and M5 as these have recurrences longer than 720 frames. Instead the filter changed the results for G3, M1 and M5. This already proves that the lengths of detected recurrences vary from one episode of a TV program to another.

| Test | **Initial** | | $\Delta_{max}$ **= 720 f** | | $\Delta_{min}$**= 14 f** | | $\Delta_{min}$ **= 17 f** | | $\Delta_{min}$ **= 22 f** | | $\Delta_{min}$ **= 41 f** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Datasets | **P** | **R** | **P** | **R** | **P** | **R** | **P** | **R** | **P** | **R** | **P** | **R** |
| **G1** | 0.820 | 0.977 | 0.820 | 0.977 | 0.828 | 0.977 | 0.854 | 0.977 | 0.863 | 0.977 | 0.882 | 0.977 |
| **G2** | 0.900 | 1 | 0.900 | 1 | 0.900 | 1 | 0.900 | 1 | 0.900 | 1 | 0.900 | 1 |
| **G3** | 0.935 | 0.954 | 0.955 | 0.932 | 0.935 | 0.954 | 0.935 | 0.954 | 0.935 | 0.954 | 0.935 | 0.954 |
| **G4** | 1 | 0.723 | 1 | 0.723 | 1 | 0.723 | 1 | 0.723 | 1 | 0.723 | 1 | 0.723 |
| **G5** | 0.143 | 0.028 | 0.143 | 0.028 | 0.167 | 0.028 | 0.182 | 0.028 | 0.182 | 0.028 | 0.182 | 0.028 |
| **Global** | 0.849 | 0.649 | 0.852 | 0.646 | 0.860 | 0.649 | 0.876 | 0.649 | 0.880 | 0.649 | 0.888 | 0.649 |
| **M1** | 0.769 | 0.167 | 0.750 | 0.146 | 0.833 | 0.167 | 0.833 | 0.167 | 0.818 | 0.146 | 0.900 | 0.146 |
| **M2** | 1 | 0.566 | 1 | 0.566 | 1 | 0.559 | 1 | 0.559 | 1 | 0.559 | 1 | 0.539 |
| **M3** | 0.636 | 0.101 | 0.636 | 0.101 | 0.667 | 0.101 | 0.667 | 0.101 | 0.611 | 0.079 | 0.687 | 0.079 |
| **M4** | 0.931 | 0.302 | 0.931 | 0.302 | 0.931 | 0.302 | 0.931 | 0.302 | 0.931 | 0.302 | 0.931 | 0.302 |
| **M5** | 0.808 | 0.826 | 0.822 | 0.804 | 0.808 | 0.826 | 0.808 | 0.826 | 0.808 | 0.826 | 0.892 | 0.739 |
| **M6** | 0.84 | 0.75 | 0.84 | 0.75 | 0.84 | 0.75 | 0.869 | 0.714 | 1 | 0.357 | 1 | 0.357 |
| **Global** | 0.882 | 0.399 | 0.885 | 0.395 | 0.889 | 0.397 | 0.892 | 0.395 | 0.904 | 0.370 | 0.935 | 0.357 |

TABLE 2.7: Evaluation of the impact of a length filter on the visual recurrences.

Moreover, the data in Table 2.6(b) was computed on all the detected recurrences. In order to have a real view on the differences in length, the separators should be compared to their corresponding recurrences (see Table 2.8). To make it more clear, we illustrate this in Figure 2.17. In orange is the episode of a TV program that we want to structure, and the recurrences detected on this episode. In blue are the separators of the ground truth created for this episode. If we consider the example in this figure, then, in Table 2.6(a) there are presented the statistics on the lengths of S1, S2 and S3, in Table 2.6(b) those on R1, R2, R3, R4 and R5, while Table 2.8 presents information about S1 and S2 and respectively their correspondents R1 and R3.

For a general idea, we also computed in Figure 2.9 the histograms for the lengths of recurrences that are separators, corresponding to Table 2.8.

Table 2.8 shows that the mean lengths computed for the recurrences corresponding to separators are still generally longer than those of the separators in the ground truth for some of the programs (i.e. G1, G4, M1, M4, M5).

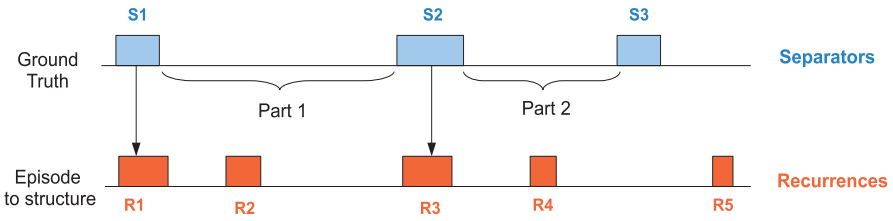


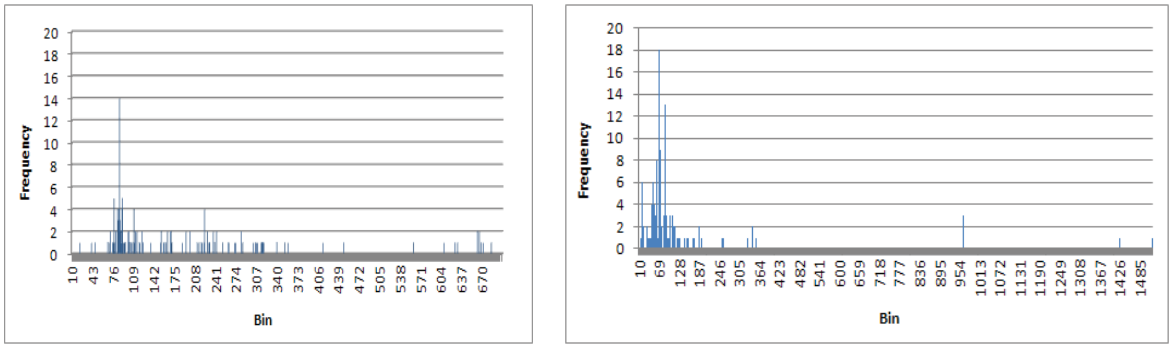

FIGURE 2.17: Separators and their corresponding recurrences for an episode of a TV Program.

| Training Datasets | Separators | | | Recurrences | | | Difference (abs. val.) | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Min** | **Max** | **Mean** | **Min** | **Max** | **Mean** | **Min** | **Max** | **Mean** |
| **G1** | 47 | 112 | 92.75 | 41 | 282 | 104.94 | 0 | 171 | 19.03 |
| **G2** | 109 | 118 | 113.33 | 109 | 114 | 110.89 | 1 | 9 | 2.89 |
| **G3** | 193 | 720 | 399.75 | 193 | 684 | 400.21 | 0 | 296 | 36.39 |
| **G4** | 69 | 243 | 144.17 | 22 | 447 | 208.26 | 0 | 368 | 96.76 |
| **G5** | 100 | 164 | 125.3 | 74 | 74 | 74 | 40 | 40 | 40 |
| **M1** | 42 | 121 | 75.87 | 35 | 151 | 82.75 | 0 | 46 | 20.12 |
| **M2** | 44 | 99 | 70.52 | 45 | 155 | 68.9 | 0 | 79 | 4.71 |
| **M3** | 24 | 195 | 137.8 | 14 | 193 | 88.8 | 2 | 137 | 49 |
| **M4** | 59 | 342 | 120.36 | 43 | 966 | 170.83 | 0 | 907 | 62.94 |
| **M5** | 45 | 126 | 68.15 | 17 | 1523 | 127.02 | 0 | 1440 | 71.71 |
| **M6** | 13 | 133 | 51.55 | 17 | 166 | 41.28 | 1 | 89 | 14.39 |

TABLE 2.8: Visual Corresponding Separators-Recurrences Lengths Statistics - Training Dataset.

Separators are generally preceded and followed by frames with the anchor or with a long view of the set which are considered similar and thus are added to the detected occurrences. This however increases the length with only a few frames while we observe in the column "Difference (abs. val.)", where the differences between the separators and their corresponding recurrences can go to more than a few frames. These situations correspond to the cases when a separator is detected together with a part of the program that follows it. The most common example is when the part of the program is composed of a front view with the anchor on an intense dominant-colored static background (M5), a long view of the set (M4) or a small view of the competitors on synthetic images background (G3).

What generally happens is that the algorithm for the detection of video recurrences contains a part that proceeds with the extension of the detected recurrent sequences with a few frames, if these ones are similar among the occurrences of the recurrent sequence, within a certain threshold. Basically, in a first step, the algorithm (see Section 2.1) detects identical sequences using sophisticated and robust descriptors (KVDs). Than it proceeds




(a) (Games) (b) (Mag&News)

TABLE 2.9: Histogram of the lengths of the visual recurrences that are separators - Training Dataset.

with the extension of the sequences using a rougher and less precise DCT-based descriptor (BVDs) that allow the detection of visually similar frames. This last descriptor can find frames that are similar but not identical, particularly when comparing synthetic images or containing less information. In this cases the energy is distributed among the coefficients of the DCT and is no longer concentrated in the low frequencies. The energies for images with little information is thus dispersed or it is very small, leading to small Hamming distances between the compared images.

An example is in Figure 2.18 where 2 occurrences from 2 different episodes of a such recurrence are presented. The first part of the occurrences corresponds to the true separator while the second part shows the anchors in the set presenting the next topic of the show. Obviously the detected recurrence is longer than the separator. Nevertheless, we consider this a separator as our interest is to delimit this part of the show from the others and indicate its beginning. This is done, as the recurrence starts with the separator that marks the end of the previous part of the show and the beginning of the new one. We prefer thus to consider it as a detection instead of considering it a false alarm and completely neglect its detection.

Other similar examples are illustrated in the bottom of the Figure 2.18. These correspond and replace the images displayed for the 2 episodes, but for other programs.

Also a similar situation is when right after/before the separator the price won by the competitors of a TV game is presented. If this price remains the same from one episode to another, the separator and the price will be detected as one recurrence as the sequence of images is identical within the analyzed episodes.

All these make it very hard to find a maximal length of separators and moreover find a general threshold. For some programs a separate training for each program could work but these cases are however not too frequent to worthwhile. This is shown by the high values of the precision presented in Table 2.7. Furthermore as we have already said, the length of recurrences vary a lot (we have noticed the difference between the training and test dataset) and depend of a lot of factors like the different editing effects of the TV program and also the strengths and weaknesses of our recurrence detection algorithm.

Nevertheless, if we compare the statistics on the visual recurrences in 2.6(b) and the corresponding ones in Table 2.8 we notice that for some cases the mean length is smaller


FIGURE 2.18: Example of visual recurrence containing a separator and a part of the program.

in the second table (G1, G5, M1, M6), which means that generally the false alarms are long, but for others the mean length increases (G2, G3), meaning that the false alarms are smaller. For others (G4, M2, M3, M4) the value remains almost the same meaning that the recurrences and the separators have very similar lengths.

We have thus decided to test a length filter imposing a minimal value for the recurrences that are separators. In this case we have chosen a minimal threshold ($\Delta_{min}$) that generally corresponds to some minimal lengths of recurrences that are separators, and filtered out all the recurrences having the length smaller than the threshold. The results are presented in Table 2.7 in the columns represented by $\Delta_{min}$.

Visibly for the case of TV games, the minimal length filter has a positive impact consisting in an increase of the precision with 4% without influencing the recall. It is not the case for the magazines and news where recall is decreasing with the increase of the precision.

In conclusion, the length filter can be useful for some TV programs but we have to keep caution when using it as the choice of the threshold, even if trained on a significant training dataset, can be influenced by noise induced by the different editing effects of TV programs and also by the algorithm used for the detection of recurrences.

Consequently, for the moment, we prefer not to use this filter as our interest is to keep as many separators as possible (a high recall value) for the next phase of classification (Chapter 4). Moreover, the results we have for the precision, at the time being are quite satisfactory and the improvement the length filters could bring is however not very important. We do however keep it in mind as a possible post processing step.

### 2.4.6 Exp.4: Program structuring using a database of separators

As already mentioned in Section 2.3.2, another interesting use-case would be to perform the structuring of each episode as soon as it arrives using separators that have been previously detected and stored in a database. For evaluating the performance of this approach, we used a dataset composed of episodes from some of the programs described in Annex A. We refer to G1, G3, G4, M2, M5, M6.

For each TV program, three consecutive episodes were used for the detection of separators to be stored in the database. The next episodes, broadcasted during one week, are used for testing the effectiveness of the solution.

As described in Section 2.3.2, the algorithm that performs the video matching works on set of frames (sliding window). In order to find the most suited number ($n$) of frames in such a set, we computed several experiments by varying n. The obtained results are presented as graphs in Figure 2.19.

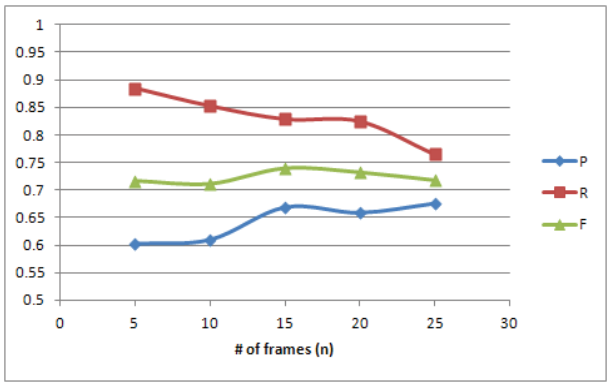



FIGURE 2.19: Results obtained for video matching when varying the number of frames in the sliding window.

The results presented in Figure 2.19 show that the number of frames in the sliding window is a significant parameter. The smaller the number of frames the higher is the recall. As usual there is a compromise between precision and recall as the precision increases with the increase of number of frames in the sliding window. This is explained by the fact that with a small $n$, the matching is computed between less frames. Consequently, the chance to obtain a similarity measure that exceeds the imposed threshold is higher than when testing on a large number of frames. Moreover small separators are rather detected when using a small sliding window. Likewise, separators that have only small parts that are identical can be detected and then extended as described in Section 2.3.2. Just for information, a large n increases the computation time as we used an exhaustive algorithm for matching.

To compare the results obtained when using a database of separators with those obtained with the classical approach described in Section 2.3.1, we placed them together in Table 2.10.

The first 2 lines in Table 2.10 correspond to the classical approach. "*Classic init.*" or *A1* refers to the results obtained for the proposed dataset, in the first stage of the classical approach, meaning the recurrences obtained without any filtering (see Section 2.1 where the corresponding approach is detailed). "*Classic final*" or *A2* shows the results obtained with the classical approach in the final stage, meaning after recurrence filtering (please

| | Approach | Precision | Recall | F |
|---|---|---|---|---|
| A1: | Classic init. (A1) | 0.521 | 0.884 | 0.656 |
| A2: | Classic final (A2) | 0.801 | 0.849 | 0.824 |
| A3: | DB max F (A3) (15 frames) | 0.668 | 0.829 | 0.740 |
| A4: | DB max R (A4) (5 frames) | 0.602 | 0.884 | 0.716 |
| A5: | DB max P (A5) (25 frames) | 0.675 | 0.765 | 0.717 |

TABLE 2.10: Exp.4: Performance database structuring approach.

refer to Section 2.2 for more details).

The last three lines in the table (*A3, A4, A5*) provide the results obtained with the algorithm that uses for structuring the database of separators. Each corresponds to the results obtained with different lengths of the sliding window: the first represents the best obtained F measure, the second the best recall and finally the third the best obtained precision.

In these experiments we have used the Viterbi algorithm to compute the temporal coherence of the detected matching frames, as described in Section 2.3.2. We have however also tested another algorithm that considers at each iteration all the possible sequences of matching frames and finally chooses the longest one. This last algorithm returned very similar results. The computation time and the resources used by this last algorithm are though much more elevated. We have also tried using a differential cost, meaning that for consecutive ids of matching frames we increased the cost by 1 while for frame ids that differ by 2 (due to the offset that we allowed - see Section 2.3.2 for more details) we increased the cost by 0,5. The results obtained in this case were still the same.

Back to Table 2.10, if we compare A1 and A4, we notice that the recall is the same, meaning that the two approaches recover the same number of separators. The precision for A4 is higher as the separators that have been stored in the database and that were used for matching, correspond to filtered recurrences. Consequently in the database there should be few or even no false alarms. If we however compare the final results of A2 with the results corresponding to the best F measure (A3), as should also be most important, the classical approach outperforms the one based on a database (see F measure).

In conclusion the two approaches can be similar in performance, but the parameters settings have to be carefully chosen in order to obtain the best desired results. Nevertheless, the use of a database includes some risks. The most important would be the absence of separators from the database. If a separator was not included in the database, it is impossible to detect it in future episodes. This problem could still be resolved by periodically updating the database. For the moment we do not dispose of a sufficiently long test dataset, that would allow us to experiment the minimal time period when the database should be updated. Consequently we prefer using for the experiments in the next sections the classical approach.

## 2.5 Conclusion

This chapter, studied the possibility of structuring episodes of a large category of TV programs, using the separators detected among the intra- and inter-episode visual recurrences. In a first step, the initial hypothesis, that separators can be found among the visual recurrences of a program, has been validated. Indeed a large number of separators have been detected and used afterwards to structure the analyzed episodes. There were also some limitations regarding the separators that present a logo superposed on live images with the set or separators animated by different editing effects. They do not have recurrent images, as their background is different, and thus they were not detected among the visual recurrences. Therefore, they were not identified as separators.

Among the detected visual recurrences there were also false alarms, presenting sequences with similar images but that are not separators. To filter them out, in a second step, a prior and a temporal filter have been used. The prior filter targeted the intra-episode false alarms. It obviously improved some of the results as it is hardly possible to have a separator created specifically for a single episode. Exceptions existed also in this case but these were due to the noise in videos or to some specific cases treated differently in the recurrence detection algorithm. The temporal filters that targeted mostly the inter-episode false alarms, showed important results but only for the programs with temporal stability, as the TV game shows. A length filter has also been studied but the obtained results were not conclusive.

Two different structuring methods have been presented, one using directly the separators for structuring, the other using a database of separators. Both approaches showed similar performances.

In this chapter we have only used the visual information of separators. The promising results we obtained with the visual approach encouraged us to proceed and consider also the audio information of separators. The idea is that when using both approaches, one could overcome the limitations of the other and provide thus the necessary for a better structuring. In the next chapter we will thus focus on the audio content of separators and propose a separator detection approach based on audio recurrences.

# Chapter 3

# Audio analysis of recurrent TV programs

**Summary**



The objective of this thesis is to propose an approach for unsupervised TV Program structuring. We suggested a method based on separators. Separators are recurrent audio/visual sequences that separate the different parts composing a program. In the previous chapter, we have considered only the visual information of separators. We were then able to detect only separators that have the same visual content. Our solution did not address the audio jingles, that are separators which share the same audio content but not necessarily the same visual content (i.e. the same sequence of images). In this chapter we extend our study by considering also audio information of separators.

We make use of the same assumption on which the visual approach is based, meaning that separators are repeated and can be detected as recurrences. However, in this case we consider the audio recurrences.

This chapter focuses thus in a first place on audio recurrences. Our target is to deal with a large category of TV programs. In this sense, our purpose is to study different types of recurrences that can appear during a TV program broadcast and propose the best suited descriptors that allow their identification. We also focus on a method for recurrences detection. The objective is to evaluate the effectiveness of the considered method and choose the best suited descriptor that will allow us to detect the separators needed to structure programs.

## 3.1 The different types of Audio recurrences

We have conducted first a study of the different types of audio recurrences that can appear during a TV program broadcast. As a result, we have classified them into the following three categories:

1. **Near-identical recurrences**. Examples of such recurrences are TV jingles, opening credits, program specific sounds, music...
2. **Noisy near-identical recurrences**. These recurrences generally refer to jingles with background noise like applauses or music.
3. **Speech recurrences**. This last category of recurrences refers to the spoken language and points out repeated words, expressions or even phrases.

To detect all these recurrences, we propose a method for audio recurrence detection and a set of basic audio descriptors that perform differently for each of the categories previously defined.

The algorithm for audio recurrence detection and the proposed descriptors are presented in the next sections.

## 3.2 Audio recurrence detection algorithm

The structure of the audio recurrence detector is presented in Figure 3.1 and is described below.

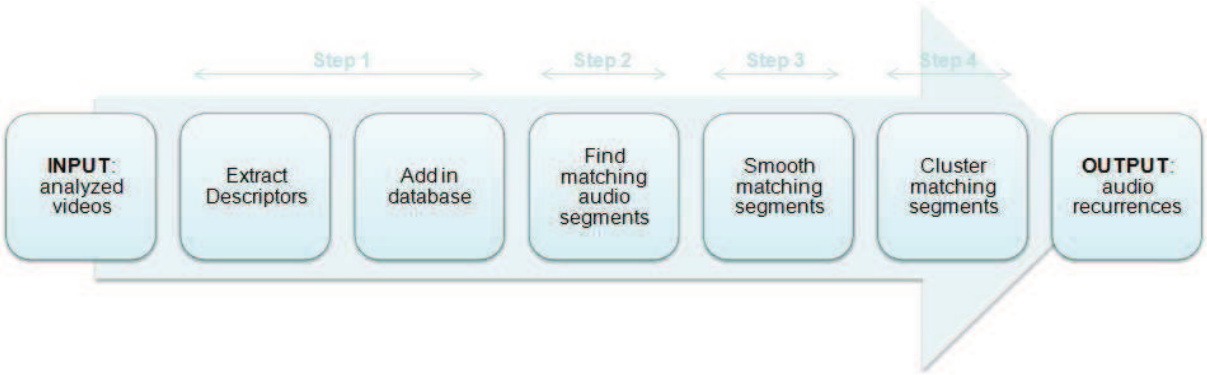



FIGURE 3.1: Audio recurrence detection scheme.

**Step 1**: Based on a sliding audio frame, basic audio features called *sub-descriptors* are extracted from all the analyzed episodes.

For each episode, a descriptor containing all the time-sorted sub-descriptors of the program in addition to some identification data is built. All the computed descriptors are then stored in a reference database.

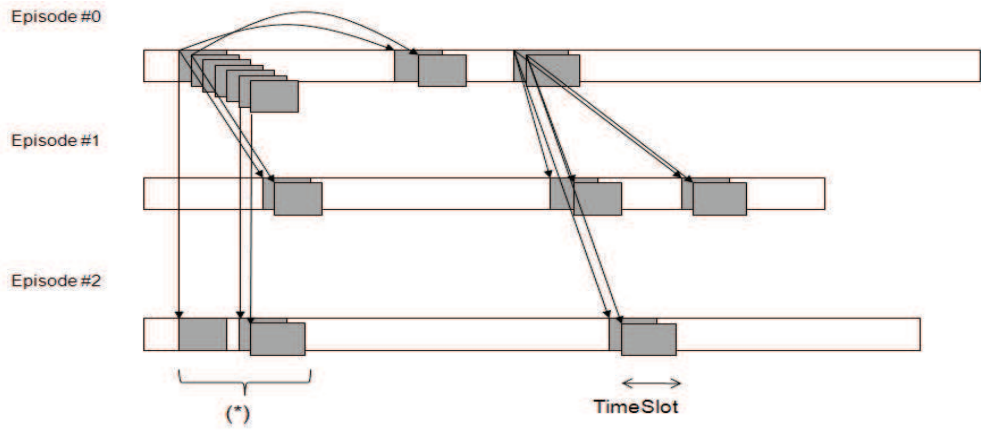



FIGURE 3.2: Audio recurrence detection - Step 2 - Matching segment pairs. "*" refers to neighboring segments that are merged during Step 3.

**Step 2**: Each of the episodes in the database in turn, will become a query. A sliding window (having a predefined number of sub-descriptors) that we call *TimeSlot* is used to match the sub-descriptors of a query with the rest in the reference database.

A match is confirmed if the distance between the two falls below a pre-determined similarity threshold. Obviously, self-detections will not be considered further. At the end of the processing, a set of matching segment pairs are obtained (as depicted in Figure 3.2).

**Step 3**: The matched segments identified in Step 2 are smoothed and extended with overlapping segments. If the temporal distance between two detected segments falls under a certain threshold, these segments are merged (see (*) in Figure 3.2).

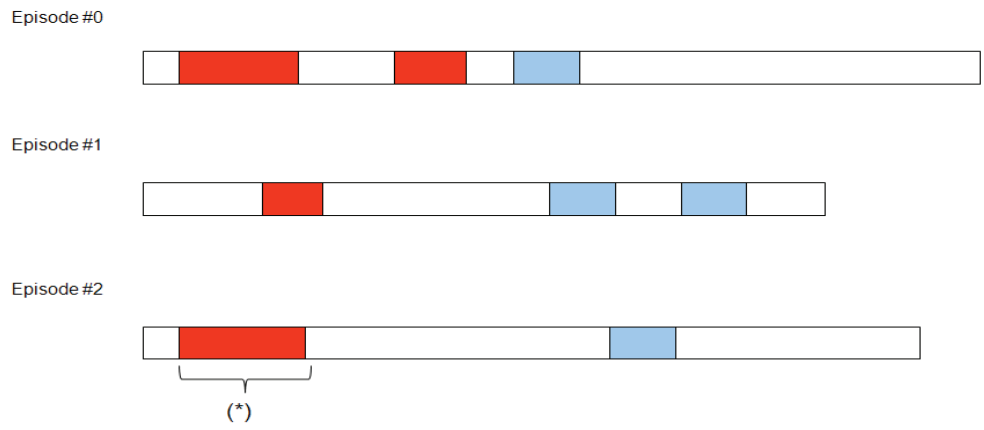



FIGURE 3.3: Audio recurrence detection - Step 4 - Clusters of matched pairs of segments.

**Step 4**: The matched segments are analyzed and clustered. First, each pair of segments obtained in Step 3 instantiates a cluster. Second, overlapping clusters are merged: Two clusters overlap if a segment from a same program belongs simultaneously to both clusters.

Figure 3.3 shows the results of the clustering corresponding to the detections in Figure 3.2. The two colors of the segments in Figure 3.3 (red and blue) correspond to the two clusters to which the sets of matching segments in Figure 3.2 belong to.

In Step 2, *TimeSlot* is fixed. This means that we try to match two sliding windows of the same size. The matching of the query and the reference sliding windows is based on a linear procedure that computes a distance between sub-descriptor $i$ in the query sliding window and only sub-descriptor $i$ in the reference sliding window. The matching procedure considers in this case all of the computed distances. However, it is only able to match near-identical audio segments. Low distortions such as timeshifting or signal alteration might occur between recurrences. A typical issue is enabling the alignment of identical words pronounced slightly differently even by a same speaker. Recent works [MGB09, PG08] propose a way to allow more variability by using Dynamic Time Warping (DTW). As shown in Figure 3.4.a, DTW fills a distance matrix between two segments and computes the best alignment by summing up distances along all possible paths, allowing insertions and deletions of sub-descriptors. A perfect alignment between two identical sub-descriptors results in a diagonal path from the first matrix element (up-left) to the last element (bottom-right) with a 0 distance value.

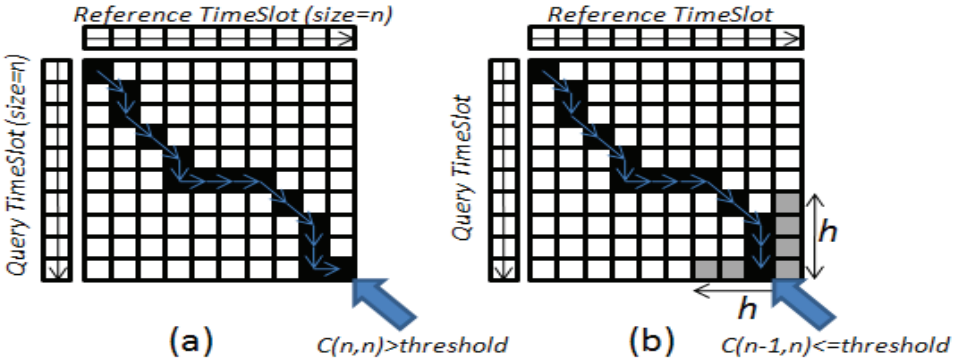



FIGURE 3.4: (a) Best alignment of the query&reference using DTW/(b) Extended ending points for DTW.

Because of allowed insertions and deletions of sub-descriptors, the resulting score on the last matrix element may be higher than our fixed similarity threshold. It is possible that, the best found path joins the last row or the last column earlier with a score lower than the similarity threshold. We propose thus to extend the number of possible ending points in the last row and in the last column of the DTW distance matrix. In our implementation, we extended these ending points by $h$ elements (typically 20% of the *TimeSlot* length). Figure 3.4.a shows a unique ending point on the last matrix element with a score higher than the threshold, whereas Figure 3.4.b shows the same path ending on the next to last column of the matrix with a score validating the recurrence. On this figure, the added up distance function along the path is labeled C(i,j), where i and j are the indexes of the matrix.

## 3.3 Audio descriptors

Three types of audio descriptors have been investigated in this study: Perceptual Audio Hashing, MFCC- and Phoneme-based. They are described in the following subsections.

### 3.3.1 Perceptual Audio Hashing Descriptor

This descriptor has been proposed by the popular approach described in [HKO01] for audio track identification. It is often referred to as Perceptual Audio Hashing (PAH). It relies on the short time spectrum variations. Since its introduction, it has received much attention and many improvements, refinements and studies have been published [BHMS2, PKY06].

Yet in this paper, we have decided to stick as much as possible to the original scheme since it offers already sufficient provision for detecting nearly identical audio recurrences at a reduced complexity.

Our basic schema for extracting the PAH descriptors is presented in Figure 3.5, as introduced in [HKO01]. The audio signal is converted to mono at 16 kHz. It is then divided into overlapping frames and 33-band energies are computed for each frame. The sign of the energy variations along the band and time axis are derived giving a 32-bit sub-descriptor per frame.

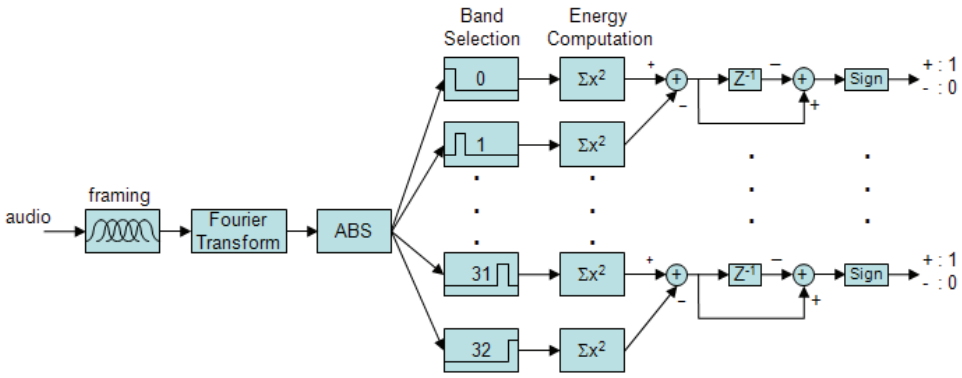



FIGURE 3.5: PAH sub-descriptors extraction.

The set of time-sorted sub-descriptors of an episode is called a *descriptor*. Descriptors of all of the episodes are stored in a reference database. In parallel, the position of each 32-bit sub-descriptor is recorded in a look-up table. The recorded position is composed of the reference episode identifier and the frame number in the corresponding table. When a query is processed, the position of each sub-descriptor is searched for in the reference database thanks to the look-up table. This will indicate positions in the reference descriptors where the query and the reference match by at least one sub-descriptor. In order to cope with potential audio alterations, we have introduced a parameter, denoted $\varphi$ (the *accuracy*). It can tolerate a certain difference when searching in the look-up table, as also explained in [HKO01]. This increases the complexity but improves the recall in case of strong alterations. The $\varphi$ parameter will tolerate, when searching in the look-up table, a distance of 0 bit ($\varphi$="high"), 1 bit ($\varphi$="medium") or 2 bits ($\varphi$="low").

The distance chosen to compare the query and the reference sliding windows is the average Hamming distance computed between sub-descriptors.

In the experiments referred to in this thesis we have used the following parameters: Sampling rate: 16000 Hz, Frame length: 1024 samples (64 ms), Hop size (distance from the first sample in the current frame to first sample in the next overlapping frame): 64 samples (4 ms), Number of bands: 33, Lower frequency: 1000 Hz, Higher frequency: 7000 Hz.

### 3.3.2 MFCC-based Descriptor

Considering works described in [MGB09] and [PG08], we propose to consider Mel Frequency Cepstral Coefficients (MFCC) as sub-descriptors in our study. Whereas audio hashing generally focuses on exact and music-oriented matches, MFCC-based descriptors associated to Dynamic Time Warping (DTW) enable fuzzier matches such as words or multi-words repeated by a same speaker.

Based on a sliding frame, features are extracted from the audio signal. The parameters that we used are: frame size of 32ms, hop size of 10ms, 26 channels and 22 cepstral lifters. A 13-dimensional vector is created for each frame, containing the 12 first cepstral static coefficients and the signal energy of the frame.

The distance chosen to compare two MFCC vectors is generally the Euclidian distance. We use it with respect to the choices made in the literature, even if it does not provide a [0,1] distance interval, which makes a friendly similarity threshold more difficult to find. Of course, hashing based indexing techniques cannot be directly applicable on these descriptors. Therefore, the query processing time would be higher compared to hashing-based systems. The exhaustive similarity search will be controlled by the $\varphi$ parameter which, in this case, will allow in high mode to search the matching of all the sub-descriptors, in medium mode of half sub-descriptors and in low mode of only a fifth sub-descriptors.

### 3.3.3 Phoneme-based Descriptor

Previous descriptors are expected to be efficient for music jingles or repeated records, clean or lightly noisy, and potentially for some spoken repetitions. Nevertheless, several spoken recurrences which may be useful for structuring would not be detected because of various intra- and inter-speakers differences. From this perspective, we propose to test another kind of audio descriptor: the phoneme. It is the smallest segmental unit of sound employed to form meaningful contrasts between utterances, and it has the same function for all speakers[1]. By translating an audio signal into a phonetic sequence (as shown in Figure 3.6), we project a continuous vocal signal to a discrete and finite space of $N$ phonemes. Literature already mentions some phoneme-based techniques to discover patterns in speech [SDV07].

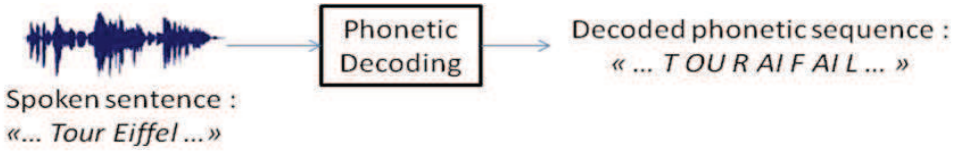



FIGURE 3.6: : Example of phonetic decoding.

[1]Source: Wikipedia

In our case, the audio signal is processed by the syllable-based phoneme recognizer that is described in [BC07]. It decodes 35 phonemes of the French language in a 1-best transcription. It achieves a 23% phoneme error rate on news. No evaluation has been done on the entertainment shows related in this article, but the phoneme error rate will probably be higher, due to a more spontaneous kind of speech.

In our algorithm, phonemes can be seen as hashing values. The distance chosen to compare two phonemes is the strict equality. Identical phonemes will have a distance value of 0, and different phonemes will have a distance value of 1. Future improvements of this distance can be achieved by injecting the information of proximity between phonemes. This can be provided for example by a phoneme confusion matrix which captures the systematic errors of the phonetic decoder.

Whereas the PAH and MFCC-based descriptors process speech and music in a similar way, this phonetic decoder is only trained on speech. As no speech/music segmentation is made on the audio signal, the results on music will probably be silence or weird sequences of phonemes. Nevertheless, it is interesting to analyze the decoding results on music segments to conclude on the utility of a phonetic decoding for recurrences detection.

Contrary to other descriptors, phonemes do not have a fixed duration. The "TimeSlot" parameter will not be in this case defined in "seconds" but in "number of phonemes". The $\varphi$ parameter is not needed.

## 3.4 Experiments

We have performed experiments, as for the visual case, on real TV broadcasts. In a first experiment we study each of the proposed descriptors and evaluate the types of recurrences these descriptors are able to identify. In a second experiment we return to the main subject of this thesis meaning to detect separators as repeated sequences in recurrent TV programs and use them to structure these programs. The objective is, as for the visual case, to validate the main idea behind our approach, meaning that separators can be detected as recurrences among the episodes of a TV program. Only that this time, separators are searched among audio recurrences. As already stated, not all the recurrences are separators. Consequently, the third experiment evaluates the effectiveness of filtering "false alarms" from the set of detected recurrences. In a last experiment we propose and study a filter-based on the length of separators.

We present hereafter, the experimental context of our evaluation, the evaluation protocol used to analyze the obtained results and finally the experiments and obtained results.

### 3.4.1 Experimental context

For experiments, we have used TV programs from the dataset described in Annex A. Two experimental contexts are needed.

**A first experimental context**, is used for a first experiment where the goal is to study the proposed descriptors and the types of recurrences they detect.

To do so we used a dataset composed of five programs described in Annex A. We refer to $G_3$, $G_4$, $G_5$, $M_2$, $M_3$ and $M_6$.

As it is very hard even impossible to create an exhaustive audio recurrence ground-truth, we made a selective evaluation, in which we focus on audio recurrences that could

be used for structuring purposes.

Regarding the chosen dataset, special audio/video sequences (*jingles*) are introduced between the different stages of the game, for the case of the TV game shows, or between the anchor person scenes and the reports, for the case of magazines and news programs. These sequences are generally recurrent, and they can repeat inside or between different episodes of the same program. These recurrences are thus very relevant structuring elements. We will thus focus on these ones when analyzing the detected recurrences.

The dataset for this experiment is split into two categories: TV Games Programs (DS1) and separately Magazines and News Programs (DS2). The reasoning behind this split is due to the similarities of magazines and news programs that generally have the same structure and share the same types of recurrences.

**A second experimental context** is used for the rest of experiments in this chapter. It is no longer focused on recurrences but more on separators that are manually annotated in each episode in order to create the ground-truth. As the separators of a TV show are the same regardless the audio or visual content, for the audio analysis, we used the same ground-truth as the one manually created for the visual analysis in Chapter 2.

We compute experiments on set of four episodes (the current episode plus three episodes in the history) in order to detect the separators for the fourth episode. Only these last separators will be considered for evaluation. The study in Section 3.6 gives more details about the number of episodes considered in the history.

This experimental context has already been used in Chapter 2. For more details the reader can refer to the mentioned chapter, Section 2.4.1.

#### 3.4.1.1 Dataset DS1 of the first experimental context

For the case of DS1, the possible recurrences that could be identified are the jingles used to separate the different stages of the game, specific sounds - like the beeps when pushing a button, the sound introduced right after a right/wrong answer or the sound that marks a new question or an important decision made by a competitor, but also background music and repeated words. However the recurrences relevant for the structuring of the program can be divided on two classes:

1. for *light* structuring: jingles that separate the stages in a game - in our case these are inter-episode recurrences. These type of jingles will be identified as separators in the next experiments.

2. for *dense* structuring: jingles and specific sounds - in our case these are inter- and intra-episode recurrences. Some of these jingles could be used for structuring inside each part of a program. We have however to keep in mind that this thesis focuses on structuring programs in their main parts and what is inside each part is not our scope for the moment.

In order to study both intra- and inter-episode recurrences, we have divided the DS1 into subsets. Each subset is composed of two episodes. We performed experiments on each subset separately. In this manner the results will contain both intra- and inter-episode recurrences.

#### 3.4.1.2 Dataset DS2 of the first experimental context

For the case of DS2, the possible recurrences that could be identified are the jingles (*"clean"* or with background noise), repeated words or music. The recurrences that are relevant for the structuring are the jingles. Excepting the opening credits, most of the recurrences are also repeated inside the same episode. Therefore, for this dataset, we perform experiments separately on each episode as there is no need to use sets of episodes that generally consume more time and resources.

### 3.4.2 Evaluation protocol

For the first experiment, as it is very hard to create an exhaustive ground-truth regarding the recurrences that can be detected with the different descriptors and the proposed approach, we did a selective manual evaluation of the results. The outcomes/findings of the manual analysis of the detected recurrences are detailed in the next subsections for each descriptor separately and for both DS1 and DS2.

For the remaining experiments the evaluation protocol we used is the one described in Chapter 2, Section 2.4.2

### 3.4.3 Exp.1: Analysis of audio recurrences detected with the proposed approach and descriptors

#### 3.4.3.1 PAH Descriptor

As relevant recurrences are generally short synthetic recorded audio sequences with speechless music and considering the descriptors presented in Section 3.3, the most appropriate would be to use the Perceptual Audio Hashing to detect these recurrences. Depending on the category they belong to, they can vary in length between 250ms (for specific sounds) and 30s (for jingles). Consequently, we will use a TimeSlot between 200ms and 500ms and a threshold of 0.35.

The main observations by manual inspection of the results are:

• Most of the repeated jingles and specific sounds were found. What is important to notice is that identical jingles/specific sounds but that have a noise in the background (like applause or words) are detected only if they are compared to similar but "clean" sequences. If two sequences that have both applauses on the background are compared they will not be identified as recurrences of the same sequence.

• What was surprising in the first place, are the sequences with only applause, that were detected. On a closer look, we found that parts of these sequences were identical, meaning that these were recorded sequences. Indeed, the producers play recorded applause, in order to inform the public that they have to start applauding. However, for the case of DS2, we can consider these sequences relevant for structuring as quite often, the end of a report and the return to an anchor sequence is marked directly by applause of the public present in the set. As no jingle is used, we find that it might be useful to consider applause as structural elements.

• We also noticed that jingles combining recorded music and speech are generally detected. Contrarily, jingles combining recorded music with natural speech (2 in the case of each episode in dataset $M_3$ and 1 in the case of each episode of the dataset $M_4$) are not detected since different vocal frequencies are used even for a same utterance.

• Besides targeted recurrences, other audio recurrences were also detected. Examples are short identical sequences of background music. For example if a person is speaking with music in the background, the pauses between the words, where only music can be heard, are detected. Other detected recurrences are repeated words/groups of words. Generally for DS2, at the beginning of each episode, there is a brief summary of the reports that will be presented in the episode. Words or groups of words found in the report will also be repeated in this summary. Our approach will thus detect them. Our approach detects also several special sounds, like the gun shots or explosions, that appear in the reports of one of the magazines from DS2 or short.

• We also varied the $\varphi$ parameter. For a high $\varphi$, more false alarms appear. Most of them are composed of *subwords* pronounced in the same way, by the same person (generally women) or by different people.

• Knowing that most of the false alarms and the recurrences that are not relevant for structuring have very small lengths, we have increased the *TimeSlot* parameter (up to 1 or 2s) in order to avoid their detection. This indeed reduced the number of false alarms, but many specific sounds in TV games were thus missed. This could be important in the case of DS1 for the dense structuring task but not for light structuring. Another possibility is to decrease the similarity threshold but the risk is higher as recurrences are often not strictly identical. Also longer recurrences will be more segmented as only short parts will be found to be identical.

#### 3.4.3.2 MFCC- and Phonemes-based Descriptor

We noticed previously, in the description of the results obtained with the PAH descriptor, that there are jingles, relevant for structuring, that combine music and natural speech. As indicated in the previous section, these are not detected by PAH descriptor. We have consequently experimented the MFCC- and the phonemes-based descriptor for which we know that they are more adapted for word discovery.

Moreover, TV programs in DS2 have all informational content, with a carefully chosen and elaborate language. Generally, in order to accentuate an idea, certain key words are repeated throughout the report. We can use our approach in order to detect these repeated keywords, that could afterwards be exploited to construct audio summaries or to permit a fast access to relevant parts of a program [MGB09].

##### *MFCC-based Descriptor*

As we want to detect matches of words or multi-words, we used the MFCC-based descriptor associated with the DTW. We conducted the experiments using a low $\varphi$ as the computation with high $\varphi$ and DTW is very time consuming.

The results are similar to those obtained using PAH descriptors. Recurrences containing live/natural speech have not been detected. Some differences exist however. With MFCC-based descriptors, sequences with applause, alone or with background noise, are detected but are often wrongly clustered together with specific game sounds or with music. These side effects appear due to the clustering method we used, which takes into consideration temporal overlapping rather than the model of the pattern. Moreover, applause is often followed or superposed with specific sounds (for the case of DS1) or background music making these errors very possible. Public reactions or competitors hesitations of the same person are also detected. The differences are mainly due to the use of MFCC descriptor combined with the DTW. Without DTW, the results are quite similar to those obtained using the PAH descriptor. However, for the case of MFCC descriptor (when using or not the DTW), identical jingles but that have a noise on the background (applause) are grouped together even with other noisy jingles too (not only with "clean" jingles).

***Phonemes-based Descriptor***

We have also experimented the phonemes-based descriptor. For such descriptor, the *TimeSlot* and similarity threshold values must be chosen very carefully as they are highly correlated. We also have to consider the phoneme error rate which in our case might be around 30%. When using a small *TimeSlot* (that only covers a small number of phonemes), the similarity threshold should also be small and DTW is not relevant. On the contrary, when using a larger *TimeSlot* the use of DTW would be helpful.

In order to find relevant recurrences and considering the previous discussion, we use the repeated detection algorithm together with the DTW. We have chosen a TimeSlot of 2s and a similarity threshold of 0.2.

As expected, many repeated words and expressions have been found. For the case of DS1, these are generally competitors' names, words belonging to the description of the rules of the game, questions that are repeated by the anchor when a competitor did not understand it in a first instance. For DS2, important keywords and even sentences have been detected, e.g. *"Journée internationale des enfants disparus", "Festival de Cannes", "Dominique Strauss Kahn en attente de sa convocation devant la cour criminelle de New York"...*

Detected rates of jingles having natural speech were up to 100% for dataset $M_3$ and 30% for dataset $M_4$. In this case, detected recurrences were speaker independent. However, the background music still impacts the detection. If this music has a rich content, or it has more or less rythm or the volume is too high, there are high chances for the phonetical transcription to be erroneous (as in the case of the jingle in dataset G4). This could lead to missed detections.

Jingles or specific sounds are sometimes discovered but are not always coherently clustered as a phonetical transcription is not suited for this case. Other false alarms are clusters of words that are similar but that are not identical (e.g. the French words "*déborder*" and "*démontrer*").

It is often in TV games for the anchor person to introduce the next stage of the game by saying its name. We find that discovering the names of each stage of the game would be very interesting not only for structuring but also for indexing. The problem is that these names vary in length from 6 to 11 phonemes or more (i.e. *"Les people", "Le puzzle",*

*"Le coup par coup", "Le coup d'envoi", "Le coup fatal"...*). We have tried thus a very short *TimeSlot* of 0.5 (5 phonemes), and a threshold of 0.1. The results showed that too many recurrences and false alarms were detected. Even with a threshold of 0 (identical recurrences) the number of detected recurrences and false alarms is still too high. This is due to the small *TimeSlot* that allows the detection of too many similar sequences in the episode, but also to the clustering method we used which clusters together all the sequences that overlap even only by 1 phoneme. We have also tried to analyze the results obtained for a higher *TimeSlot*, using the DTW and a threshold of 0.1. Part of targeted recurrences has been identified but these were polluted by the still too many other detections. Either way, it is clear that it is very difficult to find the best values parameters. A small value for the *TimeSlot* means the detection of too many recurrences and false alarms. A small value for the similarity threshold results in missed detections while a higher one increases the number of false alarms. This would suggest recurrence detection alone is not sufficient to focus on specific recurrences. A post-processing classification step is required.

#### 3.4.3.3 Conclusion

The experiments showed that the PAH descriptor is the most appropriate for the detection of identical recurrences like jingles, music, recorded spoken language, special sounds. Most of the recurrences relevant for structuring are identified using this type of descriptor. Besides them other recurrences, not relevant for structuring or some false alarms are also detected but their number depends on the *TimeSlot* and similarity threshold parameters. No jingles having natural spoken language can be detected using this descriptor.

The results obtained using the MFCC based descriptor are similar to the previous ones. We can however say that it is less sensitive to noise. Even if we expected for the MFCC descriptor associated with the DTW to detect words or group of words, we have found that this descriptor is not adapted for the detection of matches between natural speech (words or group of words). It allows though the detection of public reactions or competitors hesitations while being person dependent. Also the applause sequences, which are detected by the PAH descriptor only if they are recorded sequences, are identified by this descriptor only in combination with the DTW.

The phonemes-based descriptor is the most appropriate for the detection of repeated words or groups of words. It is speaker independent and in combination with the DTW can even detect coherent sentences. The problem in this case is the parameters adjustment as the TimeSlot, the similarity threshold and the phonemes error rate are very correlated. It is almost impossible to detect small recurrences while keeping the results coherent. On the other hand, if we are interested in recurrences that have more than 15-20 phonemes this descriptor is the most appropriate.

### 3.4.4 Exp.2: Evaluation of audio recurrences detection results

In the previous section, we concluded that the most appropriate descriptor used for the detection of identical recurrences, like jingles, is the Perceptual Audio Hashing. In this section, we return to separators. Consequently, to detect separators we use the PAH descriptor and the audio recurrence detection approach described in Section 3.2.

Beforehand, we made a manual analysis of the results obtained with this descriptor. Hereafter, we show the obtained results, in terms of precision, recall and F score, relative

to the separators ground truth (Section 2.4.2). The lines *"Global G"* and *"Global M"* correspond to the results obtained globally for all the games in the dataset and separately for the magazines. The last line in table shows the global results on the entire dataset.

| Dataset | Precision | Recall | F |
|---|---|---|---|
| G1 | 0.217 | 1.000 | 0.357 |
| G2 | 0.024 | 1.000 | 0.046 |
| G3 | 0.107 | 0.940 | 0.192 |
| G4 | 0.129 | 0.885 | 0.225 |
| G5 | 0.052 | 0.918 | 0.098 |
| Global G | 0.088 | 0.942 | 0.161 |
| M1 | 0.230 | 0.948 | 0.370 |
| M2 | 0.420 | 1.000 | 0.592 |
| M3 | 0.411 | 0.722 | 0.524 |
| M4 | 0.284 | 0.986 | 0.441 |
| M5 | 0.116 | 1.000 | 0.208 |
| M6 | 0.705 | 0.890 | 0.787 |
| Global M | 0.298 | 0.929 | 0.451 |
| Global G&M | 0.137 | 0.934 | 0.239 |

TABLE 3.1: Exp2: Separators among detected audio recurrences using the PAH descriptor.

Table 3.1 shows a good detection of separators with very high recall values for games and magazines and news as well but poor precision.

As expected after the analysis computed in Section 3.4.3.1, the separators that have not been detected, generally correspond to separators containing natural speech. Even if the person who speaks says the same thing, she/he can say it with different frequencies. As the audio descriptors we used in our approach are not suited for spoken recurrences, these separators will not be detected. This is moreover the case for datasets $M_3$ and $M_4$. Also, separators that have a noise on the background (like applause or speech) are detected only if during the clustering, they are compared to other similar but "clean" separators. If two identical sequences but that have both applause on the background are compared they will not be found as similar. This problem mostly appears for separators in datasets $G_5$ and $M_3$. For dataset $M_6$, which refers to the news program, there are some separators that are only visual. They mark the transition from the anchor person scenes, where the next topic is introduced, to the report that presents this topic. They are composed of only images with a map of the country where the report was filmed.

Regarding the precision, as expected, the results are very poor for all the programs in the dataset. Together with the separators a lot of false alarms are detected. For magazines, these correspond, as also detailed in Section 3.4.3.1, to sequences of background music, repeated words/expressions, repeated parts of a report, special sounds from reports (i.e. gunshots, explosions), recorded applause, silences, repeated special editing effects or jingles for dense structuring (that could be used for structuring inside each part of a program - these are however not our concern as we try to structure programs in their main parts - consequently these were not included in the ground-truth).

On the other hand, TV games have 20% more false alarms than the magazines. Besides

repeated sequences of background music and silences, these majorly correspond to specific sounds played when the competitors push a button to answer a question, or the sounds indicating a correct or a wrong answer, recorded public reactions or recorded applause, etc.

With such small values for the precision, it is clear that a filtering step is needed. The next experiment treats this problem and presents the results obtained when a filter is used to remove false alarms from the detected recurrences.

### 3.4.5 Exp.3: Analysis of audio recurrences after filtering

In Chapter 2, we proposed two filters to remove the false alarms from the detected video recurrences. A first one, described in Section 2.2.1 filters out all the repeated sequences having all the occurrences coming only from the same episode of a program. The principle is that it is very unlikely/rare for a separator to be created specifically for a single episode. We continue with supporting this principle and thus apply the prior filter on the audio recurrences as well. The obtained results, in terms of precision (P), recall (R) and F-score (F), are presented in Table 3.2(b). As in the previous experiment the *"Global G"* and *"Global M"* correspond to the results obtained globally for all the games in the dataset and separately for the magazines, while the last line in table shows the global results on the entire dataset.

| Dataset | $P_{i-1}$ | $R_{i-1}$ | $F_{i-1}$ |
|---|---|---|---|
| G1 | 0.217 | 1.000 | 0.357 |
| G2 | 0.024 | 1.000 | 0.046 |
| G3 | 0.107 | 0.940 | 0.192 |
| G4 | 0.129 | 0.885 | 0.225 |
| G5 | 0.052 | 0.918 | 0.098 |
| Global G | 0.088 | 0.942 | 0.161 |
| M1 | 0.230 | 0.948 | 0.370 |
| M2 | 0.420 | 1.000 | 0.592 |
| M3 | 0.411 | 0.722 | 0.524 |
| M4 | 0.284 | 0.986 | 0.441 |
| M5 | 0.116 | 1.000 | 0.208 |
| M6 | 0.705 | 0.890 | 0.787 |
| Global M | 0.298 | 0.929 | 0.451 |
| Global G&M | 0.137 | 0.934 | 0.239 |

(a) (before prior filter - step (i-1))

| Dataset | $P_i$ | $R_i$ | $F_i$ |
|---|---|---|---|
| G1 | 0.247 | 1.000 | 0.396 |
| G2 | 0.024 | 1.000 | 0.046 |
| G3 | 0.120 | 0.940 | 0.212 |
| G4 | 0.138 | 0.885 | 0.238 |
| G5 | 0.052 | 0.918 | 0.099 |
| Global G | 0.093 | 0.942 | 0.169 |
| M1 | 0.405 | 0.948 | 0.568 |
| M2 | 0.523 | 1.000 | 0.687 |
| M3 | 0.459 | 0.722 | 0.561 |
| M4 | 0.373 | 0.986 | 0.541 |
| M5 | 0.271 | 1.000 | 0.426 |
| M6 | 0.779 | 0.890 | 0.831 |
| Global M | 0.437 | 0.929 | 0.595 |
| Global M&N | 0.155 | 0.934 | 0.266 |

(b) (after prior filter - step i)

TABLE 3.2: Exp3: Performance of prior filter on audio recurrences.

The global precision of the entire dataset increases with only 2% after this filtering step. However if we take a closer look to the results for each category of programs (games/magazines and news) separately, we observe that the global improvement of the precision is actually decreased by the small precision obtained in the case of game programs. For these last ones the improvement is too small to be considered. The false alarms described in the previous section, for TV games, are inter- and intra-recurrences and thus

these can not be filtered out with this prior filter that addresses only the intra-episode recurrences. On the contrary, the prior filter shows a contribution for the magazines and news where the precision increases with 14%, with a minimal increase of 4% for dataset $M_3$ and a maximal of 17% for dataset $M_1$.

The prior filter therefore proved to be more or less useful depending on the type of program that is analyzed. We will thus consider it in all the next experiments of the upcoming chapters.

Nevertheless, this prior filter addresses only the intra-episode false alarms. In Section 2.2.2, we presented a second filter that refers to inter episode false alarms. It is based on the temporal stability of separators and computes a kernel-based density estimation to find the areas of high concentrations where the separators are broadcasted. Still, in Section 2.4.4.2 we concluded that this filter is useless in the case of magazines and news as the temporal stability condition is not respected by the separators in these programs. Magazines and news are generally composed of reports that differ in number and length from one episode to another. We will thus not apply this filter on these programs.

We could however use it in the case of TV game shows. Though, the number of audio false alarms is too big for these programs. A density estimation of separators would thus be difficult to compute as false alarms are too many and spread during the entire episodes. These will thus influence a lot the choice of parameters for the temporal filter.

The use of the temporal filter is thus not justified for the filtering of audio recurrences. Consequently we will not consider it in the further experiments.

### 3.4.6 Exp.4: Audio Recurrence Length Filtering

The audio recurrence detection algorithm finds a large number of recurrences, especially for the game TV programs, among which a lot of them are not separators. This is due to the fact that each sound that is found similar to another, even if it is very short in length will be identified. As a result, short recurrences like the sound made when a competitor pushes a button to answer a question or the sound that indicates if an answer is true or false are detected. In a TV game for example, these kinds of sounds appear quite often leading to the large number of detections. As they are short, we thought that it might be interesting to test the length filter in this case too and see if it can improve the results by reducing the number of false alarms.

To do so, we first need to know the length of separators so that we would not filter out these ones too. We recall thus in Table 3.3(a) some statistics on the separators' length. In order to avoid overfitting we split the dataset into two. The statistics presented in Table 3.3 were made on the first half of the dataset. This will allow us to choose a threshold based on this data and to apply it on the second half of the dataset (the test dataset).

Our interest is the minimal length of separators. The fixed threshold should be under this minimal value. We inform that the data presented in Table 3.3 refers to separators excluding the opening and closing credits which generally are the longest jingles in a TV show.

As illustrated in Table 3.3(a) this minimal value is around 10-13 frames which correspond to 400-520 milliseconds. We conducted thus experiments on the second half of the dataset (the test dataset), and filtered out the recurrences having their length smaller than this minimal value. The obtained results made no difference, meaning that no recurrences

were filtered. This might happen in the case when the detected recurrences are of at least 13 frames. To verify this hypothesis, we computed the length of the audio recurrences resulted after the prior filtering. Table 3.3(b) shows obtained length values.

| Training Datasets | **Separators** | | |
|---|---|---|---|
| | **Min** | **Max** | **Mean** |
| **G1** | 47 | 122 | 92.68 |
| **G2** | 109 | 118 | 114.33 |
| **G3** | 193 | 720 | 390.8 |
| **G4** | 59 | 243 | 139.69 |
| **G5** | 100 | 164 | 125.30 |
| **M1** | 10 | 122 | 51.97 |
| **M2** | 44 | 157 | 83.2 |
| **M3** | 12 | 204 | 29.12 |
| **M4** | 59 | 342 | 113.64 |
| **M5** | 29 | 126 | 66.97 |
| **M6** | 13 | 133 | 47.8 |

(a) (Stats on Separators)

| Training Datasets | **Recurrences** | | |
|---|---|---|---|
| | **Min** | **Max** | **Mean** |
| **G1** | 13 | 343 | 44.08 |
| **G2** | 13 | 500 | 34.54 |
| **G3** | 13 | 684 | 49.51 |
| **G4** | 13 | 237 | 34.32 |
| **G5** | 13 | 563 | 28.87 |
| **M1** | 13 | 226 | 54.80 |
| **M2** | 13 | 171 | 61.37 |
| **M3** | 14 | 43 | 29.36 |
| **M4** | 13 | 884 | 67.10 |
| **M5** | 14 | 282 | 62.70 |
| **M6** | 13 | 792 | 100.13 |

(b) (Stats on Audio Recurrences)

TABLE 3.3: Separators/Detected Recurrences Lengths Statistics (in frames) - Training Dataset.

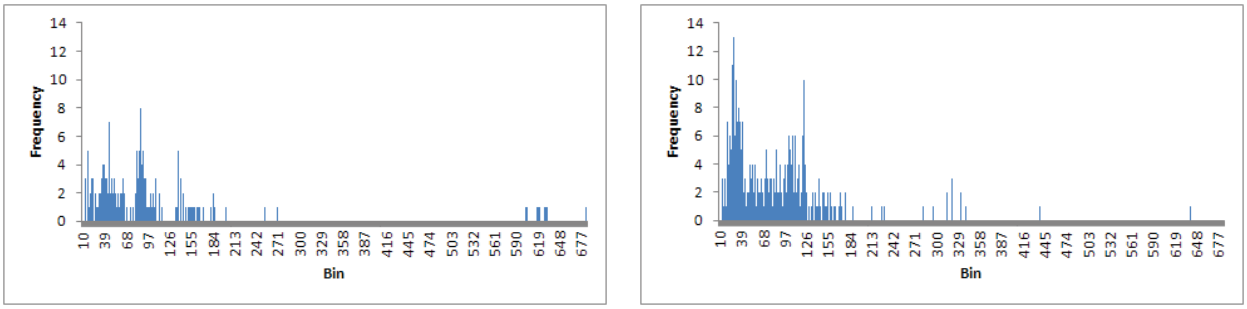
Generally we notice that the shortest length of all the recurrences is of 13 frames. This explains the fact that none of the audio recurrences was filtered previously. This, plus the noticeable differences between the corresponding columns of Table 3.3(a) and Table 3.3(b), implies that there are differences between the lengths of separators and those of recurrences. To get an idea about these differences, we have computed the lengths of only the separators and their correspondent recurrences.

Table 3.4 shows that the absolute values of the differences between separators and recurrences vary from 0 to even 522 frames. The values 0 and 522 are extreme values that represent the minimum and maximum that we found among all the analyzed audio recurrences. However, in mean the differences are between 13.47 and 166 frames which corresponds to 520 milliseconds up to 7 seconds. We took a closer look in order to find out where do these differences come from. One case often encountered is that of a separator with applause or public cheers in the background. As already explained in Section 3.4.3.1, these kind of audio sequences are hard to detect. However, the part where the separator is clean and does not contain applause or public cheers in the background is detected as recurrence, and corresponds thus to a shorter separator. Another case is that of separators containing natural speech. We deal with the same issue as previously. Only the parts of the separators where there is no speaker are detected as recurrences. This is also the case of G3 where a maximum difference of 522 frames is registered. The separator actually corresponds to a sequence where the price to be won is presented. This sequence lasts for 684 frames but it can go up to 720 frames. As most of this sequence is dominated by a speaker presenting the price superposed on a background music, only the small part (162 frames) where the music continues but the speaker finished its speech is detected.

Another situation is when the detected recurrence is longer than the separator. This

| Training Datasets | Separators | | | Recurrences | | | Difference (abs.val.) | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Min** | **Max** | **Mean** | **Min** | **Max** | **Mean** | **Min** | **Max** | **Mean** |
| **G1** | 47 | 112 | 92.35 | 51 | 254 | 106.17 | 0 | 144 | 18.68 |
| **G2** | 109 | 118 | 113.33 | 19 | 92 | 63.66 | 19 | 90 | 49.66 |
| **G3** | 193 | 720 | 390.8 | 20 | 684 | 226.6 | 0 | 522 | 166 |
| **G4** | 59 | 243 | 134.52 | 13 | 158 | 70.22 | 1 | 225 | 64.3 |
| **G5** | 101 | 164 | 125.82 | 13 | 138 | 51.40 | 7 | 130 | 76.70 |
| **M1** | 14 | 122 | 57.3 | 23 | 188 | 89.31 | 4 | 151 | 33.50 |
| **M2** | 44 | 157 | 81.31 | 13 | 171 | 74.63 | 1 | 123 | 39.15 |
| **M3** | 12 | 29 | 16.18 | 15 | 43 | 29.66 | 0 | 27 | 13.47 |
| **M4** | 59 | 342 | 114.04 | 20 | 438 | 134.11 | 0 | 273 | 51.04 |
| **M5** | 29 | 126 | 67.11 | 18 | 282 | 83.58 | 0 | 162 | 22.04 |
| **M6** | 13 | 133 | 48.28 | 36 | 640 | 108.75 | 3 | 507 | 60.96 |

TABLE 3.4: Audio Recurrences-Separators Lengths Statistics - Training Dataset.




(a) (Games) (b) (Mag&News)

TABLE 3.5: Histogram of the lengths of the audio recurrences that are separators - Training Dataset.

happens for example when two neighboring separators in the ground-truth are found as a single recurrence (M4). We explain the existence of neighboring separators by the fact that initially between these two separators there was an advertising. The advertising moments have been removed from the programs we work on, resulting thus in the closeness of 2 separators. Also it might happen to detect a recurrence that contains a separator together with a sequence of a report in a magazine (M5,M6). This generally occurs when the program begins with a summary presenting small parts from all of the reports in the show separated by separators. These segments are also repeated during the show, therefore they will be detected as recurrences. Important to recall is also the fact that the ground truth was created when considering the visual sequences. However the audio jingles are not always superposed with the images (M1,M3). If visually the separator is just a transition effect, the audio content will surely be longer than the visual one, resulting thus in a

difference in length between the audio and visual part of the separator.

As a parenthesis, we call into question the computation of precision and recall measures described in Section 2.4.2 and 3.4.2. All the situations described above justify the choices we made to validate a separator. We refer to the fact that we considered a separator as being correctly identified if it overlaps even with a minimal number of frames with its correspondent in the ground truth. Close parenthesis.

Returning to the length filter, we conclude that it is difficult to find a threshold for the minimal length of separators. This is due to the fact that recurrences that are true separators and those that are not have similar lengths, the length of separators vary from one program to another, and the length of detected separators depends a lot of the environmental conditions of the set where the program is being recorded (public/applause/background music/speech).

Imposing a threshold will filter out false alarms but with them a part of the true separators too. To get a more precise idea about the importance of the impact of such a filter, we computed in Figure 3.5 the histograms for the length of recurrences that are separators corresponding to Table 3.4. These were computed for the games and separately for the magazines and news.

We have also computed some experiments on the test dataset, using arbitrary thresholds from the interval 13-25 fames. The results are shown in Table 3.6.

| Test | **Initial** | | **thresh = 14 fr** | | **thresh = 18 fr** | | **thresh = 20 fr** | | **thresh = 25 fr** | |
|---|---|---|---|---|---|---|---|---|---|---|
| Datasets | **P** | **R** | **P** | **R** | **P** | **R** | **P** | **R** | **P** | **R** |
| **G1** | 0.235 | 1 | 0.236 | 1 | 0.269 | 1 | 0.295 | 1 | 0.373 | 1 |
| **G2** | 0.025 | 1 | 0.026 | 1 | 0.029 | 1 | 0.031 | 1 | 0.037 | 1 |
| **G3** | 0.107 | 0.886 | 0.109 | 0.886 | 0.107 | 0.863 | 0.110 | 0.863 | 0.109 | 0.818 |
| **G4** | 0.118 | 0.954 | 0.115 | 0.923 | 0.136 | 0.876 | 0.139 | 0.830 | 0.157 | 0.723 |
| **G5** | 0.048 | 0.889 | 0.048 | 0.875 | 0.052 | 0.861 | 0.055 | 0.847 | 0.071 | 0.791 |
| **Global** | 0.088 | 0.944 | 0.088 | 0.933 | 0.096 | 0.916 | 0.101 | 0.902 | 0.120 | 0.856 |
| **M1** | 0.417 | 0.875 | 0.415 | 0.875 | 0.444 | 0.854 | 0.455 | 0.833 | 0.512 | 0.75 |
| **M2** | 0.593 | 0.776 | 0.592 | 0.776 | 0.616 | 0.743 | 0.620 | 0.743 | 0.668 | 0.710 |
| **M3** | 0.754 | 0.876 | 0.760 | 0.876 | 0.776 | 0.854 | 0.805 | 0.854 | 0.847 | 0.752 |
| **M4** | 0.273 | 0.991 | 0.278 | 0.990 | 0.307 | 0.991 | 0.320 | 0.982 | 0.391 | 0.973 |
| **M5** | 0.554 | 1 | 0.549 | 0.978 | 0.584 | 0.978 | 0.592 | 0.978 | 0.681 | 0.978 |
| **M6** | 0.380 | 1 | 0.268 | 1 | 0.4 | 1 | 0.421 | 1 | 0.470 | 1 |
| **Global** | 0.421 | 0.891 | 0.423 | 0.889 | 0.455 | 0.873 | 0.468 | 0.869 | 0.532 | 0.829 |

TABLE 3.6: Evaluation of the impact of a length filter.

Indeed a threshold used to filter the recurrences having a smaller length improves the precision but in some cases it also decreases the recall.

Training a threshold might work for some of the TV Programs but we do have to keep

in mind that the detection of the recurrences that are separators is strongly dependent of a lot of factors among which the limitations of the audio approach that considers the noise in the background of the analyzed program, the type of speech (natural/registered) that might appear along the show, etc, but also the fact that producers might play with the length of the separators when the show for example started to deviate from the allocated time schedule. All these could influence the length of separators.

Another possibility would be to try a length filter for the maximal duration of a separator as for the visual case. However, this would be useless as, if we take a look in Tables 3.3(b) and 3.6, the length of recurrences that are separators is smaller than of those that are not.

Our interest for the moment is to keep as many separators as possible which corresponds to the higher recall value (we need as many examples as possible in the training of the decision trees). In this context we decided thus not to apply this filter for the time being. We keep it however in our minds as a possible later filtering.

## 3.5 Conclusion

This chapter studied the possibility of structuring episodes of a large category of TV programs, using the separators detected among the intra- and inter-episode audio recurrences.

First, the focus has been on recurrences. Three types of audio descriptors have been investigated: Perceptual Audio Hashing, MFCC-based and Phoneme-based descriptors. The different types of recurrences these descriptors can identify have been evaluated. Experiments showed that the PAH descriptor is the most appropriate for the detection of the recurrences relevant for structuring.

Second, the focus moved towards separators and the main idea behind our approach, meaning that separators can be detected as recurrences among the episodes of a TV program, has been validated. Indeed the experiments showed a very good detection of separators with very high recall values.

However, a poor precision has been registered. A lot of false alarms representing background music, repeated words/expressions, repeated parts of a report, special sounds from reports (i.e. gunshots, explosions), recorded applause, silences, repeated special editing effects or jingles for dense structuring, have been detected. Consequently, in a third step a filtering process has been examined. The prior filter, that focuses on intra-episode false alarms, proved to be more or less useful depending on the type of program that is analyzed. The temporal filter has not been used as it is not appropriate for the case of audio recurrences which are too many and spread during the entire episodes. A length filter, as in the case of visual recurrences, showed inconclusive results.

Chapter 2 and Chapter 3, studied the possibility of detecting separators among the visual and audio recurrences of a large category of TV programs. The studies were performed for each modality (audio or video) separately. Both approaches showed promising results regarding the detection of separators and the structuring of the analyzed episodes. However, both approaches have their limitations. For instance in the case of audio recurrence detection algorithm the separators containing natural speech cannot be detected. For the case of the video recurrence detection algorithm, the separators that are not identical are not detected. It might also happen that the clustering algorithms, which involve

an optimal setting of large number of parameters, to wrongly perform for certain specific cases. Moreover, for both approaches, besides the separators the algorithms detect some false alarms. For the audio approach these are quite numerous and pollute the obtained results.

In order to try solving all these problems, in Chapter 4, we combine the visual and audio recurrences and propose a classification and selection module based on decision trees. In this way, when using both approaches (visual and audio), one could overcome the limitations of the other and provide thus the necessary for a better structuring. The role of the classification module is to integrate both of the proposed approaches and filter out the false alarms by distinguishing among the detected recurrence the separators.

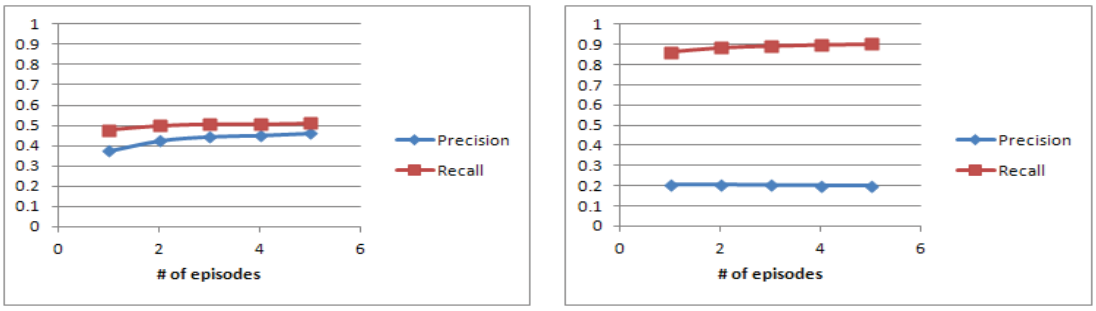
## 3.6 Impact of the number of episodes in the history

Before passing to the next chapter of this thesis, and in the same time to the next stage of the proposed approach, we describe in detail a study previously cited in this thesis. It focuses on the number of episodes on which the recurrence detection algorithms need to be applied in order to recover the separators for a certain episode of a TV program.

The proposed solutions analyze a set of episodes of a recurrent program in order to detect the separators and to structure the episodes.

For the use-case where we have to structure each episode as soon as it arrives, the number of previously broadcasted episodes to put together with the current episode to structure, is an important parameter. The idea is to determine how many previously broadcasted episodes should be considered in order to get the best structure of the current one. We call the number of previously broadcasted episodes *"history"*. Intuitively, if we take a very long history, we may have more noise. Conversely, if we take a shorter history, we may miss separators that do not repeat in the considered episodes.

This experiment evaluates thus the performance of recurrences when the number of previously broadcasted episodes varies. We consider a dataset composed of 2 TV games, 2 magazines and the news program. For each episode of this dataset, we use a history of N previous episodes, with N ranging from 1 to 5. We apply the algorithm on each set of N+1 episodes in order to detect the separators for the $N^{th}$+1 one. Precision and recall of the obtained structuring results (only on the episode to structure) are computed. The obtained results are presented in Figure 3.7.




(a) (Visual) (b) (Audio)

TABLE 3.7: Impact of the number of episodes on the detection of visual/audio recurrences.

Figure 3.7 shows that the number of episodes in the history does not have a very high impact on the detection of visual and audio recurrences. Generally the separators do not change from one episode to another for quite a long time as the content producers create a structure of a program that is respected for all the episodes composing the program. This helps creating a loyal audience that can easily follow the different episodes composing the program. Consequently even with only one episode in the history there are high chances to recover the separators of a new episode. However, it may happen from time to time for producers to integrate a new separator to present some breaking news or related to a new price to win for example. Most often are the cases when separators present prices to win that repeat every two or three shows. This assumes of course a longer history. Other situations when a longer history is useful are due to the limitations of the recurrence detection techniques. These can lead to the miss detection of some recurrences due to their visual/audio content that sometimes is slightly different (i.e. for visual case, the order of the frames in the separator is different or only the first or last part of the separator is played; for the case of audio, separators can sometimes be superposed to applause). Increasing the number of episodes in the history increases also the chances for the detection of such recurrences that might be more similar to sequences from older episodes.

A detailed presentation of the results on each of the tested programs can be found in Annex E. Globally the recall always has a tendency to increase or remains the same. Regarding the precision, in some of the cases it increases in others decreases. The decrease can easily be explained by the fact that a longer history can lead to the detection of more false alarms. This happens especially for the case of audio recurrence detection as the sounds in a show repeats quite often without being separators.

The increases of the precision are on one hand due to the increase of the recall. The detection of more separators without an increase of the detection of false alarms results in the increase of the precision. On the other hand, for the visual algorithm, there are cases when more episodes in the history change the way in which the recurrences are clustered together. An example is in Figure 3.7.

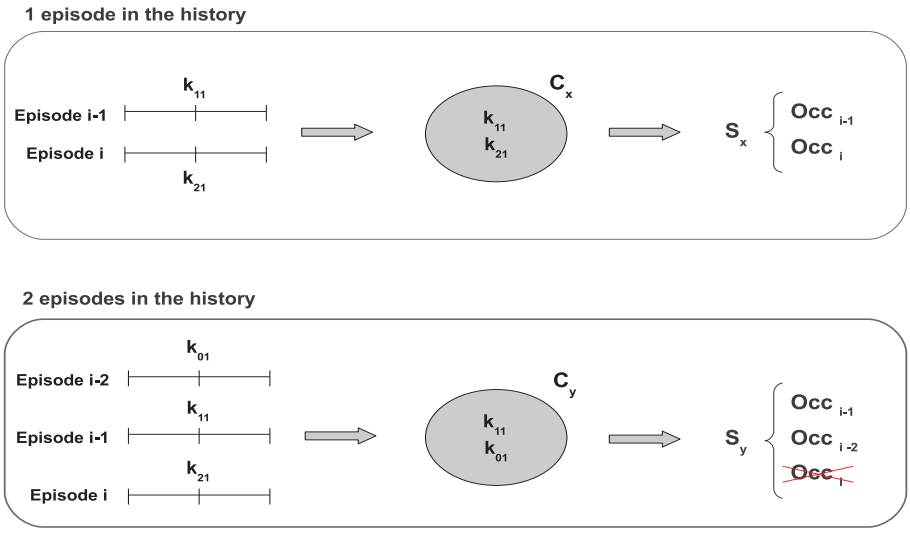



FIGURE 3.7: History impact on the detection of recurrences.

Having two similar sequences of 1 shot each, from 2 episodes on which we apply the recurrence detection algorithm, their corresponding keyframes will be clustered together ($C_x$) and thus a recurrent segment containing the two occurrences will be formed. In a second case, where 2 episodes are considered in the history, the keyframes corresponding to the sequences of the first two episodes are clustered together resulting the cluster $C_y$. The keyframe $k_{21}$ of the third episode is no longer inserted in the cluster $C_y$ as it does not respect the clustering conditions (see Section 2.1). This leads to an increase of the precision, if the recurrent sequence is a false alarm, but in the same time could lead to a decrease of the recall, if the recurrent sequence is a true separator.

However, Figure 3.7 shows that, when searching for the length of the history, the best trade-off seems to be around a history of 3 episodes. This is very good for real-world applications as only three previously broadcasted episodes are needed to structure the current one.

# Chapter 4

# Recurrence classification

**Summary**



THIS CHAPTER continues our study of audio and video recurrences detected between or within different parts of a TV program. Among these there are possible separators that delimit the different parts in a TV program. We recall that our interest is to find these separators and use them to recover the structure of a program.

When analyzing the separators that we are so interested to detect, we noticed that besides the fact that they repeat inside or between different episodes of a TV program, other properties come forth. These properties could help us to detect and filter among the audio and video recurrences those that are separators.

To make an idea about these properties and the attributes that describe them, we conceived a manual modeling of the structure of the programs by analyzing a set of different types of programs, like TV games, magazines and news. One could say that analyzing the programs induces prior knowledge in the proposed method. This is not completely true. Indeed when analyzing the programs we make an idea about what properties might be used. But when implementing an algorithm we will not impose these properties, but these will be automatically created/detected based on some attributes that were found interesting during the analysis.

## 4.1 Manual modeling of the structure of programs

***TV Games Programs Structure***

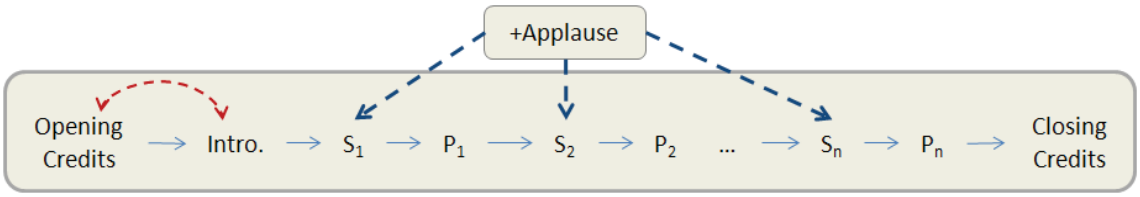



FIGURE 4.1: Manually modeled structure of TV Games.

To compute the manual structure of TV games we analyzed 10 different French TV games, from 3 French generalist TV channels. Generally the structure of TV games is quite stable and repetitive. It is composed of an initial introduction part where the anchor presents the competitors followed by the stages of the games ($P_i$) that are most often separated by separators ($S_i$) (see Figure 4.1). Important to notice is that generally the anchor is the one that closes the parts of the show and that announces the passage from one stage of the game to another. The transitions are also marked by the cheers of the public, music or applause. All the activity happens in the set and the number of persons that are present in the set and participate to discussions can vary from 2 (the anchor and the competitor that arrived in the final) to even 10 or more (i.e. the first part of the game when generally there are more competitors compared to the final part where only the competitors qualified in the finals are present).

***Magazine Programs Structure***

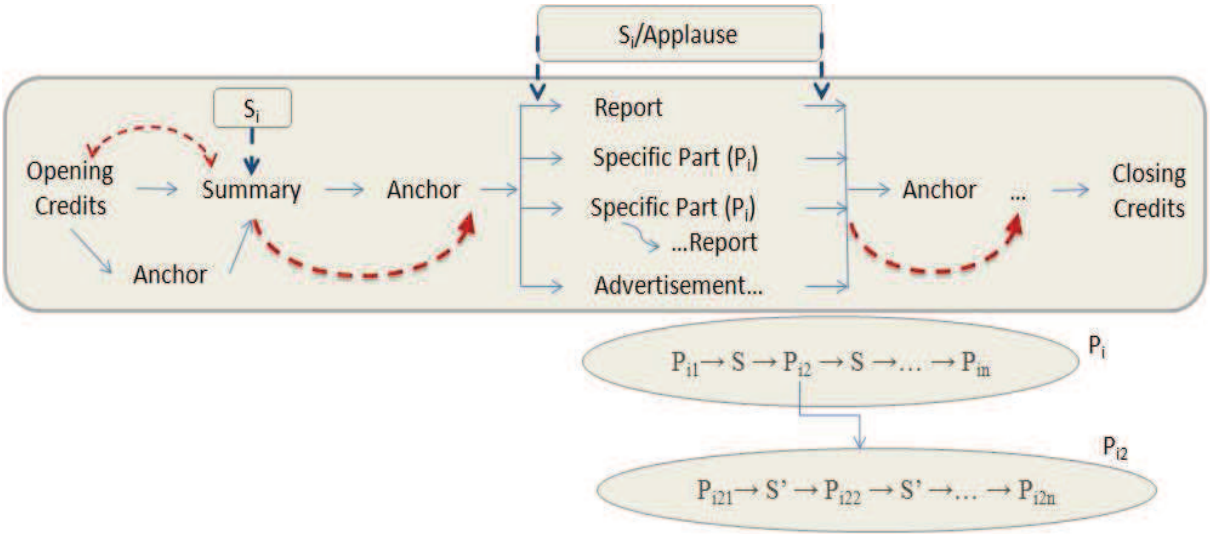



FIGURE 4.2: Manually modeled structure of TV Magazines.

Regarding the magazines we have studied 10 different magazines from French TV

channels. In this case the structure can be more complicated and a simplified scheme is presented in Figure 4.2. These programs generally start with a summary where the reports that will be presented during the program are briefly reviewed. These short reviews can sometimes be separated by very short separators ($S_i$). They continue then, with a suite of reports and sequences of the set where the anchors introduce the next topic. These can be separated by separators ($S_j$) but also by the cheers and applause from the public present in the set. These can however completely miss as often the magazines do not have public on the set. The reports can sometimes be replaced by specific parts of the magazine ($P_i$) (i.e. the horoscope, a fashion section or a culinary part) that generally are stable from one episode to another of the show. The reports/specific parts themselves can sometimes be subdivided in smaller parts also separated by other separators. In our case we are most concerned by the structure of the entire program than that of the parts composing the program as our scope is to segment a program in its main parts.

In this case, the action changes from the set to the outfield where the report takes place. In the set however only the anchors appear and sometimes their guests, invited for an interview.

***News Programs Structure***

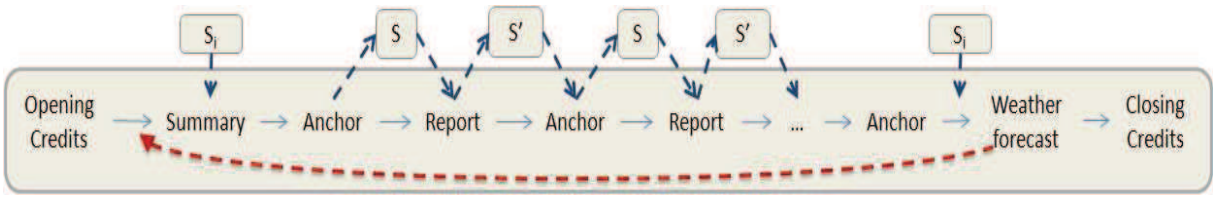



FIGURE 4.3: Manually modeled structure of TV News.

For the news programs the format is very similar to the one presented for the magazines. It is composed of reports and scenes with the anchor that introduces the next topic, both delimited by separators. As for the magazines, only the anchors and sometimes their guests, invited for an interview, are present in the set. The scenes change between the set and the places presented in the reports.

From all these observations, some important references stand out, as the presence of the anchor, of applause, the transition from one scene to another. The idea is to make use of these references and use them to create a set of related attributes. Based on these attributes, and using a learning algorithm a model is then built. The model is afterwards used to classify the audio and visual recurrences as separators or non-separators. We found appropriate for our case the decision trees classification method. The proposed solution is described in detail in the next sections.

## 4.2 Decision Trees Classification

The recurrences that remained after the filtering at 2.4.4 and 3.4.5, are passed through a classification and selection module based on decision trees. We have chosen to work with decision trees as these are appropriate for our goal and the type of data we use as explained hereafter. Moreover, the results can be easily analyzed and understood, the number of tests is not too high (as there are not many parameters to adjust) and in the same time it is a powerful tool for classification.

The idea is to build a model through a sequence of questions made on a set of several input variables. Each next question depends on the previous answer (see Figure 4.4). Once the model is build, it can be used afterwards to predict the value of a target variable based on the input variables. In our case, we will consider all the variables as binary. We define the target variable as the category to which the recurrence belongs to. A recurrence is a separator if the assigned category is 1 and it is a false alarm if the assigned category is 0.

We call the input variables attributes. These are issued from a set of techniques as applause detection, scene segmentation, speaker diarization and clustering and finally face detection and clustering. The modules and their corresponding attributes are further described.

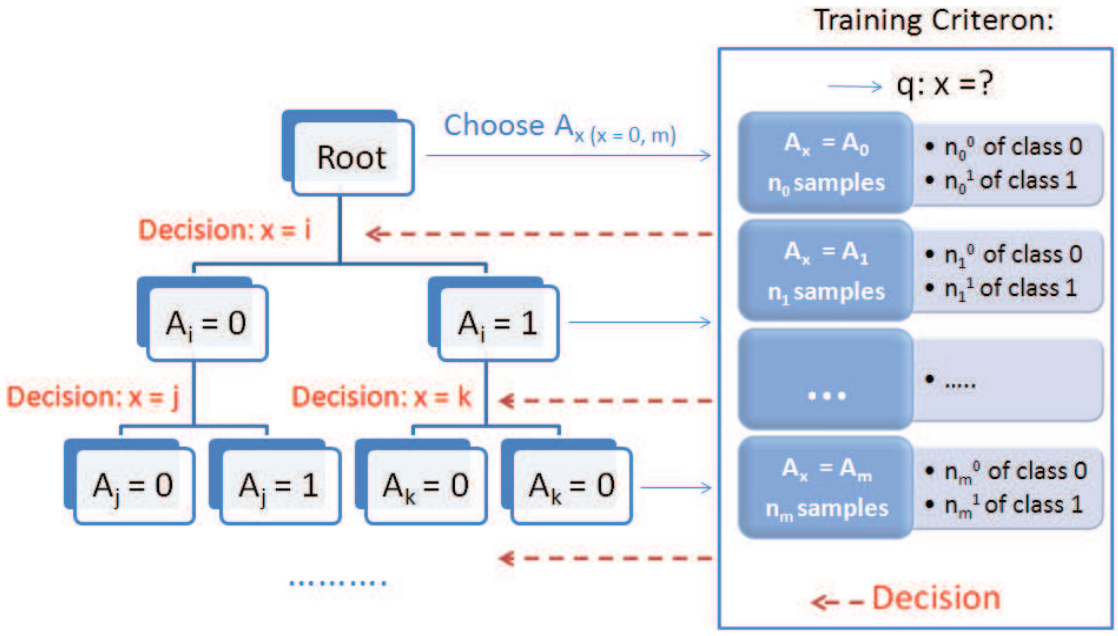



FIGURE 4.4: Decision Trees Algorithm.

### 4.2.1 Attributes description

The first attribute, denoted A0, is issued from the intersection of the audio and visual recurrences. It values 1, if for an analyzed visual recurrence, there is a corresponding audio recurrence and vice versa.[1]

#### 4.2.1.1 Applause segmentation

The method used for the detection of applause, is part of an audio classification approach that allows the segmentation of an audio visual content into five classes (speech, music, applause, silence, other). It is based on the computation of low-level features that are used to discriminate between different types of sound. It is similar to the one described in [CLZC03]. The proposed solution uses four features: the energy modulation at 4 Hz ($f_0$), the mel-frequency cepstral coefficients ($f_1$), the spectral flux ($f_2$) and the chroma ($f_3$) computed each 10ms on an audio frame of 30ms. For the case of $f_0$ and $f_2$, the mean and variance computed on a sliding window of 1s were considered.

The classifier is based on the Bayes theorem and the features are modeled using Gaussian Mixture Models (GMMs), as described in [SS97].

For a given audio frame, the probability for a class to correspond to a set of computed features ($f_0$, $f_1$, $f_2$, $f_3$) is given by the Bayes formula:

$$p(C_j/f_0, f_1, f_2, f_3) = \frac{p(C_j)p(f_0, f_1, f_2, f_3/C_j)}{p(f_0, f_1, f_2, f_3)} \tag{4.1}$$

Consequently, for a given frame, the most likely class is the one that corresponds to the maximum $p(C_j/f_0, f_1, f_2, f_3)$, meaning the index J so that:

$$J = arg\max_j p(C_j/f_0, f_1, f_2, f_3) = arg\max_j p(C_j)p(f_0, f_1, f_2, f_3/C_j) \tag{4.2}$$

The size of the four features is 28 (2+12+2+12). To ease the training phase, the classifier used is the naive Bayes classifier that assumes that the the features are independent. The training is thus performed per class and per feature. The fact that the features are independent, resumes the equation 4.2 to:

$$J = arg\max_j p(C_j)p(f_0/C_j)p(f_1/C_j)p(f_2/C_j)p(f_3/C_j) \tag{4.3}$$

where $p(f_i/C_i)$is the probability of the $i^{th}$ feature to have the value $f_i$ when the signal belongs to class $C_j$ and $p(C_j)$ is the prior probability of the class $C_j$.

The probability $p(f_i/C_i)$ is modeled by a GMM. In this implementation, 64 gaussians have been used.

To undermine the variability of the extracted features, the results need to be smoothen over a set of consecutive audio frames. Therefore the likelihood for a set of $N$ consecutive frames will be defined as:

[1] In order to avoid any confusions, we recall that a description of the vocabulary, and implicitly a definition for the *recurrence*, is given in the *Introduction* Chapter, Section *Description of the vocabulary employed in this thesis*.

$$P(C_j/f_0, f_1, f_2, f_3, k) = \prod_{n=kN}^{kN+N-1} P(C_j/f_0, f_1, f_2, f_3, n) \tag{4.4}$$

and the decision for the most likely class will be computed by:

$$C_j/J = arg\,\underset{j}{max}\, P(C_j, f_0, f_1, f_2, f_3, k) \tag{4.5}$$

Three attributes are issued from this module:

- A1 = 1 if applause exist in the analyzed recurrence.
- A2 = 1 if applause exist in the right neighboring (5 seconds) of the analyzed recurrence.
- A3 = 1 if applause exist in the left neighboring (5 seconds) of the analyzed recurrence.

#### 4.2.1.2 Scene segmentation

The algorithm used for the segmentation of videos into scenes is a hierarchical classification algorithm that emphasize the similarities of shots that are temporally close. These ones generally belong to a same scene. The idea is to build a similarity tree where each leaf represents a shot and each node that groups two leafs or several nodes in a superior level, represents a possible scene. The similarity criteria used for the grouping of leafs/nodes is the calorimetrical distribution.

The first step is thus to segment the video into shots. For each shot, a descriptor $H_i$ is computed, representing a three dimensional histogram of the calorimetrical distribution of an image in the RGB space. The algorithm proceeds then with the construction of the similarity tree. To do so, the distance between shots is defined as the Chi-Square distance:

$$D^{ij} = \begin{cases} \sum_{k=0}^{511} \frac{(H_i^k - H_j^k)^2}{H_i^k + H_j^k} & if \quad \Delta T^{ij} \leq \Delta T^{MAX} \\ D^{MAX} & elsewise \end{cases} \tag{4.6}$$

and the distance between scenes as:

$$DS^{ij} = \underset{k \in \Gamma^i, l \in \Gamma^j}{min}\, D^{kl} \tag{4.7}$$

where $\Delta T^{ij}$ represents the temporal distance between the middle of the shot $i$ and the middle of shot $j$ and $\Gamma^i$ represents the group of shots that belong to scene $i$. In order to assure the temporal proximity of shots in a scene, a maximal temporal distance between shots $\Delta T^{MAX}$ is defined. To create the similarity tree, at first, a list of scenes is computed, each containing one shot of the video. The distances between all the scenes are computed and those having the minimal distance are merged into a new scene. The latter is introduced in the list while the two scenes that were merged are taken out of the list. This process is reiterated until there will be only one scene in the list.

Once the similarity tree built, the next step is to find the most suitable time instances where a scene cut might appear, according to the following constraints: each scene must have a duration between $D^{MAX}$ and $D^{MAX}/2$, where $D^{MAX}$ is the maximum duration of a

scene; having N scenes, the duration of each scene must be found between $D^{MEAN} - D^{VAR}$ and $D^{MEAN} + D^{VAR}$, where $D^{MEAN} = D^{VAR}/N$.

The first constraint represents the validity interval when choosing the cutting hypothesis. The tree is scanned form up to down. For each node the minimum and maximum time borders of each shot situated under the considered node, are extracted. If one of the borders is situated in the considered validity interval, this cutting hypothesis is retained along with the score of the considered node. The cutting hypothesis are arranged in descending order, so that the first one has the higher score. Each hypothesis made on a scene implies a new validity interval for the end of the succeeding scene and thus new hypothesis for the end of this later one. In this manner several lists of hypothesis are computed. Each list is validated only if its last scene verifies the defined constraints. If not, the list is removed form the set of hypothesis lists. For the remained lists, a mean score is computed (as the mean of the scores of each scene cut in the list). The list with the higher mean score will correspond to the best sequence of scenes cuts.

The attribute issued from this module is:

- A4 = 1 if there is a scene cut in the close neighboring (10 seconds) of the analyzed recurrence

#### 4.2.1.3 Face detection and clustering

The framework we used for the face detection and tracking is the one presented in [FMG07]. It is based on the convolutional face finder (CFF) described in [GD04]. The CFF provides a convolutional neural network that allows the detection of semi frontal faces of variable size and appearance, rotated up to ±20 degrees in image plane and turned up to ±60 degrees. It consists of a pipeline of convolution and subsampling modules that perform automatic features extraction and classification. In order to adapt the algorithm to video indexing, in [FMG07] one of the stages of the algorithm is used for tracking the detected faces in consecutive frames. First, one image among $N$ is computed by coarse detection on the whole image. Then, a fine detection is applied on the following images at each candidates' face location given by the previous image. A micro-clustering technique is then used to cluster together the most similar faces, that belong to the same person.

Four attributes are issued from this module:

- A5 = 1 if there is a face in the analyzed recurrence
- A6 = 1 if the face in the right neighboring (5 seconds) of the analyzed recurrence is the same as the one inside the analyzed recurrence
- A7 = 1 if the face in the left neighboring (5 seconds) of the analyzed recurrence is the same as the one inside the analyzed recurrence
- A8 = 1 if the same face is in the right and left neighboring of the analyzed recurrence

#### 4.2.1.4 Speaker diarization and clustering

The algorithm used for the speaker diarization and clustering is similar to the one described in [CBG06]. It proposes to index people independently by the audio information. The audio signal is first segmented into speech/non speech sequences. Each speech sequence is then

segmented into speaker turn. The sequences corresponding to the same person are then clustered together using a *Bayesian Information Criterion* and *Cross-likelihood Ratio*. At the end of the process each audio sequence is associated to an audio label automatically determined.

Four attributes are issued from this module:

- A9 = 1 if there is speech in the analyzed recurrence
- A10 = 1 if the speaker in the right neighboring (5 seconds) of the analyzed recurrence is the same as the one inside the analyzed recurrence
- A11 = 1 if the speaker in the left neighboring (5 seconds) of the analyzed recurrence is the same as the one inside the analyzed recurrence
- A12 = 1 if the same speaker is in the right and left neighboring of the analyzed recurrence

### 4.2.2 Training phase

In order to build the decision tree, a set of $n$ samples (recurrences) from the training data is used. Each sample is described by the 13 binary attributes previously presented and by a target category that confirms if the recurrence is a separator or not. The tree is built by asking questions on the attributes.

#### 4.2.2.1 Single/Complex attribute questioning

When questioning the attributes, two approaches have been considered:
In a **first approach** one attribute at time is selected for questioning. This attribute is chosen according to a predefined criterion. The tests on this attribute are used to split the current training data set into subsets. The algorithm is recursive and progressively splits the training data set into smaller subsets as long as the subset is not "pure"(all the samples belong to the same category) or there is at least another attribute to test. The split can be stopped beforehand by imposing a constraint on the parameters of the chosen criterion. In this case we declare the node a leaf and assign to it a category. The assigned category depends on the predefined criteria that has been used.

Important to note is the fact that when building the decision trees, it might happen that none of the attributes employed in the training phase, used alone, can not bring any improvement (see an example in Annex D). In this case, no model can be built and the assigned category will be represented by the one with a majority of training samples.

However, in order to overcome this limitation and to extend further the use of decision trees we propose as a **second approach** the use of combinations of attributes when training the models. The idea is to offer the possibility to question several attributes in the same time instead of a single one. Two cases have been considered. In a first one, the number of attributes to be questioned is given by a parameter that we named "*complexity (x)*". In a second one all the possible combinations of the entire set of attributes are tested.

To do so, in a first step, all the possible combinations of attributes are generated regarding one of the proposed possibilities. The first case corresponds to the generation of

combinations of $x$ (complexity) attributes from the set of $n$ ($n = 13$ in our case) attributes that we proposed. For the second case the same algorithm that generates the combinations of $x$ attributes from the set of $n$ is used, but this time the parameter $x$ is not given by the user but it takes all the values from 1 to $n$. All these possible combinations will have to be then tested according to the predefined criteria.

In order to make it more clear we illustrate an example where we dispose of 4 attributes (A0,A1,A2,A3).

When using the first approach, one attribute is questioned at a time. 4 steps, corresponding to each of the 4 attributes will thus have to be done in order to find the best suited attribute.

For the second approach:

- First case: $n = 4$ attributes, complexity $x = 2$. All the possible combinations of attributes that can be generated are: $A_0A_1$, $A_0A_2$, $A_0A_3$, $A_1A_2$, $A_1A_3$, $A_2A_3$.

  Consequently, we will not have only 4 attributes to test, as in the previous approach, but 6, which correspond to $C_n^x$.

- Second case: n = 4 attributes, complexity x = (1 to n). All the possible combinations of attributes that can be generated are: $A_0$, $A_0A_1$, $A_0A_2$, $A_0A_3$, $A_0A_1A_2$, $A_0A_1A_3$, $A_0A_2A_3$, $A_0A_1A_2A_3$, $A_1$, $A_1A_2$, $A_1A_3$, $A_1A_2A_3$, $A_2$, $A_2A_3$, $A_3$.

  In this case a much higher number of combinations have to be tested. This corresponds in our case to 15 and can be computed by $\sum_{x=1}^{n} C_n^x$

Once the possible combinations of attributes have been generated, in a second step, the values that these attributes can take have to be considered. Consequently, for each of the attributes combinations, all the possible combinations of attributes values have to be computed and then tested when building the decision trees.

This would correspond for example, in the case of a combination of $x = 2$ attributes (A0 and A1) that can take each $k = 2$ values (0 and 1) to the sequences 00, 01,10, 11.

The entire algorithm for building the decision tree could be summarized/formalized as in Algorithm 2:

#### 4.2.2.2 Training Criteria

We used three different criteria for the choice of the attribute/combination of attributes to be tested in the current node. For simplicity when describing the criteria we will use *best attribute* but we refer to both cases, meaning best attribute or best combination of attributes.

***Purity criterion***

This criterion considers the number of samples classified in each leaf. The idea is to choose the attribute that makes the immediate descendant nodes (the leafs) as pure as possible, meaning as many as possible of the samples in the node to belong to the same class. To do so, the purity of a leaf is computed as the number of samples belonging to the most representative class, while the purity of an attribute is the sum of the purities of its leafs. Let us consider $A_i$, i = 0,...12, one of the defined attributes and j = 0,1 the values that

**input** : The number of attributes $n$, the complexity $x$, the number of attributes values $k$, and the file $DataInFile$ containing the samples for training
**output**: The possibles sequences of values of an attribute combination

$ACFile \leftarrow GenerateAttributeCombinations(n, x)$ `// for` $2^{nd}$ `case` $x = 1 \rightarrow n$
$DataIn \leftarrow ReadFile(DataInFile)$ `// read the samples for training`
**begin**
  `// Build the tree node by node`
  **for** $line \leftarrow ReadLineByLineFromFile(ACFile)$ **do**
    *create a new vector* array *where to put the possible combinations of attributes values of the line*;
    array $\leftarrow GenerateAttributeValuesCombinations(line)$;
    $cost \leftarrow TestAttribute$(array); `// compute cost using a predefined criteria`
    $best \leftarrow DecideIfBestAttribute(cost, bestAttribute)$;
    **if** $best = 1$ **then**
      $bestAttribute \leftarrow line$;
    **end**
  **end**
  Apply split in current node and pass to next node;
**end**

**Algorithm 2:** Build decision trees

attribute $A_i$ can take. Suppose that $n_{cj}$ is the number of samples of category c, c = 0,1, the purity function computed for each leaf is:

$$p_j = \max_c n_{cj} \tag{4.8}$$

while the purity for attribute $A_i$ is:

$$P_{A_i} = \sum_j p_j \tag{4.9}$$

The attribute with the greatest purity $P_{A_i}$ will be chosen.

In this case the split is stopped if the purity can no more be improved or if the node is pure.

***Entropy criterion***

This second criteria follow the same idea as the previous one, meaning to choose the attribute that makes the descendant nodes as pure as possible. For the previous criterion, when computing the purity the number of samples of the most representative category in each leaf is considered. Considering the fact that the size of the categories can be very different, this could bias the results. Consequently, we propose to try a new criterion based on the entropy.

In this case, the purity is treated as an entropy impurity function and the chosen attribute will be the one that provides the smaller impurity. This could be interpreted

as the attribute that has the greatest cross correlation with the distribution of training samples into the two categories.

Considering the same notation as for the previous case, the purity function computed for each leaf is:

$$p_j = -\sum_c \frac{n_{cj}}{n_j} * \log \frac{n_{cj}}{n_j}, \quad where \quad n_j = \sum_c n_{cj} \tag{4.10}$$

and the purity for attribute $A_i$ is:

$$P_{A_i} = \sum_j \frac{n_j}{n} * p_j \tag{4.11}$$

In this case the split is stopped if the purity can no more be improved or if the improvement that can be added is below a certain threshold. This threshold is used in order to avoid the "over-training" of the tree.

***Weighted Class (Error Minimization)***

This last criterion takes in consideration the fact that the number of samples assigned to each of the two categories is very different. Moreover, the first criterion tried to obtain a good performance by maximizing the number of well classified samples (recurrences). In this case, we focus on obtaining a good performance from the point of view of minimizing the number of errors.

We deal with two kinds of errors:

- the FP (false positive), which correspond to the recurrences that were wrongly classified as separators
- the FN (false negative), which correspond to the recurrences that were wrongly classified as non-separators

The cost of each type of error is weighted by a weight parameter $\lambda_i$. The idea is to choose the category with the smaller cost. For a better understanding, we briefly describe the algorithm hereafter. Considering the same notation as for the previous cases, the cost in each leaf is computed as:

- if the assigned category would be c = 1, than: $C_{1j} = \lambda_1 * n_{0j}$
- if the assigned category would be c = 0, than: $C_{0j} = \lambda_0 * n_{1j}$

The cost for the leaf j in this case is:

$$C_j = \min_c C_{cj} \tag{4.12}$$

The cost of attribute $A_i$ is then computed as:

$$C_{A_i} = \sum_j C_j \tag{4.13}$$

At each step, the cost of all attributes is computed and the attribute that minimizes the cost of the errors is chosen. The split is then done according to the tests made on this attribute.

The split is stopped when the cost can no longer be reduced.

### 4.2.3 Classification phase

According to the trained model and the corresponding attributes values, each recurrence in the test dataset will be classified in one of the two categories (separator/non-separator).

### 4.2.4 Recurrence Classification Results

#### 4.2.4.1 Datasets

We used the same experimental dataset described in Annex A.

In this case, the dataset has been divided in 2 groups: one containing the games and the other one containing the magazines and news programs. The objective was to group programs depending on their properties, and we have noticed that magazines and news programs have almost the same structure (set of reports and anchor person scenes), different from that of the games (see Section 4.1for more details).

We recall in Table 4.1 the description of the dataset.

| Program Type | Program Name | Program Id | Nb. of episodes | Nb. of separators/episode |
|---|---|---|---|---|
| Games | Les Z'amours | $G_1$ | 28 | 6 |
| | Mot de passe | $G_2$ | 5 | 4 |
| | Motus | $G_3$ | 21 | 4 |
| | Tlmvpsp | $G_4$ | 26 | 5 |
| | Les 12 coups de midi | $G_5$ | 20 | 8 |
| Magazines News | Comment ca va bien | $M_1$ | 24 | 8/14 |
| | 10h Le Mag | $M_2$ | 5 | 20/23 |
| | Cine, Series et cie | $M_3$ | 7 | 15/17 |
| | 50mn Inside | $M_4$ | 14 | 12/18 |
| | 7 A 8 | $M_5$ | 10 | 5/7 |
| | 19h45 | $M_6$ | 9 | 20/27 |

TABLE 4.1: Dataset description.

#### 4.2.4.2 Evaluation measures

The starting point of our study is the detection of recurrences. The main assumption of our approach is that separators are repeated and can be detected as recurrences. The idea is to find as many separators as possible among these recurrences, that would allow us afterwards to structure the different episodes of various TV programs.

There are two possible evaluation methods:

***Evaluation relative to Separators' Ground-Truth***

The first one, that has already been used, is to evaluate the detection of separators relative to the separators ground-truth. To be more clear, we illustrated this first case in Figure 4.5.

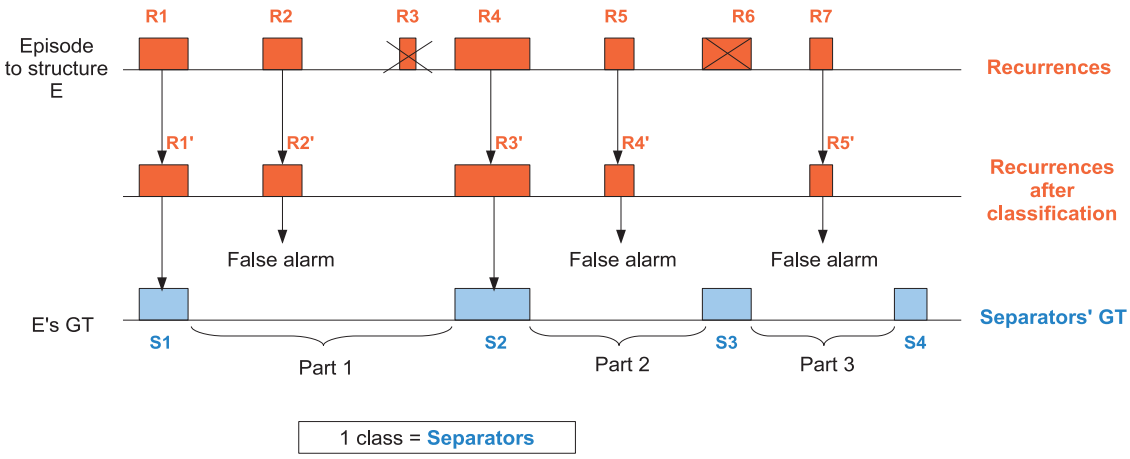



FIGURE 4.5: Separators' Ground-Truth.

After computing the audio and video recurrences (first line in Figure 4.5), the classifier selects the recurrences that are true separators. These are represented on the second line in the figure. For evaluation, these are then projected on a separators ground-truth that contains all the separators that were manually annotated in the analyzed episode (third line in the figure - Separators' GT). Among the recurrences there are the separators (i.e. R1' corresponds to S1, R3' to S2), but not all the recurrences are separators (i.e. R2', R4', R5' are not true separators and consequently they will be treated as false alarms). The evaluation in this case is thus computed as precision and recall measures relative to the single class of separators.

For a global evaluation the F1 score is used.

For the specific case in Figure 4.5 these are:

$$P = \frac{2(\rightarrow R1', R3')}{5(\rightarrow R1', R2', R3', R4', R5')}$$

$$R = \frac{2(\rightarrow R1', R3')}{4(\rightarrow S1, S2, S3, S4)}$$

$$F1 = \frac{2 * P * R}{P + R}$$

***Evaluation relative to Recurrences' Ground-Truth***

The second type of evaluation is relative to what we called *recurrences ground-truth*. For this case, representative is Figure 4.6. We consider two classes: separator (class 1) and non-separator (class 0). After computing the audio and video recurrences (first line in Figure 4.6), the classifier classifies each of the recurrences in one of the two classes (second line in the figure). For each of the recurrences, it is already known the class to which it really belongs to. This information is provided by the ground-truth manually created for the separators of the analyzed episode, and will represent the recurrences' ground truth (line three in figure). The evaluation is thus computed relative to existing recurrences and assumes the presence of the two classes.

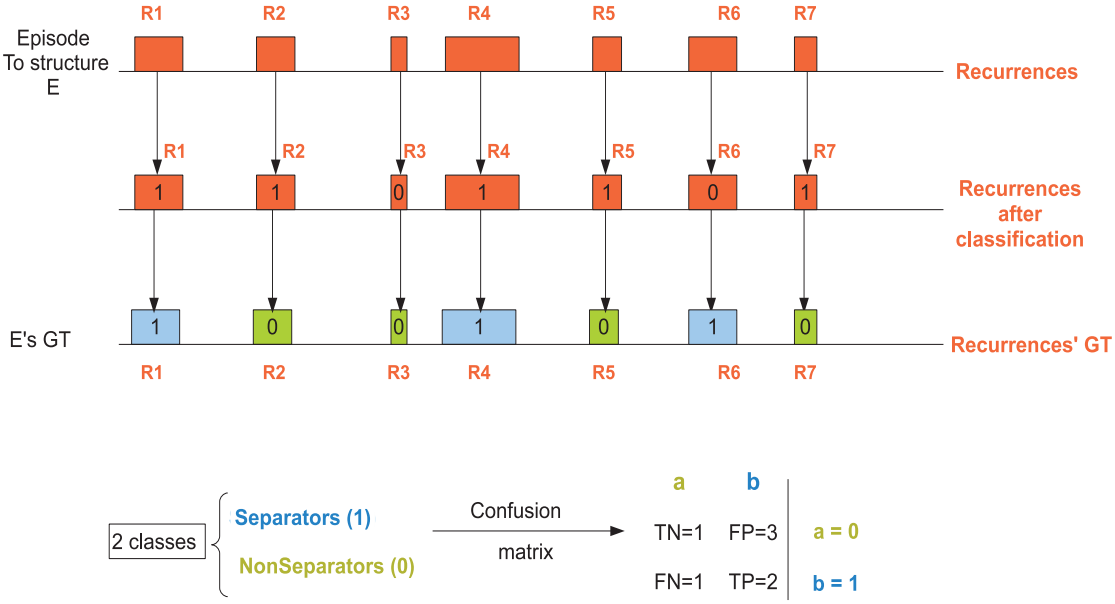



FIGURE 4.6: Recurrences' Ground-Truth.

For the moment, there is no study that could relate the type of errors to the end-user satisfaction. Consequently, we cannot decide whether when browsing inside a TV show, it is more annoying to have some missing separators (related to false negative errors) or to have to skip some recurrences that are not separators (related to false positives errors). Therefore, we use for evaluation the precision and recall for each of the two categories (separator/non-separator). We also compute the classification accuracy as a global measure that computes the proportion of true results in the entire test dataset.

For the specific case in Figure 4.6 we have :

$$\begin{cases} TN = 1(\rightarrow R3); \\ TP = 2(\rightarrow R1, R4); \\ FN = 1(\rightarrow R6); \\ FP = 3(\rightarrow R2, R5, R7) \end{cases}$$

$$class0 : P = \frac{1(\rightarrow TN)}{2(\rightarrow TN + FN)}, R = \frac{1(\rightarrow TN)}{4(\rightarrow TN + FP)}$$

$$class1 : P = \frac{2(\rightarrow TP)}{5(\rightarrow TP + FP)}, R = \frac{2(\rightarrow TP)}{3(\rightarrow TP + FP)}$$

$$Accuracy : A = \frac{3(\rightarrow TP + TN)}{7(\rightarrow TP + TN + FN + TN)}$$

When trying to evaluate and compare the performances of different classifiers, we will use the second evaluation approach as most important is to test the classifiers ability to classify the recurrences into the correct class. In a final step we will also present the results relative to the separators ground-truth as these represent the global evaluation of the entire solution that includes the detection of recurrences and their classification.

#### 4.2.4.3 Analysis of recurrences in the dataset

After applying our recurrence detection technique on each episode following the methods described in Chapters 2 and 3, we get a set of 14,283 recurrences among which 14,130 are audio recurrences and 1,002 are visual recurrences. We recall here that in order to compute recurrences of an episode, we have considered the 3 previously broadcasted episodes and we applied the recurrence detection algorithm. Only the recurrences obtained for the analyzed episode are considered into the result set of recurrences. This way, we can detect intra-episode recurrences as well as inter-episode recurrences.

For all our experiments, this dataset has been split into 2 datasets: a training dataset and a test dataset. The first one is used for training the decision trees while the second one is used for evaluation of the previously learned trees (test). Ideally would have been to use 3 datasets: one for learning, one for validation and one for testing. But considering that we are in an exploratory phase on this kind of data and that we do not dispose of a very big dataset, we skipped the validation phase and used directly the testing one.

For the audio approach, the recurrences obtained after the prior filter, presented in Section 2.2.1 have been used. For the visual one, we have considered the recurrences obtained after the prior filter and also, for the programs with temporal stability, the temporal filter.

We have first evaluated the proportion of recurrences that are separators in the training dataset. This is equivalent to consider a naive classifier that assigns all the recurrences to the class “separators”. Tables 4.2 and 4.3 summarizes the obtained precision, recall and F-Measure for both program categories "games" and "magazines and news", and also for audio and visual recurrences.

Table 4.2 shows that for the TV games, among the audio recurrences, only almost 6% are separators while the rest of 94% should be filtered out as they belong to the category “non-separators”. These correspond to specific audio sequences that can be found in TV games. An example is the sound of pushing a certain button when a competitor wants to answer a question, or the audio sequence that announces if the answer was true or false.

On the contrary, for the visual recurrences, most of the recurrences (82%) are separators and only 18% are “non-separators”.

For the case of TV magazines and news programs, we noticed the same phenomenon even though the percentage of separators among the audio recurrences is higher (41%) than for the correspondent in TV games.

| Games | Audio rec. | | | Visual rec. | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| | 0.0612 | 1 | 0.1154 | 0.8248 | 1 | 0.9040 |

TABLE 4.2: Analysis of games recurrences in the training dataset.

| Mag&News | Audio rec. | | | Visual rec. | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| | 0.4072 | 1 | 0.5787 | 0.7442 | 1 | 0.8533 |

TABLE 4.3: Analysis of magazines and news recurrences in the training dataset.

These differences between games and magazines+news (see also Section 4.1) and their correspondent audio and visual recurrences, encouraged us to continue the experiments considering separately games and magazines+news, and also training and performing classification separately for visual recurrences and audio ones.

#### 4.2.4.4 Recurrence Classification Results - Single attribute training

The objective is to evaluate the ability of a classifier to distinguish if a recurrence in the test dataset is a separator or not.

As already stated in the previous section, we have trained a classifier per modality that is, a classifier is trained for audio (resp. visual) recurrences of the training dataset and used to classify audio (resp. visual) recurrences of the test dataset. We also separated the TV games from magazines+news.

***Classification Results - Purity Criterion***

Tables 4.4 and 4.5 show the results obtained on the test dataset, when applying the decision trees learned with the purity criterion.

| Purity Criterion | Games | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **Audio** | 0.9960 | 0.9701 | 0.9829 | 0.8848 | 0.5015 | 0.6401 | 0.9673 |
| **Visual** | 0.2581 | 0.5714 | 0.3555 | 0.8832 | 0.9667 | 0.9231 | 0.8625 |

TABLE 4.4: Evaluation of the classification results when using the purity criterion for training - Games.

| Purity Criterion | Mag&News | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **Audio** | 0.8656 | 0.8054 | 0.8344 | 0.7139 | 0.6159 | 0.6613 | 0.7776 |
| **Visual** | 0.2683 | 0.7333 | 0.3928 | 0.8750 | 0.9813 | 0.9251 | 0.8667 |

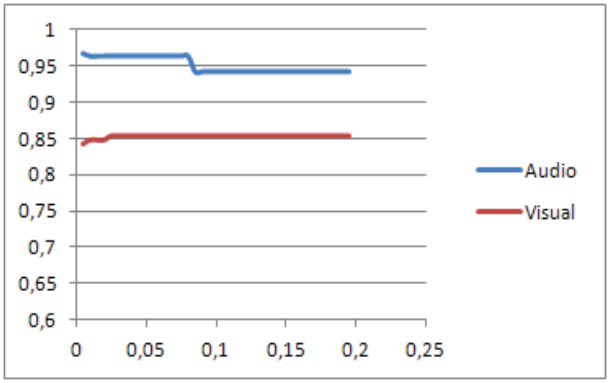
TABLE 4.5: Evaluation of the classification results when using the purity criterion for training - Mag&News.

It is obvious that for the case of audio game recurrences a lot of the false alarms are correctly classified as non-separators resulting in an increase of precision with about 80%. However the recall decreases to 50% but still the global F measure auguments to 64%. An increase of the precision and F measure with a decrease of the recall is also registered for the case of video game recurrences but the impact of the classification tree is quite small, with variations between 2 and 6%. The audio tree better classifies the reccurrences in the non-separator category while the video one those in the separator category. This can be explained by the fact that in the training dataset, most of the audio reccurrences belong to the class non-separtor while for the video ones most of the recurrences are separators. The tree that is built is obviously influenced by the majoritary category.

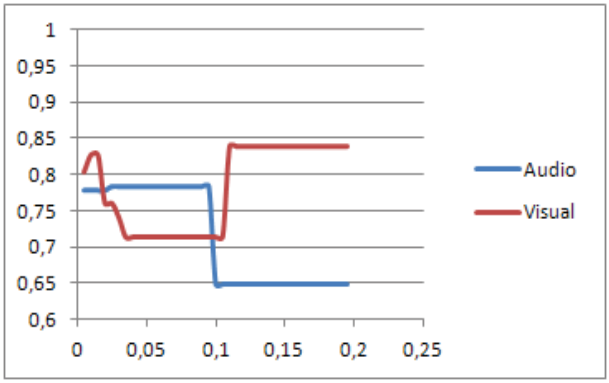
The same conclusions can be drawn for the case of magazines and news programs. Nevertheless, for the audio recurrences, the increase of the precision in not as important as for the games, but in the same time the number of recurrences belonging to the class non-separator is smaller.

### *Classification Results - Entropy Criterion*

For the training of decision trees with the entropy criterion, we tested different thresholds (see Section 4.2.2.2). The results are represented as graphs in Figure 4.6. On the abscissa there are the values of the threshold while on the ordinate the resulted accuracy.




(a) (Accuracy - Games)




(b) (Accuracy - Mag&News)

TABLE 4.6: Accuracy results when using the Entropy Criterion for training the decision trees - Test Dataset.

For each of the cases, starting from a certain threshold on, the situation when none of the attributes, used alone, can not bring an improvement above the imposed threshold appears. We recall that in such situations the decided class is the majority one and we call this a "naive classifier". For the games audio recurrences, this happens from 0.08 while for the magazines and news from 0.1. Before these values, the variations are very small but the accuracy is clearly higher than that obtained with the naive classifier especially for the case of magazines and news. The difference here is more important and can be explained by the number of recurrences belonging to the class 0 which is much smaller than that of the games audio recurrences. As in both cases, the class 0 is the majority, the naive classifier will assign it to all of the recurrences. This will result in a higher accuracy

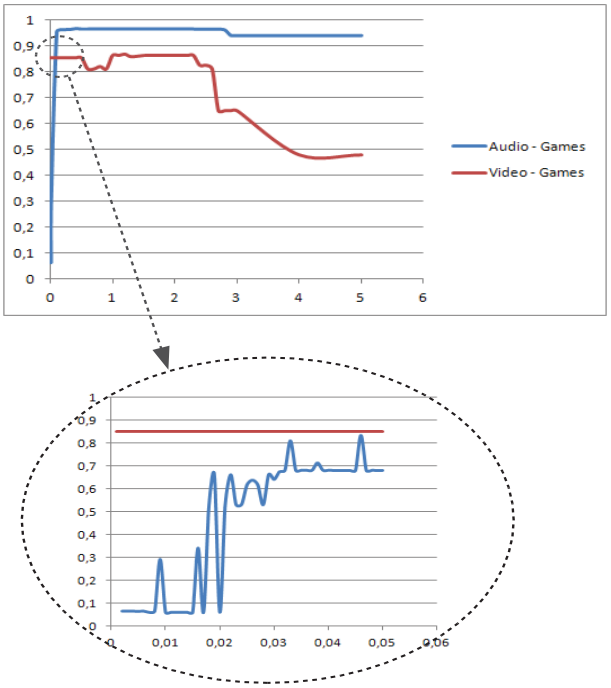
for the games and a much smaller one for the magazines and news. This confirms that the naive classifier is more appropriated for the games, where most of the recurrences are non-separators, than for the magazines and news, where there are less recurrences that are non-separators.

For the case of visual approaches the equivalent situation appears from 0.03 for the games respectively 0.11 for magazines and news. Contrariwise to the audio recurrences, in this case the best accuracy corresponds to the naive classifiers certifying again that for the visual case no classification is needed. An important variation can also be observed for the case of magazines and news. When increasing the threshold, the trees that are built perform weaker and then stabilize for a certain interval.

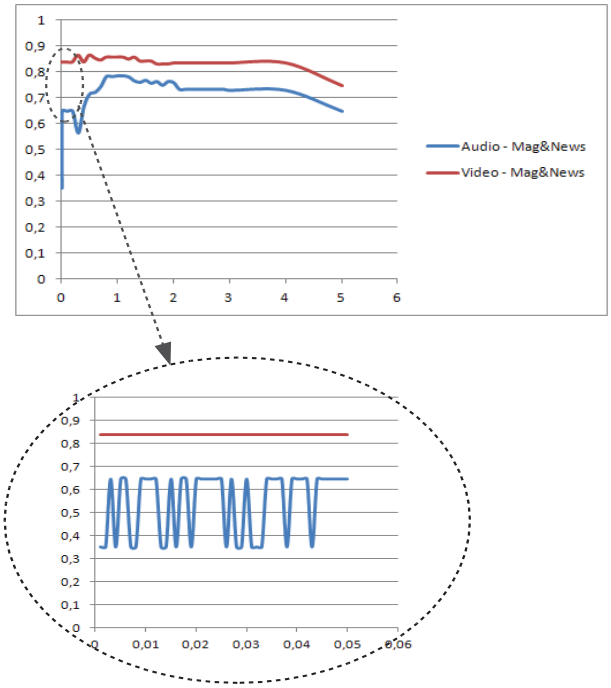
### *Classification Results - Error Minimization Criterion*

When training the decision trees using the error minimization criterion, the idea is to test different weights for the cost of the 2 types of errors presented in Section 4.2.2.2. In order to easy our task we used only one parameter designating the weight of one of the types of errors while keeping the other one to 1. We denote this parameter $\lambda$, and use it for the case of FP (false positive) errors.

Figure 4.7 shows the accuracy results obtained for different values of $\lambda$.




(a) (Accuracy - Games)




(b) (Accuracy - Mag&News)

TABLE 4.7: Accuracy results when using the Error Minimization Criterion for training the decision trees - Test Dataset.

It is obvious that, as expected, for a $\lambda$ very small the assigned category is generally the category of separators (1). This happens due to the fact that the number of false alarms is

significantly reduced by the $\lambda$ parameter. As the algorithm chooses the category for which the number of errors is the smallest, the separators category is thus in advantage.

For the case of games' audio recurrences (Fig 4.7(a)), where only 6% of the recurrences are separators (see Section 4.2.4.3 for more details) this results in a very small accuracy that increases though with the increase of $\lambda$. The acurracy stabilizes to a value arond 0.95-0.97 for $\lambda$ in the interval (0.2, 2.8). For a higher value of $\lambda$, the trained classifiers correspond to the naive classifier that assigns to all the recurrences the category 0. This happens because the number of false alarmes that is already high, is multiplied by a large value of $\lambda$. The algorithm will consequently choose the category for which the number of errors is the smallest, meaning the non-separators category.

In the case of audio recurrences originated from magazines and news, the accuracy results are presented in Figure 4.7(b). The zoom in of the graph shows accuracies computed for small values of $\lambda$. Generally the maxima and minima correspond to the results obtained with the 2 naive classifiers: the minima are obtained when all the recurrences are assigned the class 1 and the maxima when all the recurrences are assigned the class 0. As previously, the first case can be explained by the fact that the $\lambda$ parameter decreases significantly the number of false alarms and consequently the chosen class is 1. The appearance of the second case is what caught our attention. At a closer look, we have noticed that for most of the cases in this situation, none of the attributes, used alone could bring an improvement and consequently the decided category is the majority one, meaning category 0. As the number of recurrences that are separators and those that are not, are quite close (40% respectivelly 60% - see Section 4.2.4.3 for more details), this situation appears quite often for different values of $\lambda$. However from $\lambda = 0.3$ the trees that are bulid start spliting the recurrences among the 2 classses. The accuracy increases and than slightly decreases until, $\lambda = 5$ when all the recurrences are classified as non-separators.

For the audio recurrences coming from games, the number of recurrences that are non-separators is very large (5424 samples) so it is normal to register important variations of the accuracy for $\lambda$ in an interval of small values (0.001-0.1). On the contrary, for the case of magazines and news, the the number of recurrences that are non-separators is smaller (724 samples) so the important variations of the accuracy will appear for greater values of $\lambda$ (0.3-1.5).

Passing to the visual recurrences, for both games and magazines&news, the number of recurrences that are separators is considerably higher than that of non-separators (see Section 4.2.4.3 for more details). Consequently, for a small value of $\lambda$, the category 0 is further reduced as number of samples and thus, most of the times, the category 1 is assigned to the recurrences. For a higher value of $\lambda$, (2 for games and 4 for magazines and news), the accuracy starts decreasing as the number of false alarms is increased by $\lambda$, advantaging thus the choice of the category 0.

#### *Classification Results - Comparing the different criteria*

In order to be able to compare the different criteria, the obtained results are summarized in Tables 4.8, 4.9, 4.10 and 4.11. The lines in the tables correspond to the results obtained when during the training step of decision trees, there were used each of the 3 criteria described in Section 4.2.2 but also the naive classifier. Consequently, $N$ corresponds to the classification results obtained with the naive classifier, M1 corresponds to

the classification results obtained with the decision tree trained using the purity criterion, M2 to the classification results obtained with the decision tree trained using the entropy criterion and finally M3 to the classification results obtained with the decision tree trained using the error minimization criterion.

| Games | **Audio** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **N** | 1 | 0.9420 | 0.9701 | 0 | 0 | 0 | 0.9420 |
| **M1** | 0.9960 | 0.9701 | 0.9829 | 0.8848 | 0.5015 | 0.6401 | 0.9673 |
| **M2** | 0.9961 | 0.9700 | 0.9829 | 0.8889 | 0.4985 | 0.6388 | 0.9673 |
| **M3** | 0.9912 | 0.9761 | 0.9836 | 0.8095 | 0.6053 | 0.6927 | 0.9688 |

TABLE 4.8: Evaluation of the classification results on audio recurrences - Games.

| Games | **Visual** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **N** | 0 | 0 | 0 | 0.8531 | 1 | 0.9207 | 0.8531 |
| **M1** | 0.2581 | 0.5714 | 0.3555 | 0.8832 | 0.9667 | 0.9231 | 0.8625 |
| **M2** | 0 | 0 | 0 | 0.8531 | 1 | 0.9207 | 0.8531 |
| **M3** | 0.6774 | 0.5385 | 0.6000 | 0.9419 | 0.9000 | 0.9204 | 0.8673 |

TABLE 4.9: Evaluation of the classification results on visual recurrences - Games.

| M&N | **Audio** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **N** | 1 | 0.6475 | 0.7860 | 0 | 0 | 0 | 0.6475 |
| **M1** | 0.8656 | 0.8054 | 0.8344 | 0.7139 | 0.6159 | 0.6613 | 0.7776 |
| **M2** | 0.9474 | 0.7709 | 0.8501 | 0.8333 | 0.4828 | 0.6114 | 0.7837 |
| **M3** | 0.9474 | 0.7709 | 0.8501 | 0.8333 | 0.4828 | 0.6114 | 0.7837 |

TABLE 4.10: Evaluation of the classification results on audio recurrences - Mag&News.

| M&N | **Visual** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **N** | 0 | 0 | 0 | 0.8392 | 1 | 0.9125 | 0.8392 |
| **M1** | 0.2683 | 0.7333 | 0.3928 | 0.8750 | 0.9813 | 0.9251 | 0.8667 |
| **M2** | 0 | 0 | 0 | 0.8392 | 1 | 0.9126 | 0.8392 |
| **M3** | 0.1951 | 0.8889 | 0.32 | 0.8658 | 0.9953 | 0.9261 | 0.8667 |

TABLE 4.11: Evaluation of the classification results on visual recurrences - Mag&News.

In the training stage, when using the entropy criteria, different thresholds have been tested (see Section 4.2.4.4). The results in the tables correspond to the threshold that returned the best classification accuracy. For the games' audio recurrences this was obtained for a threshold in the interval $(0.005, 0.080)$. while for the magazines and news

for a threshold in the interval $(0.030, 0.095)$. For the visual recurrences, starting from a threshold up, the results were identical to those obtained with the naive classifier, but as these were better than those obtained for other values of the threshold, these last ones have been considered as best performing.

For the case of the error minimization criteria, we have also experimented different values of $\lambda$ (see Section 4.2.4.4) when training the decision trees. Changing this parameter influences the values of the precision and recall for each of the two categories but when evaluating the global measure of accuracy, the best results were obtained, for a $\lambda \in (0.1, 2.8)$ for the games' audio recurrences, a $\lambda \in (1, 2.3)$ for the games' video recurrences, a $\lambda \in (0.8, 1.2)$ for the mag and news' audio recurrences and finally a $\lambda \in (0.5, 1.3)$ for the mag and news' video recurrences .

When comparing the three methods, we notice that the performances regarding the classification accuracy are quite similar and generally, at least equal or better than those obtained with the naive classifier.

It is interesting to notice that for the case of visual recurrences and for both games and magazines+news, the classification step does not significantly improve the results. The naive classifier that considers all the recurrences as separators performs as good as the decision tree-based classifiers. In particular, when trying the entropy criteria the tree that is build assigns the category "separator" to all the recurrences. This means that for the case of visual recurrences, recurrences that are obtained after applying the filters are good enough and the classification step using decision trees was unable to further improve the results.

As already explained, there is no study that could relate the type of errors to the end-user satisfaction. Nevertheless, in a browsing use-case, separators can be used as anchors that allows users to skip a part and to go directly to the next one, or for instance, to go back to the beginning of the current part. In this case, recall for category 1 becomes most important. Consequently, for visual recurrences and according to Tables 4.9 and 4.11, the naive classifier should be preferred as it provides a recall of 1.

As for the audio recurrences, in the case of TV games the error minimization criterion returns the best recall measure. This criterion allows balancing the importance of samples in the two classes. It is particularly important when the samples of the two classes in the training dataset are unbalanced, which is the case of audio recurrences (94% of the recurrences were actually non-separators). For the audio recurrences of magazines and news programs the best recall measure for the class of separators remains the purity criterion.

We do however have to keep in mind that the results presented correspond to the parameters that returned the best accuracy (we refer to the parameters used for the entropy and error minimization criteria). If we consider the parameters that returned the best recall with an acceptable F measure for category 1, the results are presented in Tables 4.12 and 4.13. We excluded the case of video recurrences classification where we obtained a maximum recall (of 1) for the naive classifiers.

In this case, the classification with trees built using the error minimization criterion outperforms the other criteria for both games and magazines and news. These perform an increase of the R1 with 13% and respectivelly 47% with a decrease of the acurracy not as important, of 2% and respectivelly 20%, comparing to the previous results. We expected from the error minimization criterion to return, in this case, results that show

| Games | Audio | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **N** | 1 | 0.9420 | 0.9701 | 0 | 0 | 0 | 0.9420 |
| **M1** | 0.9960 | 0.9701 | 0.9829 | 0.8848 | 0.5015 | 0.6401 | 0.9673 |
| **M2** | 0.9896 | 0.9723 | 0.9809 | 0.7625 | 0.5430 | 0.6343 | 0.9637 |
| **M3** | 0.9698 | 0.9831 | 0.9764 | 0.5985 | 0.7299 | 0.6577 | 0.9559 |

TABLE 4.12: Evaluation of the classification results on audio recurrences (Best R1) - Games.

| M&N | Audio | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **N** | 1 | 0.6475 | 0.7860 | 0 | 0 | 0 | 0.6475 |
| **M1** | 0.8656 | 0.8054 | 0.8344 | 0.7139 | 0.6159 | 0.6613 | 0.7776 |
| **M2** | 0.9264 | 0.7752 | 0.8441 | 0.7893 | 0.5064 | 0.6170 | 0.7784 |
| **M3** | 0.5070 | 0.9559 | 0.6626 | 0.5138 | 0.9571 | 0.6687 | 0.6656 |

TABLE 4.13: Evaluation of the classification results on audio recurrences (Best R1) - Mag&News.

an important difference comparing to the other criteria. This is explained by the fact that this criterion allows balancing the proportion of samples in the two categories.

It is particularly important when the samples of the two classes in the training dataset are unbalanced, which is the case of audio recurrences.

***Evaluation of the whole solution***

To evaluate the entire solution up to this point, that includes the detection of recurrences and their classification, it is important to assess its performance in detecting separators. For this evaluation, we have used the test dataset and the separators ground-truth (see Section 4.2.4.2). This ground-truth is composed of all the separators manually annotated on all the episodes composing the test dataset. Precision, recall and F-measure have been used to measure the performance of separator detection.

| | **Audio** | | | **Visual** | | | **Audio $\bigcup$ Visual** | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| **Games** | 0.09 | 0.94 | 0.16 | 0.89 | 0.65 | 0.75 | 0.09 | 0.96 | 0.17 |
| **M&N** | 0.42 | 0.89 | 0.57 | 0.88 | 0.40 | 0.55 | 0.51 | 0.92 | 0.66 |

TABLE 4.14: Separator detection using only recurrences, without any classification.

Table 4.14 shows the performance of separator detection if we consider all of the detected recurrences as separators, that is no classification is performed on the detected recurrences. As expected, the lowest precision corresponds to the audio recurrences in the TV games dataset. Most of these recurrences are not separators. They are false alarms. Regarding the visual recurrences, the precision is much higher meaning that most of the

visual recurrences are separators. The best recall is obtained for the audio approach where a high number of recurrences is found. The recall for the visual recurrences is not as high due to the fact that the visual recurrence detection algorithm that we proposed detects only near-identical recurrences. Separators containing, for example, the logo of the show superposed on natural images of the set , are not detected with our algorithm.

The last column in the table represents the union of the results obtained for the audio and visual approaches separately. We added this column as it can provide information about the contribution of each approach but also about the overall performance in detecting separators, when considering both approaches (audio and video).

To evaluate the recurrence detection and classification algorithm we applied on the detected recurrences, the classifiers presented in the previous subsection and we evaluated the results using the separators ground-truth. As the three tested methods used for training the trees showed similar results for the accuracy (see Tables 4.8, 4.9, 4.10 and 4.11), we preferred using the classifier based on the purity criterion that does not implies any parameters while performing almost similar as the other ones.

The obtained results are presented in Table 4.15.

| | **Audio** | | | **Visual** | | | **Audio $\bigcup$ Visual** | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| **Games** | 0.92 | 0.59 | 0.72 | 0.93 | 0.63 | 0.75 | 0.93 | 0.64 | 0.76 |
| **M&N** | 0.77 | 0.50 | 0.61 | 0.92 | 0.39 | 0.55 | 0.79 | 0.50 | 0.62 |

TABLE 4.15: Separator detection after applying recurrence classification (Parameters chosen for Best Accuracy).

The general tendency after classification is an improvement of precision at the cost of a decrease for the recall. This means that recurrences that are not separators are filtered out but in the same time some of the true separators too. There are several reasons that could lead to such consequences. Among them, the noise induced by the different technologies that we used when computing the attributes. Another reason, equally important, could be the erroneous choice of attributes that are not as relevant for our task as we might have expected. Also the limitations of the decision trees should be considered.

Nevertheless, if we take a close look in Table 4.15 for the case of visual recurrences the results are not highly impacted. The most important changes appear for the case of audio recurrences and implicitly for the audio-visual final results. In this manner, when considering the games, there is an important increase of the precision with 83% after classification, with a decrease in the recall of 35%. However the F-score that combines the precision and recall measures is superior with 52%. For the magazines and news the variation of the precision and recall is around 30-40%, with an F-score that decreases with 4%.

As already said, these results were obtained when using the classifiers chosen based on the accuracy results. However, if we reconsider the case discussed in the final part of the previous subsection, meaning the classifiers that perform best in terms of recall for category 1, the results are showed in Table 4.16. We recall that these classifiers correspond to the ones trained with the error minimization criteria ($\lambda$ chosen for best R1 measure) for the audio recurrences and to the naive classifiers for the visual recurrences(see Tables 4.8, 4.9, 4.10 and 4.11).

| | Audio | | | Visual | | | Audio ⋃ Visual | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| **Games** | 0.66 | 0.77 | 0.71 | 0.89 | 0.65 | 0.75 | 0.69 | 0.81 | 0.74 |
| **M&N** | 0.62 | 0.84 | 0.71 | 0.88 | 0.40 | 0.55 | 0.63 | 0.84 | 0.72 |

TABLE 4.16: Separator detection after applying recurrence classification (Parameters chosen for Best R1).

Obviously, the recall is higher in this case, but the precision decreases. However, the F-score slightly varies and in some cases increases of even 10%. Comparing to the results obtained without any classification (Table 4.14) the same tendancy is observed but in this case the cost in recall for having a higher precision is smaller especially for magazines and news resulting thus in higher F scores.

Regarding the attributes implied in the construction of the decision trees, the first positions in most of the trees generally correspond to $A_0$, $A_2$, $A_6$ and $A_7$. For more details the reader can refer to Annex B. To make an idea about the most important attributes that are implied in the construction of the decision trees, we looked at the first 4 attributes (first 4 levels) of the trees we used for classification. For the case of games, $A_0$, $A_1$, $A_2$, $A_4$, $A_5$ and $A_7$ are employed while for the magazines and news the most important attributes are $A_0$, $A_1$, $A_2$, $A_3$, $A_4$, $A_5$, $A_6$ and $A_7$. However the first positions in most of the trees generally correspond to $A_0$, $A_2$, $A_6$ and $A_7$. For more details the reader can refer to Annex B.

As a parenthesis, we have to mention that during our study, we have tried to add 2 more attributes: a first one referring to the presence of music in the studied recurrence and a second one concerning the presence of speech in the recurrence. However, the results were strongly similar to the previous ones and thus we preferred continuing with only the set of 13 attributes. What we noticed however is that the attribute related to the presence of speech is sometimes replacing those referring to the presence of faces or speakers. But as already said this does not influence the results.

#### 4.2.4.5 Recurrence Classification Results - Complex attribute training

In order to extend further the use of decision trees in Section 4.2.2.1, we have proposed the use of combinations of attributes in the training stage. Two cases have been presented: In the first one, the complexity parameter (c) is introduced by the user while in a second one, all the possible combinations of attributes are tested. We computed experiments for both approaches and also for each of the proposed training criteria described in Section 4.2.2.2. The obtained results are presented in the tables from Annex C.

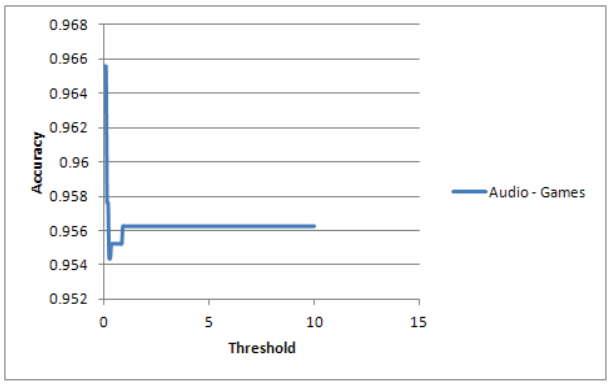
An overall analysis reveals that the results in terms of accuracy are not significantly different. However, globally, the trees trained with single attribute questioning perform best while those obtained when testing all the possible combinations have poorer performances, especially for the purity and entropy criteria. Forcing to test a certain number of attributes imposes some limitations when searching the best attribute to test. For example not in all cases a combination of $c$ attributes exceeds the performance of a single one. However, we would have expected that the classification results obtained with trees trained when all the possible combinations are tested, to outperform.

This is not at all the case for the purity and entropy criteria. In this cases generally the combination of attributes that is chosen as best, contains all or most of the attributes.

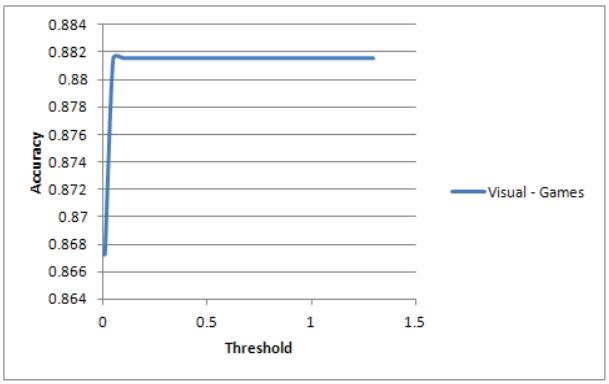
This is likely due to an overfitting problem, as the more attributes we use, the more the entropy decreases and the purity improves.

Nevertheless this does not happen when using for training the error minimization criterion. For this case, the accuracies are comparable to those obtained for single attribute training, but still do not exceed these last ones. When training the trees using the error minimization criterion, adding more attributes improves the error at first, but when adding too many it starts to degrade it. The training stops thus before the 13 attributes and the performance on the test dataset is better.

As previously said, one of the possible causes of the obtained results might be the overfitting of the data. To bear out this supposition we computed experiments using different thresholds to stop the training of the tree and to avoid the overfitting. The results obtained are presented as graphs in Figure 4.17 and 4.18.

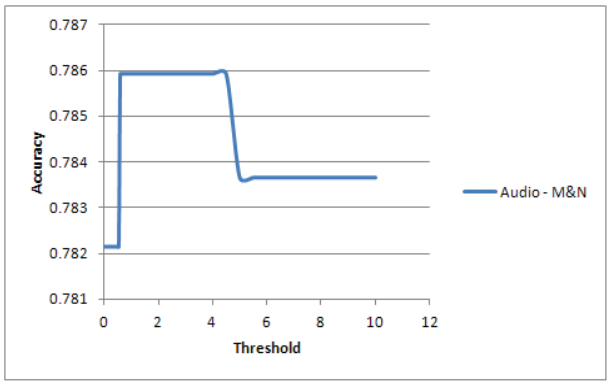



(a) (Audio)

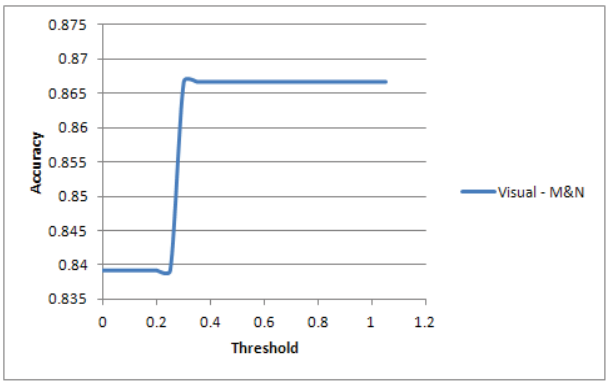



(b) (Visual)

TABLE 4.17: Accuracy results when using the Err. Min. Criterion for training decision trees with different thresholds and *complexity* = *all* - Test Dataset - Games .




(a) (Audio)




(b) (Visual)

TABLE 4.18: Accuracy results when using the Err. Min. Criterion for training decision trees with different thresholds and *complexity* = *all* - Test Dataset - Mag&News.

For three of the four cases (the exception is the case of audio recurrences provided by game TV shows), stopping the training of the tree before its end, seems a good solution as the accuracy improves. However, this improvement is too small to be considered, with an improvement of maximum 3% comparing to the previous accuracy value and 2% comparing to the accuracy obtained for single attribute training.

It is difficult to conclude on the reasons that may influence these results, but we do have to keep in mind that the proposed solution might be too complex relative to the number of samples in the training set. When testing all the possible combinations of attributes, for 13 attributes that can take the values 0 or 1, there are $2^{13}$ possibilities to test. Moreover, these attributes are not totally independent. Overfitting may also appear. To avoid it, a threshold could be used but is also difficult to find the optimal threshold.

In conclusion, in our case, it certainly is not worth to follow this solution, as, first of all, the gain is too small. Second, the threshold chosen to stop the training is difficult to find and depends of the training data. And third the computation time for building decision trees when all the possible attributes combinations are tested, is much larger than when performing a simple training. Consequently, the benefits are too small compared to the costs of the solution.

## 4.3 SVM vs. Decion Tree Classification

In this section, we would like to consider the use of another type of classifier, besides the decision trees that have been used up to this point for the classification of audio and visual recurrences. This would allow us to compare the results we have until now and conclude if these are more influenced by the limitations of decision trees or by other factors like the noise induced by the different technologies that we used when computing the attributes or even the erroneous choice of attributes that are not as relevant for our task as we might have expected.

### 4.3.1 Suport Vector Machine Classifier

As the data we deal with is characterized by a set of binary attributes, and the intended goal is to classify the data into 2 categories (0 and 1), one appropriate classifier would be the Support Vector Machine (SVM). Just as the decision trees, this is a supervised learning method capable to analyze and recognize patterns in a set of input samples and to assign them to one of the two categories. Based on a set of training samples, for which the belonging category is known, the SVM algorithm builds a pattern/model. This model is afterwards used to assign new samples into one category or the other.

The samples are represented as points in an high-dimensional space and are mapped so that the samples belonging to the 2 categories are separated by a gap as wide as possible. With an appropriate nonlinear mapping to the higher dimensional space, a set of hyperplanes that separate the 2 categories can be built. The goal of training a Support Vector Machine is to find the maximum-margin hyperplane that insures the best separation, meaning the largest distance to the nearest training point (see Figure 4.7). The idea is that with a larger margin the generalization of the classifier is better. When classifying new samples, these are also mapped in the high dimensional space and assigned to one of the 2 categories, regarding the side of the hyperplane they fall in.

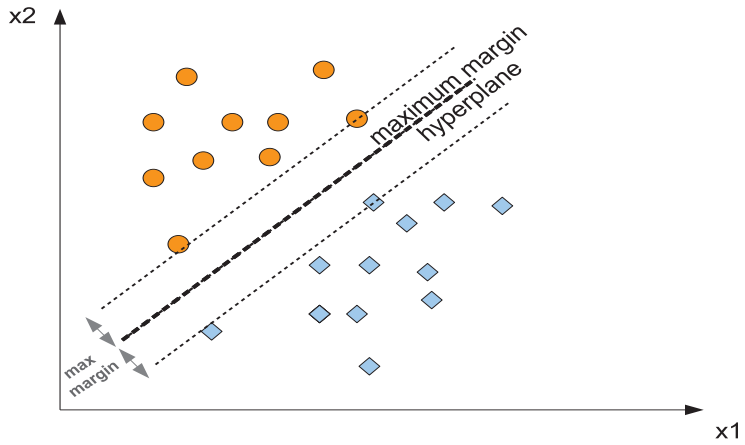



FIGURE 4.7: Find optimal hyperplane when training a support vector machine.

To map the samples into higher-dimensional space kernel functions are used. Some basic kernel functions often employed are:

- linear: $K(x_i, x_j) = x_i^T x_j$.
- polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$.
- gaussian radial basis function (RBF): $K(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2)$, $\gamma > 0$.
- sigmoid: $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$.

where $\gamma$, $r$ and $d$ are kernels parameters ( [wHcCjL10]).

SVMs allow thus the use of high-dimensional spaces where a linear separation is possible, due to a mapping of the data made by kernel functions.

For more details the reader can refer to [DHG01, Mit97].

### 4.3.2 Recurrence Classification Results

In order to classify the audio and visual recurrences with SVM classifiers, we have used an open source data mining software in Java, named Weka, that contains a collection of machine learning algorithms for data mining tasks.

Among the implemented classifiers there is also the Support Vector Machine (linear, polynomial and RBF kernel) with Sequential Minimal Optimization Algorithm as described in [Pla98].

To compute the experiments, we have followed the steps suggested in [wHcCjL10]. The kernel we used is the RBF kernel. The choice of this type of kernel is motivated, as in [wHcCjL10], by the fact that, first of all, this is a nonlinear kernel. As the number of features that describe our samples is not too large (13), a non linear kernel that maps data to higher dimensional spaces is suited. Moreover, the linear and the sigmoid kernel, perform similar to RBF for certain parameters of the kernels( [KLX03, tLL03]). Comparing to the polynomial kernel, RBF has fewer parameters to adjust which influence the complexity of the model selection.

When using an SVM with an RBF kernel 2 parameters have to be considered: $\gamma$, the kernels' parameter and $C$, the soft margin parameter. The parameter $C$ represents the cost parameter of the SVM model. Ideally, an SVM analysis should produce a hyperplane that completely separates the feature vectors into two non-overlapping groups. However, perfect separation may not be possible, or it may result in a model with so many feature vector dimensions that the model does not generalize well to other data (over-fitted model).

To allow some flexibility in separating the categories, a cost parameter, $C$, is used. It controls the trade-off between allowing training errors and forcing rigid margins. It creates a soft margin that allows some misclassifications. Increasing the value of C increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well (`http://www.dtreg.com/svm.htm`).

The accuracy of the SVM model is thus dependent on the choice of these 2 parameters. In order to find the optimal values, we have used a grid search with exponentially growing values of C and $\gamma$: $C \in \{2^{-5}, 2^{-3}, ...2^{16}\}$, $\gamma \in \{2^{-16}, 2^{-13}, ...2^{3}\}$.

Each combination of parameters (C, $\gamma$) is tested using cross validation and the one with the best cross validation accuracy is picked.

Tables 4.19, 4.20, 4.21 and 4.22 show the results on the training dataset when using the grid search (SVM2) or not (SVM1). A small improvement is noticed for the accuracy in

the case when using the grid search, but the differences are not very important. However, we keep both methods for future experiments.

| Games | **Audio** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **SVM1** | 0.970 | 0.994 | 0.982 | 0.848 | 0.534 | 0.655 | 0.965 |
| **SVM2** | 0.978 | 0.987 | 0.983 | 0.776 | 0.664 | 0.715 | 0.968 |

TABLE 4.19: SVM classification of audio recurrences from the training set - Games.

| Games | **Visual** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **SVM1** | 1 | 0.1 | 0.182 | 0.843 | 1 | 0.915 | 0.845 |
| **SVM2** | 0.632 | 0.3 | 0.407 | 0.869 | 0.964 | 0.914 | 0.850 |

TABLE 4.20: SVM classification of visual recurrences from the training set - Games.

| M&N | **Audio** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **SVM1** | 0.738 | 0.877 | 0.802 | 0.754 | 0.548 | 0.635 | 0.743 |
| **SVM2** | 0.743 | 0.892 | 0.811 | 0.779 | 0.552 | 0.646 | 0.754 |

TABLE 4.21: SVM classification of audio recurrences from the training set - Mag&News.

| M&N | **Visual** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **SVM1** | 0.822 | 0.487 | 0.612 | 0.847 | 0.964 | 0.902 | 0.843 |
| **SVM2** | 0.804 | 0.592 | 0.682 | 0.873 | 0.951 | 0.910 | 0.860 |

TABLE 4.22: SVM classification of visual recurrences from the training set - Mag&News.

The models previously learned are then applied on the test datasets and the results obtained are presented in Tables 4.23, 4.24, 4.25 and 4.26. In order to facilitate the comparison with the results obtained for the decision trees, the first two lines in the tables correspond to these last ones. The first line presents the results obtained with the decision trees in terms of best accuracy. The second one shows the same but in terms of best R1 measure (see Section 4.2.4.4). The last two lines in the tables correspond to the classification made with the SVM models.

When comparing the two experimented methods, the Decision Trees and the Support Vector Machine, no significant difference can be noticed. As it can be seen in the tables above, none of the methods implied, reported significantly higher accuracy comparing to the other one. This conclusion is significant as it represents a strong evidence that the results obtained were not highly impacted by the limitations of the method chosen for classification. It is thus most probable that much more important are the attributes we have defined and the limitations of the methods used when computing these attributes.

| Games | Audio | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **DC1** | 0.991 | 0.976 | 0.984 | 0.809 | 0.605 | 0.693 | 0.969 |
| **DC2** | 0.970 | 0.983 | 0.976 | 0.598 | 0.730 | 0.658 | 0.956 |
| **SVM1** | 0.970 | 0.996 | 0.983 | 0.889 | 0.499 | 0.639 | 0.967 |
| **SVM2** | 0.977 | 0.987 | 0.982 | 0.752 | 0.629 | 0.685 | 0.966 |

TABLE 4.23: Evaluation of the classification results on audio recurrences - Games (Test dataset).

| Games | Visual | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **DC1** | 0.677 | 0.538 | 0.600 | 0.942 | 0.900 | 0.920 | 0.867 |
| **DC2** | 0 | 0 | 0 | 0.8531 | 1 | 0.921 | 0.853 |
| **SVM1** | 1 | 0.067 | 0.125 | 0.865 | 1 | 0.928 | 0.870 |
| **SVM2** | 0.579 | 0.367 | 0.449 | 0.901 | 0.956 | 0.927 | 0.871 |

TABLE 4.24: Evaluation of the classification results on visual recurrences - Games (Test dataset).

| M&N | Audio | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **DC1** | 0.947 | 0.771 | 0.850 | 0.833 | 0.483 | 0.611 | 0.784 |
| **DC2** | 0.507 | 0.956 | 0.663 | 0.514 | 0.957 | 0.669 | 0.666 |
| **SVM1** | 0.801 | 0.915 | 0.854 | 0.788 | 0.584 | 0.671 | 0.798 |
| **SVM2** | 0.798 | 0.927 | 0.858 | 0.810 | 0.569 | 0.668 | 0.801 |

TABLE 4.25: Evaluation of the classification results on audio recurrences - Mag&News (Test dataset).

| M&N | Visual | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **DC1** | 0.195 | 0.889 | 0.320 | 0.866 | 0.995 | 0.926 | 0.867 |
| **DC2** | 0 | 0 | 0 | 0.839 | 1 | 0.913 | 0.839 |
| **SVM1** | 0.542 | 0.325 | 0.406 | 0.883 | 0.949 | 0.914 | 0.850 |
| **SVM2** | 0.560 | 0.350 | 0.431 | 0.886 | 0.949 | 0.916 | 0.854 |

TABLE 4.26: Evaluation of the classification results on visual recurrences - Mag&News (Test dataset).

## 4.4 Conclusion

The proposed idea in this chapter was to compute the performance of a detection and classification of audio and visual recurrences system based on decision trees, on the detection of "separators", as these represent the root of an automatic recurrent TV program structuring system.

Experiments showed us that our main assumption, that separators are repeated and could be detected as recurrences is validated. However, audio and visual recurrence detection algorithms perform differently. The audio one detects a high number of recurrences. Among them there are the separators but the majority of the recurrences are non-separators. To notice is the fact that TV games provide the highest number of audio recurrences that are "non-separators" as specific sounds (i.e. pushing a button) are intensively repeated. The visual-based algorithm detects less recurrences with a majority of separators.

When classifying these recurrences, using decision trees trained with the 3 criteria proposed in Section 4.2.2.2, and single attribute questioning method, the performances regarding the classification accuracy are similar. Moreover, for the case of visual recurrences, these do not exceed significantly the naive classifier meaning that in this case a classification step in not really necessary.

The evaluation of the performance in detecting separators of the whole solution, showed that a lot of the recurrences that are not separators are filtered out (especially for the audio recurrences) but with them a part of true separators too. This resulted in an increase of the precision but at the cost of a decrease in the recall. However, when analyzing globally the results using the F-measure, especially for the audio case, the results are better after performing the classification step.

To further extend the use of decision trees, a second approach that considered multiple attributes questioning, when performing the training, has been tested. Nevertheless, this revealed to be too costly comparing to the benefits it brought comparing to the solution where single attribute training was performed. The single attribute training approach remains thus the most appropriate.

As already stated, the compromise that generally unveiled between the precision and recall, could be a result of several reasons like the noise introduced by the technologies used for the attributes computation, the proposed attributes that are not relevant enough or the limitations of the decision trees.

The last cited speculation can however be tested by employing a different classifier appropriated for the type of data we are dealing with. One known powerful classification tool that we considered suited is the Support vector machine. We have used it thus for classification of the same dataset. The results were very similar to those obtained with the decision trees, which rules out thus the hypothesis that the decision trees limit the results obtained for the classification of audio and visual recurrences. Consequently, the limitation of the obtained results are more likely due to the attributes that might need some improvements.

# General conclusion

*That which does not kill us makes us stronger.*

Friedrich Nietzsche, philosophe

This thesis concludes three years of work focused on the proposal of a method that aims at structuring in an unsupervised manner a large category of TV programs. TV Programs are generally broadcasted in a linear way. Services like TV-on-Demand or Catch-up-TV break this linearity and provide users with the possibility to watch video or audio content on demand. However, these do not offer tools for users to directly access and browse within TV programs. Such features could however be available if the structure of the programs would be known. To find the structure of TV programs, we proposed thus a method based on the detection and classification of audio and visual recurrences. It allows us to identify the different parts composing a program without the use of any prior knowledge about the program or about the number of parts the program is composed of. Knowing the different parts composing a program enables features like browsing, indexing and querying or the creation of more accurate video summaries. We summarize hereafter the proposed approach and we propose some possible perspectives.

## Synthesis and contributions

The method we proposed analyses a set of episodes of a recurrent TV program in order to extract the "separators". These are short video sequences, inserted between different parts of a program, and that can be repeated between and/or within the episodes of the same program. Once the separators found, these can be used afterwards to structure each of the input episodes or each new arriving episode.

The basis of our approach are the separators. These represent the root of our automatic recurrent TV program structuring system. The main assumption we made and also validated, is that separators are repeated and can be detected as recurrences. The starting point of our system becomes thus the detection of recurrences. As separators have visual or/and audio recurrent content, we first followed two axis and tested the performances of detecting separators among recurrences for each modality (visual/audio) separately.

Among the detected recurrences there are also the separators but not all the recurrences are necessarily separators. Consequently a filtering process based on the temporal and spatial distribution of the recurrences is required.

Audio and visual recurrence detection algorithms perform differently. The audio one detects a high number of recurrences. Among them there are the separators but the major-

ity of recurrences are non-separators. The visual-based algorithm detects less recurrences with a majority of separators. Both approaches have limitations: for instance in the case of audio recurrence detection algorithm the separators with overlaid natural speech cannot be detected. For the case of the video recurrence detection algorithm, the separators that are not identical are not detected. It might also happen that the employed clustering algorithms, that involve an optimal setting of a certain number of parameters, to wrongly perform for certain specific cases. We presumed thus that when combining both approaches, one could overcome the limitations of the other and provide thus the necessary for a better structuring. Therefore, in order to combine the two modalities and recover the maximum number of separators we proposed, in a last step, to pass all the remained audio and visual recurrences, through a classification and selection system, based on decision trees, that decides whether a recurrence is a separator or not. To build the decision trees, attributes issued from techniques like applause detection, scenes segmentation, face/speaker detection and clustering were used.

The obtained results showed that a lot of the recurrences that are not separators are filtered out (especially for the audio recurrences) but with them part of true separators too. However, when analyzing globally the results using the F-measure, especially for the audio case, the results are better after performing the classification step.

We have also considered and used for comparison another classifier based on SVMs. No significant differences were registered for the obtained results. This represents a strong evidence of the fact that the results were not highly impacted by the method chosen for classification but moreover by the chosen attributes and the limitations of the methods used when computing these attributes.

In order to conclude this section, we resume hereafter what we consider as beeing the main contributions of this thesis.

First of all, we propose an original approach for structuring TV programs, that is not specific to a certain type of program and that addresses a large category of programs like the recurrent TV programs.

Second, the approach we propose for structuring is largely unsupervised. There is no need of any prior knowledge about the programs or about the number of parts a program has. The first part of the proposed approach, meaning the detection and filtering of audio and visual recurrences is completely unsupervised.

The last part of the proposed approach where a classification module based on decision trees or SVMs is employed questions the notion of supervision. However, even in this case, we showed that for the visual recurrences there is no need of a classification phase which allows thus for the visual approach to remain completely unsupervised. Nevertheless, for the case of audio recurrences too many recurrences are detected and a classification step is needed. The classification shows interesting results and becomes imperative for the case of audio approach even if it makes this last part of the approach supervised.

Third, the proposed approach exploits the visual and the audio content of TV programs. It is based on classification techniques that filter and select *separators*, among a set of detected visual and audio recurrences. Finally, experiments are computed on real TV broadcasts in order to validate the ideas on which the proposed approach is based on and to test the proposed classification algorithms.

## Perspectives

The work presented in this thesis could still be continued with new perspectives. Some of them refer to the improvements that could be brought to this thesis. Others go further and propose new domains where the proposed approach could be tested and exploited.

One first important perspective is derived from the difficulties we encountered when analyzing the obtained results with the proposed evaluation measures of precision, recall and F score. For the moment, no study has been made to relate the type of errors to the end-user satisfaction for a specific application. For instance, if we consider the specific case of browsing, it is difficult to decide whether when browsing inside a TV show, it is more annoying to have some missing separators (related to recall) or to have to skip some recurrences that are not separators (related to precision). Likewise, the number of recurrences that are not separators that could be tolerated by an user is not known. An important point, when using the structuring results for a real world service, would thus be to relate the recall and precision measures to the end-user satisfaction. This analysis would give an idea about the types of errors of little disturbance or those more important for the user. It will allow thus to select the best compromise for the process with highest quality and reduced computational cost.

Another interesting perspective could be the classification of separators into different categories. During experiments we noticed that some separators were not appropriated for our approach or others were impossible to detect as recurrences (when not considering a sufficiently large history). For the former, examples for the visual approach are separators having on the background images with the set, the anchor or the audience, separators with highly animated logos and/or moving backgrounds. Examples of such separators can be found in Section 2.4.3. For the later examples are separators presenting an object that will be won. This object may or may not be different from one show to another. Hence, the recurrence based algorithm will or will not be able to detect the corresponding sequence.

In the case of the audio approach such situations also appear. Examples are presented in Section 3.4.4 and refer to separators containing natural spoken language or separators having background noise like applause, music or audience cheers. All these are not handled by our algorithm since our focus was on near identical repeated sequences.

Classifying the separators into different categories would suppose the identification of such special cases of separators and their labeling accordingly. This will allow afterwards an easier and more precise analysis of the results.

To mention is also the fact that we have been constrained by a short history when computing the recurrences detection for both audio and visual approaches, as the dataset on which we worked on is not extremely large. A bigger dataset, would become a technical challenge as large volumes of data would have to be handled. On the other hand, it would allow a more precise analysis on a longer period of time and on a higher number of broadcasted episodes. Moreover, it could also solve the problems of the proposed classification system. A long period of time, means a large number of episodes, with a higher number of recurrences. The more samples we have, the better the training of classifiers that would then provide more stable classes.

During experiments we have also noticed that there are recurrences detected within the main parts of the program. Some of these represent markers that could be used for structuring the part of the program they are found in. The scope of this thesis was however

to structure TV programs in their main parts, without any interest in the content of each part. These recurrences were thus not included in the ground truth. Nevertheless, as a further perspective, these recurrences could be used for a higher level of structuring TV programs or a "dense" structuring, that considers also the content of each part composing a program. In such a case, it would be usefully a classification of recurrences on different levels like identical recurrences, near-identical recurrences, repeated words, same visual content but different points of view, etc. Depending on the class it belongs to, a recurrence will be used to separate the parts of a program, to separate different topics inside a certain part of a program, to identify important events inside a certain part of a program, to identify important informations or track stories.

Another perspective goes farther, towards video summarization. The intra program structuring could be viewed as a preliminary stage for video summarization. Knowing the structure of a program improves the comprehension of the program which could lead to more precise and evolved summaries. To verify this statement, video summarization algorithms should take into consideration the different parts composing a program, when choosing the most important events to be included in the summary.

# Annexes

Summary

Annex A

# The experimental dataset

All the experiments computed in this thesis are based on a single experimental dataset. In order to facilitate the reading of this manuscript we describe in detail this dataset in this Annex.

| Program Type | Program Name | Program Id | Nb. of episodes | Nb. of separators/episode |
|---|---|---|---|---|
| Games | Les Z'amours | $G_1$ | 28 | 6 |
| | Mot de passe | $G_2$ | 5 | 4 |
| | Motus | $G_3$ | 21 | 4 |
| | Tlmvpsp | $G_4$ | 26 | 5 |
| | Les 12 coups de midi | $G_5$ | 20 | 8 |
| Magazines News | Comment ca va bien | $M_1$ | 24 | 8/14 |
| | 10h Le Mag | $M_2$ | 5 | 20/23 |
| | Cine, Series et cie | $M_3$ | 7 | 15/17 |
| | 50mn Inside | $M_4$ | 14 | 12/18 |
| | 7 A 8 | $M_5$ | 10 | 5/7 |
| | 19h45 | $M_6$ | 9 | 20/27 |

Table 27: Experimental dataset description.

The experimental dataset contains about 112 hours of videos, corresponding to 169 episodes (with 1594 separators) of 11 TV programs, broadcasted on 4 French TV channels (TF1, France2, M6 and Orange Cinemax).

Five of the programs, are TV games. The separators in this case delimit the different stages of the game. The duration of an episode varies between 25 and 45 minutes. Six others are TV magazines. The separators in this case delimit the reports and the scenes where the anchor presents the next topic. Each episode lasts about 50 minutes. The last program is a news program and it is similar to the magazines. It lasts 20 minutes and it is composed of a set of reports and anchor person scenes.

As magazines and news programs have almost the same structure (set of reports and anchor person scenes), different from that of the games, these are generally presented in a same section, just as in Table 27. The same reasoning explains the id assigned to news programs which is similar to the ones assigned to magazines. However, during experiments, besides global results, all the programs will also be treated separately and results will be presented for each.

For all the programs presented in Table 27 the separators are inter- or/and intra-episode repeated separators.

Annex B

## Examples of decision trees

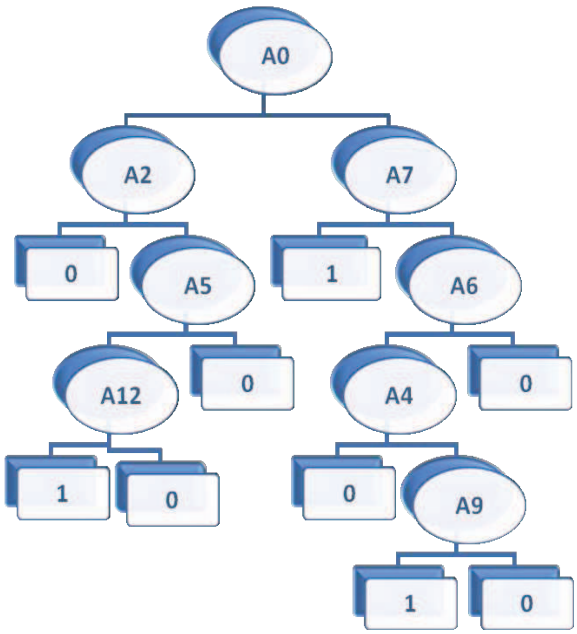



FIGURE 8: Decision tree built for Games' audio recurrences - Best Accuracy.

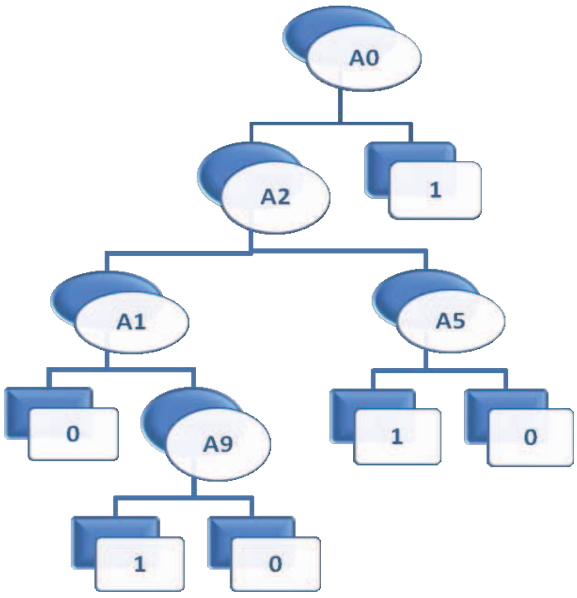



FIGURE 9: Decision tree built for Games' audio recurrences - Best R1.

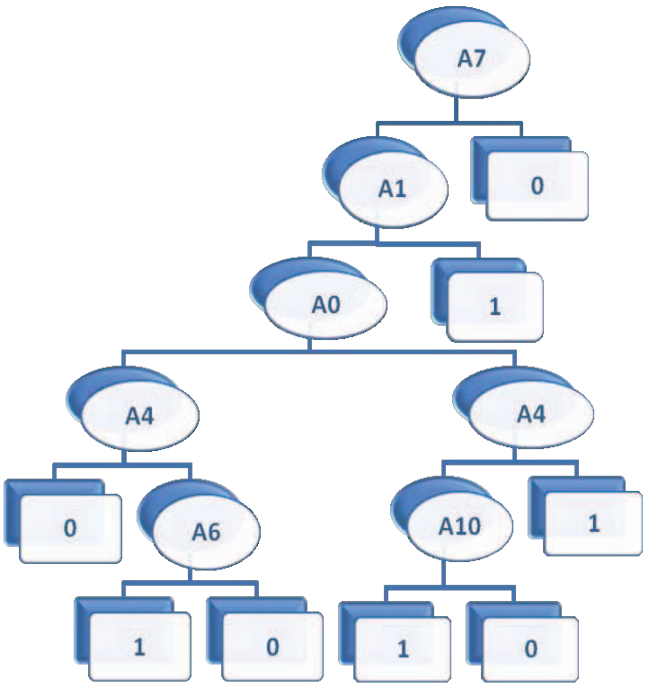



FIGURE 10: Decision tree built for Games' visual recurrences - Best Accuracy.


FIGURE 11: Decision tree built for Magazines&News' audio recurrences - Best Accuracy.

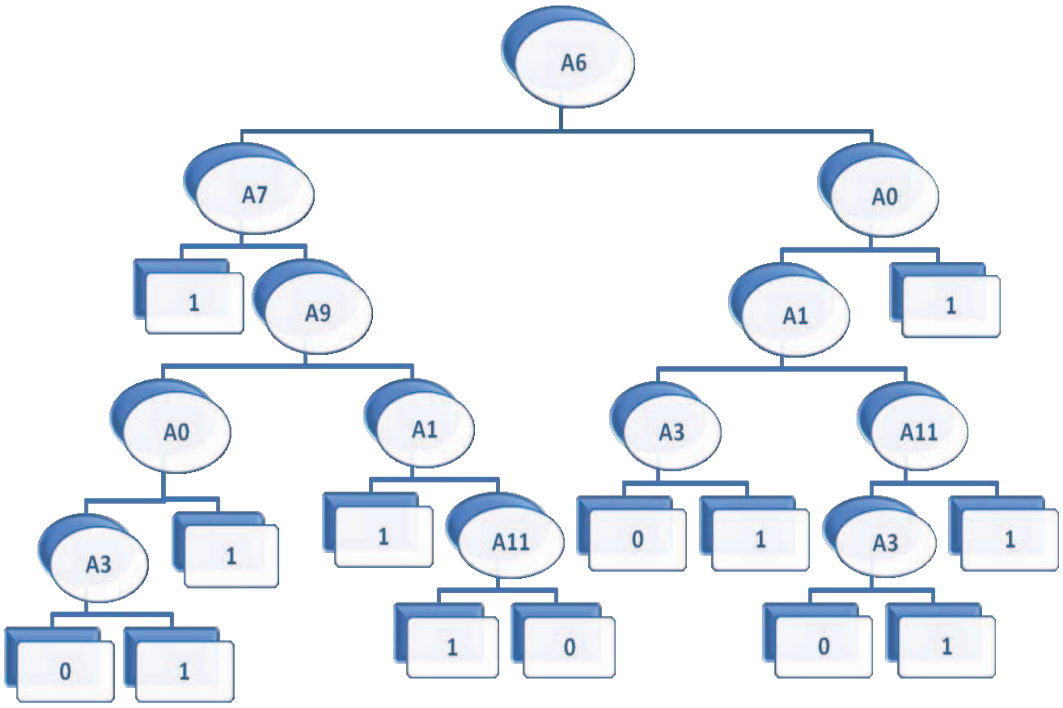



FIGURE 12: Decision tree built for Magazines&News' audio recurrences - Best R1.

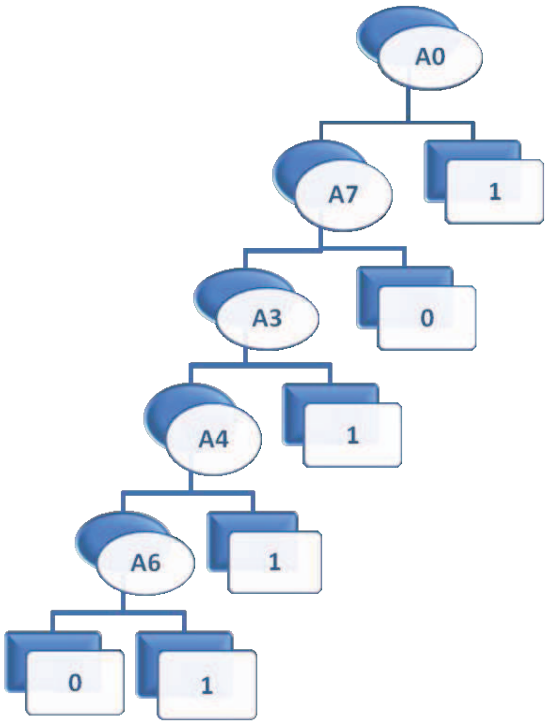



FIGURE 13: Decision tree built for Magazines&News' visual recurrences - Best Accuracy.

Annex C

# Classification results on test dataset - trees built with complex attribute training

The tables below, show the results obtained when classifying the audio and visual recurrences int the test datasets with the decision trees trained with different criteria. The lines in the tables correspond to the two approaches proposed in Section 4.2.2.1: single and respectively complex attribute training. The first line provides the results obtained after classification with decision trees trained with single attribute questioning. The next three present the results obtained after classification with decision trees trained with complex attribute questioning, where the *complexity*($c$) is either equal to 2 (combinations of 2 attributes are tested), 4 (combinations of 4 attributes are tested), or "all" (all the possible combinations of attributes are tested).

## Classification results on test dataset - Purity Criterion with complex attributes training

| Purity Criterion | **Audio - Games** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.9960 | 0.9701 | 0.9829 | 0.8848 | 0.5015 | 0.6401 | 0.9673 |
| **c = 2** | 0.9932 | 0.9735 | 0.9833 | 0.8363 | 0.5608 | 0.6714 | 0.9681 |
| **c = 4** | 0.9874 | 0.9767 | 0.9820 | 0.7509 | 0.6172 | 0.6775 | 0.9659 |
| **c = all** | 0.9587 | 0.9652 | 0.9620 | 0.3957 | 0.4392 | 0.4163 | 0.9286 |

TABLE 28: Classification results on audio recurrences - Purity Criterion - Games.

| Purity Criterion | **Visual - Games** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.2581 | 0.5714 | 0.3555 | 0.8832 | 0.9667 | 0.9231 | 0.8625 |
| **c = 2** | 0.3226 | 0.3448 | 0.3333 | 0.8846 | 0.8944 | 0.8895 | 0.8104 |
| **c = 4** | 0.3548 | 0.3793 | 0.3667 | 0.8901 | 0.9000 | 0.8950 | 0.8199 |
| **c = all** | 0.1935 | 0.1224 | 0.1500 | 0.8457 | 0.7611 | 0.8012 | 0.6777 |

TABLE 29: Classification results on visual recurrences - Purity Criterion - Games.

| Purity Criterion | Audio - M&N | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.8656 | 0.8054 | 0.8344 | 0.7139 | 0.6159 | 0.6613 | 0.7776 |
| **c = 2** | 0.9100 | 0.7798 | 0.8399 | 0.7616 | 0.5279 | 0.6236 | 0.7753 |
| **c = 4** | 0.8750 | 0.8045 | 0.8383 | 0.7263 | 0.6094 | 0.6628 | 0.7814 |
| **c = all** | 0.7815 | 0.6744 | 0.7240 | 0.4333 | 0.3069 | 0.3593 | 0.6142 |

Table 30: Classification results on audio recurrences - Purity Criterion - Mag&News.

| Purity Criterion | Visual - M&N | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.2683 | 0.73333 | 0.39283 | 0.8750 | 0.9813 | 0.9251 | 0.8667 |
| **c = 2** | 0.4634 | 0.4524 | 0.4578 | 0.8967 | 0.8925 | 0.8946 | 0.8235 |
| **c = 4** | 0.4146 | 0.4146 | 0.4146 | 0.8878 | 0.8878 | 0.8878 | 0.8118 |
| **c = all** | 0.4634 | 0.1712 | 0.2500 | 0.8472 | 0.5701 | 0.6816 | 0.5529 |

Table 31: Classification results on visual recurrences - Purity Criterion - Mag&News.

### Classification results on test dataset - Entropy Criterion with complex attributes training

| Entropy Criterion | Audio - Games | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.9961 | 0.9700 | 0.9829 | 0.8889 | 0.4985 | 0.6388 | 0.9673 |
| **c = 2** | 0.9830 | 0.9794 | 0.9812 | 0.7066 | 0.6647 | 0.6850 | 0.9645 |
| **c = 4** | 0.9784 | 0.9809 | 0.9797 | 0.6638 | 0.6914 | 0.6773 | 0.9618 |
| **c = all** | 0.9587 | 0.9652 | 0.9620 | 0.3957 | 0.4392 | 0.4163 | 0.9286 |

Table 32: Classification results on audio recurrences - Entropy Criterion - Games.

| Entropy Criterion | Visual - Games | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.0000 | 0.0000 | 0.0000 | 0.8531 | 1.0000 | 0.9207 | 0.8531 |
| **c = 2** | 0.2581 | 0.4211 | 0.3200 | 0.8802 | 0.9389 | 0.9086 | 0.8389 |
| **c = 4** | 0.3226 | 0.4348 | 0.3704 | 0.8883 | 0.9278 | 0.9076 | 0.8389 |
| **c = all** | 0.1935 | 0.0822 | 0.1154 | 0.8188 | 0.6278 | 0.7107 | 0.5640 |

Table 33: Classification results on visual recurrences - Entropy Criterion - Games.

| Entropy Criterion | **Audio - M&N** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.9474 | 0.7709 | 0.8501 | 0.8333 | 0.4828 | 0.6114 | 0.7837 |
| **c = 2** | 0.9428 | 0.7745 | 0.8504 | 0.8250 | 0.4957 | 0.6193 | 0.7852 |
| **c = 4** | 0.9346 | 0.7805 | 0.8506 | 0.8114 | 0.5172 | 0.6317 | 0.7874 |
| **c = all** | 0.7815 | 0.6744 | 0.7240 | 0.4333 | 0.3069 | 0.3593 | 0.6142 |

Table 34: Classification results on audio recurrences - Entropy Criterion - Mag&News.

| Entropy Criterion | **Visual - M&N** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.0000 | 0.0000 | 0.0000 | 0.8392 | 1.0000 | 0.9126 | 0.8392 |
| **c = 2** | 0.2195 | 0.6429 | 0.3273 | 0.8672 | 0.9766 | 0.9187 | 0.8549 |
| **c = 4** | 0.4878 | 0.3279 | 0.3922 | 0.8918 | 0.8084 | 0.8480 | 0.7569 |
| **c = all** | 0.4634 | 0.1712 | 0.2500 | 0.8472 | 0.5701 | 0.6816 | 0.5529 |

Table 35: Classification results on visual recurrences - Entropy Criterion - Mag&News.

## Classification results on test dataset - Error minimization Criterion with complex attributes training

| Err. min. Criterion | **Audio - Games** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.9912 | 0.9761 | 0.9836 | 0.8095 | 0.6053 | 0.6927 | 0.9688 |
| **c = 2** | 0.9841 | 0.9780 | 0.9811 | 0.7129 | 0.6409 | 0.6750 | 0.9642 |
| **c = 4** | 0.9691 | 0.9775 | 0.9733 | 0.5599 | 0.6380 | 0.5964 | 0.9499 |
| **c = all** | 0.9872 | 0.9765 | 0.9818 | 0.7473 | 0.6142 | 0.6743 | 0.9656 |

Table 36: Classification results on audio recurrences - Err. min. Criterion - Games.

| Err. min. Criterion | **Visual - Games** | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.6774 | 0.5385 | 0.6000 | 0.9419 | 0.9000 | 0.9205 | 0.8673 |
| **c = 2** | 0.3871 | 0.5714 | 0.4615 | 0.9000 | 0.9500 | 0.9243 | 0.8673 |
| **c = 4** | 0.0000 | 0.0000 | 0.0000 | 0.8531 | 1.0000 | 0.9207 | 0.8531 |
| **c = all** | 0.3871 | 0.5714 | 0.4615 | 0.9000 | 0.9500 | 0.9243 | 0.8673 |

Table 37: Classification results on visual recurrences - Err. min. Criterion - Games.

| Err. min. Criterion | Audio - M&N | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.9474 | 0.7709 | 0.8501 | 0.8333 | 0.4828 | 0.6114 | 0.7837 |
| **c = 2** | 0.8879 | 0.7966 | 0.8398 | 0.7391 | 0.5837 | 0.6523 | 0.7806 |
| **c = 4** | 0.9334 | 0.7765 | 0.8477 | 0.8055 | 0.5064 | 0.6219 | 0.7829 |
| **c = all** | 0.9486 | 0.7689 | 0.8494 | 0.8346 | 0.4764 | 0.6066 | 0.7821 |

TABLE 38: Classification results on audio recurrences - Err. min. Criterion - Mag&News.

| Err. min. Criterion | Visual - M&N | | | | | | |
|---|---|---|---|---|---|---|---|
| | P0 | R0 | F0 | P1 | R1 | F1 | Acc |
| **single att.** | 0.1951 | 0.8889 | 0.3200 | 0.8659 | 0.9953 | 0.9261 | 0.8667 |
| **c = 2** | 0.2927 | 0.5455 | 0.3810 | 0.8755 | 0.9533 | 0.9128 | 0.8471 |
| **c = 4** | 0.2683 | 0.6471 | 0.3793 | 0.8739 | 0.9720 | 0.9204 | 0.8588 |
| **c = all** | 0.3171 | 0.5000 | 0.3881 | 0.8777 | 0.9393 | 0.9074 | 0.8392 |

TABLE 39: Classification results on visual recurrences - Err. min. Criterion - Mag&News.

Annex D

## Example of case when none of the attributes, used alone could bring an improvement

Node 0: size 1222 Root
724 samples of category 0, 2.172, ratio=0.592471
498 samples of category 1, 498, ratio=0.407529
Node cost = 2.172

**Trying attribute 0:**
    for value 0, 668 samples of category 0
    for value 0, 272 samples of category 1
    for value 1, 56 samples of category 0
    for value 1, 226 samples of category 1
now computing purity/entropy/cost
    for value 0, purity = 2.004
    for value 1, purity = 0.168
cost for attribute 0 = 2.172
best attribute: 0 with purity 2.172
**Trying attribute 1:**
    for value 0, 578 samples of category 0
    for value 0, 340 samples of category 1
    for value 1, 146 samples of category 0
    for value 1, 158 samples of category 1
now computing purity/entropy/cost
    for value 0, purity = 1.734
    for value 1, purity = 0.438
cost for attribute 1 = 2.172
best attribute: 0 with purity 2.172
**Trying attribute 2:**
    for value 0, 599 samples of category 0
    for value 0, 387 samples of category 1
    for value 1, 125 samples of category 0
    for value 1, 111 samples of category 1
now computing purity/entropy/cost
    for value 0, purity = 1.797
    for value 1, purity = 0.375
cost for attribute 2 = 2.172
best attribute: 0 with purity 2.172
**Trying attribute 3:**
    for value 0, 627 samples of category 0
    for value 0, 352 samples of category 1

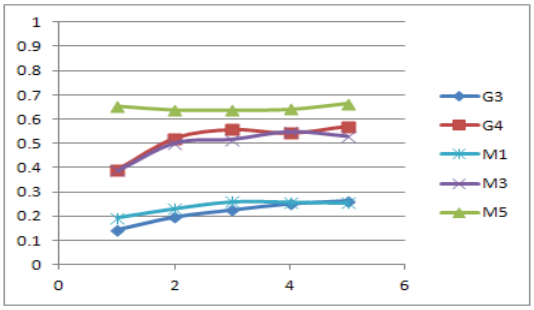
for value 1, 97 samples of category 0
for value 1, 146 samples of category 1
now computing purity/entropy/cost
for value 0, purity = 1.881
for value 1, purity = 0.291
cost for attribute 3 = 2.172
best attribute: 0 with purity 2.172
**Trying attribute 4:**
for value 0, 448 samples of category 0
for value 0, 269 samples of category 1
for value 1, 276 samples of category 0
for value 1, 229 samples of category 1
now computing purity/entropy/cost
for value 0, purity = 1.344
for value 1, purity = 0.828
cost for attribute 4 = 2.172
best attribute: 0 with purity 2.172
**Trying attribute 5:**
for value 0, 218 samples of category 0
for value 0, 329 samples of category 1
for value 1, 506 samples of category 0
for value 1, 169 samples of category 1
now computing purity/entropy/cost
for value 0, purity = 0.654
for value 1, purity = 1.518
cost for attribute 5 = 2.172
best attribute: 0 with purity 2.172
**Trying attribute 6:**
for value 0, 410 samples of category 0
for value 0, 424 samples of category 1
for value 1, 314 samples of category 0
for value 1, 74 samples of category 1
now computing purity/entropy/cost
for value 0, purity = 1.23
for value 1, purity = 0.942
cost for attribute 6 = 2.172
best attribute: 0 with purity 2.172
**Trying attribute 7:**
for value 0, 417 samples of category 0
for value 0, 423 samples of category 1
for value 1, 307 samples of category 0
for value 1, 75 samples of category 1
now computing purity/entropy/cost
for value 0, purity = 1.251
for value 1, purity = 0.921
cost for attribute 7 = 2.172

best attribute: 0 with purity 2.172
**Trying attribute 8:**
for value 0, 454 samples of category 0
for value 0, 377 samples of category 1
for value 1, 270 samples of category 0
for value 1, 121 samples of category 1
now computing purity/entropy/cost
for value 0, purity = 1.362
for value 1, purity = 0.81
cost for attribute 8 = 2.172
best attribute: 0 with purity 2.172
**Trying attribute 9:**
for value 0, 590 samples of category 0
for value 0, 362 samples of category 1
for value 1, 134 samples of category 0
for value 1, 136 samples of category 1
now computing purity/entropy/cost
for value 0, purity = 1.77
for value 1, purity = 0.402
cost for attribute 9 = 2.172
best attribute: 0 with purity 2.172
**Trying attribute 10:**
for value 0, 627 samples of category 0
for value 0, 426 samples of category 1
for value 1, 97 samples of category 0
for value 1, 72 samples of category 1
now computing purity/entropy/cost
for value 0, purity = 1.881
for value 1, purity = 0.291
cost for attribute 10 = 2.172
best attribute: 0 with purity 2.172
**Trying attribute 11:**
for value 0, 646 samples of category 0
for value 0, 428 samples of category 1
for value 1, 78 samples of category 0
for value 1, 70 samples of category 1
now computing purity/entropy/cost
for value 0, purity = 1.938
for value 1, purity = 0.234
cost for attribute 11 = 2.172
best attribute: 0 with purity 2.172
**Trying attribute 12:**
for value 0, 657 samples of category 0
for value 0, 436 samples of category 1
for value 1, 67 samples of category 0
for value 1, 62 samples of category 1

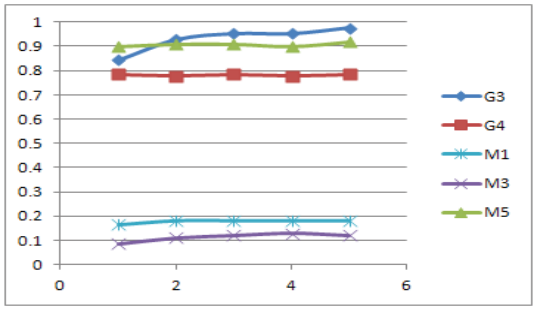
```
now computing purity/entropy/cost
    for value 0, purity = 1.971
    for value 1, purity = 0.201
cost for attribute 12 = 2.172
best attribute: 0 with purity 2.172
```

Annex E

# Impact of the number of episodes used for the detection of visual and audio recurrences

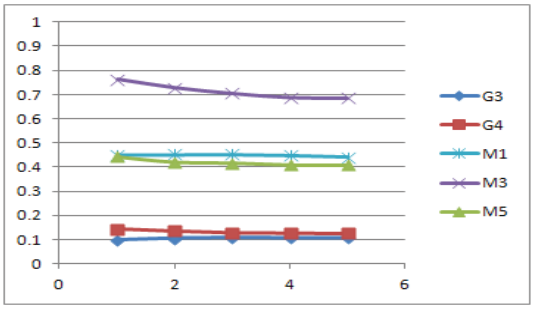



(a) (Precision)

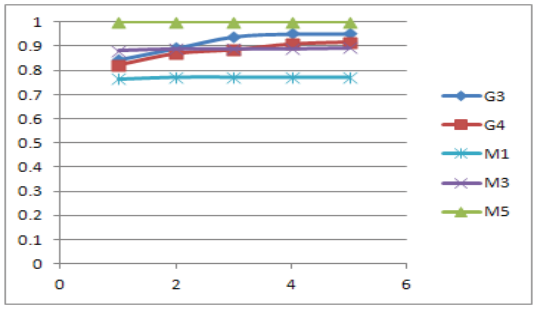



(b) (Recall)

TABLE 40: Impact of the number of episodes for the detection of visual recurrences.




(a) (Precision)




(b) (Recall)

TABLE 41: Impact of the number of episodes for the detection of audio recurrences.

Annex F

## Examples of false alarms among the visual recurrences

The figures below represent examples of images from visual recurrences that correspond to false alarms. This means that even though their content is very similar or even identical, they do not correspond to separators. Figures 14, 15, 16 represent images selected from intra episode recurrences. These generally correspond to front view of the anchor or of the competitiors, long views of the set, or, for the magazines and news, scenes presented in the summary of the show and replayed later when the concerned subject is detailed in a longer report.

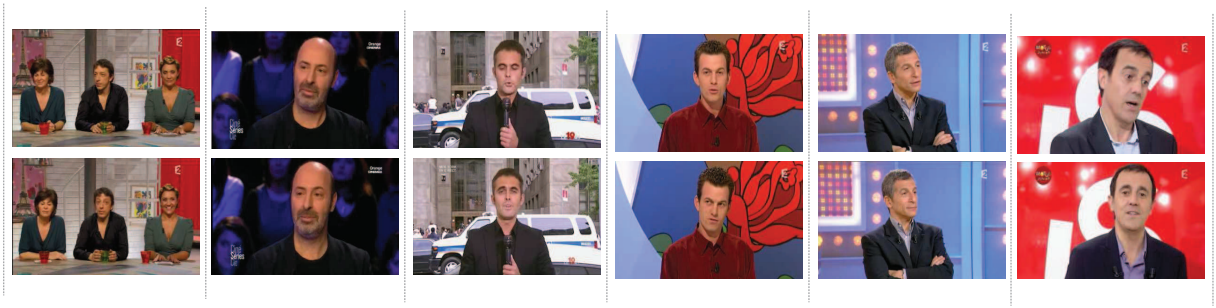


FIGURE 14: Examples of intra episode false alarms presenting the anchors in TV games and magazines.


FIGURE 15: Examples of intra episode false alarms presenting the competitors and the set in TV games.

Figures 17, 18, 19 correspond to images selected from inter episode recurrences. These present parts of different episodes of a same program that are visually similar (Figures 17), black or white frames that may appear during a program (Figure 18), or significant reports that are resumed over several days or weeks (Figure 19).

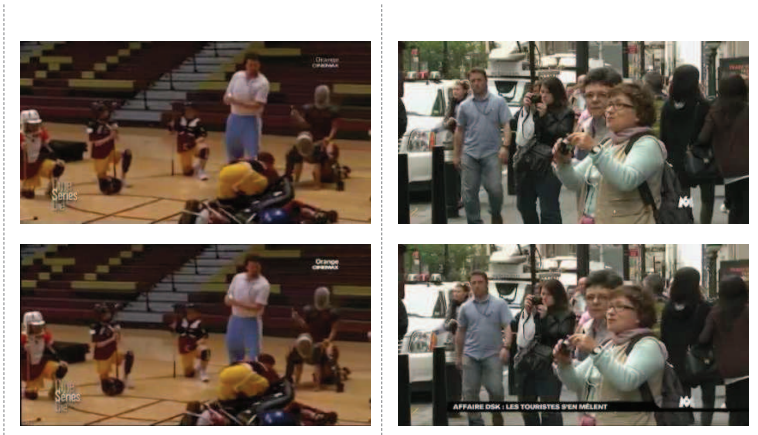



FIGURE 16: Examples of intra episode false alarms presenting sequences of a report replayed in the summary of a magazine.


FIGURE 17: Examples of inter episode false alarms presenting visually similar parts of a game show.

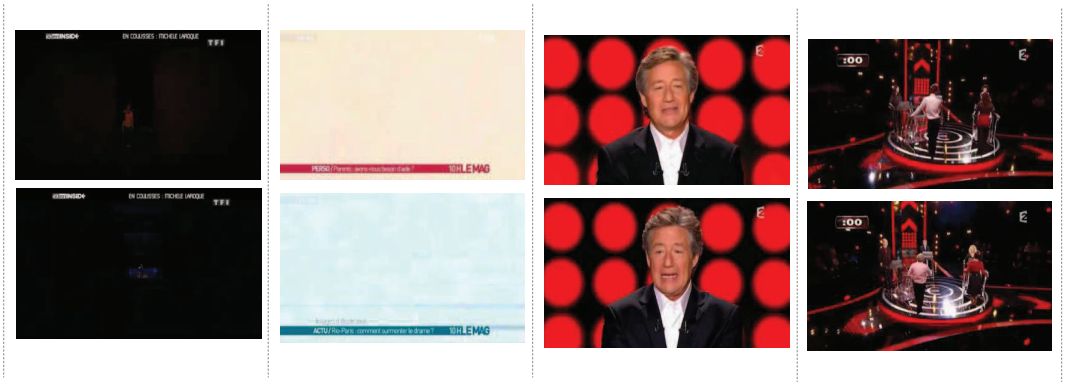


FIGURE 18: Examples of inter episode false alarms - black/white frames, images of the anchor or of the set from games and magazines.

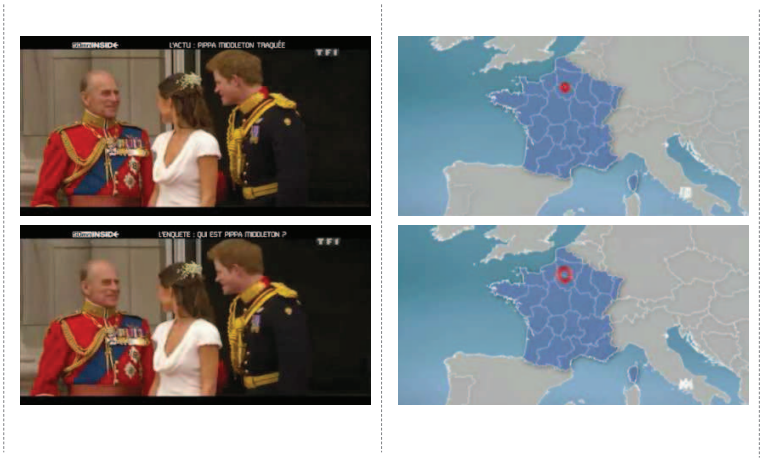



FIGURE 19: Examples of inter episode false alarms presenting reports that are rerun in different episodes of magazines or news.

Annex G

## Author's references

This thesis has been written based on the following work submitted to international specialized scientific community.

**Book chapter**:

A. E. Abduraman, S.-A. Berrani, B. Merialdo. TV Program Structuring Techniques: A Review. In TV Content Analysis: Techniques and Applications, Y. Kompatsiaris et al. (eds.). RC Press, Taylor & Francis Group. ISBN-13: 978-1439855607. March 2012.

**International conferences**:

A. E. Abduraman, S.-A. Berrani, B. Merialdo. Audio/Visual Recurrences and Decision Trees for Unsupervised TV Program Structuring. Proc of The 8th International Conference on Computer Vision Theory and Applications, Barcelona, Spain. February 2013.

A. E. Abduraman, S.-A. Berrani, B. Merialdo. Audio Recurrence Contribution to a Video-based TV Program Structuring Approach. Proc of The IEEE International Symposium on Multimedia, Dana Point, CA, USA.December 2011.

A. E. Abduraman, S.-A. Berrani, J.-B. Rault, O. Le Blouch. From audio Recurrences to TV Program Structuring. Proc. of the ACM International Workshop on Automated Media Analysis and Production for Novel TV Services, Scottsdale, AZ, USA. December 2011.

A. E. Abduraman, S.-A. Berrani, B. Merialdo. An Unsupervised Approach for Recurrent TV Program Structuring. Proc of The European Interactive TV Conference, Lisbon, Portugal. June 2011.

# Bibliography


[ABCB02] Jürgen Assfalg, Marco Bertini, Carlo Colombo, et Alberto Del Bimbo. Semantic annotation of sports videos. *IEEE MultiMedia*, 9(2):52 – 60, April 2002.

[ABM11] Alina Elma Abduraman, Sid-Ahmed Berrani, et Bernard Merialdo. An unsupervised approach for recurrent tv program structuring. *Proceedings of the 9th edition of the European Interactive TV Conference*, pages 123–126, Lisbon, Portugal, June-July 2011.

[ATK00] Yannis S. Avrithis, Nicolas Tsapatsoulis, et Stefanos D. Kollias. Broadcast news parsing using visual cues: a robust face detection approach. *Proceedings of the IEEE International Conference on Multimedia and Expo*, number 3, pages 1469 – 1472, New York , USA, August 2000.

[BAG03] Sid-Ahmed Berrani, Laurent Amsaleg, et Patrick Gros. Approximate searches: k-neighbors + precision. *Proceedings of the 12th ACM International Conference on Information and Knowledge Management*, pages 24–31, New Orleans, Louisiana, USA, November 2003.

[BBN06] Marco Bertini, Alberto Del Bimbo, et Walter Nunziati. Automatic detection of player's identity in soccer videos using faces and text cues. *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 663–666, Santa Barbara, CA, USA, October 2006.

[BBP01] M. Bertini, A. Del Bimbo, et P. Pala. Content-based indexing and retrieval of tv news. *Pattern Recognition Letters*, 22(5):503–516, April 2001.

[BC07] Olivier Le Blouch et Patrice Collen. Automatic syllable-based phoneme recognition using ester corpus. *Proceedings of the WSEAS International Conference on Signal Processing, Computational Geometry & Artificial Vision*, pages . 81–85, Athens, Greece, August 2007.

[BCR07] Michael Betser, Patrice Collen, et Jean-Bernard Rault. Audio identification using sinusoidal modeling, and application to jingle detection. *Proceedings of the 8th International Conference on Music Information Retrieval*, pages 1–4, Vienna, Austria, September 2007.

[BG11] Mathieu Ben et Guillaume Gravier. Unsupervised mining of audiovisually consistent segments in videos with application to structure analysis. *Proceedings of the IEEE International Conference on Multimedia and Exhibition*, pages 1–6, Barcelone, Espagne, July 2011.

[BHMS2] Félix Balado, Neil J. Hurley, Elizabeth P. McCarthy, et Guénolé C. M. Silvestre. Performance analysis of robust audio hashing. *IEEE Transactions on information forensics and security*, 2(2):254 – 266, June 2.

[BML08] Sid-Ahmed Berrani, Gael Manson, et Patrick Lechat. A non-supervised approach for repeated sequence detection in tv broadcast streams. *Image Communication*, 23(7):525–537, August 2008.

[BPPC12] Jenny Benois-Pineau, Frederic Precioso, et Matthieu Cord. *Visual indexing and retrieval*. Springer, 2012.

[CBG06] Sylvain Meignier Claude Barras, Xuan Zhu et Jean-Luc Gauvain. Multistage speaker diarization of broadcast news. *IEEE Transations On Audio, Speech and Language Processing*, 14(5):1505–1512, September 2006.

[CC06] Lekha Chaisorn et Tat-Seng Chua. Story boundary detection in news video using global rule induction technique. *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 2101–2104, Toronto, Ontario, Canada, July 2006.

[CC08] Padraig Cunningham et Matthieu Cord. *Machine Learning Techniques for Multimedia*. Springer, 2008.

[CCL02] Lekha Chaisorn, Tat-Seng Chua, et Chin-Hui Lee;. The segmentation of news video into story units. *Proceedings of the IEEE International Conference on Multimedia and Expo*, number 1, pages 73 – 76, Lusanne, Switzerland, August 2002.

[CCL03] Lekha Chaisorn, Tat-Seng Chua, et Chin-Hui Lee. A multi-modal approach to story segmentation for news video. *World Wide Web*, 6(2):187–208, 2003.

[Cha06] Wei Chai. Semantic segmentation and summarization of music. *Proceedings of the IEEE Signal Processing Magazine*, volume 23, pages 124–132, March 2006.

[CLZC03] Rui Cai, Lie Lu, Hong-Jiang Zhang, et Lian-Hong Cai. Highlight sound effects detection in audio stream. *Proceedings of the International Conference on Multimedia and Expo*, pages 37–40, Baltimore, Maryland, July 2003.

[CMPP08] Angelo Chianese, Vincenzo Moscato, Antonio Penta, et Antonio Picariello. Scene detection using visual and audio attention. *Proceedings of the ACM Int. Conf. on Ambi-Sys workshop on Ambient media delivery and interactive television*, Quebec, Canada, February 2008.

[DHG01] Richard O. Duda, Peter E. Hart, et David G.Stork. *Pattern Classification*. Wiley-Interscience, 2001.

[DXC+03] Ling-Yu Duan, Min Xu, Tat-Seng Chua, Qi Tian, et Chang-Sheng Xu. A mid-level representation framework for semantic sports video analysis. *Proceedings of the ACM International Conference on Multimedia*, pages 33 – 44, Berkeley, CA, USA, November 2003.

[ETM03] Ahmet Ekin, A. Murat Tekalp, et Rajiv Mehrotra. Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, 12(7):796–807, July 2003.

[EWIR01] S. Eickeler, F. Wallhoff, U. Iurgel, et G. Rigoll. Content based indexing of images and video using face detection and recognition methods. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 1505–1508, Salt Lake City, Utah, USA, May 2001.

[FMG07] SébastienRoux Franck Mamalet et Christophe Garcia. Real time video convolutional face finder on embedded platforms. *EURASIP Journal on Embedded Systems*, 2007(1), January 2007.

[FZF06] Yong Fang, Xiaofei Zhai, et Jingwang Fan. News video story segmentation. *Proceedings of the 12th International Multi-Media Modelling Conference*, pages 397–400, Beijing, China, January 2006.

[GCP12] David Gorisse, Matthieu Cord, et Frederic Precioso. Locality-sensitive hashing for chi2 distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):402–409, February 2012.

[GD04] Christophe Garcia et Manolis Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, November 2004.

[GFT98] B. Gunsel, A. Ferman, et A. Tekalp. Temporal video segmentation using unsupervised clustering and semantic object tracking. *Journal of Electronic Imaging*, 7(3):592–604, 1998.

[GSSD09] Abhinav. Gupta, Praveen Srinivasan, Jianbo Shi, et Larry S. Davis. Understanding videos constructing plots learning a visually grounded storyline model from annotated. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Miami, Florida, USA, June 2009.

[GT02] Xinbo Gao et Xiaoou Tang. Unsupervised video-shot segmentation and model-free anchorperson detection for news video story parsing. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(9):765 – 776, September 2002.

[Gué02] André Guéziec. Tracking pitches for broadcast television. *Computer*, 35(3):38 – 43, March 2002.

[GWND07] Naveen Goela, Kevin Wilson, Feng Niu, et Ajay Divakaran. An svm framework for genre-independent scene change detection. *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 532–535, Beijing, China, July 2007.

[Her06] C. Herley. Argos: automatically extracting repeating objects from multimedia streams. *Proceedings of the IEEE Transactions on Multimedia*, volume 8, pages 115–129, 2006.

[Hin71] David V. Hinkley. Inference about the change-point from cumulativesum tests. *Biometrika*, 58(3):509–523, December 1971.

[HKG+10] Raffay Hamid, Ram Krishan Kumary, Matthias Grundmannz, Kihwan Kimz, Irfan Essaz, et Jessica Hodgins. Player localization using multiple static cameras for sports visualization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 731 – 738, San Francisco, CA, June 2010.

[HKO01] J Haitsma, T Kalker, et J Oostveen. Robust audio hashing for content identification. *Proceedings of the International Workshop on Content-Based Multimedia Indexing*, Brescia, Italy, September 2001.

[HS95] A. G. Hauptmann et M. A. Smith. Text, speech and vision for video segmentation : The infomedia project. *Proceedings of the AAAI Fall Symposium, Computational Models for Integrating Language and Vision*, Cambridge, Massachusetts, USA, November 1995.

[Jac06] Arne Jacobs. Using self-similarity matrices for structure mining on news video. *Advances in Artificial Intelligence*, 3955:87–94, 2006.

[KCtK+02] Jae-Gon Kim, Hyun Sung Chang, Young tae Kim, Kyeongok Kang, Munchurl Kim, Jinwoong Kim, et Hyung-Myung Kim. Multimodal approach for summarizing and indexing news video. *ETRI Journal*, 24(1):1–11, 2002.

[KGS+10] Kihwan Kim, Matthias Grundmann, Ariel Shamir, Iain Matthews, et Jessica Hodgins and Irfan Essa. Motion fields to predict play evolution in dynamic sport scenes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 840 – 847, San Francisco, CA, June 2010.

[Kij03] E. Kijak. *Structuration multimodale des vidéos de sports par modèles stochastiques.* Thèse de doctorat, Université de Rennes 1, 2003.

[KLP03] E. Kijak, L.Oisel, et P.Gros. Hierarchical structure analysis of sport videos using hmms. *Proceedings of the International Conference on Image Processing*, volume 3, pages 1025–1028, September 2003.

[KLX03] S. S. Keerthi et C. J. Lin-Xie. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15(7):1667–1689, July 2003.

[KX08] Chien-Chuan Ko et Wen-Ming Xie. News video segmentation and categorization techniques for content-demand browsing. *Proceedings of the Congress on Image and Signal Processing*, pages 530 – 534, Sanya, Hainan, China, May 2008.

[LC00] Beth Logan et Stephen Chu. Music summarization using key phrases. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 749–752, Istanbul, Turkey, June 2000.

[LGS+00] Francis C. Li, Anoop Gupta, Elizabeth Sanocki, Li wei He, et Yong Rui. Browsing digital video. *Proceedings of the ACM Computer-Human Interaction*, pages 169 – 176, Hague, The Netherlands, 2000.

[LTW+10] Haojie Li, Jinhui Tang, Si Wu, Yongdong Zhang, et Shouxun Lin. Automatic detection and analysis of player action in moving background sports video sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(3):351 – 364, March 2010.

[LWZ04] Lie Lu, Muyuan Wang, et Hong-Jiang Zhang. Repeating pattern discovery and structure analysis from acoustic music data. *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 275 – 282, New York, NY, USA, October 2004.

[MGB09] Armando Muscariello, Guillaume Gravier, et Frédéric Bimbot. Audio keyword extraction by unsupervised word discovery. *Proceedings of the Conference of the International Speech Communication Association (Interspeech)*, pages 2843–2846, Brighton UK, 2009.

[MHG+10] Hemant Misra, Frank Hopfgartner, Anuj Goyal, P. Punitha, et Joemon M. Jose. Tv news story segmentation based on semantic coherence and content similarity. *Proceedings of the 16th International Multimedia Modeling Conference*, pages 347–357, Chongqing, China, January 2010.

[Mit97] Tom M. Mitchell. *Machine Learning.* McGraw-Hill Science/Engineering/Math;, 1997.

[MLLR99] Bernard Mérialdo, Kyung-Tak Lee, Dario Luparello, et Jeremie Roudaire. Automatic construction of personalized tv news programs. *Proceedings of the 7th ACM International Multimedia Conference*, pages 323–331, Orlando, Florida, USA, October 30-November 5 1999.

[MMM97] Andrew Merlino, Daryl Morey, et Mark Maybury. Broadcast news navigation using story segmentation. *Proceedings of the ACM international conference on Multimedia*, pages 381–391, Seattle, WA, USA, November 1997.

[NGG06] Xavier Naturel, Guillaume Gravier, et Patrick Gros. Fast structuring of large television streams using program guides. *Proceedings of the 4th International Workshop on Adaptive Multimedia Retrieval*, volume 4398, pages 223–232, Geneve, Suisse, July 2006.

[NK97] Yuichi Nakamura et Takeo Kanade. Semantic analysis for video contents extraction - spotting by association in news nideo. *Proceedings of the ACM international conference on Multimedia*, pages 393–401, Seattle, WA, USA, November 1997.

[OSA07] Paul Over, Alan F. Smeaton, , et George Awad. The trecvid 2007 bbc rushes summarization evaluation pilot. *Proceedings of the International workshop on TRECVID video summarization*, Augsburg, Germany, September 2007.

[OSA08] Paul Over, Alan F. Smeaton, , et George Awad. The trecvid 2008 bbc rushes summarization evaluation. *Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 1–20, Vancouver, BC, Canada, October 2008.

[PAO04] Julien Pinquier et Regine Andre-Obrecht. Jingle detection and identification in audio documents. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 329–332, Montreal, Quebec, Canada, May 2004.

[PG08] Alex S. Park et James R. Glass. Unsupervised pattern discovery in speech. *IEEE Transaction on Acoustic, Speech and Language Processing*, 16(1):186–197, 2008.

[PGGM04] Kok Meng Pua, John M. Gauch, Susan E. Gauch, et Jedrzej Z. Miadowicz. Real time repeated video sequence identification. *Computer Vision and Image Understanding*, 93(3):310–327, March 2004.

[PKY06] Mansoo Park, Hoi-Rin Kim, et Seung Hyun Yang. Frequency-temporal filtering for a robust audio fingerprinting scheme in real-noise environments. *Electroncs and Telecommunications Research Institute*, 28(4):509–512, August 2006.

[Pla98] John C. Platt. Machines using sequential minimal optimization. *B. Schoelkopf, C. Burges, et A. Smola, editors,* Advances in Kernel Methods - Support Vector Learning. 1998.

[PLE01] Silvia Pfeiffer, Rainer Lienhart, et Wolfgang Efflsberg. Scene determination based on video and audio features. *Multimedia Tools and Applications*, 15(1):59–81, September 2001.

[Pol07] J-P. Poli. *Structuration automatique de flux televisuels.* Thèse de doctorat, Universite Paul-Cezanne-Aix-Marseille III, 2007.

[Pol08] Jean-Philippe Poli. An automatic television stream structuring system for television archives holders. *Multimedia Systems*, 14(5):255–275, January 2008.

[PWR03] Marcus J. Pickering, Lawrence Wong, et Stefan M. Rüger. Anses: summarisation of news video. *Proceedings of the Proceedings of the 2nd international conference on Image and video retrieval*, pages 425–434, Urbana, IL, USA, July 2003.

[PWY04] Rui Cui Peng Wung et Shi-Qiung Yung. Contextual browsing for highlights in sports video. *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1951–1954, Taipei, Taiwan, June 2004.

[RGA00] Yong Rui, Anoop Gupta, et Alex Acero. Automatically extracting highlights for tv baseball programs. *Proceedings of the ACM International Conference on Multimedia*, pages 105 – 115, New York, USA, October 2000.

[RS05] Zeeshan Rasheed et Mubarak Shah. Detection and representation of scenes in videos. *IEEE transactions on multimedia ISSN 1520-9210*, 7(6):1097–1105, December 2005.

[SDV07] Veronique Stouten, Kris Demuynck, et Hugo Van. Discovering phone patterns in spoken utterances by non-negative matrix factorisation. *IEEE Signal Processing Letters*, 15:131–134, 2007.

[Sil86] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[SMK+09] Panagiotis Sidiropoulos, Vasileios Mezaris, Ioannis Kompatsiaris, Hugo Meinedo, et Isabel Trancoso. Multi-modal scene segmentation using scene transition graphs. *Proceedings of the 17th ACM International Conference on Multimedia*, pages 665–668, Beijing, China, October 2009.

[SS97] Eric Scheirer et Malcolm Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 1331–1334, Munich, Germany, April 1997.

[TA09] Koji Yamamoto Shunsuke Takayama et Hisashi Aoki. Semantic segmentation of tv programs using corner-subtitles. *Proceedings of the IEEE 13th International Symposium on Consumer Electronics*, pages 205 – 208, Kyoto, Japan, May 2009.

[TC10] Dian W. Tjondronegoro et Yi-Ping Phoebe Chen. Knowledge-discounted event detection in sports video. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 40(5):1009–1024, September 2010.

[tLL03] Hsuan tien Lin et Chih-Jen Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. Technical report, 2003.

[TV06] Ba Tu Truong et Svetha Venkatesh. Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 3(1):1–37, February 2006.

[TZ04] Wallapak Tavanapong et Junyu Zhou. Shot clustering techniques for story browsing. *IEEE Transactions on Multimedia*, 6(4):517 – 527, August 2004.

[VRB00] E. Veneau, R. Ronfard, et P. Bouthemy. From video shot clustering to sequence segmentation. *Proceedings of the 15th International Conference on Pattern Recognition*, volume 4, pages 254–257, Barcelona, Spain, September 2000.

[WC03] Jihua Wang et Tat-Seng Chua. A cinematic-based framework for scene boundary detection in video. *The Visual Computer*, 19(5):329–341, 2003.

[WDL+08] Jinqiao Wang, Lingyu Duan, Qingshan Liu, Hanqing Lu, et Jesse S. Jin. A multimodal scheme for program segmentation and representation in broadcast video streams. *IEEE Transactions on Multimedia*, 10(3):393 – 408, April 2008.

[wHcCjL10] Chih wei Hsu, Chih chung Chang, et Chih jen Lin. A practical guide to support vector classification, April 2010.

[XMZY05] Gu Xu, Yu-Fei Ma, Hong-Jiang Zhang, et Shi-Qiang Yang. An hmm-based framework for video semantic analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(11):1422 – 1433, November 2005.

[XXC+01] Peng Xu, Lexing Xie, Shih-Fu Chang, A. Divakaran, A. Vetro, et Huifang Sun;. Algorithms and system for segmentation and structure analysis in soccer video. *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 721 – 724, Tokyo, Japan, August 2001.

[XXC+04] Lexing Xie, Peng Xu, Shih-Fu Chang, Ajay Divakaran, et Huifang Sun. Structure analysis of soccer video with domain knowledge and hidden markov models. *Pattern Recognition Letters*, 25(7):767 – 775, May 2004.

[XZT+06] Ziyou Xiong, Xiang Sean Zhou, Qi Tian, Yong Rui, et Thomas S. Huangm. Semantic retrieval of video - review of research on video retrieval in meetings, movies and broadcast news, and sports. *IEEE Signal Processing Magazine*, 23(2):18 – 27, March 2006.

[YL95] M. M. Yeung et B. Liu. Efficient matching and clustering of video shots. *Proceedings of the International Conference on Image Processing*, volume 1, pages 338–341, Washington D.C., USA, October 1995.

[YLL09] Xinguo Yu, Liyuan Li, et Hon Wai Leong. Interactive broadcast services for live soccer video based on instant semantics acquisition. *Visual Communication and Image Representation*, 20(2):117–130, February 2009.

[YTX07] Xian-Feng Yang, Qi Tian, et Ping Xue. Efficient short video repeat identification with application to news video structure analysis. *IEEE Transactions on Multimedia*, 9(3):600–609, April 2007.

[YWX+07] Jinhui Yuan, Huiyi Wang, Lan Xiao, Wujie Zheng, Jianmin Li, Fuzong Lin, et Bo Zhang. A formal study of shot boundary detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(2):168 – 186, February 2007.

[YY96] M.M. Yeung et Boon-Lock Yeo. Time-constrained clustering for segmentation of video into storyunits. *Proceedings of the 13th International Conference on Pattern Recognition*, volume 3, pages 375–380, Vienna, Austria, August 1996.

[ZGST94] HongJiang Zhang, Yihong Gong, S.W. Smoliar, et Shuang Yeo Tan. Automatic parsing of news video. *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 45 – 54, Boston, Massachusetts, May 1994.

[ZLCS04] Dong-Qing Zhang, Ching-Yung Lin, Shi-Fu Chang, et John R. Smith. Semantic video clustering across sources using bipartite spectral clustering. *Proceedings of the IEEE International Conference on multimedia and expo ( ICME )*, pages 117– 120, June 2004.

[ZS06] Yun Zhai et Mubarak Shah. Video scene segmentation using markov chain monte carlo. *IEEE Transactions on Multimedia*, 8(4):686 – 697, August 2006.

[ZWW+07] Yanjun Zhao, Tao Wang, Peng Wang, Wei Hu, Yangzhou Du, Yimin Zhang, et Guangyou Xu. Scene segmentation and categorization using ncuts. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 343–348, Minneapolis,Minnesota,USA, June 2007.

# List of Figures

# List of Tables

# List of Algorithms

## Abstract - Unsupervised TV Program Structuring

TV programs have an underlying structure that is lost when these are broadcasted. The linear mode is the only available reading mode when viewing programs recorded using a Personal Video Recorder or through a TV-on-Demand service. The fast-forward/backward functions are the only available tools for browsing. In this context, program structuring becomes important in order to provide users with novel and useful browsing features. In addition to advanced browsing features, TV program structuring can also be used for summarization, indexing and querying, archiving, etc.

This thesis addresses the problem of unsupervised TV program structuring. The idea is to automatically recover the original structure of the program by finding the start time of each part composing it. The proposed approach is completely unsupervised and addresses a large category of programs like TV games, magazines, news... It is based on the detection of "separators" which are short audio/visual sequences that delimit the different parts of a program. To do so, audio and visual recurrences are first detected from a set of episodes of a same program. In order to extract the separators, the recurrences are then classified using decision trees. These are built based on attributes issued from techniques like applause detection, scenes segmentation, face and speaker detection and clustering.

**Keywords**: TV Program Structuring, Non-linear Browsing, Audio/Video Recurrence Detection, Decision Tree Classification.

## Résumé - Structuration intra-programme de contenus TV

Les programmes TV possède une structure qui, en général, est perdue quand les programmes sont diffusés. Les programmes qui ont été enregistrés via un enregistreur vidéo personnel ou disponibles via des services comme la TV à la demande, ne peuvent être visionnés que d'une façon linéaire. La navigation y est réalisée en utilisant les fonctions basiques d'avance/retour rapide. Dans ce contexte, la structuration automatique de programmes TV apporte une solution originale. En retrouvant la structure d'origine du programme, elle permet d'offrir aux utilisateurs des outils de navigation originaux. Elle peut également servir pour d'autres applications comme la construction des résumés vidéo, l'indexation et la recherche...

Cette thèse s'intéresse ainsi à la structuration automatique des programmes TV. L'objectif est de retrouver automatiquement la structure d'origine d'un programme en déterminant le début et la fin de chaque partie qui le compose. L'approche proposée est complètement non-supervisée et adresse une large catégorie de programmes TV comme les jeux, les magazines, les journaux TV... Cette approche exploite les «séparateurs » qui sont de séquences courtes insérées dans les programmes pour en délimiter les différentes parties. Pour cela, une détection des récurrences audio et visuelles est réalisée sur un ensemble d'épisodes du même programme. Ces récurrences sont ensuite classées à l'aide d'arbres de décision pour en extraire les séparateurs. Les attributs utilisés pour la construction des arbres de décision porte sur la détection des applaudissements, la segmentation en scènes, la détection et le clustering des visages et des locuteurs.

**Mots clés**: Structuration de programmes TV, Navigation non-linéaire, Détection des récurrences audio/visuelles, Classification basée arbres de décision.

# Structuration intra-programme de contenus TV
# - Résumé long -

# Sommaire

# Introduction

En raison du nombre élevé de chaînes de télévision, la quantité d'émissions de télévision diffusées a considérablement augmenté. Par conséquent, l'organisation des données et des outils pour manipuler efficacement et gérer les programmes de télévision sont nécessaires. Les chaînes de télévision diffusent le flux vidéo d'une façon linéaire. Cette linéarité génère des contraintes temporelles dans la visualisation du contenu. Des services comme la *télévision de rattrapage* et la *PVR (enregistreur vidéo personnel)* enlèvent ces contraintes et permettent aux utilisateurs de regarder des émissions de télévision préalablement diffusées.

A partir du flux TV, un grand nombre de programmes de télévision peuvent être extraits, stockés, indexés et préparés pour une utilisation ultérieure. Une étape d'indexation préalable du contenu est toutefois encore nécessaire. Après avoir choisi un programme, l'utilisateur pourrait vouloir obtenir un aperçu du programme avant de regarder. Il / elle peut aussi vouloir accéder directement à une partie spécifique du programme, trouver un certain moment d'intérêt ou sauter une partie du programme et passer à la suivante ou même directement à la partie finale. Ces caractéristiques représentent une alternative pour les fonctionnes basiques d'avance/retour rapides.

Pour permettre ces fonctionnalités, après avoir été extrait, chaque programme de télévision doit être structuré. C'est à dire, sa structure originale doit être récupérée et tous les moments possibles d'intérêt doivent être précisément étiquetés. Une option similaire existe pour les DVD et offre à l'utilisateur un résumé et la possibilité de voir un moment donné du film. Mais pour obtenir une telle représentation, un observateur humain est nécessaire pour regarder la vidéo en entier et pour localiser les endroits des moments importantes. Évidemment, cela pourrait se faire aussi dans le cas de programmes de télévision, mais ces étapes de prétraitement manuel sont très coûteuses, en particulier lorsqu'elles traitent une grande quantité d'émissions de télévision, comme c'est le cas dans les services réels de télévision. Le défi élevé est donc de développer des outils automatiques basés contenu pour la structuration des programmes de télévision. Ces outils permettront aux utilisateurs de bénéficier de la structure des programmes et de regarder seulement les parties des programmes susceptibles de les intéresser.

Dans ce contexte, la structuration des programmes devient essentielle afin de fournir aux utilisateurs des outils de navigation nouveaux et utiles. Fondamentalement, l'objectif de structuration du programme est de récupérer la structure d'origine du programme. En d'autres termes, l'objectif de structuration est de détecter les instants de début et de fin de chaque partie composante du programme. Cela permet aux utilisateurs une fonctionnalité avancée d'accès rapide et non linéaire qui pourrait être une alternative à la seule navigation possible actuellement qui est l'avance rapide. En plus de la navigation, la structure pourrait également être utilisée pour la construction de résumés vidéo. L'utilisation de la

structure permet la construction de résumés équilibrés où chaque partie du programme est représentée par rapport à son importance. La structuration des programmes TV peut également servir pour des applications comme l'indexation et la recherche, l'archivage, la mesure d'audience intra-programmes. Les services interactifs pourraient également bénéficier, par exemple, en fournissant des informations ou des caractéristiques spécifiques en fonction de la partie actuellement diffusée.

Notre travail de recherche porte sur la segmentation automatique des programmes TV, dont le but est de retrouver automatiquement la structure d'origine d'un programme et ainsi de le segmenter en ses parties principales. Nous proposons une approche largement non-supervisée pour la structuration d'un large catégorie des programmes TV. Comme il est difficile de trouver une méthode générique qui couvre tous les types de programmes de télévision existants, nous nous sommes concentrés sur les programmes TV "*récurrents*".

Un programme de télévision récurrent est un programme composé de plusieurs "*épisodes*" qui sont périodiquement diffusés (par exemple quotidiennement, hebdomadairement, mensuellement ...). Des exemples de ce type de programmes sont les jeux TV, les émissions de divertissement, les magazines d'information, les journaux TV. Le choix de ce type de programmes est motivé à la fois par leur intérêt applicatif, car ils représentent un pourcentage important des émissions diffusées sur une chaine de télévision généraliste. De plus, il s'agit de programmes qui possèdent naturellement une structure bien définie avec des parties bien distinctes. Ces parties sont délimitées par des séquences audio/vidéo caractéristiques que nous appelons «*séparateurs*». Les séparateurs sont des séquences vidéo courtes, insérées entre les différents parties d'un programme et qui peuvent être répétées entre et / ou à l'intérieur des épisodes d'un même programme. L'idée de base de notre étude est de retrouver ces séparateurs d'une façon largement non supervisée et de les utiliser ensuite pour la structuration. La méthode ne nécessite aucune autre connaissance préalable sur la structure d'un programme ou sur le nombre de parties du programme.

## Contributions de cette thèse

Les principales contributions de cette thèse sont les suivantes : Tout d'abord, nous proposons une approche originale pour la structuration des programmes TV, qui n'est pas spécifique à un certain type de programme et qui aborde une large catégorie de programmes TV comme les programmes TV récurrents.

Deuxièmement, l'approche que nous proposons pour la structuration est largement non-supervisée. Il n'y a besoin d'aucune connaissance préalable sur les programmes ou sur le nombre de parties du programme.

Troisièmement, l'approche proposée exploite le contenu visuel et le contenu audio des émissions de télévision. Il est basé sur des techniques de classification qui filtrent et sélectionnent les séparateurs parmi un ensemble de récurrences visuelles et audio détectées.

Dernièrement, les expériences sont réalisées sur des émissions réelles de la télévision, afin de valider l'idée sur laquelle l'approche proposée est basée et pour tester les algorithmes de classification proposés.

# Approches existantes

Les techniques existantes pour la structuration intra-programme peuvent être classifiés en deux catégories : les méthodes spécifiques et les méthodes génériques.

**Les méthodes spécifiques**

Les méthodes spécifiques exploitent la connaissance a priori du domaine afin de construire un modèle de la structure de la vidéo analysée. Elles peuvent être appliquées seulement pour certains types de programmes comme les journaux télévisés, les programmes sportifs, les séries, les publicités, etc...

Elles font usage de la connaissance a priori du type de l'émission de télévision analysée afin d'en extraire les données pertinentes et construire son modèle de structure. Ces méthodes sont supervisées car ils nécessitent généralement la création préalable et l'annotation manuelle d'un ensemble d'apprentissage utilisé pour apprendre la structure.

Une classe de programmes télévisés souvent analysés par des méthodes spécifiques sont les **programmes sportifs**. Ceux-ci ont une structure très bien définie. Les règles du jeu fournissent de la connaissance a priori qui peut être utilisée ensuite pour fournir des contraintes sur l'apparition d'événements ou de la succession de ces événements. Ces contraintes sont très utiles pour améliorer la précision de la détection d'événements et leurs classification. Nous distinguons deux niveaux dans l'analyse vidéo des programmes sportifs : segment et événement. Dans le premier cas, l'objectif est de segmenter la vidéo en segments narratifs  emph jeux et  emph non-jeux par une analyse bas niveau de la vidéo [XXC+04]. Le deuxième, suppose une analyse de la vidéo de haut niveau et son objectif est d'identifier les moments intéressants (événements importants) de la vidéo. Les deux sont prédéterminés par le type de sport. Par exemple, un événement dans le cas du football pourrait être la détection d'un but tandis que pour le tennis ce sera les points de match. Par conséquent, afin d'atteindre ces objectifs, l'utilisation de la connaissance a priori devient nécessaire. Cette connaissance a priori peut être lié au type du sport analysé (i.e. surface de jeu, nombre de joueurs, les règles de jeu), mais aussi les règles de production du programme vidéo (i.e. slowmotion replay, l'emplacement et la couverture de la caméra, le mouvement de la caméra, le texte superposé [WDL+08]). Les méthodes utilisées impliquent en général le suivi des objets spécifiques (joueur [KGS+10, HKG+10], balle [Gué02]), détection des évènements comme penaltys ou carton jeune/rouge ( [ETM03]), l'identification des sons spécifiques (applaudissements, sifflet [XZT+06], le son de l'eau dans les vidéos de plongée [PWY04]) mais aussi l'analyse du texte par exemple pour détecter le score sur le tableau de score [WDL+08]. Les techniques abordées utilisent des règles heuristiques ou même des méthodes d'apprentissage automatique comme les HMMs ( [KLP03, PWY04, XMZY05, XZT+06, LTW+10]), réseaux des neurones ( [ABCB02]), graphs AND-OR [GSSD09].

Une autre classe de programmes appropriés pour les approches spécifiques sont les

**journaux de télévision**, qui ont aussi une structure très claire. Ils sont réalisés en utilisant presque les mêmes règles de production. Ils consistent généralement en une succession de reportages et de plans présentateur. Les méthodes spécifiques analysent la structure temporelle (plateau, reportages, publicité, prévisions météo, etc) et la structure spatiale (images du présentateur, logos, etc.). La plupart des travaux se fondent sur la recherche des plans présentateur pour en déduire ensuite les séquences de plans représentant les reportages. Les plans présentateur ont beaucoup de caractéristiques qui facilitent leur détection. Certain approches utilisent des mesures de similarité, des méthodes de détection de visages [KX08, ATK00] et des modelés construits [ZGST94, GFT98] en se basant sur des caractéristiques comme : le même présentateur apparaît au cours de la même émission, les images de fond restent de fois inchangés pendant toute la prise de vue, les plans sont généralement filmés avec une caméra statique de sorte que le présentateur est toujours situé au même endroit de l'image. En outre, chaque chaîne de télévision dispose d'objets sémantiques représentatifs, comme les logos qui sont affichés uniquement pendant le programme.

Afin d'éviter l'utilisation d'un modèle pour la reconnaissance des plans présentateur, une autre propriété de ces plans est utilisée c'est-à-dire leurs apparition récurrente pendant la diffusion du programme. Bertini *et al.* [BBP01] se base sur l'idée que les plans présentateurs sont répétés à des intervalles de longueur variable et leur contenu est très similaire. La fréquence des plans présentateurs et leur similitude est également utilisée dans [Pol07] où une méthode de classification permet de regrouper les plans similaires dans des groupes, le plus petit représentant le groupe des plans présentateurs. D'autres approches utilisent les SVMs [MHG$^+$10] ou les HMMS [EWIR01]. Une fois les plans classés, une tâche plus difficile est de segmenter la vidéo en segments plus cohérentes. Cela implique de trouver les limites de chaque segment cohérent qui se succède dans le flux vidéo. Pour cela différentes approches sont utilisées : parmi eux, des modèles temporelles de la structure [GT02, ZLCS04], les métadonnées [KX08], des règles heuristiques [GFT98], de la programmation dynamique [NK97], des automates fini [KCtK$^+$02, MMM97], les HMMs [CCL03, FZF06] ou même des techniques en se basant sur l'information textuelle [MHG$^+$10].

**Les approches génériques**

Les approches génériques essayent de trouver une approche universelle pour la structuration de vidéos, indépendamment de leur type et sont basées uniquement sur leurs caractéristiques de contenu. Celles-ci tentent de structurer la vidéo d'une manière non supervisée, sans utiliser la connaissance a priori. En raison du fait qu'elles ne reposent pas sur un modèle spécifique, elles sont applicables à une large catégorie de vidéos.

Dans cette catégorie, la littérature reflète souvent l'importance des ***scènes*** comme éléments structurels d'une vidéo et dépeint les techniques correspondantes. Une scène est généralement composée d'un nombre réduit de plans tous liés au même sujet, un événement en cours ou un thème [WDL$^+$08, ZS06]. Toutefois, la définition d'une scène est très ambiguë et subjective car elle dépend de la compréhension de chacun. Dans la littérature, les scènes sont également nommées «paragraphes vidéo» [HS95], «vidéo segments» [ZS06, VRB00], «story units» [SMK$^+$09, YY96, YL95] ou «chapitres» [TA09]. En général la segmentation en scènes se fait soit en regroupant les plans en scènes en fonction de leurs similitudes, ou en soulignant les différences entre les scènes. Même si le but est le même, les différences apparaissent dans le choix des paramètres et leurs seuils. Le défi

consiste donc à trouver l'ensemble approprié de fonctionnalités qui mèneraient à une identification correcte des scènes dans une vidéo. Les approches utilisent en général la similarité visuelle des plans en considérant ou pas des contraintes temporelles [WDL+08], des Graphs de Transition [RS05], des Modèles Markov [ZS06]. La similarité multimodale est aussi exploitée en [PLE01] ou dans des graphs de transition en [SMK+09] ou SVMs [GWND07]. Cependant, une évaluation objective de ces méthodes suppose l'existence d'une vérité terrain au niveau des scènes. Mais cette vérité-terrain est générée manuellement par une personne. Il est donc difficile de faire une comparaison fiable des performances des différentes approches fondées sur des jugements subjectifs.

Parmi les approches génériques, une caractéristique couramment utilisée dans la plupart des travaux récents dans le domaine est ***la récurrence*** [ABM11, BG11, Her06, Jac06]. En effet, il est très fréquent que les programmes soient composés de segments récurrents qui agissent comme des points d'ancrage dans les programmes. Des exemples de telles récurrences sont les images du présentateur dans les JTs, des jingles sonores qui annoncent le passage d'un sujet à l'autre dans un magazine de télévision/radio ou le passage à une autre étape d'une émission de jeu télévisé. Ces récurrences sont introduites volontairement dans le but de permettre aux téléspectateurs/auditeurs de suivre facilement le programme et d'identifier sa structure même s'ils ne l'ont pas regardé dès le début. En ce sens, des récurrences visuelles sont détectées dans les magazines et JTs en exploitant l'auto similarité [Jac06] ou des detectors en cascade [YTX07]. Des répétitions audio-visuelles sont aussi détectées et utilisées pour la macro-segmentation du flux TV [BML08] ou pour l'identification des séparateurs insérés entre les différentes parties d'un programme [ABM11]. Les récurrences audio sont également détectées dans les radiodiffusions pour identifier les différents sujets [BCR07] ou pour réaliser une macro segmentation du flux audio [PAO04]. Des phrases clés sont détectées et utilisées dans le sommaire des résumés audio [LC00]. Des motifs sonores fréquents sont pareillement détectés dans des émissions de radio afin d'extraire les informations significatives qui pourrait servir pour les résumés audio ou pour accélérer l'accès aux parties pertinentes des programmes [MGB09]. D'autres approches utilisent la cohérence audiovisuelle et des SVMs pour extraire les évènements importants.

En conclusion, les approches spécifiques et génériques ont toutes les deux des avantages et inconvénients. D'une part, les méthodes spécifiques peuvent être appliquées à des types des programmes très spécifiques. Elles sont caractérisées par un manque de généralité en raison de l'utilisation des règles, des modèles et des algorithmes d'apprentissage basés sur l'analyse précédente des vidéos spécifiques. D'autre part, les méthodes génériques tentent de structurer une vidéo sans utiliser la connaissances apriori. En ce qui concerne la segmentation en scènes, la définition d'une scène est très ambigue et dépend de la compréhension du chacun. Il est donc difficile de trouver une définition objective et de comparer les performances des approches existantes. L'utilisation des récurrences semble toutefois être une approche prometteuse avec beaucoup d'applications possibles. Nous avons donc décidé de l'utiliser comme un point de départ pour l'approche que nous proposons dans cette thèse.

# Solution proposée

La solution proposée se concentre sur la détection de la structure des programmes TV récurrents. Ceux-ci sont principalement des programmes diffusés périodiquement comme les programmes de divertissement, les jeux TV, les magazines, les JTs. L'une des principales propriétés de ces programmes est leur structure claire et stable. Les différentes parties de chaque épisode sont généralement délimitées par de courtes séquences vidéo, que nous avons nommé ***«séparateurs»***. Dans la Figure 1, des exemples de séparateurs sont illustrés sur un épisode d'un jeu TV français. Les boîtes représentent les séparateurs qui délimitent les parties principales du programme. Les 3 images sont extraites de trois séparateurs du jeu.

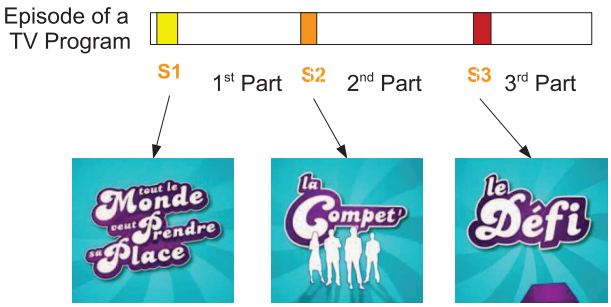



Figure 1 – Exemples de séparateurs.

L'idée principale de notre approche est de détecter ces séparateurs car ils présentent des caractéristiques qui rendent cette tâche possible. Les parties de chaque épisode sont alors indirectement identifiées en utilisant les frontières des séparateurs, par exemple la fin (l'extrémité droite) d'un séparateur représente le début d'une nouvelle partie. La structure résultante pour un épisode est donc composée d'un ensemble d'indicateurs temporels qui se réfèrent au début et à la fin de chaque partie. Les principales étapes de l'approche proposée sont : dans une première étape, un ensemble d'épisodes d'un programme récurrent est analysé, afin de détecter séparément les récurrences visuelles et audio. Les récurrences peuvent se retrouver à l'intérieur d'un même épisode (récurrences intra-épisode) et/ou entre les différents épisodes d'un même programme TV (récurrences inter-épisode). Parmi les récurrences détectées certaines sont des séparateurs mais pas toutes les récurrences sont nécessairement des séparateurs. En conséquence, dans une deuxième étape, un processus de filtrage basé sur la distribution spatiale et temporelle des récurrences est appliqué. Jusque-là, les récurrences audio et visuelles sont traitées séparément. Cependant, nous allons démontrer que chacune des deux approches a ses limites. L'approche visuelle est limitée aux récurrences identiques et l'approche audio fournit beaucoup trop de récurrences qui

ne sont pas nécessairement des séparateurs. En outre, l'utilisation des récurrences audio et visuelles ensemble pourrait améliorer les résultats de structuration. Par conséquent, afin de combiner les deux modalités et récupérer le nombre maximum de séparateurs, dans une troisième étape, toutes les récurrences obtenues après le filtrage sont passées par un module de classification basé sur les arbres de décision. Afin de pouvoir tester les limites des arbres de décision nous avons également employé les Machines à Vecteur Support (SVM). Les séparateurs détectés, sont finalement utilisés pour structurer les épisodes. Ils peuvent également être stockés dans une base de données et utilisés plus tard pour la structuration de nouveaux épisodes à venir.

## 0.1 L'algorithme de détection des récurrences visuelles

Cette partie du système effectue la description du contenu visuel des épisodes à structurer. Il procède ensuite avec la détection des récurrences à l'aide de l'approche décrite dans [BML08]. La première étape est la segmentation en plans qui se base sur la similarité de couleur de deux trames consécutives d'une fenêtre glissante. Pour chaque plan, quelques images-clés sont choisies selon la méthode décrite dans [BML08]. Cette détection est basée sur le test Page-Hinkley [Hin71] qui détermine des changements importants dans le signal. Une description des trames à deux niveaux est utilisée. D'abord, un descripteur visuel basique (BVD) de 64 bits, basée DCT, est calculé pour chaque trame. Son rôle est seulement de délimiter les frontières des récurrences et il doit être invariant seulement aux petites variations dues à la compression par exemple. Les BVDs sont facilement comparées en utilisant une distance de Hamming. Le deuxième niveau porte sur les images clés et associées à chaque image clé un descripteur plus sophistiqué et plus robuste (KVD). Il s'agit d'un descripteur de 30 dimensions, également basé DCT. Son but est d'identifier les images presque identiques. La métrique utilisée pour comparer les KVD est la distance euclidienne. Les KVD sont regroupés en utilisant une technique de micro-clustering pour identifier les plans similaires. C'est une technique qui construit d'une manière itérative des clusters sphériques. Un KVD est introduit dans un cluster tant que le rayon du cluster reste en dessous d'un certain seuil. Pour plus de détails sur la méthode de clustering, les lecteurs peuvent se référer à [BAG03]. Le nombre de KVDs par cluster correspond au nombre de fois qu'une séquence est répétée. Par exemple, une séquence répétée 2 fois, chacune des occurrences contenant 3 images clés, correspondra à un nombre de 3 clusters ayant chacun 2 KVDs. Un KVD est associé à une trame d'une récurrence mais ne fournit pas d'informations sur frontière de la séquence. Ce sont les BVDs qui sont utilisés, afin de déterminer avec précision les frontières en comparant les trames correspondantes dans toutes les occurrences de la séquence répétée. Les clusters obtenus sont analysés, et en fonction de la diversité temporelle des KVDs d'un cluster et des relations inter-clusters, l'ensemble des récurrences est créé.

## 0.2 L'algorithme de détection des récurrences audio

Précédemment nous avons considéré seulement l'information visuelle de séparateurs. Nous étions alors capable de détecter uniquement des séparateurs qui ont le même contenu visuel sans considérer les jingles sonores, qui sont des séparateurs qui partagent le même

contenu audio, mais pas nécessairement le même contenu visuel (c'est à dire la même séquence d'images. Nous étendons maintenant l'étude en tenant compte également des informations audio de séparateurs. Nous utilisons la même hypothèse que celle sur laquelle l'approche visuelle est fondée, c'est à dire, que les séparateurs sont répétés et peuvent être détectés comme des récurrences. Cependant, dans ce cas, nous considérons les récurrences audio.

Le schéma général de la solution comporte 4 étapes principales :

**Etape 1.** En utilisant une trame audio glissante, des caractéristiques audio basiques, nommés sous-descripteurs, sont extraites de tous les épisodes analysés et sont stockées dans une base de données.

**Etape 2.** Chacun des épisodes analysés sera considéré, à son tour, comme une requête alors que les autres vont devenir des références. Une fenêtre glissante (avec un nombre prédéfini des sous-descripteurs), est utilisée pour comparer la requête avec la référence. Une correspondance est considérée comme détectée si la distance entre les deux ne dépasse pas un certain seuil de similarité. A la fin du traitement, un ensemble des paires de segments correspondants est obtenu.

**Etape 3.** Les segments, identifiés lors de la 2e étape, sont lissés et prolongés lorsque des segments se chevauchent. Si la distance temporelle entre deux segments est en-dessous un certain seuil, ces segments sont fusionnés.

**Etape 4.** Les segments correspondants sont ensuite analysés et regroupés. Au début, chaque paire de segments obtenus lors de la 3e étape instancie un cluster. Les clusters qui se chevauchent sont ensuite fusionnés.

Trois types de descripteurs audio ont été considérés : Perceptual Audio Hashing (PAH), un descripteur basé MFCC et un descripteur basé Phonème. Cependant, le descripteur PAH est le plus approprié pour la détection des récurrences pertinentes pour la structuration. C'est un descripteur de 32 bits, basé sur le signe des différences d'énergie le long de l'axe fréquentiel et de l'axe temporel pour deux trames audio consécutives. Plus de détails sur ce descripteur figurent dans [HKO01].

## 0.3 Le module de filtrage des récurrences

Les récurrences détectées lors de l'étape précédente, ne sont pas nécessairement des séparateurs. Il peut arriver que dans les épisodes analysés, il y ait des séquences qui sont rediffusées ou qui sont très similaires. Celles-ci ne sont pas des séparateurs. Par exemple, les plans montrant le modérateur dans la même position mais à différents moments de l'épisode, pourraent être détectés comme occurrences d'une séquence répétée. Ils ne sont pas des séparateurs mais leur contenu est très similaire. Nous appelons ces détections des «*fausses alarmes*» et afin de les filtrer, une étape de post-traitement est utilisée. Dans un premier temps, les récurrences obtenues sont passées par un **filtre apriori** qui supprime toutes les séquences de répétition avec toutes les occurrences provenant seulement d'un même épisode. Même si un séparateur peut être répété dans le même épisode, il doit être également répété dans au moins encore un épisode pour être valide. Il est très peu probable d'avoir un séparateur créé spécialement pour un seul épisode. Ce filtrage va éliminer toutes les fausses-alarmes intra-épisode. Mais comme dit auparavant, des fausses-alarmes inter-épisode peuvent aussi apparaître.

Toutefois, les séparateurs ont une propriété (i.e. la **stabilité temporelle**) qui permet

leur identification parmi les récurrences. Généralement, les épisodes d'un même programme de télévision ont une durée et une structure similaire. Cela signifie qu'ils correspondent généralement à des intervalles de temps similaires et les différentes parties qui composent le programme restent les mêmes d'un épisode à l'autre. Les séparateurs qui délimitent les différentes parties se trouvent donc à peu près au même endroit d'un épisode à l'autre pour un même programme récurrent. Nous utilisons donc la stabilité temporelle des séparateurs afin de filtrer les récurrences qui sont des fausses alarmes. Pour ce faire, une étude de la densité temporelle des occurrences de séquences de répétition détectées est effectuée. Toutes les occurrences de différents épisodes sont projetées sur un même axe temporel. De cette projection, un histogramme est calculé en comptant le nombre d'occurrences chaque 40ms (chaque trame). Une estimation de la densité basée sur un noyau gaussien est alors effectuée [Sil86] :

$$f_i = \sum_{j=i-3\sigma}^{i+3\sigma} h_j e^{-\frac{(j-i)^2}{2\sigma^2}}, \tag{1}$$

où $f_i$ représente le résultat obtenu après filtrage pour la trame $i$ et $h(j)$ représente le nombre d'occurrences calculées de l'histogramme, qui correspondent à la trame $j$.

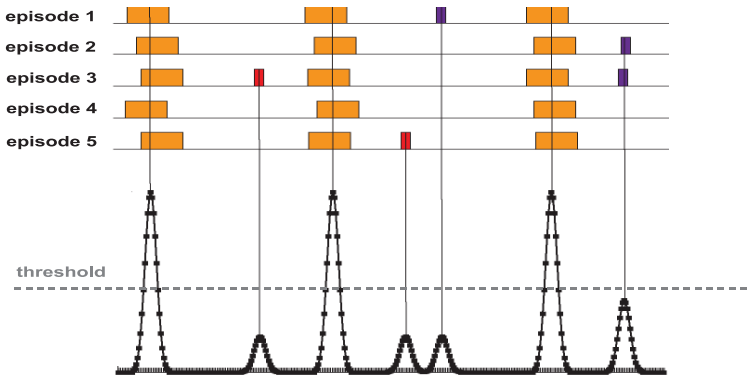



FIGURE 2 – Densité temporelle des récurrences.

Le résultat de l'analyse temporelle de la densité est une courbe de distribution pour laquelle un maximum représente une zone de forte concentration de séparateurs. Un seuil est ensuite empiriquement déterminé et les récurrences qui ont une densité inférieure au seuil sont rejetées. Celles-ci correspondent à des récurrences isolées et sont susceptibles d'être des fausses alarmes dues au fait qu'elles apparaissent quasi-aléatoirement, sans aucune stabilité temporelle. Un exemple illustratif est donné dans la Figure 2. Les occurrences en bleu et rouge sont des cas isolés qui correspondent aux fausses alarmes. Par conséquent elles seront filtrées.

Le filtre temporel peut être appliqué seulement à des programmes qui ont une stabilité temporelle, comme les jeux de télévision par exemple. Dans le cas des journaux TV, où le nombre de reportages et leurs durées sont différents d'un épisode à l'autre, la condition de stabilité temporelle n'est pas satisfaite. En outre, dans le cas de la modalité audio, les récurrences détectées sont souvent beaucoup trop nombreuse et réparties de façon aléatoire. Nous avons décidé alors d'appliquer le filtre temporel uniquement pour les récurrences visuelles qui proviennent de programmes stables temporellement.

## 0.4 Le module de classification

Les récurrences obtenues après le filtrage décrit dans la section précédente, sont ensuite passées par un module de classification et de sélection basé sur les arbres de décision. L'idée est de construire un modèle à travers une séquence de questions posées sur un ensemble de variables d'entrée (Figure 3). Chaque question suivante dépend de la réponse précédente. Une fois le modèle construit il peut être utilisé par la suite pour prédire la valeur d'une variable cible calculée sur la base des variables d'entrée. Dans notre cas, on considère toutes les variables comme étant des variables binaires. Nous définissons la variable cible comme la catégorie à laquelle appartient la récurrence. Une récurrence est un séparateur si la catégorie attribuée est 1 et est une fausse alarme (non-séparateur) si la catégorie attribuée est 0. Nous appelons les variables d'entrée attributs. Ceux-ci sont au nombre de 13 et représentent le résultat d'un ensemble de modules technologiques parmi lesquelles la détection des applaudissements, la segmentation de scène, la segmentation et le regroupement des locuteurs et finalement la détection et le regroupement des visages.

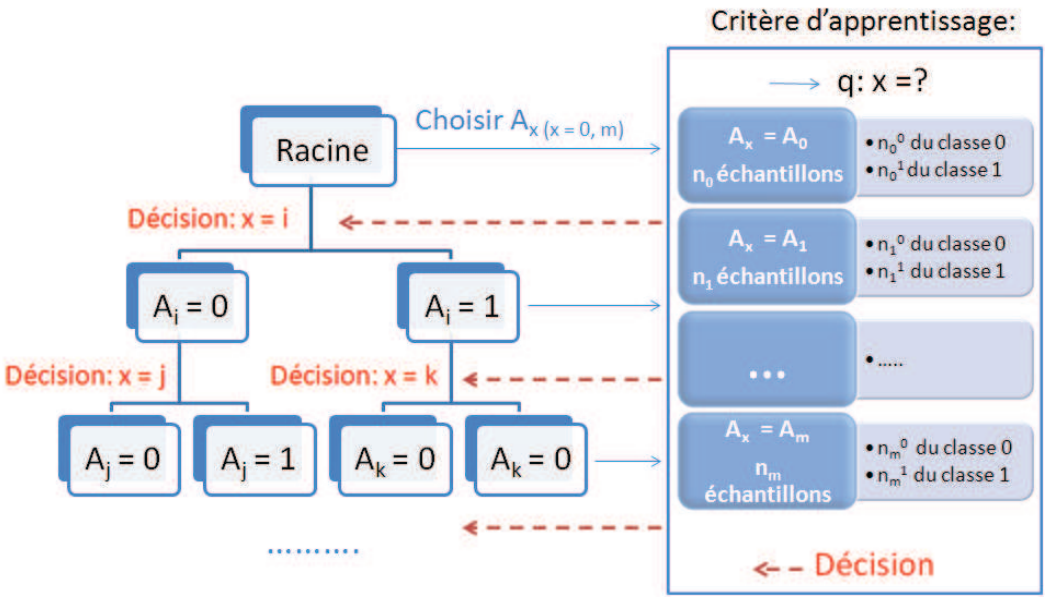



FIGURE 3 – L'algorithme de construction des arbres de décision.

Pendant la phase d'apprentissage, chaque échantillon est décrit par les 13 attributs binaires et par une catégorie cible qui confirme si la récurrence est un séparateur ou pas. L'arbre est construit en posant des questions sur les attributs. L'attribut choisi pour une questionne est sélectionné par un critère prédéfini. Nous avons testé 3 critères différents afin de choisir celui qui rend le meilleur résultat.

Les tests sur l'attribut choisi seront utilisés pour diviser l'ensemble d'apprentissage en sous-ensembles. L'algorithme est récursif et va diviser progressivement l'ensemble d'apprentissage dans des sous-ensembles tant que le sous-ensemble n'est pas «pur» (tous les échantillons appartiennent à la même catégorie) ou qu'il existe au moins un autre attribut à tester. La division peut être arrêtée à l'avance en imposant une contrainte sur les paramètres du critère choisi. Dans ce cas, le noeud est déclaré comme feuille et une catégorie

lui est attribuée.

Afin d'étendre encore l'utilisation des arbres de décision, nous proposons aussi l'utilisation de combinaisons d'attributs lors de la phase d'apprentissage. L'idée est d'offrir la possibilité d'interroger plusieurs attributs en même temps au lieu d'un seul. Deux cas ont été considérés. Dans un premier, le nombre d'attributs à interroger est donné par un paramètre. Dans un deuxième toutes les combinaisons possibles de l'ensemble des attributs sont testées.

Selon le modèle construit et les valeurs des attributs correspondants, chaque récurrence de l'ensemble de données de test sera classée dans l'une des deux catégories (séparateur / non-séparation (fausse alarme)).

Pour pouvoir comparer les résultats obtenus avec les arbres de décision, afin de conclure sur leurs influences et leurs limitations, nous envisageons aussi l'utilisation d'un autre classifieur c'est-à-dire les machines a vecteurs support.

# Expérimentations

Afin d'évaluer la solution proposée pour la détection des séparateurs dans les programmes TV récurrents, nous avons effectué des expérimentations en utilisant des émissions réelles de la télévision française.

## 0.5 Contexte expérimental

La base de données utilisée pour les expérimentations est composée de 169 épisodes de 11 programmes de télévision française, parmi lesquels des jeux, des magazines et un journal télévisé. Ceux-ci sont décrits dans le tableau ci-dessous (Table 1).

| Type du Programme | Nom du Programme | Id du Programme | No. des épisodes | No. des séparateurs/épisode |
|---|---|---|---|---|
| Jeux | Les Z'amours | $G_1$ | 28 | 6 |
| | Mot de passe | $G_2$ | 5 | 4 |
| | Motus | $G_3$ | 21 | 4 |
| | Tlmvpsp | $G_4$ | 26 | 5 |
| | Les 12 coups de midi | $G_5$ | 20 | 8 |
| Magazines | Comment ça va bien | $M_1$ | 24 | 8/14 |
| | 10h Le Mag | $M_2$ | 5 | 20/23 |
| | Cine, Series et cie | $M_3$ | 7 | 15/17 |
| | 50mn Inside | $M_4$ | 14 | 12/18 |
| JT | 7 A 8 | $M_5$ | 10 | 5/7 |
| | 19h45 | $M_6$ | 9 | 20/27 |

Tableau 1 – Description des données.

Pour tous les épisodes de l'ensemble de données expérimentales, nous avons annoté manuellement les séparateurs. Chaque séparateur a été déterminé avec précision, en indiquant son début et sa fin. Ceci représente la vérité terrain et sera utilisé pour évaluer la méthode proposée. Nous insistons sur le fait que nous n'avons pas annoté toutes les récurrences visuelles ou audio existantes dans les épisodes, ce qui serait une tâche très difficile, voire impossible. Nous utilisons la vérité terrain pour évaluer la méthode proposée pour la détection des séparateurs. Ces séparateurs peuvent se retrouver parmi les récurrences détectées. Nous n'avons pas évalué la capacité des méthodes pour détecter les récurrences, mais leur capacité à détecter les séparateurs. Les expérimentations sont réalisées sur des ensembles de quatre épisodes (l'épisode courant plus trois épisodes dans l'historique) afin de détecter les séparateurs du quatrième épisode. Seuls ces derniers séparateurs seront pris

en considération pour l'évaluation. Le nombre d'épisodes considérés dans l'historique a été empiriquement déterminé.

Pour l'évaluation du système proposé, nous utilisons la précision, le rappel et la moyenne harmonique (F). On considère un séparateur correctement identifié s'il chevauche avec son correspondant de la vérité terrain.

## 0.6 Résultats obtenus pour l'approche visuelle

### 0.6.1 Analyse des récurrences visuelles détectées

L'objectif de cette première expérimentation est d'évaluer l'efficacité de l'algorithme de détection des récurrences visuelles pour la détection des séparateurs. Le point de départ de notre étude est la détection de récurrences. L'hypothèse principale de notre approche est que les séparateurs sont répétés et peuvent être détectés comme récurrences. Afin de valider notre hypothèse, nous avons évalué dans cette première expérimentation la proportion des séparateurs détectés comme récurrences. Pour ce faire, nous avons appliqué la méthode de détection de récurrences visuelles présentée dans la Section 0.1. Les résultats obtenus se retrouvent dans le tableau ci-dessous.

| **Données** | | Précision | Rappel | F |
|---|---|---|---|---|
| **Jeux** | G1 | 0.778 | 0.968 | 0.863 |
| | G2 | 0.173 | 0.950 | 0.293 |
| | G3 | 0.224 | 0.952 | 0.362 |
| | G4 | 0.556 | 0.785 | 0.651 |
| | G5 | 0.033 | 0.013 | 0.018 |
| | Global G | 0.441 | 0.663 | 0.530 |
| **Mag & JT** | M1 | 0.259 | 0.183 | 0.214 |
| | M2 | 0.791 | 0.955 | 0.865 |
| | M3 | 0.515 | 0.120 | 0.194 |
| | M4 | 0.702 | 0.550 | 0.617 |
| | M5 | 0.638 | 0.910 | 0.750 |
| | M6 | 0.479 | 0.702 | 0.569 |
| | Global M | 0.614 | 0.480 | 0.538 |
| **Global** | | 0.502 | 0.563 | 0.531 |

Tableau 2 – Exp1 : Analyse des récurrences visuelles.

Les résultats dans le Tableau 2 montrent que pour certaines émissions on a un très grand rappel(G1, G2, G3, M2, M5) ce qui signifie une très bonne détection des séparateurs. Pour d'autres émissions par contre, on retrouve des séparateurs spécifiques qui ne sont pas adaptés à l'approche utilisée pour la détection des récurrences visuelles. Un exemple se trouve dans l'émission G5 (Figure 4) où les séparateurs sont composés d'un logo superposé à des images mobiles du plateau, du présentateur ou du public.

Le logo est très animé et les images du fond mobiles. Un autre exemple se trouve dans M1 - les séparateurs dans ce cas sont animés par différentes effets d'édition. Le même type de séparateurs se retrouve aussi dans M6 (Figure 5) . Ce type de séquences ne peu pas


FIGURE 4 – Exemple de séparateurs de l'ensemble de données G5.

être donc considéré comme des séquences récurrentes et est en conséquence impossible à détecter par notre approche.

FIGURE 5 – Séparateur de l'ensemble de données M6 avec un effet de «fold out».

Par rapport à la précision les résultats sont plus faibles. Les fausses-alarmes détectées se réfèrent en général à des images similaires du plateau, du présentateur, des concurrents, à des images noires ou blanches, ou à des images très similaires mais pas identiques comme celles dans la Figure 6.


FIGURE 6 – Exemples des fausses-alarmes.

### 0.6.2 Analyse des récurrences visuelles après filtrage

Pour filtrer les fausses alarmes détectées dans l'expérimentation précédente, nous avons proposé 2 filtres : un premier, *filtre a priori*, pour les fausses alarmes intra-épisode, et un deuxième, *filtre temporel*, pour les fausses alarmes inter-épisode. Les résultats obtenus après l'utilisation de ces filtres se trouvent dans les résultats du Tableau 4

Par rapport au filtrage a priori, si on compare les résultats globaux obtenus, avec les résultats globaux du Tableau 2, on observe une importante augmentation de la précision, de 33%, avec une légère dégradation du rappel, de 2%. Cette dégradation ne vient pas des jeux, où le rappel reste stable et la précision augmente fortement (de 40%), mais elle vient plutôt des magazines et JT où la plupart des séparateurs sont des récurrences intra- et inter-épisode. On insiste sur le fait que cette dégradation est peu importante par rapport au gain apporté par ce filtre.

| Données | P | R | F |
|---|---|---|---|
| Jeux | 0.847 | 0.663 | 0.744 |
| Mag. & JT | 0.816 | 0.462 | 0.590 |
| Global | 0.830 | 0.539 | 0.653 |

(après filtrage apriori)

| Données | P | R | F |
|---|---|---|---|
| Jeux | 0.878 | 0.663 | 0.755 |
| Mag. & JT | 0.856 | 0.419 | 0.562 |
| Global | 0.878 | 0.511 | 0.646 |

(après filtrage apriori+temporel)

Tableau 3 – Exp2 : Résultats obtenus pour les récurrences visuelles après filtrage.

Si suite au filtrage a priori on applique le filtrage temporel, les résultats obtenus se retrouvent dans la partie droite du Tableau 4. Sur les résultats globaux ce filtre n'apporte pas grand-chose. Si on analyse séparément, pour les jeux on a une légère augmentation de la précision avec un rappel qui est préservé. Par contre pour les magazines et JT les variations du rappel et de la précision sont similaires - c'est-à-dire l'augmentation de la précision est similaire à la dégradation du rappel. Ce filtre n'apporte donc rien pour ce type d'émissions. Cela pourrait s'expliquer par le fait que les magazines et les JT sont composés d'une série de reportages et de plans présentateurs qui ne sont pas d'une même longueur et de plus leur nombre peut différer d'une émission à l'autre. La condition de stabilité temporelle n'est donc pas satisfaite pour ces émissions et on ne va plus considérer ce filtre pour ce type de programmes.

Un filtre sur la longueur des séparateurs a également été étudié, mais les résultats obtenus n'ont pas été concluants.

## 0.7 Résultats obtenus pour l'approche audio

### 0.7.1 Analyse des récurrences audio détectées

Comme pour l'approche visuelle, dans cette expérimentation nous évaluons l'efficacité de l'algorithme de détection des récurrences audio pour la détection des séparateurs. La proportion des séparateurs détectés parmi les récurrences audio obtenue lors de l'application de l'algorithme décrit dans la Section 0.2 se retrouvent dans la partie gauche du tableau ci-dessous.

| Données | P | R | F |
|---|---|---|---|
| Jeux | 0.088 | 0.942 | 0.161 |
| Mag. & JT | 0.298 | 0.929 | 0.451 |
| Global | 0.137 | 0.934 | 0.239 |

(avant filtrage apriori)

| Données | P | R | F |
|---|---|---|---|
| Jeux | 0.093 | 0.942 | 0.169 |
| Mag. & JT | 0.437 | 0.929 | 0.595 |
| Global | 0.155 | 0.934 | 0.266 |

(après filtrage apriori)

Tableau 4 – Exp3 : Résultats obtenus pour les récurrences audio avant et après filtrage a priori.

Les résultats montrent des valeurs très grandes du rappel ce qui veut dire que la plupart des séparateurs ont été retrouvés. Les séparateurs qui n'ont pas été détectés correspondent en général à des séparateurs qui contiennent du langage naturel. Le descripteur utilisé dans l'approche proposée n'est pas adapté pour les récurrences avec du langage naturel et donc celles-ci ne peuvent pas être détectées. D'autres exemples de séparateurs non-détectés sont

ceux contenant un bruit de fond comme des applaudissement ou de la parole.

Concernant la précision les résultats sont beaucoup plus faibles. L'approche audio détecte beaucoup de fausses alarmes qui correspondent à des récurrences des morceaux de musique, des mots ou groupes de mots répétés, des sons spécifiques (appui d'un bouton dans le cas des jeux TV, explosions ou coups de feu dans les magazines ou JT), des parties de reportages qui se répètent, des silences, des applaudissements enregistrés, etc. Il est important de noter que par rapport à l'approche visuelle, on a 14 fois plus de récurrences, la plupart étant des fausses alarmes. La majorité de ces fausses alarmes proviennent des jeux TV et sont représentées par des sons spécifiques issus de l'appui d'un bouton par un concurrent pour répondre à une certaine question, ou de sons qui annoncent qu'une réponse est correcte ou pas, ou par des séquences enregistrées avec des réactions du public ou des applaudissements.

### 0.7.2 Analyse des récurrences audio après filtrage

Pour filtrer ces fausses alarmes nous avons appliqué ensuite le filtrage a priori. Les résultats obtenus se retrouvent dans la partie droite du Tableau 4.

Par rapport au rappel les résultats restent stables donc le filtre n'élimine pas les séparateurs. La précision globale de l'ensemble des données augmente de seulement 2% après ce filtrage. Cependant, si nous regardons de plus près les résultats pour chaque catégorie de programmes (jeux / magazines et JT) séparément, on constate que l'amélioration globale de la précision est en fait diminuée par la précision petite obtenue dans le cas des jeux. Pour ces derniers l'amélioration est trop faible pour être considérée. Les fausses alarmes décrites dans la section précédente, pour les jeux télévisés, sont inter- et intra-récurrences et donc elles ne peuvent pas être filtrées avec ce filtre a priori qui s'adresse uniquement aux fausses-alarmes intra-épisode. Au contraire, le filtre montre une contribution pour les magazines et JT où la précision augmente de 14%. Le filtre a priori s'est avéré être donc plus ou moins utile selon le type de programme qui est analysé. Nous allons en tenir compte dans toutes les prochaines expériences.

Ce filtre a priori ne traite que des fausses alarmes intra-épisode. Dans la Section 0.3, nous avons présenté un deuxième filtre, le filtre temporel, qui se réfère à des fausses alarmes inter-épisode. Pourtant, dans la Section 0.6.2 nous avons conclu que ce filtre est inutile dans le cas des magazines et les JT où les séparateurs ne respectent pas la condition de stabilité temporelle. Ce filtre ne sera pas donc appliqué sur ces programmes. Nous pourrions cependant l'utiliser dans le cas des émissions de jeux télévisés, bien que le nombre de fausses alarmes audio est trop grand pour ces programmes. Une estimation de la densité de séparateurs serait donc difficile à calculer vu que les fausses alarmes sont trop nombreuses et réparties dans les épisodes entiers. Ceux-ci vont ainsi influencer le choix des paramètres du filtre temporel. L'utilisation du filtre temporel n'est donc pas justifié pour le filtrage des récurrences audio. Par conséquent, ce filtre ne sera pas considéré pour les autres expériences qui impliquent les récurrences audio.

Un filtre sur la longueur des séparateurs a également été étudié, mais les résultats obtenus n'ont pas été concluants.

## 0.8 Résultats obtenus pour le module de classification

Jusqu'à maintenant nous avons traité les 2 approches (visuelle et audio) séparément. Nous avons vu que chacune a ses limitations. Pour l'approche visuelle, seulement les séparateurs presque identiques sont identifiés. Pour l'approche audio la plupart des séparateurs sont identifiés mais au même temps beaucoup trop de fausses-alarmes sont détectées. Afin de surmonter ces limites et de plus, pour fusionner les 2 approches afin d'obtenir de meilleurs résultats, nous avons, dans une étape suivante, passé les récurrences par un module de classification basé sur les arbres de décision. Son but est de classifier les récurrences en séparateurs et non-séparateurs (fausse alarme) mais aussi d'intégrer les 2 approches afin de pouvoir cumuler les bénéficies de chacune.

Pour ces expérimentations, la base de données a été divisée en 2 ensembles : un ensemble d'apprentissage et un ensemble de test. Le premier est utilisé pour la construction des arbres de décision tandis que le second est utilisé pour l'évaluation des arbres précédemment appris (test).

Les récurrences utilisées sont, pour l'approche audio, les récurrences obtenues après le filtre a priori. Pour l'approche visuelle, nous avons utilisé les récurrences obtenues après le filtre a priori et aussi, pour les programmes de stabilité temporelle, après le filtre temporel. De cette manière, l'ensemble de récurrences consiste en 14,283 récurrences parmi lesquelles 14.130 sont des récurrences audio et 1.002 sont des récurrences visuelles.

### 0.8.1 Analyse des récurrences de l'ensembles de données

Nous avons d'abord évalué la proportion de récurrences qui sont des séparateurs dans l'ensemble d'apprentissage. Ceci est équivalent à envisager un classificateur naïf qui attribue à toutes les récurrences la classe «séparateurs». Le tableau 4.2 résume les résultats obtenus.

| Données | Récurrences audio | Récurrences visuelles |
|---|---|---|
| Jeux | 6.12% | 82.48% |
| Mag. & JT | 40.72% | 74.42% |

Tableau 5 – Exp4 : Proportion de récurrences qui sont des séparateurs dans l'ensemble d'apprentissage.

Pour les jeux télévisés, parmi les récurrences audio, seulement près de 6% sont des séparateurs tandis que le reste des 94% devrait être filtré car ils appartiennent à la catégorie «non-séparateurs». Ceux-ci correspondent à des séquences audio spécifiques que l'on retrouve dans les jeux télévisés. Un exemple est le son produit par un certain bouton quand un concurrent veut répondre à une question, ou la séquence audio qui annonce si la réponse était vraie ou fausse. Au contraire, pour les récurrences visuelles, la plupart des récurrences (82%) sont des séparateurs et seulement 18% sont des «non-séparateurs». Pour le cas des magazines de télévision et des journaux télévisés, nous avons remarqué le même phénomène, même si le pourcentage de séparateurs parmi les récurrences audio est plus élevé (41%) que pour le correspondant dans les jeux télévisés.

### 0.8.2 Résultats obtenus dans la phase de classification

Pour la phase d'apprentissage, 3 critères ont été testés. Parmi eux, le critère de la pureté qui choisit l'attribut qui rend les noeuds immédiats descendants aussi purs que possible, a donné les meilleurs résultats.

Nous avons aussi expérimenté l'apprentissage avec des attributs complexes décrit dans la Section 0.4. Les résultats obtenus ne sont pas significativement différents et de plus la solution s'est avérée trop coûteuse et trop complexe par rapport au gain apporté.

En appliquant les arbres appris avec le critère de la pureté et attribut unique, les résultats obtenus se trouvent dans le Tableau 7. Le Tableau 6 reprend les résultats avant classification.

| **Données** | **Récurrences audio** | | | **Récurrences visuelles** | | | **Audio ∪ Visuel** | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| G | 0.07 | 0.95 | 0.13 | 0.75 | 0.74 | 0.73 | 0.08 | 0.97 | 0.14 |
| Mag & JT | 0.42 | 0.92 | 0.55 | 0.71 | 0.48 | 0.55 | 0.43 | 0.93 | 0.56 |

Tableau 6 – Exp5 : Détection des séparateurs - avant classification - ensemble de test.

| **Données** | **Récurrences audio** | | | **Récurrences visuelles** | | | **Audio ∪ Visuel** | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| G | 0.84 | 0.76 | 0.79 | 0.78 | 0.65 | 0.71 | 0.85 | 0.78 | 0.82 |
| Mag & JT | 0.84 | 0.52 | 0.65 | 0.74 | 0.47 | 0.57 | 0.85 | 0.62 | 0.72 |

Tableau 7 – Exp6 : Détection des séparateurs - après classification - ensemble de test.

Les trois dernières colonnes des tableaux représentent l'union des résultats obtenus pour l'approches audio et visuelle séparément. On a ajouté ces colonnes parce qu'elles apportent une information sur les contributions de chaque approche séparément mais aussi sur la performance globale de détection des séparateurs quand on considère les 2 approches en même temps.

La tendance globale est une augmentation de la précision avec une baisse du rappel. Dans le cas des récurrences visuelles les résultats après la classification sont très peu impactés. Si on considère la F mesure, en général les résultats après classification sont meilleurs. Cette amélioration est apportée par l'approche audio pour laquelle la précision augmente fortement après classification. Cela influence aussi les résultats qui prennent en considération les 2 approches en même temps où on a une forte augmentation de la F mesure de 68% pour les Jeux et 16% pour les magazines et JT.

Les résultats obtenus après classification pourraient être influencés par plusieurs facteurs comme le bruit introduit par les technologies utilisées pour le calcul des attributs, les attributs proposés qui ne sont pas suffisamment pertinents ou les limites des arbres de décision. La dernière spéculation citée peut toutefois être testée en utilisant un autre classifieur approprié pour le type de données que nous traitons. Un outil puissant de classification que nous considérons adapté sont les Machines à Vecteurs Support. Nous l'avons utilisé ainsi pour classification du même ensemble de données. Les résultats sont très similaires à ceux obtenus avec les arbres de décision, ce qui exclut donc l'hypothèse que

les arbres de décision limitent les résultats obtenus pour la classification des récurrences audio et visuelles. Par conséquent, les résultats sont plus probablement influencés par les attributs choisis et les technologies utilisées.

# Conclusions et Perspectives

La méthode proposée analyse un ensemble d'épisodes d'un programme de télévision récurrent pour extraire les "séparateurs". Les séparateurs sont de séquences vidéo courtes, insérées entre les différentes parties d'un programme, et qui peuvent être répétés à l'intérieur et / ou entre plusieurs épisodes d'un même programme. Une fois les séparateurs retrouvés, ils peuvent être utilisés par la suite pour structurer chacun des épisodes analysés ou chaque nouvel épisode à arriver. Les séparateurs représentent la base de notre système automatique de segmentation des programmes de télévision récurrents. La principale hypothèse que avons considérée et aussi validée, est que les séparateurs sont répétés et peuvent être détectés comme des récurrences. Le point de départ de notre système devient ainsi la détection des récurrences. Les séparateurs ont un contenu visuel et / ou audio récurrent. Nous avons donc suivi deux axes et testé les performances de détection des séparateurs parmi les récurrences pour chaque modalité (visuelle / audio) séparément. Parmi les récurrences détectées il y a aussi les séparateurs mais pas toutes les récurrences sont nécessairement des séparateurs. Par conséquent, un processus de filtrage basé sur la distribution spatiale et temporelle des récurrences est nécessaire. Les algorithmes de détection des récurrences audio et visuelles se comportent différemment. L'approche audio détecte un nombre élevé de récurrences. Parmi elles il y a des séparateurs, mais la majorité de récurrences sont des non-séparateurs. L'algorithme basé sur le contenu visuel détecte moins de récurrences la plupart étant des séparateurs. Les deux approches ont chacune leurs limites : par exemple dans le cas de la détection des récurrences audio, les séparateurs contenant de la parole ne peuvent pas être détectés. Pour le cas de l'algorithme de détection des récurrences visuelles, les séparateurs qui ne sont pas identiques ne sont pas détectés. Il se peut aussi que les algorithmes de clustering employés, qui comportent un réglage optimal d'un certain nombre de paramètres, ne sont pas appropriés pour certains cas spécifiques. En conséquence, nous avons considéré que lorsque l'on combine les deux approches, on pourrait surmonter les limites de l'autre et ainsi fournir le nécessaire pour une meilleure structuration. Par conséquent, afin de combiner les deux modalités et de récupérer le nombre maximal de séparateurs nous avons proposé, dans une dernière étape, de passer les récurrences audio et visuelles obtenues par un système de classification et de sélection basé sur les arbres de décision. Son but est de décider si une récurrence est un séparateur ou pas. Pour construire les arbres de décision, des attributs issus de techniques telles que la détection des applaudissements, la segmentation des scènes, la détection et le clustering des visages ou des locuteurs ont été utilisés. Les résultats obtenus ont montré que beaucoup des récurrences qui ne sont pas des séparateurs sont filtrées (surtout pour les récurrences audio) mais avec elles une partie des séparateurs aussi. Toutefois, lorsque l'on analyse globalement les résultats en utilisant la F-mesure, en particulier pour l'approche audio, les résultats sont meilleurs après avoir effectué l'étape de classification.

Nous avons également examiné et utilisé pour la comparaison un autre classificateur basé sur les SVM. Des différences significatives n'ont pas été enregistrées. Cela représente une preuve importante qui atteste le fait que les résultats n'ont pas été très influencés par la méthode choisie pour la classification, mais plutôt par les attributs choisis et les limites des méthodes utilisées lors du calcul de ces attributs.

Le travail présenté dans cette thèse pourrait encore se poursuivre avec de nouvelles perspectives. Certains d'elles se réfèrent aux améliorations qui pourraient être apportées à cette thèse. D'autres vont plus loin et proposent de nouveaux domaines où l'approche proposée pourrait être testée et exploitée. Une première perspective importante provient des difficultés que nous avons rencontrées lors de l'analyse des résultats obtenus avec les mesures d'évaluation proposées (la précision, le rappel et la F mesure). Il serait intéressant de réaliser une étude pour relier les différentes types d'erreurs (faux négatifs/ faux positifs) à la satisfaction de l'utilisateur pour en déduire quelles erreurs sont plus perturbantes pour celui-ci. Une autre perspective intéressante pourrait être la classification des séparateurs dans différentes catégories. Cela supposerait l'identification des cas particuliers de séparateurs et de leur étiquetage en conséquence. Cela permettrait ensuite une analyse plus facile et plus précise des résultats. A mentionner aussi le fait que nous avons été contraints par un historique court lors du calcul des récurrences audio et visuelles. Un ensemble de données plus large permettrait une analyse plus précise sur une plus longue période de temps et sur un plus grand nombre d'épisodes diffusés. En outre, il pourrait aussi résoudre les problèmes du système de classification proposé. Au cours des expériences nous avons également remarqué qu'il y a des récurrences détectées à l'intérieur des parties principales du programme. Certaines d'entre elles représentent des marqueurs qui pourraient être utilisés pour structurer les parties des programmes auxquels elles appartiennent. On pourrait donc envisager comme perspective, un nouveau niveau de structuration des programmes TV avec une structuration plus «dense», qui prend en compte également le contenu de chaque partie du programme. Une autre perspective considère les résumés vidéo. La structuration intra-programme pourrait être considérée comme une étape préliminaire pour la construction de résumés vidéo. L'information apportée sur la structure d'un programme améliore la compréhension du programme qui pourrait conduire à la création de résumés plus précis et plus évolués.

# Bibliographie


[ABCB02] Jürgen Assfalg, Marco Bertini, Carlo Colombo, et Alberto Del Bimbo. Semantic annotation of sports videos. *IEEE MultiMedia*, 9(2) :52 – 60, April 2002.

[ABM11] Alina Elma Abduraman, Sid-Ahmed Berrani, et Bernard Merialdo. An unsupervised approach for recurrent tv program structuring. *Proceedings of the 9th edition of the European Interactive TV Conference*, pages 123–126, Lisbon, Portugal, June-July 2011.

[ATK00] Yannis S. Avrithis, Nicolas Tsapatsoulis, et Stefanos D. Kollias. Broadcast news parsing using visual cues : a robust face detection approach. *Proceedings of the IEEE International Conference on Multimedia and Expo*, number 3, pages 1469 – 1472, New York , USA, August 2000.

[BAG03] Sid-Ahmed Berrani, Laurent Amsaleg, et Patrick Gros. Approximate searches : k-neighbors + precision. *Proceedings of the 12th ACM International Conference on Information and Knowledge Management*, pages 24–31, New Orleans, Louisiana, USA, November 2003.

[BBP01] M. Bertini, A. Del Bimbo, et P. Pala. Content-based indexing and retrieval of tv news. *Pattern Recognition Letters*, 22(5) :503–516, April 2001.

[BCR07] Michael Betser, Patrice Collen, et Jean-Bernard Rault. Audio identification using sinusoidal modeling, and application to jingle detection. *Proceedings of the 8th International Conference on Music Information Retrieval*, pages 1–4, Vienna, Austria, September 2007.

[BG11] Mathieu Ben et Guillaume Gravier. Unsupervised mining of audiovisually consistent segments in videos with application to structure analysis. *Proceedings of the IEEE International Conference on Multimedia and Exhibition*, pages 1–6, Barcelone, Espagne, July 2011.

[BML08] Sid-Ahmed Berrani, Gael Manson, et Patrick Lechat. A non-supervised approach for repeated sequence detection in tv broadcast streams. *Image Communication*, 23(7) :525–537, August 2008.

[CCL03] Lekha Chaisorn, Tat-Seng Chua, et Chin-Hui Lee. A multi-modal approach to story segmentation for news video. *World Wide Web*, 6(2) :187–208, 2003.

[ETM03] Ahmet Ekin, A. Murat Tekalp, et Rajiv Mehrotra. Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, 12(7) :796–807, July 2003.

[EWIR01] S. Eickeler, F. Wallhoff, U. Iurgel, et G. Rigoll. Content based indexing of images and video using face detection and recognition methods. *Proceedings*

*of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 1505–1508, Salt Lake City, Utah, USA, May 2001.

[FZF06] Yong Fang, Xiaofei Zhai, et Jingwang Fan. News video story segmentation. *Proceedings of the 12th International Multi-Media Modelling Conference*, pages 397–400, Beijing, China, January 2006.

[GFT98] B. Gunsel, A. Ferman, et A. Tekalp. Temporal video segmentation using unsupervised clustering and semantic object tracking. *Journal of Electronic Imaging*, 7(3) :592–604, 1998.

[GSSD09] Abhinav. Gupta, Praveen Srinivasan, Jianbo Shi, et Larry S. Davis. Understanding videos constructing plots learning a visually grounded storyline model from annotated. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Miami, Florida, USA, June 2009.

[GT02] Xinbo Gao et Xiaoou Tang. Unsupervised video-shot segmentation and model-free anchorperson detection for news video story parsing. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(9) :765 – 776, September 2002.

[Gué02] André Guéziec. Tracking pitches for broadcast television. *Computer*, 35(3) :38 – 43, March 2002.

[GWND07] Naveen Goela, Kevin Wilson, Feng Niu, et Ajay Divakaran. An svm framework for genre-independent scene change detection. *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 532–535, Beijing, China, July 2007.

[Her06] C. Herley. Argos : automatically extracting repeating objects from multimedia streams. *Proceedings of the IEEE Transactions on Multimedia*, volume 8, pages 115–129, 2006.

[Hin71] David V. Hinkley. Inference about the change-point from cumulativesum tests. *Biometrika*, 58(3) :509–523, December 1971.

[HKG+10] Raffay Hamid, Ram Krishan Kumary, Matthias Grundmannz, Kihwan Kimz, Irfan Essaz, et Jessica Hodgins. Player localization using multiple static cameras for sports visualization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 731 – 738, San Francisco, CA, June 2010.

[HKO01] J Haitsma, T Kalker, et J Oostveen. Robust audio hashing for content identification. *Proceedings of the International Workshop on Content-Based Multimedia Indexing*, Brescia, Italy, September 2001.

[HS95] A. G. Hauptmann et M. A. Smith. Text, speech and vision for video segmentation : The infomedia project. *Proceedings of the AAAI Fall Symposium, Computational Models for Integrating Language and Vision*, Cambridge, Massachusetts, USA, November 1995.

[Jac06] Arne Jacobs. Using self-similarity matrices for structure mining on news video. *Advances in Artificial Intelligence*, 3955 :87–94, 2006.

[KCtK+02] Jae-Gon Kim, Hyun Sung Chang, Young tae Kim, Kyeongok Kang, Munchurl Kim, Jinwoong Kim, et Hyung-Myung Kim. Multimodal approach for summarizing and indexing news video. *ETRI Journal*, 24(1) :1–11, 2002.

[KGS+10] Kihwan Kim, Matthias Grundmann, Ariel Shamir, Iain Matthews, et Jessica Hodginsand Irfan Essa. Motion fields to predict play evolution in dynamic sport scenes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 840 – 847, San Francisco, CA, June 2010.

[KLP03] E. Kijak, L.Oisel, et P.Gros. Hierarchical structure analysis of sport videos using hmms. *Proceedings of the International Conference on Image Processing*, volume 3, pages 1025–1028, September 2003.

[KX08] Chien-Chuan Ko et Wen-Ming Xie. News video segmentation and categorization techniques for content-demand browsing. *Proceedings of the Congress on Image and Signal Processing*, pages 530 – 534, Sanya, Hainan, China, May 2008.

[LC00] Beth Logan et Stephen Chu. Music summarization using key phrases. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 749–752, Istanbul, Turkey, June 2000.

[LTW+10] Haojie Li, Jinhui Tang, Si Wu, Yongdong Zhang, et Shouxun Lin. Automatic detection and analysis of player action in moving background sports video sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(3) :351 – 364, March 2010.

[MGB09] Armando Muscariello, Guillaume Gravier, et Frédéric Bimbot. Audio keyword extraction by unsupervised word discovery. *Proceedings of the Conference of the International Speech Communication Association (Interspeech)*, pages 2843–2846, Brighton UK, 2009.

[MHG+10] Hemant Misra, Frank Hopfgartner, Anuj Goyal, P. Punitha, et Joemon M. Jose. Tv news story segmentation based on semantic coherence and content similarity. *Proceedings of the 16th International Multimedia Modeling Conference*, pages 347–357, Chongqing, China, January 2010.

[MMM97] Andrew Merlino, Daryl Morey, et Mark Maybury. Broadcast news navigation using story segmentation. *Proceedings of the ACM international conference on Multimedia*, pages 381–391, Seattle, WA, USA, November 1997.

[NK97] Yuichi Nakamura et Takeo Kanade. Semantic analysis for video contents extraction - spotting by association in news nideo. *Proceedings of the ACM international conference on Multimedia*, pages 393–401, Seattle, WA, USA, November 1997.

[PAO04] Julien Pinquier et Regine Andre-Obrecht. Jingle detection and identification in audio documents. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 329–332, Montreal, Quebec, Canada, May 2004.

[PLE01] Silvia Pfeiffer, Rainer Lienhart, et Wolfgang Efflsberg. Scene determination based on video and audio features. *Multimedia Tools and Applications*, 15(1) :59–81, September 2001.

[Pol07] J-P. Poli. *Structuration automatique de flux televisuels.* Thèse de doctorat, Universite Paul-Cezanne-Aix-Marseille III, 2007.

[PWY04] Rui Cui Peng Wung et Shi-Qiung Yung. Contextual browsing for highlights in sports video. *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1951–1954, Taipei, Taiwan, June 2004.

[RS05] Zeeshan Rasheed et Mubarak Shah. Detection and representation of scenes in videos. *IEEE transactions on multimedia ISSN 1520-9210*, 7(6) :1097–1105, December 2005.

[Sil86] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[SMK+09] Panagiotis Sidiropoulos, Vasileios Mezaris, Ioannis Kompatsiaris, Hugo Meinedo, et Isabel Trancoso. Multi-modal scene segmentation using scene transition graphs. *Proceedings of the 17th ACM International Conference on Multimedia*, pages 665–668, Beijing, China, October 2009.

[TA09] Koji Yamamoto Shunsuke Takayama et Hisashi Aoki. Semantic segmentation of tv programs using corner-subtitles. *Proceedings of the IEEE 13th International Symposium on Consumer Electronics*, pages 205 – 208, Kyoto, Japan, May 2009.

[VRB00] E. Veneau, R. Ronfard, et P. Bouthemy. From video shot clustering to sequence segmentation. *Proceedings of the 15th International Conference on Pattern Recognition*, volume 4, pages 254–257, Barcelona, Spain, September 2000.

[WDL+08] Jinqiao Wang, Lingyu Duan, Qingshan Liu, Hanqing Lu, et Jesse S. Jin. A multimodal scheme for program segmentation and representation in broadcast video streams. *IEEE Transactions on Multimedia*, 10(3) :393 – 408, April 2008.

[XMZY05] Gu Xu, Yu-Fei Ma, Hong-Jiang Zhang, et Shi-Qiang Yang. An hmm-based framework for video semantic analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(11) :1422 – 1433, November 2005.

[XXC+04] Lexing Xie, Peng Xu, Shih-Fu Chang, Ajay Divakaran, et Huifang Sun. Structure analysis of soccer video with domain knowledge and hidden markov models. *Pattern Recognition Letters*, 25(7) :767 – 775, May 2004.

[XZT+06] Ziyou Xiong, Xiang Sean Zhou, Qi Tian, Yong Rui, et Thomas S. Huangm. Semantic retrieval of video - review of research on video retrieval in meetings, movies and broadcast news, and sports. *IEEE Signal Processing Magazine*, 23(2) :18 – 27, March 2006.

[YL95] M. M. Yeung et B. Liu. Efficient matching and clustering of video shots. *Proceedings of the International Conference on Image Processing*, volume 1, pages 338–341, Washington D.C., USA, October 1995.

[YTX07] Xian-Feng Yang, Qi Tian, et Ping Xue. Efficient short video repeat identification with application to news video structure analysis. *IEEE Transactions on Multimedia*, 9(3) :600–609, April 2007.

[YY96] M.M. Yeung et Boon-Lock Yeo. Time-constrained clustering for segmentation of video into storyunits. *Proceedings of the 13th International Conference on Pattern Recognition*, volume 3, pages 375–380, Vienna, Austria, August 1996.

[ZGST94] HongJiang Zhang, Yihong Gong, S.W. Smoliar, et Shuang Yeo Tan. Automatic parsing of news video. *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 45 – 54, Boston, Massachusetts, May 1994.

[ZLCS04] Dong-Qing Zhang, Ching-Yung Lin, Shi-Fu Chang, et John R. Smith. Semantic video clustering across sources using bipartite spectral clustering. *Proceedings of the IEEE International Conference on multimedia and expo ( ICME )*, pages 117– 120, June 2004.

[ZS06] Yun Zhai et Mubarak Shah. Video scene segmentation using markov chain monte carlo. *IEEE Transactions on Multimedia*, 8(4) :686 – 697, August 2006.