Dealing with Television Archives: Television Structuring

Xavier Naturel, Patrick Gros

To cite this version:

Xavier Naturel, Patrick Gros. Dealing with Television Archives: Television Structuring. [Research Report] RR-7301, INRIA. 2010, pp.24. <inria-00507752>

HAL Id: inria-00507752

https://hal.inria.fr/inria-00507752

Submitted on 30 Jul 2010

Dealing with Television Archives: Television Structuring

Xavier Naturel — Patrick Gros

N° 7301

June 2010

Vision, Perception and Multimedia Understanding

ISSN 0249-6399 ISRN INRIA/RR--7301--FR+ENG

Dealing with Television Archives: Television Structuring

Xavier Naturel, Patrick Gros

Theme : Vision, Perception and Multimedia Understanding
Perception, Cognition, Interaction
Équipes-Projets texmex

Rapport de recherche n° 7301 — June 2010 — 23 pages

Abstract: This paper investigates the problem of managing very large digital television archives. This problem is called television structuring (or TV broadcast macro-segmentation) and is defined as the process of identifying the structure of a television stream as watchers perceive it: a succession of programs. This is the very first step in order to manage a television collection. In this paper, a complete solution for television structuring is proposed, which makes use of simple yet efficient methods in order to deal with huge datasets. Methods from commercial detection are generalized to be able to distinguish regular programs from non-programs. It is shown how television program guides can be used to label the identified programs. It is finally shown how an update procedure can improve the segmentation results over time. Results are provided on 3 weeks of French television.

Key-words: Video indexing, Television structuring, macro-segmentation, Perceptual hashing

Structuration vidéo pour les archives de télévision

Résumé : Ce rapport de recherche s'intéresse à la structuration de larges volumes d'archives de télévision. Par structuration, nous entendons l'identification des programmes de télévision, leur début et leur fin, dans le flux, et donc le découpage de ce flux en une succession de programmes. Ceci est la toute première étape dans un processus d'indexation d'un flux de télévision, afin de le rendre facilement naviguable et requêtable. Nous présentons une solution complète basée sur des méthodes simples afin de pouvoir traiter de très importantes quantité de données. Nous généralisons des méthodes provenant de la détection de publicités télévisées afin de distinguer les programmes des inter-programmes. Il est également montré comment les guides de programmes peuvent être utilisés afin d'étiqueter les programmes identifiés. Nous proposons finalement une procédure de mise à jour, qui permet d'obtenir des résultats constants au cours du temps. Des résultats sur trois semaines de télévision française permettent de vérifier l'efficacité des méthodes.

Mots-clés : Indexation vidéo, Structuration de télévision, macro-segmentation, hachage perceptuel

Contents

1 Introduction

Television is an important part of today's source of information, as well as an important part of our cultural heritage. Information retrieval from television streams is however still in its infancy, whereas the amount of television content is increasing, with an ever larger set of available channels. For example, since September 2006, the National Institute of Audiovisual in France (INA) is archiving 540.000 hours of television per year. There is thus a need to develop methods to retrieve information from very large television collections, which might come with few or no additional information apart from the video stream itself.

The context of the work is assumed to be television archives. It means that weeks, months, or years of continuous recordings have to be analyzed to extract relevant information. We assume that television program guides are available together with the video data. While this is actually the case in France at INA, it may be different in other countries. In this context, we are interested in finding the structure of the television stream as TV watchers perceive it: a succession of well-identified programs, together with some non-program events (commercials, trailers, sponsoring...). This is what we call television structuring.

More precisely, the goal is first to perform a segmentation of the television stream into programs and non-programs, and in a second step, to label these programs, with information coming from the program guide. This process may also be viewed as metadata refinement: the process takes as input the video stream as well as some metadata (the program guide), and outputs a corrected version of the program guide, where the given schedules have been checked in order to match with the actual video stream.

This may be seen as a trivial or non-needed problem since the EPG can be thought as self-sufficient. Berrani et al. [1] have shown that this is not the case, and that precise television structuring cannot be achieved only with the provided information coming from channels and/or broadcasters. They assess the necessity of content-based techniques.

Even if our first goal is archiving, quite a large number of applications can benefit from this conceptually simple task of finding programs in a television stream. In the case of television archives, it is obvious that exact program boundaries should be available when browsing or querying a television corpus. Manual structuring is a very tedious and time-consuming task, and automatic or semi-automatic methods should be investigated to reduce the need of manual parsing and labeling. Monitoring television may also be an application, for example to verify legal regulations about commercials, or provide some statistics about delays between the actual broadcast time and the scheduled one. A more user-oriented application might be to manage recorded programs, allowing features like skipping commercials or finding programs that may not be in the EPG[1].

Our approach is stream based and bottom-up. Three kinds of information are mainly used. Joint silence and monochrome image detection allows to find program boundaries and to detect non-program segments. Repetition detection [15] using a reference video dataset (RVD) allows to find similar segments appearing several times in the stream and is useful to characterize many

[1]In the paper, program guides and EPG (Electronic Program Guides) are used as synonyms.

non-programs. Finally, the program guide is used to assign labels to program segments [16].

The paper is organized as follows. Section 2 provides a brief overview of the state-of-the-art. Section 3 presents the structuring technique. Section 4 explains the problem of updating the RVD. Section 5 conludes the work.

2 Previous work

Video structuring for specific programs like sports or news is a well-studied domain. The aim is to infer the structure of the program by analyzing the video and audio streams [10, 6, 2]. Studies have also been conducted on collections of programs, leading to non-obvious tasks like topic threading [9]. These works are dealing with sets of homogeneous programs (mainly news or a specific sport), and are looking for a structure inside the program itself. They thus are quite different from our task, and are not likely to be suited to find program boundaries.

A more relevant topic is commercial detection. This is a well-studied domain, where effective solutions have been proposed to identify commercials in a TV stream. Some simple but effective rule-based methods have been proposed, which use detection of monochrome frames and silence between commercials [12, 14, 19]. Classification techniques have also been proposed [5, 22, 13], as well as techniques based on recognition [8, 4, 21, 18]. Commercial detection is of major importance for television structuring because it can detect non-programs. However, some non-programs are not commercials, and these techniques have thus to be extended to handle all types of non-programs.

Very few works have considered television structuring. Liang *et al.* [11] proposed to detect programs by their lead-in/lead-out. Interesting results are obtained on their dataset but cannot be generalized to other TV channels, which may not flag their programs by systematics lead-in and lead-out.

A very different work is proposed by Poli [17]. The basic idea is to implement a top-down approach using a very large set of already annotated data to learn a model of the TV stream. Poli proposes to use a hidden Markov model and a decision tree for that purpose. The result is a weekly program providing an approximate start time and duration and the type for each program and non-program during the week. This model is eventually checked with the stream.

To the best of our knowledge, these two methods are the only ones dealing with television structuring.

3 Structuring method

3.1 Definitions and overview

First, let us define more precisely the notion of **program** and **non-program**. Programs are regular television broadcasts which make the core of a channel broadcasting (e.g. news, weather forecast, movies, shows...). Non-programs are either commercials, sponsoring, channel promotion (e.g. trailers, jingles).

An overview of the proposed method is shown on figure 1. Three inputs are used: the video stream itself, the program guide, and a Reference Video

Dataset (RVD). This RVD is a dataset composed of manually labeled programs and non-programs, with the following information:

- a category (program or non-program).
- a title

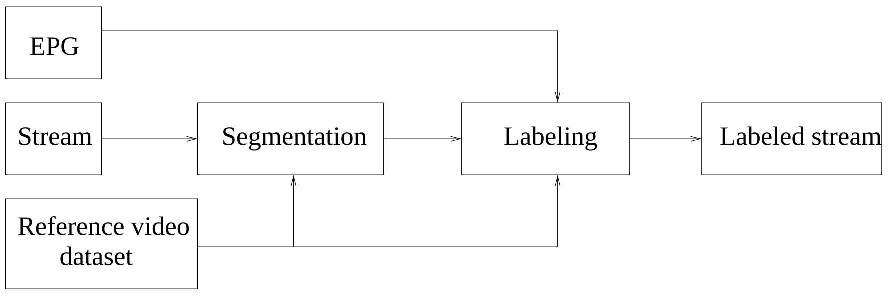

Figure 1: Overview of the structuring method

The structuring method is in two steps: segmentation and labeling. Segmentation cuts the stream into **segments**[2], which are then classified into either program and non-program. Labeling assigns labels to every segment classified as a program.

To test the method, a corpus of three weeks of television has been recorded from a french channel (France2) from 5/9/2005 to 5/30/2005. This corpus is composed of 21 files, each one representing 24h of TV. Manual structuring has been performed on this corpus to obtain ground truth.

3.2 Segmentation and classification

The aim of the segmentation is to find the different segments of programs and non-programs. Methods developed in the context of commercial detection are well suited to make this segmentation. However, as stated in section 2, the task has to be generalized to detect all kinds of non-programs, e.g. commercials, trailers, jingles, sponsoring... Since trailers and jingles have different characteristics from commercials, in terms of shot length and visual activity, approaches based on classification are not likely to perform well. We focus on methods that are able to generalize to all non-programs: recognition-based methods, and joint silence and monochrome frames detection.

3.2.1 Separation detection

We call *separations* simultaneous occurrences of monochrome frames and silence that happen between commercials. This is a very popular feature for detecting commercials [12, 19, 14] and it is used on every French TV channel.

To detect monochrome frames, a 48-bin histogram on the luminance channel is first computed. Detecting monochrome frames is then achieved by thresholding the histogram entropy. For an histogram h quantized into N bins, its entropy

[2]The term segment will be used in the remaining of the paper as the result of this segmentation process.

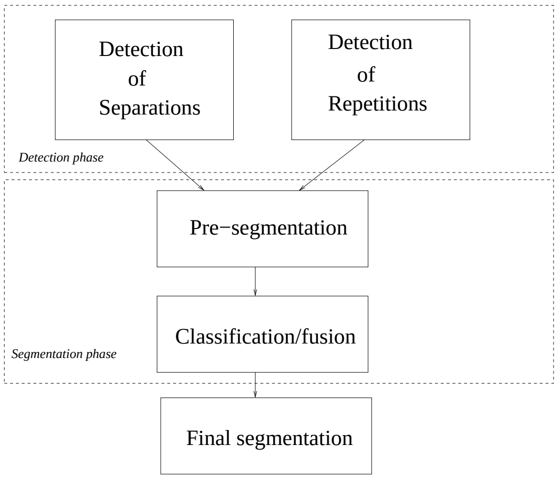

Figure 2: Overview of the segmentation and classification processes

is given by:

$$H = -\sum_{i=1}^{N} p_i \log p_i \qquad \text{with } p_i = \frac{h(i)}{\sum_k h(k)}$$

Figure 3 shows a sample of the histogram entropy on 1 hour of our corpus. The threshold is set experimentally to 2.

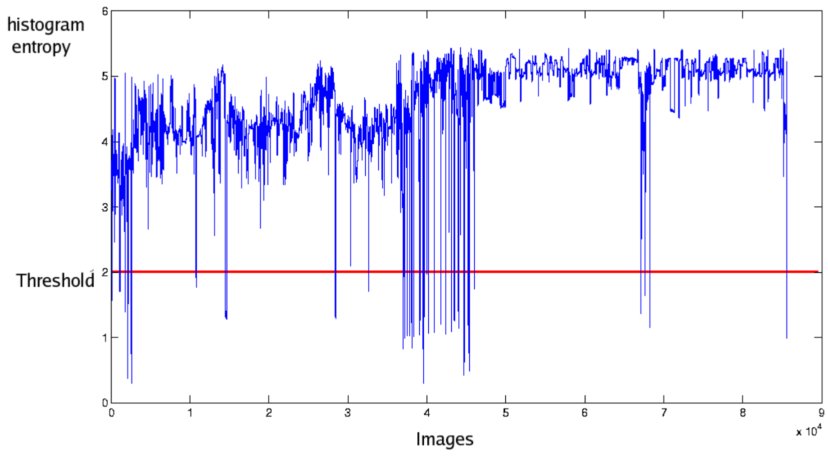

Figure 3: Variation of the luminance histogram entropy on one hour of TV

To detect silence, a very simple method is used. It consists in building overlapping audio frames of 10 ms, and computing the log-energy on each frame using the standard formula:

$$E_{db}(i) = 10\log_{10}\sum_{n=1}^{N} x_n^2(i)$$

The threshold is set to 60 (see figure 4), and only segments longer than 30ms are kept. The reason for using a so simple method is given by figure 4, which shows the variation of energy on 1 hour of TV. One can easily see two separations, where the energy is actually zero. This phenomenon has been observed on every French channel. It might be different in other countries.

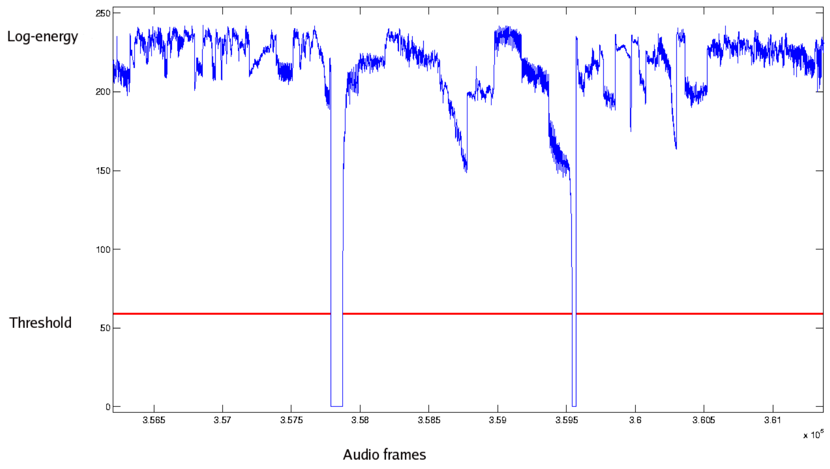

Figure 4: Variation of the audio energy on a few minutes of TV

The results of the silence and monochrome frames detection are then merged using a successive analysis. Since the audio feature is far more discriminative than the image one, the process consists in taking the segments detected by the audio as candidate segment, and then check correctness using the image feature. Results are shown in table 1 where the results are the precision and recall computed over the number of images correctly detected as belonging to a *separation*.

Modality	Precision	Recall
Audio	0.82	0.9
Image	0.41	0.89
Fusion	1	0.9

Table 1: Separation detection results

3.2.2 Repetition detection

Television streams are highly redundant. Detecting repetitions can greatly help to uncover the structure of the stream. In particular, all kinds of non-programs are frequently repeated with no modification except for broadcast and compression noise. Because only minor transformations exist between two instances of the same video clip, it is quite easy to detect those repetitions. However, it is important to be able to deal with a very large database and to have a low complexity. Therefore, a repetition detection method has to put the emphasis on those two aspects. A popular method to achieve both efficency and effectiveness in the context of commercial detection is perceptual hashing [4, 8]. Note that this method is particularly suited in our context because it can deal with all kinds of non-programs.

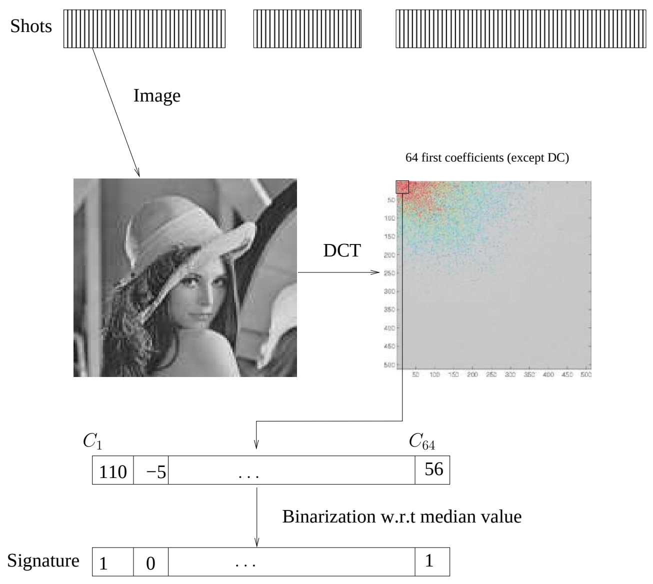
Shots
Image
DCT
64 first coefficients (except DC)
C_1
C_{64}
110
−5
...
56
Binarization w.r.t median value
Signature
1
0
...
1

Figure 5: Image signature computation.

To detect repeated video clips using perceptual hashing the method proposed in [15] is used. Shots are considered as the recognition unit, i.e. we detect repeated shots. A visual signature is built for each image of each shot. The signature extraction is presented on figure 5. For each image, the DCT is applied on the whole image, on the luminance channel only, and the 8x8 top-left submatrix is extracted from the lowest AC frequency coefficients, (DC coefficient is not taken into account). The median value of this sub-matrix is computed, and coefficients are then binarized according to this median, thus making a 64 bits signature.

This signature is sufficiently robust to noise to be queried by exact matching, allowing the use of a fast retrieval structure like a hash table. The retrieval process makes indeed use of a hash table, in which a pair (signature, shot id) is stored for every frame of every shot of the database. When a query is made, i.e. we want to know if a certain shot has a duplicate in the database, signatures of each frame of this query shot are computed, and then queried one by one against the hash table. If an exact match occurs, that is a signature of the query shot s_q is equal to a signature s_d in the hash table, a pair (s_d, shot id) is recovered.

This shot id gives a candidate shot, which is further analyzed by computing a similarity distance between this candidate shot and the query shot.

The similarity distance between shots is defined as the average Hamming distance between the signatures of the retrieved and query shots. This distance makes use of the relative positions of the matched signatures to align the shots, thus gaining robustness to temporal variations and shot segmentation artifacts. To decide if two shots match, this distance is thresholded.

This method is used several times throughout the structuring process. It is used together with the reference video dataset (RVD) previously defined, which is chosen as the first day of our 3 weeks corpus, day 5/9/2005, and was manually labeled.

3.2.3 Classification and fusion

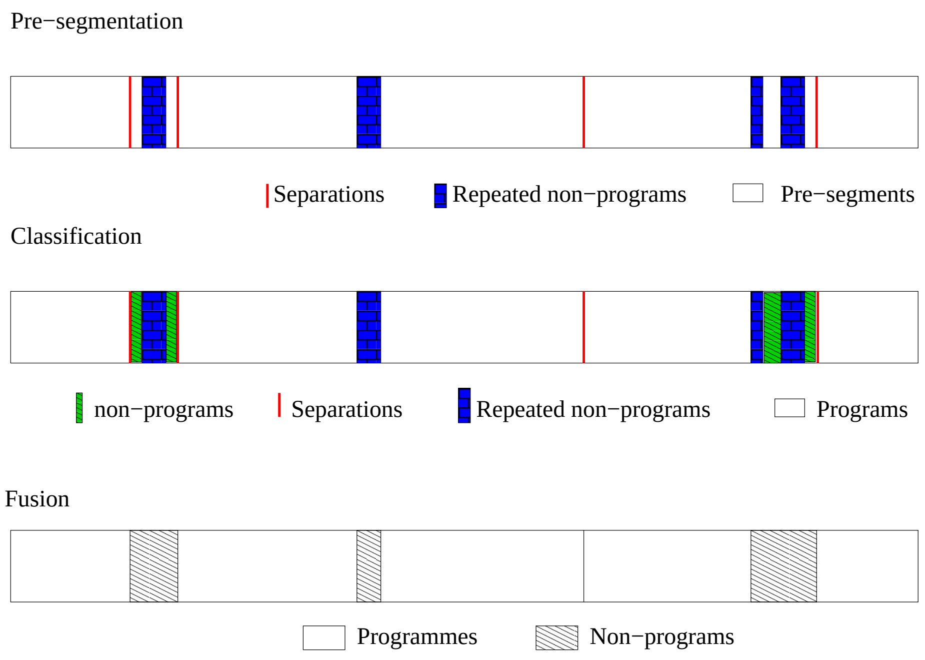

Figure 6: The three steps of segmentation: pre-segmentation, classification, and fusion.

The detection of separations and repetitions yields a pre-segmentation of the stream, as can be seen in figure 6. Once this pre-segmentation is computed, the next step is to classify pre-segments as either program or non-program. The decision is taken by simply thresholding the length of the pre-segment, short pre-segments are classified as non-programs and long ones as programs. The threshold T_s is chosen so as to maximize the F-measure of correct classification on a sample day of our corpus.

Finally, contiguous segments of repetitions, separations, and non-programs are merged into a single non-program segment. Figure 6 sums up the segmentation process.

3.2.4 Results

The segmentation is evaluated as a task of binary classification of images, evaluated by the F-measure. Classification into the class non-program is shown on figure 7. Classification in programs is not given because program frames are so numerous compared to non-program frames that this measure is not really informative (superior to 99%). This figure shows that results are quite good in average ($\approx$ 90%), but are decreasing over time. This can be explained by the fact that there is a very high number of repetitions between two consecutive days, but that this number is much lower for temporally distant days. Since the RVD used for detecting the repetitions has been recorded on May 9th 2005, the number of repetitions decreases over time and so is the quality of the segmentation.

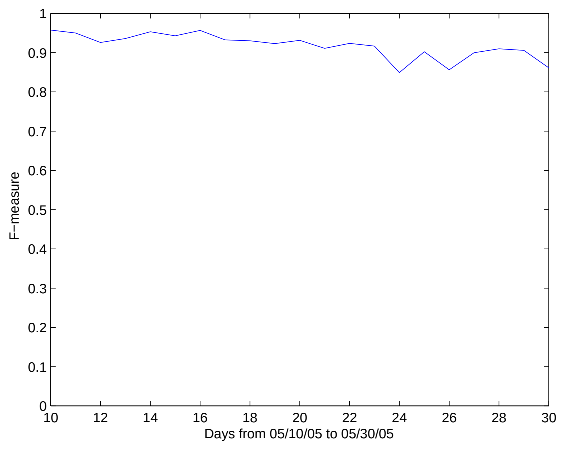

Figure 7: F-measure of retrieval of non-program images on 20 days of TV.

3.3 Labeling

3.3.1 Alignment using DTW

Now that the stream is segmented, the next step is to add information by labeling the segments and especially the programs. The idea is to use the program guide, which either directly comes with the video stream, in case of live digital television, or which may also be archived in case of TV archives (e.g. at INA). The program guide provides useful information about programs, like title, genre, and sometimes other information such a short description, a list of actors...

It has been proposed in [16] to align the program guide with the segmentation using Dynamic Time Warping (DTW). The DTW is a well-known method [20] which computes a path and a distance between 2 sequences X and Y. This distance can be interpreted as the minimum cost to transform X into Y by a set of weighted edit operations. These operations are usually substitution, insertion and deletion. The path of minimum cost provides the best alignment between X and Y, with respect to the edit operations.

For two given sequences $X = (x_0 \dots x_N)$ and $Y = (y_0 \dots y_M)$, this path is computed via a cost matrix D, which is computed recursively using the formula:

$$D(i,j) = \min \begin{cases} D(i-1, j-1) + c_{sub}(x_i, y_j) \\ D(i, j-1) + c_{del}(x_i, y_j) \\ D(i-1, j) + c_{ins}(x_i, y_j) \end{cases}$$

where $c_{sub}, c_{del}, c_{ins}$ are the costs for the operations of substitution, deletion and insertion respectively. This matrix can be efficiently computed by dynamic programming.

The final value $D(N, M)$ is the distance between X and Y. To recover the best path from the computation of the cost matrix, paths have to be stored in a path matrix P, indicating which edit operation has led to the result at each step. The path is then easily found going backwards, i.e. from $P(N, M)$ to $P(0, 0)$.

In our case, the costs are defined as a distance between a segment from the program guide and a segment from the automatic segmentation. A segment x is a couple of values indicating the start and end of the program $x = (x_s, x_e)$. The distance between a segment x and a segment y is:

$$d(x,y) = \underbrace{|x_e - x_s - (y_e - y_s)|}_{\text{similarity of length}} + \underbrace{|x_e - y_e| + |x_s - y_s|}_{\text{similarity of start/end time}}$$

This distance is a sum of two terms, the first one measuring the similarity of length of those segments, and the second one measuring the difference between their (supposed) time of broadcast. The costs of substitution C_{sub}, deletion C_{del}, and insertion C_{ins} are then defined as:

$$\begin{aligned} C_{sub}(x_i, y_j) &= \gamma d(x_i, y_j) \\ C_{del}(y_j, i) &= d(x_i, y_j) \\ C_{ins}(x_i, j) &= d(x_i, y_j) \end{aligned}$$

$1 < \gamma < 2$ to favor a substitution over a deletion plus insertion.

3.3.2 Improvements

So far, the labeling process does not use the repetition information, only the program guide. Two improvements to the DTW labeling are proposed in order to take this information into account.

The first improvement constrains the path of the DTW so that it has to go through a **landmark**. The resulting process is called landmarked DTW or LDTW. A landmark is defined as a segment which contains a repetition whose label is equal to the label of a close program in the program guide. Intuitively, this means that the repetition has confirmed the information proposed by the program guide and the label is thus more than likely to be correct. This is done by filling the cost matrix with infinite values for the impossible paths, and set the landmark to a null cost, prior to the dynamic programming procedure. Computation of the cost and path matrices and the backtracking step are unchanged.

The second improvement is a post-process, it takes place after the LDTW, once a set of labels have already been assigned to segments. It is designed to look for specific locations where a repetition is present, and where the label from the repetition and the label assigned by the LDTW are contradictory. A Bayesian hypothesis test is used to decide which label is more likely to be correct, the one coming from the repetition (Hypothesis H_0) or from the LDTW (Hypothesis H_1). The test is:

$$\frac{P(O|H_1)}{P(O|H_0)} > \frac{P_0}{P_1} \text{ then } H_1 \text{ else } H_0$$

where P_i is the apriori probability of hypothesis H_i. To estimate $P(O|H_i)$, observation O is considered to be composed of three elementary independent observations: the length of the segment, a binary variable indicating whether a repetition is present at the beginning of the segment, and another binary variable for the end of the segment. The idea behind these two binary variables is that repetitions at the boundaries are often correct, because they are lead-in or lead-out of the program. Conditional probability of the length is modeled by a Gaussian and is estimated by maximum likelihood, binary variables are estimated through simple counting. The training set is one day long. An example

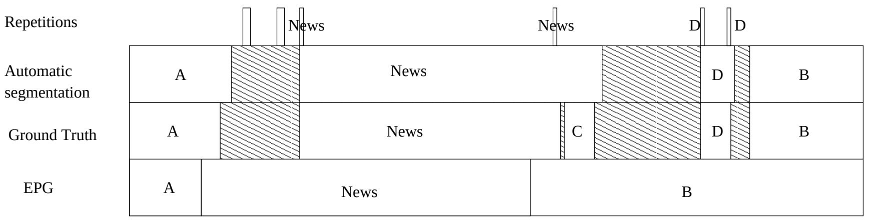

Figure 8: An example of alignment with the EPG, with the help of repetitions. French names of programs are replaced by letters (A, B, C, D).

of alignment using all these improvements is given in figure 8. Note that two programs (C and D) are not present in the EPG. The correct labeling of program D is thus obtained thanks to the repetition information and the hypotheis testing presented above.

3.3.3 Results

The quality of the labeling is evaluated by 2 measures. The first one is image-based: the number of correctly labeled images is counted, together with the number of wrongly labeled images, and the number of non-labeled images. With these 3 numbers, the precision, recall and F-measure are computed. The second measure is program-based and follows the same principle.

For both measures, the results are computed on our 20 days corpus, and shown on figure 9. The varying parameter is T_s, the threshold used for classification of segments into programs or non-programs. The difference between the program and the image measure comes from the fact that a high number of small programs (5 to 15 minutes) are wrongly labeled, which thus penalizes the program measure, but not the image one.

Figure 10 shows the improvements obtained by the different proposed improvements: the LDTW and the post-processing step (LDTW2). It can be seen

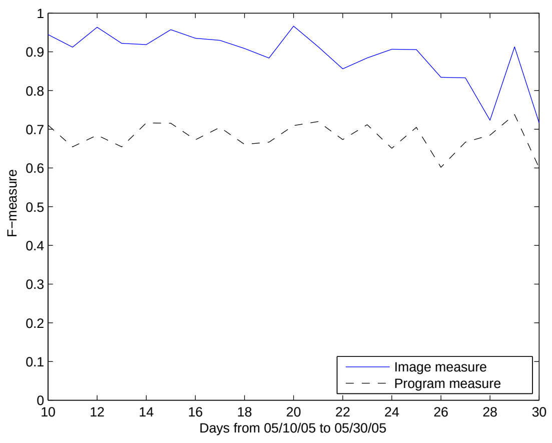

Figure 9: F-measure of correct labeling from day 05/10 to 05/30

that the post-procesing step is especially useful for labeling small programs, since there is a large increase of reesults with the program measure with this scheme.

Compared to the segmentation results, the labeling results seem to decrease much more slowly. This is due to the fact that the age of the reference video dataset (RVD) has much less influence over labeling than over segmentation. The useful segments for labeling are recurrent parts of programs, e.g. a lead-in, which usually do not vary over time. These results also show that the quality of the segmentation does not really affects the quality of the labeling.

There are still issues about missing labels. We computed that 26% of the programs were not announced in the program guide. Additional labels may come from the manual labeling of the RVD but this is not a satisfactory solution. Indeed, using the proposed solution, there are still 10% of labels missing. On the other hand, it was measured that there is an average difference of 7 minutes between the scheduled time and the actual time of broadcast. This difference is reduced to 4 seconds with our solution.

The next section investigates how the quality of the segmentation may be kept constant over time by updating the RVD.

4 Updating the reference video dataset

Using repetitions has an important drawback: the reference video dataset (RVD) has to be up-to-date. The last section showed that the results were decreasing over time if the RVD was static. To keep the RVD up-to-date, one must analyse new video streams, detect new non-programs in those, and add them to the RVD. Only a few works have tackled this problem in the context of commercial detection [12, 7].

The integration of the update procedure into the global structuring process is illustrated by figure 11. Note that the update does not need to be synchronized with the structuring, it can be done for example once a day or once a week.

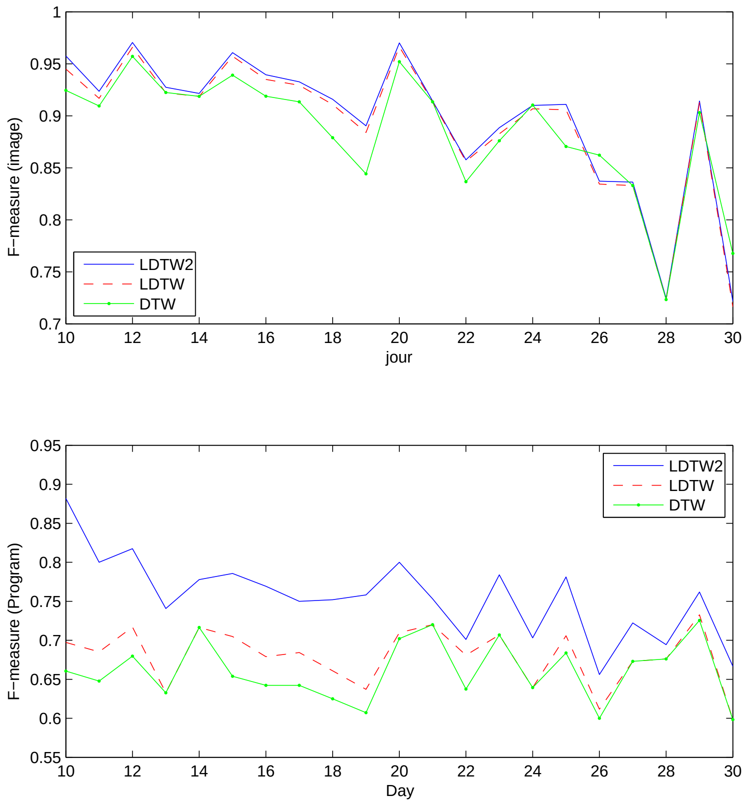

Figure 10: Labeling results for DTW, LDTW and LDTW with post-processing (LDTW2)

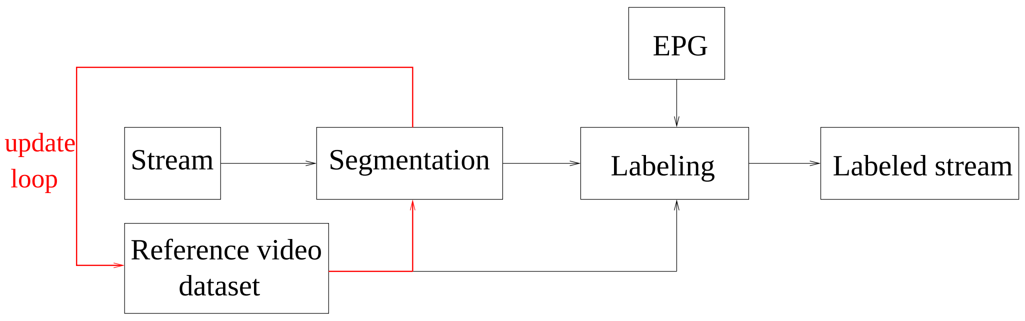

Figure 11: Integration of the udpdate procedure in the global structuring process

The RVD update procedure needs basically two things: the RVD itself, and a input video stream, in which to find new information. All previous steps, segmentation, classification and labeling are applied to this input video stream, so that it is segmented and labeled. The update process is mainly about analyzing the parts of the stream that are still unlabeled after this process. The next sections explain how to achieve this RVD update using this processed input video sream.

4.1 Identification of unknown non-programs

To update the RVD we must find unknown non-programs in the input video stream, that is to say, non-programs that are not in the RVD. The idea is to use the results of the segmentation/classification procedure which has been applied to the input video stream. During this classification, some pre-segments are inferred as being non-programs and are not present in the RVD (they would have been detected as repetitions otherwise). These segments are thus interesting candidates for updating the RVD.

This update can be iterative: the update step can be followed by another segmentation/classification of the input video stream, in which the updated RVD is used to detect repetitions. These two steps can be iterated until the process converges, there are no new non-programs to infer: the segmentation is stable.

	Non-programs	
Method	Precision	Recall
Without update	98.7	85.9
With update	45	98.5

Table 2: Comparison of averaged segmentation results on our 3 weeks dataset

Table 2 shows the results of this iterative scheme. The precision has decreased dramatically, it is obvious that something is wrong. This is due to trailers. Shots belonging to a trailer can be repeated in two different contexts: a repetition of the trailer itself (case A), or in the program that the trailer announces (case B). This configuration is shown in figure 12, where it can be seen that in case B, the repeated shots will segment the program into many small segments, thus producing over-segmentation.

This problem does not occur in the segmentation phase presented in section 3.2.3, because existing trailers in the RVD are labeled. If the trailer label is known, a simple rule is enough to prevent over-segmentation. This rule is the following. Suppose that using the process of section 3.2.2 some shots in the input stream are detected as being repetitions of some existing shots in the RVD. Suppose now that in the RVD those shots are labeled as trailers, with label say Z. We then look in the program guide, in the temporal neighborhood of the repetitions, for a program with label Z. If such a program is found, it is case B, and segmentation is not performed. Otherwise, in case A, segmentation is performed as described in section 3.2.

To solve the over-segmentation problem, there are two solutions:

- identify trailers in the input video stream, label them and add them to the RVD.
- identify trailers in the input video stream, and do not add them to the RVD

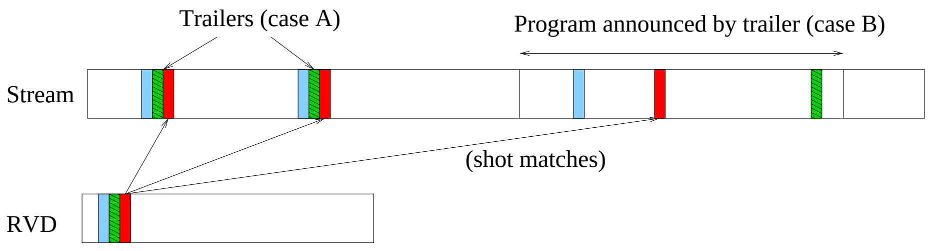

Figure 12: The problem of shots from trailers: two possible ways of being repeated.

In any case, the first step is to be able to further analyze the input video stream, in order to identify trailers. This is explained in the next section.

4.2 Identifying trailers

Identifying trailers in the inferred segments is done in two steps. The first one is to organize the inferred segments into sequences.

4.2.1 Identifying sequences

A sequence is a set of contiguous shots which forms a semantically homogeneous set (e.g. a commercial, a trailer are examples of a sequence).

These sequences are detected using a method coming from natural language processing (NLP): collocations. Collocations are sequences of words that often co-occur. They are usually estimated by a bigram model (sequences of 2 words) and a measure of association, which statistically measures if the co-occurrences are significant. A classical measure is the mutual information [3], defined for words x and y as:

$$I(x,y) = \log_2 \frac{p(x,y)}{p(x)p(y)}$$

The straightforward analogy words $\leftrightarrow$ shots allows to use this model. However, there are some differences with the context assumed in NLP. First, the estimation of collocations is not as obvious as with words. It is done using the repetition detection method of section 3.2.2 over the 3 weeks corpus. This is a noisy process, some shots may be missed, estimation of collocations may be erroneous. Second, the alphabet is infinite, and sequences may be very long: up to 30 or 40 shots.

In our context, collocations are shots which co-occur more often then it would have been expected. Because of the complexity of finding co-occurrences, the estimation of collocations is done with a bigram model. Using a measure like mutual information may be problematic because of the difficulty of estimating the probability of a shot, considering that the alphabet is infinite. To overcome this, and to be able to detect long sequences, it is proposed to view an inferred segment as a concatenation of sequences generated by a left-right Markov chain with a terminal state, as can be seen in figure 13.

A state x_t of the chain is a shot, characterized by a set of successors $S(x_t) = \{(x_{k_i}, \beta_t^{k_i})\}_{i=1\dots n_\beta^t}$ and predecessors $P(x_t) = \{(x_{p_i}, \alpha_t^{p_i})\}_{i=1\dots n_\alpha^t}$. The scalar

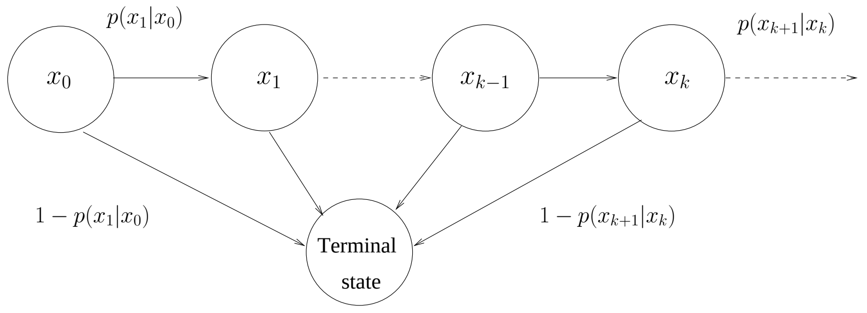

Figure 13: Left-right Markov chain for finding sequences

$\alpha_t^{p_i}$ represent the number of occurrences of state x_{p_i} as a predecessor of current state x_t. Similarly, $\beta_t^{k_i}$ represent the number of occurrences of state x_{k_i} as a successor of state x_t. The space state is the set of shots from the considered inferred segment(s).

Using these definitions, the number of different predecessors for state x_t is n_α^t and the total number of predecessors is:

$$N_\alpha^t = \sum_{i=1}^{n_\alpha^t} \alpha_t^{p_i}$$

The total number of successors is defined in the same way by:

$$N_\beta^t = \sum_{i=1}^{n_\beta^t} \beta_t^{k_i}$$

The transition probability is defined by:

$$p(x_t|x_s) = \begin{cases} \frac{\alpha_t^s + \beta_s^t}{N_\alpha^t + N_\beta^s} & \text{if } s = t-1 \\ 0 & \text{otherwise} \end{cases}$$

To identify the sequences using this model, the inferred segment is parsed in the temporal order. The initialisation of a start state sequence occurs by thresholding the transition probability $p(x_t|x_{t-1}) > \gamma$. Once the process is started, a first solution could be to decide that vector $(x_0, \ldots, x_k)$ is a sequence if $p(x_0 \ldots x_n) > \gamma^{n+1}$. This can be easily computed, since from the Markovian property we have:

$$p(x_0 \ldots x_n) = p(x_0) \prod_{k=1}^{n} p(x_k|x_{k-1})$$

taking the logarithm to facilitate computations, the decision becomes:

$$\log p(x_0) + \sum_{k}^{n} \log p(x_k|x_{k-1}) > (n+1) \log \gamma$$

Unfortunately, this decision is not precise enough to identify the precise boundaries. A simpler solution is to take a decision at each time instant t instead of a global decision, simply by thresholding: $p(x_t|x_{t-1}) > \gamma$. An heuristic on the number of hypothesis can be also used in order to increase recall, a high number of hypothesis is indeed an indication that this is the end of sequence. The decision, taken at each time instant is then:

$$(p(x_t|x_{t-1}) > \gamma) \textbf{ and } \left(n_\alpha^t + n_\beta^{t-1} < \delta\right)$$

Thresholds are set to $\gamma = 0.1$ and $\delta = 6$.

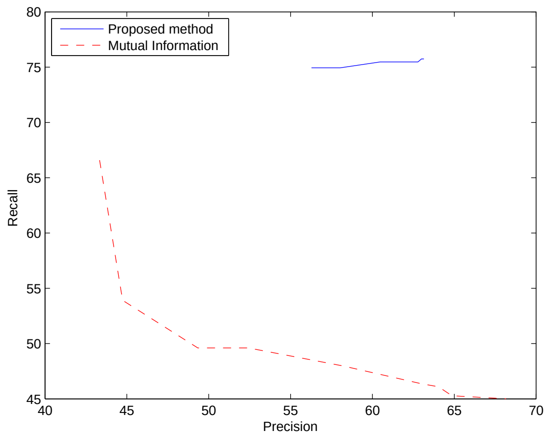

Figure 14: Results of identifying sequences for two measures of association

Results of the proposed method are shown in figure 14 for the proposed method and the classical method using mutual information as the measure of association. The evaluation is done in the same way as a shot segmentation, i.e. precision and recall are computed on the boundaries of the sequences. The varying parameter is γ (δ does not really change the results).

4.2.2 Identifying and labeling trailers

The last step allowed us to find sequences of shots in the input video stream. The next step is to identify which of these sequences are trailers. The idea is to use the specific property of trailers explained in section 4.1 and can which be seen in figure 12.

The algorithm 1 explains the method in details. Briefly, it takes a sequence as input, and look for any repetition of each of its shots in the input video stream. For each detected repetition, the algorithm tries to identify if we are in case A or case B, as defined in section 4.1. Case A is not interesting, it is merely a repetition of the sequence. Case B is interesting, because it is a case where shots of a sequence are repeated with a different pattern, and is thus a strong hint that the sequence currently tested is a trailer.

The final decision is taken with respect to set of hypothesis h[3] and their respective number of votes v. The hypothesis with the maximum number of

[3]The set of hypothesis h corresponds to the variable *SequenceLabels* in algorithm 1.

votes is chosen. More precisely, the sequence is said to be a trailer of program h_k , $k = \arg\max_i v_i$ if:

$$\frac{v_k}{\sum_i v_i} > \alpha \text{ and } v_k \geq 2$$

Threshold α is set to $\frac{1}{2}$.

Algorithm 1: Pseudo code for identifying and labeling trailers.

```
Data: Sequence: list of $M$ shots
Data: VideoStream: list of $N$ shots
Data: labels: list of labels
Function IdentifyTrailers(Sequence, VideoStream);
i,j,k,f: integer;
foreach shot $s_k$ in Sequence do
    [shotFound index] = find repetition of shot $s_k$ in VideoStream;
    if shotFound then
        VideoStream(index) is the found repetition, check now
        if the entire sequence is repeated, by checking shot by
        shot;
        i = index;
        for j=1 to M do
            if $Sequence[j]$ equals $VideoStream[i - k + 1]$ then
                Nb_of_common_shots++;
            end
            i++; j++;
        end
        if Nb_ of_ common_ shots $\geq 0.8M$ then
            it is a repetition of Sequence (case A) Do Nothing
            else
                possibly case B, store label of program, (if
                exists, if VideoStream[f] does not appear in a
                program then there is no hypothesis and thus no
                vote) or add one vote if already there
                SequenceLabels $\leftarrow$ Label of VideoStream[f];
                Votes[Label of VideoStream[f]]++;
            end
        end
    end
end
Now take the label with the max number of votes in vector
Votes
```

Note that this algorithm allows not only to identify trailers, but also to find their labels. This is actually very interesting because this can overcome the over-segmentation problem caused by incorrectly or non-labeled trailers, using heuristic defined in section 4.1. One shortcoming is that the segmentation and/or labels of the input video stream video stream might be uncorrect.

4.3 Update procedure and results

A final decision must be taken on which segments can be used to update the RVD. There are three types of segments: labeled sequences (type 1), non labeled sequences (type 2), isolated shots (type 3). Three strategies of update can be defined, from the most conservative to the most aggressive:

1. Type 1: Add only segments of type 1
2. Type 2: Add segments of type 1 and 2
3. Type 3: Add all kinds of segments (1, 2 and 3)

The results of these different strategies is given in figure 15. Segmentation/classification results are given by the two upper part figures, and the results are not surprising: the most "daring" method (Type 3) has the highest recall, but also the lowest precision, and conversely, the method with no update has the lowest recall and the highest precision. The choice of one of the method may depend on the application and the desired balance between recall and precision. One interesting thing to note is that the recall does not seem to decrease over time for methods of type 2 and 3.

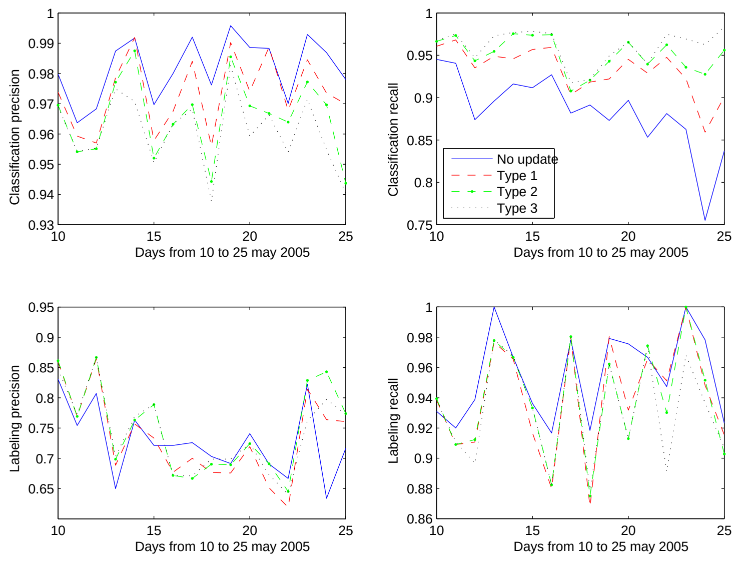

Figure 15: Comparaison of update methods. Segmentation results are in the upper part, labeling results in the lower part of the figure.

Labeling results are given by the two lower part figures, where the F-measure is computed using the program measure. These results are really close and it is difficult to find any clear improvement. Results can even decrease, due to the over-segmentation artifact caused by some mislabel or undetected trailers, which make the alignment procedure by DTW very difficult. On the other hand, it was shown in section 3.3 that labeling results did not decreased over time. The update step had therefore just to keep the labeling results stable.

5 Conclusion

A complete process of television structuring has been proposed. Methods from commercial detection are used and generalized to take into account all kinds of non-programs. Using a pre-labeled reference video dataset, repetition detection and separation detection, a method is presented to segment the stream into program and non-program segments. An alignment procedure is then proposed to label these segments, using dynamic time warping and the program guide. Eventually, a method to update the reference video dataset is proposed, which has to take into account some specificities about trailers. Good results are obtained both in terms of segmentation and labeling over a three weeks dataset from French television.

Limitations come from the need of initial manual labeling of the RVD, which is only partially solved by the update method. There is also a lack of label information due to the high imprecision of the program guide. A solution could be to use screen text as an additional source of labels.

References

[1] Sid-Ahmed Berrani, Patrick Lechat, and Gaël Manson. Tv broadcast macro-segmentation: metadata-based vs. content-based approaches. In *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 325–332, New York, NY, USA, 2007. ACM.

[2] J. Boreczky and L. Wilcox. A hidden markov model framework for video segmentation using audio an image features. In *Proc. IEEE ICASSP*, Seattle, 1998.

[3] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, pages 76–83, Morristown, NJ, USA, 1989. Association for Computational Linguistics.

[4] Michele Covell, Shumeet Baluja, and Michael Fink. Advertisement detection and replacement using acoustic and visual repetition. In *MMSP'06, IEEE 8th workshop on Multimedia Signal Procesing*, October 2006.

[5] P. Duygulu, Ming-Yu Chen, and Alex Hauptmann. Comparison and combination of two novel commercial detection methods. In *ICME*, Taipei, Taiwan, june 2004.

[6] Stefan Eickeler and Stefan Müller. Content-Based Video Indexing of TV Broadcast News Using Hidden Markov Models. In *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2997–3000, Phoenix, 1999.

[7] John M. Gauch and Abhishek Shivadas. Finding and identifying unknown commercials using repeated video sequence detection. *Comput. Vis. Image Underst.*, 103(1):80–88, 2006.

[8] John M. Gauch and Abhishek Shivadas. Real-time commercial recognition using color moments and hashing. In *ACM SIGMM Int. Workshop on Multimedia Information Retrieval*, Santa Barbara, CA, USA, 2006.

[9] Ichiro Ide, Hiroshi Mo, and Norio Katayama. Threading news video topics. In *MIR '03: Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, pages 239–246, New York, NY, USA, 2003. ACM Press.

[10] E. Kijak, G. Gravier, L. Oisel, and P. Gros. Audiovisual integration for sport broadcast structuring. *Multimedia Tools and Applications*, 30:289–312, 2006.

[11] Liuhong Liang, Hong Lu, Xiangyang Xue, and Yap-Peng Tan. Program segmentation for tv videos. In *ISCAS, IEEE International Symposium on Circuits and Systems*, volume 2, pages 1549–1552, 2005.

[12] R. Lienhart, C. Kuhmunch, and W. Effelsberg. On the detection and recognition of television commercials. In *International Conference on Multimedia Computing and Systems*, pages 509–516, 1997.

[13] Tie-Yan Liu, Tao Qin, and Hong-Jiang Zhang. Time-constraint boost for tv commercials detection. In *ICIP*, pages 1617–1620, 2004.

[14] T. McGee and N. Dimitrova. Parsing tv program structures for identification and removal of non-story segments. In *in SPIE Conf. on Storage and Retrieval for Image and Video Databases*, 1999.

[15] X. Naturel and P. Gros. A fast shot matching strategy for detecting duplicate sequences in a television stream. In *CVDB'05*, pages 21–27, Baltimore, june 2005.

[16] Xavier Naturel, Guillaume Gravier, and P. Gros. Fast structuring of large television streams using program guides. In *4th International Workshop on Adaptive Multimedia Retrieval (AMR)*, Geneva, Switzerland, 2006.

[17] Jean-Philippe Poli and Jean Carrive. "modeling television schedules for television stream structuring. In *Proceedings of ACM MultiMedia Modeling MMM*, pages 680–689, Singapour, january 2007.

[18] Kok Meng Pua, John M. Gauch, Susan E. Gauch, and Jedrzej Z. Miadowicz. Real time repeated video sequence identification. *Comput. Vis. Image Underst.*, 93(3):310–327, 2004.

[19] D. Sadlier, S. Marlow, N. OConnor, and N. Murphy. Automatic tv advertisement detection from mpeg bitstream. *Journal of the Patt. Rec. Society*, 35:2–15, 2002.

[20] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

[21] Juan María Sánchez, Xavier Binefa, and Jordi Vitrià. Shot partitioning based recognition of tv commercials. *Multimedia Tools Appl.*, 18(3):233–247, 2002.

[22] Hong-Jiang Zhang Xian-Sheng Hua, Lie Lu. Robust learning-based tv commercial detection. In *ICME*, july 2005.

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399