Signal Processing: *Image Communication* 23 (2008) 525–537

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

A non-supervised approach for repeated sequence detection in TV broadcast streams

Sid-Ahmed Berrani *, Gaël Manson, Patrick Lechat

Orange Labs—France Telecom, Division R&D—Technologies, 4, rue du Clos Courtel, 35510 Cesson-Sévigné, France

ARTICLE INFO

Article history:
Received 25 April 2008
Accepted 29 April 2008

Keywords:
Video indexing
TV broadcast macro-segmentation
TVoD service
Clustering

ABSTRACT

In this paper, a novel method for repeated sequence detection in an audio-visual TV broadcast is proposed. This method is required for TV broadcast macro-segmentation which is at the root of many novel services related to TV broadcast and in particular to the TV-on-Demand service. Repeated sequence detection allows inter-program detection (commercials, jingles, credits, ...), which allows the segmentation of the TV broadcast and the extraction of useful programs. Our method is completely non-supervised, that is, it does not require a manually created reference database. It relies on a micro-clustering technique that groups similar audio/visual feature vectors. Clusters are then analyzed and repeated sequences are detected. This method is able to continuously analyze the TV broadcast and to periodically return analysis results. The efficiency and effectiveness of the method have been shown on two real broadcasts of 12 h and 7 days.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

The significant increase in the amount of digital video content (TV channels in particular) and the diversification of broadcast possibilities and storage devices have recently given rise to the emergence of many new services and novel TV program consumption schemes. These new services are basically aimed at making audio-visual content available to users without any constraints on location and/or time.

In particular, a highly interesting emerging service is TV-on-Demand (TVoD). This service allows users to watch TV in a non-linear manner. The principle is to give them the possibility to access past TV programs from a potentially high number of channels and to compose their own program schedules. Basically, this service requires the macro-segmentation of the audio-visual TV stream and the extraction of *useful* (or relevant) programs. These programs then have to be classified into a set of predefined categories and have to be annotated in order to allow users to browse through the set of provided programs.

Due to the high number of channels and the heterogeneity of the audio-visual content, these processing steps, in particular TV stream macro-segmentation and program extraction, are highly time consuming. Traditional approaches based on manual annotation, even the best in terms of accuracy, are barely conceivable in a real world service.

Currently, TV channels offering a TVoD service through dedicated websites only provide a small selection of their past programs that have been manually prepared. Even though the market of "on-demand" audio-visual content is growing very quickly, the TV content provided is still very limited. The complexity of copyright rules on digital content and the fact that they differ from one country to another are the main reasons behind that. The other reasons are related to the technical limitations and the

* Corresponding author. Tel.: +33 2 99 12 49 37; fax: +33 2 99 12 40 98.
E-mail addresses: sidahmed.berrani@orange-ftgroup.com (S.-A. Berrani), gael.manson@orange-ftgroup.com (G. Manson), patrick.lechat@orange-ftgroup.com (P. Lechat).

0923-5965/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2008.04.018

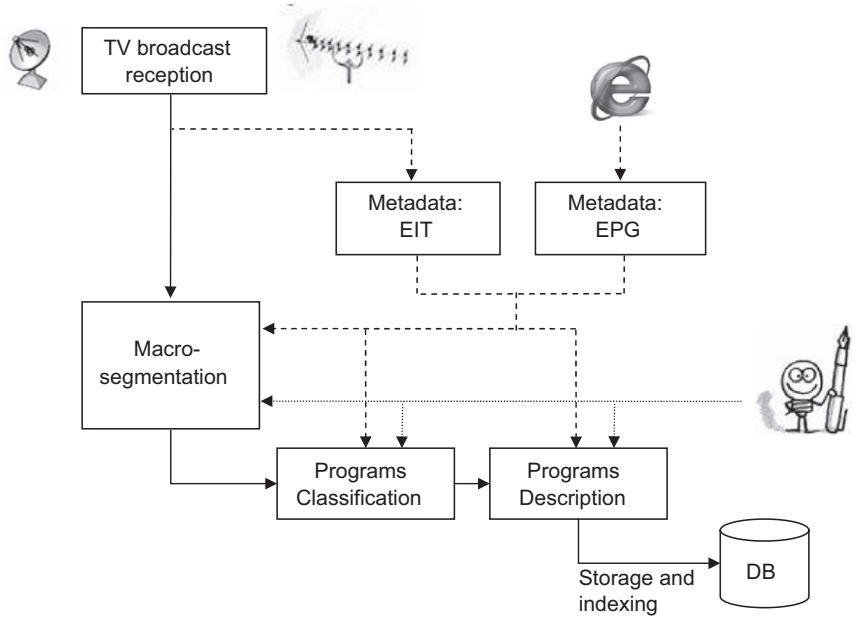

Fig. 1. Program extraction and description for a TVoD service.

highly time consuming processes that are required to build a TVoD service. Indeed, the service is useful and profitable only if it covers a large number of TV channels and a wide range of programs.

The general scheme of a system for TV broadcast macro-segmentation toward building a TVoD service is depicted in Fig. 1. The audio-visual stream possibly comes with a metadata stream in the form of Event Information Table (EIT).[1] These metadata contain information such as the approximate start and end times, the title, and possibly a summary of the program currently being broadcasted and the following one. Other kinds of metadata like the Electronic Program Guide (EPG) could also be retrieved from the web.

This paper focuses on the first and critical step in the process of building a TVoD service, that is TV stream macro-segmentation. This step is aimed at precisely segmenting and extracting useful programs from the audio-visual stream. It allows any user support to be discarded, and automatically handles hundreds of TV channels.

The simplest and most straightforward way to achieve audio-visual stream macro-segmentation is to make use of metadata (EIT). However, these metadata are not always available. They are moreover imprecise, incomplete, inconsistent and static (i.e. late modifications of the program schedule are not taken into account). A deeper analysis of weaknesses of metadata-based approaches is given in [5]. In [5], we have experimentally shown that these approaches are unpractical and cannot be directly used in a real-world application. We have shown in particular that over a 24 h broadcast, more than 40% of the programs start more than 5 min earlier or later than that expected from the metadata.

Therefore, one can think that TV channels can solve the macro-segmentation problem as they can *know* the actual start and end times of their broadcasted programs. In other words, TV channels can provide accurate EIT metadata. Unfortunately, this is far for being easy in practice for two main reasons:

(1) The audio-visual chain involves too many participants. Some of them are internal to the channels, others are responsible for delivery, and the interaction and the cooperation between these participants are not standardized. Hence, even if metadata are properly included, there is no guarantee that they will remain coherent and complete until the end of the chain. This is due to the technical complexity of the broadcasting process. Therefore, relying on channels to solve this problem depends on their willingness and on a very expensive and huge effort of standardization.
(2) The TVoD service could be implemented within a Network Personal Video Recorder (NPVR) service. In this case, TV programs are segmented, recorded and stored on a server of the final provider of the TV stream in the broadcasting chain. They are then made available as if the user has asked to record the whole TV broadcast. Hence, all of the required TV stream processing steps are performed at the end point of the audio-visual chain. TV channels are not involved at all with the service.

Few recent works have investigated how to solve the macro-segmentation problem using visual and audio content-based techniques. A detailed state-of-the-art is given in Section 2. Broadly speaking, automatic audio-

[1] Analog TV being in the process of becoming defunct, we restrict our study to digital TV. We therefore do not consider former broadcasted metadata like PDC.

visual stream macro-segmentation relies on detecting what are commonly called "inter-programs", which include commercials, trailers, jingles and credits. These are easier to detect automatically than long useful programs. Indeed, inter-programs have a set of common properties related to their duration, visual and audio content, ... whereas long useful programs are heterogeneous and generally do not share any common feature. The idea is then to detect inter-programs and to deduce long useful programs as the rest of the stream. Metadata (like EIT or EPG), when available, can be used afterward, in the following steps of processing to annotate programs, for instance. *The inter-program detection problem is therefore one of the main and key problems to be solved in order to build a fully automatic TV stream macro-segmentation system.*

The main contribution of this paper is a novel inter-program detection method. Like the techniques proposed by [10,22,16], our solution also relies on detecting inter-programs as *repeated sequences* in the stream. Indeed, the common feature of all kinds of inter-programs is the fact that they are broadcasted several times a day, and even several times a hour for some of them. Our solution is based on a novel clustering-based method. It addresses limitations and shortcomings of [10,22,16], that are mainly related to efficiency issues and strong hypotheses on the size of detected repeated sequences and their frequency (cf. Section 2). Our technique is also able to match trailers with the corresponding announced programs. This is highly interesting for structuring audio-visual stream as it contributes to automatically labeling long programs if no metadata is available: the title of the program is very well highlighted in the trailer and can be deduced using text detection and recognition techniques (OCR) [32,24]. Large-Vocabulary Continuous Speech Recognition methods [17] could also be used on the audio comment of the trailer. The trailer also provides a very good summary of the associated program, which would save the efforts of automatically creating summaries that can never be of the same quality as those created by professionals.

The rest of the paper is organized as follows. Section 2 presents existing techniques for audio-visual stream macro-segmentation and focuses on repeated sequence detection methods. Section 3 presents the proposed solution for repeated sequence detection. Section 4 presents the experimental study we conducted to show the efficiency and the effectiveness of our solution. Section 5 concludes the paper and discusses future extensions.

2. TV stream macro-segmentation: a state-of-the-art

In order to be useful in real-world applications, TV stream macro-segmentation techniques have to fulfill the following main requirements:

- *Effectiveness*: The stream should be segmented accurately. Extracting an incomplete program is, in general, useless.
- *Efficiency*: Processing time should ideally be real-time or at least with a constant shifting time, which means segmentation results are returned with a constant delay after each event (the start or the end of each program).
- *Automaticity*: Segmentation should not require any manual annotation or user input. Annotating TV streams is very time consuming and any required manual processing would make the system unpractical, in particular when handling a large number of TV broadcasts.

As discussed in the Introduction and as shown in [5], metadata coming with the audio-visual stream are useless for macro-segmentation. Apart from traditional techniques for metadata integration and fusion [14,27], which could be used to enrich them and to increase their accuracy, only very few studies (among which [30]) have proposed novel ways to make use of these metadata.

Poli et al. [30] propose a statistical predictive approach that allows the correction of an EPG using a model learned from a ground-truth created over a one year broadcast. This approach is based on a simple observation: channels have to respect a certain regularity in their program schedule in order to preserve and increase their audience. The main drawback of this approach is the required ground-truth data for training. These data are required for each channel as program schedule differs from one channel to the other. It is therefore very difficult and highly time consuming to collect such training data. Poli's study was feasible because it has been conducted at INA, the French National Audiovisual Institute.[2] On the other hand, the model does not take into account program schedule of special events that may occur without any regularity from one year to the next (e.g. political events, sports competitions, ...).

Content-based approaches directly use the audio-visual content. In general, the first level of segmentation is only signal-based, that is, it only detects basic transitions in the signal. A more elaborate analysis is then performed upon this to properly delimit and extract programs.

As introduced in the previous section, macro-segmentation generally relies on the detection of inter-programs. These can be performed using three different approaches. (1) They can be detected using dedicated solutions using the intrinsic features of inter-programs (commercials are typically detected this way); (2) they can be detected using a content-based matching procedure with respect to a reference database (DB) of previously stored inter-programs; or (3) they can be automatically detected as repeated sequences.

These three approaches are described in the following sections.

2.1. Detection-based techniques

The majority of existing works on inter-program detection using their intrinsic features focus on commercials.

[2] INA is in charge of indexing and archiving French TV channels (www.ina.fr).

Lienhart et al. [25] propose a set of techniques to detect commercials. The proposed criteria and features are, among others:

- Monochrome frames: These frames are used by many TV channels to separate two consecutive advertisements. Lienhart et al. [25] propose to study the standard intensity deviation of each frame and to compute static decision thresholds.
- Scene breaks: The frequency and style of cuts (hard or fades) are analyzed.
- Action: Analysis of edge change ratio and motion vector length.

Hauptmann and Witbrock [21] also propose a similar approach to [25] within a general system for story segmentation in broadcast news. Albiol et al. [1] present a system that labels shots either as commercials or program shots. The system uses two criteria: logo presence and shot duration. These observations are modeled using HMMs. Duan et al. [12] have recently proposed a multimodal approach for commercial detection and categorization. Commercial shot detection is performed using visual and audio features; categorization and identification are done using extracted scene text using OCR methods.

All these approaches are limited to commercials and are therefore not sufficient to perform macro-segmentation.

2.2. Reference DB-based techniques

Storing inter-programs, including commercials, jingles and opening/final credits in a reference DB, reduces the macro-segmentation problem to content-based real-time sequence identification in an audio-visual stream. Inter-programs from the reference DB are identified and the stream is macro-segmented.

Methods following this principle use audio or video fingerprinting [7,29] to detect, in the stream, referenced sequences stored in the DB. Perceptual hashing can also be used [29,2,9].

Naturel et al. [28] propose a hybrid technique. In addition to a content-based identification of stored and labeled shots, a time-warping procedure has been proposed in order to *correct* metadata provided in the EPG.

These approaches have two main drawbacks, both related to the reference DB. Firstly, this DB has to be created manually for each TV channel. Secondly, the DB has to be periodically updated as new inter-programs are introduced.

2.3. Repetition-based techniques

Inter-programs are broadcasted several times a day. They can therefore be detected as repeated sequences[3] in the TV stream. Few recent papers have used this property to segment a TV stream. Inspired by video retrieval techniques, Gauch and Shivadas [15,16] propose a video shot-based solution. Shots are described and indexed using perceptual hashing. Repeated shots are then detected using a two-step procedure. The first step is based on collisions in the hash table. The second is based on the visual similarity between shots. Adjacent repeated shots are merged and classified (commercials or not). Covell et al. [10] propose an approach following the same principle as Gauch et al., but technically different. Repeated objects are detected using audio features and a hashing-based method. Detections are then verified using visual information. As for Herley [22], inter-programs are detected as *repeating objects* using a correlation study of audio features. At time t, the current *object* (an audio segment of predefined length) is compared to a past stored buffer of fixed size in order to detect any possible correlation.

In all these approaches, a post-processing step is required to select from the set of detected repeated sequence those that are actually inter-programs. Indeed, repeated sequences also include sequences that are broadcasted several times but that are not inter-programs. Examples of such sequences are news reports, flashback sequences in movies and series, ... This post-processing step is, however, quite easy to perform. In addition to be repeated sequences, inter-programs have other properties like their duration and frequency that can be used to separate them from the rest of detected repeated sequences. Inter-programs are also gathered temporally, whereas other repeated sequences that are not inter-programs are relatively isolated in the stream. They can hence be filtered out using a basic neighborhood analysis.

Despite their genericity, these solutions are technically limited. Solutions proposed in [15,16,10] focus on the detection of commercials. They suffer drawbacks of content-based matching techniques using hash-tables that are mainly related to the difficulty of choosing a suitable hash function with respect to the target similarity. They are also *brute force* in the sense that all descriptors of the whole audio-visual stream have to be inserted in the hash table and also saved, which could raise efficiency problems when dealing with a large amount of audio-visual data or with a continuous TV broadcast in a real-world system.

The method by Herley [22] requires some crucial parameters related to the size of descriptors, the search window and the length of the search buffer. These restrict the depth of the search and limit the detection to a *predefined fixed* size range of repeating objects.

3. The proposed solution for repeated sequence detection

The general working scheme of our solution is as follows. The stream is segmented and described on the fly. The accumulated stream and associated descriptors are then processed *periodically or on-demand* in order to detect repeated sequences using a clustering-based analysis procedure. This analysis also allows the trailers

[3] Repeated audio/video sequences are in this context near-identical audio/video sequences.

to be matched with the associated programs. Descriptors and data related to these labeled sequences are removed. The rest of the descriptors, however, are kept and added to those that are analyzed in the following period. That way, two occurrences of the same commercial for instance, can be detected as repeated sequences independently of the elapsed time between them. Obviously, in practice, the elapsed time is bounded and very "old" descriptors that did not generate any repeated sequence are removed.

The first time our solution is launched for the analysis of a TV stream, it needs to accumulate a sufficient amount of stream (about 24 h according to Exp. 1, Section 4.2). This boot time is required in order to accumulate a set of repeated sequences in the stream. After that, the repeated sequence detection can be launched at *any moment* using the accumulated stream.

Our solution does not make any hypothesis on the length or the frequency of repeated sequences. It is incremental which means that it is applied on a first "period" of the stream, delivers results, and continues with the following period while taking into account the previously processed parts of the stream. This way, it does not suffer the shortcomings of a buffer-based solution (like [22]).

The technique for repeated sequence detection that we propose is composed of three parts. The first one performs the description of the stream, i.e. the extraction of visual descriptors that allow matching similar shots of the video stream. The description scheme that we propose focuses on visual data. In Section 3.4, we explain how our solution can be extended to cope with audio data. The second part performs clustering over the set of extracted descriptors. The idea is to gather similar descriptors and thus similar images and the corresponding shots within the same group. The last part of the procedure is the exploitation of clusters. Their analysis provides a set of possible repeated sequences that are then checked and precisely delimited. The three steps are described in the three following subsections.

3.1. Video stream description

We propose to use a two-level description scheme. At the first level, an exhaustive description is performed. A basic visual descriptor (BVD) is extracted from each frame of the video stream. Its role is to match nearly identical frames and needs to be only invariant to small variations due to compression, for instance. The second level focuses on carefully chosen keyframes of the video stream. The descriptor associated with these keyframes (called key visual descriptor (KVD)) is, however, more sophisticated and has to be more robust.

The rationale behind this choice is related to the way repeated sequences are detected in the rest of the process. KVDs are first used during the clustering step to cluster similar shots and to create the set of repeated sequences. However, at this stage, the boundaries of detected repeated sequences cannot be determined. A KVD is associated with a frame of the repeated sequence, but does not provide any information on the sequence boundary. The BVD is thus used to precisely determine these boundaries by matching corresponding frames in all the occurrences of the repeated sequence.

Fig. 2 depicts the description scheme. BVDs are computed on all the frames of the stream. Given the simplified task of this description level, we have chosen a DCT-based description method to compute BVD as follows:

(1) The frame is divided into 4 blocks.
(2) Each block is sub-sampled to a matrix of 8×8.
(3) A DCT transform is then applied on each block and one DC coefficient and the 15 first AC coefficients (according to the zigzag order) are computed.
(4) Each coefficient is then binarized and a final 64-bit descriptor is created.

To measure the similarity between two BVDs, we have used the Hamming distance.

As for the second description level, a shot detection step is needed to segment the stream. Abundant literature exists on this topic [8,26,31,33,19] and mature solutions exist. The common principle of these techniques is to study motion variations and inter-frame pixel differences to detect shot transitions. These can be hard cuts or more gradual transitions (like dissolves). Hard cuts are very easy to detect, whereas gradual ones are more challenging. For our application, we do not need a perfect shot detection but only a segmentation that is quite the same for all the occurrences of a repeated sequence.

To achieve this, we used a method based on that proposed in [20]. It makes use of a study of the color similarity between each two consecutive frames. This similarity is measured using the χ^2 distance between the three-dimensional color histograms computed in the RGB color space (RGB color space, 512 bins issued from an 8 bins per channel uniform quantification). To detect transitions, we study the local variations of this measure over a sliding window centered around each frame. For hard cuts, there is a transition at the studied point if there is a peak in the color similarity there, i.e. if the similarity is greater than 10 times the standard deviation of the similarity in the rest of the window. The gradual transitions are, however, more difficult to detect. The method we used is based on the cumulated sum of the color similarity over a sliding window. In the case of a gradual transition, the graph of the cumulated sum is composed of three line segments. The first one has a small positive slope. The second segment has a much greater positive slope and the third has a small positive slope. This shape corresponds to the way a gradual transition happens. Studying the shape of the graph of the cumulated sum of the similarity can thus determine whether or not there is a gradual transition at the center of the sliding window.

Once the shot detection is performed, a set of keyframes is selected from each shot. The idea is to choose a keyframe to represent each homogeneous part of the shot and hence to cover the whole visual content of the shot. The topic of keyframe detection has been widely studied [11,13]. In our case, we propose to use motion information

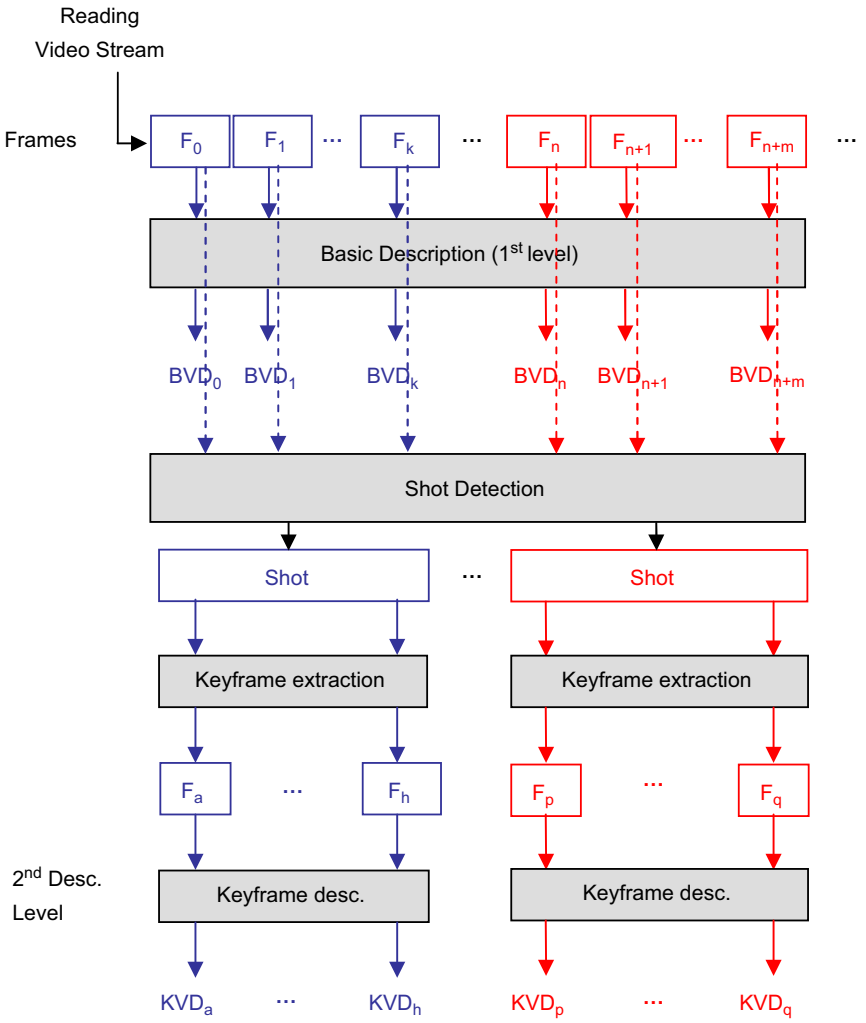

Fig. 2. Description of the video stream.

to select keyframes. The technique we have used relies on the inter-frame difference. This difference is computed between every two consecutive frames of the shot. The obtained one-dimensional signal is then smoothed using a Gaussian filter to remove small artifacts. The signal is then transformed to the cumulative difference. At each frame, instead of considering the difference with respect to the previous frame, we consider the sum of all the inter-frame distances between previous frames. Important changes in this function are then detected using a signal change detection method like the Page–Hinkley test [3]. They correspond to moments where an important motion arises in the scene. A keyframe is then selected in the middle between each two detected important changes.

From each selected keyframe, a visual descriptor is computed (a KVD). Again, we have used a DCT-based description scheme like for the BVD. This scheme is robust to usual variations like illumination. To make it robust to spatial variations like subtitles incursion or logo insertion/removal, we propose to divide the image into 3×2 blocks. Six independent descriptors are computed on the six blocks and then concatenated into a single descriptor, the KVD. To compute block descriptors, each block is first subsampled to an 8×8 matrix. A DCT is computed on this matrix and the first five coefficients (according to the zigzag order) are selected to build the descriptor. The KVD is hence a 30-dimensional vector. To measure the similarity between two KVDs we used the L_2 distance.

3.2. Clustering of keyframes descriptors

To gather similar keyframes that will be used to detect repeated sequences, we propose to use a clustering technique. Clustering is a non-supervised classification that allows grouping similar vectors and isolating outliers.[4] For a detailed presentation of clustering techniques, the interested reader can refer to [23,6]. In general, clustering techniques either ask for the number of clusters to create or require a set of parameters regarding the

[4] Here, outliers refer to uniformly distributed or isolated vectors.

shape of clusters or their population (i.e. number of vectors per cluster). Some of them also are able to isolate outliers. They are used in many fields like data mining, artificial intelligence, indexing, etc.

In our case, the clustering algorithm should fulfill a set of conditions related to the objectives of our application. Indeed, unlike most of applications using clustering, we are interested in finding a high number of very small clusters within a huge amount of uniformly distributed and isolated vectors. The number of vectors per cluster is determined by the number of times a sequence is repeated. If a sequence is repeated three times, and it is described by five KVDs (i.e. five keyframes have been selected from the sequence), then we should ideally discover five clusters with three vectors inside each one. The number of vectors per cluster thus ranges from two vectors to several hundreds. The number of clusters corresponds to the number of KVDs in the repeated sequences. As for the rate of outliers, it corresponds to the rate of KVDs that do not belong to repeated sequences, which could be very important. The clustering algorithm should also be able to process KVDs on the fly as they are computed from the video stream and should be able to work in an incremental mode.

Based on these criteria, we propose to use a *micro-clustering* technique that is able to generate a high number of clusters and isolate outliers. The method we have chosen is the one derived from BIRCH [34] and used in [4] to index a large set of image descriptors. In the following paragraphs we give a description of this method. The interested reader can refer to [4] for further details.

It is an iterative procedure that builds spherical clusters whose radii are controlled and must be below a threshold *Ft*. At the beginning, *Ft* is chosen to a very low value. During the first iteration, KVDs are inserted. A KVD is associated with a cluster of previously inserted KVDs only if the radius of the resulting bounding box of the cluster is less than Ft_1 (Ft_1 is the value of *Ft* in the 1st iteration). If no existing cluster can *absorb* the KVD, it is put in a new cluster as a singleton.[5] To characterize clusters and facilitate their use, a CF (clustering feature) vector is associated to each cluster. A CF vector is a triple composed of the number and the sum of the KVDs belonging to the cluster, and the radius of its bounding box. For example, when merging two clusters, the CF vector of the resulting cluster can be easily computed using only the CF vectors of the two clusters.[6] That way, KVDs are read only once during the first iteration. In the following iterations, only CF vectors are manipulated.

After each iteration, the outliers are isolated. In our application, outliers are singletons. If the number of clusters falls below a fixed number (the maximum number of clusters), the clustering process has finished. Otherwise, the *Ft* is increased and a new iteration is performed. Clusters (including singletons) are inserted and two clusters are merged if the resulting cluster has a radius less than the new value of *Ft*. Hence, the number of clusters decreases each iteration.

This method can be used in an incremental mode as new KVDs can be included at any moment in the clustering process as singletons.

As explained previously, the maximum number of clusters is related to the number of repeated sequences and the number of associated keyframes. It can therefore be experimentally determined by a statistical study over a sample of TV broadcast. We will also show in Section 4.3 that the optimal number of clusters is not critical.

3.3. Clusters analysis and repeated sequence extraction

Before starting to analyze returned clusters, it is important to remove those that have no chance of generating a repeated sequence. It is the case of clusters that contain KVDs extracted from the same shot or from neighboring shots. For example, this happens during a debate when shots are alternatively centered around different antagonists. To remove these clusters, we define a measure on the temporal diversity (TD) of KVDs in the cluster. This measure is computed as follows:

$$\mathrm{TD}(C_i) = \frac{\sum_{k=2}^{r} f_i(k)}{r-1}, \tag{1}$$

where $f_i(k)$ is the temporal distance between the kth and the $(k+1)$th KVDs of the cluster C_i, and r is the number of KVDs of C_i.

If the TD of a cluster is below a threshold, the cluster is filtered out and not considered in the following steps of repeated sequence detection. In practice, the TD could be set to few seconds. In our implementation, it has been set to 4 s.

The rest of the process is composed of two main parts: the computation of inter-cluster similarity and the detection of repeated sequences.

3.3.1. Inter-cluster similarity

To exploit selected clusters, we define the inter-cluster similarity. This similarity measures the chance for two clusters to generate a repeated sequence. To explain how it is computed, let us consider two clusters C_i and C_j, containing n KVDs each. The set $\{I_1,\ldots,I_n\}$ (resp. $\{J_1,\ldots,J_n\}$) is the set of keyframes whose KVDs belong to C_i (resp. C_j). The first condition for two clusters to generate a repeated sequence concerns the temporal order of keyframes $\{I_k\}_k$ and $\{J_k\}_k$. They have to be alternating as depicted in Fig. 3. In this case, we say that C_i and C_j are interlaced.

Interlacing, however, gives information only on the order of keyframes. The temporal distance between each couple of keyframes from the two clusters is the second condition. Indeed, temporal distances between each couple (I_k,J_k) must be nearly constant if the $\{I_k\}_k$ and the $\{J_k\}_k$ are keyframes of a repeated sequence. The chance for C_i and C_j to generate a repeated sequence is thus related to the constancy of the temporal distance (Tdist) between their keyframes. The inter-cluster similarity must be one if all the temporal distances are identical and decreases when they are not. Therefore, we propose the following

[5] A singleton cluster is a cluster that contains only one KVD and of which the radius is equal to 0.

[6] In this case, the radius of the resulting cluster is an over estimation of the actual radius, but this has no effect on the final clustering result.

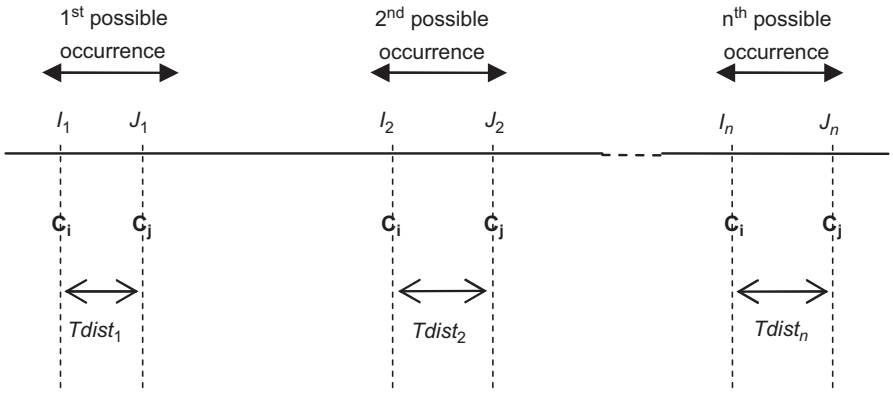

Fig. 3. Two interlaced clusters with n KVDs each.

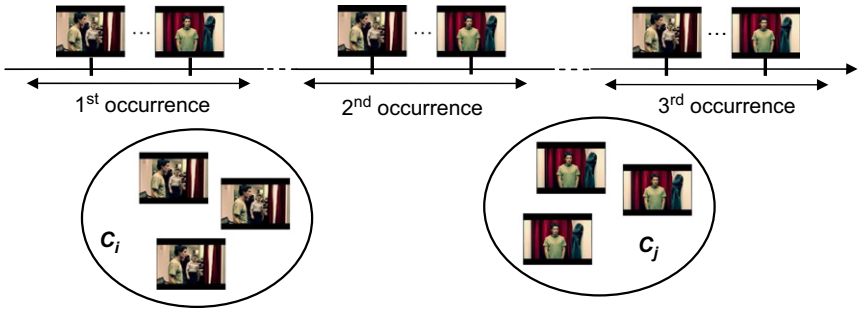

Fig. 4. Generation of a three times repeated sequence from two clusters.

formula to compute the inter-cluster similarity between clusters C_i and C_j:

$$\mathrm{Sim}_{ij} = \begin{cases} 0 & \text{if } C_i \text{ and } C_j \text{ are not interlaced,} \\ e^{-\sigma} & \text{otherwise,} \end{cases} \quad (2)$$

where σ is the standard deviation of $\mathrm{Tdist}_1, \ldots, \mathrm{Tdist}_n$ (cf. Fig. 3).

3.3.2. Detection of repeated sequences

The inter-cluster similarity is computed between each couple of clusters. The results are stored in a matrix B, where element b_{ij} is the similarity between clusters C_i and C_j (that is, $b_{ij} = \mathrm{Sim}_{ij}$). To process clusters, we define a basic relation between clusters as follows. If $b_{ij} > \delta_{\mathrm{sim}}$ then clusters C_i and C_j are related. This relation is aimed at gathering clusters that have a high inter-cluster similarity and, hence, that have a good likelihood to generate a repeated sequence. The influence of the parameter δ_{sim} is empirically studied in Section 4.3.

The rest of the process is summarized in the following steps:

(1) Select clusters that have at least one relation with another cluster.
(2) Select the set of most populated clusters, i.e. clusters with the highest number of KVDs.
(3) Perform transitive closure within the selected set. Again, the objective is to partition the set into subsets where each cluster is related to one or more other clusters from the same subset.
(4) Generate a repeated sequence from each subset as depicted in Fig. 4 and remove the used clusters from B.
(5) Continue with the process from step (1) until no cluster is selected.

In step (2), we have chosen to start the process by the most populated clusters in order to first retrieve the most frequent repeated sequences. Indeed, as depicted in Fig. 3, for a chosen subset of clusters, the number of occurrences of a repeated sequence is equal to the number of KVDs per cluster (we recall that all the clusters within the subset are interlaced and each have the same number of KVDs).

In step (4), the generation of repeated sequences consists of defining the boundaries of each occurrence of the repeated sequence. First, the boundaries are defined by the left-most and right-most keyframes. These boundaries are then extended using BVDs computed for all the frames of the stream. The occurrences are extended to the left (resp. the right) if BVDs of all the left (resp. right) neighboring frames for all the occurrences are similar. To make the extension procedure more robust, we propose to simultaneously compare a set of neighboring frames, that is, the extension procedure compares the m left (resp. right) frames of all the occurrences every time. If the average dissimilarity is less than a threshold, the occurrences are extended to the left (resp. right) by m frames.

3.3.3. The special case of trailers

As claimed in the Introduction, our solution can handle trailers. It is not only able to detect them as repeated sequences, but also to match some of them with the associated long program. This last feature is possible if (1) the trailer is composed of a subset of the shots of the corresponding program, and (2) these shots appear in the same order as their appearance in the program. These conditions are, however, not a hard limitation as trailers of movies, which are the most relevant for a TVoD service, generally fulfill the criteria. In the rest of this section, we focus on this specific kind of trailer.

During the clustering step, KVDs from the program will also be gathered with KVDs extracted from occurrences of the trailer. The only difference between KVDs of the trailers and KVDs of the program is the temporal distance: it is coherent and almost constant for the trailer occurrences but not for the program (cf. Fig. 5). Therefore, to be able to detect the trailers as repeated sequences and associate them with the corresponding long programs, the temporal distance of the last occurrence (i.e. the last KVDs) has to be ignored when computing the inter-cluster similarity. In other terms, in Eq. (2), the standard deviation σ has to be computed only on $\mathrm{Tdist}_1, \ldots, \mathrm{Tdist}_{n-1}$. Obviously, the extension procedure also has to be modified in order to ignore the last occurrence if the temporal distance computed on this last occurrence is not coherent with the rest of temporal distances.

3.4. Extension to audio

In the previous sections, we have presented our method for repeated sequence detection using visual information. This method allows the detection of inter-programs that are repeated with near-identical visual information. For the macro-segmentation task, it is important to also detect inter-programs that share the same audio content but a different visual content. In this section, we present how our solution can be extended to cope with audio data.

The adaptation of the method for detecting repeated audio sequences is quite straightforward. As there is no analogy in audio stream description to the way visual stream is described using keyframes, we propose to perform a regular and uniform segmentation of the audio stream. Each basic audio segment is described using the local audio signal features. These basic sub-descriptors (SD), however, are not sufficient to individually describe an audio sequence. Therefore, we propose to create a second description level into which a sequence of sub-descriptors is gathered in order to compose an audio-description (AD). The proposed scheme description is depicted in Fig. 6. Descriptors ADs are composed this way in order to increase robustness and take into account the local temporal context.

By analogy with the visual stream description scheme we have proposed in Section 3.1, the SD corresponds to the BVD and the AD corresponds to the KVD. Hence, to apply our method for detecting repeated audio sequences, the ADs have to be clustered and the SDs are used to precisely delimit repeated sequences in the final step of the process.

To compute SD, we propose to use the commonly used audio description scheme. The audio signal is windowed into frames of 64 ms length, with a 32 ms overlap. Each frame is then transformed to the frequency domain using the discrete Fourier transform and divided into bands. Five

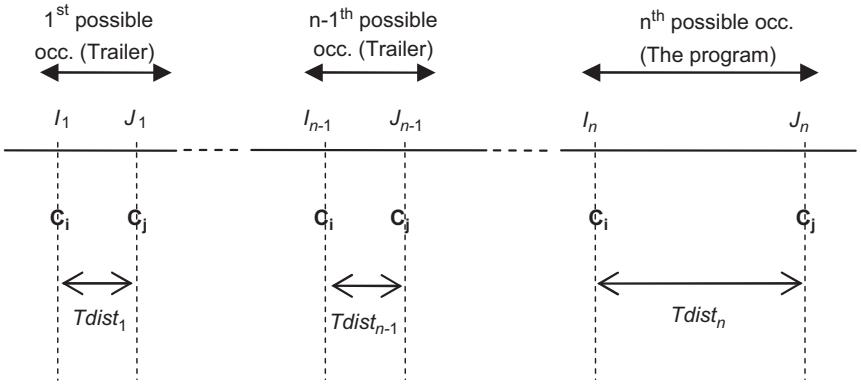

Fig. 5. Special case of trailer detection.

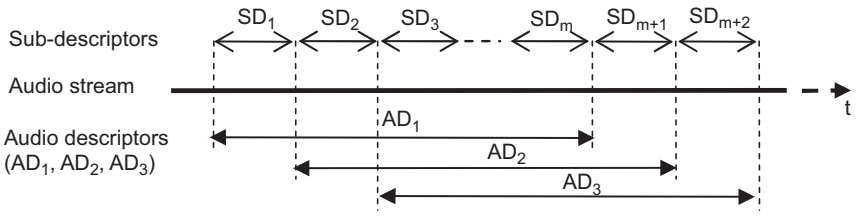

Fig. 6. Description scheme of audio stream.

adjacent bands are used, ranging from the most relevant frequency range 300–3000 Hz, according to [18]. For each band, the energy is computed, leading to an SD of dimension five computed each 32 ms. In our implementation, the AD is composed of 16 SDs.

4. Experiments

To evaluate and validate our method for repeated sequence detection, we performed a set of experiments using real TV broadcasts. The first experiment studies the repetition of inter-programs. The objective is to validate the main idea behind our approach, i.e. the detection of inter-programs as repeated sequences. The next two experiments evaluate our solution using visual description on two TV broadcast records. The third experiment presents results that validate the extension of our approach to cope with audio data.

All the algorithms have been developed in C++. Experiments have been performed on a PC under Windows XP. Its CPU is a 2 GHz Intel Xeon, with 2 GB of main memory.

4.1. Experimental data sets

The two TV broadcasts we have used to evaluate our approach using the visual description scheme are: (1) TV12h: a 12 h sequence and (2) TV7D: a 7 day sequence.

Table 1 gives some information on these two broadcasts and the extracted visual descriptors.

TV12h has been manually segmented and annotated. A set of 59 repeated sequences has been discovered. The most frequent sequence is a trailer that has been broadcasted six times. The set of repeated sequences is composed of 48 commercials, 10 trailers and one opening credit. The total number of occurrences of these sequences is 150. TV12h provides a ground-truth that is used to evaluate the precision and recall of our solution.

As for TV7D, it has not been completely annotated. Repeated sequence annotation is a very difficult and time consuming task. Therefore, we have first listed commercials, trailers and jingles (without determining exactly the start and the end times). Then, we have focused on trailers and have manually and precisely annotated them (using their positions in the stream and their durations). The list of commercials, trailers and jingles has been used to study the repetition of inter-programs in the stream in order to show the ability to detect inter-programs using only the repetition property. The data set TV7D has also been used to evaluate the precision on a set of randomly selected subset of detected repeated sequences and the recall on the set of manually annotated trailers. The objective of TV7D is also to show the ability of our solution to handle large TV broadcasts.

To validate the extension of our approach to cope with audio data, we have used a recorded TV broadcast of 1 h and 17 min. In the remainder of this section we call it TV1h17.

Table 1
Information on TV12h and TV7D

	Duration	Num. of frames = Num. of BVDs	Num. of keyframes = Num. of KVDs
TV12h	11 h 58 min 59 s	1,078,485	28,397
TV7D	168 h (7 days)	15,120,000	285,182

4.2. Exp. 1: inter-programs repetition study

As explained earlier, the aim of this first experiment is to validate the main idea behind our solution and hence show that inter-programs generally repeat in the stream. The objective of this experiment is also to study the proportion of inter-programs that repeat with respect to the duration of the analyzed TV stream. In other words, this study aims to show the minimal duration of TV stream that has to be analyzed in order to have most or all of the inter-programs that repeat at least twice.

Fig. 7 shows the proportion of inter-programs that repeat with respect to the accumulated TV stream. It shows in particular that after two days, all jingles repeat and can therefore be detected as repeated sequences by our solution. Moreover, after four days, all of the inter-programs repeat apart from 3% of the commercials and 10% of the trailers. These can, however, be detected using the neighborhood in a post-processing step using simple rules. For instance, a very short sequence that is located between two commercials is a commercial.

This experiment clearly validates the idea behind the paper. It shows also that to be efficient, our solution needs to start the stream analysis after having accumulated at least 24 h. After that, results can be delivered periodically or on demand.

4.3. Exp. 2: influence of clustering and δ_{sim}—reliability

The second set of experiments we performed focuses on TV12h. The objective is to test the influence of the number of clusters and the parameter δ_{sim} on the performance of the solution, and its reliability.

We recall that the threshold δ_{sim} is used to decide whether the inter-cluster similarity between two clusters is sufficient to relate them and, hence, to use them together toward generating a repeated sequence. Inter-cluster similarity is computed using Eq. (2) and varies between 0 and 1.

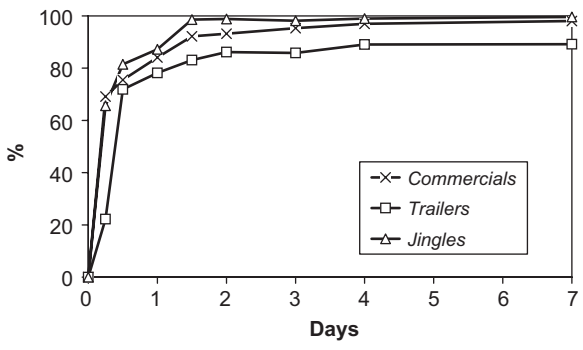

Fig. 7. Proportion of inter-programs that repeat in the TV7D stream.

As the number of clusters and δ_{sim} are the only two parameters of our method, we have studied the performance simultaneously varying their values.

To generate different partitions with different numbers of clusters, we have varied several times the stop criterion of the algorithm, i.e. the maximum number of clusters. Table 2 summarizes the results obtained on the set of KVDs extracted from TV12h. We can notice that when the number of clusters decreases, the number of outliers also decreases, whereas the average number of KVDs per cluster grows. This is due to the fact that when increasing the *Ft* parameter to reduce the number of clusters, the clusters enlarge and absorb isolated vectors.

These results confirm the intuition we had about the relationship between clusters, repeated sequences and the distribution of KVDs. Moreover, they show the effectiveness of the clustering technique we have used and its reliability in having the desired behavior with respect to the distribution of KVDs and what we would like to extract from them.

As for δ_{sim}, we have varied it between 0.75 and 0.95 with a step of 0.05. Then, we have run our method for repeated sequence detection on the different generated clusterings and varying δ_{sim}. For each experiment, we computed precision and recall of the obtained result with respect to the ground-truth. We consider that an occurrence has been correctly detected if its overlapping rate with the corresponding occurrence in the ground-truth is greater than 45%. The obtained results are summarized in Table 3.

We can notice that recall is independent of δ_{sim}. However, the precision is better when δ_{sim} is at its maximum value. Indeed, when increasing δ_{sim}, the number of related clusters decreases during the repeated sequence detection process, and so does the number of detected sequences.

If we analyze the detected sequences, we notice that when $\delta_{sim} = 0.95$ and when considering the highest number of clusters, only one sequence that is repeated three times has not been detected. Otherwise, apart from one false detection, all sequences that have been detected are repeated sequences, but some of them do not belong to the ground-truth, i.e. some of them are not inter-programs. This is why the precision is relatively low. This side effect concerns all of repetition-based detection techniques. It has been analyzed in Section 2.3 and is not really a shortcoming. Repeated sequences that are not inter-programs can easily be filtered out using a post-processing step (cf. Section 2.3).

Table 2
Clustering results on TV12h

Num. of clusters	Num. of outliers	Average num. of KVDs per cluster
7319	9359	2.60
6454	8154	3.13
5602	6879	3.84
4931	5487	4.64
4058	2724	6.32
3265	942	8.40
2210	126	12.79
1348	7	21.06

Table 3
Exp. 2

Num. of clusters	δ_{sim}									
	0.75		0.80		0.85		0.90		0.95	
	P	R	P	R	P	R	P	R	P	R
1348	1	0.027	1	0.027	1	0.027	1	0.027	1	0.027
2210	0.840	0.284	0.905	0.257	0.905	0.257	0.905	0.257	0.905	0.257
3265	0.958	0.622	0.958	0.622	0.958	0.622	0.956	0.581	0.956	0.581
4058	0.872	0.689	0.872	0.689	0.895	0.689	0.927	0.689	0.927	0.689
4931	0.854	0.831	0.854	0.831	0.872	0.831	0.872	0.831	0.872	0.831
5602	0.757	0.905	0.757	0.905	0.770	0.905	0.827	0.905	0.854	0.905
6454	0.727	0.919	0.727	0.919	0.727	0.919	0.760	0.919	0.805	0.919
7419	0.751	0.980	0.751	0.980	0.751	0.980	0.784	0.980	0.806	0.980

Recall (R) and precision (P) of repeated sequence detection on TV12h varying the number of clusters and the value of δ_{sim}.

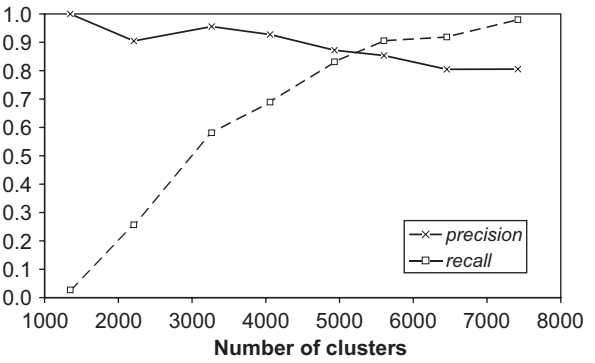

Fig. 8. TV12H, $\delta_{sim} = 0.95$—performance of repeated sequence detection with respect to the number of clusters.

Regarding trailers, only one trailer is broadcasted along with the associated program in TV12h. All the occurrences of this trailer have been detected and correctly associated with the program.

Overall, the performance of the method is very good. However, we can notice that the best performance is obtained with the highest number of clusters (precision = 0.806 and recall = 0.980). This was also expected. When the number of clusters is high, their population is low and they are less influenced by outliers. In general, as depicted in Fig. 8, when increasing the number of clusters, precision decreases and recall increases.

We have also computed precision and recall on a per shot basis for $\delta_{sim} = 0.95$ (the value that achieves the best results). Obtained results are summarized in Table 4. We can notice that these results are better than those computed considering the whole inter-programs. This is mainly due to the fact that missed inter-programs are generally very short and contain only one or two shots.

To further evaluate the performance of our solution, we have also analyzed the accuracy of the segmentation of detected repeated sequences. For that, we have measured the absolute value of the difference between the start

Table 4
Exp. 2

Num. of clusters	$\delta_{sim} = 0.95$	
	Precision	Recall
1348	0.923	0.029
2210	0.916	0.316
3265	0.922	0.725
4058	0.924	0.807
4931	0.902	0.922
5602	0.894	0.969
6454	0.884	0.975
7419	0.874	0.991

Recall and precision on a per shot basis of repeated sequence detection on TV12h varying the number of clusters.

Table 5
Exp. 2

	μ	σ
Start	18 frames (<1 s)	11 frames (<1 s)
End	31 frames (≈ 1.3 s)	17 frames (<1 s)

Accuracy of repeated sequence detection.

(resp. the end) time of each detected repeated sequence and the actual start (resp. end) time given in the ground-truth. Table 5 gives the mean (μ) and the standard deviation (σ) of these differences on the whole set of repeated sequences. In this study, we have considered only the clustering with the highest number of clusters (7319) and $\delta_{sim} = 0.95$ (as they provide the best results).

Table 5 shows that our solution is very effective in precisely segmenting repeated sequences: a mean error of 24.5 frames, i.e. about 1 s. This mean error accounts for the mean duration of monochrome frames between commercials which are generally added to commercials in the detected repeated sequences, but that are not taken into account in the ground-truth.

As for efficiency, the clustering process of the 28,397 KVDs extracted from TV12h lasts only 63.45 s. The detection of repeated sequences has been performed in 4.22 s. Thus, our method is able to process a 12 h broadcast in less than 2 min.

4.4. Exp. 3: influence of the TV stream size

To evaluate the ability of the method to handle large scale TV streams, we have run our method on TV7D. Based on the conclusion of Exp. 1, we have used the clustering with the highest number of clusters and we have set δ_{sim} to 0.95.

Clustering has returned 58,424 clusters and has isolated 108,316 outliers. Each cluster contains on average 3.03 KVDs. The obtained results are the following:

- 975 repeated sequences have been detected.
- The most frequent detected sequence is a commercial of 4.24 s that is repeated 67 times.
- The average number of occurrences per repeated sequence is 4.63.

To evaluate the precision of the results, we have selected the 50 most repeated sequences and we have randomly chosen 50 other sequences. Then, we manually evaluate the results. The observed precision is equal to one.

We have also evaluated the recall considering only the set of 42 trailers that have been manually annotated. These 42 trailers are repeated 376 times. Our solution has been able to detect 372 occurrences, achieving a recall rate of 0.99.

The computation time for clustering the 285,182 KVDs extracted from TV7D was approximately 1 h and 15 min. Repeated sequences have been detected in 1 min and 44 s. Thus, the total time to process a 7 day broadcast is less than an hour and a half. This shows that our method can effectively and efficiently handle very large TV broadcasts.

4.5. Exp. 4: extension to audio

To study how our approach performs in detecting audio repeated sequences (using audio features), we have used TV1h17 and we have executed the technique described in Section 3.4. In this experiment, an audio descriptor (AD) has been created as a sequence of 16 sub-descriptors (SD). Knowing that each SD is composed of five values (energies of band frequencies of the signal), the dimension of an AD is 80 ($= 16 \times 5$). The number of ADs computed on TV1h17 is 144,375.

The set of ADs has been clustered. A set of 4636 clusters has been created and 122,375 outliers have been isolated. The analysis of these clusters generates seven repeated sequences for a total number of 14 occurrences.

The evaluation of the accuracy of the obtained results, however, is very difficult. Annotating audio documents is very difficult and even completely impossible for very large documents. If it is easy to check that two sequences are visually similar or near-identical by visualizing them simultaneously, it is impossible to simultaneously play several audio sequences and it is very hard to compare their similarity. Nevertheless, in order to evaluate the performance of our approaches on detecting audio repeated sequences, we have first checked if the technique is able to detect repeated sequences that have been manually annotated based on visual data. Here, we focus on repeated sequences that have the same visual and audio data. The obtained results show that our technique using audio data is able to detect all of these repeated sequences, that are one opening credit that is repeated twice, two commercials that are repeated each twice and a trailer that is repeated twice. We have also noticed that two additional repeated sequences have been detected: (1) two occurrences of a jingle that share the same audio but with different visual information, and (2) two occurrences of very short jingle that lasts 1.8 s. These two repeated sequences have not been manually annotated for the following reasons. The first one has different

visual information and the second one is too short and has been missed during annotation.

These preliminary results validate the extension of our approach to detecting repeated audio sequences. It shows also its complementarity with repeated visual sequence detection.

However, we note that the audio description scheme computes a larger set of descriptors than the visual description scheme. This may raise efficiency problems when dealing with real large audio streams. Additional work on the audio description scheme is required.

5. Conclusions and future works

In this paper, we have presented a method for repeated sequence detection within TV broadcasts. It detects inter-programs as repeated sequences using a clustering-based approach. Our method relies on visual descriptors but can also be applied on audio data. It is also able to detect trailers and to match them with the associated program. This is highly relevant for TV broadcast macro-segmentation.

Future works will focus upon mainly two points. The first point concerns the extension of our approach to detect repeated audio sequences. A more extensive study has to be performed in order to optimize the description scheme and to validate it. Experiments presented in this paper are only preliminary results and their objective was only to show how our approach can be applied on audio data. The cooperation between audio- and video-based approaches also has to be thoroughly investigated.

The second point is related to the TV stream macro-segmentation process. We now have an efficient and effective technique for repeated sequence detection that has to be integrated and evaluated within a macro-segmentation system.

References

[1] A. Albiol, M.J. Fulla, A. Albiol, L. Torres, Detection of TV commercials, in: Proceedings of the IEEE International Conference on Acoustics Speech, and Signal Processing, vol. 3, Montreal, Quebec, Canada, 2004.

[2] J. Barr, B. Bradley, B. Hannigan, Using digital watermarks with image signatures to mitigate the threat of the copy attack, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, vol. 3, Hong Kong, 2003.

[3] M. Basseville, Detecting changes in signals and systems—a survey, Automatica 24 (3) (1988) 309–326.

[4] S.-A. Berrani, L. Amsaleg, P. Gros, Approximate searches: *k*-neighbors+ precision, in: Proceedings of the 12th ACM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, 2003.

[5] S.-A. Berrani, P. Lechat, G. Manson, TV broadcast macro-segmentation: metadata-based vs. content-based approaches, in: Proceedings of the ACM International Conference on Image and Video Retrieval, Amsterdam, The Netherlands, 2007.

[6] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer, Berlin, 2006.

[7] P. Cano, E. Batlle, T. Kalker, J. Haitsma, A review of algorithms for audio fingerprinting, in: Proceedings of the IEEE Workshop on Multimedia Signal Processing, St. Thomas, Virgin Islands, USA, 2002.

[8] L.F. Cheong, Y. Wang, H.L. Wang, Establishment shot detection using qualitative motion, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, vol. 2, Madison, WI, USA, 2003.

[9] B. Coskun, B. Sankur, Robust video hash extraction, in: Proceedings of the IEEE 12th Signal Processing and Communications Applications Conference, Vienna, Austria, 2004.

[10] M. Covell, S. Baluja, M. Fink, Advertisement detection and replacement using acoustic and visual repetition, in: Proceedings of the 8th IEEE International Workshop on Multimedia Signal Processing, Victoria, BC, Canada, 2006.

[11] M.S. Drew, J. Au, Video keyframe production by efficient clustering of compressed chromaticity signatures, in: Proceedings of the 8th ACM International Conference on Multimedia, Marina del Rey, CA, USA, 2000.

[12] L.-Y. Duan, J. Wang, Y. Zheng, J.S. Jin, H. Lu, C. Xu, Segmentation categorization and identification of commercial from TV streams using multimodal analysis, in: Proceedings of the ACM Multimedia, Santa Barbara, CA, USA, 2006.

[13] B. Fauvet, P. Bouthemy, P. Gros, F. Spindler, A geometrical key-frame selection method exploiting dominant motion estimation in video, in: Proceedings of the International Conference on Video and Image Retrieval, Dublin, Ireland, 2004.

[14] I. Foster, R.L. Grossman, Data integration in a bandwidth-rich world, Commun. ACM 46 (11) (2003) 50–57.

[15] J. Gauch, A. Shivadas, Identification of new commercials using repeated video sequence detection, in: Proceedings of the IEEE International Conference on Image Processing, vol. 3, Genova, Italy, 2005.

[16] J.M. Gauch, A. Shivadas, Finding and identifying unknown commercials using repeated video sequence detection, J. Comput. Vision Image Understanding 103 (1) (2006) 80–88.

[17] J.-L. Gauvain, L. Lamel, Large-vocabulary continuous speech recognition: advances and applications, Proc. IEEE 88 (8) (2000) 1181–1200.

[18] J. Haitsma, T. Kalker, J. Oostveen, Robust audio hashing for content identification, in: Proceedings of the International Workshop on Content-based Multimedia Indexing, Brescia, Italy, 2001.

[19] A. Hampapur, T. Weymouth, R. Jain, Digital video segmentation, in: Proceedings of the 2nd ACM International Conference on Multimedia, San Francisco, California, USA, 1994.

[20] A. Hanjalic, Shot-boundary detection: unraveled and resolved? IEEE Trans. Circuits Syst. Video Technol. 12 (2) (2002) 90–105.

[21] A. Hauptmann, M. Witbrock, Story segmentation and detection of commercials in broadcast news, in: Proceedings of the Advances in Digital Libraries Conference, Santa Barbara, CA, USA, 1998.

[22] C. Herley, Argos: automatically extracting repeating objects from multimedia streams, IEEE Trans. Multimedia 8 (1) (2006) 115–129.

[23] A.K. Jain, M.N. Murty, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 265–323.

[24] A.K. Jain, B. Yu, Pattern recognition, Pattern Recognition 31 (12) (1998) 2055–2076.

[25] R. Lienhart, C. Kuhmunch, W. Effelsberg, On the detection and recognition of television commercials, in: Proceedings od the IEEE International Conference on Multimedia Computing and Systems, Ottawa, Ontario, Canada, 1997.

[26] T. Liu, F. Qi, Shot detection based on dominant region tracking, in: Proceedings of the 6th International Conference on Signal Processing, vol. 1, Beijing, China, 2002.

[27] A. Motro, P. Anokhin, Fusionplex: resolution of data inconsistencies in the integration of heterogeneous information sources, Inf. Fusion 7 (2) (2006) 176–196.

[28] X. Naturel, G. Gravier, P. Gros, Fast structuring of large television streams using program guides, in: Proceedings of the 4th International Workshop on Adaptive Multimedia Retrieval, Geneva, Switzerland, 2006.

[29] J. Oostveen, T. Kalker, J. Haitsma, Feature extraction and a database strategy for video fingerprinting, in: Proceedings of the 5th International Conference on Recent Advances in Visual Information Systems, Hsin Chu, Taiwan, 2002.

[30] J.-P. Poli, Predicting program guides for video structuring, in: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence, Hong Kong, China, 2005.

[31] E. Stringa, C. Regazzoni, Real-time video-shot detection for scene surveillance applications, IEEE Trans. Image Process. 9 (1) (2000) 69–79.

[32] V. Wu, R. Manmatha, E.M. Riseman, Textfinder: an automatic system to detect and recognize text in images, IEEE Trans. Pattern Anal. Machine Intell. 21 (11) (1999) 1224–1229.

[33] H.J. Zhang, A. Kankanhalli, S.W. Smoliar, Automatic partitioning of full-motion video, Multimedia Syst. 1 (1) (1993) 10–28.

[34] T. Zhang, R. Ramakrishnan, M. Livny, Birch: an efficient data clustering method for very large databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Montreal, Canada, 1996.