INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION INTERNE
N° 1790

DETECTING REPEATS FOR VIDEO STRUCTURING

XAVIER NATUREL AND PATRICK GROS

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Detecting Repeats for Video Structuring

Xavier Naturel * and Patrick Gros **

Systèmes symboliques
Projet Texmex

Publication interne n1790 — March 2006 — 34 pages

Abstract: Television daily produces massive amounts of videos. Digital video is unfortunately an unstructured document in which it is very difficult to find any information. Television streams have however a strong and stable but hidden structure that we want to discover by detecting repeating objects in the video stream. This report shows that television streams are actually highly redundant and that detecting repeats clearly outlines the underlying structure of the video. A method for detecting these repetitions is presented here with an emphasis on the efficiency of the search in a large video corpus. Very good results are obtained both in terms of effectivness (98% in recall and precision) as well as efficiency since one day of video is queried against a three weeks dataset in only one second.

Key-words: Video structuring, video indexing, television, video signature, repetitions

(Résumé : tsvp)

* xavier.naturel@irisa.fr
** patrick.gros@irisa.fr

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Détection de répétitions pour la structuration vidéo

Résumé : La télévision produit en permanence des flux vidéo d'une ampleur considérable, sans que l'on sache pourtant ce qui a été exactement diffusé. Nous nous intéressons ici à la détection de répétitions dans des vidéos de grande taille. On montre que les flux de la télévision sont en effet extrêmement répétitifs et que la détection des *auto-similarités* permet de découvrir la structure sous-jacente du flux. Ce rapport présente en détail une méthode de détection d'objets répétitifs dans une optique de recherche rapide dans d'importants volumes vidéos. De très bons résultats sont obtenus tant en qualité (98% de rappel et précision) qu'en rapidité, puisqu'il suffit d'une seconde pour requêter une journée entière de vidéo contre une base de trois semaines.

Mots clés : Structuration vidéo, indexation vidéo, télévision, signature vidéo, répétitions

Contents

1 Context and objectives

1.1 Introduction

Television experienced drastic changes recently with an ever increasing number of channels and new ways of delivering its content to the user, e.g. TV over ADSL, mobiles, internet, terrestrial digital TV. This growth has unfortunately not been linked with a better understanding of how to use this video material. Querying and retrieving information from a large television (or video) corpus is still a challenge, for both professional archivers and simple TV users alike.

One simple but surprisingly unaddressed issue is to retrieve an entire program with its exact boundaries, i.e. from its first to its last frame. Due to time-shifting in the broadcast, the precise time of broadcast is elusive, resulting in annoying commercials or channel events being found instead of the desirable program. It would be convenient to automatically identify the program segments from a recorded television stream, thus building an updated and enhanced program guide. This may be of interest to many applications, end-user VCR, regulation authorities or channels themselves, which may want to store or monitor precisely what has been broadcasted. All these tasks are usually done manually. This is a very tedious work especially if precise timestamps are needed.

The main idea behind this work is to look for repetitions in the television stream. A lot of segments are indeed repeated in a TV stream, and the goal of this work is to see whether these repetitions can help to discover the hidden structure of the stream. Considering that the repeated segments are mostly commercials, trailers, channel lead-in, lead-out, we believe the detection of these repetitions make sense from a structuring point of view, as long as the process is run over a large dataset.

The article is structured as follows: section 1 is an introductory section explaining the context and the goal of the method. Section 2 is dedicated to features extraction and thus explain the choice and the construction of the features used for the retrieval. The organisation of the database is presented in section 3 and the retrieval process in section 4. Results are eventually presented in section 5.

1.2 Related work

A lot of research has been done to identify similar video clips in a video dataset. Most of this research has been conducted with two main purposes in mind: video copy detection and video similarity search for content-based retrieval. To our knowledge, no work has been conducted on detecting repetitions for video structuring, although some authors mentionned this possibility [12, 31] or investigated very close topics [25]. Most of the techniques used in these previous works are thus slightly unadapted to our requirements, though very close. The first application aims at retrieving videos that originated from the same source, possibly with strong tampering between them. A solution is to build a robust fingerprint from the content, able to resist to some predefined modifications [23]. Due to the necessity to capture the essential geometric properties of the image and to resist to tampering, fingerprints

are usually based on interest points [11] or computed from the frequency domain of some transform, popular examples being the wavelet or DCT domain [15, 6].
The second application is interested in retrieving clips from a video dataset that are similar to a certain query video clip. Since retrieval is based on similarity, the features used are far less discriminative, resulting in a wide use of global features such as color, texture, shape, motion or audio descriptors [28, 31, 3].
The common point of these methods is that these descriptors are vectors in a high dimensional space in which the search is performed. Numerous smart indexing schemes and search methods have been proposed to efficiently search within these high-dimensional spaces [31, 5, 18]. These methods are complex and dealing with large datasets in this context is still a difficult issue. One can however mentionned Joly[11], who manages to handle very large video databases. Joly uses a 20-dimension descriptor computed from an interest point detector and propose a so-called statistical query to perform approximate search in this high dimensional space. The effectiveness of his method is shown by querying effectively a dataset of 50,000 hours of video.
The methods used in copy detection and similarity search are however not suited to match identical sequences because they allow important differences between them. Detection and recognition of commercials is much closer to our topic. Lienhart et al. [19] use a fingerprint based on color coherence vector and compare the fingerprints using a standard approximate substring matching algorithm. Sanchez et al. [26] build a subspace by computing principal component analysis on key-frames of the spots to recognize. Matching is then done by computing the minimum Euclidean distance of the query commercials key-frames representation in this subspace. An interesting attempt to see commercials as repetitions is by Duygulu et al. [7]. Unfortunately, the method does not seem to be very effective and has to be combined with a classification based on low-level features to isolate commercials from broadcast news. The drawback of these methods is that they are unfortunatly restricted to commercials and do not consider seriously large dataset and efficiency issues.

Very few works have considered indexing mechanisms based on mono-dimensional descriptors. Pua et al. [25] proposed a hashing mechanism based on color moment vectors to efficiently retrieve repeated video clips. Oostveen et al. [23] used a lookup table to store fingerprints based on mean luminance of image blocks. Both these methods use exact matching of image fingerprints to locate a candidate sequence rather than exhaustive search in the fingerprint set, and are thus very efficient. These two works are the closest to our own.

1.3 Challenges and overview

1.3.1 Requirements

This report proposes a very fast method for detecting common elements between two video streams. The purpose of our research is to apply this method to television streams for monitoring and indexing purposes. It is therefore assumed that the set of transformations between two repetitions of the same segment is rather small. These transformations are basically coming from broadcast noise (additive gaussian noise, color shift, digitization...)

and also from edition at the production stage (small temporal variations, text, banners...). These latter transformations are supposed to be limited, in the sense that if the difference is too large, then the images surely appears in different context, and should therefore not be seen as "repetitions". Our view is quite different from both the fingerprinting/watermarking and retrieval paradigms, and the method will differ adequately.
Apart from being robust to this limited set of transformations, a desired property is to be able to search efficiently in a fairly large video dataset. In order to detect repetitions a kind of reference dataset must indeed be built to act as the "memory" of the channel. This memory should be a long term memory, e.g. able to remember programs lead-in and lead-out, as well as a short term memory, e.g. able to remember trailers or commercials. The dataset has to be rather large, but not huge, since it should reflect the current state of the channel broadcast. The dataset is therefore dynamic with video data coming in and out, resulting in a video dataset of a somewhat constant size. The constraints on the video dataset are not as hard as in [11] where the dataset may grow endlessly. We estimate that a size of 100 to 200 hours should be far than enough for our purpose. However, the precise evaluation of the dataset size and the dynamic update of the dataset are left for future research.
A more original constraint is the query size. Almost all previous work considered the query to be a small clip. In our context, the query is a live video stream or a really large video file (24 hours in practice). Some new problems emerge, such as temporal segmentation of the query, and impose some hard time constraints on matching two video segments. These constraints should result in a process able to handle a 24 hours query against a 200 hours reference dataset in a rather short time.

1.3.2 Method overview

Consider the problem where a video recorded from television has to be indexed. This article does not consider the whole indexing mechanism, the reader is referred to [22] for this purpose, but only the detection of repetitions, i.e. identifying which parts of this video have already been broadcasted and belong to a reference video dataset (RVD). The video to be indexed is the query, and a typical query size of 24 hours is assumed throughout the article. The RVD is assumed to be 100 to 200 hours.
The first problem is to define the elements on which the comparison will be performed, i.e. find a temporal segmentation for both the query and the RVD that allows to match small video segments without performing a full linear search in the RVD. Section 2.1 explains why shot segmentation is a reasonably good choice.
The crucial step is the choice of the features to represent an image. The quality and number of features of course deeply impact the effectivness but also drives the retrieval complexity. Most previous work on this topic [11, 31, 13] favor robust descriptors or descriptors with multiple features, sometimes only computed on keyframes. Unfortunately these descriptors generate high dimensional spaces in which search is very difficult to perform. A less popular solution is to choose a single non-robust but discriminating descriptor, which can be easily searched using hash or look-up tables, such as in [23, 25]. Despite the weakness of the descriptor these methods can nonetheless reach high effectiveness by using tempo-

ral information more effectively than the others methods. This latter solution is the most suited to our context. The descriptor, or signature, is built for each video frame from the low-frequency coefficients of the Discrete Cosine Transform (DCT). Section 2.2 details the signature construction.
The next step is to define an architecture for the storage and retrieval. A signature is used as a key to a hash table to retrieve the shot containing this specific signature. Section 3 explain this database organisation. Section 4 defines how to decide that the retrieved shot is the repetition of the query.

2 Features extraction

2.1 Temporal segmentation

A first issue is to define the elements on which the comparison will be performed. In most previous researches, the query is a small manually segmented clip, a notable exception being Joly [11]. In early works, this clip is compared to the RVD sequentially, i.e. the distance is computed at every possible positions in the RVD. This is the naïve method and this is of course not efficient even with a rather small dataset. A temporal segmentation of both the query Q and the RVD will help to define these elements of comparison as well as help the search by defining a structure on the stream which can help the search process.
Shot segmentation is a good candidate because it is quite uncommon in television broadcast to edit the shots themselves. A shot is thus chosen as the basic unit for recognition. Another property is that the partition created by the shot segmentation is a first structure of the stream. This partition is still way too precise to make sense from a global point of view, but the global structure can be found by merging shots. This is actually a very popular idea [1, 24, 26, 14] and proves to be a very effective way of finding scenes or structures in a video. Shot segmentation is unfortunately not a perfect process and inaccuracies might appear, but this is not critical. It is important for the shot segmentation not to miss important boundaries like commercial ending or boundaries of a repeated segment. However, shots may be missed or falsely detected in a feature film without any quality loss from the recognition point of view. The most needed feature for the shot segmentation is its repeatability, i.e. identical videos broadcasted at different times must have the same shot segmentation.
A standard algorithm based on adaptative thresholding of luminance histogram based on [29] is chosen, with improvements to detect dissolves and fades. The algorithm is robust enough to give satifactory results on any kind of video from different television channels, without any thresholds modifications.

2.2 Signature definition

Defining the signature is the most important part of the algorithm. Our choice is to define a simple, one-dimension descriptor rather than building a complex set of descriptors which

will generate a high dimensional space. With this goal in mind, a signature that can be mapped into a 64-bit integer is build for each frame.
One of the most common way to reduce dimensionality in image processing is to work in the Discrete Cosinus Transform (DCT) domain. This transform is widely used because for natural images, it approximates the Karhunen-Loeve transform (KLT), which provide optimal energy compaction and coefficients decorrelation [10]. It has been shown that the DCT is one of the best approximation, without the computational burden of the KLT, which is data-dependent.
For an image I of size (N, M) , the $DCT(u, v)$ coefficient is given by :

$$DCT(u,v) = \alpha(u,v) \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} cos\left[(2x+1)\frac{\pi u}{2N}\right] cos\left[(2y+1)\frac{\pi v}{2M}\right] I(x,y)$$

with $\alpha(u,v) = \frac{2}{\sqrt{NM}} C(u)C(v)$ and $C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u = 0 \\ 1 & \text{otherwise} \end{cases}$

The DCT transform is applied on the whole image (luminance channel only) thus capturing global properties. The next step is to use the dimensionality reduction property of the DCT in a rather extreme way, by extracting a small superior left nxn submatrix in the DCT matrix and by quantizing it aggressively, i.e. each coefficient is quantized on only one bit.
Since very little information is kept, a very discriminating feature must be encoded by the quantization scheme. One fairly intuitive idea is to encode the sign of the DCT coefficient, which is known to be a robust feature. The DCT nxn submatrix is thus quantized as such:

$$\sigma(i) = \begin{cases} 1 & \text{if } DCT\left(\lfloor \frac{i}{n} \rfloor, i - \lfloor \frac{i}{n} \rfloor n\right) \geq m \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i \in [1, n^2]$$

To encode the sign we can merely take $m = 0$. Here however, m is taken as the median value of the first n^2 coefficients. This is almost equivalent since the median is usually very close to zero, but it emerged as slightly more robust and satisfactory from a theoretical point of view: it has maximal entropy, i.e. the information kept in the descriptor is maximized by this code. A similar quantization scheme has already been proposed by Coskun et al. [6] and Barr et al. [4] for copy detection purposes, and has been shown to be robust to severe transformations. However, our emphasis is here more on size than robustness.
Although the sign is a robust feature it may be worrying to lose all the modulus information. As an alternative choice, we introduce a quantization scheme with deadzone aiming at encoding the signifiance of the coefficient. All coefficients in the deadzone are considered insignificant and encoded as '0'. The boundaries of the deadzone are given by the inter-quartile range. Every coefficient outside of the deadzone is encoded as a '1'. The resulting descriptor has also maximum entropy.
These two quantization scheme results in a vector of size n^2, n is chosen between 5 and 8, allowing the binary vector σ to be mapped into a 64-bit integer. Note that for our purpose the DC coefficient is removed because it is useless in its quantized form. The DC coefficient,

Figure 1: Original Lena image

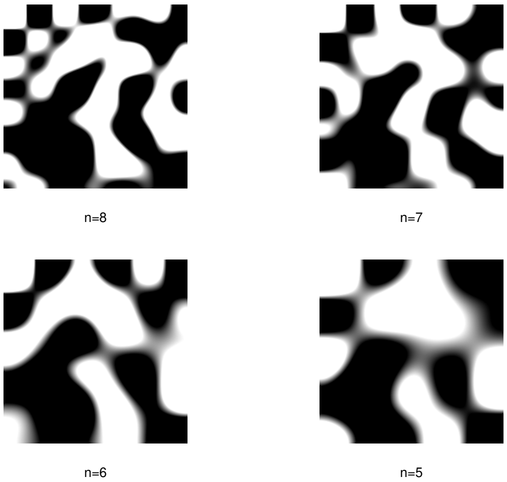

Figure 2: Reconstructed image of Lena based on its signature

i.e. DCT(0,0) is the mean and is always the greatest coefficient. The DC coefficient is thus always quantized to 1 and does not bring any information. The following coefficient, DCT(n,0), is append to σ to make a 64-bit signature.

Figure 2 shows a visual interpretation of the signature of the Lena image. These images are obtained by applying an inverse DCT on the quantized DCT matrix, where all coefficients are set to zero, except for the upper-left nxn sub-matrix which is quantized as explained above. It is interesting to see the information kept in the signature, especially for $n = 8$ where the shape of Lena can be guessed, although we have only 64 bits of data.

2.3 Features extraction process

Algorithm 1 sums up the feature extraction process.

Function FeatureExtraction(V : **video stream**) : **list of shots**
    $S^V = ShotSegmentation(V)$;
    For each shot S_i^V in S^V **do**
        For each image I_{ik} in S_i^V **do**
            $\sigma_k = ComputeDescriptor(I_{ik})$;
            $S_i^V = S_i^V \cup \sigma_k$; // attach each descriptor to its shot
        end For
    end For
    return S^V;
End

Algorithm 1: Feature extraction

3 Database organization

3.1 Strategy

The main idea behind the database organization is to take advantage of the fact that *exact matching* can be used betweeen the signatures. This property allows to use fast retrieval data structures such as hash tables, which provide constant time insertion and look-up.
In practice, a pair (i, k) is stored in a hash table for every image, using its signature as the key. i is the index of the shot containing the signature and k is the signature position in this shot. Shots are themselves stored in an array A. The shot which contained the signature is then easily retrieved by $A[i]$. Algorithm 2 sums up the database construction in the general case.
Two methods of database organisation are defined:

- Single Reference Scheme: a key points to a single pair (i, k), i.e. for a given signature, the pair (i, k) is stored only if the signature is not already in the hash table.
- Multiple Reference Scheme: a key points to several pairs, i.e. several candidates shots are considered for a single signature. Though it has higher memory demands, this scheme is hoped to lead to better results. This is also the approach used by [23] and [25] and is very close to the inverted file technique used in text retrieval [9].

Storing the signature position k is not necessary. This information is in fact used by the best of the algorithms presented in section 4.2.5 and is thus included in the database construction.

```
Function databaseConstruction( V : video stream) : (array of shots, hash table)
    S^V : array of shots;
    ht : hash table;

    S^V = featureExtraction(V);
    For each shot S_i^V in S^V do
        For each descriptor σ_ik in S_i^V do
            ht(σ_ik) = (i, k);
            // i: index of the shot
            // k: descriptor position in the shot

        end For

    end For
    return (S^V, ht);

End
```

Algorithm 2: Database construction

3.2 Hash function

A hash table is a data structure that allows to retrieve an element in a constant time, i.e. O(1). It is composed of an array and a hash function. This hash function maps an element from the input domain E to the hash domain H which is usually a subset of $\mathbb{N}$, $h : E \rightarrow H$, with $|H| \ll |E|$. h is not one-to-one, meaning that two different elements of E can be mapped to the same hash value. This event is called a collision. The choice of h is crucial for the search performance. Designing h usually means to find a trade-off between minimising the number of collisions, and low complexity.
In our case, the input domain is the signature domain S which is considered as a binary domain of dimension N: $S = \{0, 1\}^N$. Since the signature is a 64-bit integer the theoretical size of S is $|S| = 2^{64}$ which is huge. The property of maximum entropy of the descriptor reduces the allowable space to a size of $|S| = \begin{pmatrix} N \\ N/2 \end{pmatrix}$, which is still huge ($\approx 10^{18}$). The hash domain is also defined as a binary domain $H = \{0, 1\}^M$, of size $|H| = 2^M$. In practice,

$M = 32$. The next section gives bounds for the number of collisions N_c by working in the ideal case where the hash function has a uniform distribution.

3.2.1 Study of the uniform distribution

We now suppose that h has a random uniform distribution. A collision is the event:

$$\exists\,(s_1, s_2) \in S\ s_1 \neq s_2 \ \text{ and } h(s_1) = h(s_2)$$

p signatures are to be stored in a hash table of size $|H| = 2^M$. If $p > |H|$ the pigeonhole principle states that the probability of collision is one. If not, this the classical "birthday problem" and the probability of collision is given by:

$$P_{collision} = 1 - \prod_{k=0}^{p-1} \left(1 - \frac{k}{|H|}\right)$$

What is however more interesting when studying hash tables is the expected number of collisions N_c, that can be estimated by [21]:

$$N_c = \binom{p}{2} \frac{1}{|H|} = \frac{p(p-1)}{2|H|}$$

if $M = \log_2(p)$ then we have enough space for every input element and h could be one-to-one, i.e. collision-free ! However, note that since h is a random function, we still have $N_c \approx 2^{M-1}$, meaning that even in the "best" case we still have a rather large number of collisions. If we really want to reduce the collisions to a minimum, for example, $N_C < 1$, we should have $p < \sqrt{2^{M+1}}$. Considering our current case of a hash table of size $2^M = 2^{32}$, the maximum size of the signature subset S_{real} would be only 92681 elements, a little bit more than 2 hours of video[1] ! We thus have to find a trade-off between the size of the hash table and an acceptable rate of collisions.

One interesting point is to find the minimum size of the hash table which can give an acceptable number of collision, for example N_c must be below a fraction of the input size , i.e. $N_c < \frac{p}{r}$, which yields:

$$|H| > \frac{r(p-1)}{2}$$

Conversely, we wish to know the maximum number of elements that S_{real} can hold, given a constraint r and a hash table size $|H|$:

$$p < 1 + \frac{2|H|}{r}$$

With $|H| = 2^M = 2^{32}$, $r = 100$, the inequality yields a maximum size of 86 millions signatures, about 80 days of video, which is quite comfortable.

[1]it has been observed experimentally that in a segment of N images, only $N/2$ signatures are distinct

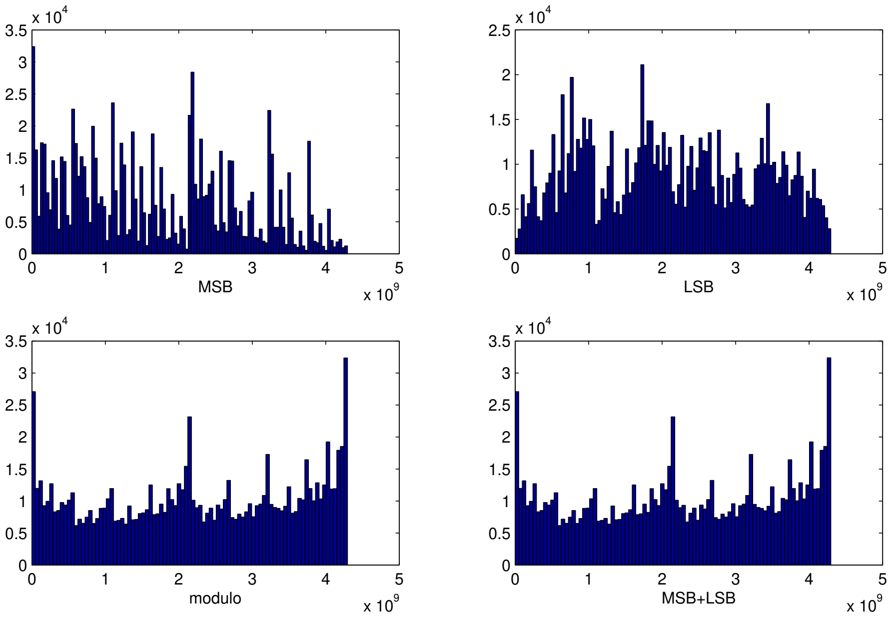

Figure 3: Distributions of the proposed hash functions

3.2.2 Discussion on several hash functions

To prevent collisions we must look at the initial distribution of the signatures and find a way of transforming it into a uniform distribution in the reduced space. The function has however to be as simple as possible. We thus define hash functions with only elementary operations such as shifts and sums. The modulo function is usually a popular one and is presented here for comparison purposes.

1. $h(\sigma) = \sigma_{1 \text{ to } 32}$ (MSB - Most Significant Bits)
2. $h(\sigma) = \sigma_{32 \text{ to } 64}$ (LSB - Least Significant Bits)
3. $h(\sigma) = \sigma \mod (2^{31} - 1)$ (modulo)
4. $h(\sigma) = \sigma_{1 \text{ to } 32} + \sigma_{32 \text{ to } 64}$ (LSB+MSB)

Experimentally, the LSB function is the fastest. The LSB+MSB function behaves also quite well. In order to justify the choice of the hash function, an histogram is built to visualize and estimate the closeness of the distribution to the uniform one. The Freedman-Diaconis rule [8] is used to estimate an acceptable bin width to perform distribution estimation. The bin width W is given by:

$$W = \frac{2 * IQR}{\sqrt[3]{N}}$$

where N is the number of samples, and IQR the interquartile range. To give a hint of how close the distribution is to the uniform one, the Kullback-Leibler divergence (KLD) is computed. The KLD is defined for two distributions p and q as:

$$d(p, q) = \sum_k p_k \log_2 \frac{p_k}{q_k}$$

The KL divergence, the standard deviation of the histogram and the percentage of collisions are given in Table 1. The LSB function achieves the best value for the standard deviation, although it does not have the closest distribution to the uniform one in the Kullback-Leibler sense. The Kullback-Leibler divergence gives a more global opinion on the smoothness of the distribution, while the standard deviation measures the fact that the samples are more evenly distributed around the mean, as can be seen on Figure 3.
An example of the impact of the hash function complexity is given by functions modulo and MSB+LSB. These functions have in fact very similar distribution. Let's justify this rapidly. For n a 64 bit integer and $p = 2^{32} - 1$ we have

$$n = (p + 1) * MSB(n) + LSB(n)$$

and thus

$$n \mod p = MSB(n) + LSB(n)$$

for $LSB(n) \neq p$, $MSB(n) \neq p$. In practice though, the LSB+MSB is a bit faster, because of its lesser complexity. The retrieval time indicated in Table 1 is obtained by querying a

Table 1: Hash functions properties

Function	**KLD**	**Std deviation**	**Collisions (%)**	**Retrieval time (s)**
Uniform	0	0	0.01	-
MSB	0.44	6755	42.9	1.34
LSB	0.12	3777	19.1	1.24
Modulo	0.1	4269	2.3	1.47
LSB+MSB	0.1	4269	2.3	1.4

24-hours video file against a RVD of one week. The collisions are resolved through chaining. What is really hindering the retrieval is to have some buckets with a large number of records. The percentage of collisions (i.e. the percentage of input signatures which will actually suffer collision) may be high as long as the collisions are distributed in many buckets. The most pertinent indicators are thus the standard deviation and of course the actual retrieval time. The choice of the function is now indeed to find a trade-off between the good properties of its distribution and its low complexity. From the retrieval time presented in Table 1 the LSB function seems a good choice It is also very easy to implement since the LSB hash function is a simple cast from a 64-bit integer to a 32 bit-integer.

RVD : **video stream**; *[Reference Video Dataset]*
Q : **video stream**; *[Query]*
ht : **hash table**;
D : **shot distance function**;
Threshold : **integer**;
S^RVD, S^Q : **array of shots**

$(S^{RVD}, ht) = databaseConstruction(RVD)$;
$S^Q = featuresExtraction(Q)$;
For each shot S_i^Q of Q **do**
  For each descriptor σ_{ik} of S_i^Q **do**
    If $(\sigma_{ik} \in ht)$ **then**
      $(n,p) = ht(\sigma_{ik})$;
      // n: index in S^{RVD} of the shot containing σ_{ik}, i.e. S_n^{RVD}
      // p: position of the matching signature in S_n^{RVD}

      If $(D(S_n^{RVD}, S_i^Q) < Threshold)$ **then**
        break; // shots match, go on to the next query shot
      end If

    end If

  end For

end For

Algorithm 3: Retrieval process

4 Retrieval Method

4.1 Algorithm definition

Given a query shot, our goal is to determine if this shot has been broadcasted before, i.e. if this shot is present in the RVD. Algorithm 3 shows the different steps followed by the retrieval process.

It works as follows: once the features have been extracted and the database has been built, the signatures of the query shot are queried one by one against the hash table. If a signature yields a match then a candidate shot from the RVD can be retrieved thanks to its index and a global distance between the query and the candidate shots is computed.
Note that a single signature of the query shot is sufficient to retrieve a candidate shot. This is an essential property of the retrieval algorithm because all the signatures of two duplicate shots are not equal in practice. Although the signature was built to be invariant to allowed transformations, transmission noise prevent this to happen in general. However there is a very high probability that two identical shots will share at least one common signature. This is the main assumption of the method and results in Section 5 show that this assumption is correct most of the time. Therefore all the signatures of the query shot are tested until one of them yields a correct match.
The definition of various distances for matching two shots together is the subject of the next section.

4.2 Shot Distance definition

The DCT signature is built to be used as a key to a hash table and is thus a bit crude. It might not be the best solution to use it as a feature when computing distances between shots. However, as a first solution, it is a good compromise to use the signature as a feature because it prevents us from computing costly features and still yields more than acceptable results (see section 5).

4.2.1 Naïve distance

Since the signatures can be compared using a simple equality test, a voting procedure, i.e counting the number of equal signatures in two shots, could be considered. However, because small variations in the frame may also cause very small variations in the signature, the number of frames with identical signatures may be low. On the contrary, the Hamming distance is well suited to measure small variations between the signatures vectors.
Consider two shots $S_q = \{\sigma_{q_1}, ..., \sigma_{q_N}\}$ and $S_c = \{\sigma_{c_1}, ..., \sigma_{c_M}\}$ and suppose that the ideal case where $N = M$ happens. The naïve distance between two shots $D_{Na}(S_q, S_c)$ is defined as the average Hamming distance:

$$D_{Na}(S_q, S_c) = \frac{1}{N}\sum_{i=1}^{N} d_h(\sigma_{q_i}, \sigma_{c_i})$$

with $d_h(\sigma_{q_i}, \sigma_{c_i})$ is the Hamming distance between the binary vectors σ_{q_i} and σ_{c_i}. It is called naïve because the case where $N = M$ almost never happens.
Figure 4 shows the effectivness of the Hamming distance between the signatures. This figure represent the naïve distances between a hand-picked shot of 100 frames and every possible positions of a 100-frames shot in a 24-hours Reference Video Dataset (RVD). The test shot belongs to the RVD (near frame 250000). Two others instances of this test shot are easily

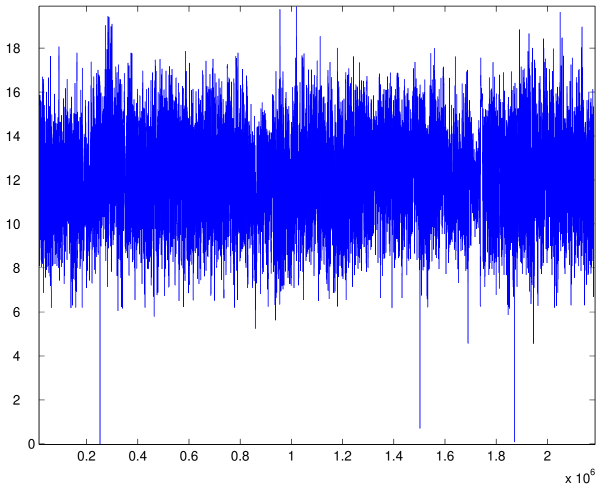

Figure 4: Naïve distances between a 100-frames commercial lead-in and a 24-hours video

identified in the 24-hours video. This figure also helps to choose the threshold α which represent the mean number of bit errors allowable by signature. The threshold is set to $\alpha = 4$ in practice.

The naïve distance is however not usable as such. Because of complex and various transition effects or because the property of repeatability desired for the shot segmentation algorithm is not respected, two identical video sequences may have a very different temporal segmentation. For example, consider a channel lead-in that is broadcasted many times over one day, but which is introduced and ended with different transition effects. The shot segmentation algorithm will produce shots of different size, with a possible large shift in the frame order. One shot may also be split in two or three because of false alarms in the shot segmentation, and conversely, shots may be concatened in only one because of missed transitions. The next sections are dedicated to the definition of distances robust to such effects.

4.2.2 Sampled Naïve Distance (SND)

This distance is a simple adaptation of the naïve distance to shots of different lengths. The shots are linearly sampled to have roughly the same size and the larger one is then truncated, yielding eventually shots of identical length. The naïve distance is then computed on these sampled shots. Note that this a very basic definition and that it is present mainly for comparison purposes.

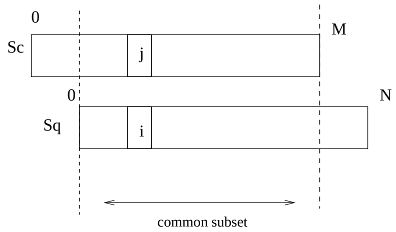

Figure 5: Shot alignment with AAHD

4.2.3 Exhaustive search

Consider a query shot S_q and a candidate shot S_c of respective length N and M, $N < M$. We want to find the position in the candidate shot that the minimize the naïve distance between S_q and a subset of S_c of length N. Formally, we look for the position k_{min} :

$$k_{min} = \text{Arg}\min_k \sum_{i=1}^{N} d_h(\sigma_{q_i}, \sigma_{c_{i+k}}) \text{ with } 0 \leq k \leq M - N$$

While this is a very basic search strategy, the exhaustive search is not intractable even for a very large RVD, because we are only comparing shots between themselves. M and N are therefore quite small in general. Note that this search may be easily quickened up by stopping the search as soon as the distance falls below the threshold.

4.2.4 Normalized Edit distance

Many authors have proposed to view a sequence of images as a string, and use string matching algorithms to resolve this problem of alignement [19, 16]. The edit distance is often used for this purpose and may be more robust to noise or to complex video editing as mentionned in [2]. One important drawback is that the edit distance is sensitive to the length of the strings and requires some type of normalization. Marzal and Vidal [20] have defined the *Normalized Edit Distance* which solves this problem and could therefore be used.

4.2.5 Aligned Average Hamming Distance (AAHD)

This distance is based on the relative positions of the signatures in the shot. Finding a candidate shot S_c is based on the fact that a signature σ_{c_i} has matched to a signature σ_{q_j} using the hash table, i.e. $h(\sigma_{c_i}) = h(\sigma_{q_j})$. Suppose now the relative positions of these signature in their original shots, i.e. i and j, have also been stored. The method uses i and j to align the shots and to compute the naïve distance between the common subset of S_q and S_c. Figure 5 gives an example of a such an alignement. Note that several positions may be tried, since every signature of the query shot is being tested. The same candidate shot may be found several times, with a different alignement.

Table 2: Signature robustness

Transformation	**PSNR (dB)**	**SSIM**	**Identical sgn (%)**	**Hamming distance**
Gaussian blur	28.8	0.93	95%	0.38
Brightness change 10%	21.2	0.94	10%	2.5
Small text	35.5	0.99	79%	0.97
Random noise 3%	31.6	0.83	76%	0.39
Large text	25.6	0.97	3%	4.3

5 Results

All the tests except the simulations in 5.1.1 were made on two video files recorded from the same french channel on 2 consecutive days. The file used as a reference video dataset is 24 hours long (2180727 frames, 19046 shots) and the file used as a query is 1 hour long (85601 frames, 734 shots). The videos are encoded in MPEG-2, in PAL resolution (720x576) at 25 frames/s. Four sizes of signature $n = 5, 6, 7, 8$ were tested, producing binary vectors of size 25, 36, 49 and 64.
To perform detection we extract metadata from the video stream. These metadata are a list of shots containing the signature of every image of the shot. It may also contain a label if these metadata are to be used as a RVD rather than as query. These metadata weight only 17 Mo for a 24 hours video i.e. 0.04% of the original file size.

5.1 Signature robustness

Retrieval of a candidate shot is uniquely based on the capacity of the query signature to have at least one exact correspondance in the candidate shot. It is therefore critical to test if this assumption holds. Two experiments are conducted. The first one operates on a single short video sequence subjected to various simulated noises. The second one test the robustness of the signature on real data.

5.1.1 Simulations

This section evaluates the robustness of the signature when subjected to various noises. A short sequence of 110 images is filtered with different noises considered to be representative of the broadcast noise on television sequences. VirtualDub [17] is used to create the noisy sequences. Figure 6 shows some of these. In Table 2, for each noisy sequence we give the classical PSNR and the SSIM of Wang and Bovik [30], to give both an idea of the amount of modified data and the visual impact. The results are given by two indicators: the percentage of signatures that have not been modified by the transformation, and the average Hamming distance between the original sequence and the noisy one. It shows that the signature is robust to common noise like blur or random impulse noise but withstand insertions with difficulty. These tests are made with $n = 8$.

Figure 6: Test videos. On the left: original, middle: small text, right: large text

5.1.2 Retrieval capacity on real data

The retrieval capacity of the signature is tested by doing the following experiment. Each signature of the query file is queried against the hash table, to see if the signature can retrieve an identical shot if there is one, or if not, it is checked that it does not produce any false alarms. The results are expressed in terms of precision and recall in Table 3 for the simple and multiple reference schemes. This test procedure is diffferent from the shot matching procedure devised earlier since here every single signature is queried against the hash table. In practice only one exact signature by shot is needed.
It is important to understand that the recall does not need to be very high because only one signature by shot is enough to retrieve a shot. On the contrary, precision is important since false alarms will lead to useless computations. We cannot therefore conclude on the superiority of the simple or multiple reference scheme over the other for now. However, it is obvious that $n = 5$ produces too weak signatures (only 25 bits per image !). It leads to a considerable number of false alarms and thus cannot be trusted.

Table 3: Signature retrieval capacity

	Simple reference		Multiple reference	
n	Precision	Recall	Precision	Recall
8	98.7	79.3	96.5	98.4
7	98.3	85.2	97	99.2
6	94.1	90.3	95.7	99.6
5	58.9	94.1	70	99.9

5.2 Shot Matching

5.2.1 Signature choice

The first experiment aims at deciding which size of signatures should be used and which kind of database organization performs better, i.e. single or multiple reference scheme ? Tables 4 and 5 presents the results of the complete repetition detection for two distances: exhaustive and AAHD. The next section will show that these distances are actually the worst and the best of the distances defined so we are not considering any particular case.

The most obvious fact from these Tables is that the multiple reference scheme does not

Table 4: Exhaustive search

	Simple reference		Multiple reference	
n	Precision	Recall	Precision	Recall
8	99.3	97.3	98.6	97.3
7	96.6	97.2	96.6	97.2
6	92.6	96.5	92.7	97.2
5	70.3	97.9	68.1	98.6

Table 5: AAHD

	Simple reference		Multiple reference	
n	Precision	Recall	Precision	Recall
8	99.3	98	98.6	98
7	95.9	98.6	95.9	98.6
6	92.2	98.6	92.2	98.6
5	66.3	97.9	62.9	97.2

bring any ameliorations. Since it is also more demanding in memory and processing power, it will not be considered further. The results show in fact that the signature is robust enough to avoid a scheme like multiple reference.

Let us now focus on the results for different size of signatures, i.e. n. It does not come as a surprise that the precision decreases with n. Most of the false alarms are almost monochrome shots, that may be easily filtered out. Precision is however crucial since two shots that have nothing in common may be linked together, thus confusing a process trying to make sense of these repetitions, as in [22]. The case of $n = 5$ where "real" false alarms appear due to the weakness of the signature has to be avoided.

Concerning recall, the missed shots were caused by artifacts due to the interlaced mode of television, and incoherent shot segmentation. Static shots in which the signature is constant are also a problem because the retrieval becomes an all or nothing mode. In general, shots that contain natural images are very well identified, but shots containing very few information or synthetic images are far less well represented by the DCT, and may be missed or falsely recognized. Figure 7 shows an exemple of shots that might be missed. Since high precision is preferred over high recall, a signature size of 64 bits, i.e. $n = 8$, will be used from now on.

Figure 7: Example of possible miss

5.2.2 Quantization scheme comparison

Table 6 gives the results for the two quantization schemes introduced in section 2.2. $Q1$ is the quantization scheme which encodes the sign of the coefficient and $Q2$ encodes its significance. $Q1$ has a higher recall than $Q2$ but seems to be more prone to false alarms. $Q1$ seems to be the best although the limited ground truth prevents to reach a clear conclusion. These two quantization scheme prove nonetheless their effectivness. Q1 is used from now on as the default quantization scheme.

Table 6: Quantization schemes

	query 24h RVD: 1h		query: 1h RVD: 24h	
method	Precision	Recall	Precision	Recall
Q1	100	96.6	99.3	98
$Q2$	100	95.1	100	96.6

5.2.3 Shot distances comparison

The different distances defined in section 4.2 are examined here. Precision/Recall curves for the different distances are given in Figure 8.
What may be surprising is that the exhaustive search is actually the worst. It may be explained by the fact that the problems usually happen at shot boundaries, which are not

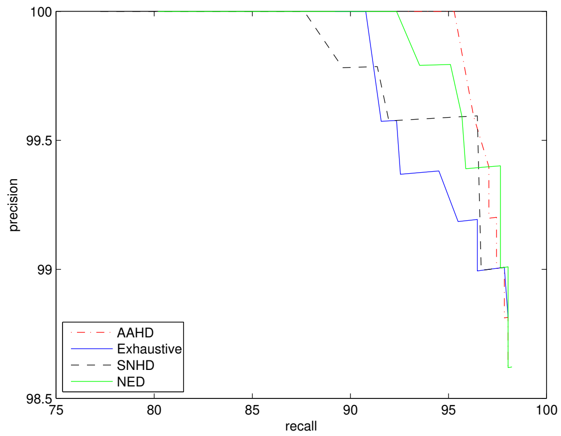

Figure 8: Precision/Recall curves for different shot matching distances

handled by the exhaustive search. On the contrary, AAHD may consider only a subset of the shots, thus finding alignements that the exhaustive search does not explore. This is especially helpful when identical shots are begun and ended with different progressive transition types. The normalized edit distance behaves quite well, with the usual minor defect that it may lose a bit precision in favor of recall. It is however far more complex than other methods, as can be seen on figure 9. This figure shows the time spent in the matching process, which includes hash table lookup and retrieval as defined in section 4.
From Figure 8 and 9 it is clear that the best distance is the AAHD. This is the distance which will be considered in the following.

5.3 Efficiency and Scaling

Speed was a key factor in designing the algorithm. This section is dedicated to test the method on a large scale, using the best parameters deduced from the previous sections, i.e. size of signature of 64 bits, single reference scheme, AAHD. Two process are distinguished: metadata extraction, and shot matching.
Metadata extraction means shot segmentation and signature computation. Since video decoding is needed, the decoding process is also given as a reference: 230 frames/s on a standard 3Ghz PC running Linux Fedora 3. Metadata extraction performs at a frame rate of 115 frames/s, twice as slow as decoding, but still much faster than real-time.

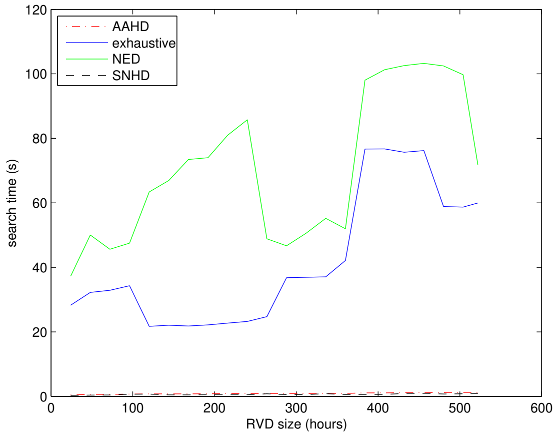

Figure 9: Efficiency comparison of different shot distances. query=24 hours, RVD=24 to 500 hours

5.3.1 Study of AAHD consistency

Since AAHD emerged as the best solution, this section is dedicated to analyse its performance with large video RVDs. The test data set is three weeks of French Television cut into 21 files of 24 hours each. The elementary test is to query a file against another one, count the number of detections and measure the search time. Each combination of file is tested, a file playing alternately the role of RVD or query. Figure 10 shows the results. These results are not easy to interpret since outliers can be seen in the right hand side of the figure. Some test takes indeed 8 seconds whereas the average is 1s. This stems from the fact that in the retrieval algorithm, every single signature of the query shot is tested (because it might lead to different alignments) until the shots are recognized to be identical or the number of signatures in the query shot is exhausted. What may happen is that if two shots share a high number of identical signatures but the global distance never falls below the threshold, then many distances are calculated in vain. To overcome this drawback we set a parameter k to the maximum number of tries in a single shot. Figure 11 shows the evolution of the average search time with k.

Figure 12 shows some results with different values of k. Note that for $k = 1$ we easily see the linear dependancy between the number of detections and the search time. In practice $k = \infty$ may be a acceptable value since the maximum search time is below 10 seconds. However if a quick answer must be made and jitter in the response time is not welcomed, Figure 11 shows that $k \approx 20$ is a reasonnable choice, yielding a very small response time while missing only a few detections.

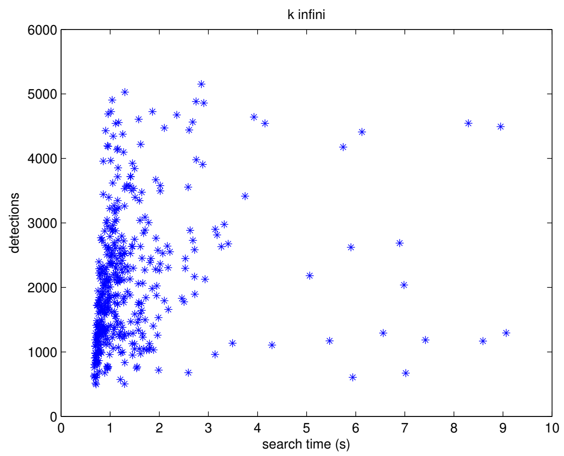

Figure 10: Number of detections versus search time, $k = \infty$

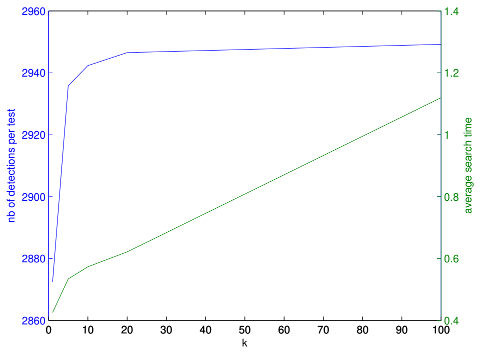

Figure 11: Evolution of the average number of detections versus average search time for different k

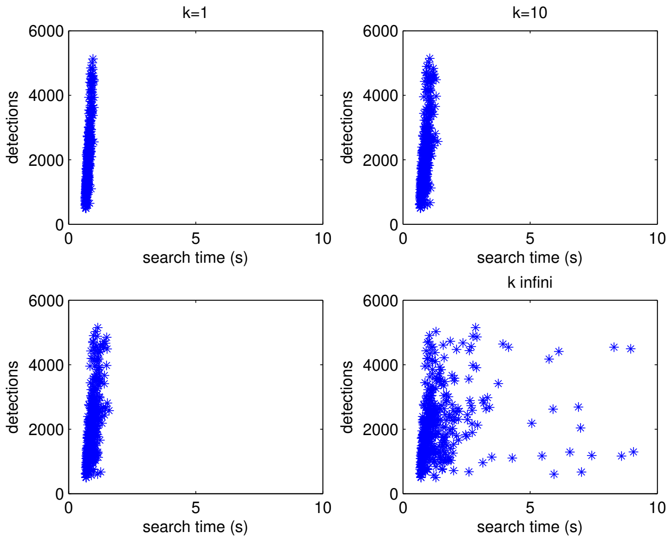

Figure 12: Influence of k over the search time

5.3.2 Scaling to large video dataset

The nature of the retrieval method should implies that the RVD and the query sizes have no influence over the search time because of the constant lookup time provided by the hash table. What has however an influence is the number of shot distance computations. This number of distance computations is clearly correlated with the number of *detections* because a detection means that at least one distance has been computed. It also seems to be correlated with the size of the RVD since a large RVD increases the number of false alarms of matching signatures, which therefore leads useless distance computations. Figure 13 indeed show this dependency between the number of detections and the search time. The query size is 24 hours, the RVD size ranges from 24 to 500 hours, AAHD with $k = 20$ is used. Two different queries are shown. On the left-hand side the query has been recorded one week before the RVD, while on the right-hand side the query is 6 months old. The former has a steadily increasing number of detections while the latter has a number of detections which only slightly increase with the RVD. Figure 13 shows that the search time overal slightly increase with the size of the RVD, but can also decrease ! It thus show that the response time is not dependant from the size of the RVD but rather from its content. It also shows that the method can put up with datasets of 500 hours and yield a response time only slightly superior to 1s. Figure 10 also shows that even with a high number of detections, the response time is still well under 2 seconds.

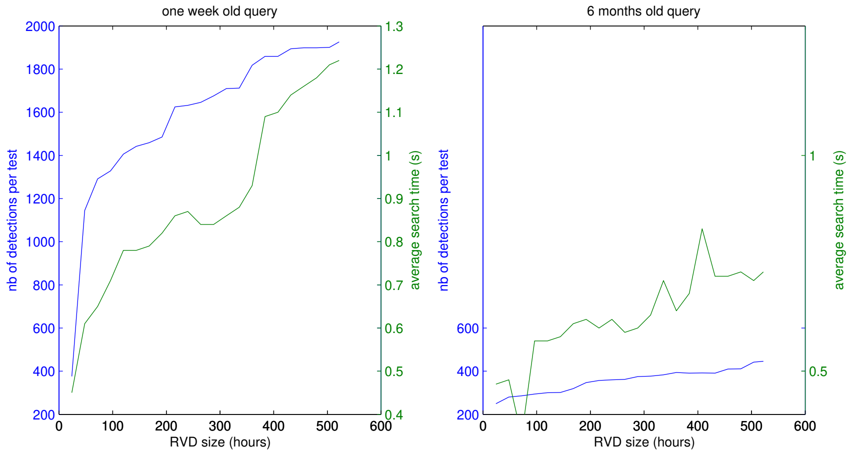

Figure 13: Influence of the RVD size and the number of detections on the retrieval time

6 Detecting Repeats

Detecting the common elements between a query video and a RVD is now something that can be done easily using the method just presented. This section gives some insights about the actual interest of repetitions for video structuring, mainly by giving some applications and results.

6.1 RVD influence over the number of detections

One of the first thing to notice is the influence of the size and 'quality' of the RVD upon the retrieval results. Table 7 shows the retrieval results with the same two files used in section 5 except that query and RVD are swapped.
One obvious thing is that if the RVD is small, the number of detections per hour is also very small because the RVD does not contain enough data to be recognized. What is however interesting is that the recall is slightly lower when using a small RVD, meaning that a redundant RVD improves the retrieval results. Indeed if the RVD contains multiple instances of the same video clip, with possibly different transformations due to noise, it increases the retrieval probability.
All the RVD used in these tests are non structured, i.e. they have not been edited by hand. A RVD is just a recording of TV broadcast of a certain length. A carefully built RVD would increase the number of detections and their quality. By careful, we mean that the RVD should include only clips supposed to be re-broadcasted Multiple instances of the same

video clip are also desired since it increases the probability detection.

Table 7: RVD size influence

24h RVD - 1h query			1h RVD - 24h query		
Precision	Recall	Detections/h	Precision	Recall	Detections/h
99.3	98	145	100	96.6	20.5

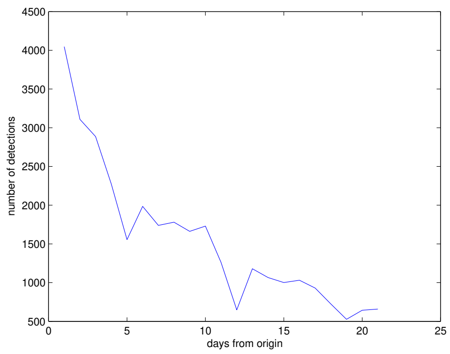

Figure 14: Evolution of the number of detections over 3 weeks with a static RVD. The small drops happen on a saturday, which happens to have less (or different) commercials than the other days.

Another interesting test is to measure the number of detections over a long period of time with a static RVD. Figure 14 shows the number of detections over a period of three weeks. The RVD used is 24 hours long and is the day just before the three weeks test.
One surprising but welcome figure is the number of detections between two consecutive days. In this example the number of common shots between two consecutive days is 4411 or put another way, about 20 to 25% of the shots have already been broadcast the day before !
This remanence drops sharply in the next few days. If the RVD is 6 months old with respect to the query, only 200 common shots are left between the two days. This of course shows the necessity of updating the RVD to give as much detections as possible to increase the precision and reliability of the structuring process.

6.2 First applications

One of the first application of our method is commercial monitoring. Commercial monitoring is about checking that commercials from a certain company are aired the way they intend it to be. The system has to monitor the TV stream and give back the instants where the

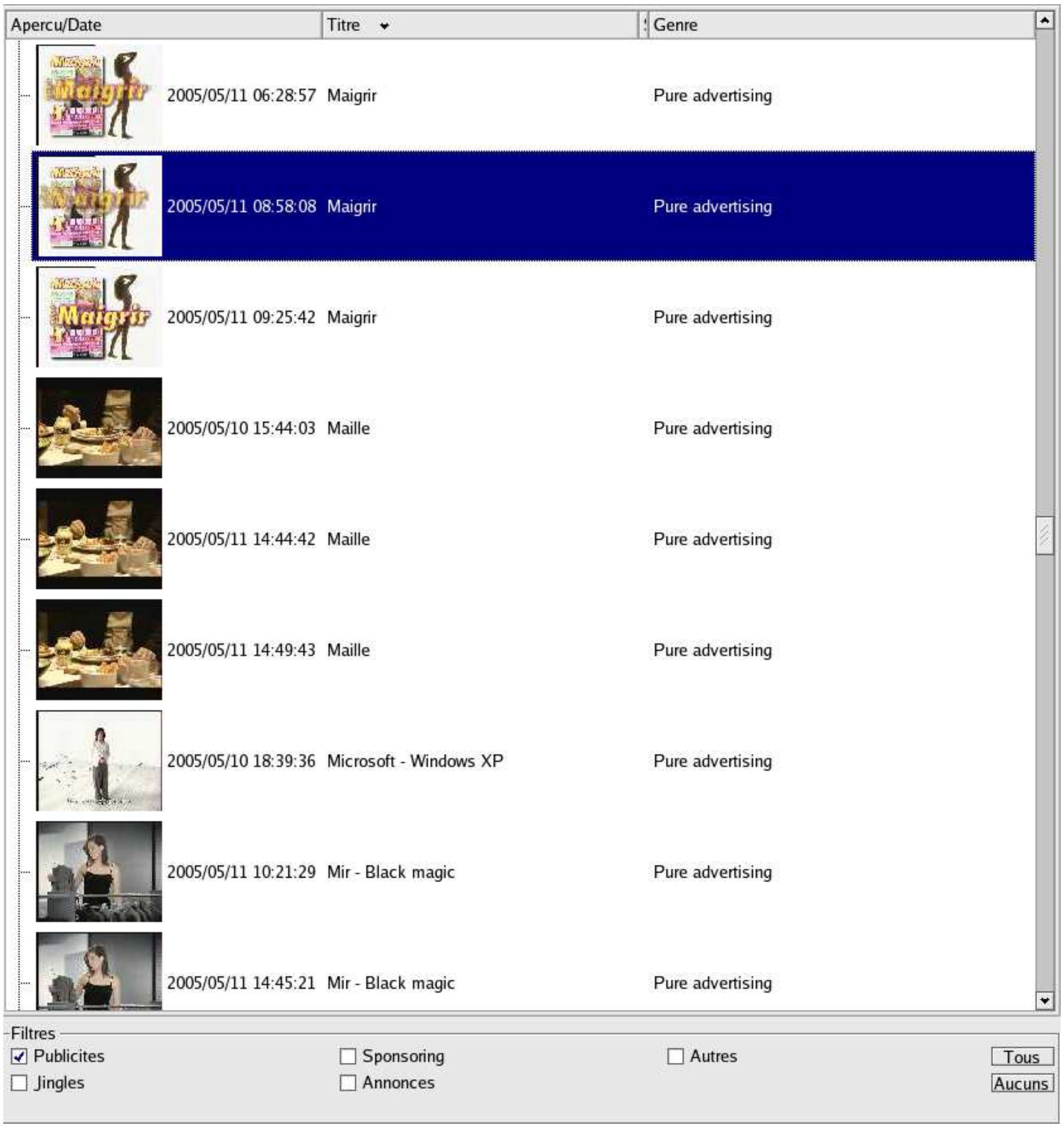

Figure 15: Example of results for commercial monitoring

company spots have been broadcast. Some solutions already exists, for example based on watermarking. Note however that solutions based on watermarking cannot deal with queries

like: I want to monitor my competitor's commercials, or I want to study the broadcast frequency of these specific commercials for a sociological study. Figure 15 shows an example of commercial detection using our system.
The second application is specific to French television. Legal regulation specifies that a commercial break should not exceed 4 minutes, and that in average over one day, a channel should not broadcast more than 6 minutes of commercial per hour [27]. These constraints differ from channel to channel and are supposed to be monitored by a special regulation authority [27]. The monitoring information provided to this authority is done manually or semi-automatically.
Table 8 shows some results obtained on several days. The ground truth in brackets shows

Table 8: Enforcing legal regulations

day	maximum duration (ground truth)	average duration (ground truth)
Legal regulation	4	6
Saturday 05/14/2005	1.4 (4.72)	2.43 (2.7)
Sunday 05/15/2005	4.01 (4.6)	3.34 (4.42)
Monday 05/16/2005	4.04 (4.56)	3.5 (4.2)

that the method still fails to detect all commercials. This is mainly due to the RVD used which does not contain every single commercial that has been broadcast on these three days. One of the key point in future work is to build a meaningful RVD. However, the method can nonetheless raise an alarm when the legal regulations are obviously not respected, as in days 15 and 16.
A third application, already mentionned by Pua et al.[25] is compression. We have found in section 6.1 that television streams are highly redundant with 20 to 25% of the shots being common between two consecutive days, possibly reaching 30% with an appropriate RVD. An intelligent storage scheme for digital television archiving would be to skip the parts of the incoming stream that are already in the archive, thus achieving both 20 to 30 % compression gain. This might also be used to "clean" the archive from its redundancy, making here a clear analogy with traditionnal relational database cleaning. This redundancy might be annoying from an application point of view, see Joly[11].

7 Conclusion

A very fast shot matching strategy to retrieve identical shots from a television broadcast has been presented. The algorithm computes a frame signature which can be mapped to a 64-bit integer, thus allowing exact retrieval and the use of a hash table. We also presented a simple but nonetheless effective strategy to align two candidate shots based on the signature position. Results show the effectiveness of our method on a very large video corpus recorded from french television.
This method could be used in many potential applications: commercials monitoring, tele-

vision archives compression, intelligent digital vcr. Our first motivation was however to detect duplicate sequences for structuring television archives. The next step in our research is therefore to find efficient ways of using this repetition information for video structuring.

References

[1] B. Adams, S. Venketesh, H. Bui, and C. Dorai. A probabilistic framework for extracting narrative act boundaries and semantics in motion pictures. *Multimedia Tools and Applications*, 27:195, 2005.

[2] D. A. Adjeroh, M. C. Lee, and I. King. A distance measure for video sequences. *Comput. Vis. Image Underst.*, 75(1-2):25–45, 1999.

[3] N. Babaguchi, T. Ishida, and K. Morisawa. Scene retrieval with sign sequence matching based on video and audio features. In *ICME*, pages 1107–1110, 2004.

[4] J. Barr, B. Bradley, and B. Hannigan. Using digital watermarks with image signatures to mitigate the threat of the copy attack. In *ICASSP*, 2003.

[5] S.-A. Berrani, L. Amsaleg, and P. Gros. Approximate searches: k-neighbors + precision. In *Proc. of the 12th ACM International Conference on Information and Knowledge Management*, pages 24–31, New Orleans, Louisiana, USA, 2003.

[6] B. Coskun and B. Sankur. Robust video hash extraction. In *EUSIPCO: European Conf. On Signal Processing*, Vienna, 2004.

[7] P. Duygulu, M. yu Chen, and A. Hauptmann. Comparison and combination of two novel commercial detection methods. *Proceedings of the 2004 IEEE International Conference on Multimedia and Expo (ICME 2004)*, june 2004.

[8] D. Freedman and P. Diaconis. On the histogram as a density estimator: L2 theory. *Zeitschrift fur Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57:453–476, 1981.

[9] D. Harman., E. Fox, R. Baeza-Yates, and W. Lee. *Data Structures and Algorithms: Inverted Files.* Prentice Hall, 1992.

[10] A. K. Jain. *Fundamentals of digital image processing.* Prentice hall information and system sciences series, 1989.

[11] A. Joly. *recherche par similarité statistique dans une grande base de signatures locales pour l'identification rapide d'extraits vidéos.* PhD thesis, Université de la Rochelle, 2005.

[12] A. Joly, O. Buisson, and C. Frélicot. Robust content-based video copy identification in a large reference database. In *Proceedings of the International Conference on Image and Video Retrieval*, 2003.

[13] K. Kashino, T. Kurozumi, and H. Murase. A quick search method for audio and video signals based on histogram pruning. *IEEE Transactions on Multimedia*, 5:348–357, 2003.

[14] E. Kijak, G. Gravier, L. Oisel, and P. Gros. Audiovisual integration for sport broadcast structuring. *Multimedia Tools and Applications*, 2004.

[15] W. H. Kim and I. H. Park. Image authentication using relationships of vectors. *STEG'03, Pacific Rim Workshop on Digital Steganography 2003*, July 2003.

[16] Y.-T. Kim and T.-S. Chua. Retrieval of news video using video sequence matching. In *International Multimedia Modelling Conference*, 2005.

[17] A. Lee. http://www.virtualdub.org/.

[18] H. Lejsek, F.-H. Ásmundsson, B. Pór Jónsson, and L. Amsaleg. Efficient and effective image copyright enforcement. In *Journées Bases de Données Avancées (BDA 2005)*, Saint Malo, France, October 2005.

[19] 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.

[20] A. Marzal and E. Vidal. Computation of normalized edit distance and applications. *IEEE Trans. PAMI*, 15(9):926–932, 1993.

[21] D. McKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[22] X. Naturel, G. Gravier, and P. Gros. Fast structuring of large television streams using program guides. In *4th International Workshop on Adaptive Multimedia Retrieval*, Geneva, Switzerland, july 2006.

[23] J. Oostveen, T. Kalker, and J. Haitsma. Feature extraction and a database strategy for video fingerprinting. In *VISUAL '02: Proceedings of the 5th International Conference on Recent Advances in Visual Information Systems*, pages 117–128. Springer-Verlag, 2002.

[24] S. Pfeiffer, R. Lienhart, and W. Effelsberg. Scene determination based on video and audio features. *Multimedia Tools and Applications*, 15:59–81, 2001.

[25] K. M. Pua, J. M. Gauch, S. E. Gauch, and J. Z. Miadowicz. Real time repeated video sequence identification. *Comput. Vis. Image Underst.*, 93(3):310–327, 2004.

[26] J. M. Sánchez, X. Binefa, and J. Vitrià. Shot partitioning based recognition of tv commercials. *Multimedia Tools Appl.*, 18(3):233–247, 2002.

[27] C. supérieur de l'audiovisuel. Publicité, parrainage et téléachat à la télévision et à la radio, http://www.csa.fr/.

[28] Y.-P. Tan, S. R. Kulkarni, and P. J. Ramadge. A framework for measuring video similarity and its application to video query by example. In *Proceedings ICIP 99*, 1999.

[29] B. T. Truong, C. Dorai, and S. Venkatesh. New enhancements to cut, fade, and dissolve detection processes in video segmentation. In *MULTIMEDIA '00: Proceedings of the eighth ACM international conference on Multimedia*, pages 219–227, New York, NY, USA, 2000. ACM Press.

[30] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Processing*, 13, 2004.

[31] J. Yuan, L.-Y. Duan, Q. Tian, and C. Xu. Fast and robust short video clip search using an index structure. In *MIR '04: Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 61–68. ACM Press, 2004.