Mining TV broadcasts 24/7 for recurring video sequences

Ina Döhring, Rainer Lienhart

Angaben zur Veröffentlichung / Publication details:

Döhring, Ina, and Rainer Lienhart. 2010. “Mining TV broadcasts 24/7 for recurring video sequences.” In *Video Search and Mining*, 327–56. Berlin [u.a.]: Springer. https://doi.org/10.1007/978-3-642-12900-1_13.

Mining TV broadcasts 24/7 for recurring video sequences

Ina Döhring and Rainer Lienhart

Abstract Monitoring and analyzing TV broadcasts is an important task in the media as well as the advertising business. An important subtask is the frame-accurate detection of recurring video sequences. Examples of recurring video sequences are commercials, channel advertisements, channel intros, and newscast intros. Most of these different kinds of repeating video clips can automatically be classified by further analyzing their temporal and visual properties. In this work we introduce an algorithm and a real-time system for recognizing recurring video sequences frame-accurately in a highly effective and efficient manner. The algorithm does not require any temporal pre-segmentation by shot detection and can thus, in principle, be applied to any kind of temporal signal. It is frame-accurate, meaning that it exactly identifies with which frame a repeating sequence starts and ends. Thus, the temporal accuracy is 40 milliseconds for PAL and 33 milliseconds for NTSC videos. On a standard PC desktop a 24-hour live-stream can be processed in about 4 hours including the computational expensive video decoding. To achieve this efficiency the algorithm exploits an inverted index for identifying similar frames rapidly. Gradient-based image features are mapped to the index by means of a hash function. The search algorithm consists of two steps: firstly searching for recurring short segments of 1 second duration and secondly assembling these small segments into the set of repeated video clips. In our experiments we investigate the sensitivity of the algorithm concerning all system parameters and apply it to the detection of unknown commercials within 24 and 48 hours of various TV channels. It is shown that the method is an excellent technique for searching for unknown commercials. Currently the system is used 24 hours a day, 7 days a week in various countries to log all commercials broadcast without manual intervention.

Ina Döhring
Multimedia Computing Lab, University of Augsburg, Germany e-mail: doehring@informatik.uni-augsburg.de

Rainer Lienhart
Multimedia Computing Lab, University of Augsburg, Germany e-mail: lienhart@informatik.uni-augsburg.de

1 Introduction

The influence of digital media on our lives is steadily growing. Lots of data, which were formerly presented by audio, photography, or books, is now shared as digital videos. Information from numberless broadcast stations are available 24 hours a day, 7 days a week. These large amounts of data require effective strategies for keeping it accessible. Thus there is an urgent need for effective search and mining algorithms for a variety of applications. Common tasks in video retrieval range from browsing through video collections over copy detection in the World Wide Web to finding special types of sequences in broadcasts such as commercials or news themes.

In this chapter we will focus on the detection of recurring sequences in TV broadcasts such as commercials, music clips, jingles, news stories, and similar content. For performance analysis and evaluation, however, we will only concentrate on the detection of recurring commercials as they have the advantage that we can easily and fast create frame-accurate ground truth annotations with a video wheel; something that is normally not easily possible for other kinds of recurring video sequences.

Regarding commercials as recurring sequences is a highly effective method to search for them. It only needs to rely as little as possible on vague, and/or easily alterable audio-visual characteristics such as increased audio level or cut-frequency. Usually simple duration constraints are enough to distinguish them from other recurring sequences. As shown in Sec. 4 the percentage of repeated commercials within 24 hours in our test videos is about 90%. In other words, if we can detect all recurring commercial clips, we cover 90% of all broadcast advertisements per day. For longer mining periods such as days or weeks, this fraction will increase, because some commercials that are shown only once a day will be repeated within the next days. The detection of unknown commercials can be used, for instance, to create an ad database as the basis for reliable recognition of broadcast advertisements [1].

This chapter is structured as follows: After some comments on the characteristics of the videos we deal with, we will introduce and discuss two types of image features for fingerprinting our sequences in Sec. 2. In Sec. 3 we explain our search algorithm in more detail. Experimental results are discussed in Sec. 4. We investigate the sensitivity of the algorithm concerning all system parameters and apply it to the detection of unknown commercials within 24 and 48 hours of various TV channels. We give survey on related work in Sec. 5 and finally a summary in Sec. 6.

2 Fingerprinting Video Streams

Video streams are temporal sequences of individual images. With proper image features the essence of these video frames can be captured. Hence a video stream can be considered as a temporal sequence of image features. There already exists a very large variety of image features. For instance, they can be derived from the colors and/or the structures in the picture. Additionally, it may be worthwhile to take temporal information into consideration. After explaining in Sec. 2.1 the context in

which we operate with our search algorithm, we introduce in Sec. 2.2 two types of image features – a color-based and a edge-based feature. Both image features are suitable for real-time fingerprinting [10].

2.1 Video Model

There exist different intentions for searching for repetitions in videos. Depending on the actual goal the search methods may vary, for instance, in the requirements on the image features or the quality and precision of the results that have to be achieved.

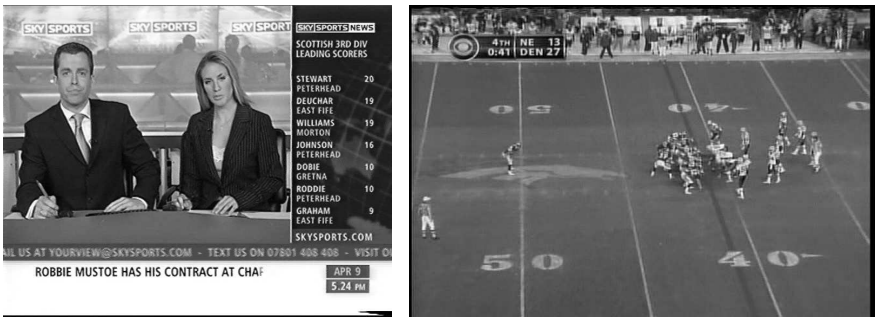
We will focus on the mining of repeated sequences from a single sensor such as a specific TV capture card for building a comprehensive database from TV live-streams. Here, we have the constraint to operate in real-time, which means that we need image features that are fast to compute and a very fast algorithm for detecting recurring sequences. Storage needs are assumed to be moderate, because the recurring sequences in live TV streams are expected to be of moderate length.

We also want to point out that our algorithm on purpose is designed for a data stream coming from a fixed, but arbitrary sensor such a TV capture card from a specific vendor. As every human observes the world through the same two eyes and has to learn everything it can see for these two eyes, our system observes everything through the same kind of video capture cards for the same kind of signals. Any kind of artifact created by the sensor is consistent throughout time in the input data stream. Thus our work does not address the copy detection problem on the Internet or across different video formats and video fidelities. This is a completely different problem domain. Nevertheless our task is also challenging since the focus lies on temporal precision.

As we are interested in assembling a database of all commercials broadcast during our 24/7 monitoring, we have to detect repeated sequences in real-time not only within the stream but also in the database that is built on-the-fly during mining. In this setting it can be an advantage to choose image features which are less robust, but posses a ability to reliably distinguish between different images, since it may lead to a better overall performance as precision is more important than recall.

2.2 Image Features

In this Section we introduce two image features: the *Color Patches Feature* (CPF) and the *Gradient Histogram* (GH). A color patches feature is derived by computing color average values for small parts of the image. Gradient Histograms are based on the edges in an image. Advantages and disadvantages of both types of image features are discussed in length in [10]. In the following we will evaluate quantitative and qualitative properties of these image features. For visualization we take two sample video frames from two different TV channel recordings (see Fig. 1).



(a) (b)

Fig. 1: Sample images from (a) a UK broadcast in PAL norm and (b) a US broadcast in NTSC norm.

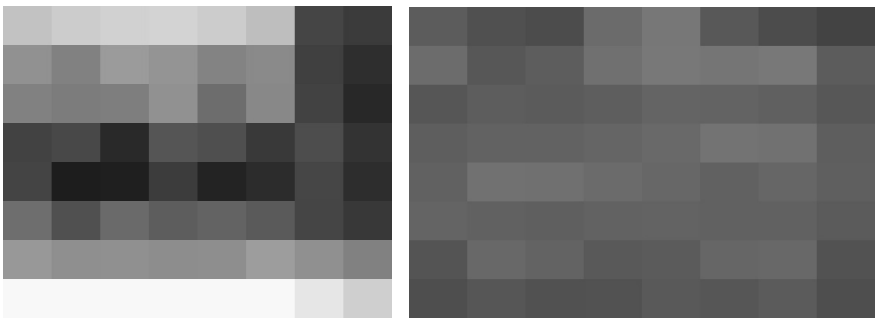
Color Patches Features We choose the color patches feature (CPF) as our color-related image feature. It has been shown that it is robust and outperforms, for instance, Color Coherence Vectors for the task of image matching [10]. Many variants of CPFs are widely used and amongst others described in [17].

CPFs are derived from mean intensities within small subareas of each color channel. The whole image is divided into $N \times M$ rectangular blocks. For each block the intensity of each color component red, green, and blue is averaged:

$$P_{nm}^{C} = \frac{1}{\sum_{(x_1,x_2)\in I_{nm}} 1} \sum_{(x_1,x_2)\in I_{nm}} C(x_1,x_2) \tag{1}$$

with colors $C \in (R,G,B)$ and subareas I_{nm}, $n = 1,\ldots,N$ and $m = 1,\ldots,M$.

Fig. 2 illustrates the effect of the CPF calculation. Each single-colored rectangle represents three mean RGB intensities in the CPF vector. Thus, the size of the CPF vector is $3 \times N \times M$.



(a) (b)

Fig. 2: The color patches features (CPFs) for the two sample images of Fig. 1: (a) for the UK broadcast image and (b) for the US broadcast image.

Gradient Histograms An alternative to color-based features are edge-based features. They evaluate color intensity gradients instead of absolute intensities. Edge-based features similar to our gradient histograms are for instance investigated in [17, 19, 26].

In our work we operate on grayscale images only and use intensity differences to approximate the true gradients:

$$\frac{\partial}{\partial x_1} I(x_1,x_2) \sim [I(x_1+1,x_2) - I(x_1-1,x_2)] \tag{2}$$

$$\frac{\partial}{\partial x_2} I(x_1,x_2) \sim [I(x_1,x_2+1) - I(x_1,x_2-1)] \tag{3}$$

We again divide an image into $N \times M$ subareas I_{nm}. For each subarea we generate a gradient direction histogram of K bins by summing sum up in each bin the gradient magnitudes of all gradients, whose directions fall into the interval of the bin:

$$H^k_{nm} = \frac{1}{\sum_{(x_1,x_2)\in I_{nm}} m_g(x_1,x_2)} \sum_{(x_1,x_2)\in I_{nm}} \mathscr{M}_k(x_1,x_2), \tag{4}$$

with the gradient magnitude m_g, orientation θ_g, and the auxiliary variables $\mathscr{M}_k$ and θ_k defined by

$$m_g = \sqrt{(I(x_1+1,x_2) - I(x_1-1,x_2))^2 + (I(x_1,x_2+1) - I(x_1,x_2-1))^2}, \tag{5}$$

$$\theta_g = \arctan\left(\frac{I(x_1,x_2+1) - I(x_1,x_2-1)}{I(x_1+1,x_2) - I(x_1-1,x_2)}\right). \tag{6}$$

$$\mathscr{M}_k = \begin{cases} m_g(x_1,x_2) & \text{if } \theta_k \le \theta_g(x_1,x_2) < \theta_{k+1}, \\ 0 & \text{else,} \end{cases} \tag{7}$$

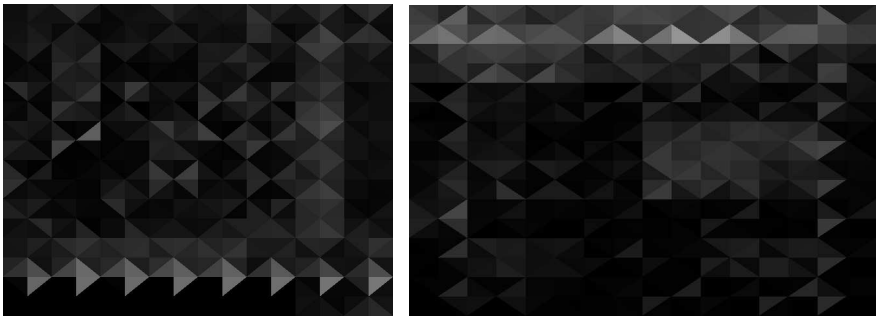
$$\theta_k = (k-1)\,360^\circ/K \tag{8}$$

Figure 3 illustrates GHs. Each sample image has the same partitioning as for the CPFs in Fig. 2. A rectangular subarea is divided into $K = 8$ directional bins to visualize the histogram. The histogram values are encoded by the grayscale intensity: the lighter the gray, the larger the value of that histogram bin. Thus, darker parts in the image mark areas with little structure, whereas rectangles with only a few nearly white segments are typical for strong straight edges (compare to Fig. 1).

Thus a gradient histogram fingerprint consists of $N \times M \times K$ values H^k_{nm}. In our further work we use a compressed feature vector

$$\mathscr{H}^k_{nm} = \min\left(256/L \cdot H^k_{nm}, 255\right), \tag{9}$$

which maps all floating point values to 1-byte integers. For $N = M = K = 8$ $L = 0.02$ is a good choice [10].



(a) (b)

Fig. 3: The gradient histograms (GHs) of the two sample images in Fig. 1: (a) for the UK broadcast image and (b) for the US broadcast image.

Distance Measure We measure the distance between two images I_1 and I_2 with the L_1-Norm of the particular feature vector

$$D^{FV}(I_1,I_2) = \begin{cases} \frac{1}{3NM}\sum_{C\in(R,G,B)}\sum_{n=1}^{N}\sum_{m=1}^{M}\left|P_{nm}^{C}(I_1)-P_{nm}^{C}(I_2)\right|, & FV=CPF, \\ \frac{1}{NMK}\sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{k=1}^{K}\left|\mathscr{H}_{nm}^{k}(I_1)-\mathscr{H}_{nm}^{k}(I_2)\right|, & FV=GH, \end{cases} \tag{10}$$

where FV is the place holder for the feature vector name. The distance between two sequences S_1 and S_2 of length L is given by

$$D_L^{FV}(S_1,S_2) = \frac{1}{L}\sum_{l=1}^{L} D^{FV}(S_1(l),S_2(l)). \tag{11}$$

The distance measures in Eqs. 10 and 11 provide the possibility for defining the equality of two images or image sequences, respectively. We call two images to be equal to each other, if the feature vector distance is less than a threshold Δ_I^{FV}

$$I_1 = I_2 \Leftrightarrow D^{FV}(I_1,I_2) < \Delta_I^{FV}. \tag{12}$$

Accordingly, we define two sequences S_1 and S_2 of length L_1 and L_2 to be equal, if $D_L^{FV}(S_1,S_2)$ is less than a threshold Δ_S^{FV} and their duration difference is less than Δ_L:

$$S_1 = S_2 \Leftrightarrow D_L^{FV}(S_1,S_2) < \Delta_S^{FV}, L = \min(L_1,L_2), |L_1-L_2| < \Delta_L. \tag{13}$$

3 Searching Frame-Accurately

In this section we introduce our algorithm for frame-accurate identification of duplicate sequences in the fingerprints of live video streams.

The whole algorithm is composed of nearly independent parts, which are sequentially processed and which in principal can be replaced by other choices. After having fingerprinted each frame of a video we need a fast and efficient image search method. We use an inverted index to identify similar images rapidly. and a hash function to map an image feature vector to an one-dimensional table index. Details of this step are explained in Sec. 3.1.

On the basis of the inverted image index we search next for short repeating sequences of about one second and call them *clips*. Candidate duplicate clip pairs are built by looking for short sequence pairs with a required minimal percentage of hash-value-identical frames. Details are explained in Sec. 3.2.

Clip pairs are grouped to longer target sequences according to their temporal coherence and are aligned for proper estimation of start and end frames. This step is explained in Sec. 3.3.

The application of filters dedicated to a search of special interest like commercial mining is discussed in Sec. 3.4.

3.1 Inverted Index and Locality Sensitive Hashing

An inverted index is an efficient method for fast search through large databases. In the text domain, for instance, an inverted index contains a list of all words occurring in a text corpus. For each word, a list of all its positions in the text is provided. Thus, a single look-up in the index is sufficient to retrieve all occurrences. No expensive sequential or tree-based search through the text corpus is required. The only expensive step is the creation of the index. This, however, must only be done once.

A difference between image and text retrieval arises from the high-dimensional and often continuous feature space in image representations. There are several approaches to construct inverted indices for such feature vectors. Hampapur and Bolle (2001) used an inverted index for each component of the feature vector [18]. Shivadas and Gauch (2007) mapped the complete feature vector by means of a hash function to a single scalar value [30]. Hash functions are designed for deterministic but non-injective mapping a sparse representation of a large feature space to an index space whose size is in the order of the data's dense representation. The mapping of different values to the same hash value is called a collision and can be minimized by good mixing properties of the hash function. The modulo function is often a proper choice [24].

The disadvantage of hash functions is that due to the mixing characteristics, no distance measure can be directly deduced from the index values. Neighbored indices do not necessarily belong to features which are close together. One possible answer to this problem is *Locality Sensitive Hashing (LSH)*. Here hash functions are used for which the probability of mapping similar features to the same index is much higher than the probability for mapping more distant features to the same index [15, 23].

Color Patches Features For the CPFs we evaluate the inverted index on the basis of average image intensities C_I, $C \in (R,G,B)$, which can be easily obtained:

$$C_I = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} P_{nm}^{C} \qquad \text{with} \qquad C \in (R,G,B). \tag{14}$$

Taking the image averages provides locality sensitivity. The distance in the reduced feature space is always less than or equal to the original distance (see Appendix for a prove). Small variations are handled in the second step: we take the first b bits of each of the averaged values to generate a scalar $3b$-bits integer index value:

$$h^{C}(I) = C_I \div 2^{(8-b)}, \quad C \in (R,G,B), \quad b \in \{0,...,8\}, \tag{15}$$

$$h^{CPF}(I) = 2^{2b} h^{R}(I) + 2^{b} h^{G}(I) + h^{B}(I). \tag{16}$$

In this way images with close color average intensities are grouped into the same bins [6]. The bin size $S_{h^{CPF}}$ and the index range $R_{h^{CPF}}$ depend on the choice of b:

$$S_{h^{CPF}} = (2^{8-b})^3, \quad R_{h^{CPF}} = (2^{b})^3, \quad b \in \{0,...,8\}. \tag{17}$$

Because of the limited index range $R_{h^{CPF}}$ we do not need any further sample space reduction for our index evaluation.

Gradient Histograms For GHs we use a three step hashing algorithm to achieve locality sensitivity in our index. We first reduce the feature vector size:

$$\mathscr{H}^{k} = \sum_{n=1}^{N} \sum_{m=1}^{M} \mathscr{H}_{nm}^{k}. \tag{18}$$

The image related gradient distribution $\mathscr{H}^{k}$ is robust to small changes as all feature vectors are mapped to vectors with equal or less distance (see Appendix). The size of the $\mathscr{H}^{k}$ is $8+P$ bits, if P is the smallest integer with $NM \leq 2^{P}$.

In the second step we evaluate a first intermediate hash value for every component $\mathscr{H}^{k}$ by taking the first b bits of every single value $\mathscr{H}^{k}$:

$$h^{1}(I) = \sum_{k=0}^{K-1} \left(2^{b}\right)^{k} h_{k}^{1}(I), \quad \text{with} \quad h_{k}^{1}(I) = \mathscr{H}^{k} \div 2^{(8+P-b)}. \tag{19}$$

The number of possible values $R_{h^1}(b,K)$ gives the range of this first hash value

$$R_{h^1}(b,K) = \left(2^{b}\right)^{K}, \quad b \in \{0,...,8+P\}. \tag{20}$$

In dependence on the values of K and b the index range R_{h^1} may be quite large, because we deal with typical sizes of $K = N = M = 8$ [10]. Therefore we apply a modulo function with index range R_{h^2} to the first hash value $h^{1}(I)$

$$h^{2}(I) = h^{1}(I) \mod R_{h^2}. \tag{21}$$

We evaluate Eq. 21 in an iterative way with the Horner scheme:

$$h_1^2(I) = h_1^1(I) \mod R_{h^2}, \tag{22}$$

$$h_k^2(I) = \left(2^b \cdot h_{k-1}^2(I) + h_k^1(I)\right) \mod R_{h^2} \quad \text{for} \quad k = 2,\ldots,K,$$

$$h^{GH}(I) = h_K^2(I). \tag{23}$$

For our calculations we use an index range $R_{h^2} = 100,003$, which meets the criteria for choosing a good table size [24].

The first two steps preserve the locality sensitivity, whereas in the third step, the evaluation of h^2, we use a classical hash function with good mixing properties.

Distance Measure The hash value representation is the basis for the similarity definition. We call two Images I_1 and I_2 to be *similar*, if their hash values are equal

$$I_1 \sim I_2 \Leftrightarrow h^{FV}(I_1) = h^{FV}(I_2), \quad FV \in (CPF, GH), \tag{24}$$

and we call two image sequences S and T of length L_S and L_T, $L_S \leq L_T$ without loss of generality, to be similar, if a certain amount $\alpha \in (0,1)$ of images is similar

$$S \sim T \Leftrightarrow \sum_{i=1}^{L_S} \theta(S(i)) > \alpha L_S, \tag{25}$$

with

$$\theta(S(i)) = \begin{cases} 1, & \text{if} \quad \exists j \in (1, L_T), \quad S(i) \sim T(j), \\ 0, & \text{else.} \end{cases} \tag{26}$$

Thus we distinguish between *identical* (Eq. 13) and *similar* (Eq. 24) images. A small image feature distance (Eq. 10) describes the visual identity of two images. For similar images there is only a certain probability that they are actually identical.

In our search algorithm we first search for similar images and clips, which can be done very fast with the index table, and then refining our results by evaluating the more time consuming image feature distance to identify actually duplicate clips.

Implementation There are several approaches to implement an inverted index for live-stream applications. One possibility to handle the index search is to deal with a dynamic index table, which is updated after processing each frame. In this case you may execute a table search just in time. Beside the – for our application – unnecessary additional computational overhead, an even more serious issue is that all hashing functions are designed for a certain frame number range of video frames. For instance, the index range $R_{h^2} = 100,003$ for the GH was chosen with a 2 hour mining period in mind, i.e., for 180,000 hash values for PAl videos. Without causing to many collisions the search period cannot just be increased to maybe 24 hours or longer.

As we want to process the live-stream in real-time, while it is acceptable to obtain results with a certain delay, we split the video stream into suitable pieces of about two hours. The resulting fingerprint size of such segments fits into the system

memory in a comfortable way. For each slice we build a separate inverted index and store it on the hard drive. Every two hours we invoke a sequence search that takes the latest two hours of fingerprints as the input sequence, searches through it, and then continues through the previous two hours of fingerprints one after the other until the period we want to search through has been processed (e.g., 24 hours).

3.2 Clip Search

A live video stream appears as an endless sequence of images without any direct hints about embedded repeating sequences. Some information about the underlying structure of the video stream could be gained by shot segmentation. However, in the digital age we can find beside classical hard cuts many hardly to detect complex transitions such as dissolves, wipes, and morphings. The correct detection of all transitions would require separate reliable algorithms for each of them. To avoid errors from cut detection and save the detection time we operate on the raw video stream: we pick random pieces, called clips, out of the stream and search for repetitions of these clips. The duration of a clip may be arbitrarily chosen, but it should be much shorter than the lengths of our target sequences to ensure that there exist clips that are completely contained within each repeated sequence. The smallest unit to search for is a single image. However, single images are often not sufficiently distinctive, and therefore we will settle on one-second clips.

The clip search approach is adopted from DNA sequencing. Complete DNA strands are far too long to be sequenced in one single run. Therefore, they are broken up into much smaller pieces. This can be done in a deterministic or random way (shotgun method). In the latter case all fragments have to be aligned after analysis. There are a variety of sequence alignment methods known in bioinformatics [20].

We can use some pieces of information from the stream to make a proper choice for the processed clips and only start a new clip search at an image, if we find similar images in our inverted index. We scan a video frame by frame, calculate the hash index of each frame and retrieve all similar frames by a look-up in the inverted index. We reject matches, which are temporally very close to our query frame. A minimal gap of 2000 frames is required to avoid detection within the same scene. Furthermore we neglect frames belonging to hash indices with a very large number of entries, i. e. we ignore non-distinctive frames such as black frames or parts of long self-similar sequences.

Figure 4 shows two clips of 25 frames (1 second in PAL). Similar frames with identical hash values are marked. It is possible, that there is no one-to-one mapping between similar images, as visually similar images can be mapped to the same index.

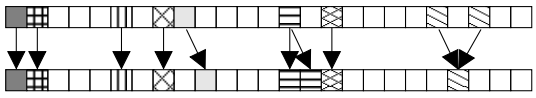


Fig. 4 Two clips with frames matched by hash values.

If we find more than 20% similar frames within clips s_1 and s_2, we consider $s_1 \sim s_2$ with $\alpha = 0.2$ (Eq. 25). For all duplicate clip candidates $\mathscr{C}(s_1, s_2)$ we calculate the distance $D_L^{FV}(s_1, s_2)$ (Eq. 11) between them. All unequal candidates (Eq. 13) are discarded, all equal ones are further processed.

3.3 Frame-Accurate Repeated Sequences Search

The next task in our algorithm is to build the repeated sequences from the identified clip pairs. At first we group the pairs, which are temporally coherent together. After aligning these groups of clips we can estimate the start and end of all found repeated sequences.

We identify clip pairs $\mathscr{C}$ belonging to the same recurring sequence by looking at their times of occurrence. Two clip pairs are assumed to belong to the same recurring video sequence, if both clips of a pair are closer than 150 frames to the corresponding clips in the other clip pair. This step results in a number of sequences, which are pair-wise identical or similar in parts. Each pair of such sequences is called a duplicate $\mathscr{D}$. Each duplicate is described by two lists S_1 and S_2 representing the corresponding sequences:

$$\mathscr{D} = (S_1, S_2)\,. \tag{27}$$

Each list S_i contains the start frame numbers of the clips building the duplicate:

$$S_i = \left(\text{frame}_1^i, \ldots, \text{frame}_n^i\right), \tag{28}$$

with n denoting the number of clips forming the duplicate. Note that frame_i^1 and frame_i^2 are the start frame numbers of the two clips of a clip pair i. According to our search strategy the frames in S_1 and S_2 are not necessarily in the same temporal order, especially for long still scenes.

Figure 5 illustrates the state in our search algorithm at this point. We have identified a number of small pieces of our target sequence. However, we still miss precise informations about the boundaries.

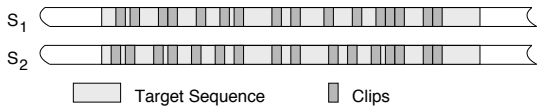


Fig. 5 Recurring video sequence with the clip pairs that were found.

To reduce the input for the next steps it is possible to do a coarse filtering at this stage. For instance we discard sequences, which do not meet the required minimum number of clips. Content related filtering is discussed in Sec. 3.4.

As it can be seen in Fig. 5 both sequences S_1 and S_2 due to their construction may be shifted to each other, i. e., the offsets o_i with

$$o_i = \text{frame}_i^2 - \text{frame}_i^1 \qquad i = 1, \ldots, n, \tag{29}$$

between corresponding clips are not the same for all n. Hence, we need a method for estimating the real offset between the repetitions of the target sequence.

One simple technique is the iterative alignment during the detection of start and end frames. We find the start and end frame of a found sequence by going backward at the start and forward at the end frame by frame as long as the feature vector distance between the two associated frames is smaller than a threshold. If the image distance signals different frames, we have recognized the end or the start, respectively. For sequence alignment we do not stop this procedure, once we have found different frames, but continue the search with a modified offset derived from the original offset by adding a relative offset. We set initially this relative offset to +1 frame. If the distance between two frames under this modified offset is smaller than our threshold, we test the next two frames with the same modified offset until we find different frames again. In the case that the current relative offset has not resulted in frame equality, we try the same relative offset in the opposite direction (i.e., we us the negative offset), and after that we increment the relative offset by one. When a maximal allowed relative offset is reached, the procedure stops and the last found identical frames are rated as start and end, respectively.

We can accelerate this operation by estimating at first the most likely offset o_0 from all offsets o_i. If $F(o_i)$ is the distribution function of the start frame differences of all clip pairs of a sequence candidate, we take the most frequent offset as the first guess for the alignment of both sequences, i.e.,

$$F(o_0) \geq F(o_i) \quad \forall i \in (1,n). \tag{30}$$

After aligning the duplicate sequences with this offset o_0, we can determine the start and end frame with the method described above. At this time, we have frame-accurately determined two occurrences of the same sequence; we know their beginnig and temporal length. To validate results, we can at this stage evaluate the feature-based distance function for the complete sequence and discard eventually false matches.

In a last step we compare the pairwise matched sequences against those from the other duplicates to group frequently occurring sequences together. All detected repeated sequences are compared with the database and added if novel.

3.4 Content-Related Filtering

As already mentioned in Sec. 3.3 we can control the result by filtering the clips found in dependence on the content we search for. Typically not all kinds of recurring sequences are of interest all times. Special kinds of sequences such as commercials can be retrieved by taking their characteristic properties into account and eliminating all non-complying sequences. A useful property is sequence length. For instance, you may distinguish between channel logos (just a few seconds long), commercials (10 to 60 seconds long), and music videos (several minutes long). At this step you may

include all features you know are characteristic for the kind of repeating sequences you are looking for. For ads this can be a higher cut rate, black frames, increased audio volume, and others attributes as discussed by Lienhart et al. in [25].

As we want as much as possible to be independent of properties that can be changed by the advertising industry, we restrict our commercial filter to sequence length.

4 Experimental Results

In our experiments we investigate the sensitivity of the algorithm concerning the most relevant system parameters. We test the algorithm by detecting commercials in 24 and 48 hours, respectively, of two manually labeled TV channels. Furthermore we explore the choice of our image features and compare the performance of CPFs against GHs. At last we apply the system to a variety of TV channels with quite different characteristics of advertisement representation.

4.1 Parameters and Numbers of Relevance

For our quantitative experiments we use two 48-hour long video sequences recorded from two different British television channels: Chart TV (a music channel) and Sky Sports News (a sports channel). Both videos are downscaled to half PAL resolution (360×288 pixels) at 25 frames per second. For these two test videos we determined the locations of all occurring commercials manually as our reference truth.

Although our proposed search algorithm mines video streams for all kinds of recurring video sequences, of which commercials are just one example, we focus in our experimental evaluation on the detection of repeating commercials for the following practical reason: It is quite easy and fast to determine manually and unambiguously all recurring commercials in a test video.

Commonly performance of search algorithms is measured by recall and precision: recall R measures the percentage of detected relevant sequences, while precision P reports the percentage of relevant sequences in the search result:

$$R = \frac{\text{number of found relevant sequences}}{\text{number of all relevant sequences}}, \tag{31}$$

$$P = \frac{\text{number of found relevant sequences}}{\text{number of all found sequences}}. \tag{32}$$

Depending on the particular motive of the search, it makes sense to specify slightly modified performance values. Recall and precision from Eqs. 31 and 32 describe the algorithm's performance concerning all repeated sequences in the video stream. As mentioned above we will concentrate on the detection of commercials.

We will use the superscript C to indicate this fact. In particular, we will consider the following values, which are either of more theoretical or more practical interest. The most important values for theoretical discussion are R_M^C and P_M^C, which stand for the detection of **M**ultiple occurring commercials, because these values describe the quality of the algorithm. Of more practical interest are the values R_{MD}^C and P_{MD}^C. This pair specifies the detection of **M**ultiple **D**ifferent occurring commercials. Here, we count, if a recurring commercial is detected. It is unimportant whether all repetitions are found. These values are of interest, if we want to build a database of commercials. Here it is sufficient to detect one repetition. Last, but not least, R^C and P^C stands for the detection of all commercials, and R_D^C and P_D^C for all **D**ifferent commercials. The last four values include commercials which are not repeated and thus cannot detected by the algorithm. They give the possibility to estimate, how successful a repeated sequence search is regarding commercial detection.

It is important to note that our precision values do not correctly reflect the performance of the algorithm. The result list of the search for repeating video sequences will (absolutely correctly) contain plenty of repeated sequences, which are not commercials. Therefore, the reported precision values concerning repeating commercials are quite low, since every recurring sequence that is not a commercial will count as a false alarm. We will try to mitigate this issue by applying a pre-filter to the raw result list that discards all repeating video sequences whose durations divert from the characteristic durations of commercials. In practice, it is our observation that the true precision values of our system are very close to 1 for repeating video sequences. We hardly remember having ever seen a false alarm for repeating video sequences.

A commercial spot is *found*, if the start and end frame differ no more than 5 frames from the exact position. This tolerance is introduced since a commercial spot is sometimes slightly shortened at the boundaries resulting in repetitions of the same spot with slightly different durations. Additionally, commercials are sometimes separated by monochromatic, mostly black frames, which are, if present at the boundaries of all repetitions, strongly spoken part of the repeated sequence, too, but not of the commercial manually marked in the basic truth.

For both 48-hours sequences we manually labeled all commercials occurring within the first 24 hours. In addition, we also labeled the second half of the Chart TV video sequence. This gives us the possibility to estimate the benefit of a 48-hours over a 24-hours search.

Table 1 reports key numbers about our test video sequences using the convention that superscript C indicates that all numbers refer to commercials only: N^C specifies the overall number of occurring commercials in the test videos. For each test video N^C can be split into the number of ads N_M^C which are repeated within the overall video sequence and N_S^C which are not repeating (subscript S stands for *single* occurrence). In other words: $N^C = N_M^C + N_S^C$. The overall number of different commercials is denoted by N_D^C, of which only N_{MD}^C are repeated in the overall video sequence. Thus, the following relation holds: $N_D^C = N_{MD}^C + N_S^D$. Thus, the subscript D signifies that a recurring video sequence is counted only once.

Table 1: Ground truth of our test videos: N^C - number of all occurring commercials, N_M^C - number of all repeatedly occurring commercials, N_S^C - number of all singly occurring commercials, N_D^C - number of different commercials, N_{MD}^C - number of repeatedly occurring different commercials, t^C - time covered by all occurring commercials, t_M^C - time covered by all repeatedly occurring commercials, and t_S^C - time covered by all singly occurring commercials.

	Chart TV 24h	Chart TV 48h	Sky Sports News 24h
N^C	486	997	737
N_M^C	428	928	650
N_S^C	58	69	87
N_D^C	164	212	245
N_{MD}^C	106	143	158
N_M^C/N^C	88.1%	93.1%	88.2%
N_{MD}^C/N_D^C	64.6%	67.5%	64.5%
t^C / video length	13.2%	13.5%	20.0%
t_M^C / video length	11.6%	12.5%	17.4%
t_S^C / video length	1.6%	1.0%	2.6%

As we can see from Table 1, around two thirds of all broadcast commercials appear more than once a day, covering around 88% airtime of all occurring spots. Additionally, the time fraction, which is allocated to TV ads, is shown. In Chart TV about 13% of airtime is devoted to commercials, whereas it is 20% in Sky Sports News. Only between 1% and 3% of the overall airtime is devoted to non-repeating spots. These are the sequences that cannot be found by even a perfect search algorithm for repeating commercials.

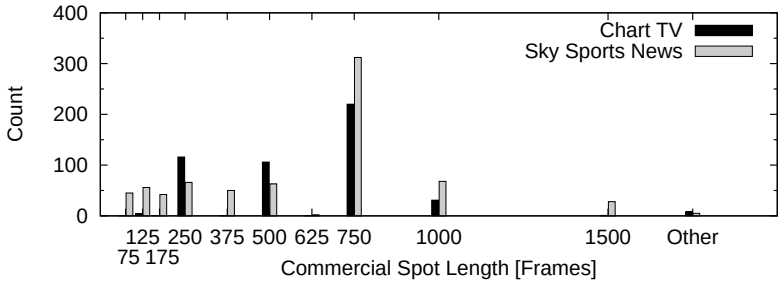


Fig. 6: Duration distribution of TV commercials.

Figure 6 depicts the duration distribution of all occurring spots. In Chart TV all commercials – with a few exceptions – are multiples of 10 seconds, whereas in Sky Sports News we find a greater variance in spot durations with a tendency to shorter spots. This distribution encourages the idea of duration filtering to improve the results concerning commercials.

Most of our test cases concerning system parameters use GHs as image features with $N = M = K = 8$. We build-up the inverted index based on hashing as explained in Sec. 3.1.

Content Related Filtering Our first experiment concerns the last step in our search algorithm - the content related filtering. As discussed above, our proposed algorithm mines videos for all kinds of recurring video sequences, of which commercials are just one example. While recall is not affected by focusing on commercials, it renders the precision value useless. We apply a "commercial filter" to the raw result list in order to give the precision values a meaning: With what precision can TV commercials be found with a repeating video clip search algorithm.

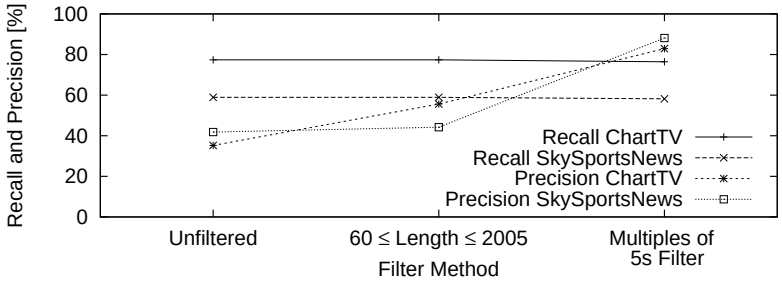


Fig. 7: Content related filter methods.

Figure 7 shows recall and precision in dependence on the filter method. We compare the unfiltered output with a simple length filter, which discards very short and very long sequences, and a more sophisticated filter, which focuses on typical durations of TV commercials. The *simple duration filter* keeps all repeating sequences with a length between 60 and 2005 frames. The more *sophistic filter* keeps all repeating sequences with durations representing multiples of 125 frames or 5 seconds (with a tolerance of $\pm$ 5 frames). Additionally, we include sequences of 75 and 175 frames length. These values are derived from the duration distribution in Fig. 6.

As shown in Fig. 7 the "typical length filter" improves the precision significantly compared to the unfiltered case. There is only a minor decrease in recall due to a small amount of commercials with non-typical durations (Fig. 6). In the following experiments we will always filter the raw result lists with the "typical length filter", i.e., with the "multiples of 5s filter".

Alignment Method We evaluate the two alignment methods introduced in Sec. 3: (1) The frame-by-frame iteratively estimated offset (SIMPLE) and (2) its variant where the initial offset is the most frequent offset (see Eq. 30) occurring between the clip pairs of a duplicate (OFFSET).

As we can see in Fig. 8 performance values for both methods show only small differences with a slight advantage of the OFFSET method . We tested both methods for several maximum relative offset values, at which the algorithm stops. It can be

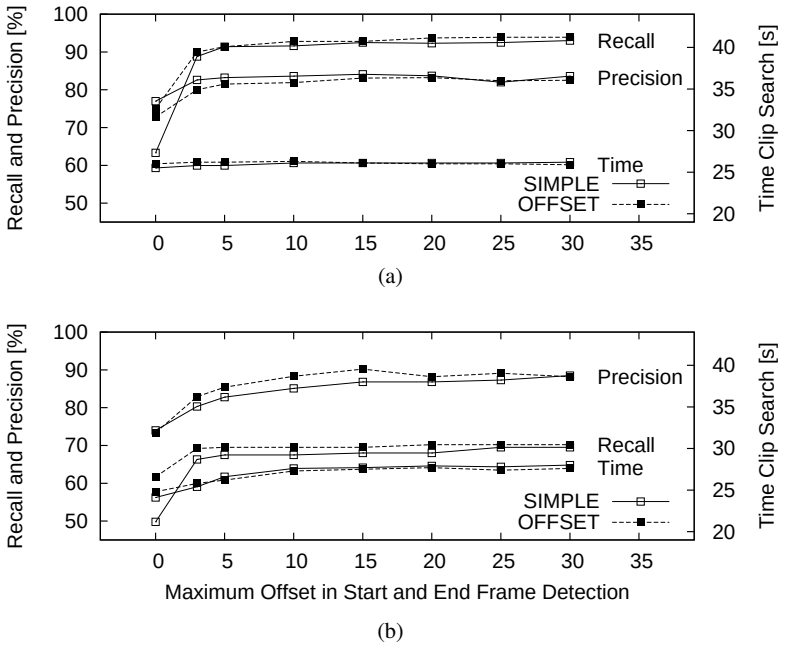


Fig. 8: Performance of the two alignment methods for (a) Chart TV and (b) Sky Sports News: SIMPLE - offset estimation by iteratively varying start and end frames, OFFSET - the offset is initialized with the most frequent start difference between corresponding clips before SIMPLE is applied. Recall, precision, and execution times for repeated sequence search are plotted against the maximum allowed offset.

seen that the performance only degrades for maximum relative offset values of two or less frames. We can reason that our clip pairs in most cases are already aligned up to two frames. However the impact of the maximum allowed offset on execution time is quite small, so we can try larger values to improve the results further.

All subsequent experiments are carried out with the OFFSET alignment method of frame offset and variable start and end frame detection with a maximum relative offset of 20 frames.

Clip Length The following three test cases concern clip search. At first we investigate the impact of the length of the short segments. Figure 9 shows the recall and precision values for both test videos as well as the execution times for clip search and the whole sequence identification. Execution times are only depicted for the Chart TV video in order to report the order of magnitude, since it is the same for both videos.

We can recognize that by and large recall decreases with an increase in the length of the clips, whereas the precision increases. In principal, the shorter the duration of the clips, the better the alignment that can be achieved due to the finer granularity of the samples. The disadvantage of shorter durations is the higher hit rate

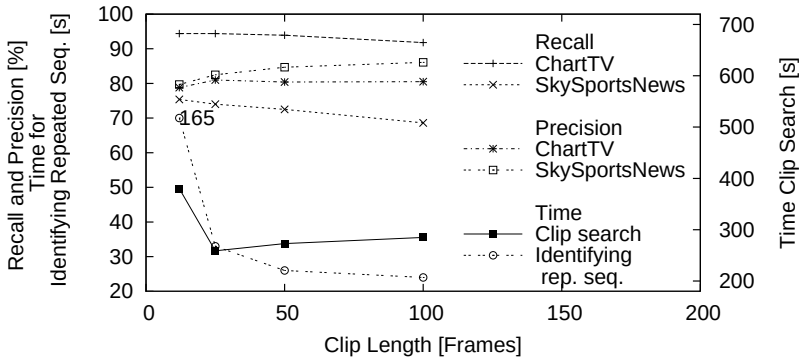


Fig. 9: Performance and execution times in dependence on the duration of the clips.

in non-commercial recurring sequences, which in turn affects precision negatively and leads to an increase in execution times. Especially the time for identifying long repeated sequences is significantly higher for short clips. The increase is mostly caused by the chosen alignment method, with other methods the difference was not that high [11]. This behavior indicates a worse alignment for clips of half a second ($\tilde{1}2$ frames in PAL), the smallest length of clips we have investigated.

All in all a clip length between 25 and 50 frames (corresponds to 1 to 2 second-segments) seems to be an appropriate choice. In our further work we use clips of 25 frames in length.

Minimum Fraction of Matched Frames In this section we discuss the parameter α which determines when two clips are regarded as similar (Eq. 25).

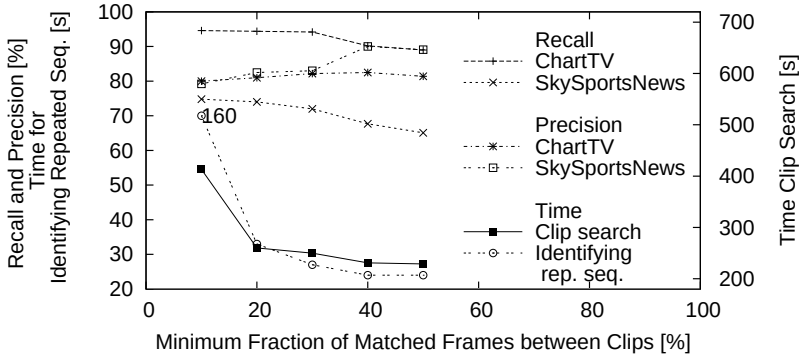


Fig. 10: Performance and execution times in dependence on the minimum required fraction of matched frames.

Fig. 10 plots performance values and execution times against different threshold values α for clip similarity. It is not surprising that recall decreases for higher

thresholds, because more segments are missed. Nevertheless, there is a range for smaller threshold values with little impact. The precision values reveal no significant dependence on the test parameter, being stable over a wide range.

Lower threshold values lead to a higher number of falsely detected clips. Most of them are discarded by the image feature based distance measure. Therefore, there is mainly an influence on evaluation time. The more visual distances must be computed, the longer the execution times. This is clearly revealed in Figure 10 by the rapid decline in execution time for clip search for match ratios larger than or equal to 20%.

As we find the range between 20% and 30% quite stable regarding performance as well as execution time, we choose a threshold of 20% (i.e., $\alpha = 0.2$) of matched frames for the further investigations.

Maximum Number of Entries in Hash Table This test case concerns the search for similar frames in the inverted index. As explained in Sec. 3.1 we find similar frames by looking up the list of frame numbers with the same hash value in the inverted index table. In practice, however, all hash values those frame number list exceeds an upper limit of entries must be disregarded. A very large number of entries can either be caused by too many collisions of the hash function or result from unspecific (generic) images such as black frames. Thus, this knockout criterion is introduced to keep evaluation times low for video streams with long self-similar or unspecific image sequences.

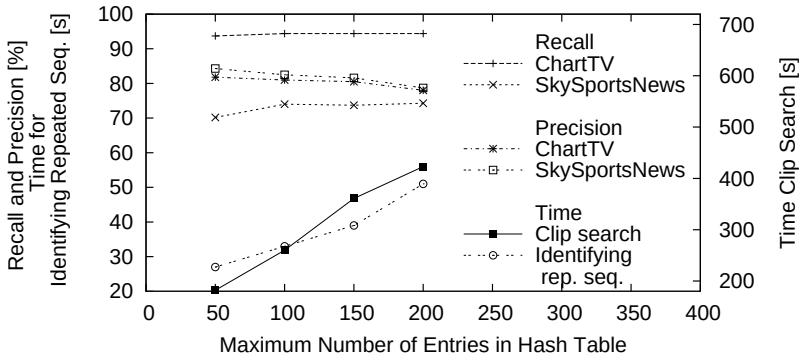


Fig. 11: Performance and execution times in dependence on the upper limit of entries per index in the hash table. All values refer to 2 hour slices.

As revealed in Figure 11, the influence of this parameter is as expected: Recall can be increased, if we expand our image lists, but simultaneously precision decreases slightly. However, we find this parameter to be most relevant for execution time. In fact, we introduced this limit after watching a tennis match that dramatically slowed down the whole system. The parameter keeps execution time nearly constant, even if there are long periods of very similar images in the stream. Keeping

in mind that commercials are short and in general of dynamic content, this parameter is not a restriction.

According to our tests a maximum number of 100 entries per hash value in order to take corresponding frames into account is a good choice for our test configuration of 2 hour slices, i. e. per 180,000 frames. Clearly, this value highly depends on the chosen image features, the applied hash function as well as the duration of the video that is mined.

Minimum Length of Duplicates This test case applies to the identification of repeated sequences only. Requiring a minimum sequences duration S_i when forming a duplicate $\mathscr{D}$ is one possible mechanism for coarse filtering as described in Sec. 3.4. In fact we require a minimum count n of clips as well as a minimum length $(\text{frame}_n^i - \text{frame}_1^i)$ before creating a duplicate as a possible candidate for a repeated clip. In this way very short and sporadic similar clips can be discarded.

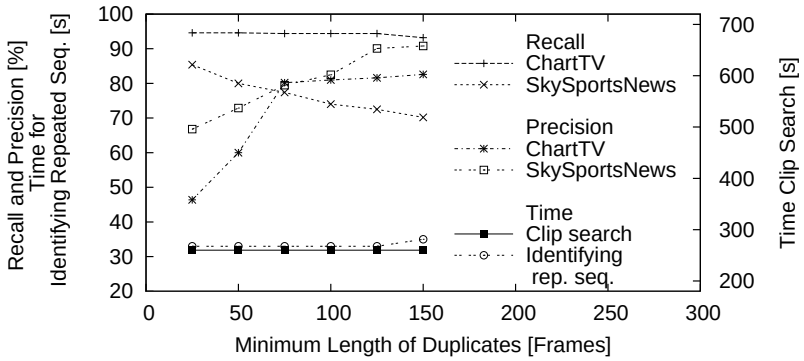


Fig. 12: Performance and execution times in dependence on the minimum length of sequence duplicates.

In Figure 12 performance values and execution times for different required sequence lengths are shown. During these experiments we scaled the required minimum number n of clip proportionally.

The recall of Chart TV is nearly constant until the minimum required length exceeds the smallest occurring commercials with a length of 125 frames (see Fig. 6). The precision is lower for small values, but keeps nearly constant for lengths greater than 80 frames. For TV content with such clear commercial characteristics as shown by Chart TV this filter is a good choice for reducing false matches. For Sky Sports News the situation is more complicated. There are many short ads of 75 frames only as well as more recurring non-commercial clips with typical ad durations than in Chart TV. Thus, we find that recall is negatively influenced by increased duration thresholds, whereas the precision can be enhanced significantly. The execution times for clip search is not influenced, because the filter is applied afterwards; the time for identifying repeated sequences is nearly independent, too. For video content like

Chart TV a minimum length of 100 frames is an appropriate value, whereas in Sky Sports News this threshold discards the very short commercials.

4.2 Detection Performance

After our sensitivity analysis in the various parameters we finally summarize the performance of our system in detecting unknown commercials by means of their repetitions over one day. Additionally, we discuss the improvements that can be achieved by searching for repeated clips throughout two broadcast days. We use the parameters, which led to the best results during our tests: we set the clip length to 25 frames, require 20% of matched frames between two clips to be considered similar, take only frames into account which belong to hash values with less than 100 entries, and require a minimum length of a 100 frames for sequences assembled from clips. We initialized the frame-by-frame search for the estimation of start and end frames with the most frequent offset from the clip pair set of the duplicate, and filter the result list of repeated clips by the 'multiple of 5s' filter that accounts for the typical ad durations.

Table 2: Recall, precision and execution times for our test videos: R_M^C and P_M^C - recall and precision of all repeatedly occurring commercials, R_{MD}^C and P_{MD}^C - recall and precision of repeatedly occurring different commercials, R^C - recall of all occurring commercials, R_D^C - recall of different commercials.

	Chart TV 24h	Chart TV 48h	Sky Sports News 24h
R_M^C	94.4%	95.8%	74.0%
R_{MD}^C	93.4%	93.7%	79.1%
R^C	83.1%	89.2%	65.3%
R_D^C	63.4%	67.5%	51.4%
P_M^C	81.0%	78.6%	82.5%
P_{MD}^C	82.0%	70.9%	80.6%
Search Time Clips [s]	260	1362	207
Search Time Rep. Seq. [s]	33	491	32

Table 2 lists the different performance values discussed above for our two 24 hours test videos as well as for the 48 hours search through Chart TV.

Concentrating on the performance of finding repeated sequences recall values R_M^C and R_{MD}^C are of interest. R_M^C is the rate for detecting all recurring commercials, while R_{MD}^C concerns all recurring different commercials. The relationship between both values depends on the number of repetitions of each spot. The detection rate for Chart TV is – independent of the video length – better than for Sky Sports News. The main reasons are already discussed above and are mainly due to the occurrence of shorter commercials in Sky Sports News.

With regard to a commercial detection system R^C and R_D^C are of further interest. R^C captures the recall with respect to the detection of all occurring commercials N^C and R_D^C to all occurring different commercials N_D^C. Repetition is not required for these recall values. Due to the high rate of repeated commercials (see Table 1), values are strongly correlated to R_M^C and R_{MD}^C, resulting in a reasonable recall concerning all commercials for Chart TV. Results for Sky Sports News could be enhanced by channel specific parameter settings.

The precision value P_M^C relates to all detected repeating video sequences, while P_{MD}^C focus on repeated different sequences. Again, the relation between both values depends on the number of repetitions of detected commercials and detected non-commercial sequences. If commercials are more often repeated, P_M^C is higher. Note that precision drops for a longer search time due to finding more repeated non-commercial sequences and a higher probability for coalescing two commercials to one if both commercials are repeated in the same temporal order.

Run times in Tab. 2 belong to search runs through 24 and 48 hours videos, respectively. Here, the search time does not scale linearly, because we compare n_T 2 hours slices with each other, resulting in $n_T(n_T+1)/2$ operations. Therefore, in our live detection system the search is carried out each 2 hours, and execution time scales linearly with the number of past slices we search through, because in this case only the actual slice is compared with those of the past resulting in n_T operations.

4.3 Color Patches Features and Gradient Histograms

All experiments in the previous two sections have been carried out with Gradient Histograms as image feature. In this section we investigate the influence of the chosen image feature on the overall performance. Thus we compare the results with GHs against the results with CPFs.

Figure 13 shows the precision and recall values for both of our test videos for both image features: CPFs and GHs. Performance is plotted against the number of significant bits b used in the locality sensitive hashing step. Both image features perform in a similar way. Searching for clips with GHs is up to 30% faster than using CPFs. This is probably due to the better mixing characteristics of the gradient-based features. Color features are usually more clustered in feature space [10].

4.4 Mining Different TV Channels

This section is dedicated to the practical operation of our system. We apply the system to a variety of broadcast stations from different countries with different content, different ad presentation styles as well as different amounts of advertisements. For every video analyzed we build a database of repeating sequences and label all entries $\tilde{N}$ in the database manually as commercials or non-commercials. According to

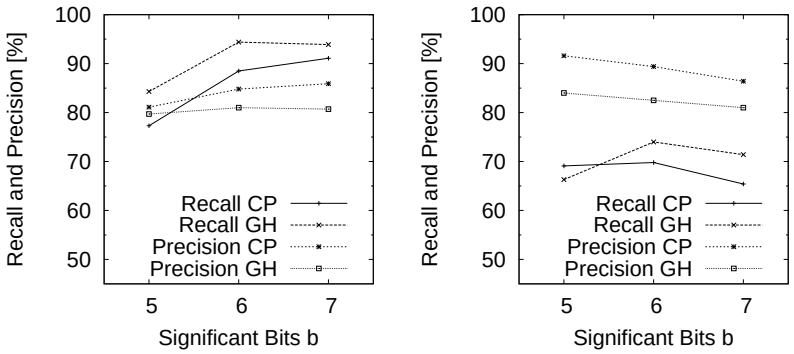


Fig. 13: Performance values for test videos (a) Chart TV and (b) Sky Sports News plotted against the number of significant bits b used for quantization in the computation of the inverted index.

these values we can evaluate the precision of our automatic mining for commercials. However, it is almost not possible to predict the recall from the maximum expected amount of advertisements, because the real fraction may vary from station to station (see Tab. 1), and is influenced by the time of day, the day of week, or even single events. Note, that the precision $\tilde{P}^C$ may differ from the true value reported in Tab. 2, because it may happen here that due to a recognition error a sequence is added twice as different sequences to the database.

Table 3: Commercial detection performance of our system for different broadcast stations. Given are the durations of the videos we mine, the number $\tilde{N}^C$ of commercials found, the number $\tilde{N}$ of all repeated clips found, the precision $\tilde{P}^C = \tilde{N}^C/\tilde{N}$, the number $\tilde{N}^C_{\pm 0}$ of all frame-accurately found commercials, and the number $\tilde{N}^C_{\pm 5}$ of commercials found if a maximum error of ± 5 frames at the start and/or end position is tolerated.

TV station	Video length [h]	$\tilde{N}^C$	$\tilde{N}$	$\tilde{P}^C$ [%]	$\tilde{N}^C_{\pm 0}$	$\tilde{N}^C_{\pm 5}$
ARD	60	10	49	20.4	2	8
RTL A	13	31	34	91.2	1	30
RTL B	11	12	16	75.0	0	12
RTL C	13	25	35	71.4	1	24
Chart Show TV	51	102	130	78.5	96	6
MTV	48	67	138	48.6	59	8
MTV with mask	48	70	115	60.9	58	12
Sky Sports News A	48	166	221	82.6	156	10
Sky Sports News B	63	178	315	56.5	147	31
Gemini	48	25	79	31.6	20	5
CBS 5 A	4	15	18	83.3	4	11
CBS 5 B	12	32	45	71.1	9	23
ESPN	10	21	27	77.8	1	20

Table 3 lists the experimental results for various broadcast stations. We tested the algorithm on TV channels, which contain only a small amount of commercials such as the German public broadcaster ARD, as well as on private broadcast channels like Sky Sports News (UK) or RTL (Germany), which devote about 20% of their broadcast time to commercials (see Tab. 1). Note that 20% is the limit in the EU for private broadcast channels due to a directive of the European Union [2], allowing a maximum of 12 minutes of advertisement per hour.

The regulations for public broadcast in Germany are even stricter. Averaged over the year only 20 minutes per day of commercials are allowed to be broadcast, but not more than 25 minutes per day (see [3]). These rules make commercials a rare event in our ARD video stream. Consequently, we only detected 10 distinct commercials within 2.5 days at a low precision of about 20%. Among the falsely detected clips are a number of channel previews with durations typical for commercials. The remaining false detections are mainly caused by repeated news stories. For the three relatively short searches in RTL we achieve a good precision of 70–90%.

MTV (UK) is not only a music channel like Chart Show TV (Chart TV), but also broadcasts shows and TV series, especially for young people, with a large amount of commercials. Here, we can improve the low precision of less than 50% by using a spatial mask, which neglects the first two rows of the 8×8 subareas. By this means we can significantly reduce false detections, which have been caused by overlaying the music clip title in the upper part of the video frames. Because these overlays may vary across repetitions of a music clip, clips are not detected as a whole. Detected subsequences may unfortunately have the typical durations of commercial.

Sky Sports News A corresponds to our test video. Sky Sports News B is a different recording of the same channel. This time precision is much lower as our algorithm fails probably due to repeatedly broadcast sports events from NTSC footage that is not repeated frame identical. This results in the detection of sequences with arbitrarily beginning and ending. The lack of robustness in this specific case is a disadvantage of our algorithm.

Gemini is an Indian TV channel. Here, we guess a precision of about 30%. However, for non-natives it has been difficult to rate the results: what are commercials and what not? For the same reason we could not analyze the problems of our algorithm.

CBS 5 and ESPN are US broadcast stations. They differ from the other stations by their TV norm NTSC instead of PAL. Nevertheless, we get comparable precision values. Most false alarms result from previews and channel advertisements, which are presented commercial-like. We include the relatively short recording named CBS 5 A into our experiments, because we know the ground truth of this video. We can determine the recall to be 78.9%.

In Tab. 3 we added information about the accuracy of the detected sequences. Usually, 80% and more of the detected commercials have been detected frame-accurate, except for the German and US stations. Here, the separating black frames are detected as parts of the repeated sequences. we identify dissolves and fade-ins and outs between commercials in the US TV as the source of problems.

5 Related Work

Analyzing TV broadcasts comprises several tasks such as the recognition of copies of known video clips, the identification of certain events, or the detection of video sequences of interest. In most cases all approaches are related to each other. So, you may first detect sequences of interest and then search for copies. Or you first identify all repeating sequences and than extract a particular subset. Commercial detection is one well-studied topic of how all of these approaches can be applied to TV streams. Other topics are, for example, news tracking or monitoring of sports events.

In the following we will give an overview of the approaches for commercial detection and cast the detection of repeated sequences in the context of video retrieval.

If we do not have a database of commercials at hand, a feature-based detection approach is needed. A first step is to identify commercial blocks based on the special characteristics of commercials as well as their presentation by the TV stations. In [25] Lienhart et al. (1997) derived typical ad properties such as their restricted lengths, their high dynamic content, which can be measured by an increased cut-rate, and the occurrence of still images at the end of spots presenting company related information, which are typically accompanied by the appearance of a certain amount of text. Additionally, they extracted characteristic features of commercial blocks, which include the separation of single commercials by black frames and an increased audio volume. The particular laws of a country may add more properties such as the requirement of intro/outro sequences which clearly separate advertisements from program content and the disappearance of the channel logo during commercial breaks.

This approach of identifying commercial blocks has been refined by other investigators. Dimitrova et al. (2002) [9] and Agnihotri et al. (2003) [4] extracted commercial triggers from low-level MPEG decoding. At this level they could derive information about the occurrence of black and unicolored frames, hard cuts, and changes from and to letterbox formats. In [4] a genetic algorithm is used to exploit the various analyzed features. Glasberg et al. (2006) [16] combined the appearance of black frames, the disappearance of station logo, shot duration, and cut rate in a decision tree to reliable recognize commercials, whereas Albiol et al. (2004) [5] limit characteristic features to logo disappearance and shot duration (cut rate). In [8] Chen et al. (2005) focused on the separation of commercials from news content. Therefore, they combined cut frequency with a caption detector due to fewer captions in commercials than in newscasts. Furthermore, they implemented a speech-music discriminator by combining typical audio-visual characteristics. Duan et al. (2006) implemented a complete system consisting of commercial boundary detection, commercial classification and commercial identification [12]. In the first step they seek for commercials using shot detection and audio scene change together with the detection of black frames, silence and still images at the end of commercial clips, which they mark as frames with product information. Especially from these frames they extract keywords by means of OCR for the classification step and get information about commercial content to identify the product line. For identifying clips they use image features like ordinal representations and histograms of them.

There are several disadvantages of methods recognizing commercial blocks based on their characteristic features. A certain amount of commercials are atypical. They have, for example, only a few cuts or even no cuts at all. They can appear like news stories, movies or cartoons. Also laws vary between countries and in time, which makes e.g. the logo disappearance during ad breaks not universally applicable. Besides the separation by black frames we can find hard cuts or dissolves between consecutive commercials. Advertisements must not be part of a commercial block at all, but can be broadcast as a single spot. This implies the need for a more universal algorithm.

Detecting known commercials avoids the limitations of the feature-based approach. Therefore, a lot of frameworks follows this idea. Lienhart et al. (1997) proposed an algorithm which makes use of approximate substring matching for comparing the video input with sequences from a database [25]. Fingerprints based on color coherence vector [27] of all known commercials are stored in the database. Alternatives for fingerprinting as well as the detection and comparison methods have been proposed. Sánchez and Binefa (1999), for example, reduced the fingerprint storage by only storing image features for key frames [29]. Each key frame represents a single shot. They reduce the dimension of the fingerprint further by applying Principle Component Analysis (PCA) to the color histograms feature vectors.

Not only image features, but also temporal information from video stream can be taken into account. Hoad and Zobel (2003) [21] improved their very compact signature from [22], which only contains the duration of shots of a given video sequence, to make it applicable to video clips containing only a few cuts. They investigate features, which describe the differences between consecutive frames, such as color shift, the distance between color histograms, and centroids, which correspond to the motion vectors of the darkest and the lightest part of the pictures.

As a fast alternative for comparing two sequences, the use of inverted indexes and hash tables has been investigated by Shivadas and Gauch (2007) in [30]. They implemented a real-time commercial recognition system on the basis of color moments with frame-level hashing. The use of hash tables provides constant access to large databases. Inverted indexes have been introduced to image retrieval by Squire et al. (1999) [31].

For building a database for commercial recognition the detection of repeated sequences is an appropriate technique. Pua et al. (2004) introduced a real-time repeated video sequence identification system [28]. They extract color moments features for all frames. Additionally they execute a temporal segmentation to get the shot length to which an image belongs. Based on this temporal segmentation they look for repeated shots by counting similar images, which are fast identified by a hash table look-up. Gauch and Shivadas (2005) extend this approach to a commercial detection system by adding a classifier, which labels repeated sequences as commercials or non-commercials [14]. At this step they investigate the already mentioned typical properties such as black frames or high cut rate.

Duygulu et al. (2004) only operate on the shot level by extracting key frames after temporal segmentation [13]. They carry out a search for similar images concerning a combination of color and edge based features together with a face detector. In their

second approach they merge audio and color features. With a proper combination of both strategies they can improve their performance values. Together with Can he developed in 2007 a system for near duplicate sequence detection, which are not real copies, but can vary in illumination or view points [7]. These aspects are more related to news tracking, and require image features, which are robust to such variations, such as SIFT features [26] and HSV statistics. They construct a tree for finding sequences within the lists of similar key frames.

In contrast to the previous approaches Yuan et al. (2007) provide an algorithm without shot detection for finding repetitive clips [32]. They chop the video stream into small overlapping segments and extract visual signatures for these small segments by averaging image color histograms and combining ordinal measures of all frames into a histogram for the video segment. They build continuous paths for finding repeated sequences through the lists of similar video segments.

6 Conclusion

We introduced an algorithm for detecting repeated sequences in TV streams. The algorithm is designed to operate in real-time on live streams. The main intention of our framework is the creation of databases of recurring sequences. For later use of such databases it is necessary to determine beginning and end of such sequences frame-accurately, in contrast to other applications, which, for instance, only count the number of repetitions. We applied the system to commercial detection as a possible field of application.

Our algorithm works well for broadcast station with only little amount of advertisements as well as for stations heavy loaded with commercials. It is applicable to both PAL and NTSC broadcasts. Our experiments showed that it is sometimes necessary to adjust the system to channel characteristics to improve the performance of repeated sequence detection in general and commercial detection based on simple duration filters in specific. For channels making heavy use of overlay information such as MTV or some sports casts, blocking out the spatial areas, which are often used to show overlay text, from the image feature computation improves the detection performance significantly. In other cases we had to deal with constantly appearing black frames for commercial separation. There was only one situation our algorithm could not handle: the non-frame accurate repetition of sequences. Our algorithm could not robustly detect these repetitions, since no provisions have been taken for this specific scenario.

In our tests we investigated two different image features: color-based Color Patches Features (CPFs) and edge-based Gradient Histograms (GHs). For our test videos both feature types performed in a similar way. Color patches are faster to evaluate and need a smaller amount of storage, but they are less discriminative and may cause problems for ill-conditioned video streams [10].

We can finally conclude that we developed an algorithm for reliable real-time detection of recurring sequences, which is successfully applied to broadcast stations from all over the world.

Acknowledgements This project was supported by Half Minute Media Ltd.

Appendix

The reduction of the feature vectors' dimensionality by summation over subsets or all components, respectively, preserves the relation of distances in the sense that images, which are close together in the feature space, are also close together in the reduced space. In the following we proof that the distance according to the L_1-norm in the reduced dimension is always equal to or less than the original distance.

Let $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$ be two n-dimensional feature vectors with the L_1-distance

$$d(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} |a_i - b_i|, \tag{33}$$

and $\mathbf{A} = (A_1, \ldots, A_k)$ and $\mathbf{B} = (B_1, \ldots, B_k)$, $k < n$, the corresponding feature vectors in the k-dimensional reduced space with each of their components being a sum of a subset of the components of the feature vectors **a** and **b**, respectively, i.e.,

$$A_i = \sum_{j=m_i}^{n_i} a_j, \qquad B_i = \sum_{j=m_i}^{n_i} b_j, \tag{34}$$

where without loss of generality

$$m_1 = 1, \qquad m_i \leq n_i \quad \forall i, \qquad m_{i+1} = n_i + 1 \quad \forall i, \qquad n_k = n.$$

Then, the L_1-distance D(**A**, **B**) is always less than or equal to d(**a**,**b**), because

$$D(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{k} |A_i - B_i| \tag{35}$$

$$= \sum_{i=1}^{k} \left| \sum_{j=m_i}^{n_i} a_j - \sum_{j=m_i}^{n_i} b_j \right| \tag{36}$$

$$= \sum_{i=1}^{k} \left| \sum_{j=m_i}^{n_i} (a_j - b_j) \right| \tag{37}$$

$$\leq \sum_{i=1}^{k} \sum_{j=m_i}^{n_i} \left| a_j - b_j \right| \tag{38}$$

$$\leq \sum_{i=1}^{n} |a_i - b_i| \tag{39}$$

$$\leq d(\mathbf{a}, \mathbf{b}). \tag{40}$$

References

1. Half minute media, www.halfminute.com
2. Council directive 89/552/eec of 3 october 1989 on the coordination of certain provisions laid down by law, regulation or administrative action in member states concerning the pursuit of television broadcasting activities (1989)
3. Staatsvertag für Rundfunk- und Telemedien (Rundfunkstaatsvertrag – RStV –) (2008)
4. Agnihotri, L., Dimitrova, N., McGee, T., Jeannin, S., Schaffer, D., Nesvadba, J.: Evolvable visual commercial detetctor. In: Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03). IEEE Computer Society, Madison, Wisconsin (2003)
5. Albiol, A., Fullà, M.J.C., Albiol, A., Torres, L.: Detection of tv commercials. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, vol. 3, pp. 541–544. Montreal, Canada (2004)
6. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM **51**(1), 117–122 (2008). DOI http://doi.acm.org/10.1145/1327452.1327494
7. Can, T., Duygulu, P.: Searching for repeated video sequences. In: MIR '07: Proceedings of the international workshop on Workshop on multimedia information retrieval, pp. 207–216. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1290082.1290112
8. Chen, J.C., Yeh, J.H., Chu, W.T., Kuo, J.H., Wu, J.: Improvement of commercial boundary detection using audiovisual features. In: Y.S. Ho, H.J. Kim (eds.) Advances in Multimedia Information Processing - PCM 2005, 6th Pacific-Rim Conference on Multimedia, Jeju Island, Korea, November 13-16, 2005, Proceedings, Part I, *Lecture Notes in Computer Science*, vol. 3767, pp. 776–786. Springer (2005)
9. Dimitrova, N., Jeannin, S., Nesvadba, J., McGee, T., Agnihotri, L., Mekenkamp, G.: Real time commercial detection using mpeg features. In: Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowlwdge-based Systems (IPMU2002), pp. 481–486. Annecey, France (2002)
10. Döhring, I., Lienhart, R.: Fast and effective features for recognizing recurring video clips in very large databases. In: International Workshop on Video and Multimedia Digital Library (VMDL 2007), pp. 65–70. Modena, Italy (2007)
11. Döhring, I., Lienhart, R.: Fast frame-accurate mining for repeating video clips. Technical Report Institut für Informatik, http://www.opus-bayern.de/uni-augsburg/volltexte/2008/1300 14, Universität Augsburg (2008)
12. Duan, L.Y., Wang, J., Zheng, Y., Jin, J.S., Lu, H., Xu, C.: Segmentation, categorization, and identification of commercial clips from tv streams using multimodal analysis. In: MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia, pp. 201–210. ACM, New York, NY, USA (2006). DOI http://doi.acm.org/10.1145/1180639.1180697
13. Duygulu, P., yu Chen, M., Hauptmann, A.: Comparison and combination of two novel commercial detection methods. In: Proceedings of the 2004 IEEE International Conference on Multimedia and Expo. Taipei, Taiwan (2004)
14. Gauch, J.M., Shivadas, A.: Identification of new commercials using repeated video sequence detection. In: Proceedings of the 2005 International Conference on Image Processing, pp. 1252–1255. Genoa, Italy (2005)

15. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases, pp. 518–529. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
16. Glasberg, R., Tas, C., Sikora, T.: Recognizing commercials in real-time using three visual descriptors and a decision tree. In: Proceedings of the 2006 IEEE International Conference on Multimedia and Expo, pp. 1481–1484 (2006)
17. Hampapur, A., Bolle, R.: Feature based indexing for media tracking. In: Proceedings of International Conference on Multimedia and Expo. New York (2000)
18. Hampapur, A., Bolle, R.: Videogrep: Video copy detection using inverted file indices. Technical report on some unpublished work, http://www.research.ibm.com/ecvg/pubs/arun-vgrep.html, IBM Research (2001)
19. Hampapur, A., Bolle, R.M.: Comparision of distance measures for video copy detection. Computer Science RC 22056, IBM Research (2001)
20. Hansen, A.: Bioinformatik : Ein Leitfaden für Naturwissenschaftler, second edn. Birkhäuser, Basel - Boston - Berlin (2006)
21. Hoad, T.C., Zobel, J.: Fast video matching with signature alignment. In: MIR '03: Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval, pp. 262–269. ACM, New York, NY, USA (2003). DOI http://doi.acm.org/10.1145/973264.973304
22. Hoad, T.C., Zobel, J.: Video similarity detection for digital rights management. In: ACSC '03: Proceedings of the 26th Australasian computer science conference, pp. 237–245. Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2003)
23. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing, pp. 604–613. ACM, New York, NY, USA (1998). DOI http://doi.acm.org/10.1145/276698.276876
24. Knuth, D.E.: Sorting and Searching, *The Art of Computer Programming*, vol. 3, second edn. Addison-Wesley, Reading, Massachusetts (1998)
25. Lienhart, R., Kuhmünch, C., Effelsberg, W.: On the detection and recognition of television commercials. In: Proc. IEEE Conf. on Multimedia Computing and Systems, pp. 509–516. Ottawa, Canada (1997)
26. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2) (2004)
27. Pass, G., Zabih, R., Miller, J.: Comparing images using color coherence vectors. In: ACM Conference on Multimedia, pp. 65–74. Boston, Massachusetts (1996)
28. Pua, K.M., Gauch, J.M., Gauch, S.E., Miadowicz, J.Z.: Real time repeated video sequence identification. Comput. Vis. Image Underst. **93**(3), 310–327 (2004). DOI http://dx.doi.org/10.1016/j.cviu.2003.10.005
29. Sánchez, J.M., Binefa, X.: Audicom: A video analysis system for auditing commercial broadcasts. Multimedia Computing and Systems, International Conference on **2**, 272 (1999). DOI http://doi.ieeecomputersociety.org/10.1109/MMCS.1999.778373
30. Shivadas, A., Gauch, J.M.: Real-time commercial recognition using color moments and hashing. In: CRV '07: Proceedings of the Fourth Canadian Conference on Computer and Robot Vision, pp. 465–472. IEEE Computer Society, Washington, DC, USA (2007). DOI http://dx.doi.org/10.1109/CRV.2007.53
31. Squire, D.M., Müller, H., Müller, W.: Improving response time by search pruning in a content-based image retrieval system, using inverted file techniques. In: CBAIVL '99: Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries, p. 45. IEEE Computer Society, Washington, DC, USA (1999)
32. Yuan, J., Wang, W., Meng, J., Wu, Y., Li, D.: Mining repetitive clips through finding continuous paths. In: MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia, pp. 289–292. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1291233.1291294