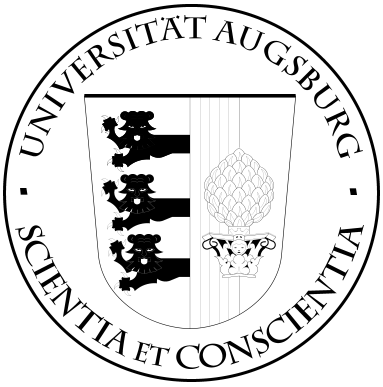
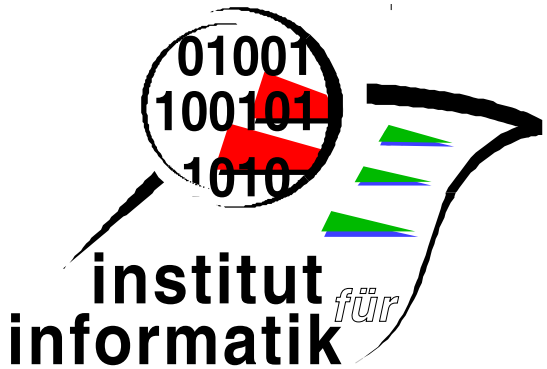
Universität Augsburg




# Fast Frame-Accurate Mining for Repeating Video Clips

**I. Döhring, R. Lienhart**

Report 2008-14 September 2008




Institut für Informatik
D-86135 Augsburg

# Fast Frame-Accurate Mining for Repeating Video Clips

Ina Döhring
Lehrstuhl für Multimedia Computing
Universität Augsburg
Augsburg, Germany
doehring@informatik.uni-augsburg.de

Rainer Lienhart
Lehrstuhl für Multimedia Computing
Universität Augsburg
Augsburg, Germany
lienhart@informatik.uni-augsburg.de

## ABSTRACT

In the broad range of multimedia content analysis tasks the detection of recurring video sequences plays an important role. We introduce an algorithm for recognizing recurring video sequences frame-accurately in a highly effective and efficient way. It does not require temporal segmentation by shot detection. A 24 hour live-stream can be processed in about 4 hours including the computational expensive video decoding. The algorithm uses 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 (e.g., 1 second long) and secondly assembling these small segments into the set of repeated whole 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 hours of two different TV channels. It is shown that the method is an excellent alternative for searching for unknown commercials.

## 1. INTRODUCTION

The steadily growing digital world provides an unmanageable amount of video data. There are several aspects that play a role in the analysis of existing video data. Well-known examples are video copy detection (mostly on the web), news tracking and commercial detection in TV programs. For these tasks several search strategies have been developed. A quite universal strategy is the search for repeating sequences, because all three tasks mentioned concern content replication. Therefore, we introduce a universal algorithm for retrieving recurring sequences of objects (e.g., frames) represented in a high-dimensional feature space from a data stream of objects. In our experiments we apply the developed system to the detection of unknown commercials.

The paper is structured as follows: in section 2 we discuss related work concerning repeating sequence identification and commercial detection. After that we introduce in section 3 our search algorithm, those major advantage is its independence of temporal segmentation and thus its applicability also to non-video content. We exploit the developed system for the detection of unknown commercials in section 4 and conclude with a summary in section 5.

## 2. RELATED WORK

Since we focus with the developed application on the detection of unknown commercials, we shortly discuss the prior art concerning this topic. An alternative to the search for repeating sequences is the detection of ads based on their technical and legal characteristics. One example is the disappearance of channel logos during ads. Its disappearance usually marks the start, its reappearance the end of a commercial break. A related algorithm is introduced by Albiol et al. [1]. Content related characteristics of ads such as high cut rate or their separation by monochrome (mostly black) frames are discussed by Lienhart et al. [10] and Dimitrova et al. [2]. A combination of channel logo detection and commercial characteristics is applied by Glasberg et al. [7].

A completely different approach is the detection of commercials on the basis of their repetition in broadcasts. This strategy works completely independent of ad-specific attributes. There are different approaches for realizing this concept. Pua et al. [12] introduce a repeated video sequence detection method, which relies on the temporal segmentation of the input video. Clip candidates are compared by their length and the number of similar frames using a color-based distance measure. Their algorithm finishes with a collection of shots, which are classified as unique or repeated. Gauch and Shivadas extend this algorithm for identifying new commercials [5]. They merge adjacent repeated shots; a subsequent classifier labels found sequences as commercials or non-commercials. The algorithm of Duygulu et al. relies on shot boundary detection, too [4]. However they restrict the similarity comparison to key frames of detected shots.

Yuan et al. avoid any shot detection [14]. They use small overlapping segments of 8 seconds and store the summarized segment features in an index using local sensitive hashing. Next they construct continuous paths of such similar small segments in order to identify repeated sequences in their original length.

## 3. SEARCH ALGORITHM

### 3.1 Overview

The main steps of our algorithm for searching for repeated sequences are depicted in Fig. 1. It can take any video or TV live stream as input.

A prerequisite for searching through a video, as it is customary, is a proper representation of the video. Therefore, the first step in our search algorithm extracts a suitable image feature from all video frames. In our case this feature is based on gradient histograms as it is explained in more detail in the next subsection.

Next we create a hash table on the feature vectors as our

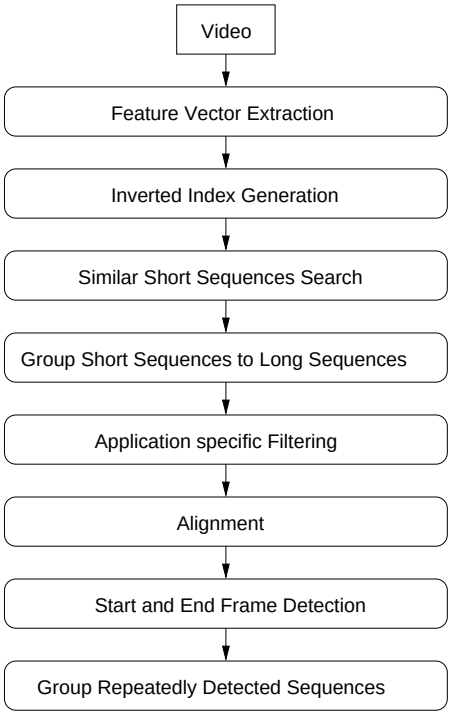



**Figure 1: Overview of our search algorithm.**

inverted index for fast search. Each hash table index contains a list of frame numbers whose image feature values were mapped to the same hash value. Frames with the same hash value are assumed to be visually similar. Thus, with one fast look-up we can retrieve all frame numbers with the same hash value and similar content to any query frame.

On the basis of this effective image retrieval we search for recurring sequences in the video. We look for duplicate small pieces with durations of about one second. Candidate duplicate sequences are rapidly identified by only requiring a minimal percentage of hash-value-identical frames. Some matches are the result of identical hash values from non-discriminative feature vectors or collisions in the hash table. Thus, candidate duplicate sequences are validated by verifying that the distance between sequences on the basis of the original feature vectors is below a maximally allowed threshold, resulting in the final list of similar short sequences.

The remaining duplicate short sequences are grouped to longer sequences according to their temporal coherence. After grouping we may insert a coarse filter to reject, for instance, very short sequences consisting of one short sequence duplicate only or very long self-similar sequences like in talk shows or some kinds of sports events. This can be done in dependence on the goal of our mining activities. It is mainly done to reduce the input to the relatively time consuming next steps: alignment and precise start/end frame detection. These two steps are required since short sequences duplicates are not necessarily aligned nor cover the whole recurring sequence. Finally we search through all sequences found to recognize and group multiple occurring sequences together. All these individual steps are explained in more detail in the next subsections.

## 3.2 Image Features

It is important to note that since we mine TV channels for repeating sequences, we do not require image features that are robust to different image qualities, different video capture cards, and/or video compression artifacts. Rather we record each video sequence on the same hardware, expecting all videos to be of the same quality. In this setting, gradient-based image features are normally more distinctive than color-based features, although the former would be less robust to distortions and transcoding operations. The more time consuming calculation of gradients can be elevated by appropriate table look-ups (see [3]). Thus, we have chosen to represent each individual frame by the *Gradient Histogram (GH)*, which was introduced and investigated in detail in [3]. It has been shown that for the same video capture equipment the GH features outperform all tested color features. The performance of color features strongly depended on the kind of video source, i.e., the kind of TV program [3].

The GH feature vector contains $N \times M$ gradient direction histograms, one for each subarea into which we have divided the whole image. For each subarea we compute a gradient direction histogram in a similar way done for the SIFT features [11]. Let $I(x_1, x_2)$ be the grayscale intensity with $I(x_1, x_2) \in (0, 255)$ and $\nabla I(x_1, x_2) = \left(\frac{\partial}{\partial x_1} I(x_1, x_2), \frac{\partial}{\partial x_2} I(x_1, x_2)\right)$ be the gradient intensity at point $(x_1, x_2)$ with intensity gradient magnitude $m_g$ and orientation $\theta_g$:

$$\begin{aligned} m_g &= \sqrt{(I(x_1+1, x_2) - I(x_1-1, x_2))^2} \\ &\quad + \overline{(I(x_1, x_2+1) - I(x_1, x_2-1))^2}, \qquad (1) \\ \theta_g &= \arctan\left(\frac{I(x_1, x_2+1) - I(x_1, x_2-1)}{I(x_1+1, x_2) - I(x_1-1, x_2)}\right). \qquad (2) \end{aligned}$$

If we use $K$ bins to cover all possible gradient directions, we can accumulate the corresponding gradient magnitude values for each bin and define the normalized gradient histogram components by

$$H^k_{nm}(I(x_1, x_2)) = \frac{1}{F} \sum_{x_1=X_n}^{X_n+H_n-1} \sum_{x_2=Y_m}^{Y_m+W_m-1} \mathcal{M}_g(x_1, x_2) \qquad (3)$$

with

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

$$\begin{aligned} (X_n, Y_m) &- \text{first sample point of subarea } I_{nm}, \\ H_n &- \text{height of } I_{nm}, \\ W_m &- \text{width of } I_{nm}, \\ n &= 1, \ldots, N, \\ m &= 1, \ldots, M, \\ \theta_k &= (k-1)\, 360^\circ / K, \\ k &= 1, \ldots, K, \end{aligned}$$

and normalisation factor

$$\begin{aligned} F &= \sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{k=1}^{K} \sum_{x_1=X_n}^{X_n+H_n-1} \sum_{x_2=Y_m}^{Y_m+W_m-1} \mathcal{M}_g(x_1, x_2), \\ &= \sum_{x_1=1}^{H} \sum_{x_2=1}^{W} m_g(x_1, x_2). \qquad (4) \end{aligned}$$

We measure the distance between two images $I_1$ and $I_2$ with the $L_1$-Norm

$$D(I_1, I_2) = \frac{1}{NMK} \sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{k=1}^{K} \left| H_{nm}^k(I_1) - H_{nm}^k(I_2) \right| \quad (5)$$

and the distance between two sequences $S_1$ and $S_2$ of length $L$ by

$$D_L(S_1, S_2) = \frac{1}{L} \sum_{l=1}^{L} D(S_1(l), S_2(l)). \quad (6)$$

Thus a gradient histogram fingerprint consists of $N \times M \times K$ values $H_{nm}^k$. Due to the normalization in Eq. 3 we deal with floating point values in the range of $(0, 1)$. To reduce the size of the feature representation we map all values to 1-byte integer values $\mathcal{H}_{nm}^k$. Because the $H_{nm}^k$ values are not uniformly distributed in $(0, 1)$, we apply a linear mapping from range $(0, L) \rightarrow (0, 255)$, $L \in (0, 1)$ with saturation at the higher bound [3]:

$$\mathcal{H}_{nm}^k = \min\left(256/L \cdot H_{nm}^k, 255\right). \quad (7)$$

For $N = M = K = 8$ $L = 0.02$ is a good choice [3].

## 3.3 Inverted Index

An inverted index is an efficient method for fast search through large databases. Inverted indices are quite common in text retrieval and DNA analysis. In the text domain, 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, if all occurrences of a certain word are needed, a single look-up in the index is sufficient to retrieve this information. No expensive sequential or tree-based search through the text corpus is required. The only expensive step is the creation of the index; however, this must only be done once. A difference between image and text retrieval arises from the high-dimensional feature space used for image representation. There are several approaches to handle these feature vectors by inverted indices. Hampapur and Bolle [8] used an inverted index for each component of the feature vector. Another possibility is the use of a hash function, which maps the complete feature vector to a single scalar value, as it is done by Shivadas and Gauch [13].

A typical hash function is designed for mapping a sparse representation of a much larger feature space to an index space whose size is in the order of the data's dense representation. A good hash function provides a deterministic but not injective mapping from the feature space to the much smaller index space. In the ideal case, the hash function possesses good mixing characteristics in the sense that all occurring feature vectors in the representation are mapped to different hash values. The mapping of different values to the same hash value is called a collision. Typical hash functions with good mixing properties are based on the use of the modulo function [9].

In our system we use a three step hashing algorithm. In the first step we reduce the feature vector size by grouping similar images together:

$$\mathcal{H}^k = \sum_{n=1}^{N} \sum_{m=1}^{M} \mathcal{H}_{nm}^k \quad (8)$$

represents the image related gradient distribution. These values are robust to small changes in the images. Therefore, similar images are mapped to the same hash value in the next step. The size of the $\mathcal{H}^k$ is $8 + P$ bits, if $P$ is the smallest integer with $NM \leq 2^P$.

Next we evaluate the first hash value for every component of the feature vector by taking the first $B$ bits of every single value $\mathcal{H}^k$

$$h_k^1(I) = \mathcal{H}^k \div 2^{(8+P-B)} \quad (9)$$

$$h^1(I) = \sum_{k=0}^{K-1} \left(2^B\right)^k h_k^1(I). \quad (10)$$

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

$$R_{h^1}(B, K) = \left(2^B\right)^K, \quad B \in (0, 8 + P). \quad (11)$$

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 $K = N = M = 8$ [3]. Therefore we apply a modulo function to the first hash value $h^1(I)$

$$h^2(I) = h^1(I) \mod R_{h^2}, \quad (12)$$

with the index range $R_{h^2}$. We evaluate Eq. 12 in an iterative way with the Horner scheme while taking Eq. 10 and the characteristics of the modulo function into account:

$$h_1^2(I) = h_1^1(I) \mod R_{h^2}, \quad (13)$$

$$h_k^2(I) = \left(2^B \cdot h_{k-1}^2(I) + h_k^1(I)\right) \mod R_{h^2}, \quad k = 2, \ldots, K, \quad (14)$$

$$h^2(I) = h_K^2(I). \quad (15)$$

For our calculations we use an index range $R_{h^1} = 100,003$, which meets the criteria for choosing a good table size [9].

We pay no attention to possible collisions, because we filter the frames with same hash values in a later step by making use of the direct image related distance from Eq. 5.

## 3.4 Short Sequence Search

Since we do not know anything about the recurring sequences we search for, we look for very short repeating sequences of about 1 second. The assumption is that every possible sequence is composed of such small identical pieces. We scan our test video frame by frame, calculate the hash index of each frame and retrieve all similar frames by a simple look up in the inverted index. We only take into account matching frames, which come earlier in time, thus searching the video only into the past. Additionally, we reject matches, which are very close to the test 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 do not take less specific frames into consideration such as black frames or parts of long self-similar sequences. For instance, a tennis match with hours of the same camera perspective and relatively little activity would otherwise produce a lot of false positives slowing down the system dramatically.

For every matched frame we either start a new candidate similar short sequence or, if there is already one started at a frame preceding the current frame by less than the short sequence's duration, we increment the sequence's counter for matched frames by one. If we find more than 20% matched frames by hash indices within two short sequences, we calculate the distance of the feature vectors for all frames in the

short sequences (Eq. 6). If this distance is below the similarity distance threshold, both short sequences are considered as being similar. Figure 2 shows two probably similar short sequences. Corresponding frames with identical hash values are marked.

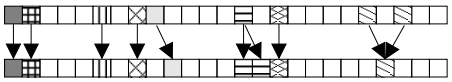


**Figure 2: Two short sequences with frames matched by hash values.**

## 3.5 Identifying Repeated Clips

### 3.5.1 Building Long Sequences

In this step we construct the raw candidate recurring sequences by grouping the short sequence candidates. Short sequence candidates, where both sequences are closer than 150 frames to the corresponding short sequences, are assumed to belong to the same recurring video sequence. After this step we have a number of sequences, which are pair-wise identical or similar in parts. Each pair of similar sequences is called a duplicate $\mathcal{D}$. Each duplicate contains two vectors $\mathcal{S}_1$ and $\mathcal{S}_2$, representing the corresponding sequences:

$$\mathcal{D} = (\mathcal{S}_1, \mathcal{S}_2)\,. \tag{16}$$

Each vector $S_i$ contains the start frame numbers of the short sequence candidates building the duplicate which are grouped to the following sequences:

$$\mathcal{S}_i = \left(\text{frame}_1^i, \ldots, \text{frame}_n^i\right), \tag{17}$$

with $n$ denoting the number of short candidates forming the duplicate and $\text{frame}_i^1$ and $\text{frame}_i^2$ originate from the same short sequence candidate. According to our search strategy the frames in $\mathcal{S}_1$ and $\mathcal{S}_2$ are not necessarily in the same temporal order, especially for long still scenes.

### 3.5.2 Coarse Filtering

At this stage it is possible to do a coarse filtering in order to reduce the input for the next steps. For instance, if we search for commercials, we discard sequences, which do not meet the required minimum length or surpass the demanded maximum length. Another possible criterion is a minimum number of short sequences. A more detailed explanation concerning this step is given in Section 3.6.

### 3.5.3 Alignment

Due to their construction both sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ may be shifted to each other and/or only be a part of the target sequence. Thus, we need proper aligned sequences with accurate start and stop frames. We use the intervals between cuts to calculate the displacement. We use this approach, because it is simple to implement and provides relatively exact results. The disadvantage is that we need more than 3 cuts for proper alignment and proportionally more, if there are errors in cut detection. We provide a fall back solution to circumvent the problem of discarding sequences with too few cuts by using the sequence distance $D_L$ from Eq. 6. Similar to Gauch and Shivadas [6] we search for a local minimum of $D_L$. We calculate the distance between two short 25 frames long substrings with varying offset. Both long sequences are then aligned by the offset, which has the (local) minimum distance of the corresponding feature vectors.

### 3.5.4 Estimation of Start and End Frames

After aligning both duplicate sequences we can compare frame by frame back from the shared start frame estimated by alignment as well forward from the estimated last frame until we identify that corresponding frames are different.

### 3.5.5 Collect Multiples

In a last step we compare the pairwise matched sequences against those from the other duplicates to group frequently occurring sequences together. We search for the case that one of both sequences also appears in another duplicate. If our detection results in pairwise different duplicates, our algorithm ends up with two final recurring sequences.

## 3.6 Content Related Filtering

As already mentioned in subsection 3.5.2 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. Special kinds of clips such as commercials can be retrieved by taking their characteristic properties into account and eliminating sequences which do not comply. A useful attribute is sequence length. For instance, you may distinguish between channel logos (typically just a few seconds long), commercials (typically 10 to 60 seconds long), and video clips (several minutes long). At this step you may include all features you know. For ads this can be a higher cut rate, black frames, higher audio volume, and others attributes, which are discussed by Lienhart et al. [10].

As we want as much as possible to be independent of most of such properties, since they can easily be changed by advertising industry, we restrict this filter for commercial detection to (a) sequence length and (b) a criterion to reject matches within long self-similar sequences as they may appear in talk shows. The last criterion is based on the assumption that the sufficiently dynamic content of commercials results in matching sequences $\mathcal{S}_1$ and $\mathcal{S}_2$, which have a well correlated temporal order. In the case of self-similarity we expect more matches of similar than identical short sequences. For instance, the presenter may occur in very similar shots at different times. As a measure for temporal correlation we use the standard deviation of the offsets between two corresponding short sequences

$$\mathcal{F}(i) = \mathcal{S}_2(i) - \mathcal{S}_1(i). \tag{18}$$

In the case, where we have a series of exact matching frames, the standard deviation

$$sd(\mathcal{S}_1, \mathcal{S}_2) = \sqrt{\frac{\sum_{i=1}^{n}\left(\mathcal{F}(i) - \overline{\mathcal{F}}\right)^2}{n-1}}, \quad \text{with} \tag{19}$$

$$\overline{\mathcal{F}} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{F}(i), \tag{20}$$

becomes zero. Therefore, we discard all sequences with a value of $sd(\mathcal{S}_1, \mathcal{S}_2)$ being greater than a threshold.

# 4. EXPERIMENTAL RESULTS

## 4.1 Experimental Setup

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 ($360 \times 288$ pixels) at 25 frames per second. 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 the ground truth of all recurring commercials in a video. Every human can recognize commercials at a glance while skimming a video. However, we are unable to manually label the ground truth for all kinds of repeating sequences within a reasonable time budget, since skimming in most cases would not work. For instance, determining the ground truth can be quite hard in case of talk shows or sportscasts. It would require from the human annotator to check conscientiously frame-by-frame whether two sequences are in fact frame-identical.

Therefore, in the course of the discussion of the system performance in its various configurations all recall and precision values will refer to the ground truth concerning recurring commercials.

The *recall* of repeating different commercials is defined by

$$R^C_{MD} = \frac{\#\text{ of found repeating different commercials}}{\#\text{ of repeating different commercials}}. \quad (21)$$

The term *different commercials* denotes that each repeating commercial is counted only once. For instance, if a commercial $A$ is repeated 3 times and a commercial $B$ 5 times, then we have 2 different commercials. The denominator will later be represented by the variable $N^C_{MD}$. 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.

Correspondingly, the *precision* of repeating different commercials is defined by

$$P^C_{MD} = \frac{\#\text{ of found repeating different commercials}}{\#\text{ of all found repeating different sequences}}. \quad (22)$$

It is important to note here that the precision value does 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. Examples are video clips in Charts TV, sports sequences (e.g., of a world record), or repetitions of whole reports in Sky Sports News. 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 used with commercials (see subsection 4.2.1). In practice, it is our observation that the true precision value of our system is very close to 1 for repeating video sequences. We hardly remember having ever seen a false alarm for repeating video sequences.

For both 48-hours sequences, we manually labeled all commercials occurring within the first 24 hours. In addition, we determined the ground truth for the second half of the Chart TV video sequence. This gives us the possibility to estimate the benefit of a 48-hours search over a 24-hours search. A two days search can detect commercials, which are only repeated once a day, but – as we will later see – at a much higher computational demand.

| | Chart TV | | Sky Sports News |
|---|---|---|---|
| | 24h | 48h | 24h |
| $N^C$ | 486 | 997 | 737 |
| $N^C_M$ | 428 | 928 | 650 |
| $N^C_S$ | 58 | 69 | 87 |
| $N^C_D$ | 164 | 212 | 245 |
| $N^C_{MD}$ | 106 | 143 | 158 |
| $N^C_M/N^C$ | 88.1 % | 93.1 % | 88.2 % |
| $N^C_{MD}/N^C_D$ | 64.6 % | 67.5 % | 64.5 % |
| $t^C$ / video length | 13.2 % | 13.5 % | 20.0 % |
| $t^C_M$ / video length | 11.6 % | 12.5 % | 17.4 % |
| $t^C_S$ / video length | 1.6 % | 1.0 % | 2.6 % |

**Table 1: Ground truth of our test videos: $N^C$ - number of all occurring commercials, $N^C_M$ - number of all repeatedly occurring commercials, $N^C_S$ - number of all singly occurring commercials, $N^C_D$ - number of different commercials, $N^C_{MD}$ - number of repeatedly occurring different commercials, $t^C$ - time covered by all occurring commercials, $t^C_M$ - time covered by all repeatedly occurring commercials, and $t^C_S$ - time covered by all singly occurring commercials.**

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

As we can see from Table 1, around two thirds of all broadcast commercials appear more than once a day, covering around 88% 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 time is devoted to non-repeating spots. These are the sequences that cannot even be found by a perfect search algorithm for repeating commercials. As expected, the fraction of repeated commercials increases for the 48-hours video due to spots, which are broadcast only once a day but repeated within the next day.

Figure 3 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

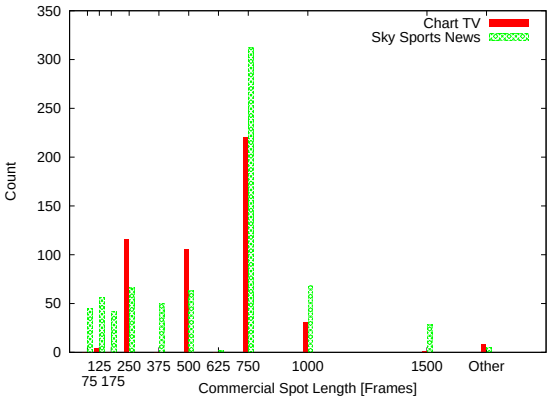



**Figure 3: Duration distribution of TV commercials.**

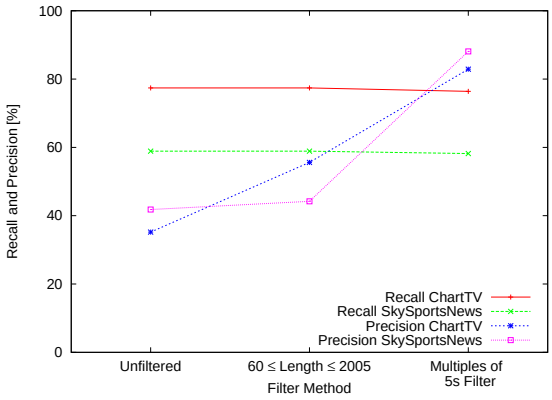



**Figure 4: Content related filter methods.**

we find a greater variance in spot durations with a tendency to shorter spots and a peak at half a minute (750 frames).

For each test video we extract the Gradient Histogram image features and build-up the inverted index based on the hash table as explained in Section 3.2. In the following sections we will discuss the influence of the various system parameters concerning the search algorithm on the retrieval performance and the required computational resources.

## 4.2 Test Cases

### *4.2.1 Content Related Filtering*

Our first experiment concerns the last step in our search algorithm - the content related filtering. We want to discuss this topic first because it has significant impact on our performance results, especially on the precision. As discussed above, our proposed algorithm mines videos for all kinds of recurring video sequences, of which commercials are just one example. However, we evaluate our system with respect to commercials, since the ground truth can easily be determined. While the 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.

Figure 4 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 Figure 3 and corresponds to sponsor advertisement in Sky Sports News. Such spots are typically broadcast at the start or end of a commercial block.

As shown in Figure 4 the "typical length filter" length filter improves the precision significantly compared to the unfiltered case. There is only a minor decrease in recall due to the small amount of commercials with non-typical duration (Figure 3). In the following experiments we will always filter the raw result lists with the "typical length filter", i.e., "multiples of 5s filter".

### *4.2.2 Alignment Method*

This experiment evaluates three different methods for aligning two found identical sequences to the proper start and end frames: One method is based on the alignment of detected cuts only, another is the method described in subsection 3.5.3, which implements a fallback solution for sequences with to few cuts based on the feature vector distances. Additionally, we investigate the case in which we apply the feature vector based method to all found sequences, completely avoiding any cut detection.

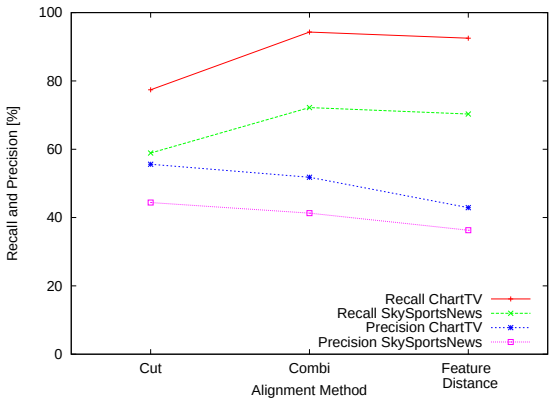



**Figure 5: Performance of the 3 alignment methods.**

Figure 5 shows the performance values for all three methods. We can see that the combination of cut based and feature distance based alignment results in the highest recall with a minor loss in precision compared to the cut based method only. This loss is due to the fact that in the cut based alignment procedure all sequences without enough cuts for proper alignment are rejected. On the other hand, in case of the feature distance based alignment none of the found recurring sequences are rejected and are all considered valid recurring sequences. This reduces the precision in ad detection. Furthermore, the chance of misalignment

is greater in this case, because start and end frames are sometimes not properly detected, leading to lower values for recall as well as for precision.

All subsequent experiments are carried out with the combined alignment method and performance values are evaluated with the "typical length filter".

### 4.2.3 Short Sequence Length

The following three test cases primarily concern the short sequence search. At first we investigate the impact of the

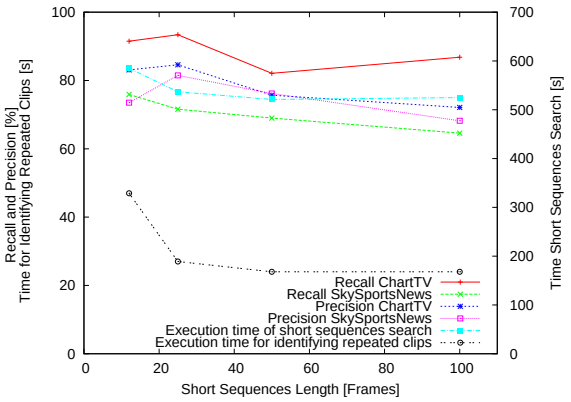



**Figure 6: Performance and execution times in dependence on the duration of the short sequences.**

length of the small segments. Figure 6 shows the recall and precision values for both test videos as well as the execution times for the short sequences search (see 3.4) and the whole clip identification (see 3.5). 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 the recall decreases with an increase in the length of the short sequences. For Chart TV the recall exhibits an exception of this rule for short sequences of 1 second duration. The precision behaves in the same way for both videos. This time, however, Sky Sports News shows an exception in the precision for 1 seconds sequences. In principal, the shorter the duration of the short sequences, 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 in (usually non-commercial) recurring sequences, which in turn affects precision negatively and leads to an increase in the execution times. Especially the time for identifying long sequences is nearly doubled. Nevertheless this step still takes only a few seconds.

All in all a sequence length of 25 frames (corresponds to 1 second-segments) seems to be an appropriate choice.

### 4.2.4 Minimum Fraction of Matched Frames

In this subsection we discuss a parameter which determines when two short sequences are regarded as candidates that must be tested for similarity. Remember, we are looking for similar images through a look up in the hash table to determine candidate sequence pairs. If the fraction of matched frames between two short sequences exceeds a threshold, their actual visual distance given in Eq. 6 is evaluated. If the fraction of matched frames between two short sequences is below the threshold, both short sequences are rejected right away as being similar.

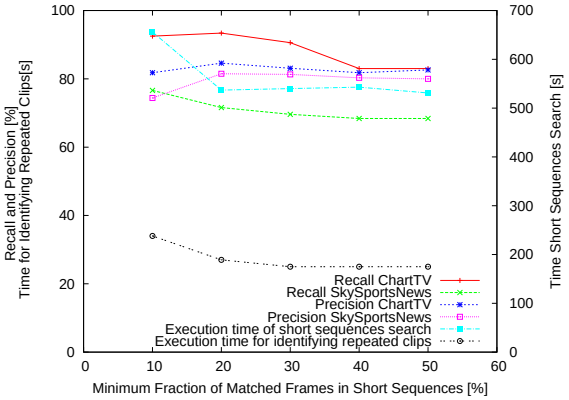



**Figure 7: Performance and execution times in dependence on the minimum required fraction of matched frames.**

Figure 7 plots performance values and execution times against different threshold values. 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 and a larger drop for values greater than 30% for Chart TV. The precision values reveal no significant dependence on the test parameter, being stable over a wide range while showing off a light diminution for very small values. Lower threshold values lead to a higher number of falsely detected short sequences. Most of them are discarded by the 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 7 by the rapid decline in execution time for the short sequence search for match ratios larger or equal 20%. However, a higher number of short sequences passing the similarity pre-filter has little consequences on the execution times for identifying repeated clips.

Taking into account performance as well as execution times, a threshold of 20% of matched frames is our preferred value.

### 4.2.5 Maximum Number of Entries in Hash Table

This test case concerns the search for similar frames in the inverted index. As explained in Section 3.4 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 – in this case the hash function must be redesigned – or result from unspecific (generic) images such as black frames. In the first case we would mark eventually completely different frames as matches; in the second case we would investigate a lot of similar but non-characteristic and non-unique matching sequences. Thus, this knockout criterion is introduced to keep evaluation times low for videos stream with long self-similar or many unspecific but similar image sequences.

As revealed in Figure 8, the influence of the investigated

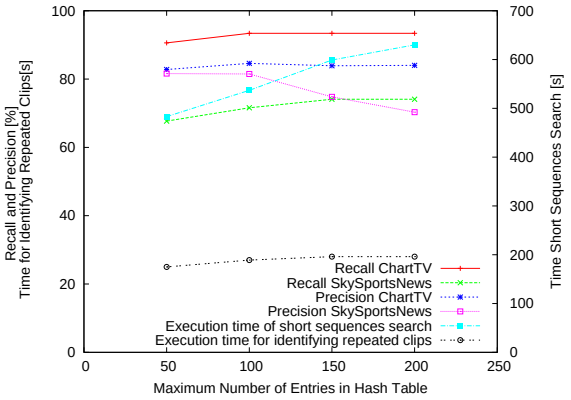



**Figure 8: Performance and execution times in dependence on the upper limit of entries per index in the hash table. All values refer to 2 hour slices.**

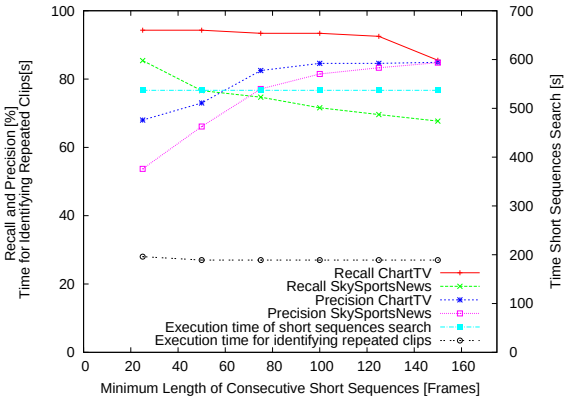



**Figure 9: Performance and execution times in dependence on the minimum length of sequences of short frames building duplicates.**

parameter on retrieval performance is quite different for our test videos due to dissimilar channel characteristics. Performance values for Chart TV search are quite independent of the specific parameter value, whereas we can observe changes for Sky Sports News. Taking hash values with a greater number of entries into account increase the recall for Sky Sports News, while decreasing the precision significantly. On Sky Sports News the transition from the programming to a commercial block and vice versa is typically accompanied by a sponsor related ad intro or ad outro. Both of these spots may vary in lengths and small details, but are repeated for many times. Thus, the various variants of the intro/outros are either not quite specific enough or repeated very often. Consequently, the recall concerning the commercials may be increased by expanding the range of frames which are investigated for similar frames.

While recall and precision for Chart TV are not affected by the variation of this parameter, the execution times needed for searching for short segments clearly increases with the upper limit for the number of entries per hash index: a greater number of matched frames increases the number of required feature distance computations for small short sequences.

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

### *4.2.6 Minimum Length of Short Segment Sequences*

This test case applies to the identification of repeated clips only. Requiring a minimum sequences duration $S_i$ when forming a duplicate $\mathcal{D}$ is one possible mechanism for coarse filtering as described in Paragraph 3.5.2. In fact we require a minimum count $n$ of small sequences 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 short sequences can be discarded.

In Figure 9 performance values and execution times for different required sequence lengths are drawn. During this experiment we scaled the required minimum number $n$ of short sequences in the same way.

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 Figure 3), whereas the precision is lowered 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 for increased duration thresholds, whereas the precision can be enhanced significantly.

The execution times for short sequence search is not influenced, because the filter is applied afterwards, the time for identifying repeated sequences can be slightly reduced for greater filter length due to fewer samples passing the filter.

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

## 4.3 Detection Performance

After our sensitivity analysis of the various parameters in subsection 4.2 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 from subsection 4.2, which led to the best results during our tests: we set the size of short sequences to 25 frames, require 20% of matched frames within a short sequence to be considered as possibly similar to another short sequence, 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 short segments. We use the combination of cut and feature vector based alignment and filter the result list of repeated clips by the 'multiple of 5s' filter that accounts for the typical ad durations.

Table 2 lists the performance values for our two 24 hours test videos as well as for the 48 hours search through Chart TV. We show different recall and precision values depending

| | Chart TV | | Sky Sports News |
|---|---|---|---|
| | 24h | 48h | 24h |
| $R_M^C$ | 91.8 % | 94.2 % | 68.9 % |
| $R_{MD}^C$ | 93.4 % | 92.3 % | 71.6 % |
| $R^C$ | 80.9 % | 87.7 % | 60.8 % |
| $R_D^C$ | 63.4 % | 66.5 % | 46.5 % |
| $P_M^C$ | 84.3 % | 80.6 % | 86.0 % |
| $P_{MD}^C$ | 84.6 % | 71.1 % | 81.5 % |

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

on the aspect a user wants to focus.

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. $R_{MD}^C$ can be higher, if for some spots not all repetitions are recognized, and smaller, if many repetitions of individual spots are reliably recognized, but others with only a few repetitions not at all. In practice, however, both values are usually close to each other. 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.

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), we achieve a reasonable recall for Chart TV, whereas for Sky Sports News these values are much smaller. According to subsection 4.2.6 the chosen values are not optimal for this kind of broadcast. Thus results could be enhanced by channel specific parameter settings. For instance, a 2-day search would increase the overall detection rate, because of several commercials, which are only repeated once a day.

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 other repeated non-commercial sequences.

## 4.4 Resources Requirements

In this subsection we discuss the execution times as well as the memory and storage requirements for our detection system. We first lay down all the inefficiencies in the implementation, so that the reader can get a more comprehensive idea of the efficiency of the proposed algorithm.

For reproducibility and system comparability all experiments are performed on the test videos, which are stored as MPEG-1 sequences on the hard drive. Thus all runtimes include the reading of the video from the disc as well as the video decoding; on a live-video stream none of these steps are required. Another important point is the modular design of the implementation: image feature calculation, hash table generation, short sequence search, and repeated clips identification are implemented as stand-alone components; the output of each step is 'piped' via the disc as input to the next step. This layout makes is easy to replace individual components, but at the cost of requiring many unnecessary and expensive disc I/O accesses. All time consuming I/O accesses are included in our runtime measurements. For technical reasons we process each test video in 2-hour slices, computing for each of them an individual inverted index.

The most time consuming step is the computation of the Gradient Histogram image features. For our test videos their calculation takes around 20 minutes for each 2 hours segment on an Intel Xeon 2.33 GHz CPU including video decoding. Generating the hash index table for 2 hours of video is about 10 seconds. Thus, the overall execution time for preparing recurring sequence detection is about 4 hours for every 24 hours of video material.

Table 3 shows the runtimes for the components concerning the search directly. Both steps, the short sequence search and the identification of repeated clips, together take about 10 minutes for a search through 24 hours, whereof the short sequence search accounts for approximately 90%. Due to the use of 2-hour slices in the actual implementation, the algorithm does not scale well as the overall duration of the video is increased to 48 hours; here it nearly takes an hour. This quadratic behaviour is an artifact of our implementation and not of the proposed algorithm which would be linear.

| Processing Step | Chart TV | | Sky Sports News |
|---|---|---|---|
| | 24h | 48h | 24h |
| Short sequences search | 537s | 2458s | 524s |
| Identifying repeated clips | 27s | 690s | 27s |

**Table 3: Execution times in seconds.**

As for the execution times the image features dominate the memory and disk space requirements. We deal with relative large feature vectors of 512 bytes per frame [3]. For 24 hours, i. e. 2,160,000 frames, we need about 1 GB to store the feature vectors. The hash table contains one entry per frame (a 4 byte integer frame number) resulting in approximately 8.2 MB a day. Additionally, each hash table consists of one integer value representing the number of entries per index value plus the table structure dependent overhead.

## 4.5 Discussion

Our experiments have shown that the investigated parameters produce relatively stable results over a wide range of settings. We introduced mechanisms to reduce execution time without losing retrieval performance and identified a parameter set with the best recall vs. precision trade-off. The runtime and the memory consumption analysis clearly identified the major computational bottlenecks of the current implementation: video decoding and image feature extraction. Note that video decoding is only needed for offline video processing, not on live streams. We have chosen to fingerprint each video frame by the relatively costly Gradient

Histogram, because our system setup is a real-time system, which constantly monitors a TV channel on the same hardware to build the commercial database. In this scenario, the edge-based image fingerprints provide better precision than color-based fingerprints [3].

# 5. CONCLUSION

The presented search algorithm is capable of detecting repeated video sequences in continuous live-video streams at a fraction of its play duration. Thus multiple channels can be monitored 24/7 on a single computer platform. By combining the results of each 24 hour search over multiple weeks, a complete record of all commercials and repeated video clips can be generated. The most time and space consuming components are the video decoding (in case of offline video) as well as the computation and the storing of the image features.

The search algorithm itself is independent of the concrete feature used except for the choice of an appropriate hash function. The introduced algorithm is also independent of prior temporal video segmentation (i.e., shot detection), and is thus universally applicable to any kind of data, not just video data. The repeating data can even be less structured and less striking than individual spots in video data. We have tested the proposed method for searching for unknown commercials on different TV channels. It could be shown that we can reach a recall over 90% of repeated commercial spots, which on average cover about 80% of all commercials broadcast each day. However, recall and precision depend on the channel characteristics, which can be compensated by channel specific parameter sets.

**Acknowledgments:** This project was supported by Half Minute Media Ltd.

# 6. REFERENCES

[1] A. Albiol, M. J. C. Fullà, A. Albiol, and L. Torres. Detection of tv commercials. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 541–544, Montreal, Canada, May 2004.

[2] N. Dimitrova, S. Jeannin, J. Nesvadba, T. McGee, L. Agnihotri, and G. Mekenkamp. 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)*, pages 481–486, Annecey, France, 2002.

[3] I. Döhring and R. Lienhart. Fast and effective features for recognizing recurring video clips in very large databases. In *International Workshop on Video and Multimedia Digital Library (VMDL 2007)*, pages 65–70, Modena, Italy, Sep 2007.

[4] P. Duygulu, M. Chen, and A. Hauptmann. Comparison and combination of two novel commercial detection methods. In *Proceedings of the 2004 IEEE International Conference on Multimedia and Expo*, pages 1267–1270, Taipei, Taiwan, 2004.

[5] J. M. Gauch and A. Shivadas. Identification of new commercials using repeated video sequence detection. In *Proceedings of the 2005 International Conference on Image Processing*, pages 1252–1255, Genoa, Italy, Sept. 2005.

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

[7] R. Glasberg, C. Tas, and T. Sikora. 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*, pages 1481–1484, July 2006.

[8] A. Hampapur and R. Bolle. Videogrep: Video copy detection using inverted file indeces. Technical report on some unpublished work, `http://www.research.ibm.com/ecvg/pubs/arun-vgrep.html`, IBM Research, 2001.

[9] D. E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, second edition, 1998.

[10] R. Lienhart, C. Kuhmünch, and W. Effelsberg. On the detection and recognition of television commercials. In *Proc. IEEE Conf. on Multimedia Computing and Systems*, pages 509–516, Ottawa, Canada, June 1997.

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[12] 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.

[13] A. Shivadas and J. M. Gauch. Real-time commercial recognition using color moments and hashing. In *CRV '07: Proceedings of the Fourth Canadian Conference on Computer and Robot Vision*, pages 465–472, Washington, DC, USA, 2007. IEEE Computer Society.

[14] J. Yuan, W. Wang, J. Meng, Y. Wu, and D. Li. Mining repetitive clips through finding continuous paths. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 289–292, New York, NY, USA, 2007. ACM.