UNIVERSITÄT AUGSBURG
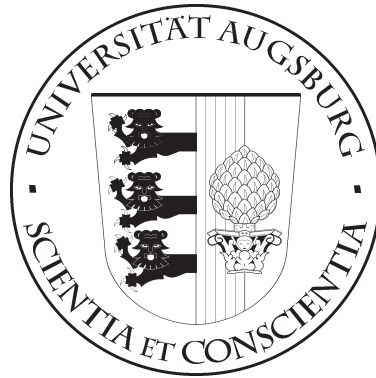
Mining TV Broadcasts for Recurring Video Sequences

I. Döhring, R. Lienhart

INSTITUT FÜR INFORMATIK
D-86135 AUGSBURG

Mining TV Broadcasts for Recurring Video Sequences [*]

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

We introduce an algorithm and a real-time system for mining TV broadcasts for recurring video sequences. The algorithm is frame-accurate, i.e., it exactly identifies with which frame a repeating sequence starts and ends resulting in a temporal accuracy of $40ms$ for PAL videos and $33ms$ for NTSC videos. The algorithm is also efficient. A 24-hour live-stream can be processed on a standard PC in less than 4 hours including the computational expensive video decoding. This efficiency is partially achieved by means of an inverted index for identifying similar frames rapidly. Images are mapped to the index by first calculating a gradient-based image feature, which in turn is mapped to the index via a hash function. The search algorithm consists of two steps: (1) searching for recurring short segments of e.g. 1 second length (called clips), and (2) assembling these small segments into sets of repeating long and complete video sequences. 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 7 days a week in various countries to log all broadcast commercials fully automatically.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms, Experimentation, Performance

Keywords

Frame-accurate video mining; recurring video clips; commercial detection

1. INTRODUCTION

Monitoring and analyzing TV broadcasts 24 hours a day 7 days a week is an important task in the media as well as advertising business. One important subtask is frame-accurate mining for recurring video sequences. Examples of recurring video sequences are commercials, channel advertisements, channel intros, and newscast intros. These different kinds of repeating video sequences can automatically further be classified by analyzing their temporal and/or visual properties. For instance, commercials can easily be extracted from the set of repeating video sequences based on their characteristic durations, temporal repetition patterns as well as visual/audio features [10].

In this work we introduce (1) a universal algorithm for retrieving recurring vector sequences from a stream of of high-dimensional vectors and (2) a real-time system that applies this algorithm (one vector per video frame) for mining TV broadcasts for recurring video sequences. The algorithm is frame-accurate. It exactly identifies with which frame a repeating sequence starts and ends. Thus the temporal accuracy in the boundary detection of repeating sequences is $40ms$ for PAL videos and $33ms$ for NTSC videos. The algorithm is also efficient. A 24-hour video-stream can be processed on a standard PC in less than 4 hours including the computational expensive video decoding. In order to monitor a complete broadcast market such as the UK or South Africa a system has to mine many TV channels on a single machine simultaneously in real-time as well as operate in streaming mode on live broadcasts such as our system does. Out of the set of repeated video sequences we show how to extract all commercials based of their characteristic features such as their specific temporal durations.

To avoid misunderstandings we want to point out that this algorithm is on purpose designed for a data stream comming 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 learning 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 accross different video formats and video fidelities. This is a completely different problem domain. To stay in the analogy, the latter problem setting would mean that a human would have to learn to see the same things with the eyes of a different person. Nevertheless our task is also challenging since the focus lies on temporal precision.

2. SEARCH ALGORITHM

The main steps of our algorithm for searching for repeating sequences are depicted in Fig. 1. It can take any offline video or TV live-stream as input. A prerequisite for searching through a video is a proper representation of the video. Therefore, our search algorithm extracts in a first step a suitable image feature from all video frames. This image feature is based on gradient histograms as explained in more detail in the next subsection. Next we create a hash table for the feature vectors as our inverted index for fast search. Each hash table entry 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 thus similar content to a query frame. On the basis of this efficient image retrieval we search for recurring *clips* – short video sequences with durations of about $1s$. Candidate duplicate clips are rapidly identified by only requiring a small percentage of hash-value-identical frames. Some matches may be the result of identical hash values from non-discriminative feature vectors or collisions in the hash table. Thus, verifying that the distance between the clips based on the original feature vectors is below a threshold validates candidate duplicate clips. The remaining duplicate clips are grouped to longer sequences according to their temporal coherence. After grouping we may insert a filter to reject, for instance, very short sequences consisting of only a single duplicate clip or very long self-similar sequences such as they can occur in talk shows or some kinds of sports events. This filtering is performed in compliance with the goal of our mining activities. It is mainly done to reduce the input to the relatively time-consuming next two steps: sequence alignment and precise start/end frame detection. These two steps are required since duplicate clips 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.
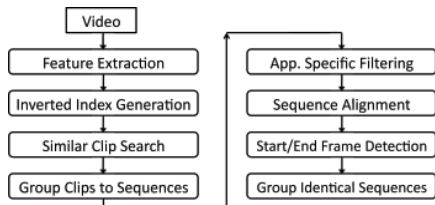

Figure 1: Overview of our search algorithm.

2.1 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 across different image fidelities, video capture cards, and/or video compression artifacts. Rather we record each video sequence on the same kind of hardware, expecting all videos to be roughly of the same fidelity. 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. Thus, we have chosen to represent each individual frame by a *Gradient Histogram (GH)*, which was introduced

and investigated in detail in [3]. It has been shown that for similar video fidelity GHs outperform all tested color features. The performance of color features strongly depended on the kind of TV programming. Also GHs can be computed rapidly with appropriate table look-ups (see [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 as done for SIFT features [11]. Let $I(x,y) \in (0, 255)$ be the grayscale intensity and $\nabla I(x,y) = \left(\frac{\partial}{\partial x} I(x,y), \frac{\partial}{\partial y} I(x,y) \right)$ be the intensity gradient at point (x, y) with magnitude m_g and orientation θ_g. 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

$$ H_{nm}^k (I(x,y)) = \frac{1}{F} \sum_{x=X_n}^{X_n + H_n - 1} \sum_{y=Y_m}^{Y_m + W_m - 1} \mathcal{M}_g^k (x,y) \qquad (1) $$

with

$$ \mathcal{M}_g^k = \begin{cases} m_g(x,y) & \text{if } \theta_k \leq \theta_g(x,y) < \theta_{k+1} \\ 0 & \text{else} \end{cases} $$

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

and normalisation factor

$$ F = \sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{k=1}^{K} \sum_{x=X_n}^{X_n + H_n - 1} \sum_{y=Y_m}^{Y_m + W_m - 1} \mathcal{M}_g^k(x,y), $$

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

$$ D(I_1, I_2) = \frac{1}{NMK} \sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{k=1}^{K} \left| H_{nm}^k(I_1) - H_{nm}^k(I_2) \right| \quad (2) $$

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

$$ D_L(S_1, S_2) = \frac{1}{L} \sum_{l=1}^{L} D(S_1(l), S_2(l)). \qquad (3) $$

Thus a GH fingerprint consists of the $N \times M \times K$ values of H_{nm}^k. Due to the normalization in Eq. 1 we deal with floating point values in the range of $[0, 1]$. To reduce the size of the GH 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 linearly map the range $[0, L] \rightarrow [0, 255]$, $L \in [0, 1]$ with saturation at the higher bound [3]:

$$ \mathcal{H}_{nm}^k = \min\left(256/L \cdot H_{nm}^k, 255 \right). \qquad (4) $$

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

2.2 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 be done only 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 the hash value is computed in three steps. In a first step we reduce the feature vector size by computing the image gradient histogram:

$$\mathcal{H}^k = \sum_{n=1}^{N} \sum_{m=1}^{M} \mathcal{H}_{nm}^k \qquad (5)$$

The size of the $\mathcal{H}^k$, i.e., $|\mathcal{H}^k|$ is $8+P$ bits, if P is the smallest integer with $NM \leq 2^P$. Next we take the B most significant bits of every component of $\mathcal{H}^k$:

$$h_k^1(I) = \mathcal{H}^k \div 2^{(8+P-B)} \qquad (6)$$

$$h^1(I) = \sum_{k=0}^{K-1} \left(2^B\right)^k h_k^1(I). \qquad (7)$$

This hash function h^1 is robust to small changes in the images and thus maps similar images to the same hash value. The corresponding number of possible values $R_{h^1}(B, K)$ gives the range of this first hash value:

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

Subject to 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 $h^1(I)$

$$h^2(I) = h^1(I) \mod R_{h^2}, \qquad (9)$$

with the index range R_{h^2}. For our calculations we use an index range of $R_{h^2} = 100,003$, 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. 2.

2.3 Clip Search

Since we do not know anything about the recurring sequences we search for, we look for very short repeating sequences named *clips* of about 1 second in length. We assume that every possible repeating 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 backwards into the past. Additionally we reject matches, which are temporally very close to the test frame. A minimal gap of 2000 frames is required to avoid detection within the same scene. We further neglect frames belonging to hash indices with an excessive large number of entries that usually result from less specific frames 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 as well as slowing down the system dramatically. For every matched frame we either start a new candidate repeating clip or, if there is already one started at a frame preceding the current frame by less than the clip's duration, we increment the clip counter for matched frames by one. If we find a clip in the past with more than 20% matched frames by hash indices to the clip ending at the current frame, we calculate the clip distance using the clips' GHs according to Eq. 3. If this distance is below a similarity distance threshold, both clips are considered as being similar. Fig. 2 shows two probably similar clips; corresponding frames with identical hash values are marked.
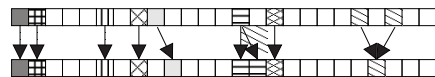

Figure 2: Two clips with frames of matching hash values.

2.4 Identifying All Repeating Sequences

We discuss next how to construct the set of all repeating video sequences up to the currently processed video frame.

Building Longer Sequences: We construct the raw candidate recurring sequences by grouping the similar clips found. Two similar clip pairs, whose corresponding clips are temporally closer than 150 frames, are assumed to belong to the same recurring video sequence. This step results in 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 $\mathcal{D}$ consists of two vectors $\mathcal{S}_1$ and $\mathcal{S}_2$ that represents the corresponding sequences: $\mathcal{D} = (\mathcal{S}_1, \mathcal{S}_2)$. Each vector S_j contains the start frame numbers of the n clips building the duplicate sequence j:

$$\mathcal{S}_j = \left(\text{frame}_1^j, \ldots, \text{frame}_n^j\right), \qquad (10)$$

with frame_i^1 and frame_i^2 denoting the respective first frame number of the i-th similar clip pair. According to our search strategy the frames in $\mathcal{S}_1$ and $\mathcal{S}_2$ are not necessarily in the correct temporal order, especially for long still scenes.

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 or maximum length. Another possible criterion is a minimum number of clips. A more detailed explanation concerning this step is given in Section 2.5.

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. For sequences with 3 or less cuts we search – similar to Gauch and Shivadas [6] – 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 (locally) minimum distance D_L.

Estimation of Start and End Frames: After aligning both duplicate sequences we can compare frame by frame backwards from the estimated start frame by alignment as well forward from the estimated last frame until we identify that corresponding frames are different.

Grouping Repeating Sequences: In a last step we compare all pairwise matched sequences against each other to group frequently occurring sequences together. We also search for the case that a sequence pair is a subsequence of a sequence group. By default this sequence pair is considered to be part of a group of its own. However, this information may adaptively be used in the content related filtering to spilt up commercials that frequently are shown in the same succession and are thus grouped into one long repeating sequence consisting of two or more commercials. This final interative process result in pairwise dissimilar groups of repeating sequences – our final mining result.

2.5 Content Related Filtering

Typically not all kinds of recurring sequences are of interest. We can adapt the final mining result to our needs by filtering the sequences in dependence on the content we search for. At this step we may include all features we know. For commercials this can be a higher cut rate, black frames, higher audio volume, and others attributes as discussed by Lienhart et al. [10]. Another simple, yet useful and almost sufficent property is sequence length. It allows for distinguishing commercials (typically 10 to 60 seconds long) from channel idents (typically just a few seconds long) and video clips (several minutes long). Thus, we detect commercials by means of (a) sequence length and (b) a measure for temporal clip order correlation between similar sequences. The latter measure rejects repeating self-similar sequences as they may appear in talk shows. It is based on the assumption that the dynamic and thus temporally distinctive content of commercials results in matching sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ of almost perfectly temporally ordered matching clips. In contrast, for non-dynamic and therefore self-similar sequences we expect more matches of similar than identical clips leading to wrongly shuffled temporal associations. Thus we measure the *temporal correlation* $sd(\mathcal{S}_1, \mathcal{S}_2)$ by the standard deviation of the offsets $\mathcal{F}(i)$ between two corresponding clips i:

$$sd(\mathcal{S}_1, \mathcal{S}_2) = \sqrt{\frac{\sum_{i=1}^{n} \left(\mathcal{F}(i) - \overline{\mathcal{F}}\right)^2}{n-1}}, \quad \text{with} \quad (11)$$

$$\mathcal{F}(i) = \mathcal{S}_2(i) - \mathcal{S}_1(i), \overline{\mathcal{F}} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{F}(i). \quad (12)$$

In the case of a series of exact matching frames, the standard deviation becomes zero. Therefore, we discard all sequences with an $sd(\mathcal{S}_1, \mathcal{S}_2)$ value greater than a threshold.

3. EXPERIMENTAL RESULTS

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 fps. 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, it is almost impossible to manually label the ground truth for all kinds of repeating sequences within a reasonable time budget, since skimming would not work in most cases. Determining the ground truth can be quite hard e.g. 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.

Our performance measures of repeating different commercials are *recall* R_{MD} and *precision* P_{MD}:

$$R_{MD} = \frac{\text{\# of found repeating different commercials}}{\text{\# of repeating different commercials}}. \quad (13)$$

$$P_{MD} = \frac{\text{\# of found repeating different commercials}}{\text{\# of all found repeating different sequences}}. \quad (14)$$

The term *different commercials* denotes that a specific, but multiple times repeated 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. A commercial spot is *found*, if the start and end frame differ no more than 5 frames from its 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.

It is important to note 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 such as of world records, 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 3.1). **In practice, it is our observation that the true precision of our system is very close to 1 for repeating video sequences**. We hardly remember having ever seen a false alarm.

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 cost.

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

Table 1: Ground truth of our test videos.

Table 1 reports key numbers about our test videos: N specifies the number of occurring commercials. For each test video N can be split into the number of ads N_M which are repeated and N_S which are occurring only once (subscript M/S stands for *multiple/single* occurrence), i.e., $N = N_M + N_S$. The overall number of different commercials is denoted by N_D, of which only N_{MD} are repeated in the overall test video. Thus, the following relation holds: $N_D = N_{MD} + N_S$. 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. 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 be found by even a perfect search algorithm for repeating commercials. As expected, the fraction of repeating commercials increases for the 48-hours video due to ads, which are broadcast only once a day but repeated within the next day.
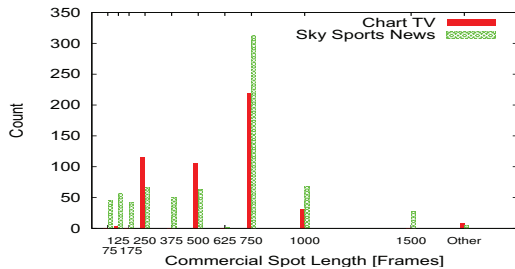
Figure 3: Duration distribution of TV commercials.

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

3.1 Test Cases

In the following we will discuss the influence of the various system parameters of the search algorithm on retrieval performance and required computational resources.

Content Related Filtering: Our first experiment concerns the last step in our search algorithm - the content-related filtering. We discuss this topic first, because it has significant impact on our performance numbers, especially 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 recall is not affected by focusing on commercials, it renders the precision value useless. Hence 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 search algorithm for repeating video sequences?*
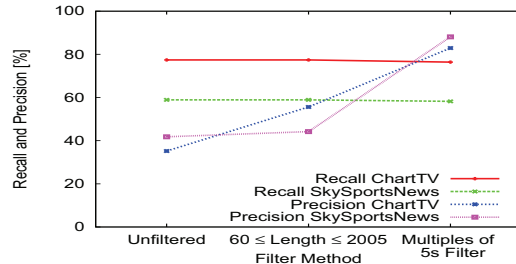
Figure 4: Content related filter methods.

Fig. 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) as well as sequences of 75 and 175 frames. These values are derived from the duration distribution in Fig. 3. The two special cases (75 and 175 frames) 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 Fig. 4 the typical length filter labeled "multiples of 5s filter" improves precision significantly compared to the unfiltered case. The decrease in recall is only minor due to the small amount of commercials with non-typical durations (see Fig. 3). All subsequent experiments are carried out with the "multiples of 5s filter".

Alignment Method: This experiment evaluates three different methods for aligning two 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 2.4, which implements a fallback solution for sequences with too 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, herewith avoiding any cut detection.

Fig. 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, but considered valid. This reduces the precision in ad detection. Furthermore, the chance of misalignment is greater in this case, leading to lower values for recall as well as for precision. All subsequent experiments are carried out with the combined alignment method.
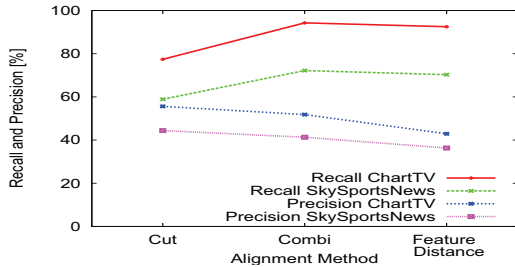

Figure 5: Performance of the 3 alignment methods.

Clip Length: In Fig. 6 we investigate the impact of the length of the clips on recall and precision as well as on execution times for clip search (see 2.3) and sequence identification (see 2.4). 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 clips. For Chart TV the recall exhibits an exception of this rule for clips of $1s$ duration. The precision behaves in the same way for both videos. This time, however, Sky Sports News shows an exception in the precision for $1s$ clips. 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 in (usually non-commercial) recurring sequences, which in turn affects precision negatively and leads to an increase in 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 clip length of 25 frames (corresponding to $1s$) seems to be an appropriate choice.
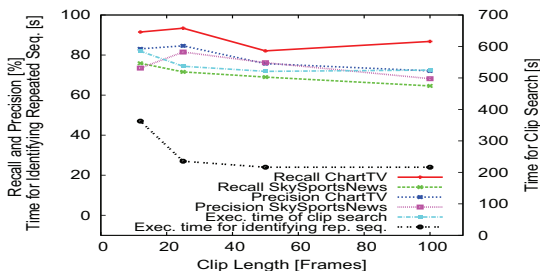

Figure 6: Performance and execution times in dependence on the clip durations.

Minimum Fraction of Matched Frames: We search for similar frames by look-ups in the hash table to determine candidate clip pairs. If the fraction of matched frames between two clips exceeds a threshold, their actual visual distance given in Eq. 3 is evaluated. If, however, it is below the threshold, both clips are rejected right away as being similar. Fig. 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 discarded. 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 and are stable over a wide range, diminishing on lightly for very small values. Lower threshold values lead to a higher number of falsely detected clips. 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 clip search for match ratios larger or equal 20%. However, a higher number of clips passing the similarity pre-filter has little consequences on the execution times for identifying repeating sequences. Taking into account performance as well as execution times, a threshold of 20% of matched frames is used in all subsequent experiments.
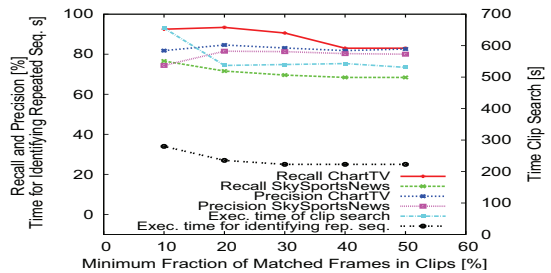

Figure 7: Performance and execution times in dependence on the minimum required fraction of matched frames.

Maximum Number of Hash Table Entries: As explained in subsection 2.3 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 length exceeds an upper limit of entries should be disregarded. An excessive number of entries is either 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 or talking heads during talk shows. In both cases we would create matching clip candidates, which are discarded by the subsequent processing steps. Thus, a knockout criterion was introduced to keep evaluation times low for videos stream with long self-similar or many generic image sequences.

As revealed in Fig. 8, the influence of the investigated parameter on retrieval performance is quite different for our test videos due to dissimilar channel characteristics. Taking hash values with a greater number of entries into account increases 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 intro or outro. Both of these spots may vary in lengths and small details, but are repeated many times. Thus, the frames of the various variants of the intro/outros are either not specific enough or repeated very often. Consequently, recall can be improved by increasing the allowed maximum number of list entries for hash values to be considered in the search. Recall and precision for Chart TV are not affected by the variation of this parameter. However, the execution times for clip search clearly increases with the upper limit

on the number of entries per hash index: a greater number of matched frames increases the number of required feature distance computations. According to our tests a maximum number of 100 entries per hash value is a good choice for a 2 hour slice (180,000 frames). Clearly, this value highly depends on the chosen image features, the applied hash function, and the duration of the mining task.
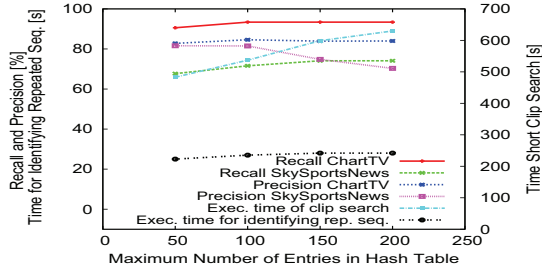

Figure 8: Performance and execution times in dependence on the upper limit of entries per index in the hash table. All values refer to 2 hour slices.

Minimum Length Sequences: When building repeating sequences S_j in subsection 2.4, we required a minimum sequences duration in order to declare a duplicate $\mathcal{D}$. In fact we required a minimum count n of clips as well as a minimum length $(\text{frame}_n^i - \text{frame}_1^i)$ before creating a duplicate sequence candidate discarding very short and accidentally similar short sequences. In Fig. 9 performance values and
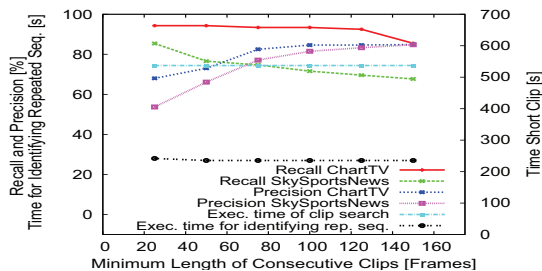

Figure 9: Performance and execution times in dependence on the minimum required sequences length for building duplicates.

execution times for different required sequence lengths are shown. The required minimum number n of clips was scaled accordingly. The recall of Chart TV is nearly constant until the minimum required length exceeds the shortest occurring commercials with a length of 125 frames (see Fig. 3), whereas 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 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.

Runtime Complexity: The most time consuming step is the computation of the GHs taking around $20min$ for each 2 hours segment on an Intel Xeon 2.33 GHz CPU including video decoding plus $10secs$ for generating the hash index table. Thus, the overall execution time for preparing recurring sequence detection is about 4 hours per 24 hours of video material. Another $27secs$ are needed to identify of all repeating sequences within in these 24 hours.

3.2 Detection Performance

We finally summarize the performance of our system in detecting unknown commercials via their repetitions over one day and the achievable improvements by extending the search to two days. We use the optimal parameters identified in subsection 3.1: we set the clip length to 25 frames, require 20% of matched frames for clips to be considered as possibly similar to each other, 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 use the combination of cut and feature vector based alignment and filter the result list 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.

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

Table 2: Recall and precision for our test videos.

Concentrating on the performance of finding repeated sequences recall values R_M and R_{MD} are of interest. R_M is the rate for detecting all recurring commercials, while R_{MD} concerns all recurring different commercials. The relationship between both values depends on the number of repetitions of each spot. In practice, both recall 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 and R_D are of further interest. R captures the recall with respect to the detection of all occurring commercials N and R_D to all occurring different commercials N_D. 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 lower. According to subsection 3.1 the chosen values are not optimal for this kind of broadcast. 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 relates to all detected repeating

video sequences, while P_{MD} focuses on repeated different sequences. Again, the relation between both values depends on the number of detected repetitions of commercials and non-commercials. If commercials are more often repeated, P_M is higher. Note that precision drops for a longer search time due to the repetition of non-commercial sequences.

4. RELATED WORK

Since we focus 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 ad breaks. A related algorithm was introduced by Albiol [1]. Content related characteristics of ads such as high cut rate or their separation by monochrome (mostly black) frames were discussed by Lienhart [10] and Dimitrova [2]. A combination of channel logo detection and commercial characteristics was applied by Glasberg [7].

A 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 [12] introduced a repeated video sequence detection method, relying on the temporal segmentation of the input video. Clip candidates were compared by their length and the number of similar frames using a color-based distance measure. Pua's algorithm finished with a collection of shots, which were either classified as unique or repeated. Gauch and Shivadas extended this algorithm for identifying new commercials [5]. They merged adjacent repeated shots; a subsequent classifier labeled found sequences as commercials or non-commercials. The algorithm of Duygulu relied on shot boundary detection, too [4]. However they restrict the similarity comparison to key frames of detected shots. Yuan avoided any shot detection [14]. They used small overlapping segments of 8 seconds and stored the summarized segment features in an index using local sensitive hashing. Next they constructed continuous paths of such similar small segments in order to identify repeated sequences in their original length.

5. CONCLUSION

The presented search algorithm is capable of detecting repeating video sequences in live-video streams at a fraction of its play duration. Thus multiple channels can be monitored 24/7 on a single computer. By combining the results of each 24-hour search over multiple weeks, a complete record of all commercials and repeated video clips can automatically be generated. The most time and space consuming components are the video decoding (in case of offline video) as well as the computation and 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 and is thus universally applicable to any kind of data, not just video data. We have tested the proposed method for searching for unknown commercials on different TV channels. A recall over 90% of repeated commercial spots was achieved, which on average covered about 80% of all commercials broadcast each day.

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 Int. Conf. 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, 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.