Skimming Rushes Video Using Retake Detection

Werner Bailer, Felix Lee and Georg Thallinger
JOANNEUM RESEARCH, Institute of Information Systems & Information Management
Steyrergasse 17
8010 Graz, Austria
{firstname.lastname}@joanneum.at

ABSTRACT

In audiovisual post-production users are confronted with large amounts of redundant unedited raw material, called rushes. Viewing and organizing this material is a crucial but time consuming task. This paper describes an approach for creating skimmed versions of the rushes video based on the elimination of unusable content and clustering of takes. Typically multiple, but slightly differing takes of the same scene can be found in the rushes video. We propose a method for clustering takes of one scene shot from the same camera position. It uses a variant of the LCSS algorithm to find matching subsequences in sequences of extracted features from the source video. The approach is evaluated by two subjective measures for the quality of the skim and by measuring the overlap between items found in the source video and the skim.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Abstracting methods*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

video skim, retake, rushes, summarization, content abstraction, TRECVID

1. INTRODUCTION

The problem of efficiently exploring collections of videos occurs in applications where users have to deal with large amounts of content. Viewing complete videos in order to locate relevant segments is prohibitive even for relatively small content sets due to the required time for viewing and the amount of data that needs to be transferred. Multimedia content abstraction methods are complementary to classical search and retrieval approaches, as they allow exploration of an unknown content set, without the requirement to specify a query in advance. This is relevant in cases where only few metadata are available for the content set, and where the user does not know what to expect in the content set, so that she is not able to formulate a query. A *video skim* [9], i.e. a short synopsis of the original video, created by sequential playback of extracted significant video and audio information, is a form of content abstraction that enables a user to view the content of a video collection more efficiently.

Our work is motivated by the problem of content management in digital cinema and video post-production. After shooting, a large amount of raw material ("rushes") must be organized and viewed in order to identify the small fraction that will be used in the final content. We focus on two aspects of this problem: Firstly and most importantly, a lot of the material is redundant as there are multiple takes of one scene. Organizing the material by takes of the same scene is a kind of alignment of the content to the script, with each group of takes representing the alternative options. The second aspect is the unusable content in each of the takes, usually at the beginning and end of the scene, such as technical setup, the director giving last instructions etc.

The aim of this work is thus to create video skims of a collection of rushes video. The redundancy is reduced by clustering the takes of the same scene and selecting relevant clips from each of the scenes.

The rest of this paper is organized as follows: Section 2 gives an overview of the skim creation process; Sections 3 and 4 present the details of the retake detection and content selection steps of this process. The results of the evaluation within the TRECVID 2007 BBC rushes summarization task [7] are discussed in Section 5 and Section 6 concludes and gives an outlook on future work.

2. SKIM CREATION PROCESS

As shown in Figure 1 the proposed video skimming process comprises five major steps: A set of content analysis algorithms are applied to extract visual features, which are used for the subsequent content selection and clustering steps. This includes the removal of unusable content (e.g. color bars), detection and clustering of retakes and the selection of representative video clips from the takes. Finally editing of the skim is performed.

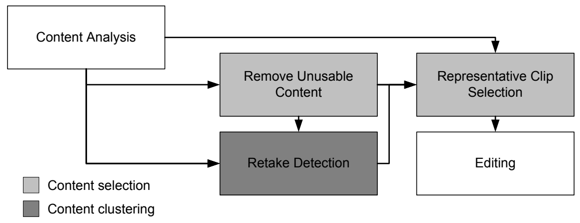

Figure 1: Overview of the skim creation process.

This skim creation process is an instantiation of the generic content abstraction process proposed in [1]. The core steps in the process are clustering and selection of content segments. The skimming process proposed in this paper applies two selection steps (removal of unusable content and selection of representative clips from the clusters) and one clustering step (detection and grouping of takes of the same scene). The editing of the skim represents the presentation step of the process. As the result is a video file, consumption reduces to viewing the created video.

In [1] a software framework for implementing the generic content abstraction process has also been presented, which has also been used to implement the skimming process. The indexing service used for ingesting metadata descriptions into the skimmin system is only slightly different than the one of the interactive browsing tool described in [1]. However, as the problem is different another summarization component has been implemented. The presentation components were reduced to a script controlling the editing operations.

2.1 Content Analysis

The content analysis module aims at extracting several low-level features from the uncompressed input video stream. In particular, this analysis includes shot boundary detection, extraction of key frames and their visual features, calculation of visual activity, detection of test patterns such as color bars and monochrome frames, and face detection.

Shot boundary detector (SBD) and test pattern detector split the entire video into smaller pieces which can be handled more easily. We do not allow the selected video segments to include shot boundaries or test patterns, since they are of no value in our context. As the BBC rushes data includes only hard cuts the detection of gradual transitions has been deliberately excluded. The SBD is implemented by calculating the pixel-wise frame differences between the current frame i and two following frames: $d_{i,1} = \sum_{x,y} |I_i(x,y) - I_{i+1}(x,y)|$, $\quad d_{i,2} = \sum_{x,y} |I_i(x,y) - I_{i+2}(x,y)|$, where $I(x,y)$ is the RGB pixel value at position (x,y) in the respective frame. In order to find the hard cut, we determine the frame position k where $d_{k,2}$ is significantly greater than $d_{k,1}$. If this criterion is fulfilled, then we assume a hard cut occurs between frame $k+1$ and $k+2$. This task is solved by using a linear support vector machine (SVM) implemented through LIBSVM [2]. To train the SVM classifier, ground truth from the TRECVID 2006 SBD task [8] has been used.

In order to facilitate sequence matching for retake detection, key frames are extracted by subsampling with a fixed rate. The MPEG-7 descriptors ColorLayout and EdgeHistogram [6] are calculated for each of the key frames.

The simplest way to obtain visual activity is again to sum up the pixel-wise differences between two consecutive frames and finally normalize by the number of pixels and channels. High visual activity indicates camera or strong object motion in the current take.

To determine if the current frame comprises only color bars or unsubstantial content (i.e. black frames), we first calculate the standard deviation of the RGB-pixel value in each pixel column and channel. When the maximum of the column's standard deviations is lower than a threshold (we set $th = 15$), the frame is labeled as unsubstantial. Note that this method utilizes the strong assumption that only standard vertical color bars appear in the content. This simple algorithm is sufficient in most cases. However, in the small part of the material that has been shot on film, color cards held in front of the camera are used for color calibration, which our algorithm does not detect.

The face detector is based on the algorithm developed by Viola and Jones [10] which is implemented in Intel's OpenCV[1] library. Facing the higher calculation complexity for face detection, only the key frames were analyzed. The face detection result forms a discrete function. To eliminate short-term false detections (i.e. sudden appearance or disappearance of faces) mathematical morphological closing and opening operators were applied to the sequence of face detection results.

2.2 Removing Unusable Content

Video segments comprising only color bars or black frames are definitely dispensable content which can be removed based on the vertical standard deviation measure calculated during content analysis. We also skip short shots (duration less than 10 seconds) and discard short content segments at the beginning and end of each shot.

2.3 Editing Videos

The final step of the process is producing the skim. Based on the selected clips that are determined as described in Section 4 an editing script is created, containing the following steps: (a) extraction of clips from the source content, (b) demultiplexing, (c) text insertion, (d) audio processing, (e) multiplexing and (f) concatenating the segments. The ffmpeg and mencoder[2] tools are used for the video related tasks, SoX[3] is used for audio processing. Text insertion is done in clips stemming from scenes for which more than one take has been identified (cf. Figure 2), displaying the number of alternative takes in the video. Audio processing consists of adding a short fade in and fade out effect to each of the clips in order to avoid artifacts at the clip boundaries. For clips shorter than 3.5 times the fade duration the audio signal is completely suppressed.

3. RETAKE DETECTION

Retake detection deals with identifying multiple takes of the same scene. The takes of one scene may be shot with different cameras. In this work, we restrict the problem to takes shot from the same or similar camera positions. The action in each of these takes will be similar, but not identical. Some of the takes may just contain parts of the

[1]`http://sourceforge.net/projects/opencvlibrary`
[2]`http://ffmpeg.mplayerhq.hu` and `http://www.mplayerhq.hu`
[3]`http://sox.sourceforge.net`

scene's action, starting or ending in the middle. There may be gaps and insertions in the action, the pace of some parts may be different, and actors or objects may be placed and move differently.

Thus we can define the problem as follows: Given a set of parts $P = \{p_k | k = 1..N\}$ of a video or a set of videos (where a part is a shot or a subshot), identify the set of scenes $S = \{s_1, \ldots, s_n\}$, where for each scene s_i there exists a set T_{s_i} of one or more takes. A take spans either a full part or a fraction of it. The takes in one set are similar in terms of a certain distance function (distance $< \theta_{sim}$) and sufficiently long (minimum length θ_{len}, i.e. a single similar frame is not a retake):

$$T_{s_i} = \{t_i | \text{dist}(t_i, t_j) \leq \theta_{sim} \wedge |t_i| > \theta_{len},\ i, j = 1 \ldots m\}.$$

3.1 Overview

The first step is to identify the parts p_k of the source video. As a starting point, shot boundary detection can be used to determine the set of shots. In many cases a shot contains one take, however, there are shots containing several takes. Ideally, shots should be split at the take boundaries, e.g. at the point where the clapper board is shown or when someone shouts "action". Due to the lack of classifiers for these events we currently split shots at short-term peak values of the visual activity that are sufficiently distant from shot boundaries. This will usually split the shot at the points where the clapper board is shown and removed, the camera is significantly moved. As we only split at high activities with a very short duration, most normal action – even fast one – does not cause a split. There are however false positives caused by abrupt strong movements of actors or objects.

The next step is to extract the sequence of features A_{p_k} for part k, consisting of elements $a_{p_k,i,j}$, where i indicates the feature and j the position in the sequence. In order to reduce the processing time and the length of the sequences to be matched, the set of features is only extracted every n^{th} frame. The maximum offset between two elements of the feature sequences to be matches is thus $n/2$ (in our experiments, we used $n = 10$). For the selected frames, we have extracted the MPEG-7 ColorLayout and EdgeHistogram descriptors as well as the visual activity (averaged over n frames).

In order to detect which parts contain retakes, pair-wise matching of the parts is performed. The LCSS algorithm described in Section 3.2 is applied to each pair $(p_u, p_v) \in P$, yielding a set of candidate subsequences C_{p_u,p_v}. As some of them may overlap, sequences contained in others are removed and overlapping matches are merged. Thus only the non-overlapping and non-adjacent subsequences C'_{p_u,p_v} remain as candidate sequences for each pair of parts. The next step is to cluster these candidate sequences into sets of similar takes. This is described in detail in Section 3.3.

Finally, for each of the takes t_i a sequence $R(t_i)$ of relevance values over time is calculated. The sampling rate of $R(t_i)$ is n like for the feature sequence. The value of each of the elements $r \in R(t_i)$ is determined as the number of matches of the corresponding element of the feature sequence a_{p_k} within the cluster to which the take is assigned. The relevance sequence thus helps to distinguish parts of a take which appear in all or most of the takes of the same scene from those that occur only in a single or few of them.

3.2 LCSS Algorithm

In order to determine the similarity of parts, we need to identify subsequences of these parts that are sufficiently long and similar, where gaps and insertions are acceptable. The problem is converted into identifying a similar subsequence in the sequences of features of the parts. We need to find an appropriate similarity measure between two such feature sequences. Evaluation of similarity measures for object trajectories [11] has shown that the Longest Common Subsequence (LCSS) model with the extension proposed in [5] provides the best performance. As the problem of measuring similarity between feature sequences is related, we have chosen this approach for our work.

The authors of [5] introduce two thresholds: a real number ϵ that defines the matching threshold between the non-discrete elements of the sequences and an integer δ that defines the maximum offset in the positions to be matched. In order to adapt the LCSS algorithm to our problem, we make the following modifications. As each element in the sequences is a multidimensional feature vector, we define a vector $\theta_{sim} = \{\epsilon_i\}$, that contains the matching thresholds for all the features. Similarly, we introduce a vector representing the relative weights of the features. The distance between two elements is given as the weighted sum of the feature distances. If the distance for feature k exceeds the threshold ϵ_i, the distance for this feature is set to 1.

The offset δ introduced in [5] is not relevant for our problem, as the matching subsequences can be anywhere in the parts. Instead, we introduce the maximum gap size γ of the subsequence. Note that the longest common subsequence does not necessarily fulfill the condition of having a gap $\leq \gamma$, so instead of just finding the LCSS we are interested in finding all sufficiently long sequences (length $> \theta_{len}$) with gaps no longer than γ.

Our implementation of the LCSS algorithm is based on the one described in [3]. We compute the table containing the lengths for the subsequence matches (the "c table"). Instead of just reading the one optimal solution (starting in the lower right corner of the table) we follow all paths that end at a length greater θ_{len}. The search is recursive and stops when either the start of the matching subsequence is reached or when the gap between two subsequent matches exceeds γ. The result is a set of (partly overlapping) matching subsequences (take candidates) $C_{p_u,p_v} = \{c_i\}$ between two parts p_u and p_v of the video.

3.3 Clustering Takes

Before the sequences can be clustered it is necessary to identify the set of distinct takes. Matches between a take candidate and take candidates from different parts may still contain partly or fully overlapping sequences. For example, the sequence [01:20:03;01:22:05] from part p_1 could match sequence [03:14:21;03:16:15] from part p_2, while sequence [03:15:00;03:16:18] from part p_2 matches sequence [10:50:24; 10:52:15] from part p_3. Thus the first step is to collate these sequences, replacing all sequences that are contained within other sequences by the longer ones. A tolerance of 15 frames is used when determining if one sequence is contained in another.

The remaining sequences represent takes t which are to be clustered into sets of takes T_{s_i} of the same scene s_i. Hierarchical clustering using a single-linkage algorithm [4] is performed. The distance function between two clusters T_{s_i}, T_{s_j}

is based on the longest common subsequence (calculated as described above) between two takes. If both of the clusters have more than one element, the cluster distance is set to the maximum value. This constraint avoids that similar takes from different scenes are added to the cluster in favour of less similar takes of the same scene. Clustering stops when the cutoff similarity is reached, which is given by the minimum length θ_{len} of a matching subsequence. The result of clustering is the set of scenes S.

4. CONTENT SELECTION

The results of retake detection together with the low level features extracted by content analysis build the base for selecting representative clips. Each frame obtains a weight which is calculated as follows: For each frame, the weight is initialized from the relevance values obtained in the retake detection. The relevance values indicate the frequency of occurrence in other takes within the cluster representing a scene. We select the take with maximum average weight as representative of the cluster and select clips from this take. In most cases, one clip is selected from each take cluster. However, if the take cluster spans a long fraction of the playtime of the video and high relevance weigths are found, more than one clip is used. In order to incorporate the visual activity, the initial weights are simply multiplied by the visual activity values. To incorporate the face detection results, the weights are reduced by 50% if no face is present. The initial run with this weighting method has shown a drawback, namely that the clapper board causing a high visual activity has often been selected. Since clapper boards occur often at the beginning of a take [4], the weight is additionally scaled proportionally to the distance between the current frame and the start of the take. In this way, one can effectively avoid clapper boards being selected, with the drawback that content from the start of the scene is less likely to be included in the skim, if no clapper board is present.

Once the weights are calculated, clips for the final skim can be selected. The skim must not exceed a length of 4% of the original video [7]. Since there is no overlap between take clusters, we can treat each cluster separately and select at most 4% from the sequence covered by takes of that cluster. In order to make the skim more comfortable to watch, we retain the video playing speed and do not insert clips of less than one second length. The process starts with determining the maximum weighted frame of the take that has been found to be representative for the cluster. Using this frame position as center, a clip around this position is extracted, taking the take boundaries into account.

5. EVALUATION

The evaluation [7] of the skims produced with the approach presented here shows average results in terms of the inclusion rate of items found in the original content and quite good results for the subjective measures easy understandability and few duplicate content. The mean inclusion rate is 0.46, the median 0.47. The mean of the score for easy understandability is 3.57 (second rank), the median 3.67 (first rank), the mean of the score for few duplicates is 3.78 (6th rank), the median 3.67 (6th rank). The reasons for the good scores for easy understandability is that very short shots are discarded and that the clip selection algorithm tries to select longer continuous clips, which makes the skim easier to comprehend. The relatively good score for having few duplicates results from clustering similar takes, which should eliminate most of them.

It is interesting to look at those skims with the minimum and maximum scores for each of the measures. The one with the minimum inclusion score (0.08) is a case where many of the assumptions in our approach did not hold. Most of the content of this video consists of material shot at a theme park containing a lot of visual activity and very strong motion blur, but no repetitions. The motion blur also caused many false positives in our shot boundary detection, resulting in a number of short shots that have been discarded. For the skim just the first scene was taken, which explains the low inclusion value, but the maximum score in terms of understandability and absence of duplicates (both 5.00).

The skim with the maximum inclusion score (0.76) contains dialog scenes in a garden and it also scores high in terms of understandability and has exactly the median score of the duplicate measure. This shows that high values in the usability measures are not generally related to low inclusion values, also statistical analysis shows no significant correlation between the different measures. The skim with the minimum score for understandability (2.33) is based on a source video that contains takes which are not in the order of action (it has to be noted that no group scored better than 3.00 on this video). Our clip selection chooses some clips showing setup or discussion among the actors instead of the real action, which contributes to the bad result.

The skims with bad scores for the duplicate measure (2.33) share some commonalities. There are some cases where retake detection failed, so that clips from different takes of the same scene have been included. In many cases these skims contain takes of the same scene but shown from clearly different camera positions or containing different camera motion. For a viewer who is just interested in what happens in the video this is clearly redundant, while for someone working in post production it is very valuable to know that the scene has been shot from different camera positions. Apart from the one mentioned above, there are two more skims with a maximum score (5.00) for the duplicate measure. In one case the original video contains no duplicate shots, so that it is trivial to avoid duplicates in the skim. In the other case it is not clearly explicable, why this skim has a higher value than other skims that also contain no real duplicates but a comparable amount of similar scenes.

The processing time for creating the skims is slightly less than twice the duration of the input video (on a Pentium D 3GHz machine). Half of this time is spent for running the content analysis tools. Removal of unusable content, retake detection and representative clip selection work on the metadata produced by the content analysis tools, thus all these operations together take only about 3% of the runtime. The rest of the time is spent creating the output video. It has to be noted that the most time consuming part of this step is the text insertion, as this requires decoding, frame-wise processing and re-encoding.

[4]There are cases where the clapper board occurs at the end, but as we split shots at the points where are clapper boards are likely to occur, this results in a short subshot at the end which is discarded.

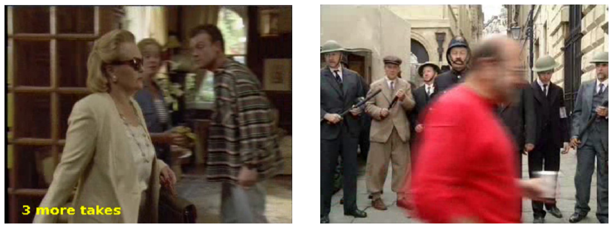

Figure 2: Two screen shots from the created skims. Left image: text insert showing number of alternative takes of the scene. Right image: clip taken from action that is not part of the scripted scene.

6. CONCLUSION AND FUTURE WORK

We have presented a system for skimming of rushes video targeted at post production environments. The skimming process is an instantiation of the generic multimedia content abstraction process presented in [1]. The approach is based on reducing redundancy by clustering multiple takes of the same scene shot from the same or a similar camera position and removing unusable content. Clustering by takes of the same scene is one of the most important organization criteria for content in post production, as this allows alignment of the content to the script. From each cluster, clips of one take are selected based on a weighting strategy. In the skim, a text overlay shows the number of additional takes of the same scene.

The evaluation shows that the approach performs well in terms of the usability measures, but that the coverage of the original video in the skim must be improved. We still need to analyze the reasons for the low inclusion scores on some of the videos, especially needs be survey whether there are any significant differences in terms of inclusion score between the classes of items (persons, events, camera movements) in the ground truth. Also the correlation between the relevance weights and the objects and events in the ground truth should be investigated.

There are several components of the skimming process which we intend to improve in the future. Shots are currently split into subshots simply by detecting activity outliers. This could be improved by using features more targeted at the usual production scenario, such as training a classifier for clapper boards or analyzing the audio signal (long silence, clapper board sound, spotting the word "action"). A detector for clapper boards would also help to avoid that they turn up in the skim later on. Retake detection could be improved by using more advanced features as input for sequence matching, such as localized motion activity, object trajectory or alignment of words in the speech to text transcript. As a further step, it would also be interesting to have a second clustering step grouping take clusters showing the same action, but shot from different camera positions.

Once clustering is done, we have found it quite challenging to decide which part of a take represents the actual action of the scene and which does not. For example, in one case someone from the production team walks through the scene with a cup of coffee in his hand immediately before the action starts (see right image in Figure 2) – how to decide that he is not an actor and this is not part of the scene? This is a complex problem on its own and needs more investigation. For presentation it would also be useful to visualize the differences between the takes within the same cluster, e.g. which cover the full action or which start or end in the middle.

7. ACKNOWLEDGMENTS

The authors would like to thank Harald Stiegler, Christian Schober, Andreas Kriechbaum and Werner Haas for their support and feedback. The research leading to this paper was partially supported by the European Commission under the contracts FP6-045032, "Search Environments for Media – SEMEDIA" (`http://www.semedia.org`) and FP6-027026, "Knowledge Space of semantic inference for automatic annotation and retrieval of multimedia content – K-Space" (`http://www.k-space.eu`).

8. REFERENCES

[1] W. Bailer and G. Thallinger. A framework for multimedia content abstraction and its application to rushes exploration. In *Proceedings of ACM International Conference on Image and Video Retrieval*, Amsterdam, NL, Jul. 2007.

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[5] D. G. Michail Vlachos, George Kollios. Discovering similar multidimensional trajectories. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, pages 673–684, San Jose, CA, USA, 2002. IEEE Computer Society.

[6] MPEG-7. Multimedia Content Description Interface—Part 3: Visual. ISO/IEC 15938, 2001.

[7] P. Over, A. F. Smeaton, and P. Kelly. The TRECVID 2007 BBC rushes summarization evaluation pilot. In *Proceedings of the TRECVID Workshop on Video Summarization (TVS'07)*, pages 1–15, New York, NY, September 2007. ACM Press.

[8] A. F. Smeaton and P. Over. TRECVID 2006: Shot boundary detection task overview. In *Proceedings of the TRECVID Workshop*, November 2006.

[9] M. A. Smith and T. Kanade. Video skimming for quick browsing based on audio and image characterization. Technical Report CMU-CS-95-186, Carnegie Mellon University, July 1995.

[10] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[11] Z. Zhang, K. Huang, and T. Tan. Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 1135–1138, Washington, DC, USA, 2006. IEEE Computer Society.