Detecting and Clustering Multiple Takes of One Scene

Werner Bailer, Felix Lee, and Georg Thallinger

JOANNEUM RESEARCH Forschungsgesellschaft mbH,
Institute of Information Systems & Information Management,
Steyrergasse 17, 8010 Graz, Austria
firstName.lastName@joanneum.at

Abstract. In applications such as video post-production users are confronted with large amounts of redundant unedited raw material, called rushes. Viewing and organizing this material are crucial but time consuming tasks. Typically multiple but slightly different takes of the same scene can be found in the rushes video. We propose a method for detecting and clustering takes of one scene shot from the same or very similar camera positions. It uses a variant of the LCSS algorithm to find matching subsequences in sequences of visual features extracted from the source video. Hierarchical clustering is used to group the takes of one scene. The approach is evaluated in terms of correctly assigned takes using manually annotated ground truth.

1 Introduction

In film and video production usually large amounts of raw material ("rushes") are shot and only a small fraction of this material is used in the final edited content. The reason for shooting that amount of material is that the same scene is often shot from different camera positions and several alternative takes for each of them are recorded, partly because of mistakes of the actors or technical failures, partly to experiment with different artistic options. The action performed in each of these takes is similar, but not identical, e.g. has omissions and insertions.

The result of this practice in production is that users dealing with rushes have to handle large amounts of audiovisual material which makes viewing and navigation difficult. Our work is motivated by two application areas where this problem exists. One is post-production of audiovisual content, where editors need to view and organize the material in order to select the best takes to be used (the ratio between the playtime of the rushes and that of the edited content is often 30:1). The other application area is documentation of audiovisual archives. While edited content is well documented (at least in larger broadcast archives) and thus reusable, raw material is in most cases not documented due to the amount of material and its redundancy [7]. In both applications, identifying the takes belonging to the same scene and grouping them could significantly increase the efficiency of the work.

In this paper we deal with the problem of identifying and grouping multiple takes of the same scene. In general the different takes of one scene may be shot

S. Satoh, F. Nack, and M. Etoh (Eds.): MMM 2008, LNCS 4903, pp. 80–89, 2008.

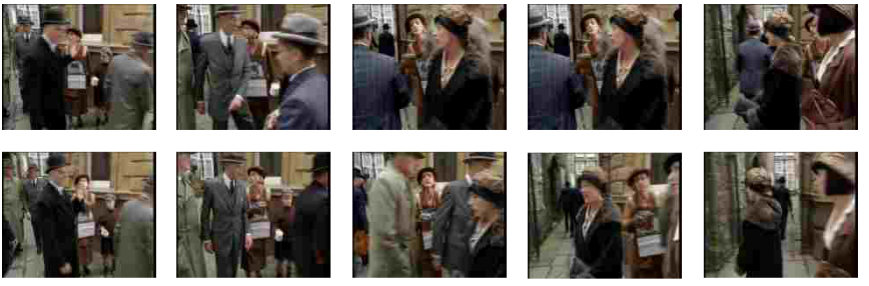

Fig. 1. Example of takes of the same scene: The upper row shows keyframes of the first take, the lower of the second take of one scene. Images taken from BBC 2007 rushes video data set.

from different camera positions. We restrict our work to the problem of takes shot from the same or very similar camera positions. Nonetheless the content between the different takes may vary, as actors and objects move differently or there are insertions and omissions in the action being performed. In addition there are takes that stop earlier (mostly due to mistakes) or start in the middle of the scene ("pickups"). The algorithm for detecting and grouping retakes has to deal with this variability.

Related research areas are the detection of similar video sequences and near-duplicates. There are many approaches for dealing with similarity matching of video clips, many of them applying still image features for clustering key frames of video sequences. Some of these approaches use more video oriented features such as sequences of key frames and camera motion [19] or motion trajectories [4]. However, while the feature extraction and matching approaches proposed in these papers are relevant for our problem, the implementation of similarity matching approaches usually finds all similar content, including takes of other similar scenes and not just those of the same scene. If clustering of takes has to be performed on a large set of material, similarity matching can be used as a preprocessing step to find candidates of related takes.

The most prominent application for near-duplicate detection is finding illegal copies of video content (cf. [11,10]). A related problem is the identification of known unwanted content in public access video databases [6]. These applications are based on the following assumptions: (i) the actual content of the videos to be matched is identical, (ii) partial matches need to be identified and (iii) the system is robust against a number of distortions, such as changes of sampling parameters, noise, encoding artifacts, cropping, change of aspect ratio etc. The first assumption does not hold for take clustering, while robustness to distortions is only necessary to a very limited degree in our application, as the content to be matched is captured and processed under very similar conditions. Another application of near-duplicate detection is topic tracking of news stories. Some approaches work on a story level and use features such as speech transcripts that are often not available in rushes [12]. Other approaches work on matching

sequences of key frames, tolerating gaps and insertions [9], which comes closer to the requirements of our application.

The rest of this paper is organized as follows: Section 2 presents a novel algorithm for identifying similar takes based on the Longest Common Subsequence (LCSS) model [5] and clustering them. In Section 3 we describe the evaluation methodology used and present the results of evaluation on the TRECVID 2007 BBC rushes data set [15]. Section 4 concludes the discussion and outlines future work.

2 Algorithm for Clustering of Takes

2.1 Problem Formulation

We can describe the problem of detecting multiple takes of the same scene as follows: Let $P = \{p_k | k = 1..N\}$ be the set of parts of a video or a set of videos, where a part is a shot or a subshot. Identify the set S of scenes, where for each scene $s \in S$ there exists a set $T_s = \{t_1, \ldots, t_M\}$ of one or more takes. Each take spans a full part p or a fraction of it: $t_i \subseteq p_k, t_i \in T_s, p_k \in P$. Each take can only be assigned to one scene, i.e. $T_{s_i} \cap T_{s_j} = \{\}, \forall s_i, s_j \in S, i \neq j$.

The takes in one set are similar in terms of a certain distance function (using the threshold θ_{sim} for the maximum distance) and sufficiently long (minimum length θ_{len}, thus e.g. a few similar frames do not constitute a take):

$$T_s = \{t_i | \text{dist}(t_i, t_j) \leq \theta_{sim} \wedge |t_i| > \theta_{len},\ i, j = 1 \ldots M\},$$

The distance function used is described in detail in Section 2.4.

2.2 Algorithm Overview

The first step is to split the input video into parts p and to extract the sequence of features A_{p_k} for each part p_k. This feature sequence consists of elements $a^i_{p_k,j}$, where i indicates the feature and j the position in the sequence. The segmentation of the input video into parts and the extraction of features is described in Section 2.3.

In order to detect which parts contain takes of the same scene, pair-wise matching of the parts is performed. The LCSS algorithm described in Section 2.4 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-connected 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 2.5.

Finally, for each of the takes a sequence of relevance values over time is calculated. The relevance values are determined from the number of matches between a take and other takes in the same scene. This information can be used to select representative clips for the takes of a scene.

2.3 Feature Extraction

The first task is to segment the input video into parts. As a starting point, shot boundary detection is used to determine the set of shots, which is used as the initial set of parts. Only hard cut detection is performed: A linear SVM is used taking frame differences of three consecutive frames as input. The SVM is implemented through LIBSVM [3]. To train the SVM classifier, ground truth from the TRECVID 2006 shot boundary detection task [16] has been used.

For the set of initial parts, a feature sequence A_{p_k} is extracted for each part p_k as follows. 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 (in our experiments, we use $n = 10$). The maximum offset between two elements of the feature sequences to be matched is thus $n/2$ frames. For these frames, the MPEG-7 descriptors ColorLayout and EdgeHistogram [13] are extracted. From the ColorLayout descriptor, the DC and the first two AC coefficients of each channel are used. The visual activity is calculated as the pixel difference of two consecutive frames and averaged over n frames. An element of the feature sequence is thus given as:

$a_{p_k,j}$=$(ColorLayout_1,...,ColorLayout_9, EdgeHistogram_1,...,EdgeHistogram_{80}, VisualActivity)^T$.

In many cases a shot contains exactly 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 split shots at short-term peak values of the visual activity – computed during feature extraction – that are sufficiently distant from shot boundaries. The result is the final set of parts that is used as input for the subsequent steps of the algorithm. Our approach is intended to split the shot at the points where the clapper board is shown and removed, the camera is significantly moved or where production staff moves close in front of the camera. 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.

2.4 LCSS Algorithm

In order to determine the similarity of parts, we need to identify subsequences of these parts that are sufficiently long and similar, accepting gaps and insertions. The problem is converted into identifying similar subsequences in the feature sequences of the parts. We need to find an appropriate similarity measure between two such feature sequences. Evaluation of similarity measures for object trajectories [18] has shown that the extension of the Longest Common Subsequence (LCSS) model proposed in [17] provides the best performance.

The authors of [17] 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 of the sequence is a multidimensional feature vector, we define a vector $\theta_{sim} = \{\epsilon_i\}$, that contains the matching thresholds for all features. Similarly, we introduce a vector $W = \{w_i\}$ representing the relative weights ($\sum_i w_i = 1$) of the features. The distance between two elements at positions j_1 and j_2 of the sequences A_{p_u} and A_{p_v} is then given as

$$\mathrm{dist}(a_{p_u,j_1}, a_{p_v,j_2}) = \sum_i w_i \cdot \mathrm{E}(a^i_{p_u,j_1}, a^i_{p_v,j_2}, \epsilon_i),$$

$$\mathrm{E}(a^i_{p_u,j_1}, a^i_{p_v,j_2}, \epsilon_i) = \begin{cases} 1, \text{ if } \mathrm{dist}_i(a^i_{p_u,j_1}, a^i_{p_v,j_2}) > \epsilon_i; \\ \mathrm{dist}_i(a^i_{p_u,j_1}, a^i_{p_v,j_2}), \text{ otherwise,} \end{cases}$$

where dist_i is the feature specific distance function. The distance functions for ColorLayout and EdgeHistogram are those proposed in [14], the distance for visual activity is the L1-norm. The offset δ introduced in [17] 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 only gaps $\leq \gamma$, 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 [5]. 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 than θ_{len} (for an example see Figure 2). 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.

2.5 Clustering Takes

Before the takes can be clustered it is necessary to identify the set of distinct sequences. 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 a sequence is contained within another. In our example the matching sequence in p_2 is [03:14:21 – 03:16:18].

The remaining sequences represent takes t which are to be clustered into sets of takes T_s of the same scene s. Hierarchical clustering using a single-linkage

	2.3	4.7	1.2	3.3	2.2	1.4
1.3	0	0	1	1	1	1
2.5	**1**	**1**	**1**	1	2	2
3.4	0	1	1	**2**	2	2
2.4	1	1	1	2	**3**	3
4.6	1	2	2	2	**3**	3
1.5	1	2	3	3	3	**4**
2.6	1	2	3	3	4	**4**

Fig. 2. Simplified example of the c table of LCSS matching (adopted from [5]). The values of the one dimensional feature sequences (shown at the top and on the left) are matched with thresholds $\theta_{sim} = 0.5$ and $\theta_{len} = 3$. The LCSS algorithm yields the sequence shown with emphasized border as the longest match. However, if we require a maximum gap $\gamma \leq 1$, this sequence is not a valid result, as there is a gap of two between the first two matching elements. Instead the sequences shown in gray are considered, which are results of the same length that additionally satisfy the gap constraint.

algorithm [8] is performed. The distance function between two clusters T_{s_i} and T_{s_j} is based on the longest common subsequence between their takes:

$$d(T_{s_i}, T_{s_j}) = \begin{cases} 1, \text{ if } |T_{s_i}| > 1 \wedge |T_{s_j}| > 1; \\ 1 - \max\limits_{t \in T_{s_i}, t' \in T_{s_j}} \mathrm{LCSS}_0(t, t'), \text{ otherwise,} \end{cases}$$

where LCSS_0 is the LCSS normalized by the lengths of the parts. This distance function prevents takes from distinct but similar scenes being clustered before 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.

3 Evaluation

3.1 Data Set and Ground Truth

The proposed algorithm has been evaluated on a subset of the TRECVID 2007 BBC rushes test data set [15]. The subset consists of six randomly selected videos out of this data set (in total 3 hours). For this subset ground truth has been manually annotated by identifying the set of scenes and the takes of each scene. All takes of a scene have been shot from the same camera position. No discrimination between complete and partial takes has been made. Shots of the video containing only test patters such as color bars or monochrome frames have been excluded from the test data.

The following parameters are used for evaluation. The minimum length θ_{len} is 30 frames, the gap size γ is 7 frames (the videos in the test have a frame rate of 25). For similarity matching the following thresholds and weights are used

(the sequence of the features is ColorLayout, EdgeHistogram, VisualActivity): $\theta_{sim} = (0.03, 0.03, 0.03)^T,\ W = (0.5, 0.2, 0.3)^T$.

3.2 Methodology

For evaluation the number of takes correctly and falsely assigned to one scene (including partial takes) is counted. The exact temporal extent of the identified takes as well as the alignment of partial takes is not taken into account. Firstly this would have enormously increased the effort for creating the ground truth and secondly the correct temporal extent is difficult to define in many cases.

The input to the evaluation for one video are the set of scenes S' as defined by the ground truth and the set of scenes S resulting from the algorithm. As the sets may be defined differently and may also have different order, the first step is to align the scenes. This is done by assigning each result scene s to that scene s' of the ground truth, for which the overlap between the takes of the scenes is maximal:

$$\text{matching\_scene}(s, S') = \underset{s' \in S'}{\text{argmax}} \sum_i \sum_j \text{overlap}(t_i, t'_j),\ t_i \in T_s,\ t'_j \in T'_s$$

where overlap calculates the temporal overlap between the takes in frames. Additionally a scene from the ground truth s' may only be assigned to at most one result scene s. After this step there may be unassigned scenes: the takes of an unassigned scene s are counted as false positives, the takes of an unassigned scene s' are counted as false negatives.

Once the scenes have been assigned, the number of temporally overlapping takes is counted. If a take of the ground truth overlaps with more than one take of the result, only a single overlap is counted. From the sum of correct overlaps of all clusters, precision and recall are calculated as follows:

$$precision = \frac{\sum_{s \in S} \text{nr\_overlaps}(s, \text{matching\_scene}(s, S'))}{\sum_{s \in S} |s|}$$

$$recall = \frac{\sum_{s \in S} \text{nr\_overlaps}(s, \text{matching\_scene}(s, S'))}{\sum_{s' \in S'} |s'|}$$

3.3 Results

Table 1 shows the evaluation results on the test set. In general the results are promising. In all but one case, precision is equal to or higher than recall. This means that the similarity between the feature sequences of takes (controlled by the threshold θ_{sim}) has been quite strictly enforced. As a result, takes in which the performed action diverges more have been grouped into separate scenes.

When analyzing the differences between the videos in the test set, it can be seen that the precision values are more uniformly distributed than the recall values (also supported by the respective differences between the mean and the median of the two measures).

Table 1. Number of scenes and takes in the test set and precision and recall results

Video	Nr. of scenes		Nr. of takes		Precision	Recall
	ground truth	detected	ground truth	detected		
MRS07063	6	4	26	17	0.7647	0.5000
MRS025913	8	9	28	28	0.8571	0.8571
MRS044731	7	9	34	31	0.5484	0.5000
MRS144760	6	5	24	26	0.8077	0.8750
MRS157475	8	8	36	32	0.6875	0.5903
MS216210	7	10	26	21	0.6190	0.5000
Mean	7.00	7.50	29.00	25.83	0.7141	0.6405
Median	7.00	8.50	27.00	27.00	0.7261	0.5556

The three videos with the lowest recall scores have similar kind of content: They contain dialog scenes with little action and all scenes have been shot in the same room. Thus the difference in visual appearance and activity between the scenes is very small (sometimes smaller than within takes of the same scene), so that takes are falsely assigned. Two of these three videos also have the lowest precision scores.

The noticeable difference between the two videos with the highest recall scores and the others is that the correct or a slightly higher number of takes has been found in these videos. This means that the lower recall scores are caused by mistakes in the segmentation of the input video into parts. There are parts that contain more than one take. Although these parts are correctly clustered, they are treated as single takes, thus missing the others contained in the same segment.

4 Conclusion and Future Work

We have presented an approach for identifying multiple takes of the same scene shot from the same or similar camera positions. The approach is based on matching sequences of extracted features (color and texture features of regularly sampled frames, visual activity) with a novel variant of the LCSS algorithm. Matching sequences are considered as take candidates and clustered by similarity in order to group them by scene.

The evaluation of our algorithm shows in general quite encouraging results, with precision being clearly better than recall. The lower recall scores are in many cases caused by insufficient segmentation of the input video, i.e. several takes are treated as one part. The strategy for splitting shots into potential takes 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”). This could improve the recall score in cases where sequence matching already performs well.

Another group of videos with lower precision and recall values are those containing a number of scenes with little action and shot at the same location

(e.g. several dialog scenes in one room). The problem here is that takes of different scenes are more similar in terms of the visual features we are using than takes from the following or previous ones. As a consequence, takes are incorrectly assigned, thus reducing both precision and recall scores. A possible solution would be to use more advanced features as input for sequence matching, such as localized motion activity, object trajectories 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.

Detection and clustering of multiple takes of one scene is a useful tool in post-production and audiovisual archiving. An earlier version of the approach presented here has been used for creating skims of rushes video by eliminating the redundancy stemming from retakes [1]. It will also be integrated as a cluster criterion in the interactive video browsing tool described in [2].

Acknowledgments

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`), IST-2-511316-IP, "IP-RACINE – Integrated Project Research Area Cinema" (http://www.ipracine.org) and FP6-027026, "Knowledge Space of semantic inference for automatic annotation and retrieval of multimedia content – K-Space" (`http://www.k-space.eu`).

BBC 2007 Rushes video is copyrighted. The BBC 2007 Rushes video used in this work is provided for research purposes by the BBC through the TREC Information Retrieval Research Collection.

References

1. Bailer, W., Lee, F., Thallinger, G.: Skimming rushes video using retake detection. In: TVS 2007. Proceedings of the TRECVID Workshop on Video Summarization, pp. 60–64. ACM Press, New York (September 2007)
2. Bailer, W., Thallinger, G.: 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 (July 2007)
3. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines, Software (2001), `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
4. Chang, S.-F., Chen, W., Meng, H., Sundaram, H., Zhong, D.: VideoQ: an automated content based video search system using visual cues. In: MULTIMEDIA 1997: Proceedings of the fifth ACM international conference on Multimedia, pp. 313–324. ACM Press, New York (1997)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. The MIT Press, Cambridge (2001)
6. Covell, M., Baluja, S., Fink, M.: Advertisement detection and replacement using acoustic and visual repetition. In: IEEE Workshop on Multimedia Signal Processing, pp. 461–466 (October 2006)

7. Delaney, B., Hoomans, B.: Preservation and Digitisation Plans: Overview and Analysis, PrestoSpace Deliverable 2.1 User Requirements Final Report (2004), http://www.prestospace.org/project/deliverables/D2-1_User_Requirements_Final_Report.pdf
8. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, Chichester (2000)
9. Duygulu, P., Pan, J.-Y., Forsyth, D.A.: Towards auto-documentary: tracking the evolution of news stories. In: MULTIMEDIA 2004: Proceedings of the 12th annual ACM international conference on Multimedia, pp. 820–827. ACM Press, New York (2004)
10. Hampapur, A., Bolle, R.M.: Comparison of distance measures for video copy detection. In: IEEE International Conference on Multimedia and Expo, pp. 737–740 (August 2001)
11. Hampapur, A., Hyun, K., Bolle, R.M.: In: Yeung, M.M., Li, C.-S., Lienhart, R.W. (eds.) Storage and Retrieval for Media Databases 2002. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, vol. 4676, pp. 194–201 (2001)
12. Hsu, W., Chang, S.-F.: Topic tracking across broadcast news videos with visual duplicates and semantic concepts. In: International Conference on Image Processing (ICIP) (October 2006)
13. MPEG-7. Information Technology—Multimedia Content Description Interface: Part 3: Visual. ISO/IEC 15938-3 (2001)
14. MPEG-7. Information Technology—Multimedia Content Description Interface: Part 8: Extraction and Use of MPEG-7 Descriptions. ISO/IEC 15938-8 (2001)
15. Over, P., Smeaton, A.F., Kelly, P.: The TRECVID 2007 BBC rushes summarization evaluation pilot. In: TVS 2007. Proceedings of the TRECVID Workshop on Video Summarization, pp. 1–15. ACM Press, New York (September 2007)
16. Alan, F., Smeaton, A.F., Over, P.: TRECVID 2006: Shot boundary detection task overview. In: Proceedings of the TRECVID Workshop (November 2006)
17. Vlachos, M., Kollios, G., Gunopoulos, D.: Discovering similar multidimensional trajectories. In: ICDE 2002: Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, pp. 673–684. IEEE Computer Society, Washington DC (2002)
18. Zhang, Z., Huang, K., Tan, T.: Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In: ICPR 2006: Proceedings of the 18th International Conference on Pattern Recognition, pp. 1135–1138. IEEE Computer Society, Washington, DC, USA (2006)
19. Zhu, X., Elmagarmid, A., Xue, X., Wu, L., Catlin, A.: InsightVideo: toward hierarchical video content organization for efficient browsing, summarization and retrieval. IEEE Transactions on Multimedia 7(4), 648–666 (2005)