Noname manuscript No.
(will be inserted by the editor)

Rushes Video Summarization and Evaluation

Dumont Emilie · Mérialdo Bernard

Received: date / Accepted: date

Abstract During the post-production stage of film making, the film editor is faced with large amounts of unedited raw material, called rushes. Developing tools to view and organize this material is an important component of video processing. This paper describes an approach for summarizing rushes video based on the detection of repetitive sequences, using a variant of the Smith-Waterman algorithm to find matching subsequences. We rely on the evaluation methodology that has been introduced in the TRECVID BBC Rushes Summarization Task. We propose an automation of the manual TRECVID evaluation using machine learning techniques to train an automatic assessor. We compare the automatic assessor evaluation to the evaluations provided by the TRECVID manual assessors.

Keywords Video Summarization · Rushes Video · TRECVID

1 Introduction

The technology of digital video is making rapid progress and powerful technologies now exist to create, play, store and transmit video data. Still, the analysis of video content remains an open and active research challenge. In this paper, we focus on video film making tools. Film making is a long process, starting from an initial story idea, followed by scriptwriting, shooting, and editing. Shooting a film involves recording video scenes, as indicated in the script. A scene is typically recorded several times, with variations indicated by the director. The result of the shooting is the rushes, video tapes which contain several takes of various scenes from the film. During post-production, the film editor first builds a rough cut, by selecting and ordering the best takes (shots) of each

E. Dumont and B. Mérialdo
Institut Eurecom
Multimedia Department
2229 route des Crêtes – BP 193
06904 Sophia-Antipolis CEDEX
Tel.: +(33)4 93 00 81 29
Fax: +(33)4 93 00 82 00
E-mail: merialdo@eurecom.fr

scene. Then, he creates a fine cut by arranging all the shots to flow smoothly in a seamless story.

Automatic video summarization is a powerful tool which allows synthesizing the entire content of a video while preserving the most important or most representative sequences. For this purpose, the content of the video sequence is analyzed, and its structure is identified, so that the most relevant video segments can be selected.

In this paper, we focus on the analysis and summarization of video rushes, as used in the TRECVID BBC Rushes Summarization Task [1]. As mentioned previously, rushes videos are the raw recordings from a camera, taken during the preparation of a movie or a documentary. Rushes exhibit a very specific structure. The recording of a movie is organized in scenes, where each scene represents a given piece of action. Typically, a scene will be recorded several times, each recording being a different take, because the director will ask for variations of the presentation within the action, or sometimes because some recordings are disturbed with unexpected mistakes. In rushes videos, a take will be a continuous recording from the camera, and, for short takes, it may happen that several takes are recorded continuously in the same video sequence. Furthermore, rushes videos will also contain auxiliary data such as test patterns, to calibrate the camera colors, or clapper sequences which identify the take and scene number in the recording and facilitate the alignment of the soundtrack with the video.

The TRECVID BBC Rushes Summarization campaign has defined a task where, given a video from the rushes test collection, one has to automatically create a MPEG-1 video summary with a maximum duration of 2%. The summary should show the main objects and events of the original video, in a way that maximizes the usability of the summary and the speed of objects/event recognition. The evaluation is performed by human assessors who watch the summaries and provide various indicators on the quality and coverage of their content.

As mentioned previously, the content of rushes videos is very specific. Rushes videos contain a lot of repetitions, often with minor variations. They may also contain long segments in which the camera is fixed on a given scene or barely moving, and reusable shots of people, objects, events, locations, that are sometimes used to fill gaps during the final editing. Although many techniques have already been proposed to automatically process the content of general videos [14] and [5], the specific structure of rushes videos require an adaptation of these techniques, and sometimes, the development of new approaches for an efficient parsing.

2 Summarization approach

In previous work [11] and [7], we have already built a first version of the rushes summarization system based on the following steps:

– detection and removal of junk frames, such as test patterns and clapper boards,
– hierarchical classification of one-second video segments, using a video similarity measure,
– selection of relevant segments for the final summary, using a criterion of maximal coverage.

In this paper, we use the specific structure of rushes video and introduce a new step in the process in which we use a sub-sequence alignment algorithm to structure the video into scenes and takes. In order to use the specific structure of rushes video, we now introduce a new step in the process in which we use a sub-sequence alignment algorithm to structure the video into scenes and takes. With this algorithm, we can detect similar video subsequences, which are likely to represent the different takes of a given scene. This detection allows us to identify the different scenes that appear in the rushes video, and to select for each scene the take that seems to be the most representative. Figure 1 shows the main steps of this process.

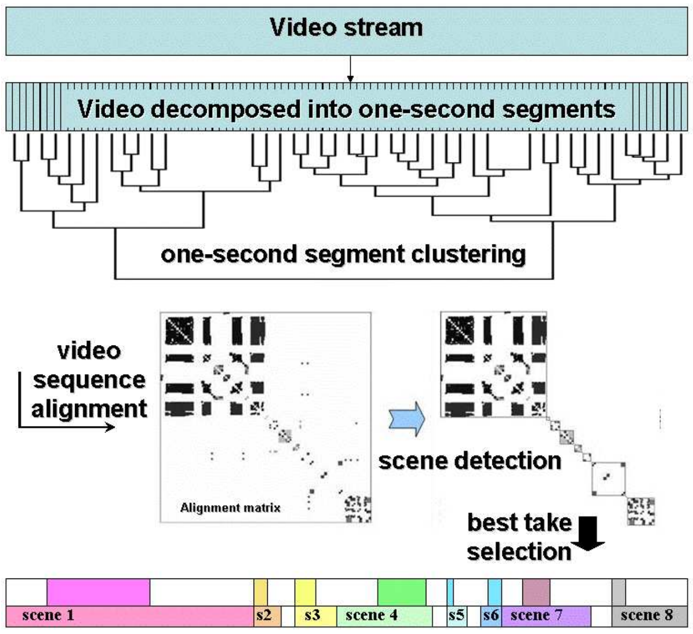


Fig. 1 General approach of video parsing

Rushes contain a lot of uninteresting sequence of frames called junk frames, for example test pattern frames, uniform color frames, clapper sequences, etc... After removing the junk frames, the rushes video is decomposed into one-second segments. Those segments are clustered using a hierarchical classification strategy. Hierarchical classification allows to tune the notion of visual similarity by selecting different levels in the hierarchy. Then a Video Sequence Alignment algorithm (VSA) finds repetitive sequences. Repetitive sequences are likely to be the different takes of the same scene, so that by grouping repetitive sequences, we can identify the various scenes occurring in the video. The comparison between the different takes allows excluding uncomplete takes and selecting the longest one as the most representative. Finally, the rushes video summary can be constructed by concatenating an accelerated copy of the best take of each scene.

A major problem to develop summaries is the fact that evaluation is difficult, in the sense that it is hard to judge the quality of a summary, or, when a performance measure is available, it is hard to understand what its interpretation is. So, in the field of automatic summarization, most papers suggest their own evaluation technique, chosen appropriately for their own task. This makes the comparison of different systems difficult, if not impossible, and creates an urgent need for a commonly accepted evaluation methodology.
Our approach is to search for an automation of the evaluation procedure proposed in TRECVID 2008 using the same quality criteria. We decided to focus on the main indicator: the percentage of topics found in the summary IN, because other subjective measures are correlated with it [6]. Previous work already tackled this problem. In [7] and [8] authors automated evaluation with a basic and efficient method: a topic is found by the automatic evaluator if a frame sequence of summary overlaps with one of the occurrences of this topic in the original video during one second. The work presented here is an extension of these approaches, in particular [9]. Authors proposed an automation of the manual TRECVID evaluation using machine learning techniques. The main difference is the definition of objects used to predict the topic presence and then, results are improved.

3 Video Sequence Alignment

3.1 Sequence Alignment Algorithms

In 1966, Levenshtein introduced the notion of edit distance by the question: "What is the minimal number of edit operations to transform a string into another?". He also proposed a dynamic programming algorithm that allows a very fast computation of this distance. The Levenshtein distance has since become a very popular metric for measuring the amount of difference between two sequences. The Levenshtein distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is either an insertion, deletion, or scoring of a single character. In 1970, Needleman-Wunsch [15] proposed a similar algorithm to perform a global alignment over two protein sequences. To find the alignment with the highest score, a two-dimensional matrix is allocated, with one column for each character in the first sequence, and one row for each character in the second sequence. Thus, if we are aligning sequences of sizes n and m, the running time of the algorithm is O(nm) and the amount of memory used is in O(nm). The memory requirement can be reduced to O(min(m,n)) if back-tracking is not needed.

In 1981, Smith-Waterman [16] proposed a variation of this algorithm to perform local sequence alignment (see figure 2): instead of looking at the total sequence, the Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure. This is done by creating a scoring matrix where cells indicate the cost of changing a sub-sequence of one sequence into one of the sub-sequences of the other sequence. The main difference with the Needleman-Wunsch algorithm is that negative scores are thresholded to zero, so that the positively scoring local alignments become visible. Back-tracking starts at the highest scoring matrix cell and proceeds until a cell with a score of zero is encountered, yielding the highest scoring local alignment. Figure 3 shows an example of the scoring matrix between HEAGAWGHEE and

Given : Two sequences $A = a_1 a_2 ... a_n$ $B = b_1 b_2 ... b_m$

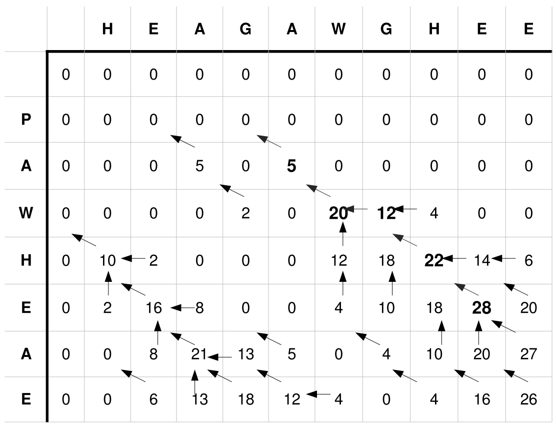
- Compute $(n+1) * (m+1)$ scoring matrix M_l where $M_l[i][j]$ represents the cost of the sub-sequence alignment ending with segments s_i and s_j.
  - initialize $M_l[i][0] = 0$, $M_l[0][j] = 0$ for all $0 \leq i \leq m, 0 \leq j \leq n$
  - compute $M_l[i][j]$, for all $1 \leq i \leq m, 1 \leq j \leq n$ by:

$$M_l[i][j] = max \begin{pmatrix} 0 \\ M_l[i-1][j-1] + match\_cost(\mathbf{i}, \mathbf{j}) \\ M_l[i][j-1] + gap\_cost \\ M_l[i-1][j] + gap\_cost \end{pmatrix}$$

- Find the best sub-sequence alignment from the maximal value $M_l[i][j]$.

Fig. 2 Smith-Waterman algorithm

PAWHEAE, where the best local alignment (in bold) is AWGHE with AW-HE. The complexity requirements of the Smith-Waterman algorithms are similar to those of the Needleman-Wunsch algorithm.



		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	5	0	**5**	0	0	0	0	0
W	0	0	0	0	2	0	**20**	**12**	4	0	0
H	0	10	2	0	0	0	12	18	**22**	14	6
E	0	2	16	8	0	0	4	10	18	**28**	20
A	0	0	8	21	13	5	0	4	10	20	27
E	0	0	6	13	18	12	4	0	4	16	26

Fig. 3 Example of scoring matrix

Sequence alignment algorithms have already been used in the summarization of rushes videos. For example, Chasanis et al. [12] decompose a video into shots and then perform a global alignment between all pairs of shots. In [13] Liu et al. perform a local alignment between successive shots, and they obtain an alignment score which is used to classify shots. In [10] Bailer et al. use Longest Common Subsequence Algorithms to find the various takes of a shot. Although these algorithms share the dynamic programing approach of the Smith-Waterman algorithm, to the best of our knowledge, our work is the first time where the Smith-Waterman algorithm is used for rushes video summarization. The advantage of using this algorithm is that local alignments are automatically found without the need for a prior segmentation of the video sequence.

3.2 Video Sequence Alignment algorithm

We adapt the Smith-Waterman algorithm to find repetitive sequences in a video: this is the VSA (Visual Sequence Alignment) algorithm. In order to detect similarities between sub-sequences of the video, we partition the video into small segments, and these segments are hierarchically clustered by visual similarity. So for a clustering level, we represent a video by a list of clusters like a list of strings. The hierarchical classification is useful because it can easily provide various similarity thresholds, so that we can adapt the degree of similarity to the variability of the visual content. Then, we search visually similar video sub-sequences for different degree of similarity. In Smith-Waterman algorithm, similarities are searched into two different sequences but our case, we search similarities into the same sequence: the global video.

The video is decomposed into non-overlapping segments of minimal temporal unit duration. This unit is the shortest sequence that could be perceived by a human. A study showed that one second is the minimal length to see a concept in a video sequence [8]. Another showed that 20.5 frames is required to see a concept [9]. We have therefore chosen to use this temporal unit of one second and decompose the rushes video into one-second segments (25 frames).

Each one-second segment is represented by a feature vector built from the average HSV histogram (18 bins for H, 3 for S and V) of the frames it contains. The clustering algorithm starts with as many clusters as there are one-second segments, then at each step of the clustering, the number of clusters is reduced by one by merging the closest two clusters, until all segments are finally in the same cluster. The distance between two one-second segments is computed as the Euclidean distance, and the distance between two clusters is the average distance across all possible pairs of segments of each cluster.

3.2.1 Scoring matrix

We search for local alignments between a video sequence and itself. We use the following definitions and assumptions to compute the scoring matrix:

- A video sub-sequence $S = s_1 s_2 ... s_n$ is a list of one-second segments.
- Two one-second segments s_1 and s_2 are **aligned** if some sub-sequences containing s_1 and s_2 are **aligned**.
- Two aligned sub-sequences cannot contain the same one-second segment.
- A pair of one-second segments can be aligned only once.
- Two aligned sub-sequences should not contain another pair of aligned sub-sequences (only minimal alignments are considered).

The video sequence is represented as a list of one-second segment clusters $S_l = c_1 c_2 ... c_m$ where c_i corresponds to the cluster of the segment i at the clustering level l. The $(m+1)*(m+1)$ scoring matrix $M_l[i][j]$ at level l is computed as : $M_l[i][0] = 0$, $M_l[0][i] = 0$ and $M_l[i][i] = 0 \, \forall i \in 0, ..., n$

$$M_l[i][j] = max \begin{pmatrix} 0 \\ M_l[i-1][j-1] + match\_cost(\mathbf{i},\mathbf{j}) \\ M_l[i][j-1] + gap\_cost \\ M_l[i-1][j] + gap\_cost \end{pmatrix}$$

where *match_cost* is the cost of aligning two segments (the distance between their clusters) and *gap_cost* is the cost to add a gap in the alignment. When $M_l[i][j] > 0$, the alignment of two sub-sequences ending in position i and j can be found by recursively adding the antecedent (u, v) which realizes the maximum of M starting from $M_l[i][j]$, until a zero value is found for $M_l[u][v]$. In the case where several antecedents realize the maximum, the diagonal is preferred. The number of antecedents is the length of the alignment $length(i, j)$.

To favor alignment quality rather than alignment length, so we normalize the scoring matrix M_l by the length of the alignments: $\bar{M_l}[i][j] = \frac{M_l[i][j]}{length(i,j)}$.

3.2.2 Visual Sequence Alignment

The Smith-Waterman algorithm can be applied iteratively to find all repetitive sequences in a video. We propose to use a varying clustering level allows to have a coarser or finer definition of the visual similarity. We start with a finer visual similarity, to detect the most similar sub-sequences first, and continue with a coarser similarity to find weaker alignments. The complete Visual Sequence Alignment algorithm is described in figure 4.

```
Given : A video sequence $S$ is defined as a list of $m$ one-second segments: $S = s_1 s_2 ... s_m$
  – Hierarchical clustering: $S_l = c_1 c_2 ... c_m$ where $c_i$ is the cluster of segment $s_i$ of
    the clustering level $l$.
  – $l = 0$.
  – Compute $(m+1)*(m+1)$ normalized scoring matrix $\bar{M_l}$ where $\bar{M_l}[i][j]$ represents
    the cost of the sub-sequence alignment ending with segments $s_i$ and $s_j$.
  – Iteratively: find the best sub-sequence alignment, i.e. the maximal value $\bar{M_l}$.
      – If $\bar{M_l} > threshold$, we store this alignment and we update the scoring matrix.
      – Else $l = l + 1$ and we update the scoring matrix.
Output: An ordered list of aligned sub-sequences.
```

Fig. 4 VSA: Video Sequence Alignment algorithm

The result of the VSA is an ordered list of aligned sub-sequences, where the order corresponds to the confidence that we can assign to the alignment, the best alignments being found first. As we let the algorithm run, erroneous alignments may be introduced. Those will be filtered in the next processing step, where scene detection is performed.

4 Rushes video parsing

Every scene is generally recorded in several takes, which are variations for the same scene. The takes for a scene will occur contiguously in the video. Scenes can be identified from the alignments that have been found by the VSA by searching for sets of contiguous subsequences which are (almost) all aligned together and (almost) not aligned with subsequences of other sets. During this process, we may remove some false alignments when two aligned sub-sequences do not belong to the same scene.

4.1 Alignment matrix

The alignment matrix is a matrix of scores which express the confidence of the alignment between two frames. We now work at the frame level where a video sequence is defined as a list of frames $V = f_1...f_n$. We construct a $n * n$ alignment matrix A where $A[f_i][f_j]$ is the rank of the alignment between segments which contain frames f_i and f_j, if one exists. If no such alignment exists, the value of $A[f_i][f_j]$ is set to the total number of alignments found plus one.

4.2 Scene detection

Figure 5 shows a picture of the alignment matrix, where black areas correspond to the best alignments (with lowest rank), and white areas to the worse or no alignment. We assume that the different takes of the same scene are visually very similar. So, in the alignment matrix, they should correspond to a black square area along the diagonal. Since two scenes are presumably visually different, we can detect the boundary between scenes by searching for white rectangle areas in the upper right corner of the alignment matrix. More precisely, we compute the confidence $rect(f)$ of a frame f to be a scene transition on the video sequence beginning at the frame $first$ and finishing at the frame $last$ by:

$$rect(f) = \frac{\sum_{f_1=first}^{f_1=f} \sum_{f_2=f}^{f_2=last} A[f_1][f_2]}{(f+1-first)(last+1-f)}$$

We search the frame $f \in [first, last]$ which maximizes the value of $rect(f)$. If this value is greater than a threshold, f delimits a scene transition. We apply this process recursively on both sides of f to find the other scene boundaries. At the beginning, we fix $first = 0 \wedge last = F$ (F is the number of frames in the video). The process is continued as long as we can find rectangles with values greater than the threshold. The threshold has been manually adjusted, and is the same for every video sequence. $rect(f)$ is a value between 0 and 1, so we tested $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.85, 0.90, 0.95, 0.98\}$. When no rectangle can be found, the decomposition into scenes is complete. We can then detect the false alignments as the alignments between subsequences from different scenes. Figure 5 shows a picture of the video scene decomposition process.

4.3 Take selection

Each alignment identifies two takes of the same scene. Among all takes, we would like to select the take that is most representative of the scene content. We can make the following remarks:

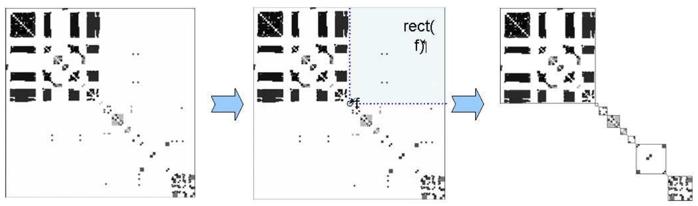


Fig. 5 Video scene decomposition

- The different takes of a given scene presumably contain very similar content, therefore it is likely that different takes (or parts of different takes) will appear in the list of aligned sub-sequences. A take should be a sequence of frames which do not contain aligned frames.
- Some takes may be shorter, for example when an unexpected event happens that does not allow a full recording of the action. The longest take is therefore a good candidate for being the best representative for the scene.

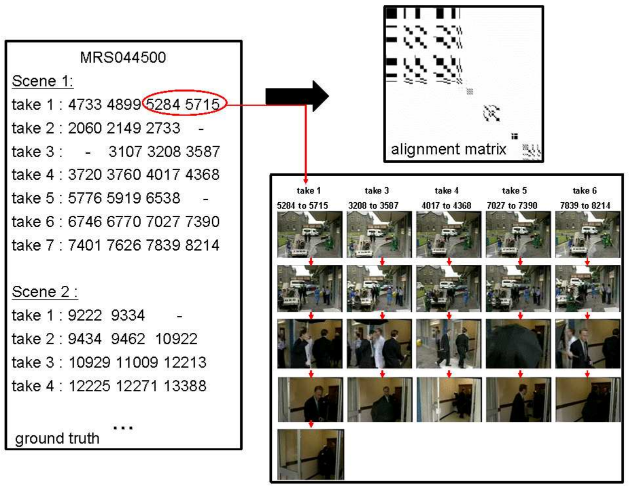
Based on these remarks, we do not search for a precise decomposition of the scene into takes, but rather we search for the longest take by searching the longest contiguous sequence of frames which do not contain frames that have been aligned together. This sequence is kept as the reference take for the scene.

4.4 Visual Sequence Alignment evaluation

We experimented our approach on videos used in the TRECVID BBC Rushes Summarization Tasks for 2008: 6 for the development and 8 for the test. It consists of unedited video footage, shot mainly for five series of BBC drama programs and was provided to TRECVID for research purposes by BBC archive.
In the ground truth, the important information to evaluate our system is the video decomposition in scenes and takes: a scene is decomposed into takes and a take can be decomposed into consecutive take fragments, (not all take fragments are present in all takes, since some takes may have been shortened). Take fragments are delimited by frame numbers. We constructed the ground truth data by manually defining the various scenes, takes and take fragments, as illustrated in figure 6. The ground truth shows the take fragments of the different scenes that can be aligned together. So, from the ground truth data, we can easily infer the ground truth alignment matrix of the video sequence.

Figure 7 compares the ground truth alignment (left) with the scene structure found by VSA (right) for some example videos. At the bottom of the matrix, the recall - precision values for our method and for JRS method [10].

Table 1 shows results on the 8 test videos. We evaluate the quality of the alignment detection by measuring the black surface of the alignment matrix. With this measure, the recall rate varies between 0.154 and 0.444, and the precision between 0.422 and 0.762. For scene detection, for a measure also based on the area, precision varies between 0.771 and 0.978, and the recall between 0.554 and 0.932. This shows that our proposed method is able to perform a sensible decomposition of the rushes video into


MRS044500

Scene 1:

take 1 : 4733 4899 5284 5715

take 2 : 2060 2149 2733 -

take 3 : - 3107 3208 3587

take 4 : 3720 3760 4017 4368

take 5 : 5776 5919 6538 -

take 6 : 6746 6770 7027 7390

take 7 : 7401 7626 7839 8214

Scene 2 :

take 1 : 9222 9334 -

take 2 : 9434 9462 10922

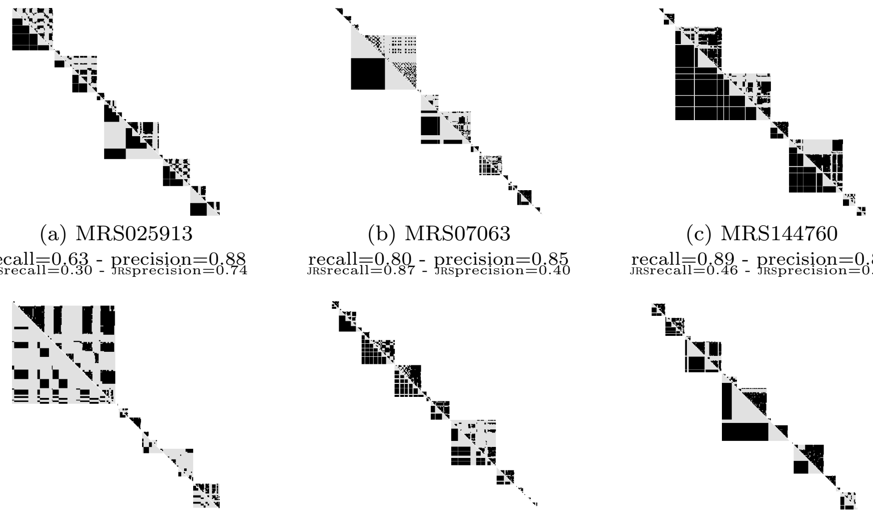
take 3 : 10929 11009 12213

take 4 : 12225 12271 13388

...

ground truth

Fig. 6 Ground truth of video MRS044500, and alignment matrix corresponding with sample of aligned sub-sequences.


(a) MRS025913
recall=0.63 - precision=0.88
JRSrecall=0.30 - JRSprecision=0.74

(b) MRS07063
recall=0.80 - precision=0.85
JRSrecall=0.87 - JRSprecision=0.40

(c) MRS144760
recall=0.89 - precision=0.87
JRSrecall=0.46 - JRSprecision=0.94

(d) MRS044500
recall=0.61 - precision=0.98
JRSrecall=na - JRSprecision=na

(e) MRS157475
recall=0.79 - precision=0.90
JRSrecall=0.47 - JRSprecision=0.61

(f) MS216210
recall=0.49 - precision=0.94
JRSrecall=0.42 - JRSprecision=0.91

Fig. 7 Matrix alignment of ground truth (below diagonal) versus video parsing found by VSA (above diagonal) in black, and surface area of scenes in gray.

scenes.

	MRS035126	MRS048773	MRS151585	MRS157479	MRS044499	MRS145229	MRS157450	MS206370
VSA Recall	0.23	0.28	0.30	0.28	0.44	0.15	0.24	0.16
VSA Precision	0.75	0.76	0.61	0.60	0.69	0.42	0.58	0.46
SD Recall	0.72	0.93	0.57	0.75	0.55	0.68	0.77	0.80
SD Precision	0.92	0.81	0.88	0.94	0.92	0.77	0.98	0.77

Table 1 Evaluation on test data

5 TRECVID manual evaluation

Automatic video summarization is a challenge. It requires decisions about the semantic content and importance of each segment in a video. This explains the difficulty of the development of automatic video summarization systems and in particular, of evaluation methods. Much of the complexity of summary evaluation arises in the fact that it is difficult to specify what one really needs to measure, without a precise formulation of what the summary is aimed to capture. The TRECVID evaluation campaign has defined an evaluation procedure where human assessors watch the summaries to evaluate the amount of content that they contain.

The ground truth is a list of important video segments, each identified by means of a distinctive object or event occurring in the segment with qualifications concerning camera angle, distance, or some other information to make each item description unique. A complete description can be found in [1]. The ground truth provided by TRECVID is a simple chronological list of topics. The average number of ground truth topics for each video is more than 20. In TRECVID, this was considered too large for human evaluators, so that the evaluation was only performed for a random list of 12 topics per video.

Each submitted summary was judged by three different human judges (assessors). An assessor was given the summary and a corresponding list of up 12 topics from the ground truth. The assessor viewed the summary in a $125mm$x$102mm$ mplayer window at 25 frames per second using only play and pause controls and then was asked which of the designated topics appeared in the summary. The percentage of topics found by each assessor determines the fraction of important segments from the full video included. The total score for a summary is the average of the scores given by the three assessors: IN. Figure 8 depicts the video summary evaluation process. The results of the manual evaluation were statistically analyzed in [1], and in conclusion authors found that there was a good agreement between assessor judgments based on the comparison of the topics detected by two assessors in a summary.

6 Summary Automatic evaluation

In order to automate the evaluation process, we propose to automate the decision on topic detection by creating an automatic assessor which predicts the topic presence in a summary based on topic, video and summary features. Figure 9 is an overview of the

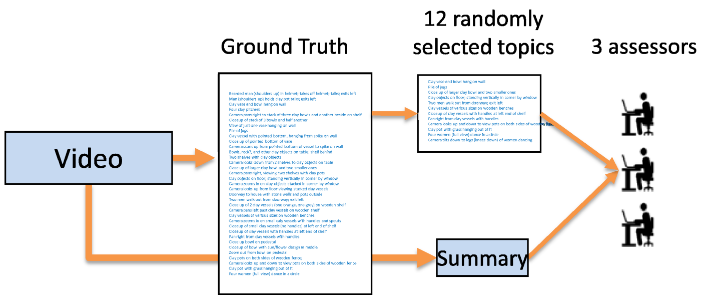


Fig. 8 TRECVID 2008 manual evaluation processus

process.

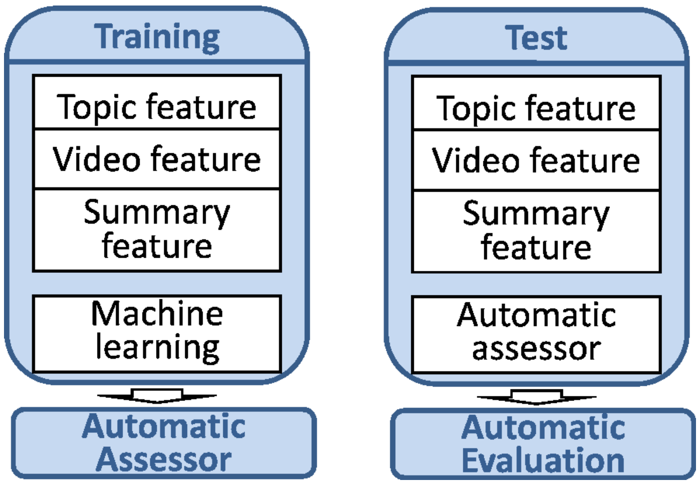


Fig. 9 Process overview

For this purpose, we need to augment the TRECVID ground truth to include the precise time boundaries for each occurrence of each topic in the videos. This augmentation was done manually, and allows us to precisely compute how many frames of a topic have been included in the summary. In other words, the summary features are automatically comptuted, but the video features are manually detected.

6.1 Modelling topic assessment

For our modeling of the automatic assessment, we define a topic instance i as a couple $(\mathbf{x}_i, y_i)$ where:

- $\mathbf{x}_i \in \mathcal{X}$ is a vector containing measurements on the occurrence of the topic,
- $y_i \in \{\text{presence}, \text{absence}\}$ is the result of the decision on the occurrence of the topic, based on the values in $\mathbf{x}_i$.

A topic can have repeated occurrences in a video. We call each of these occurrences a topic sequence. The decision of detecting a topic or not depends on the occurrences of the topic in the original video, and on its occurrences in the proposed summary. Therefore, the vector $\mathbf{x}_i$ hopefully contains all values necessary to take a decision on a topic presence. It contains information coming from the original video and the ground truth, as well as information coming from the proposed summary. In our proposed model, we include the following measurements in the description of a topic instance. The following information is obtained from the augmented ground truth and the original video:

x_1	: Does the topic contain camera motion
x_2	: Does the topic contain an event
x_3	: Number of topic sequences in the video
x_4	: Minimal length of a topic sequence in the video
x_5	: Maximal length of a topic sequence in the video
x_6	: Mean length of a topic sequence in the video
x_7	: Mean activity of topic sequences in the video
x_8	: Mean entropy of topic sequences in the video

The other measurements are obtained automatically from the content of the proposed summary:

x_9	: Number of topic sequences in the summary
x_{10}	: Minimal length of a topic sequence in the summary
x_{11}	: Maximal length of a topic sequence in the summary
x_{12}	: Mean length of a topic sequence in the summary

With this formulation, an automatic assessor will decide on the presence or absence y_i of a topic based on the values of the measurements in $\mathbf{x}_i$.

6.2 Training an automatic assessor

An automatic assessor will define a function *prediction* that predicts the presence or absence of a topic. If a topic is present, the function returns 1, else the function returns 0. So, once this prediction function is defined by a machine learning technique, we can compute automatically the IN indicator, the percentage of topics found in the summary, for a video v by the following formula:

$$IN(v) = \frac{1}{N}\sum_{i=1}^{N} \mathit{prediction}(i)$$

where N is the number of topics in the video. We can also rank summaries according IN.

From the detailed submission results to the TRECVID summarization task, we can create training data in the form of a set of topic instances $(\mathbf{x}_i, y_i)$. This list contains the various decisions y_i made by the assessors on proposed summaries, together with the corresponding measurements x_i on the occurrences of the corresponding topic. Based on this training data, we compare various machine learning techniques to construct an automated assessor, with the objective of providing decisions that are as close as possible to those of the human assessors.

6.3 Assessor evaluation

In statistics, the Pearson product-moment correlation coefficient r is a common measure of the correlation between two variables. For our problem, Pearson's correlation reflects the degree of linear relationship between manual evaluation and automatic evaluation. It ranges from $+1$ to -1. A correlation of $+1$ means that there is a perfect positive linear relationship between evaluations. A correlation of -1 means that there is a perfect negative linear relationship between evaluations. A correlation of 0 means there is no linear relationship between evaluations.

$$r = \frac{\operatorname{cov}(X, Y)}{\sqrt{(V(X)V(Y))}}$$

where $\operatorname{cov}(X, Y)$ denotes the covariance between X and Y, and $V(X)$, $V(Y)$ respectively the variance of X, and the variance of Y. We evaluate the correlation at 3 different levels :

- Topic: we compute the correlation of the prediction of the topic presence between our automatic assessor and the manual assessors.
- IN: we compute the correlation of the IN evaluation between our automatic evaluation and the manual evaluation.
- Ranking: we rank summaries based on their IN value, and compute the correlation between our automatic evaluation and the manual evaluation.

7 Automatic Assessor evaluation

We experimented our approach on 8 videos proposed by BBC Rushes Task 2008 in TRECVID. It consists of unedited video footage, shot mainly for five series of BBC drama programs and was provided to TRECVID for research purposes by BBC archive. The training instances are obtained from results of TRECVID 2008 evaluation for 10 summarization systems kindly sent to us by various participants.

In order to have the training data and the training test completely independent, we choose to use a leave-one-out technique: for each video and for each system, we train the model on 7 videos x 9 systems and we test the model on the last video and system. Figure 10 shows the Pearson correlation between manual assessors and our automatic assessor at 3 levels (topic, IN and ranking) for different machine learning methods using the WEKA sotfware [17].

These experimental results show that the best performing classifier type is the decision stump, although Bayes networks have a very close performance (in previous experiments with less data, Bayes networks were the best classifier). A decision stump is a weak machine learning model consisting of a decision tree with only a single node. To better understand the automatic assessor, we identified the decision stumps that worked best based on the leave-one-out method. We obtain only 4 different decision stumps, shown in figure 11.

This shows that simple rules are able to provide a reasonable prediction of the occurrence of a topic, based on simple measurements.

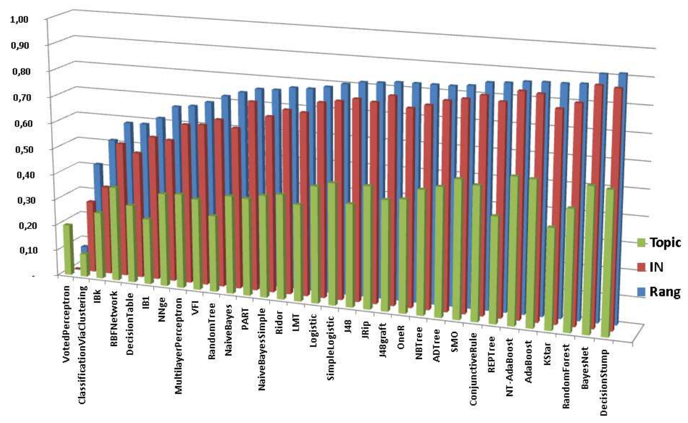


Fig. 10 Pearson correlation between manual and automatic assessors.

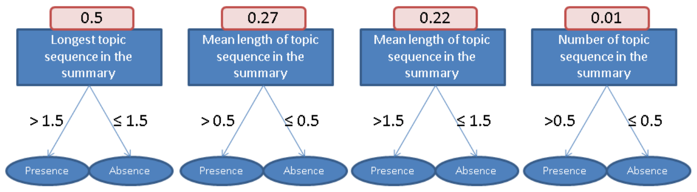

0.5
Longest topic sequence in the summary
> 1.5
≤ 1.5
Presence
Absence
0.27
Mean length of topic sequence in the summary
> 0.5
≤ 0.5
Presence
Absence
0.22
Mean length of topic sequence in the summary
>1.5
≤ 1.5
Presence
Absence
0.01
Number of topic sequence in the summary
>0.5
≤ 0.5
Presence
Absence

Fig. 11 Decision stumps found.
At the top, the ratio of occurence of the decision stump.

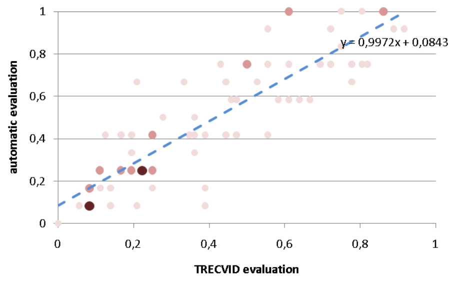
The table 2 shows the correlation between the manual prediction and the automatic prediction of topic presence. The Pearson correlation coefficient is 0.54, which indicates a moderate correlation.

Classified as	Absence	Presence
Absence	390	116
Presence	45	313

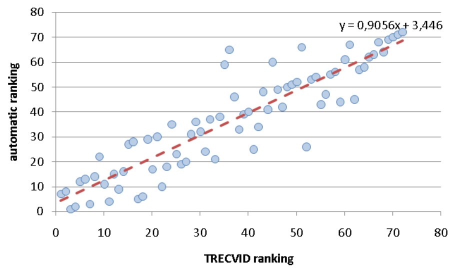
Table 2 Confusion matrix of the topic presence prediction.

Figure 12 shows the manual evaluation of IN and ranking according to the automatic evaluation. At these levels, the Pearson correlation is 0.88 and 0.91, which shows a high correlation.

In practice, manual assessors do not always agree, because of the subjective interpretation of topic occurrence. We would like a classifier that shows a close agreement with manual evaluation, if possible as close as between two human assessors. We evaluate the quality of our automatic assessor in comparison to an human assessor. We use all manual evaluations done by TRECVID. For each pair of assessors, we compute the correlation between their evaluations at the 3 levels. We average coefficients for each assessor, table 3 shows the results.



(a) *IN* level



(b) Ranking level

Fig. 12 Pearson correlation between manual assessor and our automatic assessor

Assessor	Topic	IN	Ranking
Assessor 1	0.755961	0.878687	0.914713
Assessor 2	0.789278	0.875702	0.917511
Assessor 3	0.770808	0.870860	0.911205
Assessor 4	0.775011	0.860053	0.895711
Assessor 5	0.790169	0.865818	0.897900
Assessor 6	0.750509	0.805306	0.833046
Assessor 7	0.715957	0.781069	0.824572
Assessor 8	0.702580	0.804130	0.811149
Assessor 9	0.726755	0.855810	0.892882
Assessor 10	0.790546	0.901866	0.926379
DecisionStump	0.535261	0.875906	0.913306

Table 3 Assessor correlation.

These experiments show that the automatic evaluation is at a similar level than manual evaluation for the IN and Ranking criteria: the correlation between automatic and human is as high as the correlation between manual evaluations. But at the topic level, the automatic assessor has only a moderate correlation with the manual evaluation. So, these experiments demonstrate that our automatic evaluation technique is suitable for comparing and evaluating summaries using *IN* indicator.

8 Conclusion

In this paper, we have introduced a Video Sequence Alignment algorithm, VSA, which uses a dynamic programming approach to identify similar sub-sequences in a video sequence. This algorithm is used to parse rushes video and structure them into scenes and takes. We have described the details of the algorithm and evaluated its performance on the TRECVID BBC Rushes Summarization task videos. VSA is a useful step in the construction of summaries for rushes video. In the future, we plan to extend it to other video processing applications, for example, to structure more general videos by detecting similar sub-sequences.
We have also proposed an approach to automate the summary evaluation by training a decision stump in order to remove the human interaction that was required in the TRECVID evaluation campaign. Through experiments, we showed a high correlation be-

tween the manual evaluation proposed by TRECVID2008 and our automatic evaluation. In further work, it would be interesting to generalize our approach on a larger data set including more videos and more summarization systems to improve the prediction quality.

References

1. Paul Over, Alan F. Smeaton, and George Awad, "The TRECVID 2008 BBC rushes summarization evaluation", in *Proceedings of the TRECVID Workshop on Video Summarization (TVS'08)*, Vancouver, BC, Canada, October 2008.
2. A.M. Ferman, A.M. Tekalp, "Two-stage hierarchical video summary extraction to match low-level user browsing preferences", in *IEEE Transactions on Multimedia*, vol. 5, no. 2, pp. 244–256, June 2003.
3. A. Ekin, A.M. Tekalp, and R. Mehrotra, "Automatic soccer video analysis and summarization", in *Image Processing, IEEE Transactions on*, vol. 12, no. 7, pp. 796–807, July 2003.
4. Cuneyt M. Taskiran, "Evaluation of automatic video summarization systems", in *Multimedia Content Analysis, Management, and Retrieval 2006*, Edward Y. Chang, Alan Hanjalic, and Nicu Sebe, Eds. 2006, SPIE.
5. Ba Tu Truong and Svetha Venkatesh, "Video abstraction: A systematic review and classification", in *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 3, no. 1, 2007.
6. Marcin Detyniecki and Christophe Marsala, "Adaptive acceleration and shot stacking for video rushes summarization", in *Proceedings of the TRECVID Workshop on Video Summarization (TVS'08)*, Vancouver, BC, Canada, October 2008.
7. Emilie Dumont and Bernard Mérialdo, "Split-screen dynamically accelerated video summaries", in *MM 2007, 15th international ACM conference on multimedia, September 24-29, 2007, Augsburg, Germany*, Sep 2007.
8. Alexander G. Hauptmann, Michael G. Christel, Wei-Hao Lin, Bryan Maher, Jun Yang, Robert V. Baron, and Guang Xiang, "Clever clustering vs. simple speed-up for summarizing rushes", in *TVS '07: Proceedings of the international workshop on TRECVID Video Summarization*, Augsburg, Germany, 2007.
9. Dumont Emilie and Mérialdo Bernard, "Automatic evaluation method for rushes summarization: experimentation and analysis", in *CBMI 2008, 6th International Workshop on Content-Based Multimedia Indexing, June 18-20, 2008, London, UK*, 2008.
10. W. Bailer, F. Lee, and G. Thallinger. "A distance measure for repeated takes of one scene", in *The Visual Computer*, 25:53–68, 2009.
11. R. Benmokhtar, E. Dumont, B. Mérialdo, and B. Huet. "Eurecom in TrecVid 2006: high level features extractions and rushes study", in *TrecVid 2006, 10th International Workshop on Video Retrieval Evaluation, November 2006, Gaithersburg, USA*, 2006.
12. V. Chasanis, A. Likas, and N. P. Galatsanos. "Video rushes summarization using spectral clustering and sequence alignment", in *TRECVID BBC Rushes Summarization Workshop (TVS'08), ACM International Conference on Multimedia*, pages 75–79, Vancouver, BC, Canada, october 2008.
13. Y. Liu, Y. Liu, T. Ren, and K. C. C. Chan. "Rushes video summarization using audio-visual information and sequence alignment", in *TRECVID BBC Rushes Summarization Workshop (TVS'08), ACM International Conference on Multimedia*, pages 114–118, Vancouver, BC, Canada, october 2008.
14. A. Money and H. Agius. "Video summarisation: A conceptual framework and survey of the state of the art", in *Journal of Visual Communication and Image Representation*, 2007.
15. S. Needleman and C. Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins", in *Journal of Molecular Biology*, 1970.
16. T. F. Smith and M. S. Waterman. "Identification of common molecular subsequences", in *Journal of Molecular Biology*, 1981.
17. Ian H. Witten and Eibe Frank. "Data Mining: Practical machine learning tools and techniques", in *2nd Edition, Morgan Kaufmann, San Francisco*, 2005.