# Rushes video parsing using video sequence alignment

Emilie Dumont and Bernard Mérialdo
EURECOM
Sophia-Antipolis, FRANCE
dumont@eurecom.fr, merialdo@eurecom.fr

## Abstract

*In this paper, we propose a novel method inspired by the bio-informatics domain to parse a rushes video into scenes and takes. The Smith-Waterman algorithm provides an efficient way to compare sequences by comparing segments of all possible lengths and optimizing the similarity measure. We propose to adapt this method in order to detect repetitive sequences in rushes video. Based on the alignments found, we can parse the video into scenes and takes. By comparing takes together, we can select the most complete take in each scene. This method is evaluated on several rushes videos from the TRECVID BBC Rushes Summarization campaign.*

## 1 Introduction

With rapid advances in the technology of digital video documents and although powerful technologies now exist to create, play, store and transmit those documents, the analysis of the video content is still an open and active research challenge. In this paper, we focus on video film making tools. The automatic creation of video summaries [**?**, **?**] is a powerful tool which allows making summary by synthesis the entire content of a video while preserving the most important or most representative sequences. For this purpose, the content of the video sequence has to be analyzed, and its structure has to be identified, so that the most relevant video segments can be selected. In this paper, we focus on the analysis of video rushes, as used in the TRECVID BBC Rushes Summarization campaign. Rushes videos are the raw recordings from a camera, taken during the preparation of a movie or a documentary. They are unedited, and they constitute the raw material from which the video editor will select segments and compose the final video program. 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 is 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 the rushes video, 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, the 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 they are also used for alignment of the soundtrack with the video. These characteristics require adequate processing for the analysis of rushes videos. In this paper, we propose an original approach which uses sequence alignment algorithms inspired from the bio-informatics domain to structure a rushes video into scenes and takes. In the following section, we discuss the motivation for this work, and then, we detail the video sequence alignment algorithm and finally, we evaluate this algorithm on several rushes videos proposed during the TRECVID BBC Rushes Summarization campaign.

## 2 Motivation

The TRECVID BBC Rushes Summarization campaign proposes a task where, given a video from the rushes test collection, one has to automatically create an MPEG-1 video summary with a maximum duration of $2\%$ that shows the main objects and events of the original video. The summary should minimize the number of frames used and present the information in ways that maximize the usability of the summary and 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, for example several takes of the same scene with variations due to the indications of the director, or to unexpected events and errors. They 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, the specific structure of rushes videos require an adaptation of these techniques, and sometimes, the development of new approaches for an efficient parsing. In previous work [**?**] [**?**], we have already tackled the problem of detecting and removing junk frames (such as test patterns and clapper board), defining a video similarity measure (based on a hierarchical classification of one second segments), and selecting relevant segments for the final summary (through a criterion of maximal coverage). To extend this work further, we introduce a new step in the process in which we use a sub-sequence alignment algorithm to structure the video into scenes and takes. We can compare takes of the same scene together, and select the take that seems to be the most representative. Figure **??** shows the main steps of this new process.

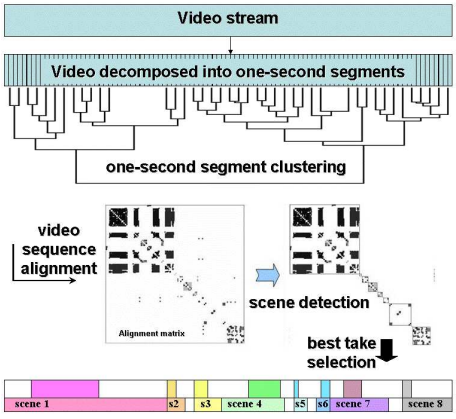


**Figure 1. General approach of video parsing**

In this process, the main steps to parse a video are: first, to decompose the video into one-second segments and to cluster these segments by a hierarchical method. Secondly, we use a Video Sequence Alignment algorithm (VSA) to find repetitive sequences. Repetitive sequences are the different takes of the same scene, so that by grouping repetitive sequences, we can identify the various scenes occurring in the video. Finally, the comparison between the different takes allows selecting the most representative one.

## 3 Local sequence alignment algorithm

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?". The Levenshtein distance is a 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 [**?**] proposed an algorithm to perform a global alignment over two sequences by dynamic programming. 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).

Given : Two nucleotide or protein sequences $A = a_1a_2...a_n$ $B = b_1b_2...b_m$

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

Output : The best sub-sequence alignment.

**Figure 2. Smith-Waterman algorithm**

In 1981, Smith-Waterman [**?**] proposed a variation of this algorithm to perform local sequence alignment (see figure **??**): 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 with cells indicating the cost to change a sub-sequence of one to the sub-sequences of the other. The main difference to the Needleman-Wunsch algorithm is that negative scoring matrix cells are set to zero, which renders the (thus positively scoring) local alignments visible. Back-tracing starts at the highest scoring matrix cell and proceeds until a cell with score zero is encountered, yielding the highest scoring local alignment. Figure **??** shows the scoring matrix between HEAGAWGHEE and PAWHEAE, the best local alignment (in bold) is AWGHE with AW-HE.

| | | 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 |

**Figure 3. Example of scoring matrix**

Several works is based on the same idea. [**?**] decomposes shots in keyframes and then perform a global alignment between all pairs of shots. Finally they construct a similarity matrix of shots. [**?**] perform a local alignment between successive shots only, so they obtain an alignment score which used to classify shot. In [**?**] use LCSS to find the same takes, we call this method JRS.

# 4 Video Sequence Alignment algorithm

In this section, we explain our adaptation of the Smith-Waterman algorithm to find repetitive sequences in a video: VSA (Visual Sequence Alignment). In order to detect similarities between sub-sequences of the video, we partition the video into one-second segments, and these segments are hierarchically clustered by visual similarity. The hierarchical classification is useful because it can easily provide various similarity thresholds, so that we can adapt it to the variability of the visual content. Then, we search visually similar video sub-sequences.

## 4.1 Temporal unit

Our first step consists in defining a temporal unit from work. 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 [**?**]. Another showed that $20.5$ frames is required to see a concept [**?**]. We have therefore chosen to use this temporal unity of a second. We decompose the rush video into one-second segments ($25$ frames).

## 4.2 Hierarchical clustering

In order to perform an adaptation of Smith-Waterman algorithm, we have to define a good match between two video sequences. A good match happens when the matching is performed between two one-second segments belonging to the same cluster. Each one-second segment is represented by the average HSV histogram ($18$ bins for H, $3$ for S and V) of those frames. The 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.

## 4.3 Scoring matrix

We search local alignments between a video sequence and itself. So, we propose 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 can not contain the same one-second segment.
- A pair of one-second segments can be aligned only once.
- Two aligned sub-sequences must have a minimal number of one-second segments.

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 : $M_l[i][0] = 0,\ M_l[0][i] = 0 \text{ and } M_l[i][i] = 0 \, \forall i \in 0, ..., n$

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

where $match\_cost$ is the cost to align two segments and $gap\_cost$ is the cost to add a gap in the alignment.

## 4.4 VSA

We can use the Smith-Waterman algorithm to find repetitive sequences in a video directly with the previous adaptation to the video domain. This process requires to define a clustering level. In order to eliminate this requirement, we propose to use a varying 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. Another way is to favor perfect alignment rather than long alignment, so we normalized the scoring matrix $M$ by the length of the alignments: $\bar{M}_l[i][j] = \frac{M_l[i][j]}{length(i,j)}$. The Sequence Alignment Algorithm is described in figure .

The result of the VSA is an ordered list of aligned subsequences, 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.

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.

**Figure 4. VSA: Video Sequence Alignment algorithm**

## 5 Rushes video parsing

Every scene is generally recorded in several takes (different versions for the same scene). We parse rushes video into scenes depending on the alignments that have been found by the VSA and we remove false alignment by the fact that two aligned sub-sequences must belong to the same scene.

### 5.1 Alignment matrix

Our alignment matrix is a matrix of scores which express the confidence of the alignment between two frames. A video sequence is defined as a list of frames $V = f_1 ... f_n$ (we now work at the frame level). 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.

### 5.2 Scene detection

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 alignment matrix. We use a recursive method: we seach the best scene boundary, and we repeat this process on the two sides of the boundary until we do not find any.

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:

$$rect(f) = \frac{\sum_{\forall f1 \in [first, f]} \sum_{\forall f2 \in [f, last]} A[f1][f2]}{\sum_{\forall f1 \in [first, f]} \sum_{\forall f2 \in [f, last]} 1}$$

We search the frame $f \in [first, last]$ maximizing the value of $rect(f)$, and if this value is greater than a threshold, $f$ delimits a scene transition and we restart the process on the two sides of $f$. 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. When no rectangle can be found, the decomposition into scenes is complete. We remove false alignments, i.e. inter scene alignments. Figure **??** shows a video scene decomposition.

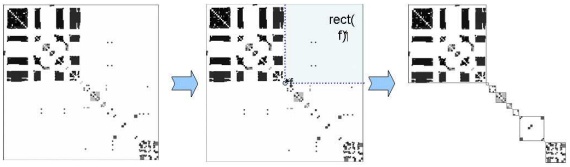


**Figure 5. Video scene decomposition**

### 5.3 Take selection

To select the best take for a given scene, we note the following comments:

- 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.

# 6 Experimental results

## 6.1 Protocol

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 **??**. 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.

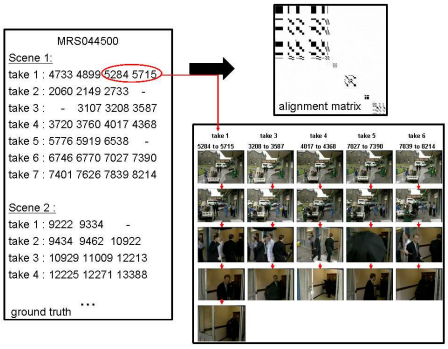


**Figure 6. Ground truth of video MRS044500, and alignment matrix corresponding with sample of aligned sub-sequences.**

To evaluate the VSA algorithm, we compare the alignments found with the ground truth data. When two aligned frames by the VSA belong to aligned take fragments in the ground truth, the frame alignment is considered as correct. This allows for some variability in the time alignment, which is required because consecutive frames are too similar to consider that only one frame-to-frame alignment is correct. Small variations around it are still perfectly valid. We use Recall and Precision rates as indicators for the performance of the VSA. The recall rate is the ratio between the number of correct pairs of frames aligned by VSA and the number of pairs of frames aligned in the ground truth. The precision rate is the ratio between the number of correct pairs of frames aligned and the total number of alignments found by VSA.
To evaluate the video parsing by scene boundaries detection, we compare the surface area of scenes found with the ground truth data. When two frames are in the same scene by scene detection and in the ground truth, this allocation of scene is considered as correct. So, we use Recall and Precision rates as indicators for the performance of the video parsing. The recall rate is the ratio between the number of correct pairs of frames allocated in the same scene by the scene detection algorithm and the number of pairs of frames allocated in the same scene in the ground truth. The precision rate is the ratio between the number of correct pairs of frames allocated in the same scene and the total number of allocations in the same scene by scene detection.

## 6.2 Results

We use 6 videos to fix thresholds: we perform a lot of tests by comparing results. In the scoring matrix, $match\_cost(\vec{i},\vec{j}) = cos(\vec{i},\vec{j}) + 1$ if $c_i = c_j$ and, $match\_cost(\vec{i},\vec{j}) = cos(\vec{i},\vec{j}) - 2$ else, and $gap\_cost = -3$. During VSA, the minimal confidence to valid an alignment of almost 2 one-second segment minimal is 1. And the scene detection threshold is 0.95.
Curve **??** shows the average precision-recall graphs on the 6 annotated videos, depending on the number of alignments found by the VSA (the VSA provides an ordered list of alignments anyway, so it is enough to consider the first N elements of the list). The precision and recall rates are computed at the frame level. We compared several variations of the VSA algorithm. If we do not use normalization, the results change very little when compared with the regular VSA. In the VSA without dynamic level, we fix the

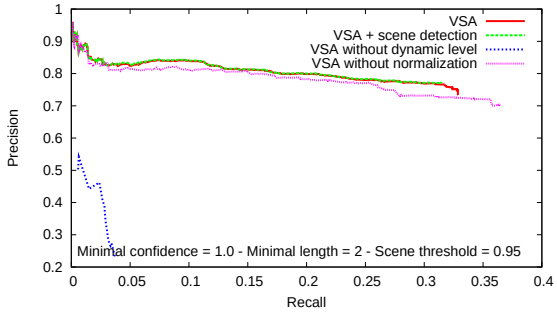


**Figure 7. Precision-Recall graph for variations of the VSA**

hierarchical clustering level. This reduces the complexity of the algorithm, but also greatly reduces the performance. We get similar effect if we remove both normalization and dynamic effect. Finally, if we filter the alignments using scene detection, we slightly improve the performance.
Figure **??** 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 [**?**].

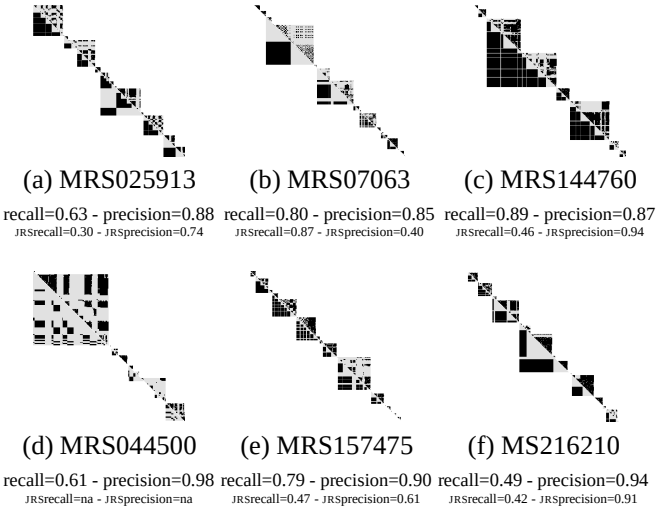


**Figure 8. Matrix alignment**

Table **??** shows results on the $8$ test videos. About the alignment, the recall varies between $0.154$ and $0.444$, and the precision between $0.422$ and $0.762$. These results are correct according to the method evaluation based on the area. For scene detection, results are good : precision varies between $0.771$ and $0.978$, and the recall between $0.554$ and $0.932$.

| | 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**

## 7 Conclusion

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.

## References


[1] W. Bailer, F. Lee, and G. Thallinger. A distance measure for repeated takes of one scene. *The Visual Computer*, 25:53–68, 2009.

[2] 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.

[3] 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.

[4] E. Dumont and B. Mérialdo. Split-screen dynamically accelerated video summaries. In *MM 2007, 15th international ACM conference on multimedia, September 24-29, 2007, Augsburg, Germany*, 2007.

[5] 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, London, UK , 2008.

[6] A. G. Hauptmann, M. G. Christel, W.-H. Lin, B. Maher, J. Yang, R. V. Baron, and G. Xiang. Clever clustering vs. simple speed-up for summarizing rushes. In *MM 2007, 15th international ACM conference on multimedia, September 24-29, Augsburg, Germany*, pages 20–24, 2007.

[7] 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.

[8] 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.

[9] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 1970.

[10] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 1981.

[11] B. T. Truong and S. Venkatesh. Video abstraction: A systematic review and classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2007.