Multimedia Tools And Applications manuscript No.
(will be inserted by the editor)

STIMO: STIll and MOving Video Storyboard for the Web Scenario

Marco Furini[1], Filippo Geraci[2], Manuela Montangero[3], Marco Pellegrini[4]

1 - Università di Modena e Reggio Emilia, Reggio Emilia. marco.furini@unimore.it
2 - IIT-CNR Institute, Pisa. filippo.geraci@iit.cnr.it
3 - Università di Modena e Reggio Emilia, Modena. montangero.manuela@unimo.it
4 - IIT-CNR Institute, Pisa. marco.pellegrini@iit.cnr.it

Received: date / Revised version: date

Abstract In the current Web scenario a video browsing tool that produces on-the-fly storyboards is more and more a need. Video summary techniques can be helpful but, due to their long processing time, they are usually unsuitable for on-the-fly usage. Therefore, it is common to produce storyboards in advance, penalizing users customization. The lack of customization is more and more critical, as the number of videos is increasing day after day, users have different demands and might access the Web with several different networking and device technologies. In this paper we propose STIMO, a summarization technique designed to produce on-the-fly video storyboards. STIMO produces still and moving storyboards and allows advanced users customization (e.g., users can select the storyboard length and the maximum time they are willing to wait to get the storyboard). STIMO is based on a fast clustering algorithm that selects the most representative video contents using HSV frame color distribution. Experimental results show that STIMO produces storyboards with good quality and in a time that makes on-the-fly usage possible.

1 Introduction

The availability of digital video contents in the Web is growing at an exceptional speed and websites like YouTube and iTunes Video, where people can upload/download videos, are receiving an enormous success. In this scenario, a tool for video browsing that is based on video contents, rather than tags (not always available, coherent, or relevant), would be really appreciated. This tool should provide users with a concise video content representation

to give an idea of a video content, without having to watch it entirely, so that a user can decide whether to download/watch the entire video or not.

Recently, the production of a concise video content representation has been the goal of the so-called *video summarization techniques*, which can produce two different types of summary: *still storyboard* (a.k.a. *static video summary*), which is a collection of still video frames extracted from the original video, and *moving storyboard* (a.k.a. *dynamic video skimming,* **moving-image abstract, or summary sequence**), which is a collection of short video clips/shots **extracted from the original video, joined in a sequence, and played as a video clip**[1]. The main difference between these approaches is that the former does not preserve the time evolving nature of the video and does not include any aural information, whereas the latter, **being itself a video (but with significant shorter duration), potentially produces storyboards with higher expressiveness and information.**

The production of a storyboard involves two aspects: the *selection* of the relevant frames/shots to be displayed, and the *layout* of the selected frames/shots [11]. The former aspect is more global in nature involving a notion of global importance and representativeness, whereas the latter is closer in spirit to the general problem of data layout onto a spatial domain (i.e the pixels of the screen) or a temporal domain (i.e. multi-thread simultaneous visualization). Both aspects are important, but since they are to a large extend independent, in this paper we address the frame/shot selection problem.

In the literature, different summarization techniques have been proposed to select the most relevant frames/shots to be displayed: some are related to specific videos (e.g., sport videos), others make use of specific information (e.g., close-caption, user preferences) and others are designed for generic videos. Among this latter, the most common approach is to use clustering algorithms to group together frames with similar features (e.g., color distribution, motion vector, etc.) and to extract a limited number (in most cases, only one) of frames per cluster (e.g., [24,30,15,12,14,19], just to name a few).

Although existing techniques produce storyboards with acceptable quality, they are computationally expensive and very time consuming, as they are usually based on complicated clustering algorithms. For instance, in [19] the computation of the storyboard takes around ten times the video length. The problem of video summarization is, in fact, amenable to several trade offs concerning the computations to be done off-line and on-the-fly. At one end of the spectrum a summary is produced completely off-line, stored, and delivered to a user when requested. The drawback of this approach is the complete lack of user customization. At the other end of the spectrum a summary is produced on-the-fly, by analyzing and aggregating video frames based on specific low level features (e.g., color distribution). This approach

[1] An example of a popular moving storyboard is a movie trailer/preview.

allows for advanced customization, but unfortunately, existing methods are too slow for doing on-the-fly processing. An intermediate approach (e.g., the one proposed in [5]) involves both an off-line computation (the selection of an appropriate, and usually large, number of key frames/shots), and an on-the-fly computation (the selection, among the pre-selected set, of frames/shots taking into account user requirements). Although this approach offers some forms of user-customization, it is not flexible as the customization relies on the first off-line phase (e.g., a change in the similarity metrics would require the off-line phase to be recomputed).

In this paper we address the approach for advanced customization by proposing a very fast video summarization technique that attains on-the-fly performance with minimum pre-computation while retaining high quality output. We are convinced that advanced customization is becoming more and more important in the current Web scenario, where users have different resources and needs, where, for instance, a mobile user might want to receive a storyboard with few frames in order to save bandwidth, whereas a DSL-user may be looking for a specific video scene and might want a more detailed storyboard.

The contribution of this paper is STIMO (STIll and MOving storyboard), a summarization technique designed to produce on-the-fly still and moving storyboards. The mechanism is designed to offer customization: users can select the length of the storyboard and can specify the time they are willing to wait in order to have the storyboard. STIMO does not use any specific information beyond audio and video information (e.g., no close-caption, or user preferences) and is designed to summarize generic videos and to produce moving storyboards equipped with completely intelligible audio. STIMO core is composed of a procedure that computes the HSV color space distribution of all the video frames, and of fast clustering algorithm that groups together similar video frames **and, for each group, determines the most representative frame based on the color similarity**.

To investigate the performance of STIMO we set-up an experimental scenario considering different categories of video that differ in color, length, and motion terms (e.g., cartoon, talk-show, TV-show, and TV-news). The investigation compares two fundamental features for the on-the-fly production of storyboards: production time and storyboard quality. These features are compared against other summarization techniques (DT Summary [19], Open Video [21], and k-means[22]). Results show that STIMO produces storyboards with computational speed-up and quality that makes it suitable for on-the-fly production, whereas other summarization techniques are too time consuming for on-the-fly usage.

This paper extends and completes the work presented in [7], with the production of storyboards composed of moving images, with an in-depth discussion of the state-of-the-art literature on this subject, with an improvement of the technique to produce still storyboards, and with an extended evaluation of still storyboards.

The remainder of this paper is organized as follows. In Section 2 we briefly present related work in the area of video summarization; STIMO is presented in Section 3, whereas its evaluation is shown in Section 4. Conclusions are drawn in Section 5.

2 Related Work

A video storyboard gives a good representation of the original video if it contains little redundancy and if it gives equal attention to the same amount of contents. Redundancy is a key aspect in video summary, as not all the extracted key frames are important (or necessary) to convey the visual content of the video. In fact, since the selected key frames are content independent, video summarization techniques based on key frames may not be significant representatives of the video content [28]. To address the problem of producing redundant-free storyboards, different approaches have been proposed in the literature. In general, **as shown in Figure 1,** these can be classified according to several categorical axis: the data domain (generic, news, etc.), the features used (visual, audio, user-context, etc.), and the duration (defined a priori, a posteriori, or user-defined). **For instance, [28] focuses on news video and proposes an automatic video summarization scheme based on affinity propagation clustering and semantic content mining; [29] focuses on rush videos and proposes a mechanism based on multi-stage clustering mechanisms to reduce key frames redundancy; [23] proposes using adaptive clustering to summarize rush videos; [9] focuses on video clip and presents a summarization algorithm based on the affinity propagation clustering algorithm.** Since our approach does not pose any constraints on the type of video to summarize, in the following we review clustering techniques that can be applied to generic videos and that use only visual and audio features (e.g., no additional information like close-caption or user-preferences).

In [30] authors propose a clustering algorithm to group frames with similar color histogram features, but they define a priori the length of the storyboard; therefore, this may compromise the quality of the result [19]. [15] presents a partitioned clustering algorithm where the frames closest to cluster centroids are selected as key-frames. In [19] an automatic clustering algorithm based on Delaunay Triangulation (DT) is proposed; here frames are described through HSV color space distribution, whereas [14] uses local motion estimation and an algorithm based on the k-medoids clustering. In [15] clustering is used to produce dynamic storyboards: the mechanism works at frame level using a partitional clustering method applied to all video frames. The optimal number of clusters is determined via a cluster-validation analysis and key frames are selected as centroids of the clusters. Video shots, to which key frames belong, are concatenated to form the

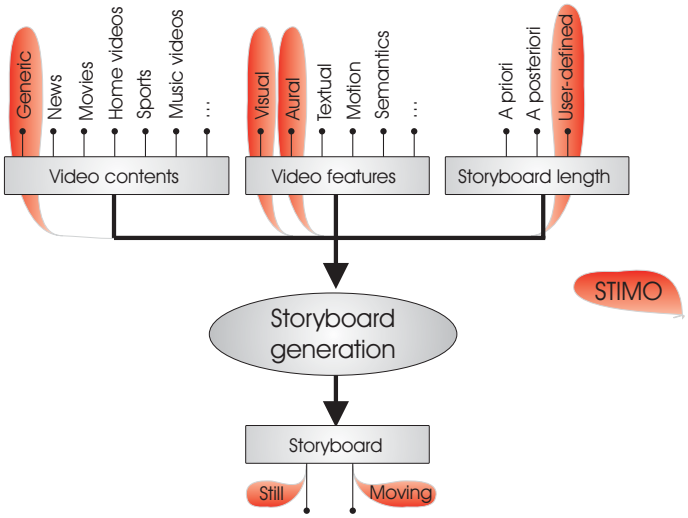


Fig. 1 Classification of video storyboard techniques. STIMO is designed to be a general video summarization technique; therefore it analyzes visual and aural features of generic video contents and produces still and moving storyboards whose length are defined by users.

moving sequence, making the dynamic efficiency/quality directly dependent from those of the static case. A different approach is considered in [25], where the problem of producing a moving storyboard is formulated as a partitioning problem over a *scene transition graph* where each node is associated to a shot, and an edge is set up if the similarity of two shots is higher than a pre-defined threshold. In this case, the clustering is done using an iterative block ordering method. In [8] a study analyzes benefits of using clustering techniques while producing moving storyboards: results showed that for some categories of video (e.g., news) a clustering approach may be not worth pursuing.

We refer the readers to [26] for an interesting survey of these and other different methods. Here, we highlight that none of the mechanisms for frame/shot selection claims to have the pure on-the-fly performance and the user-oriented flexibility that is needed in web video browsing applications. A limited customization is provided with the approach [5], where the storyboard is produced in two stages: first, the mechanism, once-for-all and off-line, selects an appropriate large set of key frames/shots, and secondly, the user can select a desired number of frames/shots to produce the storyboard. It is to note that this approach offers only a limited customization as it is based on the set of frames/shots pre-computed. For instance, a change in the similarity metrics would require the off-line phase to be recomputed. This is not reasonable if we consider that a single video sharing website may host millions of video (e.g., YouTube hosts more than 5 millions of videos).

The off-line production (either total or partial) is done to reduce the long clustering computational time, which can take up to ten times the video length (i.e., 20 minutes to summarize a 2 minutes video [19]). This enormous computational time is necessary to analyze all the characteristics of the video frames, usually stored in a huge bi-dimensional matrix that contains the characteristics of every video frames. Another approach to reduce the long clustering computational time is to reduce the size of the matrix with mathematic techniques (e.g., [12] applies the Singular Value Decomposition, whereas [19] uses the Principal Component Analysis), or sampling techniques (e.g., [19] and [20] consider only a sub-set of the video frames). Unfortunately, the former approach requires additional computational time, whereas the latter strongly affects the quality of the produced storyboard.

STIMO is designed to fill this gap. **The novelty of STIMO is that it produces on-the-fly storyboards of generic videos and it allows storyboard customization as users can select the length of the storyboard and can specify the time they are willing to wait in order to have the storyboard. Being designed to summarize generic videos, STIMO is based on the analysis of simple visual and aural information and it avoids exploiting additional video features (e.g., close-caption) as this would narrow the fields of application to specific videos (e.g., many videos are not provided with close-captions); at the same time, being designed to produce on-the-fly storyboards, STIMO does not analyze the semantic importance of each frame/shot, as this analysis would introduce an excessive computational overhead, not to mention that techniques that attempt to identify frames/shots that are semantically important may work well for some specific experimental settings, but since these methods strongly rely on heuristic rules drawn from empirical observation, they may be ineffective outside the tested sequences or specified domain [26]. Finally, it is worth mentioning that STIMO differs from proposals that aims at extracting frame/shot semantically important in order to better represent interesting events (e.g, movie highlights), as it is designed to give an extent visual coverage of the given video.**

3 Our proposal

In this section we present STIMO, a summarization technique designed to produce STIll and MOving storyboards for the Web scenario. In such a scenario, a pre-computed storyboard is usually un-desired as the Web is filled with people with different devices, technologies, resources and needs. Hence, for the same video, it is common to have a user who is seeking for a very detailed storyboard with a lot of frames, whereas another user may desire a storyboard with few frames. For this reason, STIMO is designed with the following goals: i) management of generic videos; ii) advanced users customization (storyboard length and the maximum time they are willing to

wait to get the storyboard); iii) usage of only audio and video information (e.g., no additional information like close-captions or user preferences) to produce the storyboard; iv) moving storyboards have to be coupled with completely intelligible audio; v) storyboards have to be produced in a reasonable time and with acceptable quality, so as to allow on-the-fly usage.

The core of STIMO is an improvement of the Farthest Point-First (FPF) algorithm [13,16], specifically modified for the case of video storyboard production. FPF is a clustering algorithm used to group together similar frames (based on the HSV color space distribution of the video frames), **and to select, for each group, the most representative frame**. The overall architecture of STIMO is shown in Figure 2. A still storyboard is produced by first extracting the HSV color description of each frame; second the clustering algorithm is applied to the extracted data and third, a post-processing phase removes possible redundant or meaningless video frames from the produced storyboard. A moving storyboard is produced by first segmenting the video in several video shots; secondly, the HSV color description of each shot is computed; third, the clustering algorithm is applied to the extracted data and, finally, the moving storyboard is built by sequencing the shots selected by the clustering algorithm.

In the following subsections we present details of these phases.

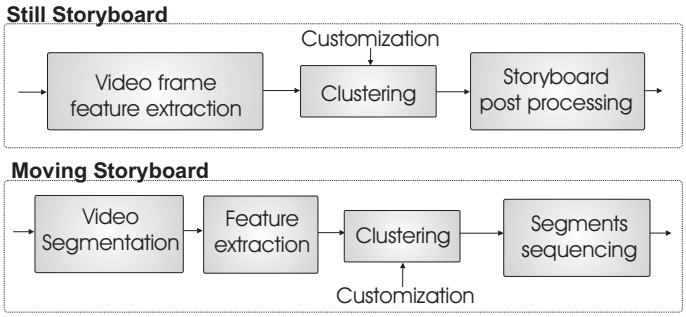


Fig. 2 The STIMO architecture to produce still/moving storyboards.

3.1 Video Frame Feature Extraction

This phase is used when producing still storyboards. Each video frame is described with a HSV histogram color distribution. This technique is simple and robust to small changes of the camera position and to camera partial occlusion, and is supported by the MPEG-7 standard. It defines the color space in terms of: Hue (the dominant spectral component, i.e., the color in its pure form), Saturation (the intensity of the color, represented by the quantity of white present), and Value (the brightness of the color). According to the MPEG7 generic color histogram description [18], in this paper, for

each input frame, we extract a 256-dimension vector, which represents the 256 bin colors histogram in the HSV color space of the given video frame. The vector is then stored in a matrix M_{HSV} for clustering purpose.

3.2 Video Segmentation and Feature Extraction

This phase is used when producing a moving storyboard. Being composed of a sequence of the most important segments of the original video, the quality of a moving storyboard depends on how segments are obtained. In particular, since STIMO is designed to produce moving storyboards with completely intelligible audio, it is of fundamental importance that the audio of a video segment is not truncated. To this end, it uses a video segmentation process that considers both audio and video features. In fact, if video is divided according only to visual information (for instance by splitting the video where there is a video cut, which happens when two consecutive video frames have few parts in common), it is likely that a video segment has a truncated audio. To avoid this, STIMO uses the approach presented in [6]: when a video cut is detected, audio energy at video transition is checked: if there is silence, the transition is considered to be the end of a segment, otherwise it is assumed that the segment is not over. As a result, the audio of every segment is not truncated, and, when sequencing segments, we get a fluid, understandable moving storyboard in which audio is completely intelligible.

For each video segment, we extract a 256-dimension vector that represents the average 256 bin colors histogram in the HSV color space of the video segment, as described in [25]. The vector is then stored in a matrix M_{avgHSV} for clustering purpose. This approach is called *scene-based* and is in contrast with the *frame-based* approach, where video segments are selected according to the most representative video frames selected. In this paper we consider the scene-based approach, as, after an initial experimental investigation, results obtained from using the frame-based approach were much worse than the one of the scene-based approach.

3.3 Clustering Algorithm to Produce Storyboards

This phase consists of a clustering algorithm that groups together similar frames (segments) and selects a representative frame (segment) per each group, so as to produce a still (moving) storyboard.

We recall that, given a set N of elements and a way to measure distance between elements (or similarity, in a dual approach), a k-clustering is a partition of N into k sets (called clusters) such that close elements are in the same cluster, whereas distant elements are in different clusters.

The general approach to produce a storyboard using a clustering algorithm can be described as follows: select a clustering algorithm and cluster frames/segments specifying a distance measure; give a method to select one

key-frame/segment per cluster and place the selected frames/segments in the storyboard. Needless to say, the selection of the clustering algorithm, of the distance function, and of the key-frame/segment within a single cluster is very important as it affects both the quality of the storyboard and the time necessary to produce the storyboard.

The engine of STIMO is a clustering algorithm that approaches the problem of clustering video frames/segments as that of finding a solution to the classic k-center problem:

Given a set S of points in a metric space M endowed with a metric distance function D, and given a desired number k of resulting clusters, partition S into clusters $C_1, \ldots, C_k$ and determine their "centers" $c_1, \ldots, c_k \in S$ so that the radius of the widest cluster, $\max_j \max_{p \in C_j} D(p, c_j)$, is minimized.

In our scenario, the metric space M is $\Re^{256}$, the set S is the frame feature matrix M_{HSV} in the static case and M_{avgHSV} in the dynamic one, and the distance function D is given by the *Generalized Jaccard Distance* (GJD) [2] defined as follows: given two vectors with non-negative components $s = (s_1, ...s_h)$ and $z = (z_1, ...z_h)$, the GJD is given by

$$D(s, z) = 1 - \frac{\sum_i \min(s_i, z_i)}{\sum_i \max(s_i, z_i)}.$$

The choice of the specific metric that implements the informal notion of similarity is critical to obtain a good clustering. We tested several classical metrics (e.g. Euclidean, City block, Cosine), and the metric that exhibits the best discriminative power in our setting is the GJD.

The k-center problem is known to be NP-hard [4], but it can be 2-approximated using the Furthest-Point-First algorithm [13,16]. We use a new variation of the FPF algorithm that we describe in the following.

Basic Algorithm and Variant Given the set S of n points, FPF increasingly computes the set of centers $C_1 \subset \ldots \subset C_k \subseteq S$, where C_k is the solution to the problem and $C_1 = \{c_1\}$ is the starting set, built by randomly choosing c_1 in S. At a generic iteration $1 < i \leq k$, the algorithm knows the set of centers C_{i-1} (computed at the previous iteration) and a mapping μ that associates, to each point $p \in S$, its closest center $\mu(p) \in C_{i-1}$. Iteration i consists of the following two steps:

1. Find the point $p \in S$ for which the distance to its closest center, $D(p, \mu(p))$, is maximum; make p a new center c_i and let $C_i = C_{i-1} \cup \{c_i\}$.
2. Compute the distance of c_i to all points in $S \backslash C_{i-1}$ to update the mapping μ of points to their closest center.

After k iterations, the set of centers $C_k = \{c_1, \ldots, c_k\}$ and mapping μ define the clustering: cluster $\mathcal{C}_i$ is defined as the set $\{p \in S \setminus C_k \mid \mu(p) = c_i\}$, for $i = 1, \ldots, k$. The overall cost of the algorithm is $O(kn)$ and experiments have shown that the random choice of c_1 to initialize C_1 does not affect neither the effectiveness nor the efficiency of the algorithm.

An improved version of FPF, called M-FPF [10], exploits the triangular inequality in order to filter out useless distance computations and speed up step two of the original algorithm. Moreover, the efficiency of the algorithm is further improved by applying it only to a random sample of size $\sqrt{nk}$ of the input points (sample size suggested in [17]) and adding the remaining points one by one. The completion of clusters is guided by medoids, rather than centers: points are placed in the cluster with closest medoid, which is updated after each new insertion. We recall that, given a cluster and a diametral pair a, b (*i.e.*, a pair of points with maximum distance), the medoid is the point that minimizes the quantity $M(x) = |D(a,x) - D(b,x)| + |D(a,x) + D(b,x)|$, over all points x in the cluster.

STIMO algorithms The process of adding points to clusters, and consequently update medoids, is very time consuming. To reduce this processing time we introduce a new heuristic that is based on the computation of a good approximation of the medoid.

The algorithm that **selects the most representative frames to produce a still storyboard of k frames works** in the following way:

a) Apply FPF to a random sample of size $\sqrt{nk}$ and obtain clusters $\mathcal{C}_i, 1 \leq i \leq k$;
b) within each cluster $\mathcal{C}_i$ determine (1) the point a_i furthest from the center c_i; (2) the point b_i furthest from a_i (intuitively the pair (a_i, b_i) is a good approximation to a diametral pair); (3) the medoid m_i;
c) add the remaining points one by one to the cluster with closest medoid and approximately update medoids and diametral pairs in the following way: if the newly added point p falls in between the diametral pair and if it is a better medoid than the current one (*i.e.*, $D(p, m_i) < \min\{D(m_i, a_i), D(m_i, b_i)\}$ and $M(p) < M(m_i)$), then update m_i by setting it to be p. Otherwise, if the new point is outside the approximate diametral pair (a_i, b_i) (*i.e.*, $D(a_i, b_i) < \max\{D(p, a_i), D(p, b_i)\}$), the pair is updated accordingly.

   Step (c) is further enhanced using the following ad-hoc heuristic: given two points p and p' that represent two consecutive frames, if their distance is under an appropriate given threshold, with high probability the two points belong to the same cluster. Hence, whenever $D(p, p') \leq 0.2$[2] we simply place p' in the same cluster of p and proceed to update medoids and diametral pair.
   This heuristic is not guaranteed to work well for all applications, but in the case of video frames it does not affect the quality of the result: we clustered the same datasets by using and not using this heuristic and we observed that the resulting clusterings show practically no differences, whereas the time needed to produce the clusterings decreases drastically.

[2] The threshold is determinated on a statistical base looking at distances between very similar frames.

When the clustering is completed, medoids are selected as representative elements to be in the storyboard.

To produce a moving storyboard of length T, we apply the FPF algorithm, enhanced using triangular inequality, to segments. At each iteration, the algorithm generates a new permanent center of a new cluster, giving a way to rank centers, *i.e.*, a selection order, completely independent from the order in which the segments appear in the original video. Hence, at the time in which centers are created, the corresponding segments are **considered representative, are** selected and inserted in the moving storyboard. The process continues until the total length reaches the desired time T.

Storyboard Customization STIMO allows users to select the length of the storyboard and the maximum time they are willing to wait in order to get the storyboard.

The storyboard length is specified either in number of frames (for still storyboards), or in seconds (for moving storyboards). However, we can not exclude the case in which a user has no idea of what such a number might be. STIMO helps the user by suggesting a possible value. For still storyboards, STIMO computes the number of abrupt scene changes applying a fast shot boundary detection to the HSV matrix and suggests this number unless it is higher than 30 (on a statistical base looking at several storyboards, 30 video frames are considered sufficient to understand the video content). Note that the computation of abrupt scene change takes a negligible overhead. For moving storyboards, STIMO suggests 1 or 2 minutes (these lengths are usually used in entertainment for previews and video recaps). However, with STIMO users are free to select the desired storyboard length.

STIMO allows users to specify the maximum time they are willing to wait to get the storyboard. This feature is offered as the storyboard production time depends on the original video length: the longer the video is, the longer the production time will be. Studies on users behavior showed that waiting time is critical. For instance in the Web scenario, up to five seconds to get a complete webpage is considered a good waiting time, whereas over 10 seconds is poor. However, if the webpage loads incrementally, waiting time up to 39 seconds are good [1]. Since storyboards are produced for the Web, we consider this threshold a reasonable waiting time to produce a storyboard.

STIMO estimates the time necessary to produce the storyboard, but gives the user the possibility to specify the maximum time he/she is willing to wait. **The estimation process exploits the knowledge that the storyboard production time is proportional to the number of distances the algorithm has to compute. By considering that every frame is characterized with the same number of bins, the computation of the distance between two frames requires a fixed amount of time. It follows that to estimate the time necessary to produce the storyboard, STIMO multiplies the cost of a distance computation with the total number of distances required to complete**

the clustering (i.e., nk, where n is the video length in frames, and k is the storyboard length).

It is worth noting that the storyboard production time might be too long in the still case, as the matrix to cluster might be very huge. This is why STIMO gives the user the possibility to specify the maximum time he/she is willing to wait. Using this maximum waiting time, STIMO takes advantage of redundancies among the number of frames per second of the input video (e.g., 25 fps) so as to reduce the number of frames to analyze (i.e., a pre-sampling technique is applied to the matrix containing the video frame characteristics; in particular, STIMO selects the lower sampling rate in order to meet the requested time-constraint). Needless to say, the user is warned that the shorter the waiting time is (i.e., the higher the sampling rate is), the poorer results might be.

3.4 Still Storyboard Post Processing

To avoid the presence of possible meaningless video frames in the final storyboard (e.g., mono-color frame due to fade-in fade-out effect or to the use of flashes very common in sport videos or in news video), STIMO uses the HSV color distribution to identify and remove such frames. This investigation consumes a negligible time, as the number of selected frames is usually very small.

3.5 Moving Storyboard: Segment Sequencing

This phase actually organizes the selected scenes according to the time in which they appear in the original video and the resulting sequence represents the moving storyboards. Note that, thanks to the used video segmentation process, moving storyboards have completely intelligible (not truncated) audio.

4 STIMO Evaluation

To produce on-the-fly storyboards two parameters are of fundamental importance: the time necessary to produce the storyboard and the quality of the produced storyboard. Therefore, we setup an experimental scenario where STIMO and other approaches extract storyboards from videos with different characteristics. In particular, the benchmark is composed of short and long videos: short videos are taken from [3] and are a subset of videos available within the Open Video Project [21], whereas long videos are chosen to cover different color, length, and motion characteristics. The set of long videos is composed of cartoons (*The Simpsons*, *Futurama*, and *Family Guy*),

TV-Shows (*Charmed, Roswell*, and *Lost*), TV-News (*Sky TV* and *BBC*), and talk-show (*MTV* and *RAI*)[3]. All the videos are MPEG-1 encoded.

STIMO is compared against DT Summary [19], Open Video Project [21], and k-means [22] when possible. It is worth mentioning that we considered k-means as well representative of a large class of general purpose clustering algorithms whose time/quality trade-off are appropriate for our problem. Other methods, like HAC (Hierarchical Agglomerative Clustering), might also be used to produce storyboards, but they are clearly too slow (being n the number of frames and k the length of the storyboard, with $n \gg k$ in our problem, HAC runs in time $O(n^2)$, whereas k-means runs in time $O(nk)$). For these reasons, it is not worth comparing STIMO against clustering methods other than k-means. Although the source code of DT Summary and Open Video Project is not available, we considered these approaches because several short videos and related still storyboards are accessible; therefore storyboards comparison is possible. In addition, STIMO is compared against a RANDOM approach that randomly produces still and moving storyboards, so as to investigate whether clustering techniques might help in producing storyboards or not.

To evaluate the storyboard production time, we use a general purpose computer (Pentium D 3.4 GHz equipped with 3GB RAM). Although more sophisticated (and expensive) hardware would decrease the storyboard production time, it is to note that the goal of this investigation is to study the relative difference among the compared approaches, and not the absolute production time.

Since a consistent evaluation framework for video summarization quality is seriously missing [26], to evaluate the storyboard quality we consider the Mean Opinion Score (MOS) test. The MOS test is a widely used technique that gives an indication of the storyboard quality by performing subjective tests to a number of people and by averaging the obtained results. As long as the obtained results do not present a large statistical difference, the MOS test can be considered effective in measuring the quality of the video storyboard. To evaluate the STIMO storyboards, we consider a group of 20 people and we ask them to score the storyboard quality on a scale from 1 (bad quality) to 5 (good quality). In particular, the group of evaluators is composed of people from both academic and private sectors, with different background (faculty, Ph.D. students, grad students, employees in the fields of computer science, engineering, and social sciences) and with no previous experience in storyboarding systems. During the evaluation, we observed that MOS results of still storyboards do not present a large statistical difference, whereas for moving storyboards there was a considerable

[3] Movies have not been considered since storyboards reveal too much contents (e.g, the end of the movie), and hence ad-hoc techniques to produce *highlights* are more suited for this category.

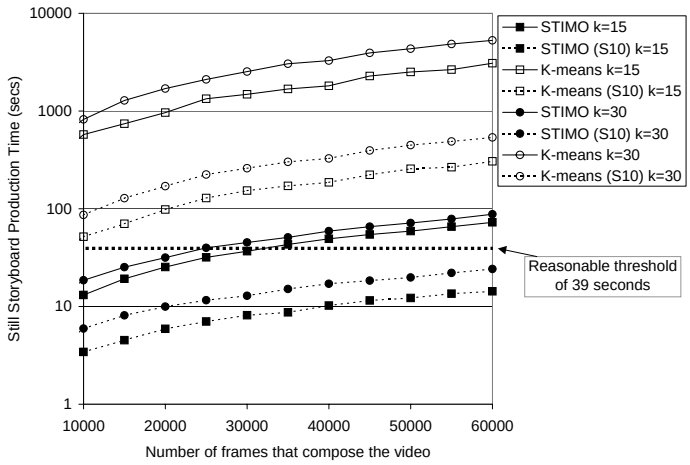


Fig. 3 Still storyboard (15 and 30 frames long) production time of long videos: k-means vs. STIMO. The reasonable threshold is taken from [1]. S10 denotes a pre-sampling of 1 out 10 on the given video. [Logarithmic Scale].

statistical difference. For this reason, the evaluation of moving storyboards is done with a ground-truth evaluation. **In particular, instead of scoring the storyboards, we produce a ground-truth reference video by splitting the original video into several segments, and by asking our evaluators to score such segments based on the semantic importance they have; successively, we use the ground-truth reference video to rate the produced storyboard.**

4.1 Evaluation of Still Storyboards

As a first investigation we consider short videos taken from [3]. Results show that STIMO, on average, is 25 times faster than k-means and 300 times faster than DT[4].

Figure 3 presents a deeper investigation where we vary the length of the video to summarize from 10000 frames (400 seconds) to 60000 frames (40 minutes), the length of the produced storyboard (15 and 30 frames), and the rate of the pre-sampling (none and 1 out of 10) that is applied to the video frame feature matrix. We compare STIMO against k-means, as the code of other approaches is not available. RANDOM production time is not presented as it is negligible. Results are related to TV-show videos, but those of other video categories are similar and therefore are not presented here (this is not surprising as the the production time only depends on the number of frames that compose the video).

[4] Results of DT are simply estimated considering that the mechanism requires between 9-10 times the video length to produce the summary [19]

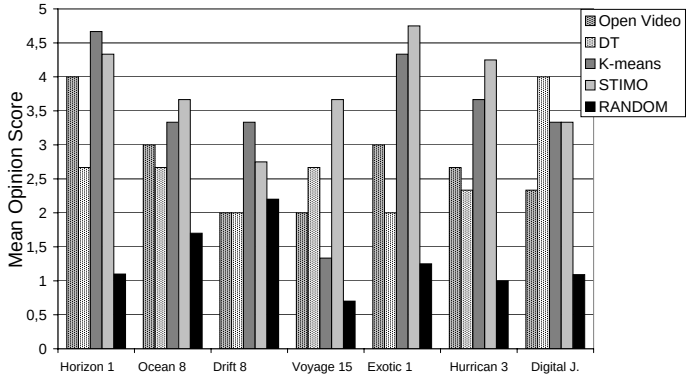


Fig. 4 MOS evaluation of some still storyboards extracted from short videos taken from [3].

With no surprise, the storyboard production time depends on its length (the longer the storyboard is, the longer is the computational time) and on the pre-sampling rate (the higher the sampling rate is, the shorter is the computational time). This applies to both STIMO and k-means.

For on-the-fly usage, by considering a maximum waiting time of 39 seconds [1], k-means is out of the game unless using a higher pre-sampling. STIMO with pre-sampling of 1 out of 10 can be used, as well as STIMO without pre-sampling for videos up to 30000 frames (20 minutes). **We recall here that the storyboard production time strictly depends on the computational power of the employed hardware. Results presented in this paper have been obtained producing storyboards with a general purpose computer (Pentium D 3.4 GHz equipped with 3GB RAM). Needless to say, the employment of faster hardware would decrease the production time. However, the obtained results are useful to compare the storyboard production time of different methods in order to understand the relative difference among the compared approaches.**

Figure 12 better highlights the speed-up comparison between STIMO and k-means for the production of storyboards of 30 frames: on average, STIMO is 55 times faster than k-means (20 times faster if using pre-sampling of 1 out of 10). For storyboards of 15 frames, STIMO is 40 times faster than k-means (18 times faster if using pre-sampling of 1 out of 10).

To investigate the quality of the storyboards, we compare STIMO against Open Video Project, DT Summary, and k-means, for the short videos taken from [3]. On the other hand, for long videos we compare STIMO only against k-means since the source code of Open Video Project and of DT Summary is not available.

As earlier mentioned, quality evaluation is investigated through a Mean Opinion Score test. The procedure is the following: we first show the video to the evaluators and then the storyboard asking whether the storyboard is

a good representation of the original video. The summary quality is scored on a scale 1 to 5 (1=bad, 2=poor, 3=fair, 4=good, 5=excellent) and people are not aware of the mechanism used to produce the storyboard. The length of the storyboard is set in order to match the other approaches (i.e., if the Open Video storyboard is of 5 frames, the length of the STIMO storyboard is set to 5 frames, too).

Figure 4 reports the average MOS results obtained from evaluating short videos taken from [3] (the MOS scores produced no significant statistical difference). DT, Open Video and RANDOM achieve poor results for most of the videos, whereas STIMO has the highest score for *Hurricane 3*, *Exotic 1*, and *Voyage 15*. With respect to the remaining videos, STIMO and k-means achieve comparable results.

Figure 5 reports the average MOS results obtained from evaluating different categories of long videos. In this case the comparison is against k-means due to source code availability. For any given video, the evaluation process considered two different storyboards: one of 15 frames and the other 30 frames. The RANDOM approach achieves the worst results. STIMO and k-means achieve comparable scores for *TV-show* and for *TV-news*; k-means gets higher scores for *cartoons* video; STIMO gets higher scores for *talk-show* videos. **Figures 6-9 show the storyboards of 15 frames produced by STIMO and k-means for the considered categories of videos: talk-show, cartoon, TV-news, and TV-show. In particular, Figure 6 shows the storyboards of a talk-show video (the category of videos where STIMO gets higher scores); here, k-means produces a storyboard with several frames that contain the same subject, whereas STIMO gives a more comprehensive overview of the people participating to the talk show.**

To summarize, the evaluation of still storyboards shows that STIMO is faster than k-means (from 25 times to 55 times by analyzing all the frames that compose a video; from 18 to 20 time by analyzing 1 out of 10 frames that compose a video). With the hardware used to perform the evaluation, STIMO could be used to generate on-the-fly still storyboards, whereas k-means can be used only if a more powerful hardware (up to 55 times more powerful) is available. The quality investigation shows that the great computational speed-up does not compromise the storyboard quality. Note that although the RANDOM approach is the fastest one, it produces storyboards with poor quality.

4.2 Evaluation of Moving Storyboards

Figure 10 presents the time necessary to produce moving storyboards of different categories of video. We present here the case of 2 minutes long moving storyboards (results of 1 minute storyboards have a similar shape) produced with STIMO and with k-means. The RANDOM approach is not reported as it requires a negligible computational time.

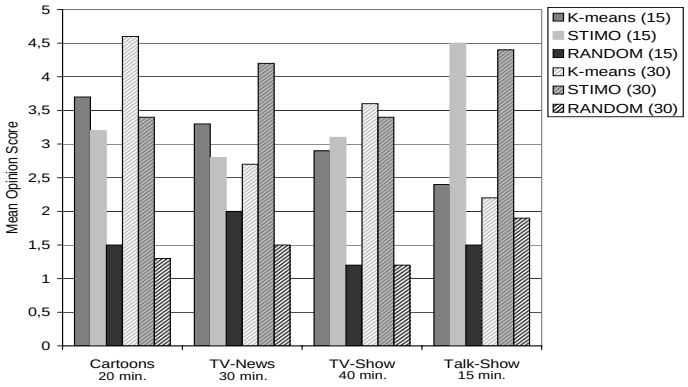


Fig. 5 MOS evaluation of still storyboards (15 and 30 frames long) extracted from long videos.

Looking at the chart, it can be noticed that production time does not necessarily increase with increasing video length: it might take less to produce a storyboard of a longer video; *e.g.*, it takes less to get the storyboard of a 45000 frames cartoon video, than of a 35000 frames cartoon video. Although this may appear strange, it is not. In fact, the HSV matrix is composed of vectors that represent the average HSV distribution of a video segment, and not of a single video frame. Therefore, since videos of the same length are likely to have a different number of video segments and video segments have different lengths, the size of the HSV matrix varies even for videos of same length. It might also happen that moving storyboards of the same length for different videos are composed of a different number of video segments. For instance, to generate the storyboard of a cartoon video 35000 frames long k-means needs to produce 29 clusters, whereas only 4 clusters are necessary to generate the storyboard of a cartoon video 45000 frames long.

On average, STIMO results 5 times faster than k-means (details concerning different categories of videos are shown in Figure 12). In addition, it is to note that to produce a moving storyboard of a desired length, k-means needs to know in advance the number of clusters k. Unfortunately, the selection of k might be very time-consuming as there is no way of knowing k in advance as it is not easy to associate the number of clusters with scene lengths; therefore, the choice of k may be incorrect. In this case, the clustering algorithms should not be obliged to start over again if the choice of k results incorrect. STIMO allows interrupting the clustering procedure without any problem, whereas k-means does not (the computation has to be redone from the beginning). The possible solution of cutting the clustering tree at the most suitable point is of no help since the clustering tree internal nodes are not correlated with scene lengths and a movie storyboard made of $k + 1$ scenes can be shorter than one made with k scenes.

STIMO K-means

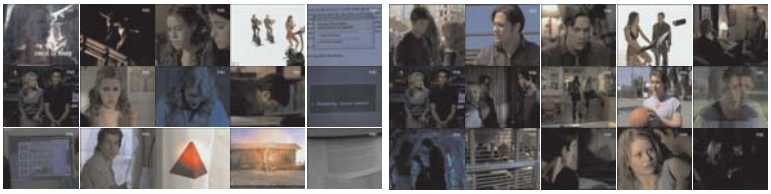
Fig. 6 Still storyboards of 15 frames extracted from a talk-show video.

STIMO K-means

Fig. 7 Still storyboards of 15 frames extracted from a cartoon video.

STIMO K-means

Fig. 8 Still storyboards of 15 frames extracted from a TV-news video.



STIMO K-means

Fig. 9 Still storyboards of 15 frames extracted from a TV-show video.

To investigate the quality of the produced storyboards, we consider a ground-truth evaluation. The choice is motivated by the facts that objective metrics like PSNR cannot be applied to moving storyboards as they are of different lengths, and that by using MOS we observed results with large statistical difference.

To get a ground-truth evaluation, **we considered the same group of evaluators (20 people from both academic and private sectors, with different background in the fields of computer science, engineering, and social sciences and with no previous experience in storyboarding systems) and we proceed by first creating a ground-**

truth reference video, and then by rating the produced storyboard with respect to the ground-truth reference video. In particular, we proceed as follows:

1. Build-up of the ground-truth reference video
a) Given the original video, we manually divide it into *Super-Scenes* (s-scene) each having a self contained meaning (*e.g.*, dialog between two characters in the kitchen). A s-scene might contain more than one video segment or can be a fraction of a segment (e.g., two different actions taking place during one single background piece of music).
b) We ask our evaluators to score each s-scene with a value from zero to five (0 = un-influent, 5 = fundamental). Then, to each s-scene we associate a score that is computed as the average of the scores given by the users.

2. Rating of the produced storyboards
Given a storyboard, each scene of the storyboard is scored with the score given to the s-scene it belongs to or with the sum of the scores given to the scenes it is composed of. The storyboard receives a score equal to the sum of the scores of the scenes it is composed of.

Note that the rating of the produced storyboards uses the evaluation done in step 1.b) and therefore several storyboards of the same video can be evaluated without incurring into a superficial scoring due to evaluators' boredom.

During the process of building the ground-truth reference video, we noticed that the ground-truth approach presented a relevant statistical difference in TV-news video, whereas more homogeneous scores have been given for cartoon, talk-show, and TV-show video. Looking deeply at the characteristics of the analyzed videos, we noticed that TV-news videos present multiple storyline, each one presented in a different video clip. We found that the lack of a single storyline causes the large statistical difference in the evaluators' scores. In fact, it may happen that a piece of TV-news (say a soccer news) is evaluated as very important by a soccer fan, but meaningless by his wife. Conversely, when there is a single (or few) storyline, evaluators tend to score the video storyline and not their specific and subjective interests.

Figure 11 reports the ratings obtained from evaluating the storyboards with respect to the ground-truth reference video. It surprises that the storyboards produced with the RANDOM approach achieve considerable scores (even better than k-means). In particular, if one considers that the RANDOM approach needs a negligible production time, it follows that clustering techniques are not worth using for TV-news and for talk-show. Conversely, for cartoon and TV-show videos, clustering algorithms introduce benefits and STIMO achieves higher scores that k-means.

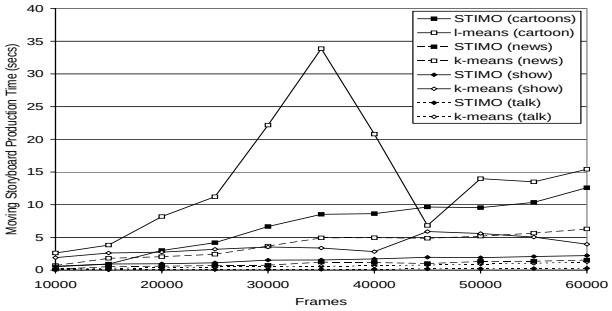


Fig. 10 Moving storyboards: Production time analysis.

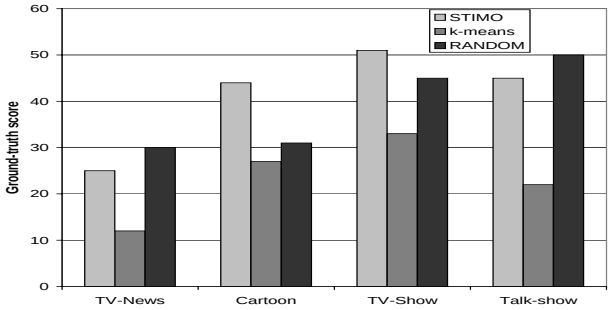


Fig. 11 Ground truth evaluation of moving storyboards (the higher the better).

In particular, results show that the RANDOM generation of storyboards is effective when applied to video with multiple storyline (e.g., TV-news and talk-show), but less effective when applied to single storyline video (e.g., cartoon and TV-show). The reason is that when dealing with multiple storyline videos, a user is usually satisfied if the produced storyboard represents as many storyline as possible. In this case, a good storyboard is the one similar to a highlight storyboard than the one that gives an extent visual coverage of the given video. The RANDOM approach, by uniformly selecting shots of the video, is likely to better highlight the multiple storyline contained in the given video. Conversely, when a video has a single storyline the uniform selection of shots may produce storyboards with poor quality as many shots may be semantically meaningless (from the storyline point of view). In this case, clustering techniques produce better results as they better group and select the most representative shots of the given video.

4.3 Still or moving storyboards?

A still storyboard provides the user with a fast indication of what the video is about, but neglects the available motion information in video. A moving storyboard provides the user with a video composed of a sequence of short video clips, but to have an insight of the entire video contents, the whole video has to be watched. Therefore, whether it is better to summarize a video with a still or with a moving storyboard is not a trivial choice.

As pointed out in [27], the selection does not only depend on content (still or moving), but also on the user preferences, on the context (e.g., mobile or fixed scenario), on the reason (e.g., video browsing to kill time or to look for something), and on the used system (e.g., smartphone, netbook, or desktop with 20 inches monitor). For instance, a user who is looking for something may prefer having a fast insight of the video contents, whereas a user who is browsing to kill time may prefer a moving storyboard; a user with a cellphone connection may prefer still storyboards to save bandwidth (and money), whereas a user with DSL residential flat-rate plans may prefer moving storyboards; a user with a device that has a small display may prefer moving storyboards, whereas a user with a desktop computer may prefer watching a set of still pictures.

To grant users' wishes, STIMO leaves the selection to them. In fact, thanks to the high speed-up of STIMO, users can have both types of storyboards in a reasonable time: a still storyboard to have a fast insight of the video content, and a moving storyboard to enjoy the motion and aural information of the video content.

4.4 Summary of Results

Although experiments have been performed with an entry-level computer, results showed that STIMO produces storyboard in a time that makes it suitable for on-the-fly usage. Since the production time is hardware dependent (with faster hardware the computational speed increases and the storyboard production time decreases), a relative comparison between STIMO and k-means is worth analyzing. For the sake of clarity, Figure 12 summarizes the comparison between STIMO and k-means with respect to the production time of storyboards (both moving and still). In particular, results present the ratio between the storyboard production time of k-means and the one of STIMO. It can be noted that, for the production of moving storyboards, STIMO, on average, results 5 times faster than k-means. By looking at the categories in detail, STIMO is 6.5 times faster while producing cartoon video storyboards, 5.3 times faster for talk-show videos, 4.6 times faster for

TV-show, and 3.5 times faster for TV-news. With respect to the production of still storyboards, STIMO is much more faster than k-means: 55 times faster when analyzing long videos, 25 times faster when analyzing short videos, and 20 times faster when analyzing 1 out of 10 frames of long videos.

The quality investigation showed that STIMO achieves comparable or better results than k-means in the production of still storyboards. The same applies to the production of moving storyboards. However, it is worth noting that for some categories of video (i.e., news and talk-show) both STIMO and k-means produced storyboards with poorer quality than a simple RANDOM approach. By looking at the characteristics of each category of video, we noticed that the RANDOM approach if effective when summarizing videos with multiple storyline as it produces a moving storyboard with a uniform coverage of the entire video. Conversely, when the video is composed of a single storyline (e.g., TV-news and cartoons), clustering techniques are worth using as they better group and select the most representative shots of the video. In this case, STIMO achieved better results than k-means.

5 Conclusions

In this paper we presented STIMO, a technique designed to produce still and moving storyboards in the Web scenario. The novelty of STIMO is the fast clustering algorithm designed to group the video frames according to the extracted HSV color space distribution. STIMO allows users to customize the produced output by specifying the length of the storyboard and the time they are willing to wait to get the storyboard. Figure 12 shows the speedup of STIMO with respect to k-means. A further investigation showed that the great speed-up does not compromise the storyboard quality (in most of the experiments STIMO achieved higher scores). Therefore, the evaluation showed that STIMO is an excellent tool for Web video browsing as it is able to generate on-the-fly storyboards that give users a quick overview of a video.

References

1. A. Bouch, A. Kuchinsky, and N. Bhatti, "Quality is in the eye of the beholder: Meeting users' requirements for Internet quality of service", in *Proc. of Conference on Human Factors in Computing Systems*, The Hague, Netherlands, April 2000, pp. 297–304.
2. M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of 34th Annual ACM Symposium on the Theory of Computing*, Montreal, CA, 2002, pp. 380–388.
3. "http://www.csee.umbc.edu/ỹongrao1/delaunay.html."

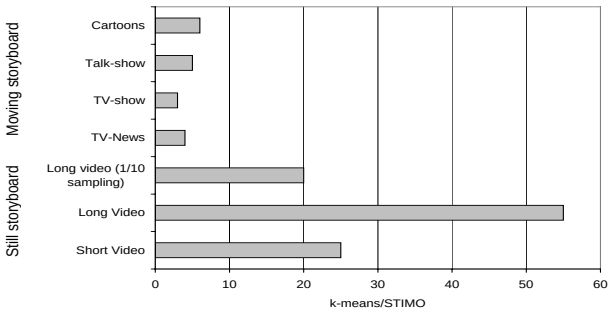


Fig. 12 Storyboard production time: ratio between k-means and STIMO for the production of storyboards of 2 minutes (moving) and of 30 frames (still). On average, STIMO is 5 times faster than k-means (still storyboards), 25 and 55 time faster for short and long videos, respectively (moving storyboards).

4. T. Feder and D. Greene, "Optimal algorithms for approximate clustering," in *Proceedings of the 28th ACM Symposium on Theory of computing*, 1988, pp. 434–444.
5. A. Müfit Ferman, A. Murat Tekalp, "Two-stage hierarchical video summary extraction to match low-level user browsing preferences," in *IEEE Transactions on Multimedia*, vol. 5, no. 2, 2003, pp. 244–256.
6. M. Furini, "On ameliorating the perceived playout quality in chunk-driven P2P media streaming systems," in *Proceedings of the IEEE International Conference on Communications (ICC)*, June 2007.
7. M. Furini, F. Geraci, M. Montangero, and M. Pellegrini, "VISTO: VIsual STOryboard for Web Video Browsing," in *In Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR 2007)*, Amsterdam, The Netherlands, July 9-11 2007, pp. 635 – 642.
8. M.Furini, F. Geraci, M. Montangero, M. Pellegrini, "On Using Clustering Algorithms to Produce Video Abstracts for the Web Scenario", in *Proceedings of the IEEE Consumer Communication & Networking 2008 (CCNC2008)*, Las Vegas NV, USA, 10-12 January 2008. IEEE Communication Society, 2008.
9. Y. Gao, Q.H. Dai, "Clip based video summarization and ranking," in *Proceedings of the 2008 ACM international conference on Content-based image and video retrieval (CIVR 2008)*, Niagara Falls, CA, 2008, pp. 135–140.
10. F. Geraci, M. Pellegrini, F. Sebastiani and M. Maggini, "Cluster Generation and Cluster Labeling for Web Snippets:A Fast and Accurate Hierarchical Solution", in *Internet Mathematics*, Vol. 3, No. 4, pp. 413-444, 2007.
11. A. Girgensohn, "A fast layout algorithm for visual video summaries,", in *Proceedings of IEEE International Conference on Multimedia & Expo (ICME)*, vol. 2, 2003, pp. 77–80.
12. Y. Gong and X. Liu, "Video summarization and retrieval using singular value decomposition," *Multimedia Syst.*, vol. 9, no. 2, pp. 157–168, 2003.
13. T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, no. 2/3, pp. 293–306, 1985.
14. Y. Hadi, F. Essannouni, and R. O. H. Thami, "Video summarization by k-medoid clustering," in *Proceedings of the ACM Symposium on Applied computing*, 2006, pp. 1400–1401.

15. A. Hanjalic and H. J. Zhang, "An integrated scheme for automated video abstraction based on unsupervised cluster-validity analysis," in *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 9, no. 8, 1999, pp. 1280–1289.
16. D. S. Hochbaum and D. B. Shmoys, "A best possible approximation algorithm for the k-center problem," *Mathematics of Operations Research*, vol. 10, no. 2, pp. 180–184, 1985.
17. P. Indyk, "Sublinear time algorithms for metric space problems," in *Proceedings of ACM Symposium on Theory of Computing*, 1999, pp. 428–434.
18. B. S. Manjunath, J. R. Ohm, V. V. Vasudevan, and A. Yamada, "Color and texture descriptors," *IEEE Transactions on Circuits and Systems For Video Technology*, vol. 11, pp. 703–715, 2001.
19. P. Mundur, Y. Rao, and Y. Yesha, "Keyframe-based video summarization using Delaunay clustering," *International Journal on Digital Libraries*, vol. 6, no. 2, pp. 219–232, 2006.
20. J. Nam and A. Tewfik, "Video abstract of video," in *IEEE 3rd Workshop on Multimedia Signal Processing*, 1999, pp. 117–122.
21. "The open video project, http://www.open-video.org."
22. S. J. Phillips, "Acceleration of k-means and related clustering algorithms," in *Proceedings of 4th International Workshop on Algorithm Engineering and Experiments*, San Francisco, US, 2002, pp. 166–177.
23. J. Ren, J. Jiang, and C. Eckes, "Hierarchical modeling and adaptive clustering for real-time summarization of rush videos in trecvid'08," in *Proc. of the 2nd ACM TRECVid Video Summarization Workshop*, Vancouver, BC, Canada, 2008, pp. 26–30.
24. B. Shahraray and D. C. Gibbon, "Automatic generation of pictorial transcripts of video programs," in *Proc. of Multimedia Computing and Networking*, vol. 2417, Mar. 1995, pp. 512–518.
25. Y.-P. Tan and H. Lu, "Video scene clustering by graph partitioning," *Electronics Letters*, vol. 39, no. 11, pp. 841–842, 2003.
26. B. T. Truong and S. Venkatesh, "Video abstraction: A systematic review and classification," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 3, no. 1, pp. 1–37, 2007.
27. G.C. Van den Eijkel, P.A.P. Porskamp, M. Van Setten, D.D. Velthausz, "Moving Storyboards: a novel approach to content-based video retrieval," Telematica Institute Internal Publication, The Netherlands, 2000.
28. X. N. Xie and F. Wu, "Automatic Video Summarization by Affinity Propagation Clustering and Semantic Content Mining," *Proc. of the 2008 International Symposium on Electronic Commerce and Security*, pp. 203–208, 2008.
29. L. Zhu, E. Zavesky, E. Shahraray, D. Gibbon, and A. Basso, "Brief and high-interest video summary generation: evaluating the AT&T labs rushes summarizations," in *Proc. of the 2nd ACM TRECVid Video Summarization Workshop*, Vancouver, BC, Canada, 2008, pp. 21–25.
30. Y. Zhuang, Y. Rui, T. S. Huan, and S. Mehrotra, "Adaptive key frame extracting using unsupervised clustering," in *Proc. of the International Conference on Image Processing*, Chicago, IL, 1998, pp. 866–870.