VISTO: VIsual STOryboard for Web Video Browsing

Marco Furini
Dipartimento di Informatica
Via Bellini 25/G
Alessandria, Italy
furini@mfn.unipmn.it

Filippo Geraci[*]
IIT - CNR
Via Moruzzi 1
Pisa, Italy
filippo.geraci@iit.cnr.it

Manuela Montangero[†]
Dip. di Ing. dell'Informazione
Via Vignolese 905/b
Modena, Italy
montangero.manuela@unimo.it

Marco Pellegrini
IIT - CNR, Via Moruzzi 1
Pisa, Italy
marco.pellegrini@iit.cnr.it

ABSTRACT

Web video browsing is rapidly becoming a very popular activity in the Web scenario, causing the production of a concise video content representation a real need. Currently, static video summary techniques can be used to this aim. Unfortunately, they require long processing time and hence all the summaries are produced in advance without any users customization. With an increasing number of videos and with the large users heterogeneousness, this is a burden. In this paper we propose VISTO, a summarization technique that produces customized on-the-fly video storyboards. The mechanism uses a fast clustering algorithm that selects the most representative frames using their HSV color distribution and allows users to select the storyboard length and the processing time. An objective and subjective evaluation shows that the storyboards are produced with good quality and in a time that allows on-the-fly usage.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms

Algorithms, Design, Experimentation

Keywords

Video Summary, Video Browsing, Clustering

---

[*]Filippo Geraci works also at the Dipartimento di Ingegneria dell'Informazione, Via Roma 56, Siena, Italy.

[†]Dr. M. Montangero works also at the IIT-CNR, Via Moruzzi 1, Pisa, Italy.

1. INTRODUCTION

Thanks to the advances in networking and multimedia technologies, the presence of digital video contents in the Web is growing at an exceptional speed; videos can be downloaded and played out from almost everywhere using many different devices (e.g., cellphones, palms, laptops) and networking technologies (e.g., EDGE, UMTS, Wi-Fi). The large popularity is highlighted by the enormous success of web sites like Google-Video, YouTube and iTunes Video, where people can upload/download videos. In such a scenario, a tool for performing video browsing would be really appreciated. This tool should provide users with a concise video content representation, so that they would be able to immediately have an idea of the video content, without watching it, so they 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 are receiving increasing attention. In particular, two different approaches are usually followed: one is the production of *static video summary*, which is a collection of still video frames extracted from the original video, and the other is the *dynamic video skimming*, which is a collection of short video clips. In both cases, the idea is to analyze some characteristics of the video stream (e.g., colors, brightness, speech, etc.) in order to find out possible aural/visual clues that would allow a semantics video understanding.

Although static video summary does not preserve the time evolving nature of the video and does not include any aural information, it is the most common technique used to give a concise and informative representation of the original video content (providing that the selected video frames would effective present the video content). In this paper we focus on summarization techniques that produce a collection of static video frames (also known as *video abstract* or *storyboard*).

In literature, different static video summarization techniques have been proposed [9, 18, 19, 20, 12, 8, 11, 10, 16], most of them based on clustering techniques. In this case, the idea is to produce the storyboard by clustering together similar frames and by showing a limited number of frames per cluster (in most cases, only one frame per cluster is selected). With this approach, it is important to select the features upon which frames are considered similar, and dif-

ferent criteria may be employed (e.g., colors distribution, luminance, motion vector, etc.).

Although existing techniques produce acceptable quality storyboards, they usually use complicated clustering algorithms and thus are computationally expensive and very time consuming. For instance, in [16] the computation of the storyboard takes around ten times the video length. This would require video web sites to pre-compute the video storyboard and to present it to the users as-is, without offering them a way of customizing. In fact, it is unreasonable to think of a user waiting for 20 minutes to have a concise representation of a video he/she could have watched in just two minutes. This is a burden, as customization is becoming more and more important in the current Web scenario, where users have different resources/need. For instance, a mobile user has less bandwidth than a DSL-connected user, and he/she might want to receive a storyboard with fewer frames in order to save bandwidth. Conversely, a user who is searching for a specific video scene might want a more detailed storyboard.

The contribution of this paper is to propose VISTO (VIdeo STOryboard), a summarization technique designed to produce customized on-the-fly storyboard. VISTO is based on low-level video frame color features extraction (using the HSV color space distribution) and on a new modification of a simple and fast clustering algorithm to group together similar video frames. The novelty of our result is that the speed up of the computation makes the technique suitable for Web video browsing, allowing users to customize the outcome storyboard according to their needs. In fact, although the mechanism suggests a storyboard length based on the video characteristics, the user can select the length of the storyboard and, thanks to the speed-up of our approach, he/she can re-run the summarization until satisfied with the result.

The evaluation of VISTO is done by investigating both the storyboard production time and the storyboard quality and by comparing the results with other approaches like Open Video [2], k-means[17] and DT Summary [16]. Results show that VISTO needs less than two seconds to produce a storyboard of a 2 minutes video and 15 seconds are necessary to compute the storyboard of a 20 minutes video. The comparison shows that the VISTO clustering time is 25 times faster than k-means and 300 times faster than DT. Furthermore, the storyboard quality investigation (measured through a Mean Opinion Score) shows that the storyboard quality is comparable to the other approaches. Results of the evaluation process candidates VISTO as a tool to provide on-the-fly, customizable and concise video content representation in the Web scenario.

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

2. RELATED WORK

Different approaches have been proposed in literature to address the problem of summarizing a video stream. A first approach relied on two phases: the idea was to first identify all the video shots and then, for each shot, to select a key-frame. Usually, one (the first) [18] or two (the first and the last) [19] key frames were chosen. A drawback of this approach is that, if the shot is dynamic, the first (or the last) frame may not be the most representative one and hence different approaches, like clustering techniques, have been proposed.

In [20] authors propose a clustering algorithm to group video frames using color histogram features. As reported in [16], the approach does not guarantee an optimal result since the number of clusters is pre-defined by a density threshold value. [12] presents a partitioned clustering algorithm where the key-frames that are selected are the ones closest to each cluster centroid. In [16] an automatic clustering algorithm based on Delaunay Triangulation (DT) is proposed; here frames are described through HSV color space distribution. Instead of color space distribution, [11] uses local motion estimation to characterize the video frames and then an algorithm based on the k-medoids clustering algorithm is used.

Although the produced storyboards may achieve a reasonable quality, the clustering computational time is the main burden of these approaches. In fact, the extraction of the video features may produce an enormous matrix (depending on the number of frames that compose the video, i.e. the matrix rows and on the number of features that represents each single frame, i.e., the matrix columns). For this reason, mathematic techniques are used in the attempt to reduce the size of the matrix. For instance, [8] applies the Singular Value Decomposition to the matrix, while [16] uses the Principal Component Analysis. Needless to say, this require additional processing time. Another common approach assumes that frames contain a lot of redundant information and hence, instead of considering all the video frames, only a subset is taken (the so-called *pre-sampling* approach) (e.g., [16]).

Our VISTO proposal does not used any mathematical technique to reduce the video feature matrix, and the decision of using the pre-sampling is left to the user. VISTO only presents the expected storyboard time production for different pre-sampling rates and the user will select the most appropriate one.

3. OUR PROPOSAL

In this paper we propose VISTO, a summarization technique designed to produce on-the-fly, concise and customizable VIdeo STOryboards. VISTO is based on simple and fast clustering algorithm that groups together similar video frames by analyzing their low-level characteristics. In particular, we propose a variation of the Furtherest Point-First algorithm [9, 13], specifically modified for the case of video summary production, while the low-level video frame characteristics are related to the HSV color space distribution.

The characteristics of VISTO are very important in the current Web scenario and will be more and more important in future years. In fact, people with different devices, different networking access and different resources/needs may access to we video servers. In such a scenario, a user may be seeking for a very detailed summary with a lot of frames, while another one may want to save bandwidth and may desire a storyboard with few frames. Hence, a pre-computed storyboard may be un-desired. Although VISTO suggests a possible storyboard length, it allows users to customize the storyboard by selecting the number of video frames that composes the storyboard. Also the storyboard production time can be customized. In fact, since this time depends on

the original video length, VISTO estimates the time necessary to produce the storyboard and gives the user the possibility of requiring a video pre-sampling.

Pre-sampling is a technique largely used to reduce the clustering time (for instance, the mechanism proposed in [16] uses it) and is based on the idea that there are redundancies among the X (e.g, 25) frames per second of the input video. By using a sampling rate, the number of video frames to analyze can be reduced. Needless to say the sampling rate assumes a fundamental importance, as the larger this sampling rate is, the shorter is the clustering time, but the poorer results might be. For this reason, VISTO simply estimates the time necessary for producing the storyboard using different sampling rate and leaves to the user the decision of selecting the desired sampling rate.

As shown in Figure 1, VISTO is composed of three phases: first, the video is analyzed in order to extract the HSV color description; second the clustering algorithm is applied to the extracted data and third, a post-processing phase aims at removing possible redundant or meaningless video frame from the produced summary. In the following we present details of these phases.
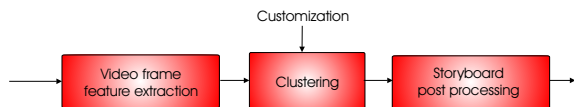

Figure 1: The three-steps ViSto Scheme.

3.1 Video Frame Feature Extraction

The goal of this phase is to describe each video frame and a common way is to do it with a histogram color distribution. This technique is simple to compute and also robust to small changes of the camera position and to camera partial occlusion. Among the possible color spaces, we consider one supported by the MPEG-7 standard, namely the HSV.

HSV defines the color space in terms of three components: Hue (the dominant spectral component, that is 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 [15], in this paper we consider the color histogram as composed of 256 bins. Hence, 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 for clustering purpose.

3.2 Summary by Clustering

The goal of this phase is to group together similar frames and to select a representative frame per each group, so to produce the storyboard sequence. This is done with a clustering algorithm, which works as follows: 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, while distant elements are in different clusters.

In particular, for storyboard production: select a clustering algorithm to cluster the frames (or a subset of the frames, if sampling is done) specifying a distance measure;

give a method to select one key-frame per cluster and place the selected frames in the storyboard. Hence, the selection of the clustering algorithm, of the distance function and of the key-frame selection method is very important as it affects both the quality and the efficiency.

3.2.1 The algorithm

We approach the problem of clustering video frames 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 F and the distance function D is given by the *Generalized Jaccard Distance* (GJD) [3] 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

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

GJD is proven to be a metric [3]. The k-center problem is known to be NP-hard [5], but it can be 2-approximated using the furthest-point-first (FPF) algorithm [9, 13].

VISTO uses the FPF algroithm enhanced with some heuristics to further speed up the computation of clusters. In the rest of this section we will first describe the basic FPF algorithm, and then the successive heuristics that are used to modify the FPF algorithm to finally obtain the clustering algorithm used by VISTO.

Basic Algorithm.

Given the set S of n points, FPF increasingly computes the set of centers $C_1 \subset \cdots \subset C_k \subseteq S$, 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 $1 < i \le k$, 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 $S \setminus C_{i-1}$ to update the mapping μ of points to their closest center.

After k iterations, the set of center $C_k = \{c_1, \ldots, c_k\}$ 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 $i = 1, \ldots, k$. Each iteration can be done in time $O(n)$, hence the overall cost of the algorithm is $O(kn)$. 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.

Heuristics.

Most of the computation is actually spent in computing distances in step two of the algorithm. In [6], authors propose an improved version of FPF, called M-FPF, that exploits the triangular inequality in order to filter out useless distance computations. M-PFP works in any metric space, hence in any vector space.

At iteration $i-1$, the algorithm associates to center c_j the set $\mathcal{C}_j$ of its closest points, stored in a ranked list; *i.e.*, $\mathcal{C}_j$ is an intermediate solution. At iteration i, when a new center c_i is added, the algorithm avoids considering points that surely do not change their closest center: scan every $\mathcal{C}_j$ in decreasing order of distance from c_i, and stop when, for a point $p \in \mathcal{C}_j$, it is the case that

$$D(p, c_j) \leq \frac{1}{2}D(c_j, c_i).$$

By the triangular inequality, any point p that satisfies this condition cannot be closer to c_i than to c_j.
Note that all distances between centers in C_i must be available; this implies an added $O(k^2)$ cost for computing and maintaining these distances, which is anyhow dominated by the term $O(nk)$. The gain is that, in practice, fewer than n points need to be scanned at step two of each iteration.

The efficiency of the algorithm is further improved by applying M-FPF not to the whole set but only to a random sample of size $\sqrt{nk}$ of the input points (sample size suggested in [14]) and adding the other points to the cluster of their closest centers, one by one. Observe that the sample of size $O(\sqrt{nk})$ is taken at random on the entire set of frames without taking into account the consecutiveness of the frames.

In [7], authors observed that centers computed by M-FPF are not good candidate points to guide the completion of clusters, hence they propose a new technique that have been shown to produce clusters of better quality: within each cluster $\mathcal{C}_i$ determine (1) the point a_i furthest from 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, *i.e.*, the point in $\mathcal{C}_i$ that minimizes

$$M(x) = |D(a_i, x) - D(b_i, x)| + |D(a_i, x) + D(b_i, x)|,$$

over all $x \in \mathcal{C}_i$.
Afterward, the remaining points are associated to the closest medoid (instead of center) one by one, according to the Generalized Jaccard Distance, updating a_i, b_i and m_i when necessary.

In this paper we introduce a new heuristic to gain time during this process of adding points to clusters, as the recomputation of medoids is a very time expensive task. First, make an approximated update of medoid and diametral pairs in the following way: if p falls inbetween the diametral pairs and is a better medoids 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 we 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.

Secondly, 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 $D(p, p') \leq 0.2$[1] 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 has been shown not to affect the quality of the result. We clustered the same

datasets by using and not using this heuristic. We have observed that the resulting clusterings show practically no differences, while the time needed to produce the clusterings decreases drastically.

Once the clustering is done, medoids are selected as keyframes and subject to the post processing.

3.2.2 Suggesting the number of clusters

Although customization allows the user to freely choose the number of frames in the storyboard, we can not exclude the case in which the user has no idea of what such a number might be. Hence, we implemented a fast way to make a reasonable estimate of the number of frames that better represents the entire video. This number is always suggested to the user and is used as a default value as the number of clusters that VISTO will compute if there is no other indication by the uses (we remind that exacly one frame per cluster is present in the storyboard). Hence, the number of suggested frames will be denoted by k.

We first take a sample $F' \subseteq F$ of the frames of the entire video, taking one out of ten consecutive frames. We then compute the pairwise distance d_i of consecutive frames f_i', f_{i+1}', according to GJD, for all such pair in F'. Figure 2 shows an example of how these distances are distributed, along time, for the video Drift Ice 6 in [1]. We observe that there are instants of time in which the distance between consecutive frames varies considerably (corresponding to peaks), while there are longer periods in which the d_i's variance is small (corresponding to very dense regions). Usually, peaks correspond to sudden movement in the video or to scene change, while in dense regions frames are more similar one to the other. Hence, frames inbetween two peaks can be considered as a bunch of similar frames and the number of peaks gives the number of such bunches of similar frames. We suggest the number of peaks as k, the default number of clusters (*i.e.*, frames in the storyboard).
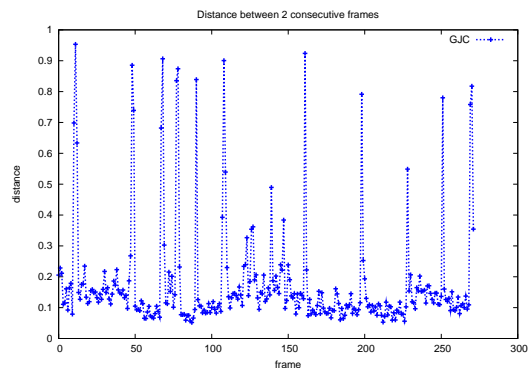

Figure 2: Drift Ice 6: Pairwise distances of sampled frames.

To estimate k we count the number of peaks using the following procedure:

1. Order all the d_i's in increasing order and, for each value v assumed by the d_i's, count how many pairwise distance are equal to v, *i.e.*, let $t(v) = |\{i \mid d_i = v\}|$;

2. Determine the value Γ for which the function $t(v)$ shows a consistent decreasing step and throw away all
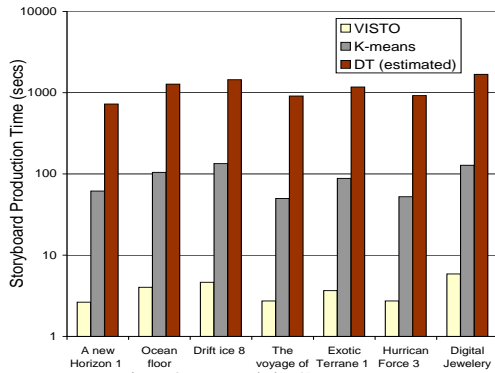
---

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

Figure 3: Storyboard production time: DT vs k-means vs VISTO [Logarithmic scale].

Figure 4: Storyboard production time: A comparison between K-means and Visto with and without sampling (Sx, with $x = 5, 12$, is the sample rate) [Logarithmic Scale].

the frames that are closer than Γ to their successive; *i.e.*, $F'' = \{f'_i \in F' \mid D(f'_i, f'_{i+1}) > \Gamma\}$;

3. Consider the set F'' of remaining frames and count in how many "separated" sets, according to time, they group; *i.e.*, partition F'' into an appropriate number of sets such that if f'_{i_j} and $f'_{i_{j+1}}$ belong to the same set, then $i_{j+1} - i_j < T$, where T is a small interval of time (meaning the two frames are displayed one shortly after the other).

The number of sets into which F'' is partitioned gives the number of peaks. Number k of frames suggested to the user is set to the number of peaks minus one (videos usually begin and end with a peak).

To test if frame sampling influences the estimate of k, we considered our prediction method using all the frames in a video and using only a sample of frames (one out of ten). We tested all the 50 video in [1] and we found out that the estimate of k is exactly the same for 47 videos, while it differs by ± 1 for the remaining three. We conclude that the prediction method is not affected by sampling.

The prediction method (with sampling) applied to the 50 videos in [1] took on the average 0.1 seconds to estimate k (with values spanning from 0.22 to 0.04 seconds).

3.3 Storyboard Post Processing

The goal of this phase is to post process storyboards in order to remove possible meaningless video frames. In fact, the clustering algorithm may select as a key-frame of a cluster a frame completely black (or of another color), due to fade-in fade-out effect or to the use of flashes (very common in sport videos or in news video). By analyzing the HSV of the key-frames selected by the clustering algorithm, it is possible to avoid its presentation to the user. This investigation consumes almost a negligible time, as the number of selected frames is usually very small. After this process, the video storyboard can be presented to the requesting user.

4. VISTO EVALUATION

VISTO is evaluated through a comparison study with other approaches: k-means [17], the Delaunay-based technique (DT) [16] and the one used by the Open Video Project [2].
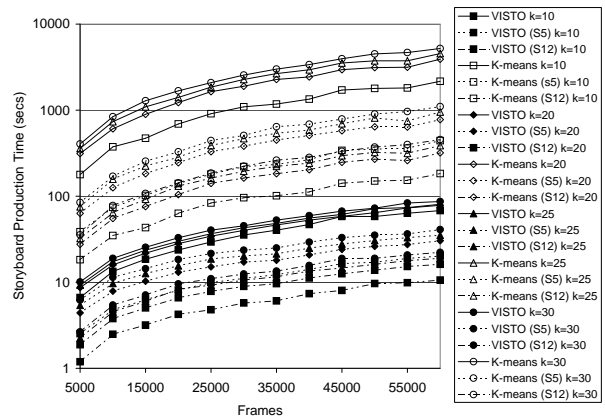
The study is carried out with two different sets of videos: one is taken from [1] and is a subset of short videos available within the Open Video Project [2] (MPEG-1 encoded with a resolution of 352x240); the second set is composed of long entertainment and informative videos (e.g., cartoon, TV-shows and TV-news), MPEG-1 encoded with a resolution of 352x288.

Note that we consider different types of video in order to evaluate our approach under different conditions with respect to color and motion. All the experiments have been done using a Pentium D 3.4 GHz with 3GB RAM, with the aim of investigating two different parameters: the time necessary to produce the storyboard and the quality of the produced summary.

4.1 Storyboard Time

The time necessary to produce a video summary of a given video is an important parameter to decide whether a mechanism can be used to produce on-the-fly summaries or not. Therefore, we investigate time by considering different videos with different lengths.

Figure 3 presents results obtained from analyzing six different videos [1], whose length spans from the 72 seconds (A New Horizon 1) to 168 seconds (Digital Jewelry). Since no statement is given about the time needed to build the storyboards in the Open Video Project [2], as well as nothing is said about the running time of the method on which the project is based [4], here we compare our VISTO approach, with k-means and with DT [16].[2] Note that results are presented on a logarithmic scale, due to the considerable difference among the compared techniques. It can be observed that the usage of k-means and of DT is not reasonable to produce on-the-fly summaries; in fact, k-means needs around 100 seconds to produce a summary of a 120 seconds length video, while DT needs around 1000 seconds. Conversely, VISTO needs less than 10 seconds and hence is well suited to produce on-the-fly summaries. Roughly, in all

---

[2]Results of DT are simply estimated using the value described in [16], where it is reported that the mechanism requires between 9 and 10 times the video length to produce the summary.
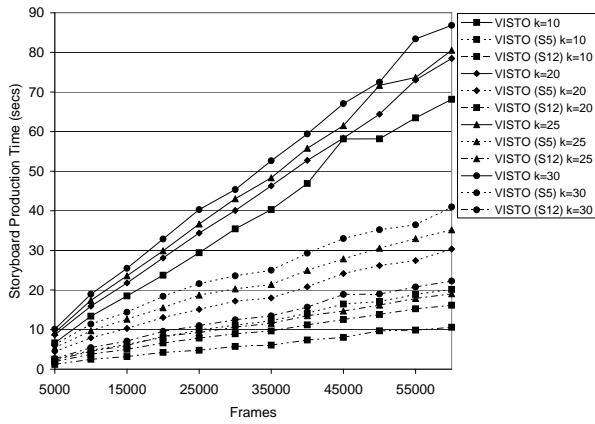
Figure 5: VISTO Storyboard Production Time with and without sampling (Sx, with $x = 5, 12$, is the sample rate).

Figure 6: VISTO: Length and quality of the storyboard can be easily customized.

the tests, VISTO is 25 times faster than k-means and 300 times faster than DT.

A more general investigation on the time necessary to produce a storyboard is presented in Figure 4. We vary the length of the given video from 5000 frames (200 seconds) to 60000 frames (40 minutes), the length of the produced storyboard (10,20,25 and 30 frames) and the rate of the pre-sampling (none, 1 out of 5 and 1 out of 12) that is applied to the video frame feature matrix. We compare k-means and VISTO (the code of the other approaches is not available).

Once again, due to the large difference between the results of the two approaches, the storyboard production time is presented on a logarithmic scale. 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 approaches. Results confirm that k-means requires a production time that causes the method to be unsuitable for on-the-fly video summarization: with no doubts, 178 seconds to summarize a 200 seconds video is too much, not to mention the 36 minutes (2165 seconds) required to summarize a 40 minutes video (60000 frames). Only with a pre-sampling of 1 out of 12, k-means can be used for short videos (18 seconds required for a 200 seconds video), but not for longer video (183 seconds required for a 40 minutes video). To better understand the VISTO behavior, Figure 5 presents a detailed close-up of Figure 4. The VISTO storyboard production time with no sampling is reasonable only for videos whose length is up to 15000 frames (10 minutes). In fact, it is not thinkable to let the user wait for more than 20-25 seconds. For longer videos, a sampling of 1 out of 5 frames produces a waiting time no longer than 20/25 seconds for videos up to 35000 frames (23 minutes). For video larger than 35000 frames, a pre-sampling of 1 out of 12 frame should be considered.

Since the pre-sampling rate might affect the storyboard quality, we let the user select whether to apply a sampling or not. Figure 6 shows the VISTO interface, where a user, in addition to the storyboard length, can also select the quality of the storyboard based on the time he/she is willing to wait. See, `http://visto.iit.cnr.it`.
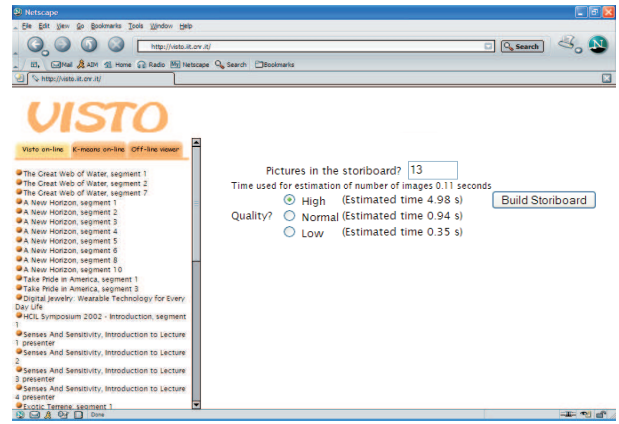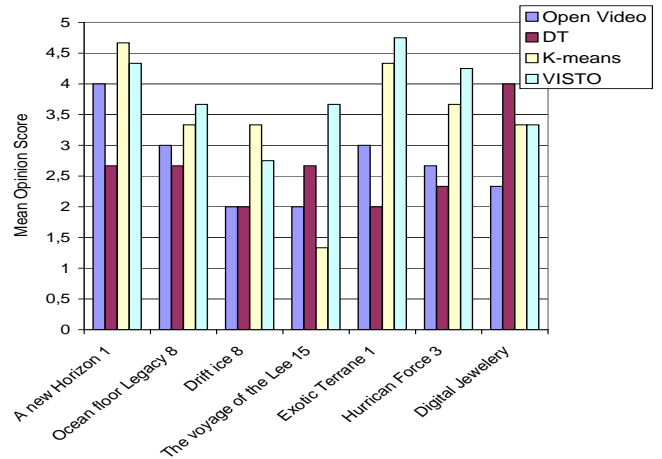

Figure 7: Mean Opinion Score of different storyboard of short videos.

4.2 Storyboard Quality

The time necessary to produce a video storyboard is an important issue, but the quality of the produced storyboard is even more important. In fact, bad quality storyboards (i.e., storyboards that do not well represent the video content) are useless, no matter if they are generated in an instant. For this reason, in the following we investigate the quality of the video summaries produced by VISTO.

The storyboard quality evaluation is carried out by comparing the VISTO results with the one of the Open Video Project, the DT and the k-means, while using the data set of [1]. A further comparison between VISTO and k-means is done using a set of long videos (up to 40 minutes).

Quality evaluation is investigated through a Mean Opinion Score (MOS) test; in particular, we ask a group of 20 people with different background (faculty, Ph.D. students, grad students, researchers) to evaluate the produced summaries. The procedure was the following: we first show them the video and then the summary, asking whether the summary was a good representation of the original video.

The quality of the video summary was scored on a scale 1 to 5 (1=bad, 2=poor, 3=fair, 4=good, 5=excellent) and people were not aware of the mechanism used to produce the

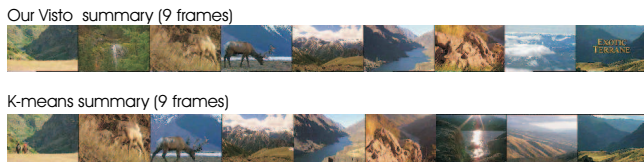**Figure 8:** *A new Horizon*: **storyboard comparison.**

Figure 9: *Exotic Terrane*: **K-means and VISTO comparison.**

Figure 10: MOS evaluation of long videos.

Figure 11: *The Simpsons*: **VISTO vs k-means.**

video summary. The length of the produced summary was set in order to match the other approaches (i.e., if the Open Video summary was of 5 frames, the length of the VISTO summary was set to 5 frames, too).

Figure 7 reports results obtained from evaluating short videos obtained from [1]: *A new Horizon 1* (72 seconds long), *Ocean floor Legacy 8* (127 seconds long), *Drift ice 8* (144 seconds long), *The voyage of the Lee 15* (90 seconds long), *Exotic Terrane 1* (117 seconds long), *Hurricane Force 3* (92 seconds long) and *Digital Jewelery* (168 seconds long). With the exception of *Digital Jewelery* for the DT method and *A new Horizon* for Open Video, these methods achieve poor results. VISTO achieves the best score for *Hurricane Force 3*, *Exotic Terrain 1* and *The voyage of the Lee 15*. With respect to the remaining videos, VISTO and k-means achieve comparable results.

Figure 8 presents the summaries of the *A new Horizon 1* video, where Open Video, K-means and VISTO achieve comparable results. As the MOS reported, it is possible to note that the output of the three storyboards achieve a comparable quality.

Figure 9 presents the summaries generated by VISTO and k-means for the video *Exotic Terrane 1*. The video is a documentary that shows a mountain landscape with some animals. Although some frames are the same in both summaries, VISTO shows a view from the sky and a frame with the video title (last two frames).

Figure 10 reports the MOS results obtained from evaluating long videos: *The Simpsons* (20 minutes long), *TV-News* (30 minutes long), tv-show *Lost* (40 minutes long) and talk-show (15 minutes long). Due to the length of these videos, we produce two different storyboards: one with 15 frames and the other with 30 frames.

Results are comparable for *Lost* and for *TV-News* and different for *The Simpsons* (k-means achieves better results) and for *talk-show* (VISTO achieves better results). These two latter cases are detailed in Figure 11 and in Figure 12,
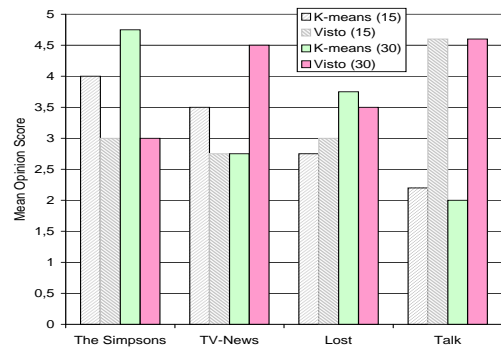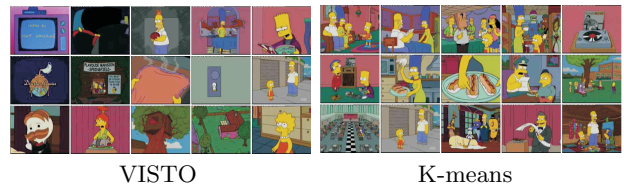
where the difference is quite clear. In particular, it is interesting to observe that the summaries of Figure 11 are completely different, although related to the same video. This can be explained considering the nature of the video taken into consideration: first, just a very small number of frames (15) composes the the storyboard of a video containing a much larger number of frames (30,000); second, in this video, many frames have the same background color and show a yellow character, resulting in high color similarity of frames. Observe that the VISTO summary is composed by frames that show significance color differences. On the other side, the summary of Figure 12 shows how some of the key-frames selected by k-means are very similar one to the other, while VISTO gives a more comprehensive overview of the people participating to the talk show.

4.3 Summary of Results

The evaluation of VISTO showed that the storyboard production time is 25 time faster than k-means and 300 times faster than DT. With respect to the quality, the achieved MOS is comparable. This result is supported by the theoretical observtion that the FPF algorithm is an efficient algorithm that computes a solution to the k-center (a good formalization of the clustering problem) that is a factor two away from the optimum. This close-to-the-optimum solution is further improved by the heuristics. Hence, the great speed up of VISTO does not compromise the storyboard quality, proposing VISTO as a mechanism to perform customized on-the-fly video summaries.

5. CONCLUSIONS

In this paper we presented VISTO, a mechanism designed to produce customized on-the-fly video storyboards. VISTO is provided with a fast clustering algorithm that groups the video frame according to the extracted HSV color space distribution, and allows user to customize the produced output specifying the number of pictures the storyboard has

Figure 12: *Talk-show*: **VISTO vs k-means.**

to have and the time he/she is willing to wait for having the summary. The approach has been evaluated using two different sets of video: one composed of short videos (less than 3 minutes) and one composed of long videos of different categories (cartoon, tv-shows, tv-news, talk-show). The evaluation investigated two fundamental metrics for a summarization scheme: the time necessary to produce the storyboard and its quality. A comparison analysis with other approaches showed that VISTO is much faster: 25 faster than k-means and 300 times faster than DT. A Mean Opinion Score using a group of 20 people has been carried out to investigate the quality of the produced storyboard. Although in most cases VISTO achieved the best score, it can be said that the quality of the storyboard produced by VISTO and k-means is comparable.

Based on the evaluation results, VISTO can be used to offer a customized on-the-fly video storyboard. We are currently working on giving more customization options to the users, allowing him/her to exclude some frames (e.g, exclude all the darkest pictures) or to include just some others (e.g., include only pictures with a lot of blue).

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] http://www.csee.umbc.edu/ yongrao1/delaunay.html.

[2] The open video projec, http://www.open-video.org.

[3] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of STOC-02, 34th Annual ACM Symposium on the Theory of Computing*, pages 380–388, Montreal, CA, 2002.

[4] D. DeMenthon, V. Kobla, and D. Doermann. Video summarization by curve simplification. In *Proc. of Computer Visualization and Pattern Recognition*, 1998.

[5] T. Feder and D. Greene. Optimal algorithms for approximate clustering. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 434–444, New York, NY, USA, 1988. ACM Press.

[6] F. Geraci, M. Pellegrini, P. Pisati, and F. Sebastiani. A scalable algorithm for high-quality clustering of Web snippets. In *Proceedings of SAC-06, 21st ACM Symposium on Applied Computing*, pages 1058–1062, Dijon, FR, 2006.

[7] F. Geraci, M. Pellegrini, F. Sebastiani, and M. Maggini. Cluster generation and cluster labelling for web snippets. In *Proceedings of the 13th Symposium on String Processing and Information Retrieval (SPIRE 2006)*, pages 25–36, Glasgow, UK., October 2006. Volume 4209 in LNCS.

[8] Y. Gong and X. Liu. Video summarization and retrieval using singular value decomposition. *Multimedia Syst.*, 9(2):157–168, 2003.

[9] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(2/3):293–306, 1985.

[10] Y. Hadi, F. Essannoui, R. Thami, and D. Aboutajdine. Video summarization by k-medoid clustering. In *Proc. of ACM Symposium on Applied Computing*, 2006.

[11] Y. Hadi, F. Essannouni, and R. O. H. Thami. Video summarization by k-medoid clustering. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1400–1401, New York, NY, USA, 2006. ACM Press.

[12] 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*, volume 9, pages 1280–1289, 1999.

[13] D. S. Hochbaum and D. B. Shmoys. A best possible approximation algorithm for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[14] P. Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of STOC-99, ACM Symposium on Theory of Computing*, pages 428–434, 1999.

[15] 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*, 11:703–715, 2001.

[16] P. Mundur, Y. Rao, and Y. Yesha. Keyframe-based video summarization using delaunay clustering. *International Journal on Digital Libraries*, 6(2):219–232, 2006.

[17] S. J. Phillips. Acceleration of k-means and related clustering algorithms. In *Proceedings of ALENEX-02, 4th International Workshop on Algorithm Engineering and Experiments*, pages 166–177, San Francisco, US, 2002.

[18] B. Shahraray and D. C. Gibbon. Automatic generation of pictorial transcripts of video programs. In A. A. Rodriguez and J. Maitan, editors, *Proc. SPIE Vol. 2417, p. 512-518, Multimedia Computing and Networking 1995, Arturo A. Rodriguez; Jacek Maitan; Eds.*, pages 512–518, Mar. 1995.

[19] H. Ueda, T. Miyatake, and S. Yoshizawa. Impact: an interactive natural-motion-picture dedicated multimedia authoring system. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 343–350, New York, NY, USA, 1991. ACM Press.

[20] Y. Zhuang, Y. Rui, T. S. Huang, and S. Mehrotra. Adaptive key frame extraction using unsupervised clustering. In *IEEE International Conference on Image Processing*, pages 866–870, 1998.