Shot and scene detection via hierarchical clustering for re-using broadcast video

Lorenzo Baraldi, Costantino Grana, Rita Cucchiara

Dipartimento di Ingegneria "Enzo Ferrari"
Università degli Studi di Modena e Reggio Emilia
Via Vivarelli 10, Modena MO 41125, Italy
name.surname@unimore.it

Abstract. Video decomposition techniques are fundamental tools for allowing effective video browsing and re-using. In this work, we consider the problem of segmenting broadcast videos into coherent scenes, and propose a scene detection algorithm based on hierarchical clustering, along with a very fast state-of-the-art shot segmentation approach. Experiments are performed to demonstrate the effectiveness of our algorithms, by comparing against recent proposals for automatic shot and scene segmentation.

Keywords: shot detection, scene detection, clustering, performance measures.

1 Introduction

In recent years video content has become the major source of Internet traffic, and the large availability of video has led to great interest in fields different from simple entertainment or news broadcasting, such as education. This has also caused a strong interest in the re-use of video content coming from major broadcasting networks, which have been producing high quality edited videos for popular science purposes, such as documentaries and similar programs.

Unfortunately, re-using videos is not an easy task, since it requires video editing skills and tools, on top of the difficulty of finding the parts of videos which effectively contain the specific content one is interested in. Indeed, accessing and browsing a video in an effective way is still a problematic task, especially when the length of the video makes the usage of common seek operations unfeasible to get an insight of the video content.

There is a growing need for managing video content as pieces of text, allowing significant parts to be easily identified, selected, copy and pasted, and so on. The basic unit for this task cannot be the single frame, as a letter cannot be the basic unit for copy and pasting meaningful content from text: higher level groupings are needed, such as DVD chapters. The problem is that most of the on-line reusable content is not provided with editor defined video sub units.

Scene detection has been recognized as a tool which effectively may help in this context, going beyond simple editing units, such as shots. The task is to identify coherent sequences of shots in videos, without any help from the editor or publisher. Of course, a fundamental requirement for scene detection is to accurately identify shot changes. Moreover, evaluating the performance of automatic systems for scene detection is not an easy task: techniques previously employed for different purposes are applied to newer problems, even if they do not perfectly match with the objective at hand, but are easily understood from previous experience. Often this approach leads to erroneous interpretations of the experimental evaluations.

In this paper we present a complete pipeline for scene detection, that includes a shot detection algorithm and a cluster based approach for grouping shots into coherent scenes, and that shows superior results when compared to state-of-the-art methods. We also try to tackle the problem of evaluating scene segmentation results, by proposing an improved definition of the coverage/overflow measures [8], which solves frequently observed cases in which the numeric interpretation would be quite different from the expected results. We publicly release the annotated dataset used for the evaluation as well as the source code of our shot segmentation algorithm.

2 Related works

Video decomposition techniques aim to partition a video into sequences, like shots or scenes. Shots are elementary structural segments that are defined as sequences of images taken without interruption by a single camera. Scenes, on the contrary, are often defined as series of temporally contiguous shots characterized by overlapping links that connect shots with similar content [3].

Most of the existing shot detection techniques relies on the extraction of low level features, like pixel-wise pixel comparisons or color histograms. Other techniques exploit structural features of the frames, such as edges. After the introduction of SVM classifiers, moreover, several approaches exploited them to classify candidate transitions [4]. Recently, algorithms that rely on local descriptors (such as SIFT or SURF) were also proposed. One of the most recent approaches to shot detection, presented in [1], is indeed based on local SURF descriptors and HSV color histograms. Abrupt transitions are detected by thresholding a distance measure between frames, while longer gradual transition are detected by means of the derivative of the moving average of the aforesaid distance. A GPU-based computing framework is also proposed to allow real-time analysis. Their method, when run on a PC with an Intel i7 processor at 3.4 GHz and a NVIDIA GPU, takes one third of the video duration to run.

On a different note, semantically coherent shots which are temporally close to each other can be grouped together to create scenes. Existing works in this field can be roughly categorized into three categories: *rule-based methods*, that consider the way a scene is structured in professional movie production, *graph-based methods*, where shots are arranged in a graph representation, and *clustering-based methods*. They can rely on visual, audio, and textual features.

Rule-based approaches consider the way a scene is structured in professional movie production. Of course, the drawback of this kind of methods is that they tend to fail in videos where film-editing rules are not followed, or when two adjacent scenes are similar and follow the same rules. Liu *et al.* [5], for example, propose a visual based probabilistic framework that imitates the authoring process and detects scenes by incorporating contextual dynamics and learning a scene model. In [2], shots are represented by means of key-frames, thus, the first step of this method is to extract several key-frames from each shot: frames from a shot are clustered using the spectral clustering algorithm, color histograms as features, and the euclidean distance to compute the similarity matrix. The number of clusters is selected by applying a threshold Th on the eigenvalues of the Normalized Laplacian. The distance between a pair of shots is defined as the maximum similarity between key-frames belonging to the two shots, computed using histogram intersection. Shots are clustered using again spectral clustering and the aforesaid distance measure, and then labeled according to the clusters they belong to. Scene boundaries are then detected from the alignment score of the symbolic sequences.

In graph-based methods, instead, shots are arranged in a graph representation and then clustered by partitioning the graph. The Shot Transition Graph (STG), proposed in [9], is one of the most used models in this category: here each node represents a shot and the edges between the shots are weighted by shot similarity. In [6], color and motion features are used to represent shot similarity, and the STG is then split into subgraphs by applying the normalized cuts for graph partitioning. More recently, Sidiropoulos *et al.* [7] introduced a new STG approximation that exploits features automatically extracted from the visual and the auditory channel. This method extends the Shot Transition Graph using multimodal low-level and high-level features. To this aim, multiple STGs are constructed, one for each kind of feature, and then a probabilistic merging process is used to combine their results. The used features include visual features, such as HSV histograms, outputs of visual concept detectors trained using the Bag of Words approach, and audio features, like background conditions classification results, speaker histogram, and model vectors constructed from the responses of a number of audio event detectors.

3 Scene detection as a clustering problem

Since scenes are sets of contiguous shots, the first step in scene detection is to identify shot boundaries. Therefore, we propose a shot segmentation approach that assures high accuracy levels, while keeping execution times low. Our method identifies shot boundaries by computing an extended difference measure, that quantifies the change in the content of two different positions in the video, where positions can be both frames and half-frames. We iteratively compare it against experimentally specified thresholds and parameters that indicate the existence of cuts and gradual transitions.

Algorithm 1: Shot detection

```
$T \leftarrow \{\}$;
/* Abrupt transition detection */
w $\leftarrow$ 0.5;
$C = \{\}$;
forall the $n \leq N$ do
    if $M_w^n > T$ then
        insert $(n, n)$ into $C$
    end
end
Merge consecutive elements of $C$;
$T \leftarrow \{t_i \in C : Peak_w(t_i) > T_P\}$
/* Gradual transition detection */
for $w \leftarrow 1$ to $W$ do
    $C = \{\}$;
    forall the $n \leq N$ do
        if $M_w^n > T$ then
            insert $(n, n)$ into $C$
        end
    end
    Merge consecutive elements of $C$;
    foreach $c \in \{t_i \in C : Peak_w(t_i) > T_P\}$ do
        if distance between c and its nearest element in $T \leq T_S$ then
            insert $c$ into $T$
        end
    end
end
```

3.1 Shot boundaries detection

Given two consecutive shots in a video sequence, the first one ending at frame e, and the second one starting at frame s, we define the transition length as the number of frames in which the transition is visible, that is $L = s - e - 1$. An abrupt transition, therefore, is a transition with length $L = 0$. The transition center is defined as $n = (e + s)/2$ and may correspond to a non-integer value, that is an inter-frame position. This is always true in case of abrupt transitions.

Given a feature $F(i)$ describing frame i, we define the extended difference measure M_n^w, centered on frame or half-frame n, with $2n \in \mathbb{N}$, and with a frame-step $2w \in \mathbb{N}$, as

$$M_w^n = \begin{cases} d\left[F(n-w), F(n+w)\right], & \text{if } n + w \in \mathbb{N} \\ \frac{1}{2}\left(M_w^{n-\frac{1}{2}} + M_w^{n+\frac{1}{2}}\right), & \text{otherwise} \end{cases} \tag{1}$$

where $d(F(i), F(j))$ is the distance between frames i and j, computed in terms of feature F. The second term of the expression is a linear interpolation adopted for inter-frame positions. This is necessary because the feature F is relative to a single frame and cannot be directly computed at half-frames. In our case, distance $d(F(i), F(j))$ is a linear combination of the sum of squared differences of frames i and j and of the χ^2 distance of color histograms extracted from frames i and j. Both measures are normalized by the number of pixels in a frame. The selected features have the property to be almost constant immediately before and after a transition, and to have a constant derivative during a linear transition.

The algorithm starts by simple thresholding the M_w^n values at all frames and half frames positions with $w = 0.5$. This gives us a set of candidate positions for transitions. Now two operations are needed: merging and validation. Merging is simply the aggregation of adjacent candidate positions, providing a list of candidate transitions $C = \{t_i = (f_i, l_i)\}$, where f_i is the first position of the transition, and l_i is the last position. These may be real transitions (most likely hard cuts), or false positives, that is shots with high level differences due to motion. Validation is then performed by measuring the transition *Peak* value, defined as:

$$Peak_w(t) = \max_{f \leq n \leq l}(M_w^n) - \min(M_w^{f-2w}, M_w^{l+2w}) \tag{2}$$

The $Peak_w(t)$ value measures the variation in difference values between the transition and the adjacent shots. In order to validate the transition, therefore, a significant variation must be observed on at least one side of the candidate transition.

To detect gradual transitions, we repeat the previous steps at increasing values of w. Doing so would possibly cause other positions to surpass the threshold value, thus changing and eventually invalidating previously found transitions. For this reason, every validated transition is protected by a “safe zone”. This in practice makes it so that only positions between previous transitions with distance superior to a certain number of frames are further analyzed.

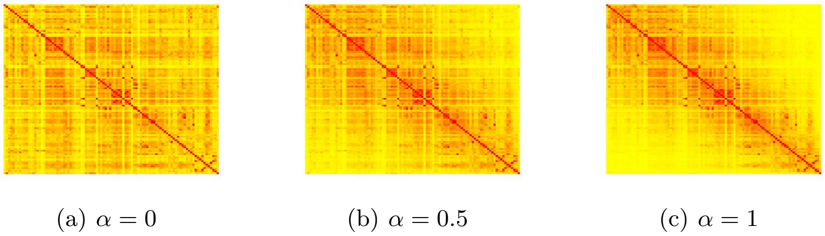
In total we need to setup four parameters for our algorithm: T, the threshold on the differences levels; T_P, a threshold on the *Peak* value, which in practice was usually set to $T/2$; T_S, the number of frames before and after validated transitions, which won't be further analyzed; finally, W, the maximum value for w. A summary of the approach is presented in Algorithm 1.

3.2 Scene detection via hierarchical clustering

Having detected shot boundaries, we now identify scenes by grouping adjacent shots. Shots are described by means of color histograms, hence relying on visual features only: given a video, we compute a three-dimensional histogram of each frame, by quantizing each RGB channel in eight bins, for a total of 512 bins. Then, we sum histograms from frames belonging to the same shot, thus obtaining a single L_1-normalized histogram for each shot.

In contrast to other approaches that used clustering for scene detection, we build a distance measure that jointly describes appearance similarity and temporal proximity. The generic distance between shots $\mathbf{x}_i$ and $\mathbf{x}_j$ is therefore defined as

$$d(\mathbf{x}_i, \mathbf{x}_j) = 1 - \exp\left(-\frac{d_1^2(\psi(\mathbf{x}_i), \psi(\mathbf{x}_j)) + \alpha \cdot d_2^2(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma^2}\right) \tag{3}$$



(a) $\alpha = 0$ (b) $\alpha = 0.5$ (c) $\alpha = 1$

Fig. 1. Effect of α on distance measure $d(\mathbf{x}_i, \mathbf{x}_j)$. Higher values of α enforce connections between near shots and increase the quality of the detected scenes (best viewed in color).

where $\psi(\mathbf{x}_i)$ is the normalized histogram of shot $\mathbf{x}_i$, d_1^2 is the Bhattacharyya distance and $d_2^2(\mathbf{x}_i, \mathbf{x}_j)$ is the normalized temporal distance between shot $\mathbf{x}_i$ and shot $\mathbf{x}_j$, while the parameter α tunes the relative importance of color similarity and temporal distance. To describe temporal distance between frames, $d_2^2(\mathbf{x}_i, \mathbf{x}_j)$ is defined as

$$d_2^2(\mathbf{x}_i, \mathbf{x}_j) = \frac{|m_i - m_j|}{l} \tag{4}$$

where m_i is the index of the central frame of shot $\mathbf{x}_i$, and l is the total number of frames in the video. As shown in Fig. 1, the effect of applying increasing values of α to d is to raise the similarities of adjacent shots, therefore boosting the temporal consistency of the resulting groups.

We then cluster shots using hierarchical clustering methods based on complete linkage, where the dissimilarity between two clusters C_x and C_y is defined as the maximum distance of their elements

$$d(C_x, C_y) = \max_{\mathbf{x}_i \in C_x, \mathbf{x}_j \in C_y} d(\mathbf{x}_i, \mathbf{x}_j) \tag{5}$$

To cluster N shots, we start with N clusters, each containing a single shot, then we iteratively find the least dissimilar pair of clusters, according to Eq. 5, and merge them together, until everything is merged in a single cluster. This process generates a hierarchy of shots, with N levels and i clusters at level i, and each level represents a clustering of the input shots.

Once a particular level is selected, our definition of distance does not guarantee a completely temporal consistent clustering (i.e. some clusters may still contain non-adjacent shots); at the same time, too high values of α would lead to a segmentation that ignores color dissimilarity. The final scene boundaries are created between adjacent shots that do not belong to the same cluster.

4 Experiments

We firstly describe the measures used to evaluate scene segmentation techniques, then, we assess the effectiveness of our shot detection and scene approaches by comparing them against recent methods. We also address two drawbacks of the existing measures.

4.1 Performance measures

We adopt the Coverage, Overflow and F-Score measures, proposed in [8], to evaluate our scene detection results. Coverage $\mathcal{C}$ measures the quantity of shots belonging to the same scene correctly grouped together, while Overflow $\mathcal{O}$ evaluates to what extent shots not belonging to the same scene are erroneously grouped together. Formally, given the set of automatically detected scenes $\mathbf{s} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_m]$, and the ground truth $\tilde{\mathbf{s}} = [\tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2, ..., \tilde{\mathbf{s}}_n]$, where each element of $\mathbf{s}$ and $\tilde{\mathbf{s}}$ is a set of shot indexes, the coverage $\mathcal{C}_t$ of scene $\tilde{\mathbf{s}}_t$ is proportional to the longest overlap between $\mathbf{s}_i$ and $\tilde{\mathbf{s}}_t$:

$$\mathcal{C}_t = \frac{\max_{i=1...,m} \#(\mathbf{s}_i \cap \tilde{\mathbf{s}}_t)}{\#(\tilde{\mathbf{s}}_t)} \tag{6}$$

where $\#(\mathbf{s}_i)$ is the number of shots in scene $\mathbf{s}_i$. The overflow of a scene $\tilde{\mathbf{s}}_t$, $\mathcal{O}_t$, is the amount of overlap of every $\mathbf{s}_i$ corresponding to $\tilde{\mathbf{s}}_t$ with the two surrounding scenes $\tilde{\mathbf{s}}_{t-1}$ and $\tilde{\mathbf{s}}_{t+1}$:

$$\mathcal{O}_t = \frac{\sum_{i=1}^{m} \#(\mathbf{s}_i \setminus \tilde{\mathbf{s}}_t) \cdot \min(1, \#(\mathbf{s}_i \cap \tilde{\mathbf{s}}_t))}{\#(\tilde{\mathbf{s}}_{t-1}) + \#(\tilde{\mathbf{s}}_{t+1})} \tag{7}$$

The computed per-scene measures can then be aggregated into values for an entire video as follows:

$$\mathcal{C} = \sum_{t=1}^{n} \mathcal{C}_t \cdot \frac{\#(\tilde{\mathbf{s}}_t)}{\sum \#(\tilde{\mathbf{s}}_i)}, \quad \mathcal{O} = \sum_{t=1}^{n} \mathcal{O}_t \cdot \frac{\#(\tilde{\mathbf{s}}_t)}{\sum \#(\tilde{\mathbf{s}}_i)} \tag{8}$$

finally, an F-Score metric can be defined to combine Coverage and Overflow in a single measure, by taking the harmonic mean of $\mathcal{C}$ and $1 - \mathcal{O}$.

We identify two inconveniences of these measures, hence we propose an improved definition. The first one is that, being computed at the shot level, an error on a short shot is given the same importance of an error on a very long shot. On the other hand, we propose to normalize $\mathcal{O}_t$ with respect to the length of $\tilde{\mathbf{s}}_t$ instead of that of $\tilde{\mathbf{s}}_{t-1}$ and $\tilde{\mathbf{s}}_{t+1}$, since we believe that the amount of error due to overflowing should be related to the current scene length, instead of its two neighbors. As an example, consider a ground truth segmentation where a long scene is surrounded by two short scenes: if the detected scene is the union of all three, the actual amount of overflow for the middle scene is quite small, while the usage of the original measures would result in a 100% overflow.

Therefore, we propose the Coverage* and Overflow* measures, where the cardinality operator # is replaced with the number of frames of a scene, $l(\mathbf{s}_i)$, and overflow is redefined as follows:

$$\mathcal{O}_t^* = \min\left(1, \frac{\sum_{i=1}^{m} l(\mathbf{s}_i \setminus \tilde{\mathbf{s}}_t) \cdot \min(1, l(\mathbf{s}_i \cap \tilde{\mathbf{s}}_t))}{l(\tilde{\mathbf{s}}_t)}\right) \tag{9}$$

Note that we limit the amount of overflow to one: this also assures that our coverage and overflow belong to $[0, 1]$, a property which was not guaranteed in Eq. 7. The corresponding $\mathcal{C}^*$ and $\mathcal{O}^*$ for an entire video can be obtained in the same way of Eq. 8, using the newly defined cardinality operator.

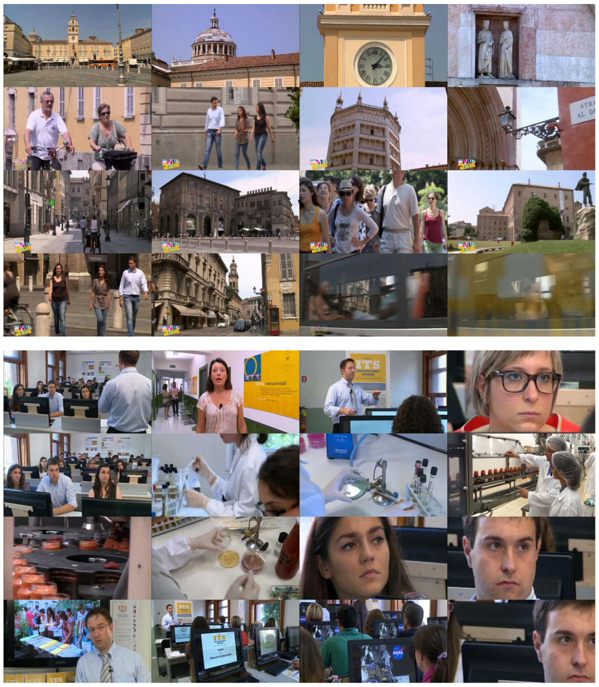


Fig. 2. Two consecutive scenes from the RAI dataset.

4.2 Evaluation

We evaluate our shot and scene detection approach on a collection of ten randomly selected broadcasting videos from the Rai Scuola video archive[1], mainly documentaries and talk shows (see Figure 2). Shots and scenes have been manually annotated by a set of human experts to define the ground truth. Our dataset and the corresponding annotations, together with the code of our shot detection algorithm, are available for download at `http://imagelab.ing.unimore.it`

For the shot detection task, our dataset contains 987 shot boundaries, 724 of them being hard cuts and 263 gradual transitions. The percentage of gradual transitions greatly varies from video to video, with V_4, V_5, V_9 and V_{10} having a percentage of gradual transitions superior to 44%, and the rest having a mean of 9%.

The performance of the proposed approach was evaluated and compared against the recent proposal of Apostolidis *et al.* [1], using the executable provided by the authors. Threshold T was set to 80, while the safe zone T_S was fixed to 20 frames, and we repeated our gradual transitions search routine up to $w = 2.5$. As it can be seen from the experimental results summarized in Table 1, our approach performs considerably well, achieving high levels of F-measure on all videos, except in videos with lots of gradual transitions. Indeed, it shows very

[1] `http://www.scuola.rai.it`

Table 1. Characteristics of the RAI Dataset and shot detection performance in terms of F-measure.

Video	% of gradual	Avg. trans. duration
V_1	0.21	3.82
V_2	0.07	1.58
V_3	0.09	1.41
V_4	0.45	14.10
V_5	0.89	19.48
V_6	0.06	0.74
V_7	0.04	0.49
V_8	0.09	1.65
V_9	0.58	12.64
V_{10}	0.44	10.53

Video	Method in [1]	Our method
V_1	0.92	**0.97**
V_2	0.92	**0.97**
V_3	0.94	**0.95**
V_4	**0.82**	0.78
V_5	**0.55**	0.38
V_6	0.93	**0.96**
V_7	0.89	**0.94**
V_8	0.94	0.94
V_9	0.75	**0.76**
V_{10}	0.77	0.77
Average	0.84	0.84

Table 2. Performance comparison on the RAI dataset using the Coverage, Overflow and F-Score measures.

Video	Chasanis *et al.* [2]			Sidiropoulos *et al.* [7]			Our method		
	F-Score	$\mathcal{C}$	$\mathcal{O}$	F-Score	$\mathcal{C}$	$\mathcal{O}$	F-Score	$\mathcal{C}$	$\mathcal{O}$
V_1	0.70	0.64	0.24	0.72	0.84	0.37	**0.73**	0.61	0.11
V_2	0.36	0.80	0.77	0.59	0.85	0.55	**0.73**	0.59	0.06
V_3	0.58	0.73	0.52	0.58	0.90	0.57	**0.60**	0.79	0.51
V_4	0.50	0.65	0.60	0.33	0.94	0.80	**0.77**	0.78	0.23
V_5	0.25	0.93	0.86	0.66	0.76	0.41	**0.76**	0.72	0.19
V_6	0.18	0.89	0.90	0.71	0.77	0.34	**0.73**	0.74	0.28
V_7	0.37	0.70	0.75	0.51	0.78	0.62	**0.70**	0.72	0.31
V_8	**0.62**	0.57	0.32	0.45	0.88	0.70	0.60	0.65	0.44
V_9	0.27	0.87	0.84	0.43	0.92	0.72	**0.83**	0.85	0.19
V_{10}	0.54	0.91	0.62	0.44	0.94	0.71	**0.61**	0.49	0.20
Average	0.44	0.77	0.64	0.54	0.86	0.58	**0.70**	0.69	0.25

good performances on abrupt and short gradual transitions, while it tends to fail on very long transitions. Overall, our method achieves competitive results when compared to [1]. Regarding time performance, the running time of a CPU-based single-thread implementation of our algorithm is about 13% of the video duration on a PC with Intel i7 processor at 3.6 GHz, which is more than twice faster than [1].

To evaluate our scene detection approach, instead, we compare our method against the multimodal approach presented in [7] and that of [2]. We use the executable of [7] provided by the authors[2] and reimplement the method in [2]. Parameters of [2] were selected to maximize the performance on our dataset.

The overall results on the dataset are shown in Table 2, using Vendrig's measures (Coverage, Overflow and F-Score), and in Table 3, using our improved definitions (Score*, Overflow* and F-Score*). Our method achieves competi-

[2] `http://mklab.iti.gr/project/video-shot-segm`

Table 3. Performance comparison on the RAI dataset using the Coverage*, Overflow* and F-Score* measures.

Video	Chasanis *et al.* [2]			Sidiropoulos *et al.* [7]			Our method		
	F-Score*	$\mathcal{C}^*$	$\mathcal{O}^*$	F-Score*	$\mathcal{C}^*$	$\mathcal{O}^*$	F-Score*	$\mathcal{C}^*$	$\mathcal{O}^*$
V_1	0.70	0.65	0.24	0.70	0.63	0.20	**0.82**	0.75	0.10
V_2	0.60	0.91	0.55	0.61	0.73	0.47	**0.67**	0.55	0.15
V_3	0.51	0.87	0.64	0.51	0.89	0.64	**0.60**	0.84	0.54
V_4	0.54	0.70	0.56	0.22	0.95	0.88	**0.73**	0.79	0.33
V_5	0.34	0.92	0.79	0.57	0.66	0.50	**0.79**	0.73	0.14
V_6	0.20	0.89	0.88	**0.74**	0.72	0.24	0.68	0.67	0.31
V_7	0.37	0.75	0.76	0.56	0.69	0.53	**0.80**	0.78	0.17
V_8	0.59	0.65	0.47	0.15	0.89	0.92	**0.62**	0.66	0.42
V_9	0.07	0.83	0.96	0.15	0.94	0.92	**0.85**	0.91	0.20
V_{10}	0.50	0.93	0.66	0.11	0.93	0.94	**0.67**	0.57	0.20
Average	0.44	0.81	0.65	0.43	0.80	0.63	**0.72**	0.73	0.26

tive results, using both measures, when compared to recent and state-of-the-art methods like [7], and features a considerably reduced overflow. When shot duration is taken into account, using our measures, the improvement of our method over the others is even clearer.

5 Conclusions

We described a novel approach to video re-use by means of shot and scene detection, which is motivated by the need of accessing and re-using the existing footage in more effective ways. We presented a shot detection approach that relies on an extended distance measure and that is capable of detecting abrupt and gradual transitions, with very low execution times, and a scene detection model that jointly considers temporal proximity and color similarity. Our scene detection results outperform the state-of-the-art algorithms by a large margin on the RAI dataset.

Acknowledgments This work was carried out within the project "Città educante" (CTN01_00034_393801) of the National Technological Cluster on Smart Communities cofunded by the Italian Ministry of Education, University and Research - MIUR.

References

1. Apostolidis, E., Mezaris, V.: Fast Shot Segmentation Combining Global and Local Visual Descriptors. In: IEEE Int. Conf. Acoustics, Speech and Signal Process. pp. 6583–6587 (2014)
2. Chasanis, V.T., Likas, C., Galatsanos, N.P.: Scene detection in videos using shot clustering and sequence alignment. IEEE Trans. Multimedia 11(1), 89–100 (2009)
3. Hanjalic, A., Lagendijk, R.L., Biemond, J.: Automated high-level movie segmentation for advanced video-retrieval systems. IEEE Trans. Circuits Syst. Video Technol. 9(4), 580–588 (1999)

4. Ling, X., Yuanxin, O., Huan, L., Zhang, X.: A method for fast shot boundary detection based on SVM. In: Image and Signal Processing, 2008. CISP'08. Congress on. vol. 2, pp. 445–449 (2008)
5. Liu, C., Wang, D., Zhu, J., Zhang, B.: Learning a Contextual Multi-Thread Model for Movie/TV Scene Segmentation. IEEE Trans. Multimedia 15(4), 884–897 (2013)
6. Rasheed, Z., Shah, M.: Detection and representation of scenes in videos. IEEE Trans. Multimedia 7(6), 1097–1105 (2005)
7. Sidiropoulos, P., Mezaris, V., Kompatsiaris, I., Meinedo, H., Bugalho, M., Trancoso, I.: Temporal video segmentation to scenes using high-level audiovisual features. IEEE Trans. Circuits Syst. Video Technol. 21(8), 1163–1177 (2011)
8. Vendrig, J., Worring, M.: Systematic evaluation of logical story unit segmentation. IEEE Trans. Multimedia 4(4), 492–499 (2002)
9. Yeung, M.M., Yeo, B.L., Wolf, W.H., Liu, B.: Video browsing using clustering and scene transitions on compressed sequences. In: IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology. pp. 399–413 (1995)