Shot, Scene and Keyframe Ordering for Interactive Video Re-use

Lorenzo Baraldi, Costantino Grana, Guido Borghi, Roberto Vezzani and Rita Cucchiara
Dipartimento di Ingegneria “Enzo Ferrari”, Università degli Studi di Modena e Reggio Emilia, Via Vivarelli 10, Modena MO 41125, Italy

Keywords: Shot Detection, Scene Detection, Performance Measures, Clustering.

Abstract: This paper presents a complete system for shot and scene detection in broadcast videos, as well as a method to select the best representative key-frames, which could be used in new interactive interfaces for accessing large collections of edited videos. The final goal is to enable an improved access to video footage and the re-use of video content with the direct management of user-selected video-clips.

1 INTRODUCTION

Videos are nowadays the largest and richest source of information in the multimedia universe; as a matter of fact, some estimations suggest that more than 70% of big data in the world consists of videos. One of the main issues, for content producers and owners, like broadcasting networks, is to re-use and make video content accessible in an enjoyable, efficient and interactive way.

We argue that there is a growing need of automatic tools to de-structure video content in useful and semantically consistent clips, especially when the length of the video makes the usage of common seek operations unfeasible to get an insight of the video content. The final goal is that of managing videos as pieces of text, allowing significant parts to be easily identified, selected, copy-and-pasted, and thus re-used. The basic unit for this purpose cannot be the single frame, as a character cannot be the basic unit for a text copy-and-pasting task. It also cannot be something like a DVD chapter, since either it is too long, or it is defined by the editor with a specific interpretation, which not necessarily matches the re-use needs.

In this work, we present a pipeline of automatic video analysis which includes shot detection, scene detection, and keyframes selection with importance ordering. The output is an XML based description, that allows a web interface to provide the user with an easier navigation system, and a way for selecting and extracting meaningful video parts. We publicly release the source code of our shot segmentation algorithm.

2 RELATED WORK

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 (Hanjalic et al., 1999).

Most of the existing shot detection techniques relies on the extraction of low level features, like pixel-wise comparisons or color histograms. Other techniques exploit structural frame features, such as edges. After the introduction of SVM classifiers, moreover, several approaches exploited them to classify candidate transitions (Ling et al., 2008). 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 (Apostolidis and Mezaris, 2014), is indeed based on local SURF descriptors and HSV color histograms.

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, graph-based methods, 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. Liu et al. (Liu et al., 2013), for example, propose a visual based probabilistic framework that imitates the authoring process and detects scenes by incorporat-
ing contextual dynamics and learning a scene model. In (Chasanis et al., 2009), shots are represented by means of key-frames, clustered with spectral clustering, 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 (Yeung et al., 1995), 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 (Rasheed and Shah, 2005), 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. (Sidiropoulos et al., 2011) introduced a new STG approximation that exploits features extracted from the visual and the auditory channel.

3 VIDEO ANALYSIS

Videos can be decomposed at three different granularity levels: frames, shots and scenes. A video, indeed, is an ordered set of frames; sequences of adjacent frames taken by a single camera compose a shot, and two consecutive shots can be spaced out by a transition, which is in turn a set of frames. Finally, sets of contiguous and semantically coherent shots form a scene.

Since scenes are sets of shots, the first step in scene detection is the identification of shot boundaries. We propose a shot segmentation approach that assures high accuracy levels, while keeping execution times low. Our method identifies shot boundaries by means of an extended difference measure, that quantifies the change in the content of two different positions in the video. Shots are then grouped into scenes with a clustering approach that includes temporal cues. We also describe a solution to sort key-frames from a scene to let the user select the level of detail of a scene summary. Finally, every shot is enriched by a number of tags, automatically detected on the selected keyframes, using the API provided by Clarifai, Inc.1.

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, 

\[ L = s - e - 1. \]

An abrupt transition, therefore, is a transition with length \( L = 0 \). The transition center, \( n = (e+s)/2 \), may correspond to a non-integer value, that is an inter-frame position: this is always true in case of abrupt transitions.

Having selected a feature \( F \) to describe frames in a video, we define the extended difference measure \( M_w^f \), centered on frame or half-frame \( n \), with \( 2n \in \mathbb{N} \), and with a frame-step \( 2w \in \mathbb{N} \), as

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

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 feature \( F \) is relative to a single frame and cannot be directly computed at half-frames. The selected features should 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 thresholding the \( M_w^f \) values at all frames and half frames positions with \( w = 0.5 \). This gives a set of candidate positions for transitions. Two operations are then needed: merging and validation. Merging is the aggregation of adjacent candidate positions, which provides a list of candidate transitions \( C = \{ l_i = (f_i, l_i) \} \), 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, i.e. shots with high level differences due to motion. A validation step is then performed to prune false positives, by measuring the transition Peak value, which is defined as:

\[
\text{Peak}_{w}(t) = \max_{f \leq n \leq l} (M_w^f) - \min_{f \leq n \leq l} (M_w^{f-2w}, M_w^{f+2w})
\]

\( \text{Peak}_{w}(t) \) 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, previous steps are repeated at increasing values of \( w \). This 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: only positions between previous transitions with distance superior to a certain number of frames are further analyzed.

In total four parameters need to be set up for our algorithm: \( T \), the threshold on difference levels; \( T_P \),

\[1\text{https://developer.clarifai.com/docs/} \]
Algorithm 1: Shot detection.

\[ T \leftarrow \{\}; // \text{set of transitions} \]

\[ \text{for } w \leftarrow 0.5 \text{ to } W \text{ do} \]

\[ C \leftarrow \{\}; // \text{set of candidates} \]

\[ \text{for } n \leftarrow 0 \text{ to } N \text{ do} \]

\[ \text{if } M^i_w > T \text{ then} \]

\[ C \leftarrow C \cup (n,n); // \text{add a candidate abrupt transition} \]

\[ \text{end} \]

\[ \text{end} \]

\[ \text{Merge consecutive elements of } C; \]

\[ // \text{make candidates gradual} \]

\[ \text{foreach } c \in \{t_i \in C : \text{Peak}_w(t_i) > T \} \text{ do} \]

\[ \text{if distance between } c \text{ and its nearest element in } T \leq T_5 \text{ then} \]

\[ T \leftarrow T \cup c; // \text{make candidate a confirmed transition} \]

\[ \text{end} \]

\[ \text{end} \]

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

Figure 1: Effect of \( \alpha \) on distance measure \( d(i,j) \). Higher values of \( \alpha \) enforce connections between near shots and increase the quality of the detected scenes (best viewed in color).

A threshold on the Peak value, which in practice was set to \( T / 2; T_5 \), the number of frames composing the safe zone; 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, scenes can be identified by grouping adjacent shots. 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 \( i \) and \( j \) is defined as

\[
d(i,j) = 1 - \exp \left( -\frac{d^1(i,j) \cdot \psi(i,j) + \alpha \cdot d^2(i,j)}{2\sigma^2} \right)
\]

where \( \psi(i,j) \) is the visual feature vector describing shot \( i \), \( d^1(i,j) \) is the Bhattacharyya distance and \( d^2(i,j) \) is the normalized temporal distance between shot \( i \) and shot \( j \), while the parameter \( \alpha \) tunes the relative importance of visual similarity and temporal distance. To describe temporal distance between frames, \( d^2(i,j) \) is defined as:

\[
d^2(i,j) = \frac{|m_i - m_j|}{I}
\]

where \( m_i \) is the index of the central frame of shot \( i \), and \( I \) is the total number of frames in the video. As shown in Fig. 1, the effect of applying increasing values of \( \alpha \) 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 \( C_i \) and \( C_j \) is defined as the maximum distance of their elements

\[
d(C_i, C_j) = \max_{i \in C_i, j \in C_j} d(i,j)
\]

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 \( i \), each level representing a clustering of the input shots.

Once a particular level is selected, the aforementioned 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 \( \alpha \) would lead to a segmentation that ignores visual dissimilarity. The final scene boundaries are created between adjacent shots that do not belong to the same cluster.

3.3 Importance Ordering of Keyframes

Presenting video content in reduced form is not straightforward: a random selection of frames, for example, may show the same subject many times, while disregarding the visual variability of a scene. The number of keyframes per scene, moreover, should be a parameter controlled by the user, based on the available space on the interface: for this reason we devise a solution to sort the keyframes in order of "presentation importance", which allows to show the more significant ones before the others.

In the extreme case of summarizing a scene with just one shot, the selected shot should convey something which is shared by all keyframes, thus a reasonable choice is to pick the median of the set of keyframes. If we then want to add another frame to the summary, this should be as different as possible from the already selected frames, so the keyframe with the maximum distance from the nearest element in the summary could be selected.
To sum up, every scene is described by a set of keyframes obtained by selecting the center frame of every shot in the scene. Distance between frames is measured as for the scene detection, and all keyframes are ordered based on their “presentation importance”. Fig. 2 shows an example of a scene shots reordering: shots are included in order of importance, if the user requests to reduce their number.

3.4 XML Description

The output of the video analysis module has to be provided to the user interface, which allows the user to effectively interact with the video resource. In order to provide a formal and extendible machine readable format we adopt an XML description for the video structure.

A <video> element is the root node and begins with the basic metadata: <url>, <title>, and <description>. Then a <shots> element starts and it lists all detected shots in the video resource. Every <shot> has an id attribute (its time position in seconds), the starting and closing frames (begin and end) and contains a whitespace separated list with the words detected by the Clarifai API (spaces in concepts are substituted by underscores). Finally a <scenes> element describes the detected scenes, as a sequence of <scene> elements which refer to the previously defined shots again with a begin and end attribute. Every scene contains the full list of shots in presentation order, so that the user interface knows the time position of the scene and the shots to show in case a summarization is required.

Fig. 3 shows an extract of a longer XML description, referring to the shots in Fig. 2. It is possible to note that ten shots are contained in the scene (from 124 to 147), and that the first one in presentation order should be the shot with id=127, that is the second one. The shot is described by the words street, architecture, town, and house, which are indeed a very good guess.

4 EXPERIMENTS

We evaluate our shot and scene detection approach on a collection of ten randomly selected broadcasting videos from the Rai Scuola video archive,

\[\text{http://www.scuola.rai.it}\]

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%. In order to evaluate the shot detection results we employ the classical definitions of Precision and Recall of a transition, summarizing them with the F-measure. To evaluate the scene detection results, instead, we adopt the improved version the Coverage, Overflow and F-Score measures proposed in (Baraldi

\[\text{http://www.scuola.rai.it}\]
et al., 2015). Coverage* measures the percentage of frames belonging to the same scene correctly grouped together, while Overflow* evaluates to what extent frames not belonging to the same scene are erroneously grouped together.

Distance $d(F(i), F(j))$ was set to a linear combination of the sum of squared differences of frames $i$ and $j$ and of the $\chi^2$ distance of color histograms extracted from frames $i$ and $j$. Both measures were normalized by the number of pixels in a frame. For the scene detection task, shots were described by means of color histograms, hence relying on visual features only: given a video, a three-dimensional histogram was computed for each frame, by quantizing each RGB channel in eight bins, for a total of 512 bins. Then, histograms from frames belonging to the same shot were summed together, thus obtaining a single $L_1$-normalized histogram for each shot.

The performance of the proposed approaches was evaluated and compared against the shot detection proposal of Apostolidis et al. (Apostolidis and Mezaris, 2014), and the scene detection approaches presented in (Sidiropoulos et al., 2011) and (Chasanis et al., 2009) using the executable provided by the authors3 for the first two and reimplementing the method in (Chasanis et al., 2009). 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 $w = 2.5$.

Our shot detection approach performs considerably well, achieving high levels of F-measure on all videos, except in those with lots of gradual transitions. Overall, our method achieves an F-measure of 0.84, exactly the same results obtained by the algorithm of (Apostolidis and Mezaris, 2014). In general, it shows very good performance on abrupt and short gradual transitions, while it tends to fail on very long transitions. 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 Core i7 processor at 3.6 GHz, which is more than twice faster than (Apostolidis and Mezaris, 2014).

Regarding scene detection, the overall results of our approach are shown in Table 1. As it can be seen, our method achieves excellent results, when compared to recent and state-of-the-art methods, featuring a considerably reduced

---

3http://mklab.iti.gr/project/video-shot-segm
Table 1: Performance comparison on the RAI dataset using the Coverage*, Overflow* and F-Score* measures.

<table>
<thead>
<tr>
<th>Video</th>
<th>Chasanis et al.</th>
<th>Sidiropoulos et al.</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F-Score*</td>
<td>C*</td>
<td>O*</td>
</tr>
<tr>
<td>V1</td>
<td>0.70</td>
<td>0.65</td>
<td>0.24</td>
</tr>
<tr>
<td>V2</td>
<td>0.60</td>
<td>0.91</td>
<td>0.55</td>
</tr>
<tr>
<td>V3</td>
<td>0.51</td>
<td>0.87</td>
<td>0.64</td>
</tr>
<tr>
<td>V4</td>
<td>0.54</td>
<td>0.70</td>
<td>0.56</td>
</tr>
<tr>
<td>V5</td>
<td>0.34</td>
<td>0.92</td>
<td>0.79</td>
</tr>
<tr>
<td>V6</td>
<td>0.20</td>
<td>0.89</td>
<td>0.88</td>
</tr>
<tr>
<td>V7</td>
<td>0.37</td>
<td>0.75</td>
<td>0.76</td>
</tr>
<tr>
<td>V8</td>
<td>0.59</td>
<td>0.65</td>
<td>0.47</td>
</tr>
<tr>
<td>V9</td>
<td>0.07</td>
<td>0.83</td>
<td>0.96</td>
</tr>
<tr>
<td>V10</td>
<td>0.50</td>
<td>0.93</td>
<td>0.66</td>
</tr>
<tr>
<td>Average</td>
<td>0.44</td>
<td>0.81</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Overflow*. Finally, a prototype user interface has been created and is available for testing at http://imagelab.ing.unimore.it/scenedemo/. A sample screenshot is shown in Fig. 4.

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. Additional improvements and a simple XML description allow the creation of an effective user interface which enables the user to interact with the video.

ACKNOWLEDGEMENTS

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

REFERENCES


