ANALYSIS OF VIDEO SEQUENCES: TABLE OF CONTENTS AND INDEX CREATION

Joan Llach, Philippe Salembier

Universitat Politècnica de Catalunya, Campus Nord - Mòdul D5, C/ Gran Capità s/n, 08034 Barcelona, Spain

E-mail: {jllach,philippe}@gps.tsc.upc.es — WWW: http://gps-tsc.upc.es/imatge

ABSTRACT

This paper deals with the representation of video sequences useful for tasks such as long-term analysis, indexing or browsing. A *Table Of Content* and *index* creation algorithm is presented, as well as additional tools involved in their creation. The proposed method does not assume any *a priori* knowledge about the content or the structure of the video. It is therefore a generic technique. Some examples are presented in order to assess the performance of the algorithm.

1. INTRODUCTION

In the framework of video analysis for indexing application (for example application related to MPEG-7), the representation of video sequences is an important issue. It is not enough to describe the content of a video but also to develop techniques which are able to automatically create these descriptions. In this paper, a new technique that automatically creates *TOCs* and *indexes* is presented. The proposed algorithm only relies on visual information, unlike other techniques which use both video and audio information [3].

The goal of the *TOC* is to define the structure of the video sequence in a hierarchical fashion. The original sequence should be subdivided in sub-sequences which can also be divided in shorter sub-sequences, etc. At the end of this division process, the shortest entity to be described is the *micro-segment*. The *index* is also a hierarchical structure but it does not describe the structure of the sequence but the occurrences of similar content.

The creation of the *TOC* and *index* follows the same strategy. The first step splits the sequence into shots. The second step divides each shot into *micro-segments* which are sub-components of a shot where the camera activity is homogeneous. These *micro-segments* constitute the lowest level of the *TOC* or *index* representations. The third step creates the hierarchical structures by clustering the detected shots.

This paper is structured as follows. Section 2 deals with the shot detection algorithm. The temporal segmentation of shots is detailed in section 3, whereas their clustering is explained in section 4. Some conclusions are driven in section 5.

2. SHOT DETECTION

The first step in the *TOC* and *index* creation splits the sequence into shots. Taking into account that a shot is a set of contiguous frames without editing effects, the algorithm has to detect the transitions between consecutive shots. These transitions can be abrupt or more sophisticated, like dissolves or fades. The shot detection algorithm consists of two main steps: computation of the *mean Displaced Frame Difference* $mDFD$ curve and its segmentation into shots.

This work is supported by Philips Research.

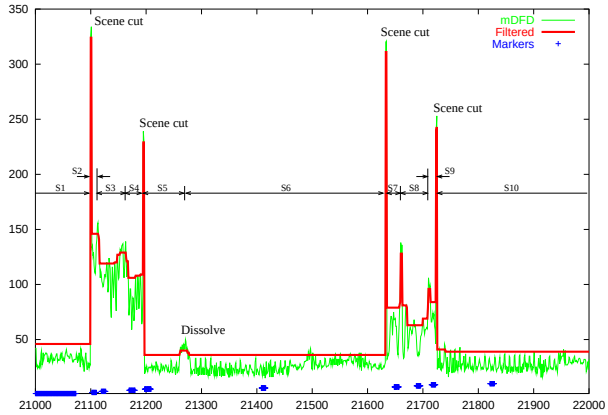

Figure 2: $mDFD$ for the *news2* sequence (frame 21 000 to 22 000, thin line). The filtered curve (thick line) and the detected markers (horizontal segments) are also plotted. S1–S10 depict the limits of the detected shots.

2.1. Computation of the $mDFD$ curve

The $mDFD$ curve is obtained using a variation of the *Displaced Frame Difference DFD* to take into account luminance and chrominance information. If I_x and I_y are the image dimensions and w's are the weights for Y, U, V components, the *mDFD* is computed as follows: $mDFD(t) = \frac{1}{I_x I_y} \sum_k^{Y,U,V} w_k \sum_{i,j}^{I_x,I_y} |DFD_k(i,j;t-1,t+1)|$.

In the example shown in Fig. 2 the weights have been set to $\{w_Y, w_U, w_V\} = \{1,3,3\}$, which are typical values. The highest peaks of the curve correspond to the abrupt transitions (frames 21 100, 21 195, 21 633 and 21 724). On the other side, the oscillation from frame 21 260 to 21 279 corresponds to a dissolve. The presence of large moving foreground objects in frames 21 100–21 195 and 21 633–21 724 creates high level oscillations of the $mDFD$ curve.

2.2. Segmentation of the $mDFD$ curve

The second block of the shot detection algorithm detects the video editing effects. Our approach has been to adapt classical spatial segmentation strategies to the case of temporal segmentation. Most of classical shot detection techniques use threshold-based segmentation to extract the highest peaks of the *mDFD* or another type of mono-dimensional curve. Although a large number of shots can actually be detected by this approach, this class of algorithms is not robust and quite sensitive to noise. It is in particular difficult to detect peaks of small contrast corresponding to fading or special effects. The solution that we have used is a homogeneity-based approach relying on morphological tools. The *mDFD* curve is processed following four steps:

Figure 1: Detected shots for *drama* sequence, using $l_{min} = 20$ and $c = 10$. Each image represents the central frame of the shot, the numbers show its starting and ending frame. Shots are ordered from left to right and from top to bottom.

1. *Simplification by temporal filtering.* Morphological closing with a 1D structuring element of length l_{min}. With this operation, negative peaks the length of which is less than l_{min} frames are removed. The l_{min} parameter defines the duration of the shortest shot to be detected.
2. *Simplification by positive contrast filter.* Peaks that have a positive contrast lower than c parameter are removed.
3. *Marker extraction.* Each marker corresponds to the kernel of one shot. So, each marker must cover a portion of the curve with a high probability to belong to a single shot. Because contiguous frames that belong to the same shot are quite similar, the value of the *mDFD* will be small around those frames. Thus, to extract the markers a negative contrast filter is used because it detects the relative minimums of the curve.
4. *Watershed.* Its purpose is to propagate the markers on the *mDFD* curve until all its points are assigned to a marker. The propagation process is performed by applying a watershed algorithm on the curve using as initial markers those obtained in the previous step.

Figure 2 shows the filtered curve (both temporal filter and positive and negative contrast filter), the resulting markers and detected shots using $l_{min} = 10$ and $c = 10$. Even though some over-segmentation appears around frames 21 150 and 21 700, both the scene cuts and the dissolve have been correctly detected. The over-segmentation is not a problem because the next steps of the algorithm eliminate this effect.

3. TEMPORAL SEGMENTATION OF VIDEO SHOTS USING CAMERA MOTION PARAMETERS

The goal of the temporal segmentation algorithm is to split each shot into several micro-segments which present a high level of homogeneity on the camera motion parameters. The data used to perform the segmentation are the camera motion parameters currently used in MPEG-7 [1]. The algorithm is applied to each shot and consists of two steps: 1) each shot is over-segmented into several *micro-segments* which must present a perfect homogeneity (see section 3.1), and 2) a merging process is applied while the homogeneity level of the set of *micro-segments* is below a predefined threshold.

In order to segment the shot, it is necessary to define a *distance* to compare micro-segments and a parameter which allows the assessment of the *quality* of a micro-segment or a partition (i.e., set of micro-segments). In both cases, we use a motion histogram, defined as follows:

$$H_s[i] = \frac{N_i}{L_s}, i \in \{\text{PanLeft, ZoomIn, Fix, } \dots\} \quad (1)$$

where s represents the label of the segment inside the shot, i the motion type, L_s the length of segment s and N_i the number of frames of segment s with motion type i. Note that each bin of the histogram H shows the percentage of frames with a specific type of motion. $\sum H_s[i]$ may be higher than 1.0, because different motions can appear concurrently.

3.1. Homogeneity

We assume that a segment is perfectly homogeneous when it presents a single combination of camera motion parameters along all its frames. A segment is not homogeneous when it presents important variations on these parameters. The segment homogeneity is computed on its histogram (Eq. 1). If a segment is perfectly homogeneous, the histogram bins are equal to either 1.0 or 0.0. If a segment is not perfectly homogeneous, the bins can present intermediate values.

To measure the segment homogeneity, we measure how much its histogram differs from the ideal one. The distance corresponding to bins with high values is the difference between the bin value and 1.0. Analogously, for bins with small values, the distance is the bin value itself. Mathematically, The homogeneity of a segment s is given by:

$$\mathcal{H}(s) = \sum_i \epsilon(i), \text{ where } \epsilon(i) = \begin{cases} 1.0 - H_s[i], & \text{if } H_s[i] \geq 0.5 \\ H_s[i], & \text{if } H_s[i] < 0.5 \end{cases} \quad (2)$$

where H_s is the histogram of the segment s and i indicates the different motion types. The homogeneity of a shot S is equal to the homogeneity of its segments weighting by the length of each of them. Note that small values of $\mathcal{H}$ correspond to high levels of homogeneity.

The distance between two segments (s_1, s_2) is the homogeneity of the union of the segments $d(s_1, s_2) = \mathcal{H}(s_1 \cup s_2)$.

3.2. Temporal segmentation algorithm

This algorithm consists of the following steps:

1. *Initial over-segmentation.* In this step, the shot is over-segmented in order to obtain a set of *perfectly* homogeneous micro-segments. Mathematically, the following condition must be fulfilled: $\mathcal{H}(s) = 0, \forall s \in S$.

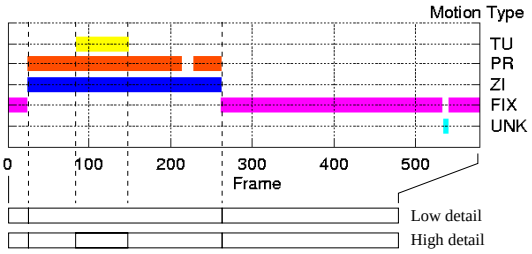

Figure 3: Example of temporal segmentation of shot #8 of MPEG-7 *nhkvideo* sequence. Above: camera motion parameters chronogram; below: two segmentations.

2. *Fusion order.* In this step, the distance between all neighboring segments (temporally connected) is computed. The closest pair of segments is then selected for possible merging in the next step.

3. *Fusion criterion.* To decide if the selected pair of segments are going to be merged, we compute the homogeneity of the shot assuming than the minimum distance segments have already been *merged* and then the following criterion is applied: merge, if $\mathcal{H}(S) \leq \Theta_{\mathcal{H}}$; do not merge, if $\mathcal{H}(S) > \Theta_{\mathcal{H}}$.

Note that the fusion criterion is global: the decision depends on the homogeneity of the resulting partition and not —exclusively— on the homogeneity of the resulting segment. If the merging is done, a new iteration starts at the second step of the algorithm. The merging process ends when there is no pair of neighboring segments that can be merged. The algorithm allows different levels of detail (see Fig. 3).

4. SHOT CLUSTERING

The shot clustering process is divided into two parts: shot merging and tree structuring. In the first step, pairs of shots are grouped together creating a binary tree. In the second step, the binary tree is restructured in order to reflect the similarity present in the video sequence.

4.1. Shot merging

The shot merging algorithm yields a binary tree which represents the merging order of the initial shots (see section 2). In this tree, the leaves represent the initial shots, the top node represents the whole sequence and the intermediate nodes represent sequences that are created by the merging of several shots. The merging criterion is defined by a distance between shots, merging first the closest shots. In order to compute the distance between shots it is necessary to define a shot model that provides the features to be compared, and to set the neighborhood links between them, which indicate what merging can be done. The *TOC* and *index* creation only differ in the neighborhood links. The former sets a link between each pair of temporally connected shots, the latter between all pair of shots.

The process ends when all the initial shots have been merged into a single node or when the minimum distance between all couples of linked nodes is greater than the specified threshold [5]. In the latter case, not a single tree but a set of binary trees —a forest— is obtained.

4.1.1. Shot and sequence model

The shot model must allow us to compare the content of several shots to decide what shots must be merged and which is their merging order. In still images, luminance and chrominance are the main features of the image [2]. In a video sequence, due to the temporal evolution, motion is an important source of information [2, 4]. So, average images, histograms of luminance/chrominance information (YUV components) and motion information (x and y components of motion vectors) are used to model the shots.

4.1.2. Merging distance

In order to compute the distance between a pair of nodes, three different cases must be considered depending on the type of nodes. At the beginning of the merging process, the nodes represent a single shot: the average of the Euclidean distance between the different components of the model is used. In the following steps of the process, the distance must be computed between nodes that may represent more than a single shot: the minimum and maximum distance between all pair of shots (one from each node) is used.

4.1.3. Merging algorithm

The merging algorithm consists of three preliminary steps and the merging process itself. In the first step, nodes are modeled according to the model previously described. In the second step, neighborhood links between shots are set and in the third step the distance between linked shots is computed.

Since the whole algorithm must be able to create both the *TOC* and the *index* of a video sequence, the criterion used to set the neighborhood links differs depending on the main objective. In the *TOC* creation process, neighborhood links are set according to temporal connectivity. On the other hand, all pair of shots will be linked and therefore may be merged in the *index* creation process.

Once the algorithm has been initialized, the merging process starts performing the following steps:

1. *Get minimum distance link.* Both the minimum and the maximum distance is computed for every pair of linked nodes. Before to check the minimum distance, the maximum distance is checked. If maximum distance is higher than the maximum distance threshold d_{max}, then the link is discarded. Otherwise, the link is taken into account. Once all links have been scanned, the minimum distance link is obtained.

2. *Check distance criterion.* In order to decide if the nodes pointed by the minimum distance link must be merged, the minimum distance is compared to the minimum distance threshold d_{min}. If the minimum distance is higher than the threshold, the process ends. Otherwise, pointed nodes are merged and the process goes on.

3. *Update links.* Links are updated to take into account the merging that has been done.

4. *Update distances.* The distance of those links which point to the new node is recomputed.

5. *Check top node.* If all initial shots have been merged into a single node, the process ends. Otherwise, a new iteration begins.

The merging process may yield a single tree or a forest depending on the degree of similarity of the initial shots. An example of binary tree for *TOC* creation is shown in Fig. 4. Inside the leaf nodes of

this tree, we have indicated its label, and between brackets [...] the starting and ending frame number of the shot. Inside the remaining nodes, we have indicated its label, between parenthesis (...) the fusion order and between brackets the minimum and maximum distance between its two siblings.

4.2. Tree structuring

The purpose of this algorithm is to restructure the binary tree into an arbitrary tree that should reflect more clearly the video structure. To this end, nodes that have been created by the binary merging process but that do not convey any relevant information should be removed. The criterion used to decide if a node must appear in the final tree is based on the variation of the similarity degree (distance) between the shots included in the node:

- If the analyzed node is the root node (or one of the root nodes, if various binary trees have been obtained after the merging algorithm), then the node should be preserved in the final tree.
- If the analyzed node is a leaf node (i.e., it corresponds to a initial shot), then it also remains in the final tree.
- Otherwise, the node will be kept in the final tree if $|d_{min}(\text{analyzed node}) - d_{min}(\text{parent node})| < \Theta$ and $|d_{max}(\text{analyzed node}) - d_{max}(\text{parent node})| < \Theta$.

As shown in Fig. 5, the tree resulting from the restructuring step represents more clearly the structure of the video sequence. Nodes in the first level of the hierarchy (28,12,13,21) represents the four scenes of the sequence, while nodes in the third —or occasionally fourth— represent the initial shots.

5. CONCLUSIONS

In this paper, techniques for *TOC* and *index* creation and video shot detection and *micro-segment* segmentation have been presented. The shot detection algorithm is able to cope with abrupt transitions as well as smooth ones even though it over-segments those portions of the video sequence with a high level of motion. In order to solve this drawback, a confidence mask will be added to the motion vectors to discard those parts of the image which can not be properly estimated when computing the *mDFD*.

The ability of the *micro-segment* segmentation algorithm to compute partitions at different level of detail allows to adjust the granularity of the resulting *TOC* or *index* to take into account different applications. Moreover, the algorithm is quite robust in front of mistakes in the camera parameters.

Finally, the *TOC* and *index* creation algorithm provides an unified method to compute both structures. Moreover, new pieces of information can easily be introduced in the merging process. For example, although the algorithm has been developed to deal with generic sequences, in the case of knowing before hand some characteristics of the sequence, this *a priori* information can be taken into account to improve the performance of the algorithm in specific scenarios.

6. REFERENCES

[1] R. Jasinschi, A. She, T. Naveen, A. Tabatabai, S. Jeannin and B. Mory, "Motion Descriptors for Content Based Video Representation", to be published in *Image Communications: Special Issue on MPEG-7 proposals*, 1999.

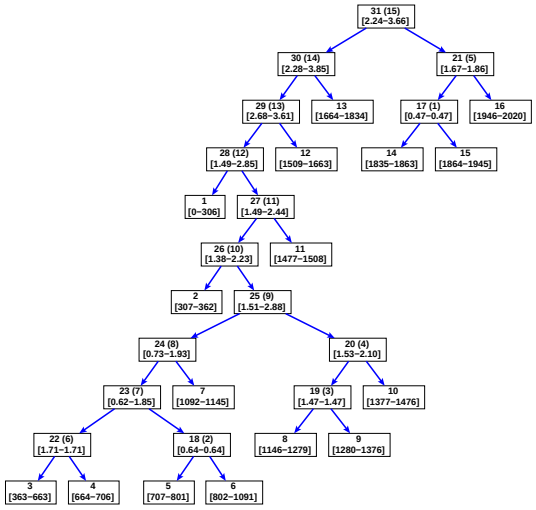

Figure 4: Binary tree created by the shot merging algorithm. The initial shots are shown in Fig. 1. The top node represents the whole sequence; the leaf nodes represent the initial shots.

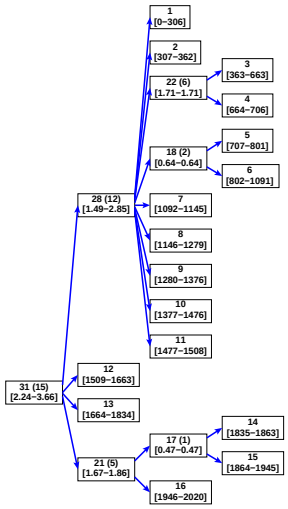

Figure 5: Tree yielded by the tree structuring algorithm.

[2] R. Mohan, “Video Sequence Matching”, in *ICASSP’98*, vol. 6, pp. 3697–3700, June 1998.

[3] C. Saraceno and R. Leonardi, “Indexing Audio-Video Databases through a joint audio and video processing”, to be published in *International Journal of Imaging Systems and Technology*.

[4] H. S. Sawhney and S. Ayer, “Compact Representations of Videos Through Dominant and Multiple Motion Estimation”, in *PAMI*, vol. 18, no. 8, pp. 814–830, 1996.

[5] M. M. Yeung and B.-L. Yeo, “Time-constrained Clustering for Segmentation of Video into Story Units”, in *ICPR’96*, pp. 375–380, 1996.