

# Automated Segmentation of Movies into Logical Story Units

Alan Hanjalic , Reginald L. Lagendijk , Jan Biemond  
Delft University of Technology  
The Netherlands

*Abstract* - We present a newly developed strategy for automated movie segmentation into logical story units. Logical story units are approximations of actual movie episodes. The proposed segmentation is the crucial first step towards a concise and comprehensive content-based movie representation. The automation aspect is becoming increasingly important with the rising amount of information to be processed in visual archives of the future. The segmentation process is designed to work on MPEG-DC sequences, taking into account that for performing content-based operations on MPEG compressed video streams at least a partial decoding is required. The proposed technique allows to carry out the segmentation into logical story units in a single pass through a video sequence.

*Index terms* - video segmentation, video content analysis

**Manuscript received** \_\_\_\_\_

*Direct all correspondence to:*

Address: A. Hanjalic  
Information and Communication Theory Group  
Department of Information Technology and Systems  
Delft University of Technology  
4, Mekelweg  
2628 CD Delft, The Netherlands  
phone: + 31 15 278 3084  
fax: + 31 15 278 1843  
e-mail: alan@it.et.tudelft.nl

## 1. Introduction

In recent years the technology has developed up to the level where vast amounts of digital information are available at low costs. At the same time digital storage media became available with a steadily increasing performance versus price ratio. The easiness and low-costs of obtaining and storing digital information as well as the almost unlimited possibility to manipulate it, make people eager to collect it and store more and more of it. We witness the rapid growth of digital archives in the professional and consumer environment. Examples are digital museum archives and Internet archives.

As a result from the developments in the field of digital video compression, the creation of digital *video* archives has become possible as well. Such video libraries are already available at commercial service providers. The expectation is that the domestic digital storage of video material will soon overtake current analog video cassette recording [9, 14, 17, 18, 19].

A particular problem with digital libraries is the management of large amounts of information. Though solutions exist for text-oriented databases (e.g. SQL), methods for browsing, querying and organization of visual information, such as images, graphics and video, and their linking to textual information are still in their infancy [12]. The MPEG-7 standardization platform [6] addresses ways of representing visual information using a standard set of descriptors, such that it can be used effectively in professional and consumer applications and especially in user-interfaces. The question as how to automatically obtain these descriptors is becoming a research topic of increasing importance. In particular the automation aspect becomes highly important with steadily rising volumes of information to be processed.

To obtain descriptors for visual information, content analysis is required. Several approaches for the content analysis of still pictures exist, most of them based on color, texture and shape analysis and comparison [5, 13, 15]. It is generally accepted that content analysis of video sequences requires a preprocessing procedure that first breaks up the sequences into temporally homogeneous segments called *video shots* [1, 2, 3, 8, 16], then condenses these segments into one or a few representative frames (*key frames*) [4, 7, 21, 24], and finally determines the relationship among shots using their key frame based representation. This last step we call video content organization. Since video streams to be dealt with in modern digital storage systems are mainly in MPEG *compressed* form, no content-related operations are possible on these streams directly. However, the complete stream decoding is not necessary either, since all processing steps can be performed on its *DC sequence* [20]. This has as main consequence that key frames are available only in subsampled formats, so-called DC images.

Most existing approaches carry out the step of video content organization by clustering video shots according to the similarity of the visual contents of their key frames. As with still pictures, the content is typically represented by color histograms, object shapes and textures. Approaches in [21, 25] show shot-based organization structures (e.g. shot cluster trees) which consider a single video shot as an elementary retrieval unit of the analyzed video. In addition to key frames temporal content variations in a video shot can be used. The scene transition graphs in [23] simulate the story flow of analyzed video sequence by temporally connecting different shot clusters.

In this paper we concentrate on *movies* as a particularly important class of video programs. We see several problems when using unconstrained shot-based cluster trees to organize movie material for retrieval purposes. In the first place, the obtained structures may be very large due to a huge number of shots in a full-length movie, and they may be non-unique and not transparent enough, especially if no

natural cluster structure exists among the video shots. Secondly, the concept of using single video shots as elementary retrieval units may be useful for some video material, but this is certainly not the most natural primitive for the retrieval of full-length movies.

We believe that for efficient movie content organization and retrieval, other strategies than an unconstrained shot clustering are required. We therefore propose a novel method to automatically segment movies into groups of video shots, contiguous in time, which approximate actual movie episodes. These groups of video shots we call *logical story units*. The result of our approach provides a top layer for the movie retrieval procedure. At lower layers, it can be enriched by existing clustering methods applied to each episode separately, to obtain a concise and comprehensive video organization scheme. The method that we propose can be carried out in parallel with the process of shot change detection and key frame extraction.

In Section 2 we will discuss the justification for our approach in more detail. Section 3 proposes the novel movie segmentation method. The proposed method is evaluated in Section 4 using several movie sequences. Conclusions to this paper can be found in Section 5.

## **2. Concept of Logical Story Units**

### **2.1 Definition of LSU**

We assume that a standard movie is produced as a series of consecutive meaningful segments, each showing strong semantic interrelations among video shots contained therein. We call these segments *episodes*. One episode lasts for several video shots and can be practically anything, from a dialog, shorter landscape scene to a longer event with complicated structure. By segmenting a movie into its episodes, a compact and comprehensive content representation can be achieved. The compactness

comes from the limited number of episodes within a standard movie, while the comprehensiveness is because humans remember episodes after watching a movie and think in terms of episodes during the retrieval process [10].

In order to obtain episode boundaries exactly, semantic information or the “ground truth” is required. We do not believe that extraction of such semantic information from a movie is feasible. We therefore approximate the episodes by *logical story units* (LSUs), which are defined on the basis of video signal’s visual characteristics and their temporal variations, measured and captured by standard image and video processing tools.

The definition of a LSU is based on the assumed temporal consistency of its visual content. It can be expected that within an episode every now and then similar *visual content elements* (scenery, background, people, faces, dresses, specific patterns, etc.) appear and some of them even repeat. Such content matching clearly may not happen immediately in successive video shots, but it will certainly within a certain time interval. The definition of a LSU can now be formulated as follows:

*A LSU is defined as a series of temporally contiguous video shots, characterized by overlapping links that connect shots with similar visual content elements.*

An illustration of a LSU and the above definition is given in Figure 1. In many of the movies that we have processed, the length of an LSU varied roughly between 1 and 10 minutes.

The above definition relies on being able to measure the visual similarity between a pair of video shots. In the next section we will propose a suitable measure and discuss it in detail. For now we assume that

we have available a *dissimilarity* measure  $A(k,k+l)$  between the shots  $k$  and  $k+l$ . Three different cases can be distinguished dependent on the relation of the current video shot  $k$  and the  $m$ -th LSU.

**Case 1:** Visual content elements from video shot  $k1$  re-appear (approximately) at shot  $k1+p1$ . In practice the maximum value of  $p1$  will have to be bounded to a fairly small number to avoid excessively long LSUs and excessive computation. Video shots  $k1$  and  $k1+p1$  then form a linked pair (illustrated in Figure 1 by the arrows). Since video shots  $k1$  and  $k1+p1$  belong to the same  $LSU(m)$ , consequently all intermediate video shots also belong to  $LSU(m)$ :

$$[k1, k1 + p1] \in LSU(m) \quad \text{if } p1 \Leftarrow \min_{l=1, \dots, c} A(k1, k1 + l) < M(k1). \quad (1)$$

Here  $c$  is the number of subsequent video shots with which the current shot is compared with to check the visual dissimilarity. The threshold function  $M(k)$  specifies the maximum dissimilarity allowed within a single LSU. Since the visual content is usually time-variant, the function  $M(k)$  also varies with the video shot under consideration. The estimation of  $M(k)$  will be discussed in Section 3.

**Case 2:** There are no subsequent video shots with sufficient similarity with video shot  $k2$ , i.e. the inequality in equation (1) is not satisfied. However, one or more shots preceding shot  $k2$  link with shot(s) following shot  $k2$  (see Figure 1). In this the current shot is enclosed by a shot pair that belongs to  $LSU(m)$ , i.e.

$$[k2 - t, k2 + p2] \in LSU(m) \quad \text{if } (t, p2 > 0) \Leftarrow \min_{i=1, \dots, r} \min_{l=-i+1, \dots, c} A(k2 - i, k2 + l) < M(k2). \quad (2)$$

Here  $r$  is the number of video shots to be considered preceding the current shot  $k2$ .

**Case 3:** If for the current shot  $k3$  neither (1) nor (2) is fulfilled, but if shot  $k3$  links with one of the previous shots, then shot  $k3$  is the last shot of  $LSU(m)$ . This can also be seen in Figure 1.

In the Section 3 we will discuss the calculation of  $A(k,n)$  and  $M(k)$ , as well as an efficient way to use the above definition in finding the LSU boundaries.

## **2.2 Use of LSU and Related Approaches**

There are various applications in digital video libraries which can benefit from LSU-based movie organization scheme. One possibility is to immediately obtain an overview over the entire movie only by looking at the LSUs and to easily retrieve each of them. Figure 2 illustrates how a movie can be broken up into LSUs and how existing content-based clustering algorithms can be applied to all video shots within a LSU. Shots on highest cluster levels can be glued together and played as movie highlights. It is also possible to browse through each individual LSU, which is an especially important feature for the case of LSUs with complicated structure. At the same time the user browses only through relevant shots that relate to the selected LSU, and is not overloaded with (the many) other shots of a sequence. For each granularity or cluster level, a key frame set is available providing video representations by pictorial summaries with different amounts of details.

Few methods dealing with higher-level movie segments can be found in literature. In [11] characteristic episodes like dialogs, high-motion and high-contrast segments are extracted for the purpose of making a movie trailer, but no attempt is done to capture the entire movie material. The audio as well as the motion information is used for extraction procedure. In [22] an approach is presented based on time-constrained clustering and label assignments to all shots in a sequence. Pre-defined models are used to analyze the resulting label sequence and recognize patterns corresponding to dialogs, action segments

and arbitrary story units. The effectiveness of this method, especially for segmenting movies into story units, depends however on the applicability of the model used for a story unit. We foresee here several practical problems such as the choice of the interval for time-constrained clustering, which puts an artificial limit on the duration of an episode. Another problem is that characterizing shots by distinct labels simplifies the real interrelation among neighboring shots far too much.

The method of detecting LSU boundaries we propose in this paper essentially finds a compromise between the totally unconstrained shot clustering approaches from [21, 23, 25] and model driven shot clustering approach from [22].

### **3. Novel approach for LSU boundary detection**

#### **3.1 Threshold Function**

The objective is to detect the boundaries between LSUs, given the definition of an LSU and the concept of linking video shots. In principle one can check equations (1) and (2) for all shots in the video sequence. This, however, is rather computationally intensive and unnecessary anyway. According to (1), if the current shot  $k$  is linked to shot  $k+p$ , all intermediate shot automatically belong to the same LSU and do not have to be checked anymore. Only if shot  $k$  cannot be linked, there is the necessity to check if at least one of  $r$  shots preceding the current shot  $k$  links with a shot  $k+p$  (for  $p>0$ , as stated in (2)). If such link is found, the procedure can continue at shot  $k+p$ , otherwise shot  $k$  is at the boundary of  $LSU(m)$ . The procedure then continues with shot  $k+1$  for  $LSU(m+1)$ . The LSU boundary detection procedure is illustrated in Figure 3.

To determine if two shots form a link, the threshold function  $M(k)$  is needed. We compute this threshold recursively from already detected shots that belong to the current LSU.

If we denote for shot  $k$  the minimum of  $A(k,n)$  found in equation (1) (or equation (2) if (1) does not hold) as the *content inconsistency value*  $C(k)$ , then the threshold function  $M(k)$  that we propose is:

$$M(k) = \alpha \bar{C}(k, N_k) \quad (3)$$

Here  $\alpha$  is a fixed parameter whose value is not critical between 1.3 and 2.0. We have used 1.4 throughout our research. Further  $\bar{C}(k, N_k)$  is computed as

$$\bar{C}(k, N_k) = \frac{1}{N_k + 1} \left( \sum_{i=1}^{N_k} C(k-i) + C_0 \right) \quad (4)$$

The parameter  $N_k$  denotes the number of links in the current LSU that have led to the current shot  $k$ , while the summation in (4) runs over the shots defining these links. Essentially the threshold  $M(k)$  adapts to the content inconsistencies found so far in the LSU. It also uses the last content inconsistency value  $C_0$  of the previous LSU for which (1) or (2) is valid, as a bias.

### 3.2 Inter-shot Dissimilarity Measure

The LSU detection algorithm and the computation of the threshold function require the use of a content-based dissimilarity function  $A(k,n)$ . In the following we define our own dissimilarity measure.

We assume that the video sequence is segmented into shots, using any of the methods found in literature [1, 2, 3, 8, 16]. Each detected video shot is represented by one or multiple key frames such that its visual information is captured as good as possible [4, 7, 21, 24]. For dynamic shots more key frames are needed than for stationary shots. Since we are using MPEG compressed video sequences, the key frames are DC images which are typically 90 x 72 pixels, i.e. 64 times smaller the original frames.

For each shot, all key frames are merged together in one large variable size image, called the *shot image*, which is then divided into blocks of  $M \times N$  pixels. Each block is now a simple representation of one element of the shot's visual content. Since we cannot expect an exact shot-to-shot matching in most cases, and the influence of particular shot content details which are not interesting for a LSU as a whole should be as small as possible, we choose to use only those features that describe the  $M \times N$  elements *globally*. Furthermore fairly large blocks have to be used, for instance  $M=N=8$  when using DC images. In this paper we use only the average color in the  $L^*u^*v^*$  uniform color space as a block's feature.

For each shot pair  $(k,n)$ , with  $k < n$ , we now would like to find the mapping between the blocks  $b_k$  and  $b_n$ , each being an  $M \times N$  block from the shot image  $k$  and  $n$ , respectively, such that

- each block  $b_k$  in a key frame of shot image  $k$  has a unique correspondence with a block  $b_n$  in shot image  $n$ . If a block  $b_n$  has already been assigned to a block  $b_k$  from a key frame belonging to shot image  $k$ , we do not allow it to be used for matching of any other block from that key frame. All blocks  $b_n$  are available only when a new key frame of shot  $k$  is to be matched. Figure 4 illustrates this in more details.
- the average distance in the  $L^*u^*v^*$  color space between corresponding blocks from the two shot images is minimized:

$$\min_{\substack{\text{all possible block combinations} \\ \text{all blocks } b}} \sum d(b_k, b_n) \quad (5)$$

where

$$d(b_k, b_n) = \sqrt{\left(L^*(b_k) - L^*(b_n)\right)^2 + \left(u^*(b_k) - u^*(b_n)\right)^2 + \left(v^*(b_k) - v^*(b_n)\right)^2} \quad (6)$$

and where all possible block combinations are given by the first item.

Unfortunately this is a problem of high combinatorial complexity. We therefore use a suboptimal approach to optimize (5). The blocks  $b_k$  from a key frame of a shot  $k$  are matched unconstrained in shot image  $n$  starting with the top-left block in that key frame, and subsequently line-fashioned scanning to its bottom-right block. A block  $b_n$  that has been assigned to a block  $b_k$  is no longer available for assignment until the end of the scanning path. For each block  $b_k$  the obtained match yields a minimal distance value  $d_1(b_k)$ . Then, this procedure is repeated for the same key frame in opposite scanning fashion, i.e. from bottom-right to top-left, yielding a difference mapping for the blocks  $b_k$  and a new minimal distance value for each block, denoted by  $d_2(b_k)$ . On the basis of these two different mappings for a key frame from shot  $k$  and corresponding minimal distance values  $d_1(b_k)$  and  $d_2(b_k)$  per block, the final correspondence and actual minimal distance  $d_m(b_k)$  per block is constructed as follows:

- $d_m(b_k) = d_1(b_k)$  , if  $d_1(b_k) = d_2(b_k)$  (7a)

- $d_m(b_k) = d_1(b_k)$  , if  $d_1(b_k) < d_2(b_k)$  and  $d_1(b_k)$  is the lowest distance value measured on the assigned block in the shot image  $n$  (one block in shot image  $n$  can be assigned to two different blocks in a key frame from  $k$ : one time in each scanning direction) (7b)

$$d_m(b_k) = \infty , \text{ otherwise.} \tag{7c}$$

- $d_m(b_k) = d_2(b_k)$  , if  $d_2(b_k) < d_1(b_k)$  and  $d_2(b_k)$  is the lowest distance value measured on the assigned block in the shot image  $n$  (7d)

$$d_m(b_k) = \infty , \text{ otherwise.} \tag{7e}$$

where  $\infty$  stands for a fairly large value, indicating that no objective best match for a block  $b_k$  could be found. The entire described procedure is repeated for all key frames of a shot  $k$ , leading to one value  $d_m(b_k)$  for each block of a shot image  $k$ .

Finally the average of the distances  $d_m(b_k)$  of the best  $B$  matching blocks in the shot image  $k$  is computed as the final inter-shot dissimilarity value:

$$A(k, n) = \frac{1}{B} \sum_{b=1}^B d_m(b_k) \quad (8)$$

The reason for taking only the  $B$  best matching blocks is that two shots should be compared only on global level. One has to allow for changes within the LSU and still be able to detect the continuity of the LSU. If  $P$  is the number of blocks within one shot picture, we found  $B=P$  to be a good value for all cases where shot  $k$  is represented by more than one key frame. If shot  $k$  contains only one key frame, the value for  $B$  should be chosen as  $P/2$ .

#### 4. Experimental validation

We have experimentally evaluated the proposed LSU boundary detection approach on four sequences, each being a part of a typical Hollywood-made movie:

- WF-1: First hour of “Four Weddings and a Funeral” (564 video shots),
- WF-2: Second hour of “Four Weddings and a Funeral” (436 video shots),
- JP-1: Ten minutes of “Jurassic Park” (91 video shots),
- JP-2: Ten minutes of “Jurassic Park” (93 video shots),
- JP-3: Ten minutes of “Jurassic Park” (107 video shots),
- JP-4: Ten minutes of “Jurassic Park” (96 video shots),

First two sequences were available in reduced form, obtained by straightforward spatial subsampling yielding a frame dimensions of 80x64 pixels. The remaining four sequences were in MPEG-2 compressed format with frame size 720x576 pixels.

We have detected the video shots using the method from [3]. Although any approach for key frame extraction can be applied for representing the shots, we used a fixed number of two key frames per shot. Since the frame dimensions of the first two sequences nearly correspond to the dimensions of DC-images of MPEG-compressed sequences, key frames are taken directly from the sequence without subsampling. In the first two sequences the first and last frame of a shot were taken as key frames, while for the other sequences DC versions of the first and the last intra-coded frame of a shot are selected as key frames. Key frames from sequences WF-1 and WF-2 can be entirely divided in 8x8 blocks, while in key frames belonging to other four sequences, the first and the last column have been left unconsidered.

The proposed algorithm requires several parameters. In our experiments it is showed that the results were fairly insensitive to the setting of these parameters, For this reason we have used the following values for all experiments:

- Multiplication factor for threshold:  $\alpha=1.4$ ,
- Look-ahead distance in searching for links for the current shot  $k$ :  $c=8$ ,
- Look-back distance in searching for links for shots preceding the current one:  $r=3$ .

To be able to evaluate the automatically obtained segmentation results, all sequences have been segmented first by hand to establish the movie episodes. These hand-produced time-consuming results are considered as “ground truth” in all experiments. The automatically detected LSU boundaries are compared with hand-produced boundaries. Table 1 gives an overview of the performance of the proposed algorithm on the test sequences. From this table all relevant information about the performance of the LSU boundary detector can be deduced, such as detection, missing, and false alarm rate.

A high percentage of correct detections is obtained, showing that the movie episodes can reliably be detected. Since movie episodes strongly relate to the “semantic” structure of a movie, we can also conclude that the proposed method - although being based entirely on color features - is useful for breaking up movies into semantically meaningful entities.

False alarms occur when LSU boundaries are detected where none exist according to the ground truth. Evaluation of the false alarm detection shows that they occur in cases where the LSU structure is very complicated. In our test sequences this occurred with nested episodes, such as embedded animations. We do not believe that these false alarms can be avoided when only visual information and simple image-based features are used.

## 5. Discussion

In this paper we presented a new approach for automatically segmenting movies into units which closely approximate actual episodes of a movie. Our approach is based on investigating the visual information of a video sequence and their temporal changes. Whether a link between two shots exists or not is determined by applying a threshold function to shot dissimilarities.

Our work shows that using only visual features of a movie sequence can provide satisfactory results for the detection of LSU boundaries. The proposed method assumes that a movie can be modeled as a series of LSUs. In cases this model assumption is violated, such as in embedded LSUs, false alarms are generated. At this place, cases should also be mentioned where *transition* shots appear, i.e. where the change from one LSU to another is not abrupt, but spread over several shots. Such shots can often be

found in movies, e.g. introducing a scenery of a new episode. For such shots, no link with any of the following shots can be established.

Though the proposed method uses only a global color feature to compare shots, an improvement in the detection rate can be obtained if object-oriented features, such as motion and shape, would be used. The use of such features will, however, not decrease the false alarm rate. On the other hand, the problem of false alarms is in this case not as important as the problem of missed detections, since it does not diminish the comprehensiveness of the obtained movie representation very strongly.

As the proposed technique computes the detection threshold recursively, and looks only a limited number of shots ahead during the LSU boundary detection, the entire process including video shot detection, key frame extraction, and LSU boundary detection, can be carried out in a single pass through a sequence.

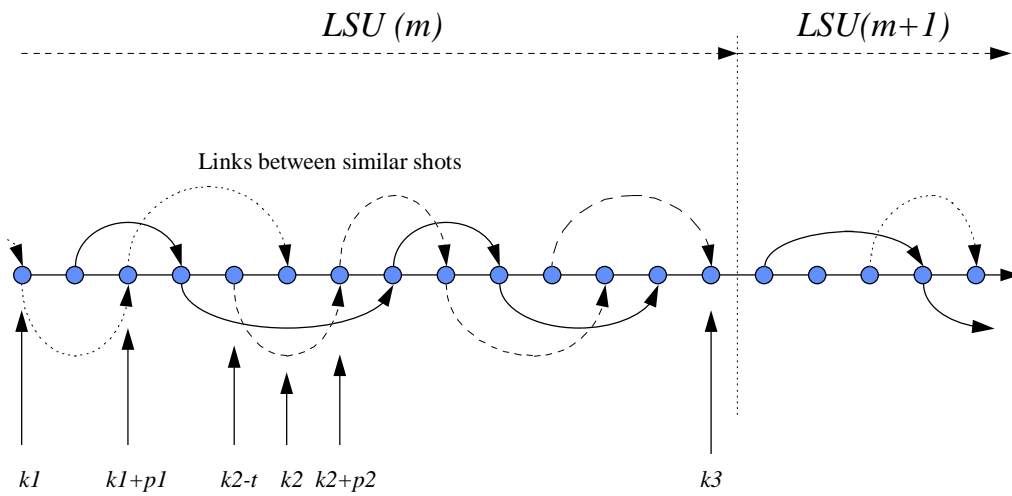
### **Acknowledgments**

We wish to express our gratitude to Philips Research in Eindhoven for providing us with test sequences. We also wish to thank the European Commission for supporting the ACTS SMASH project (AC018: *Storage for Multimedia Applications Systems in the Home*), in which we have been participating since 1995. The work presented in this paper resulted in part from our activities within this project.

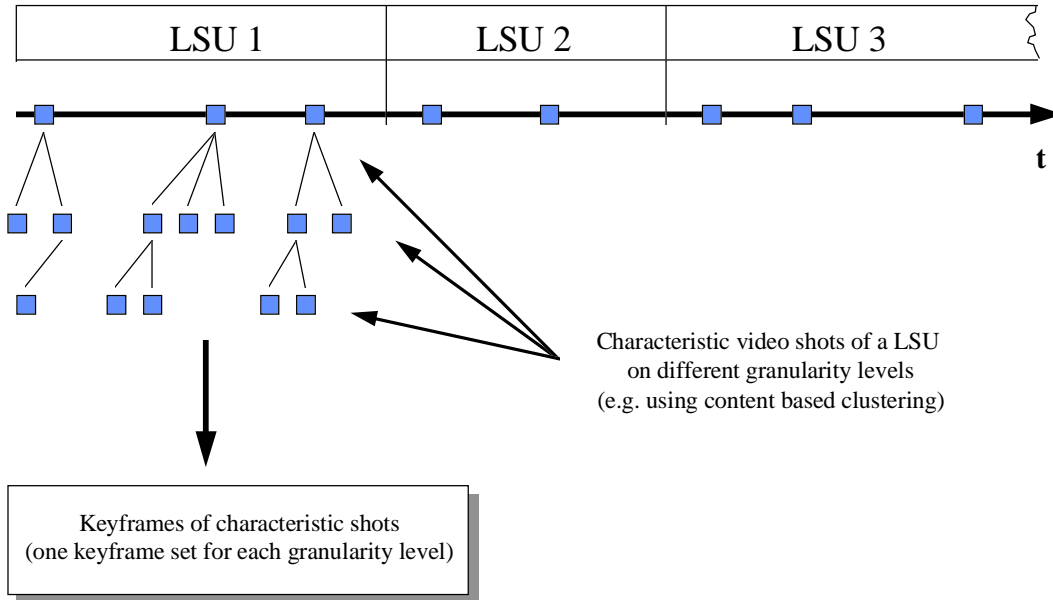
## References

- [1] Ahanger G., Little T.D.C.: “*A Survey of Technologies for Parsing and Indexing Digital Video*”, *Journal of Visual Communication and Image Representation*, Vol. 7, No.1, pp.28-43, March 1996.
- [2] Boreczky J.S., Rowe L.: “*Comparison of video shot boundary detection techniques*”, *Proceedings of IS&T/SPIE*, Vol. 2670, 1996.
- [3] Hanjalic A., Ceccarelli M., Lagendijk R.L., Biemond J.: “*Automation of systems enabling search on stored video data*”, *Proceedings of IS&T/SPIE Storage and Retrieval for Image and Video Databases V*, Vol. 3022, February 1997.
- [4] Hanjalic A., Lagendijk R.L., Biemond J.: “*A New Method for Key Frame based Video Content Representation*”, *Proceedings of First International Workshop on Image Databases and Multi Media Search*, Amsterdam, 1996 (to appear in *Image Databases and Multi Media Search*, World Scientific, Singapore).
- [5] He S., Abe N.: “*A Structural Representation and its Application to Image Retrieval*”, *Proceedings of IEEE First Workshop on Multimedia Signal Processing*, Princeton, 1997.
- [6] ISO/IEC JTC1/SC29/WG11, “*MPEG-7: Context and Objectives (v.4)*”, July 1997.
- [7] Lagendijk R.L., Hanjalic A., Ceccarelli M.P., Soletic M., Persoon E.: “*Visual Search in a SMASH System*”, *Proceedings of IEEE ICIP 1996*.
- [8] Nakajima Y., Ujihara K., Yoneyama A.: “*Universal scene change detection on MPEG-coded data domain*”, *Proceedings of IS&T/SPIE*, Vol. 3024, February 1997.
- [9] Okamoto H. et al.: “*A Consumer Digital VCR for Advanced Television*”, *IEEE Transactions on Consumer Electronics*, Vol. 39, No.3, pp. 199-204, August 1993.
- [10] van Paasen R.: “*Subjective Representation of Video: An Exploratory Study*”, MSc. Thesis, Eindhoven University of Technology (NL), 1997.
- [11] Pfeiffer S., Lienhart R., Fischer S., Effelsberg W.: “*Abstracting Digital Movies Automatically*”, *Journal of Visual Communication and Image Representation*, Vol. 7, No. 4, pp. 345-353, December 1996.
- [12] Picard R.W.: “*Light-years from Lena: Video and Image Libraries of the Future*”, *Proceedings of IEEE ICIP 1995*, Vol.1, pp. 310-313.
- [13] Servetto S., Ramchandaran K., Huang T.S.: “*A Successively Refinable Wavelet-Based Representation for Content-Based Image Retrieval*”, *Proceedings of IEEE First Workshop on Multimedia Signal Processing*, Princeton, 1997.
- [14] The SMASH project home page: <http://www-it.et.tudelft.nl/pda/smash>.

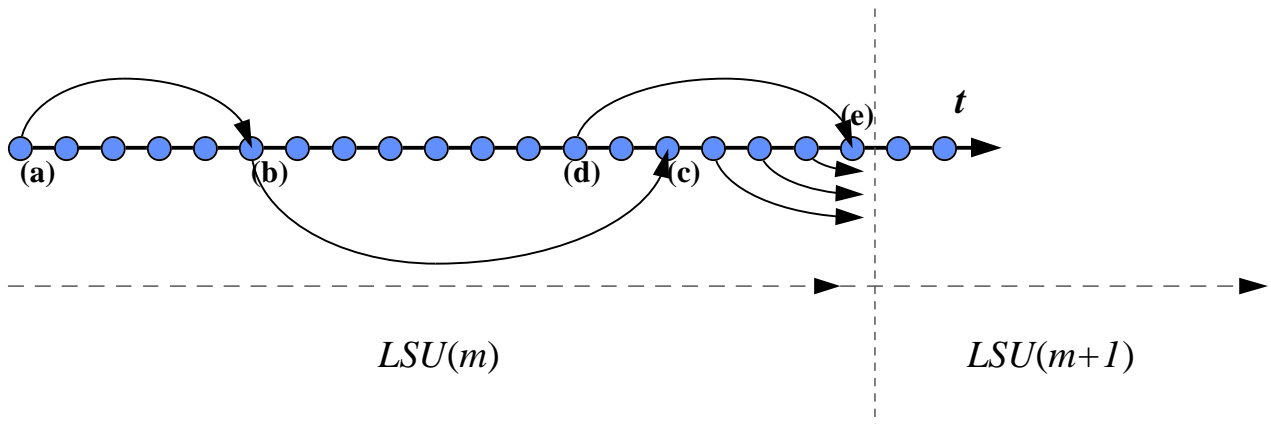
- [15] Smith J.R., Chang S.-F.: “*SaFe: A General Framework for Integrated Spatial and Feature Image Search*”, Proceedings of IEEE First Workshop on Multimedia Signal Processing, Princeton, 1997.
- [16] Wei Q., Zhang H., Zhong Y.: “*A Robust Approach to Video Segmentation Using Compressed Data*”, Proceedings of IS&T/SPIE Storage and Retrieval for Image and Video Databases V, Vol. 3022, February 1997.
- [17] de With P.H.N.: “*Data Compression Systems for Home-Use Digital Video Recording*”, IEEE Journal on Selected Areas in Communication, Vol 10, No. 1, pp. 97-121, Januar 1992.
- [18] de With P.H.N., Rijckaert A.M.A., Keesen H.-W., Kaaden J., Opelt C.: “*An Experimental Digital Consumer HDTV Recorder using MC-DCT Video Compression*”, IEEE Transactions on Consumer Electronics, Vol. 39, No.4, pp. 711-722, November 1993.
- [19] Yanagihara N., Siu C., Kanota K., Kubota Y.: “*A Video Coding Scheme with a High Compression Ratio for Consumer Digital VCRs*”, IEEE Transactions on Consumer Electronics, Vol. 39, No. 3, pp 192-198, August 1993.
- [20] Yeo B.-L., Liu B.: “*On the Extraction of DC Sequence from MPEG Compressed Video*”, Proceedings of IEEE ICIP, 1996.
- [21] Yeung M.M., Liu B.: “*Efficient Matching and Clustering of Video Shots*”, Proceedings of IEEE ICIP 1995, vol. 1, pp. 338-241, Washington DC, USA, 1995.
- [22] Yeung M., Yeo B.-L.: “*Video Content Characterization and Compaction for Digital Library Applications*”, Proceedings of IS&T/SPIE Storage and Retrieval for Image and Video Databases V, Vol. 3022, February 1997.
- [23] Yeung M., Yeo B.-L., Wolf W., Liu B.: “*Video Browsing using Clustering and Scene Transitions on Compressed Sequences*”, Proceedings of IS&T/SPIE Multimedia Computing and Networking, February 1995.
- [24] Zhang H., Low C.Y., Smoliar S.W., “*Video Parsing and Browsing using Compressed Data*”, *Multimedia Tools and Applications*, vol. 1, pp. 89-111, Kluwer Academic Publishers, 1995.
- [25] Zhong D., Zhang H., Chang S.-F.: “*Clustering Methods for Video Browsing and Annotation*”, Proceedings of SPIE, Vol. 2670, pp. 239-246, 1996.



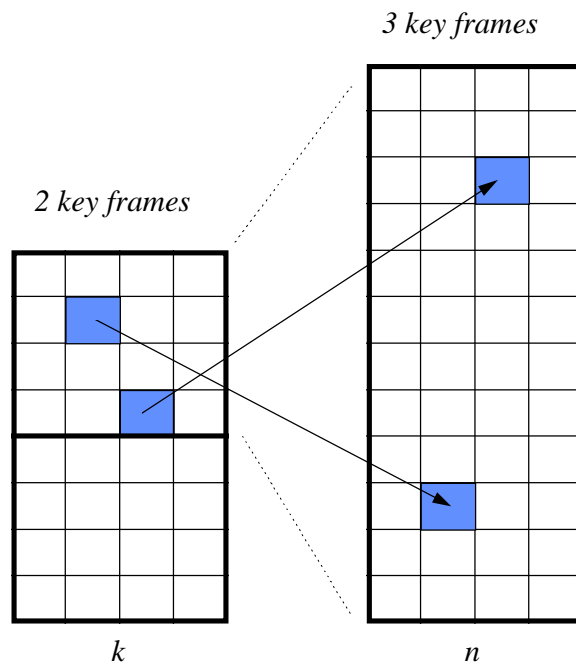
**Figure 1:** Illustration of a LSU characterized by overlapping links connecting similar video shots.



**Figure 2:** Possible scheme for movie representation using LSUs



**Figure 3:** *Illustration of the LSU boundary detection procedure. The shots indicated by (a) and (b) can be linked and are by definition part of  $LSU(m)$ . Shot (c) is implicitly declared part of  $LSU(m)$  since the shot (d) preceding (c) is linked to a future shot (e). Shot (e) is at the boundary of  $LSU(m)$  since it cannot be linked to future shots, nor can any of its  $r$  predecessors.*



**Figure 4:** *Comparison of shot  $k$  with shot  $n$  by matching  $M \times N$  blocks from each key frame of shot image  $k$  with shot image  $n$ . The shot  $k$  had 2 key frames and shot  $n$  had 3 key frames.*

**Table 1:** *LSU boundary detection results*

<b>Sequence name</b>	<b>Number of LSUs “Ground truth”</b>	<b>Number of correctly detected LSU boundaries</b>	<b>Number of false alarms</b>
FW-1	6	5	2
FW-2	8	7	0
JP-1	5	4	0
JP-2	4	3	0
JP-3	2	2	3
JP-4	1	1	0
<b>TOTAL</b>	26	22 (=85%)	5 (=19%)