

Automation of systems enabling search on stored video data

Alan Hanjalic , Marco Ceccarelli, Reginald L. Legendijk , Jan Biemond*

Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
Phone: +31-15-2783084, Fax: +31-15-2781843, e-mail: {alan,inald,biemond}@it.et.tudelft.nl
* Philips Research Laboratories, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
Phone: +31-40-2742469, Fax: +31-40-2744648, e-mail: marco@natlab.research.philips.nl

Abstract

In the European project *SMASH* [<http://www-it.et.tudelft.nl/pda/smash>] mass-market storage systems for domestic use are under study. Besides the storage technology that is developed in this project, the related objective of user-friendly browsing/query of *video data* is studied as well. Key issues in developing a user-friendly system are (1) minimizing the user-intervention in preparatory steps (extraction and storage of representative information needed for browsing/query), (2) providing an acceptable representation of the stored video content in view of a higher automation level, (3) the possibility for performing these steps directly on the incoming stream at storage time, and (4) parameter-robustness of algorithms used for these steps. This paper proposes and validates novel approaches for automation of mentioned preparatory phases. A detection method for abrupt shot changes is proposed, using locally computed threshold based on a statistical model for frame-to-frame differences. For the extraction of representative frames (key frames) an approach is presented which distributes a given number of key frames over the sequence depending on content changes in a temporal segment of the sequence. A multimedia database is introduced, able to automatically store all bibliographic information about a recorded video as well as a visual representation of the content without any manual intervention from the user.

Keywords: Shot change detection, key frame extraction, video databases, consumer digital services

1 Introduction

With the continuous increase in capacity of modern storage media, the concept of *digital libraries* becomes technically realizable [1]. Two types of such systems can be distinguished, namely large public digital libraries and smaller domestic libraries. A public digital library is reachable via wide-area networks by users world-wide, while a domestic system (at home) stores data received from the network for an individual consumer. In both types of digital libraries browsing/query facilities are required. Because of the huge amount of stored data, browsing/query typically takes place on a much smaller data set that, in one way or another, characterizes the stored information [3]. This representative data set we call an *abstract*. Although browsing/query on the two types of systems is conceptually the same, there is an important difference concerning the creation of that abstract.

In large public digital libraries different preparatory annotation activities are carried out on the stored video data so that its representation is optimal for subsequent user browsing/query processes. These activities are usually time-consuming and may include manual insertion of key words, and the selection and organization of the various information types stored. One important and time-consuming aspect of organization is the indexing of video sequences. For consumer storage systems such annotation and selection activities should be carried out fully automatically, with minimized interaction from the side of the consumer. Notwithstanding the automated procedures, an acceptable quality of the video abstract is required. Another increase in user-friendliness in consumer storage systems can be obtained by speeding-up all selection and annotation steps prior to the actual user-interaction. Therefore, it is desired to form an abstract at recording time. Furthermore, the implemented algorithms for performing the abstract-making operations should be sequence-independent.

In the European research project SMASH [<http://www-it.et.tudelft.nl/pda/smash>] a large capacity consumer storage device for multimedia data is under study. Besides the storage technology that is developed in this project, the related objective of user-friendly browsing/query of *video data* is studied as well. The described requirements for making a video abstract are taken into account in the system design.

In Chapter 2 we propose an automated detection method for abrupt scene changes which uses locally computed thresholds based on a statistical model for frame-to-frame differences. It enables the detection directly on the incoming video stream, and has similar performance for any video sequence. This performance can be estimated using statistical parameters of the model. For automated extraction of representative frames (key frames) providing a *content-based* representation of the stored video, an approach is presented in Chapter 3. It distributes a given number of key frames over the sequence depending on content changes in its elementary temporal segments. This procedure is also done directly on the incoming sequences. To apply the proposed parsing and extraction algorithms on commercial *compressed* video streams, decoding of these streams is needed. How this can be done without a drastic increase in computational complexity, is described in Chapter 4. Considering the availability of a capacity large enough to store several hours of video, there is the possibility to have a large number of programs stored in the system at the same time. A considerable amount of representative information useful for browsing/query can be obtained for each of these programs. In Chapter 5 a multimedia database is introduced, able to store all available bibliographic information about the stored video and a visual representation of the content (key frames) without any manual intervention from the user. In Chapter 6 some conclusions related to proposed methods can be found.

2 A video parsing algorithm using optimal thresholding

Video can be seen as an ordered collection of unbroken and continuous series of frames, called shots. The first step in the process of creating a video abstract is the detection of these shots in the video stream. This temporal segmentation process is also called video parsing.

2.1 General problem of video parsing

Video parsing is based on measuring changes between consecutive frames. In measuring these changes only real *content-changes* should be taken into account. For instance, small movements of objects on a static background, camera movements, focal length changes, or changes in luminance should not be registered as relevant changes [8]. To measure relevant contents changes we need an appropriate feature describing the frame content and a metric to evaluate changes of that feature. This results in a frame-to-frame difference time function $FFD(k)$. The way of obtaining the $FFD(k)$ in this paper is explained in 2.3.1.

Using the $FFD(k)$ we can detect shot changes. Abrupt shot changes can be detected as sharp peaks in the FFD , while gradual transitions between shots require more elaborate detection mechanisms. Existing detection approaches do not comply with requirements needed for consumer storage systems. This is the case even for detecting sharp peaks in the FFD curve. In the following we therefore concentrate on the problem of abrupt shot changes detection.

2.2 Current approaches for detection of abrupt shot changes

Only two thresholding approaches for detecting sharp transitions in the $FFD(k)$ function can be found in the literature. In [7] the use of a fixed threshold for the entire video sequence is proposed. This threshold is determined under the assumption that $FFD(k)$ is Gaussian distributed, except for those values resulting from shot transitions or camera movements. The drawback of this approach is that statistical information of the entire $FFD(k)$ function is needed, implying that the entire video sequence has been stored. Another problem when using a global threshold appears in cases where a very distinguishable break-peak can be observed in one stationary part of the sequence, but whose height is similar to FFD values along a high-action shot in another portion of the sequence. An example for this can be seen in Figure 1.

In [2] a local threshold is determined within a sliding window containing several last computed FFD values. The window is determined such that it cannot contain more than one shot change. For the case of sharp shot changes the middle window point is always checked (1) to be the maximum and (2) to be X time higher than the second largest FFD value in the window.

Computing the threshold locally, i.e. only using the information from a short temporal segment has several important advantages:

- the threshold function changes much more rapidly, following the function $FFD(k)$. This can enable the proper detection even in cases like in Figure 1 without causing false alarms.
- It is suitable for on-the-fly video parsing since only the current information about the sequence is needed.
- It possesses the “forgetting” property, i.e. the detection performance is not influenced by missed and false detections which happened before.

However, a disadvantage of the method proposed in [2] is that the overall performance of the system is rather sensitive to the setting of the parameter X , which makes it unsuitable for our purposes.

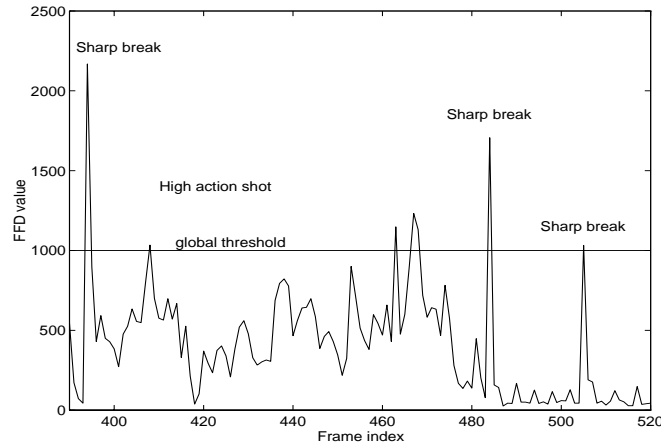


Figure 1: Problems by using global threshold

2.3 Automated video parsing

We propose a method for detecting sharp shot changes using all advantages of local thresholding but being far more sequence-independent than the method from [2], for any given parameter value. For this purpose we introduce a general statistical model for the FFD curve. Such a model enables us to work with *detection probabilities*, as shown in 2.3.3.1.

2.3.1 Features and metrics

Popular features to measure global frame contents are the color and intensity histograms [2, 3, 4, 8]. We have used YUV histograms with 64 bins for the chrominance and 32 bins for the luminance information. As metric we used the sum of the absolute histogram differences to obtain FFD values:

$$d_{YUV}(k, k-1) = \sum_{j=Y,U,V} \sum_i |h_k^j(i) - h_{k-1}^j(i)| \quad (1)$$

2.3.2 Statistical model for the $FFD(k)$ curve

In [7] it was observed that $FFD(k)$ (there obtained by comparing color code histograms) can be regarded as a realization of an uncorrelated Gaussian process, if no shot change or motion is present. This observation we like to extend to any other temporal segment with *uniform* content development, independent of the present amount of action. Within a single shot the $FFD(k)$ can then be modeled either as a single uncorrelated Gaussian process or as a temporal concatenation of multiple uncorrelated Gaussian processes. Shots themselves are separated by individual large-valued outliers, or peaks.

As an illustration of this, consider a camera following a soccer player. If initially the camera is stationary, frame-to-frame differences are caused by the moving of the soccer player within the frame window. The resulting variations of $FFD(k)$ will

be around one mean value, and are the result of a large number of small differences. The central limit theorem tells us that the resulting summed difference will be approximately Gaussian. If the camera starts panning to follow the running player, the mean and variance of $FFD(k)$ will shift to other values. If then another camera view is selected, a shot change results, yielding a peak in $FFD(k)$ followed by FFD values around another mean and with another variance. Another example would be a still camera taking from the close distance the running crowd during the marathon race. Due to a high amount of action, FFD values might be large, but they remain grouped around a (large) mean value and having a (large) variance.

Figure 2 illustrates the FFD curve of a high-action shot. At the onset and at the end of the shot the peaks due to the shot change can be seen. Figure 2 also shows the $FFD(k)$ histogram and a Gaussian curve derived from the mean and standard deviation estimated from the FFD values.

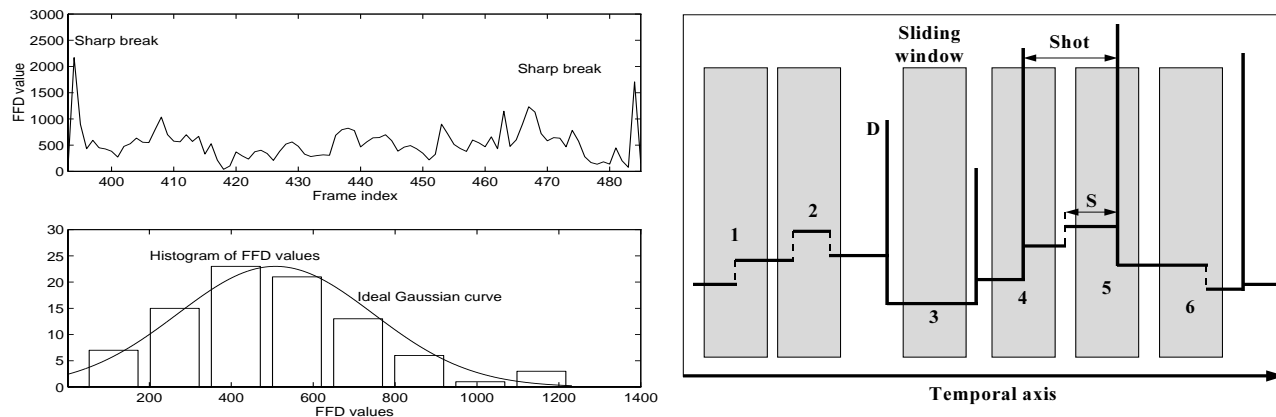


Figure 2: Statistics of a shot with strong but uniform action

Figure 3: Illustration of a statistically modeled fictive $FFD(k)$ curve with some possible positions (numerated) of the sliding window.

From the many results published in the literature and from our own experiences we conclude that a statistical model for the $FFD(k)$ function should at least show the following properties:

- $FFD(k)$ has an underlying two state model, with state “S” for being within a Gaussian shot segment, and with state “D” for shot changes. A state “S” can be followed by another state “S” or by state “D”. State “D” is always followed by state “S”;
- Each state “S” has three parameters determining the process that generates $FFD(k)$ in that state, namely: the duration of the state L, the mean and variance of the uncorrelated Gaussian process $FFD(k)$ in that state;
- State “D” has duration 1;

In Figure 3 a statistical model of a fictive $FFD(k)$ function can be seen, with each Gaussian segment “S” represented by its mean value. In principle one could now develop a recursive detector that determines the states from which the observed $FFD(k)$ values originate. This would automatically lead to a shot change detector. In the following, however, we will extend the sliding window thresholding approach from [2] to take the above model for $FFD(k)$ into account.

2.3.3 Sliding window thresholding

In the sliding window approach, detection takes place on the center value of the window [2]. The length of the sliding window is selected such that it is highly unlikely that two shot changes occur within the window. Then, at a frame-rate of n frames/second the length of the window can be chosen as not larger than $2n-1$, if it is assumed that two shot changes within one second are rare. Another issue by defining the window dimension is the minimal length of each “S” state. We assume that this minimal value of L corresponds to one half of the window length.

Detection of a shot change requires the detection of an outlier in the window. This detection takes place in two steps, namely:

- the center value in the window must be the largest FFD value in the window;

- the center value must be larger than the locally computed threshold.

The central question is still what the threshold value in step 2 should be. To determine this threshold, let us consider the various situations that can occur on the basis of the statistical model for $FFD(k)$ described in the previous section. Due to the step 1, there is no need to compute the threshold and start the detection procedure if the center window value is not the maximum of the window. Therefore, we are interested mainly in cases where this condition is satisfied. Examples of these cases can be seen in Figure 3 and can be divided in two classes:

- *Class 1*: all cases with two “S” states surrounding the middle of the window (positions 4, 1 and 3)
- *Class 2*: all other cases than in *Class 1* where the middle window point is only by chance the maximum (possible in positions 2 and 6).

2.3.3.1 Discussion on Class 1

Assuming that P is the given probability for false detection of a shot change, the threshold for the sliding window in this case can be determined from the equation

$$P = 0.5 * \int_{Threshold}^{\infty} (p_{left}(z) + p_{right}(z)) dz \quad (2)$$

with $p_{left/right}(z)$ being the function of the Gaussian distribution for the left/right side of the window. P is the only parameter to be inserted manually. It determines the probability that a FFD value belonging to a “S” state is higher than the computed threshold, which would lead to false detection. Also an approximate solution for this threshold can be used, where only the *dominant* integral contribution in (2) is taken into account. The threshold is then given as

$$T = \mu + \alpha * \sigma \quad (3)$$

with the parameter α corresponding to the given probability P and the statistical parameters taken for the normalized dominant Gaussian curve. We found the detection performance for this approximate threshold value to be fully acceptable, compared with the threshold from (2). More detailed discussion dealing with missed detections can be found in 2.4.

2.3.3.2 Discussion on Class 2

The main problem in cases belonging to this class comes from the attempt to determine the threshold, as described in 2.3.3.1, whereby at least on one window side a transition between two Gaussian segments can be found. For that window side the usage of introduced formulas is expected to give unpredictable results, which may lead to false or missed detections for these window positions.

2.4 Discussion on the overall detection performance

Although the cases from the *Class 2* can be understood as a source of possible problems in the detection, we found that their influence on the overall detection performance is not large. This leaves us the well modeled *Class 1*, as described in 2.3.3.1, to be discussed in more details.

The performance of a detection algorithm is evaluated by estimating the probability for false and missed detections. The probability for false detections is directly controlled by the parameter P . Typical values for P are 0.05 and 0.01 , resulting in appropriate values for α . Due to unknown general distribution of values belonging to “D” states, it is difficult to formulate the similar sequence-independent probability for missed detection. However, if the peaks in the $FFD(k)$ function are distinguishable enough from all other FFD values in a sequence, the distribution of “D” states is also strongly separated from any Gaussian distribution of “S” states. A “universal” threshold providing an acceptable performance for both false alarms and missed detections, can be set in the ideal case in the area in-between, where both distributions go towards zero. A near-to-ideal case can be approached by using well selected features and metrics for measuring FFD values. Using this analysis and metric (1), we obtained acceptable results for α between 5 and 6.

In the method from [2] a proper choice for the parameter X can be made to optimize the detection for a limited set of test-sequences, but no conclusions about the performance can be made if any other sequence is taken, not belonging to that set. Therefore, it cannot provide the generality and robustness of the detection performance, required for fully automated systems, as it is possible with our novel approach for any given parameter P .

2.5 Experimental results

The performance of the detection algorithm has been evaluated on “real-movie” sequences with 80x64 subsampled frames (different segments from “Four Weddings and a Funeral” (courtesy of Polygram), “Nature”). Sequences used in tests have variable global content developments, and contain stationary as well as high-action shots. Also the cases with different content developments within one and the same shot were not rare. Tests have been made using the approximate threshold computation described in 2.3.3.1. In view of the analysis from the last section, we worked only with one “universal” threshold. Best performance we obtained for $\alpha = 5$. From the total number of sharp shot changes of 130 in all sequences together, 2 missed detections (1.5%) and 1 false alarm (0.8%) were registered. Each of these three detection mistakes happened in different sequences, leading to a similar detection performance in each of them for the given value of α . The window length was chosen as 21.

3 An automated approach for key frame extraction

Video representation through characteristic frames (*key frames*) has been addressed very frequently in literature (e.g. [2, 3, 9, 10, 11]). If these frames are extracted using content-based video sampling, the content of each segment of the sequence can be recognized very easily by looking at given samples. Such video content representation appears thus to be very suitable for the purpose of video browsing. Also when considering query processes where the search for video parts containing some specific objects, persons or features is performed, the concept based on key frames can be very efficient since features collected from key frames can be used.

A simple method to select key frames is to take the first frame of each shot [10]. More reliable content representation requires non-uniform sampling of the video shot. Current approaches [9, 11], based on measuring the differences between the last selected frame and the remaining frames and extracting a subsequent key frame if the measured difference exceeds the given threshold, are typically sequential processes leading generally to unpredictable results. In particular, the final number of key frames for the whole sequence cannot be estimated for any given threshold. We can end up with a huge number of key frames or simply with too few key frames - not enough for browsing. This problem is also related to the available space for storing extracted key frames. Secondly, it is rather difficult to relate any particular parameter value by threshold setting to the key frame collection resulting from that setting. Furthermore, it is based on “subjective” thresholds which is not acceptable in fully automated and widely used systems.

Aiming at more objective and controlled key frame based video representation, suitable for applications in a consumer storage system, we have developed a new two-step key frame extraction method [5, 6]. In the first step, on-the-fly assignment of a number of key frames per shot is carried out depending on the content of the shot and on the past content development. This key frame assignment is done such that the sum of all assigned key frames along the sequence is close to a given maximal number of allowable key frames N for the entire sequence. The number N can be adjusted depending on the type of the program to be stored. More details about this can be found in Chapter 5. As a measure C_i of content in the shot i we have found the sum FFD values along the shot sufficiently representative. C_i is then defined as

$$C_i = \sum_{n=2}^L FFD(n) \quad (4)$$

where the FFD values are here computed using the equation (1). Further, n is the frame index and L the number of frames in the shot. The on-the-fly key frame assignment algorithm using the defined content measure C_i , takes on the following form:

$$K_i = \frac{C_i}{\sum_{u=1}^S C_u} N = C_i \frac{N}{T} \frac{T}{\sum_{u=1}^S C_u} \approx C_i \frac{N}{T} \frac{\sum_{u=1}^i T_u}{\sum_{u=1}^i C_u} \quad (5)$$

K_i represents the assigned number of key frames to the shot i , T is the total sequence length, and T_u is the length of the shot u . C_u is the content of the shot u and S is the number of shots in the entire sequence.

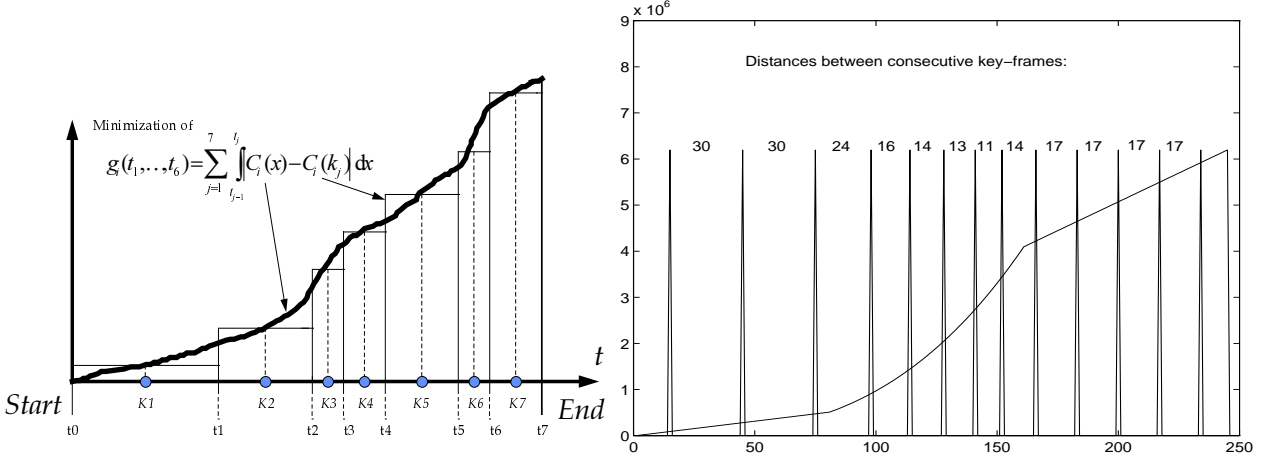


Figure 4: Illustration of the key frame distribution within a video shot by assigned 7 key frames. An approximation of the accumulation curve can be obtained using 7 flexible rectangles.

Figure 5: Application of the approach to a fictive video shot. Obtained variable key frame densities picture the actual content development

The assignment step in (5) is followed by a threshold independent and objective procedure for content-based distribution of the assigned number of key frames along each video shot [6]. Key frame distribution along the shot results from minimizing the following criterion function:

$$g(k_1, \dots, k_{K_i}, t_1, \dots, t_{K_i-1}) = \sum_{j=1}^{K_i} \int_{t_{j-1}}^{t_j} |C_i(x) - C_i(k_j)| dx \quad (6)$$

Here k_j ($j=1, \dots, K_i$) are the temporal positions of the key frames, while t_{j-1} and t_j are the *breakpoints* between the shot segments that are represented by key frame k_j . Note that t_0 and t_{K_i} are the (known) temporal begin and endpoints of i -th shot. The non-

decreasing function $C_i(x)$ represents the content development of the shot i , where each function value is obtained by accumulating *FFD* values along the shot up to the frame x . With (6) we indicate that we wish to approximate the actual content development by using the curve $C_i(k_j)$ composed out of rectangles, each one defined by k_j and t_j and each corresponding to one key frame. The minimization process gives optimal positions of key frames along the shot, such that their (variable) density best simulates the actual content development. Figure 4 illustrates the distribution approach by 7 assigned key frames to a shot with a given content development. Figure 5 shows the result of the key frame distribution over a single (fictive) shot after applying the described approach.

4 Application on MPEG-compressed streams

The incoming video stream will be available in the consumer storage system in a *compressed* form. Current compression methods used by commercial stream-providers are mainly MPEG based (e.g. DVB, as explained in Chapter 5). Since no content-based operations are possible directly on the MPEG-compressed stream, the decoding process must be performed on the streams prior to video parsing and key frame extraction. However, the complete decoding of MPEG streams is not necessary in order to perform the operations on the stream content. The described video parsing and key frame extraction method can be performed on the *DC sequence* [2].

Usage of the DC sequence for video processing has several important characteristics: it shortens the decoding process considerably so that all the processing on the stream can be done at recording time, DC images are very suitable for video parsing since working with “averaged” blocks suppresses small unimportant changes of pixel-values from frame to frame which could cause a lower reliability of the video parsing process, DC images are very suitable as key frames and they do not use too much storage space when building an abstract for browsing/query.

5 Automated generation of annotations for the video database

The consumer storage server will be able to store several video programs on a single storage medium. The content of each video can be represented through different monomedia components, including visual information (e.g. key frames), relevant audio clips and textual information. In every instance, the more organized the data in a system are, the more useful the information becomes. Therefore, a database management system (DBMS) must be provided for efficient collection, retrieval and reporting of information.

The application domain imposes particular requirements on the DBMS: it must be a light weight, embedded system, but at the same time fast and capable of managing large objects. As we are focusing on consumer applications, the system must not need any specific administrative operation and it must automatically take care of all the issues involved with management of data. It is expected that a database in a consumer storage device collects all relevant information without requiring manual insertion of data. Hence, extraction of meta-data must be automatically performed in real-time during storing operation. The user must be left the chance to insert or replace textual information in the generated database, though he should not be supposed to do so.

The visual information is essential for representation of content of video programs. However, in order to facilitate the search task, its integration with textual descriptors is needed. For that reason, the problem of extracting meaningful textual annotations (meta-data) to describe video content is one of the main issues in video database creation. While most systems solve this problem by manually inserting annotations, some other introduce automation in the indexing process by annotating structural aspects or low level features of the video in its elementary units, as camera or object motion in the shots [13, 14]. Considering our domain of application, such low level features do not provide a meaningful description of the content to the end user. Semantic attributes of the video program and its characteristics are needed, referring to the program in its entirety. We will refer to this kind of descriptors as *global meta-data*. Unfortunately, bibliographic information cannot be extracted from the video stream itself. Therefore, we will analyze techniques for automatic extraction of high level descriptors from other possible sources pertinent to our specific application, either the service provider itself or third parties.

5.1 Collecting global meta-data

The focus of our system is on storage of digital video services: DVB (Digital Video Broadcasting) is the European standard for transmission of digital services [15], now being adopted also in other countries and organizations worldwide (e.g. by Davic). For video compression and transmission, the MPEG2 standard is used. The Program Specific Information (PSI) of MPEG2 Systems [16] defines four tables which contain the necessary information to demultiplex and present programs via a Transport Stream. Some information like copyright and language can be obtained from the Program Map Table. In order to provide identification of services and events for the user, additional data is provided by the DVB standard in the Service Information Tables (DVB-SI) [17]. The DVB-SI is organized in six tables. The most useful ones for our application are the Service Description Table and the Event Information Table. These tables are embedded in the transport stream which contains the program itself. During or before recording time, the transport stream packets containing the PID code corresponding to these tables are extracted. From each table, descriptors are automatically captured and inserted as attributes of a general *Video_program* class. Some of these attributes are listed in Table 1.

Other global descriptors can be extracted via network. As a matter of fact, the distance of the world of mass consumer equipment from Internet is steadily becoming shorter. The Davic (Digital Audio Video Council) forum [18] is working on providing solutions for interactive services and Internet access on TV sets. In the context of collection of information related to a specific video program, it is possible to exploit the boundless resources of Internet for adding more details about the recorded item. The target of search can be a database about movies on a web site (see for example the sites imdb.com, allmovie.com) or the official site of the service provider or other web sites which provide general information about the scheduled programs (Web TV guides, as eurotv.com).

<i>event_id</i>	event identifier	<i>obj_id</i>	icon object identifier
<i>event_name</i>	title	<i>event_id</i>	related event
<i>content_descr</i>	subclasses (es: Movie - Comedy)	<i>format</i>	picture format (es:90x72)
<i>short_descr</i>	bibliographic descriptor	<i>encoding</i>	encoding format
<i>ext_descr</i>	more details (director, cast, summary...)	<i>time_code_begin</i>	beginning of the video segment
<i>serv_type</i>	service type (es: NVOD)	<i>time_code_end</i>	end of video segment
<i>prov_name</i>	service provider name	<i>time_code_kf</i>	position of key frame in the segment
<i>bouquet</i>	bouquet name	<i>address</i>	location on the storage media
<i>length</i>	duration in hours, minutes, seconds	<i>length</i>	duration in time of the segment
<i>date</i>	date of recording	<i>transit_in</i>	“in” mixing effect (cut, fade, dissolve)
<i>format</i>	display format	<i>transit_out</i>	“out” mixing effect
<i>asp_ratio</i>	aspect ratio (4:3, 16:9...)	<i>cam_oper</i>	camera operations (zoom, pan, tilt...)
<i>audio</i>	audio format (mono, stereo, surround...)	<i>level</i>	level of hierarchy
<i>language</i>	which languages (can be multilingual)	<i>cluster</i>	number of group
<i>subtitles</i>	foreign movies or for hard of hearing	<i>parent_node</i>	pointer to parent key frame
<i>teletext</i>	presence of embedded teletext services	<i>children_node</i>	pointer to first child key frame
<i>par_rating</i>	parental rating: country	<i>audio_oid</i>	identifier of related audio clip
<i>address</i>	location of stream on the storage system	<i>text_id</i>	related text from subtitles
...	...		

Table 1: Global descriptors: some of the attributes of the general *Video_program* class, which can be extracted from the DVB Service Information Tables.

Table 2: Local descriptors: some attributes of the *key_frame* class

5.2 Local meta-data

Every video program is analyzed and parsed in segments as described in previous chapters. The information describing the characteristics of the elementary units forming the structure of each single video program is referred to as *local meta-data*. Automatic fusion of the data resulting at recording time from several extraction modules is required for optimal management of the resources, prior to indexing in the database. The key frame extraction routines provide icons able to visually represent the content of segments of video. Each icon, with the description of its characteristics, must be integrated with specific data concerning the time codes and location of the related video segment, obtainable from Presentation Time Stamps of the MPEG Transport Stream and from the local device drivers. The attributes of the key frame objects may also concern the features detected in the referred segment, like length, mixing effects and camera operations. In order to be able to show the content at several levels of details, the key frames can be organized in different levels. All these local descriptors are used as attributes in the *key_frames* class, as shown in Table 2. Short audio clips could complete the available data set about a certain video sequence, to allow a more complete understanding of the content which may not result clearly from the icon itself.

5.3 The video database browser

A video browsing system has been implemented by taking into account the above discussed automation aspects. The core of the system is an object-relational data model. It allows user-defined datatypes and development of complex classes, is capable of handling large objects, while maintaining the advantage of a standard query formulation through SQL syntax. The Java programming language has been adopted for the development of the interface to the database and of the client

application. In this way communication is kept at high level between the different agents. When the application requests the objects to the database through method invocation, this is translated by the database agent into a proper SQL query. This also means that the interface of the client applications can be made very intuitive and that the queries will be embedded so that the user does not need to have any notion of SQL, but only click on buttons with his remote control.

The graphical user interface of the video browser implemented so far, consists of two main parts: the first, which we call Video Program Guide, deals with the selection of the program to be viewed based on textual information, the second, the Video Navigator, displays the visual abstract of a single program for navigation.

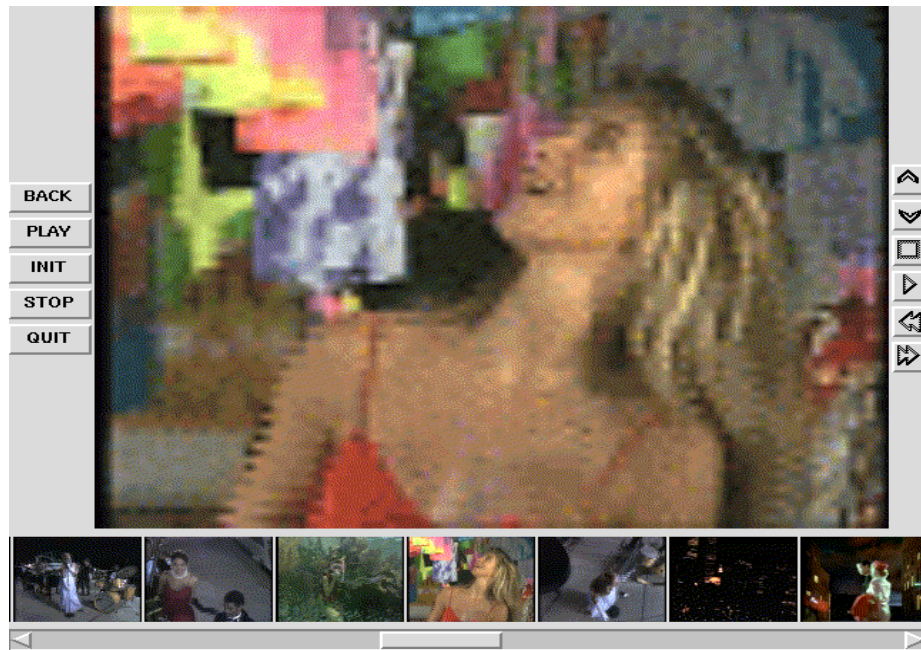


Figure 6: *The Graphical User Interface for the Video Navigator*

The Video Program Guide exploits the extracted global meta-data to show relevant textual information for enabling fast selection of the program to be viewed. All information about the stored video real_time streams is indexed in the *video_program* class. There are 11 subclasses defining the *category* of program (Movie, News, Show, Sports) which are then subdivided in other more than hundred classes defining the *genre*: for example in the Movie class, there are subclasses like drama, thriller, adventure, comedy, etc. This information, extracted from the DVB-SI tables as previously explained, is available at recording time and can be used to adjust the parameters for the extraction and indexing of key frames, as shown in Chapter 3. The extraction process will use some pre-set model, which is defined for every class. For example, the average extraction rate for a news footage can be higher than the one used for a talk show or a commercial. These models can also be adapted according to the user profile, depending on the frequency of use of such tool for the different classes of programs.

The presentation scheme may be adapted as well: the attributes of an instance to be displayed are selected according to the requested class. For instance, in the class *news*, time and date of recording, channel and subtitles are surely more relevant attributes than for a movie, where title, genre, actors are more pertinent. A simple interface for retrieval may also be implemented so that simple queries can be performed without explicitly requiring SQL statements. It is worth noting that as in our database system values are non atomic, each descriptor may contain an array of data types, e.g. more actor names can be contained and searched for in a single "actor" field.

In the Video Navigator, key frames are temporally ordered and displayed in the lower end of the screen. They can flow forward or backward on the screen depending on the selected command button. The presently active icon is enlarged and shown on a larger window for easy visualization from a distance on the TV set. Several levels of granularity can be selected by ascending or descending the organized layers. In a pseudo fast-forward mode, the icons can also be shown sequentially at

a fixed rate on the large screen, while playing the related audio clips. Subtitles are shown in a separate window, in case they are provided. Finally, the selected program can be viewed starting from the selected point by playing it at full screen on TV via a decoder.

6 Conclusions

In this paper we introduced novel approaches for performing some important preparatory steps in the process of making an abstract for stored video data. In the proposed video parsing method, good results are obtained using the statistical model for generalizing the performance of the shot change detection procedure. Our intention in further research is to make the model be more sophisticated and extend it for the detection of gradual transitions. Results of the proposed key frame assignment and key frame distribution methods show that automated key frame extraction is indeed feasible. Since the main issue in our approach is the way of measuring the shot content, further research in this direction is planned. A complete system for automated video sequence abstracting and program filtering has been proposed, and a database system for automatic indexing and browsing of video content, also using audio and textual information, has been implemented. In addition, new techniques for automatic collection and indexing of high level descriptors of video programs, exploiting new trends in digital services, have been introduced. Future work includes implementation of a query system for content based retrieval and refinement of the presentation system by using clustering techniques.

7 References

- [1] *COMPUTER* - IEEE Computer Magazine, Vol. 29, Issue 5, May 1996.
- [2] Yeo, B., Liu, B.: "Rapid Scene Analysis on Compressed Video", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.5, No.6, December 1995.
- [3] Furht, B., Smoliar, S.W., Zhang, H.: "Video and Image Processing in Multimedia Systems", Kluwer Academic Publishers, 1995
- [4] Ahanger, G., Little, T.D.C.: "A survey of Technologies for Parsing and Indexing Digital Video", *Journal of Visual Communications and Image Representations*, vol.7., No. 1, pp. 28-43,1996
- [5] Hanjalic, A., Lagendijk, R.L., Biemond J.: "A New Key frame Allocation Method for Representing Stored Video-Streams", *Proceedings of the First International Workshop on Image Databases and Multi-Media Search*, Amsterdam, 1996
- [6] Lagendijk, R.L., Hanjalic, A., Ceccarelli, M.P., Soletic, M., Persoon, E.: "Visual Search in a SMASH System", *Proceedings of ICIP '96*, Lausanne 1996
- [7] Zhang, H., Kankanhalli, A., Smoliar, S.W.: "Automatic partitioning of full-motion video", *Multimedia Systems*, Vol.1, pp 10-28, 1993
- [8] Sethi, I.K., Patel, N.: "A Statistical Approach to Scene Change Detection", in *Proceedings of SPIE*, Vol. 2420, pp.329-337, 1995
- [9] Yeung, M.M., Liu, B.: "Efficient Matching and Clustering of Video Shots", *Proc. of ICIP 1995*, vol. 1, pp. 338-241, Washington DC, USA, 1995.
- [10] Arman, F., Hsu, A., Chiu, M.-Y.: "Image Processing on Compressed Data for Large Video Databases", *Proc. ACM Multimedia '93*, Anaheim, CA, 1993
- [11] 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.
- [12] Ceccarelli, M.P., Hanjalic, A., Lagendijk, R.L.: "A Sequence Analysis System for Video Databases", to appear in *Proceedings of 5th International Workshop on Time-varying Image Processing and Moving Object Recognition*, Florence, 1996
- [13] Hampapur, A., Jain, R., Weymouth, T.E.: "Indexing in Video Databases" *SPIE* vol. 2420, Feb 1995, pag. 292-306
- [14] Ahanger, G., Benson, D., Little, T.D.C.: "Video query formulation" *SPIE* vol. 2420, Feb 1995, pp. 280-291
- [15] ETS 300 421, "Digital broadcasting systems for television, sound and data services; framing structure, channel coding and modulation for 11/12 Ghz satellite services", EBU/ETSI JTC, December 1994.
- [16] ISO/IEC 13818-1 "Information Technology - Generic coding of moving pictures and associated audio information: Systems" ISO/IEC JTC 1/SC 29, April 1995
- [17] ETS 300 468, "Digital broadcasting systems for television, sound and data services; specification for Service Information (SI) in Digital Video Broadcasting (DVB) systems", EBU/ETSI JTC, January 1996
- [18] Davic 1.1 Specifications - Digital Audio Video Council, Sept. 1996

