Combining MPEG tools to generate video summaries adapted to the terminal and network

Luis Herranz[1] and José María Martínez[2]

[1] *Key Laboratory of Intelligent Information Processing of the Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China*
[2] *Escuela Politécnica Superior, Universidad Autónoma de Madrid, 28049 Madrid, Spain*
Corresponding author: luis.herranz@vipl.ict.ac.cn

MPEG standards provide tools for a broad range of purposes, covering from coding to metadata description tools. In this paper, the combined use of tools from different MPEG standards is described in the context of a video summarization application. The main objective of the framework is the efficient generation of summaries, integrated with their adaptation to the user's terminal and network. The MPEG-4 Scalable Video Coding specification is used for fast adaptation and summary bitstream generation. MPEG-21 Digital Item Adaptation tools are used to describe metadata related to the user's terminal and network. MPEG-7 tools are used to describe the summary. Finally, the framework is compared with alternative approaches (variations and transcoding), in terms of efficiency, rate-distortion performance and other aspects.

Keywords: MPEG-4 Advanced Video Coding; Scalable Video Coding; Video summarization; Storyboard; Video skim; Video adaptation; Scalable video coding; Bitstream extraction; MPEG-21 Digital Item Adaptation; MPEG-7 Multimedia Description Schemes

1. INTRODUCTION

Due to the huge amount of content available in multimedia repositories, abstractions are essential for efficient access and navigation[1, 2]. Video summarization includes a number of techniques exploiting the temporal redundancy of video frames in terms of content understanding. Besides, in modern multimedia systems, there are many possible ways to search, browse, retrieve and access multimedia content, through different networks and using a wide variety of heterogeneous terminals. The content is often adapted to the specific requirements of the usage environment (e.g. terminal and network), performing adaptation operations such as spatial downsampling or bitrate reduction in order to accommodate the bitstream to the available screen size or network bandwidth. This adaptation is often addressed using technologies such as transcoding or scalable coding.

This paper describes a framework that uses scalable video coding for the generation of the bitstreams of summaries, which are also adapted to the usage environment. A summarization-adaptation method is proposed in [3, 4]. In this paper we extended that framework to support additional coding structures and to use other metadata tools from different MPEG standards. The main advantage is the simplicity and efficiency of the process. Extraction of audio and its synchronization with the video stream is also discussed, as well as rate-distortion performance. Finally, a comparison with other alternative architectures is also presented.

The paper is organized as follows. Section 2 briefly introduces related technologies and works. Section 3 overviews the use of MPEG standards in the proposed framework. Section 4 describes the summarization framework using MPEG-4 AVC. Section 5 shows how MPEG-7 can be used to describe summaries. In Section 6, the framework is extended to include adaptation using MPEG-4 SVC. The inclusion of audio in the framework is discussed in Section 7. Experimental results are presented in Section 8 while Section 9 provides a qualitative comparison of the different approaches studied in the paper. Finally, Section 10 concludes the paper.

2. RELATED WORK

2.1. Video summarization

Video summarization techniques provide the user with a compact but informative representation of the sequence, usually in the form of a set of key images or short video sequences[5, 6, 7, 8]. In general, a summarized sequence is built from the source sequence selecting

frames according to some kind of semantic analysis of the content. Many algorithms have been proposed for keyframe selection and video summarization, using different criteria and abstraction levels. Recent surveys[9, 10, 11] provide comprehensive classifications and reviews of summarization techniques.

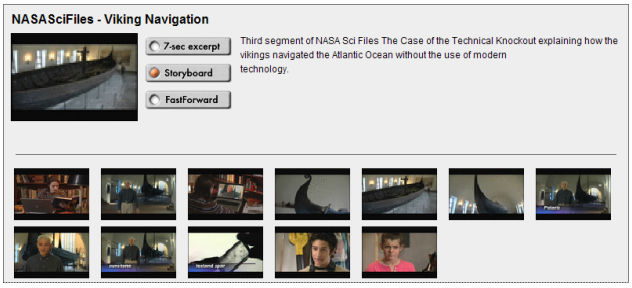


FIGURE 1. Example of summary (storyboard) in a digital library (Open Video project).

Important examples of systems using video abstractions or summaries are digital video libraries, such as YouTube[3], the Internet Archive[4] or the OpenVideo project[5][12]. Search and browsing are much easier and efficient using abstracts than browsing actual video sequences. Usually, a single key image, the title and a short description are used to represent a specific piece of content. However, other modalities of visual abstractions have been proposed, in order to include more (audio)visual information. A widely used representation is the image storyboard, which abstracts content into a set of key images that are presented simultaneously. Figure 1 shows an example of the web interface of the Open Video project. It depicts a storyboard summary in addition to the conventional textual description of the sequence.

However, when dealing with video content, often it is more useful and meaningful to present the summary as a short video sequence, instead of independent frames. Segments provide information about the temporal evolution of the sequence, that isolated images cannot provide. This representation is often known as video skim, composed of significant segments extracted from the source sequence. Several approaches have been used in video skimming, including visual attention[13], image and audio analysis[14, 15] and high level semantics[2].

Between selecting single frames and selecting segments, there is still the possibility of selecting a variable amount of frames per segment. Fast forwarding the sequence at a constant rate can provide the user with a preview of the content (see Figure 2), useful to browse it in a shorter time[16]. However, there are often less important parts that can be sped up, while more significant parts can be played at normal rate. Thus, a content-based fast forward can be obtained if the skimming of frames is performed in a frame basis guided by a semantic clue. Motion activity and camera motion have been also used as clues to drive the selection of frames[17, 18]. A related technique is frame dropping driven by some low level features, where less important frames are discarded during the transmission of the sequence in case of network congestion. [19] and [20] use MPEG-7 intensity motion descriptor and perceived motion energy, respectively, as features to guide frame dropping.

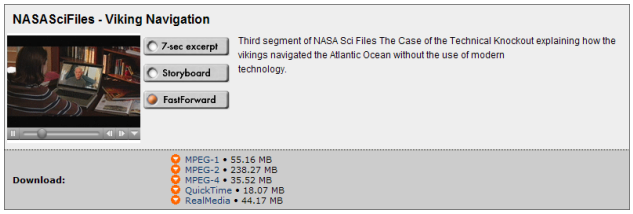


FIGURE 2. Fast forward summary and adapted versions of the content (Open Video project).

Besides these widely extended representations, there is an increasing interest in the development of more intuitive abstractions. In this direction, comics and posters have inspired several works[1, 21, 22] where the key images are presented with variable size in a layout where temporal order is replaced by a spatial scan order. Edited video is structured into more abstract units such as shots and then scenes, which typically contain several related shots. This hierarchical structure of videos (e.g. chapters, scenes, shots) can be also exploited for summarization[23] and browsing[7, 6].

In order to obtain better results, domain knowledge has been included in the methods. Thus, sports video summarization tries to exploit prior knowledge, such as the structure and characteristics of a specific sport game for better results[24, 25, 26, 27, 28]. Usually, these approaches are based on the detection of some important events that must be included in the summary (e.g. goals, replays, end of game). Other typical scenarios are news[29, 30, 31, 32, 11], which is a highly structured genre, surveillance[30] and home videos[33]. Additionally, metadata can be provided for higher level understanding of the content[34].

Recently, an intense research in rushes summarization was motivated by the TRECVid rushes summarization task[35, 36]. This content is significantly different to other video sources, as rushes are unedited footage containing retakes and they are much more redundant compared to other sources. This content also contains undesirable junk segments with blank frames, clapboards, etc. Participants in the TRECVid rushes summarization task developed systems designed to summarize this specific content[35, 36, 37].

[3] http://www.youtube.com
[4] http://www.archive.org
[5] http://www.open-video.org

2.2. Video adaptation

Content adaptation[38] is a main requirement to effectively bring the content from service providers to the actual users, using different terminals and networks and enabling the so called Universal Multimedia Access (UMA)[39, 40]. Especially important is the case of hand-held devices, such as tablets, Personal Digital Assistants (PDAs) and mobile phones, where other issues such as limited computational resources and low power consumption requirements become very important.

In contrast to content-blind adaptation (e.g. resolution downsampling, bitrate adaptation), which does not consider the content itself, content-based adaptation takes advantage of a certain knowledge of what is happening (semantics) in the content to perform a better adaptation. In [38], video summarization is considered a special type of structural adaptation, in which the summary is an adapted version of the original content. Content-based adaptation is often known as semantic adaptation, which also includes personalization[41, 29] and object based adaptation[42, 43]. The knowledge about the content that semantic adaptation needs can be extracted automatically or provided as metadata[44, 45] from previous automatic analysis or manual annotation. This knowledge ranges from very low level (shot changes, color and motion features, etc.) to high level (events, objects, actions, etc.).

2.2.1. Variations and transcoding

A first solution to the adaptation problem is the use of (offline) variations[46, 47], covering a number of predefined versions of the content which are generated and stored prior to being consumed by users. Figure 2 shows an example of adapted versions available as offline variations (e.g. MPEG-1, MPEG-2, MPEG-4). The user then can decide which version is most suitable according to codec capabilities, display resolution, network or storage capacity.

An alternative to variations is the adaptation of the bitstream on demand. The generation of the adapted bitstream requires decoding, adaptation to the target usage environment and encoding the adapted sequence. This approach to adaptation is known as transcoding[48, 49] and it can be computationally very demanding. However, the information in the coded bitstream (e.g. motion vectors, coding modes) could be still exploited for faster analysis. Thus, alternative transcoding architectures have been proposed[49, 50, 51] with reduced complexities, but at the cost of a worse rate-distortion performance.

2.2.2. Scalable Video Coding

Scalable video coding tackles the problem of adaptation at the encoding stage, in a way that simplifies the adaptation process. A scalable video stream contains embedded versions of the source content which can be decoded at different resolutions, frame rates and qualities, simply selecting the required parts of the bitstream. Thus, scalable video coding enables simple, fast and flexible adaptation to a variety of heterogeneous terminals and networks. The numerous advantages of this coding paradigm have motivated an intense research activity in the last years[52, 53, 54].

Recently, the Joint Video Team (JVT) has standarized a scalable extension of the successful H.264/MPEG-4 AVC[55] standard, supporting multiple scalabilities, notably temporal, spatial and quality scalabilities. This new specification is known as MPEG-4 SVC[54]. In this paper, the term AVC is used to refer to the H.264/MPEG-4 AVC specification and SVC to the scalable extension.

2.2.3. Bitstream modification

As video is usually distributed in a compressed format, the coding structure can also be exploited for lightweight customization of the bitstream, directly operating with the compressed data. For example, scalable bitstreams are adapted with minimum processing directly on the compressed bitstream. Bitstream Syntax Description (BSD) tools[56] of MPEG-21 Digital Item Adaptation (DIA)[57, 40] were developed for generic adaptation of coded sequences directly manipulating the bitstream.

In some cases, bitstream modification can be used for other content-based structural adaptations, such as summarization. In that case the summary is created operating with the syntax elements in the bitstream. Specifically, for AVC, in the framework of MPEG-21 DIA, [58] proposes a content-based adaptation system using a shot-based approach, while [20] uses a similar approach for frame dropping based on the perceived motion energy.

3. A SUMMARIZATION FRAMEWORK USING MPEG STANDARDS

Over the last years, MPEG specifications have tackled different aspects and requirements of multimedia systems, initially focusing on efficient and flexible coding of audio and video. Later, MPEG specifications broadened to include not only standarized descriptions of bitstreams but also standarized descriptions of the content itself and other elements and agents involved in multimedia systems. Several of these MPEG tools are combined in the application described in this paper to provide standard syntax for bitstreams, content and usage context (see Figure 3).

In the proposed framework, summaries are stored as metadata along with the bitstream. They are described following the MPEG-7 Multimedia Description Schemes (MDS) specification[59], which provides metadata tools for summarization.

A key for the success of video applications is the coding format used to compressed the huge amount

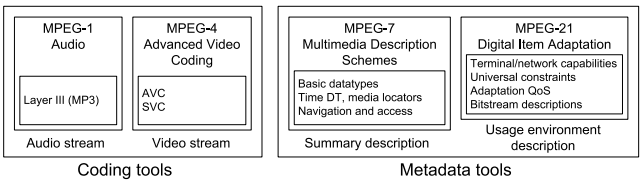


FIGURE 3. Use of MPEG standards in the framework.

of data into bitstreams that can be handled by telecommunication networks. MPEG standards do not specify the encoder nor the decoder themselves (apart from a reference decoder and implementation), only the syntax of the coded bitstream. In our case, the coding format is MPEG-4 AVC[60, 61] for nonscalable bitstreams, extended to MPEG-4 SVC[54] for scalable bitstreams.

Similarly, audio is also encoded using an MPEG coding format. We use MPEG-1 layer III (also known as MP3)[62, 63, 64], which is still widely used. Other formats such as MPEG-2 Advanced Audio Coding (AAC)[65, 66] and MPEG-4 AAC[67, 68] can be also used instead.

Finally, the MPEG-21 standard specifies a number of tools and concepts in a standard framework to enable advanced multimedia applications in heterogeneous usage environments. Particularly, MPEG-21 DIA[57, 40] tackles the adaptation for universal access, with metadata tools to describe the usage environment, including terminal capabilities, network and user characteristics.

4. GENERATION OF SUMMARIES USING MPEG-4 AVC

In general, the term video summarization is used to refer to a number of techniques that analyze the semantics of the source sequence and then create a summary according to this analysis. For convenience, we separate the whole summarization process into two stages: analysis of the input bitstream and generation of the summary bitstream. Actually, analysis is completely detached from generation and it can be performed in a previous stage and stored as metadata. Analysis consists of either manual annotation or an automatic summarization algorithm.

A summary is usually generated by the concatenation of frames. Hence, the basic unit for summarization is the frame. In the case of uncompressed video (e.g. in YUV format), it is possible to select each frame independently and build a new sequence just concatenating the values of the samples of each selected frame. We can obtain the different summaries as variations or using online adaptation. As variations are obtained also by transcoding, in this section we study the architectures of a transcoder and a bitstream extractor, applied to the problem of generating a video summary. For the case of bitstream extraction we introduce a more convenient representation of summaries in which the results of the analysis stage are referred to coding units rather than to independent frames.

4.1. Architecture based on transcoding

Transcoding is frequently used in (non content-based) bitstream adaptation to constrained bitrate conditions. Figure 4a shows a summarization architecture using a transcoder for the generation stage. A transcoder can be easily obtained from the cascade of a decoder and an encoder. This is the approach used in most of video summarization systems, because of its straightforward implementation from general-purpose decoders and encoders. Besides, it is simple to separate the analysis from the generation, as the analysis stage usually creates a summary description referred to uncompressed frames as basic units.

In order to emphasize the difference between transcoding and extraction, we first introduce a simple model to describe summaries and to guide a transcoding-based generation stage. Using a transcoder in the generation stage, the result of the analysis stage is a description of the summary in terms of the input frames. For a sequence with N frames, any summary can be described with the following binary function:

$$include\,(n) = \begin{cases} 1 & n \in summary \\ 0 & otherwise \end{cases} \quad n = 0, 1, \ldots, N-1$$

The transcoder decodes all the frames and for each of them decides either to include or discard it according to $include\,(n)$. Each frame is independent of other frames, so the resulting sequence of frames is always valid and it can be encoded using a suitable format, even different from the input format.

The transcoder shown in Figure 4a has an architecture with all the stages of a conventional decoder (entropy decoding, dequantization, inverse transform and motion compensation) and a conventional closed-loop encoder (motion estimation and compensation, transform, quantization and entropy coding). The link between them is the frame selector, which is also the entry point for summarization data.

Summary generation using transcoding is very inefficient, specially for long summaries, such as video skims. The complexity of the coding format also influences the complexity of both encoding and decoding (e.g. AVC is usually much more complex than MPEG-2). The most demanding part of the whole process is motion estimation. However, limited search ranges or simplified search algorithms, used to speed up encoding, lead to a degradation of the rate-distortion performance. Additionally, many transcoding architectures have been proposed in this context[49, 50, 51], trading off rate-distortion performance and efficiency.

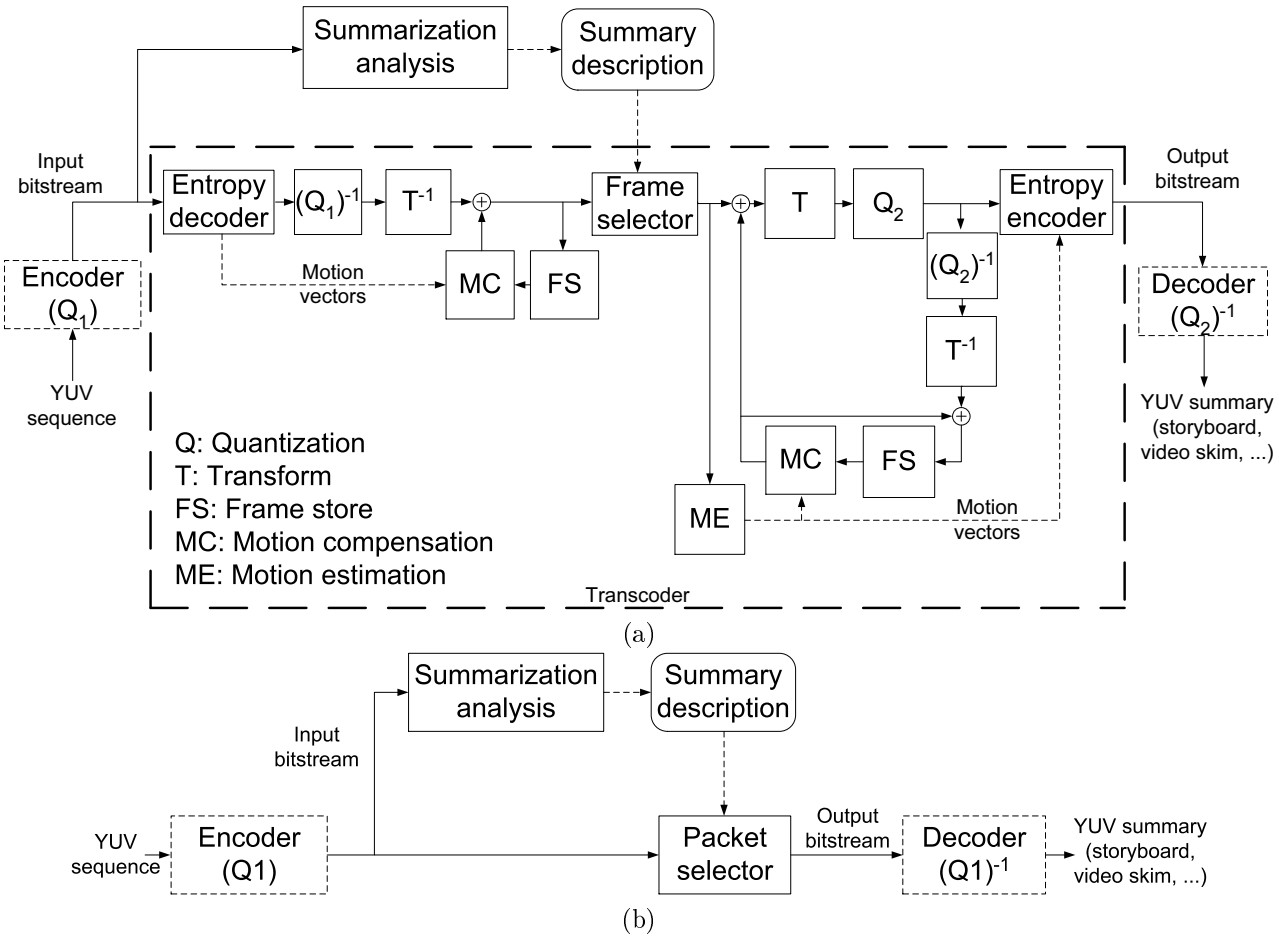


FIGURE 4. Summarization architectures: a) transcoder (decoder-encoder cascade), b) extractor.

Besides inefficiency, transcoding suffers from an inherent drawback related to the additional quantization (Q_2) introduced by the transcoder. A first loss of information occurred before the transcoding, when the input sequence was lossy encoded with a first quantization (Q_1). When comparing transcoding architectures, the decoder-encoder cascade with full range search is the optimal architecture which gives the best end-to-end rate-distortion performance, and it is used as reference in most transcoding comparisons[49, 51, 50].

4.2. Coding units and summarization units

Prior to the introduction of the extraction architecture, we briefly introduce some details about AVC. The standard specifies two conceptual layers: a Video Coding Layer (VCL), which deals with the efficient representation of the video content, and a Network Abstraction Layer (NAL), which deals with the format and header information in a suitable manner to be used by a variety of network environments and storage media. The bitstream is composed of a succession of NAL units, each of them containing payload and a header with several syntax elements. An access unit (AU) is a set of consecutive NAL units which results in exactly one decoded picture. AVC also enables the possibility of specifying much more flexible prediction structures, increasing the compression performance. In contrast to prior standards, in AVC the coding order and the presentation are completely decoupled, and any frame can be marked as reference and used for prediction of subsequent frames.

For simplicity, we will consider that each frame is coded into one slice and one NAL unit, and it corresponds to a single AU. However, concatenating the NAL units of the frames belonging to the summary will probably generate a non-decodable bitstream, as most of them are encoded predictively with respect to previous frames in the source bitstream. If these reference frames are removed from the output bitstream, predicted frames will not be decodable. For this reason it is more convenient to refer the results of the analysis to coding oriented structures rather than to individual frames, taking into acount the prediction dependencies between them.

If we consider a sequence with N frames, coded with T temporal decompositions (that means $T+1$ temporal levels), then the frame index can be notated as $n \in \{0, 1, \ldots, N-1\}$ and the temporal level as $t \in \{0, 1, \ldots, T\}$. For each temporal level, a subsampled

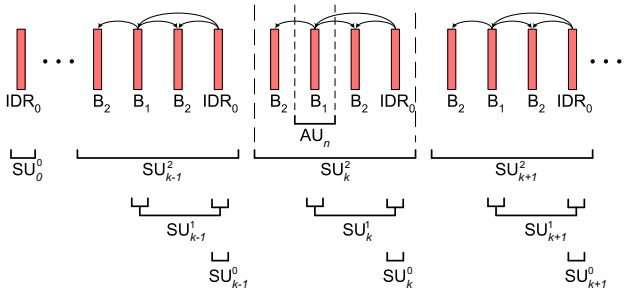


FIGURE 5. Coding structures and summarization units in H.264/AVC.

version in the temporal axis can be decoded, as there are no breaks in the prediction chain. In this case, we use an alternative representation that describes the summary using groups of frames related by prediction rather than frame indices. In this alternative representation, the basic unit for summarization is the Summarization Unit (SU). We define the *summarization unit* as a set of consecutive AUs at certain temporal level related by the prediction structure and that can be decoded independently from the other AUs in the sequence. The sequence is then partitioned into M summarization units. Figure 5 shows an example of a hierarchical coding structure and its corresponding SUs. The SU at the highest temporal level is the one formed by an Instant Decoding Refresh (IDR) frame and three B frames which are coded predictively. Obviously, it is not possible to include a B frame in the summary without including its reference frames, as it would not be decoded by the user's decoder. However, there are more SUs in the bitstream, at different temporal levels, as the one composed by the IDR and B frames at the first temporal level, and the one composed only by the IDR frame. The only requirement for these groups of NAL units is that they must be decodable independently of the rest of the bitstream (except other required non VCL NAL units such as parameter sets) and that their decoding results exactly in a set of consecutive frames at a certain temporal level.

Besides the concept of SUs, we also define the summarization constraint $tlevel\,(m)$ as the maximum temporal level for each summarization unit SU_m. This function describes how to generate the summaries, as the indices of the frames do in the case of uncompressed bitstreams. If the value of $tlevel\,(m)$ is set to -1 for a certain m it means that SU_m is not included in the summary. The objective of the analysis stage of a summarization algorithm in this framework is to determine the summarization constraint for each video sequence, based on a certain content analysis.

The selection based on SUs has the drawback of losing some accuracy in the selection of frames (i.e. not any frame can be arbitrarily selected). This accuracy depends on the length of the SU and it is given by the coding structure (e.g. the SU of Figure 5 has an accuracy of 4 frames).

4.3. Modalities of video summaries

There are different video summarization modalities that can be easily adapted to the proposed scheme. Depending on the values that $tlevel\,(m)$ takes for the SUs, we distinguish the following modalities of video summaries (see Figure 6):

- *Storyboard*: built by selecting a few independent and separated frames to represent the content as a collection of images. Within the proposed model, for convenience, we restrict the potential selected frames to be I frames belonging to the lowest temporal level. We also assume that the lower temporal resolution has only one I frame. There is no noticeable difference for practical applications, and actually most storyboard summarization algorithms use temporal subsampling to speed up analysis. With these assumptions, the storyboard is characterized as follows

$$tlevel\,(m) = \begin{cases} 0 & keyframe\ in\ SU_m \\ -1 & otherwise \end{cases}$$

- *Video skim*: the adapted sequence is shorter than the input sequence, obtained by selecting certain segments of the input sequence. In this case, the valid options for each SU are either not constraining its temporal level or skipping it. Thus, if the maximum temporal level is t_{max} the video skim can be characterized as follows

$$tlevel\,(m) = \begin{cases} t_{max} & SU_m \in skim \\ -1 & otherwise \end{cases}$$

- *Content-based fast forward*: this summarization modality is based on the acceleration and decceleration of the sequence controlled by a certain content-based criteria, in order to visualize it in a shorter time. In this case, the number of frames of each SU is variable depending on the required frame rate at each SU.

4.4. Architecture based on bitstream extraction

If the coding format is the same for both input and output bitstreams, an alternative approach for the generation of the summary bitstream is bitstream extraction (see Figure 4b). Similar to the extraction approach used in scalable bitstream adaptation (guided by usage environment constraints), the extraction for summarization is guided by the summary description. The whole transcoder of Figure 4a is replaced by an extractor which basically consists of a packet selector.

This approach has two inherent advantages. Extraction is a very simple operation which requires few resources and that can be done very efficiently. Besides,

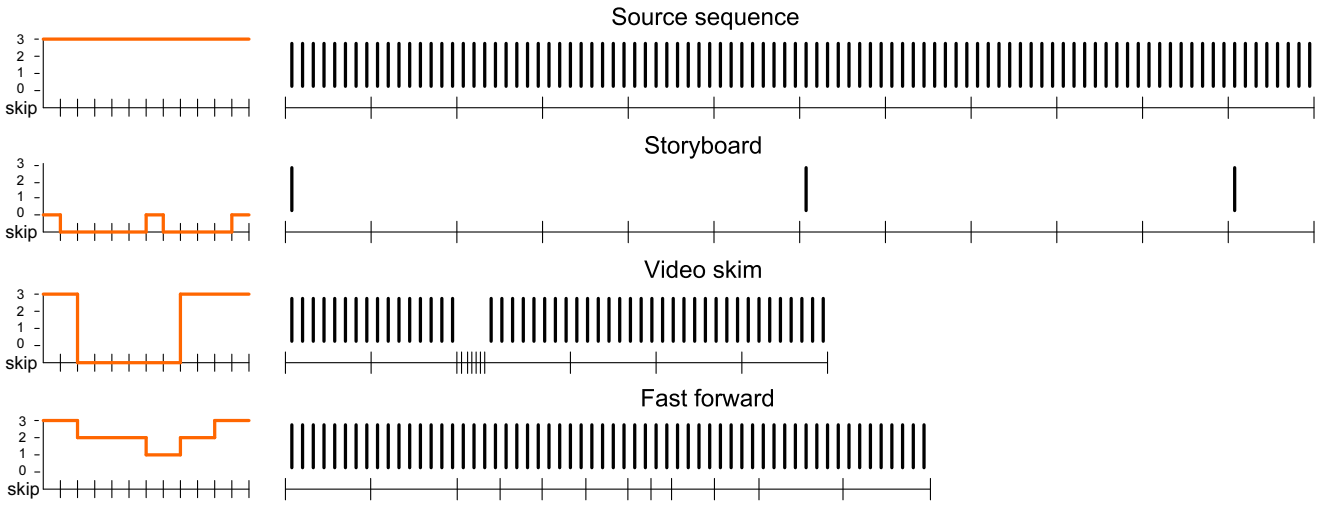


FIGURE 6. Examples of the function $tlevel\,(m)$ (left) and frames selected (right).

the frames themselves are not modified, so the quality of each frame is exactly the same of that in the input bitstream. Particularly, compared to transcoding, there is no quality degradation due to an additional quantization stage. However, the use of extraction is not always possible if the coding structure does not satisfy some requirements, as described in Section 4.2. If extraction is not possible, transcoding is required to generate the summary.

A bitstream extractor selectively copies chunks of bits from the input bitstream to the output bitstream. However, the output bitstream may be non-decodable by a compliant decoder, due to mismatches between the expected and the actual decoding status. It must be emphasized that the original bitstream was encoded using prediction dependencies according to the frames encoded previously. When the summary bitstream is decoded, all the syntax elements are referred to the decoding status of the original bitstream, which is different from the actual decoding status, as some parts were removed. For this reason, the extractor must also ensure that the decoding status is valid, fixing headers as appropriate, so the bitstream can be decoded correctly. In the following, we describe two mechanisms to obtain a decodable bitstream in the context of AVC.

In order to handle the complexity of arbitrary prediction structures, AVC specifies the operation of the decoded picture buffer (DPB), which is a buffer containing previously decoded frames which can be used as references. Frames used as reference are signalled according to the current state of the buffer and the order in which decoded frames are stored in the buffer. The DPB of the decoder must replicate the status of the multiframe buffer of the encoder, according to the memory management control operations (MMCO) included in the bitstream. The indexes of the references signalled in the header are referred to the current status of the DPB, according to the decoding process, which includes and discards frames from the buffer dynamically. In general, discarding some SUs introduces discontinuities in the status of the DPB.

Using an IDR access unit as the first access unit of each SU is the simplest mechanism to obtain a valid bitstream that can be decoded correctly. The IDR access unit flushes the DPB so the decoding of each SU begins with an empty DPB. The numbering of frames for referencing purposes (e.g. syntax element `frame_num`) must be consistent with the new sequence of AUs. With this mechanism, the numbering is also reset at the beginning of each SU, so the sequence can be decoded properly starting from any SU. Thus, the status of the decoder for a given AU is the same for both the source and summary bitstreams. The use of IDR access units in SUs leads to non-overlapped SUs.

The process of extraction in this case is depicted in Figure 7a. It shows an example of IDR based SUs of length 4, with AUs shown in coding order. Firstly, the parameter sets are copied without any modification to the output bitstream. After that, NAL units encoding AUs belonging to the summary are included, while the rest are discarded. In the example, the summary begins with IDR^0_{64}, which is the first AU included. After that AU, the rest of the AUs are included, conforming a valid bitstream. Note that no modification is done to any NAL unit and, thus, the extraction process is extremely simple in this case.

4.5. Coping with discontinuites in the decoder

Although the use of IDR access units is simple and convenient, using I frames enables to have coding references to frames in different SUs (e.g. open Groups of Pictures -GOPs-) with improved coding performance, as more frames can be used for prediction. However, the extraction process needs to take into account the decoding status. In contrast to IDR access units, I

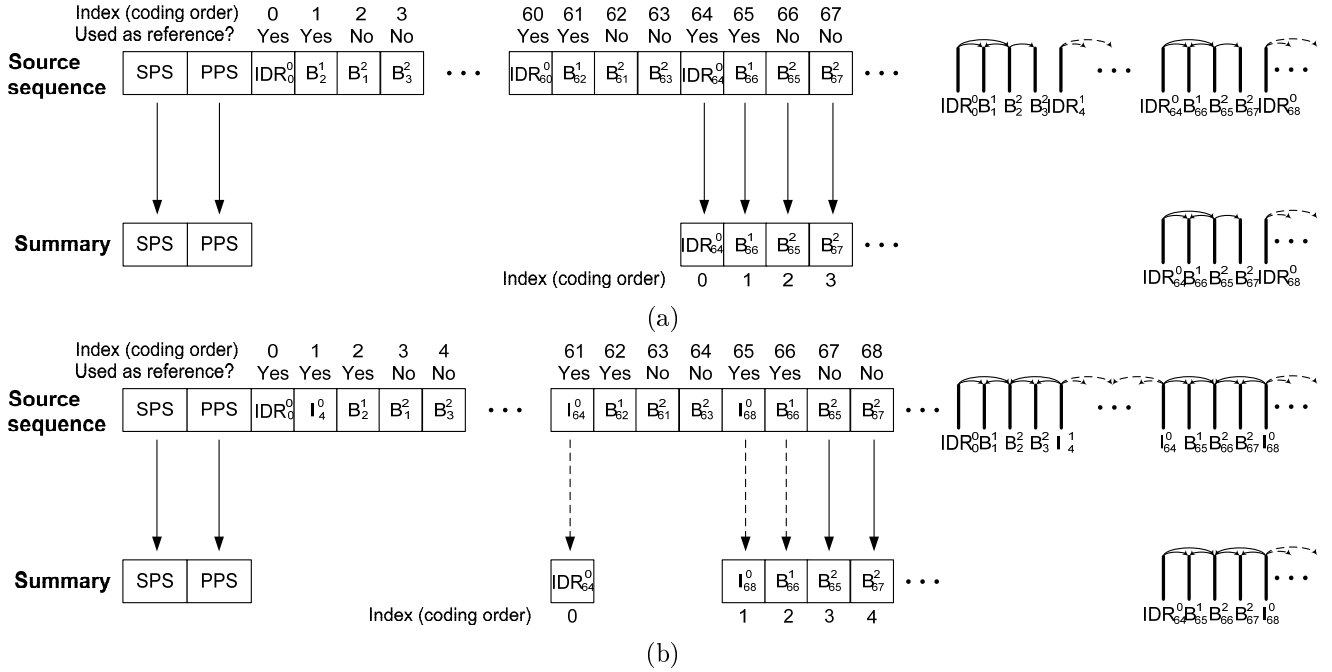


FIGURE 7. Low level extraction process: a) IDR access units (no header update), and b) I access units (header update).

access units do not reset the status of the decoder. The DPB still contains previous frames, some of them marked as reference, and the numbering is not reset. For this reason, I access units do not ensure that the status of the decoder at a given AU will be the same for both the source and summary bitstreams. Therefore, headers must be checked and updated in order to correct the discontinuities in the decoding status due to those AUs removed from the bitstream.

Whenever a gap is found in the decoding process, in order to preserve a valid status of the DPB, the extractor updates the header of NAL units (see Figure 7b for an example of the extraction process and Figure 8 for header modifications) in two cases:

- *Transformation of I access units to IDR access units.* In AVC, the first AU must be an IDR access unit. If the first AU is removed from the bitstream, it will not be compliant anymore. Therefore, the very first I access unit of the summary must be converted to an IDR access unit, which implies a major update of the header, changing the values of `nal_reference_idc`, `nal_unit_type`, and removing some syntax elements while including others (see Figure 8). The value of `frame_num` may be also required to be updated, according to the numbering mechanism specified in the standard[60]. Apart from the first frame, whenever a new gap appears, the first AU after the gap must be also converted to an IDR access unit.
- *Update of MMCO commands.* MMCO commands control how the references are managed in the DPB. These commands are signalled in the header and are referred to the current status of the buffer. A reference frame in the multiframe buffer is identified by the value of $picNumX$, which is derived differentially from the current frame (picture) number as[60]:

$$picNumX = CurrPicNum - (difference\_of\_pic\_nums\_minus1 + 1) \quad (1)$$

where $difference\_of\_pic\_nums\_minus1$ is the value of the syntax element `difference_of_pic_nums_minus1` of the MMCO command and $CurrPicNum$ is the value of the syntax element `frame_num` of the current frame. Both MMCO commands and frame numbers must be fixed, updating the value of `memory_management_control_operation` and also the value of `difference_of_pic_nums_minus1`.

In the example shown in Figure 7b, the SU consists of a four frame dyadic coding unit $I^0B^2B^1B^2(I^0)$. Both I^0 and $B^1$are used as references, while the two B^2 are not. According to the results of the analysis stage, the first frame of the summary (a video skim in this example) is I^0_{64}. As the SU is overlapped, it requires also I^0_{68} to decode the rest of the frames, so these two frames must be included first. As shown in Figure 8, the header of I^0_{64} is modified to convert the frame to an IDR access unit, IDR^0_{64}. As the frame B^1_{62}, used as reference in the source bitstream, is not included in the summary bitstream, the value of `frame_num` of IDR^0_{64} must be updated from 31 to 32 to preserve the continuity of the numbering of references. The MMCO command is also removed from I^0_{68}, as it is

no longer required because the frame it is referred to is no longer in the buffer. The header of B_{66}^{1} is also updated to obtain the correct $picNumX$ according to (1) (`difference_of_pic_nums_minus1` is changed from 2 to 1). The status of the frames used as references in the multiframe buffer for both the source and summary bitstreams are shown in Table 1, along with the MMCO commands used. Note the differences between the status of the decoder for each AU in both cases. While containing the same visual data, headers must be referred to each particular decoding status.

The resulting bitstream is correctly decoded by the JM reference decoder[69]. However, headers may be further modified if required, for example, to number summary frames starting from 0, or to change dynamically the frame rate.

5. DESCRIPTION OF SUMMARIES IN MPEG-7

In contrast to previous MPEG standards, MPEG-7 focuses on the description of multimedia content, from low level features to high level concepts, providing description tools with standarized syntax and semantics. It has been designed as a generic standard, that can be used in a broad range of applications. The MPEG-7 Part 5 covers the MPEG Multimedia Description Schemes (MDS)[59], which deals with generic and multimedia entities.

5.1. MPEG-7 summarization tools

The description tools in MPEG-7 MDS are grouped into six areas. One of these areas is *navigation and access*, including tools for summarization. These tools can be used to specify summaries of time-varying audiovisual data that support hierarchical and sequential navigations. The former uses the *HierarchicalSummary* description scheme to specify summaries used in hierarchical navigation, with several related summaries including different levels of detail. Each level can support also sequential navigation. The *SequentialSummary* description scheme specifies summaries of data that support sequential navigation. Examples of such summaries are content-based fast forward and video slideshows.

A *SequentialSummary* consists of a list of elements describing the video, audio and textual components of the summary. These elements can be synchronized using *SyncTime* elements. Each component of the summary is specified by a *VisualSummaryComponent*, *AudioSummaryComponent* or *TextualSummaryComponent* with the location of a particular frame, video segment, audio clip or textual annotation.

5.2. Examples of descriptions

In our framework, we use storyboards, content-based fast forwards and video skims, which can be described as sequential summaries. As we explained before, the summaries are described using the function $tlevel\,(m)$, which is referred to SUs. This information must be converted to a suitable representation for MPEG-7 description tools, specifying which frames must be included rather than the temporal level of a SU. Therefore, when a description is read to generate the summary, it must be converted back to $tlevel\,(m)$ in order to be used by the summarization framework.

Figure 9 shows an example of storyboard description in MPEG-7. A *SourceLocator* element specifies the source video, and several *ImageLocator* elements specify the frames selected for the storyboard. The reference of the frames in the description is relative to the source video.

```
<SequentialSummary id="StoryboardSummary"
    components="visual">
   <SourceLocator><!--Location of the source
        content -->
      <MediaUri>file://video.264</MediaUri>
   </SourceLocator>
   <VisualSummaryComponent>
      <ImageLocator><!--Locates summary
          keyframe in the original video -->
         <MediaRelIncrTimePoint
             mediaTimeUnit="PT1N25F"
             mediaTimeBase="../../../
             SourceLocator[1]">801</
             MediaRelIncrTimePoint>
      </ImageLocator>
   </VisualSummaryComponent>
   <VisualSummaryComponent>
      <ImageLocator>
         <MediaRelIncrTimePoint
             mediaTimeUnit="PT1N25F"
             mediaTimeBase="../../../
             SourceLocator[1]">961</
             MediaRelIncrTimePoint>
      </ImageLocator>
   </VisualSummaryComponent>
   ...
</SequentialSummary>
```

FIGURE 9. Storyboard description in MPEG-7.

Similarly, video skims can be easily described using the *SequentialSummary* tool, as shown in Figure 10, where several *VideoSourceLocator* elements specify the start time and the duration of each video segment.

6. INTEGRATED SUMMARIZATION AND ADAPTATION FRAMEWORK IN MPEG-4 SVC

We described a model and a framework to represent and efficiently generate video summaries, but without considering adaptation to the usage environment. Most of video summarization techniques can be formulated as a special case of video adaptation, where the adaptation

```
NALU len 19, nal_ref_idc 3, nal_unit_type 7 (SPS)

NALU len 4, nal_ref_idc 3, nal_unit_type 8 (PPS)

~~NALU len 1024, nal_reference_idc 3, nal_unit_type 5 (IDR)~~

~~NALU len 1189, nal_reference_idc 2, nal_unit_type 1 (I)~~

...............

NALU len 29985, nal_ref_idc 2, nal_unit_type 1 (I)
  first_mb_in_slice                          0
  slice_type                                 7
  pic_parameter_set_id                       0
  frame_num                                 31

  pic_order_cnt_lsb                        128


  ~~adaptive_ref_pic_buffering_flag~~          ~~1~~
  ~~memory_management_control_operation~~      ~~1~~
  ~~difference_of_pic_nums_minus1~~            ~~0~~
  ~~memory_management_control_operation~~      ~~0~~
  slice_qp_delta 2

~~NALU len 7429, nal_ref_idc 2, nal_unit_type 1 (B)~~

~~NALU len 6608, nal_ref_idc 0, nal_unit_type 1 (B)~~

~~NALU len 7607, nal_ref_idc 0, nal_unit_type 1 (B)~~

NALU len 28168, nal_ref_idc 2, nal_unit_type 1 (I)
  first_mb_in_slice                          0
  slice_type                                 7
  pic_parameter_set_id                       0
  frame_num                                 33
  pic_order_cnt_lsb                        136
  adaptive_ref_pic_buffering_flag            1
  ~~memory_management_control_operation~~      ~~1~~
  ~~difference_of_pic_nums_minus1~~            ~~0~~
  ~~memory_management_control_operation~~      ~~0~~
  slice_qp_delta                             2

NALU len 9238, nal_ref_idc 2, nal_unit_type 1 (B)
  first_mb_in_slice                          0
  slice_type                                 6
  pic_parameter_set_id                       0
  frame_num                                 34
  pic_order_cnt_lsb                        132
  direct_spatial_mv_pred_flag                1
  num_ref_idx_override_flag                  1
  num_ref_idx_l0_active_minus1               0
  num_ref_idx_l1_active_minus1               0
  ref_pic_list_reordering_flag_l0            0
  ref_pic_list_reordering_flag_l1            0
  adaptive_ref_pic_buffering_flag            1
  memory_management_control_operation        1
  difference_of_pic_nums_minus1              2
  memory_management_control_operation        0
  slice_qp_delta                             3

NALU len 10217, nal_ref_idc 0, nal_unit_type 1 (B)

NALU len 4691, nal_ref_idc 0, nal_unit_type 1 (B)

NALU len 28697, nal_ref_idc 2, nal_unit_type 1 (I)
...............
```

```
NALU len 19, nal_ref_idc 3, nal_unit_type 7 (SPS)

NALU len 4, nal_ref_idc 3, nal_unit_type 8 (PPS)

NALU len 29984, nal_reference_idc 3, nal_unit_type 5 (IDR)
  first_mb_in_slice                          0
  slice_type                                 7
  pic_parameter_set_id                       0
  frame_num                                 32
  idr_pic_id                                 0
  pic_order_cnt_lsb                        128
  no_output_of_prior_pics_flag               1
  long_term_reference_flag                   0

  slice_qp_delta                             2

NALU len 28169, nal_ref_idc 2, nal_unit_type 1 (I)
  first_mb_in_slice                          0
  slice_type                                 7
  pic_parameter_set_id                       0
  frame_num                                 33
  pic_order_cnt_lsb                        136
  adaptive_ref_pic_buffering_flag            0

  slice_qp_delta                             2

NALU len 9239, nal_ref_idc 2, nal_unit_type 1 (B)
  first_mb_in_slice                          0
  slice_type                                 6
  pic_parameter_set_id                       0
  frame_num                                 34
  pic_order_cnt_lsb                        132
  direct_spatial_mv_pred_flag                1
  num_ref_idx_override_flag                  1
  num_ref_idx_l0_active_minus1               0
  num_ref_idx_l1_active_minus1               0
  ref_pic_list_reordering_flag_l0            0
  ref_pic_list_reordering_flag_l1            0
  adaptive_ref_pic_buffering_flag            1
  memory_management_control_operation        1
  difference_of_pic_nums_minus1              1
  memory_management_control_operation        0
  slice_qp_delta                             3

NALU len 10217, nal_ref_idc 0, nal_unit_type 1 (B)

NALU len 4691, nal_ref_idc 0, nal_unit_type 1 (B)

NALU len 28697, nal_ref_idc 2, nal_unit_type 1 (I)
...............
```

FIGURE 8. Example of modification of headers in the extraction process. Left: original bitstream, right: summary bitstream

is performed in the temporal axis, and the adapted version is composed by the selection and concatenation of frames from the input sequence. For this reason it is very convenient to describe the summarization process using tools similar to those used for video adaptation.

A transcoder can be easily adapted to generate a lower resolution, lower bitrate and lower frame rate version of the video or summary, according to the requirements of the usage environment. Even a different coding format could be used. Similarly, different variations, adapted to different predefined usage environments, can be generated and stored.

We are also interested in studying how scalable video can be combined with summarization. The advantage of SVC relies on its efficient adaptation scheme. With SVC the adaptation engine is also a bitstream extractor which modifies the bitstream selecting only the parts required according to some constraints (see Figure 11). The constraints (resolution, bitrate, etc.) are imposed by the usage environment. The extractor selects the appropriate layers of the input bitstream satisfying the constraints. The output bitstream is also conforming with the SVC standard so it can be decoded with a suitable SVC decoder.

6.1. MPEG-21 tools for usage environment description

Prior to the adaptation, the specific conditions of the usage environment must be known, such as the terminal and network used to access the content. The MPEG-21[41, 70, 71] standard aims at developing a normative open framework for multimedia delivery and consumption, based on the concepts of Digital Item (DI) as the basic unit of transaction, and Users as entities that interact with DIs. The objective is to enable a transparent and augmented use of multimedia data across a wide range of networks and devices. The description of the usage environment in which the multimedia content is consumed is essential to be able to adapt the content to each case.

Frame (frame_num)	Reference	diff	MMCO command	References in DPB
$B^2_{59}(31)$	No			$I^0_{60}(29)$ $B^1_{58}(30)$
$I^0_{64}(31)$	Yes	0	$B^1_{58}(30{=}31\text{-}(0{+}1))$ marked as "unused for ref"	$I^0_{60}(29)$ $I^0_{64}(31)$
$B^1_{62}(32)$	Yes	2	$I^0_{60}(29{=}32\text{-}(2{+}1))$ marked as "unused for ref"	$I^0_{64}(31)$ $B^1_{62}(32)$
$B^2_{61}(33)$	No			$I^0_{64}(31)$ $B^1_{62}(32)$
$B^2_{63}(33)$	No			$I^0_{64}(31)$ $B^1_{62}(32)$
$I^0_{68}(33)$	Yes	0	$B^1_{62}(32{=}33\text{-}(0{+}1))$ marked as "unused for ref"	$I^0_{64}(31)$ $I^0_{68}(33)$
$B^1_{66}(34)$	Yes	2	$I^0_{64}(31{=}34\text{-}(2{+}1))$ marked as "unused for ref"	$I^0_{68}(33)$ $B^1_{66}(34)$
$B^2_{65}(35)$	No			$I^0_{68}(33)$ $B^1_{66}(34)$
$I^0_{72}(35)$	Yes	0	$B^1_{58}(34{=}35\text{-}(0{+}1))$ marked as "unused for ref"	$I^0_{68}(33)$ $I^{01}_{72}(35)$
$B^1_{70}(36)$	Yes	2	$I^0_{68}(33{=}36\text{-}(2{+}1))$ marked as "unused for ref"	$I^0_{72}(35)$ $B^1_{70}(36)$

(a)

Frame (frame_num)	Reference	diff	MMCO command	References in DPB
$IDR^0_{64}(32)$	Yes		(Flush buffer)	$IDR^0_{64}(32)$
$I^0_{68}(33)$	Yes			$IDR^0_{64}(32)$ $I^0_{68}(33)$
$B^1_{66}(34)$	Yes	1	$IDR^0_{64}(32{=}34\text{-}(1{+}1))$ marked as "unused for ref"	$I^0_{68}(33)$ $B^1_{66}(34)$
$B^2_{65}(35)$	No			$I^0_{68}(33)$ $B^1_{66}(34)$
$B^2_{67}(35)$	No			$I^0_{68}(33)$ $B^1_{66}(34)$
$I^0_{72}(35)$	Yes	0	$B^1_{58}(34{=}35\text{-}(0{+}1))$ marked as "unused for ref"	$I^0_{68}(33)$ $I^{01}_{72}(35)$
$B^1_{70}(36)$	Yes	2	$I^0_{68}(33{=}36\text{-}(2{+}1))$ marked as "unused for ref"	$I^0_{72}(35)$ $B^1_{70}(36)$

(b)

TABLE 1. Detail of the DPB management with I access units: (a) input bitstream, (b) summary bitstream.

The Usage Environment Description (UED) tools of MPEG-21 DIA[40, 57, 72] can be used to describe, among others, the terminal capabilities and network characteristics with a standardized specification. The following example shows how some basic, but important, characteristics of the terminal and the network can be described using the *TerminalCapability* and *NetworkCharacteristics* elements. It describes the context of a user who access to the multimedia using a smartphone (with resolution 480x352) through a 384 kbps network.

```
<DIA>
   <Description xsi:type="
       UsageEnvironmentPropertyType">
      <!-- Network description -->
      <UsageEnvironmentProperty xsi:type="
          NetworksType">
         <Network xsi:type="NetworkType">
            <NetworkCharacteristic xsi:type=
                "NetworkConditionType"
                maxCapacity="384000"/>
         </Network>
      </UsageEnvironmentProperty>
      <!-- Terminal description -->
      <UsageEnvironmentProperty xsi:type="
          TerminalsType">
         <Terminal id="smartphone">
            <TerminalCapability xsi:type="
                DisplaysType">
               <Display>
                  <DisplayCapability
                      xsi:type="
                      DisplayCapabilityType"
                      >
                     <Mode>
                        <Resolution
                            horizontal="480"
                             vertical="320"/
                            >
                     </Mode>
                  </DisplayCapability>
               </Display>
            </TerminalCapability>
         </Terminal>
      </UsageEnvironmentProperty>
   </Description>
</DIA>
```

In the application, each user is linked at least to one UED description. Each user may use different terminals or networks depending on the situation. The summarization and adaptation engine must know this information in advance, in order to deliver an approapriate version of the sequence or the summary.

6.2. Summarization units in MPEG-4 SVC

The SVC standard[54] is built as an extension of AVC, including new coding tools for scalable bitstreams. SVC is based on a layered scheme, in which the bitstream is encoded into a n AVC compliant base layer, and one or more enhancenment layers. Each enhancement layer improves the video sequence in one or more of the scalability types. There are different types of scalabilty, temporal, spatial and quality being the most important ones.

Spatial scalability is achieved by using interlayer prediction from a lower spatial layer, in addition

```
<SequentialSummary id="VideoSkimSummary"
    components="visual">
   <SourceLocator><!--Location of the source
        content -->
      <MediaUri>file://video.264</MediaUri>
   </SourceLocator>
   <VisualSummaryComponent>
      <VideoSourceLocator><!--Locates a
          temporal segment in the original
          video -->
         <MediaRelIncrTimePoint
            mediaTimeUnit="PT1N25F"
            mediaTimeBase="../../../
            SourceLocator[1]">793</
            MediaRelIncrTimePoint>
         <MediaDuration>PT32N25F</
            MediaDuration>
      </VideoSourceLocator>
   </VisualSummaryComponent>
   <VisualSummaryComponent>
      <VideoSourceLocator>
         <MediaRelIncrTimePoint
            mediaTimeUnit="PT1N25F"
            mediaTimeBase="../../../
            SourceLocator[1]">954</
            MediaRelIncrTimePoint>
         <MediaDuration>PT32N25F</
            MediaDuration>
      </VideoSourceLocator>
   </VisualSummaryComponent>
   ...
</SequentialSummary>
```

FIGURE 10. Storyboard description in MPEG-7.

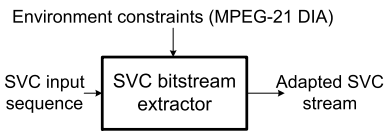


FIGURE 11. Adaptation in the SVC framework.

to intralayer prediction mechanisms such as motion compensated prediction and intra prediction. The same mechanism of interlayer prediction for spatial scalability can provide also Coarse Grain Scalability (CGS) for quality scalability. It can be also achieved using Medium Grain Scalability (MGS), which provides quality refinements inside the same spatial or CGS layer. Temporal scalability in SVC is provided using hierarchical prediction structures, already present in AVC. Each temporal enhancement layer increases the frame rate of the decoded sequence.

In SVC, the versions at different spatial and quality resolutions for a given instant form an AU, and it can contain NAL units from both base and enhancement layers. Each NAL unit belongs to a specific spatial, temporal and quality layer. This information is stored in the header of the NAL unit in the syntax elements *dependency_id*, *temporal_id* and *quality_id*. The length of the NAL unit header in AVC is extended to include this information. In SVC, the base layer is always AVC compatible. However, the extended NAL unit header would make the bitstream non compliant with AVC. For this reason, each base layer NAL unit has a non extended header, but it is preceded by an additional NAL unit containing the SVC related information. These units are called prefix NAL units. If the stream is processed by a AVC decoder, these prefix NAL units and the other enhancement layer NAL units are simply ignored, and the base layer can still be decoded.

In SVC, the concept of SU can be extended, in order to include the additional versions given by spatial and quality scalabilities. Thus, it is possible to define more SUs, with only the NAL units from the base layer, or including also NAL units from enhancement layers, having versions of each SU with different spatial resolutions and qualities.

6.3. Extraction process in MPEG-4 SVC

The extraction process in SVC is non-normative, with the only constraint that the output bitstream, obtained from discarding enhancement layers, must be compliant with the SVC standard. The JVT provides the Joint Scalable Video Model (JSVM), including a software implementation of SVC. In this section, we briefly describe the basic extraction process of SVC in the JSVM.

The extractor processes NAL units using the syntax elements *dependency_id*, *temporal_id* and *quality_id* to decide which ones must be included in the output bitstream. Each adaptation decision is then taken for each access unit AU_n, where n is the temporal instant. Each layer (base or enhancement) in AU_n can be denoted as $L(d,t,q;n)$. An operation point is a specific coordinate (d,t,q) at the temporal instant n, representing a particular (spatial and temporal) resolution and quality, related, respectively, to the syntax elements *dependency_id*, *temporal_id* and *quality_id*. If we denote the extraction process as $\mathcal{E}(OP, AU)$, the result of adapting an access unit AU_n with a particular operation point OP_n can be defined as the adapted access unit $\tilde{AU}_n = \mathcal{E}(OP_n, AU_n)$, containing all the layers and data necessary to decode the sequence at this particular resolution and quality. For each AU_n, the extractor must find the operation point OP_n satisfying the constraints and maximizing the utility of the adaptation. In a typical adaptation scenario, the terminal and network impose constraints that can be fixed (*display_width*, *display_height* and *display_supported_rate*) or variable (*available_bits*(n) related to the instantaneous network available bitrate). Thus, adaptation via bitstream extraction can be formulated as an optimization problem:

for each instant n find $OP_n^* = (d_n^*, t_n^*, q_n^*)$ maximizing

$utility\left(\tilde{AU}_n\right)$
subject to
$frame\_width\,(d) \leq display\_width$
$frame\_height\,(d) \leq display\_height$
$frame\_rate\,(t) \leq display\_frame\_rate$
$bitsize\left(\tilde{AU}_n\right) \leq available\_bits\,(n)$

In this formulation, $utility\left(\tilde{AU}_n\right)$ is a generic measure of utility or quality of the resulting adaptation. It should be computed or estimated for all the possible adapted AUs, in order to select the most appropriate one. The actual values of resolution and frame rate can be obtained indirectly from d and t, and the size of any access unit can be obtained just parsing the bitstream.

The JSVM extractor solves the problem using a prioritization approach. The NAL units in an AU are ordered in a predefined order and selected in this order until the target bitrate or size is achieved. In Figure 12 each block represents a NAL unit containing a layer $L\,(d,t,q;n)$. The base quality layer ($q = 0$) of each spatial and temporal level are placed first in the priority order. Then, NAL units including quality refinements are placed in increasing order of their temporal level. Spatial enhancement layers are placed next. The extractor just drops the NAL units with a priority lower than the required one.

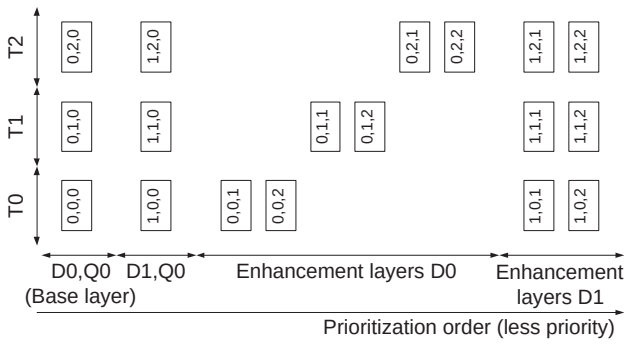


FIGURE 12. Prioritization of NAL units in the JSVM extractor (adapted from [73]).

However, this prioritization scheme does not ensure the optimality of the extraction path in terms of utility. For this reason, besides the basic extraction method, SVC provides additional tools for improved extraction, namely the optional syntax element *priority_id,* which signals explicitly the priority of each NAL unit, based on any other (non-normative) criteria[73].

6.4. Including summarization in the framework

In the previous framework, the constraints imposed to the adaptation engine are external, due to the presence of a constrained usage environment (*environment constraints*). Adaptation modifies the resolution and quality of the bitstream, but the information in the content itself does not change. However, there is no restriction on the nature of the constraints. Summarization can be seen as a modification of the structure of the bitstream based on the information in the content itself, in order to remove semantic redundancies in the temporal axis, in a constrained situation where the number of frames must be reduced considerably. For this reason, we reformulate the video summarization problem (typically, the selection of a suitable set of keyframes or segments) into the problem of finding the appropriate constraints such that the extractor generates a suitable summary. In this context, we call them *summarization constraints.* These constraints can modify the value of the temporal resolution. If both environment and summarization constraints are used together in the extraction, the result is an integrated summarization and adaptation engine which can generate summaries adapted to the usage environment using only SVC tools (see Figure 13).

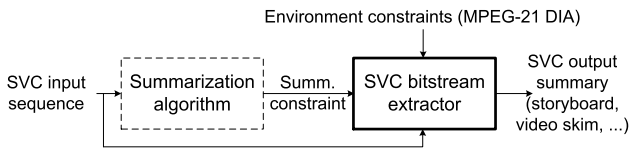


FIGURE 13. Integrated summarization and adaptation of SVC.

The adaptation process, as described previously, is performed on an AU basis. However, in the proposed summarization model, the summaries are referred to the SU index with the summarization constraint $tlevel\,(m)$, so it must be harmonized with the adaptation process. When a sequence is partitioned into SUs, each of them contains one or more AUs and, for simplicity, we assume that each AU belongs only to a single SU. Then, we define a new summarization constraint $\widetilde{tlevel}\,(n)$ for each AU_n associated to a certain SU_m:

$$\widetilde{tlevel}\,(n) \equiv tlevel\,(m)\,, AU_n \in SU_m, \forall n \in \{0,\ldots,N-1\}$$

Note that the MPEG-7 descriptions of summaries are referred to frames rather than SUs, so it is straightforward to obtain $\widetilde{tlevel}\,(n)$ from these descriptions. The problem of adaptation in the extractor, including the new summarization constraint, can be now expressed as

for each instant n find $OP_n^* = (d_n^*, t_n^*, q_n^*)$ maximizing $utility\,(\mathcal{E}\,(OP_n, AU_n))$
subject to
$frame\_width\,(d) \leq display\_width$
$frame\_height\,(d) \leq display\_height$
$frame\_rate\,(t) \leq display\_frame\_rate$
$bitsize\,(\mathcal{E}\,(OP_n, AU_n)) \leq available\_bits\,(n)$
$t \leq \widetilde{tlevel}\,(n)$

The last constraint makes the extraction process content-based, constraining directly the temporal level. The problem can be solved using the same tools described in the previous section, including the

prioritization scheme of the JSVM. Implicitly d, t and q are assumed to be positive (or zero). Thus, if $\widetilde{tlevel}(n)$ takes a negative value for a certain n, the problem has no solution, as the new summarization constraint cannot be satisfied. In that case, we assume that the extractor will skip that AU not including any of its NAL units in the output bitstream. The summarization algorithm can take advantage of this fact to signal when a certain SU must not appear in the output bitstream.

As in the case of AVC, the output bitstream must be decodable, so the extractor must take care of keeping a valid decoding status. Thus, headers must be updated accordingly. Again, the simplest solution is the use of IDR access units. In this case, enhancement layers must also have an IDR access unit at the beginning of each SU, in order to guarantee the independence of the SUs for all layers. In general, other coding units (e.g. with I access units) can be also used, but headers must be updated accordingly, for both base and enhancement layers.

6.5. Further use of MPEG-21 tools

Apart from tools to describe the usage environment, MPEG-21 provides more tools to address the challenge of developing an interoperable framework, including the adaptation of Digital Items. Particularly, MPEG-21 DIA specifies tools to describe the adaptation decision taking and the bitstream adaptation themselves. The adaptation engine has two main modules: the Adaptation Decision Taking Engine (ADTE) and the Bitstream Adaptation Engine (BAE). The ADTE uses the context information and the constraints to make appropriate decisions, while the BAE performs the actual bitstream adaptation, according to the decisions provided by the ADTE.

The proposed framework does not follow any specific standard in these two aspects, which are dependent on the coding format used (MPEG-4 AVC/SVC). In this subsection, we describe how the decision taking can be done using MPEG-21 tools. In addition, we briefly describe the MPEG-21 bitstream adaptation framework, which is independent of the coding format.

6.5.1. Assisted decision taking

MPEG-21 DIA provides tools to assist the adaptation engine to take the appropriate adaptation decisions. In addition to the UED tool, the Adapation Quality of Service (AQoS) and Universal Constraints Description (UCD) tools provide the required information and mechanism to steer the decision taking[74]. The AQoS tool describes what types of adaptation can be applied to a given adaptation unit (in our case, an AU), while the UCD tool declares the constraints between resources and usage environment involved in the decision taking.

The same optimization problem described previously can be stated using AQoS and UCD tools (see Figure 15). We used an utility based framework, in which the ADTE selects the option that maximizes the utility given a set of constraints. In the extractor described in previous section, the utility is not stated explicitly, but it is related to the predefined prioritization scheme following the values of the syntax elements *dependency_id*, *temporal_id* and *quality_id* of each AU, or the more flexible approach using *priority_id*. However, depending on the application, it can be estimated by the extractor, the encoder or any other module and be available as external metadata, using the MPEG-21 Adaptation Quality of Service (AQoS) description tool[74].

The AQoS description contains two main components: modules and IOPins. The IOPins are input and/or output parameters corresponding to fixed values and/or variables. These IOPins can be dependent or independent from other IOPins. In Figure 15, there are some independent IOPins, that correspond to the variables d_n, t_n and q_n. Other IOPins such as *frame_width* and *frame_rate* depend directly from single IOPins (e.g. d_n and t_n respectively), while others, such as *utility*, depend on the combination of all these variables. The value of the utility can be specified using one of the three available modules: Look-Up Table, Utility Function and Stack Function.

The UCD declares constraints between the characteristics of the usage environment (described in the UED) and the feasible values of the IOPins. For instance, *frame_width* cannot be greater than the value of *display_width* specified in the UED.

As explained before, the summarization process is included as additional constraints in the UCD[75]. In the same way as usage environment constraints are obtained from the UED, summarization constraints are obtained from the MPEG-7 description of the summary. In order to follow the same summarization model used in previous sections, we propose the conversion of the description of the summary to the temporal level *tlevel* of each SU, which is used in the constraint $t \leq tlevel$, declared also in the UCD. However, it is possible to use different mechanisms to link the value of the frames and segments in the summary to the UCD.

For each adaptation unit, the ADTE parses both UCD and AQoS descriptions and tries to find a feasible solution, given the constraints. If there is no feasible solution, the decision is to skip the adaptation unit, which means that it is not included in the summary. If there are feasible solutions, the ADTE looks for that with the maximum utility, as stated in the AQoS, and obtains the solution (d_n^*, t_n^*, q_n^*). Then, the BAE selects the appropriate packets from the bitstream according to the solution, and includes these packets into the bitstream of the adapted summary.

6.5.2. Bitstream syntax description framework

In our framework, the BAE corresponds to the SVC bitstream extractor of the JSVM. However, the

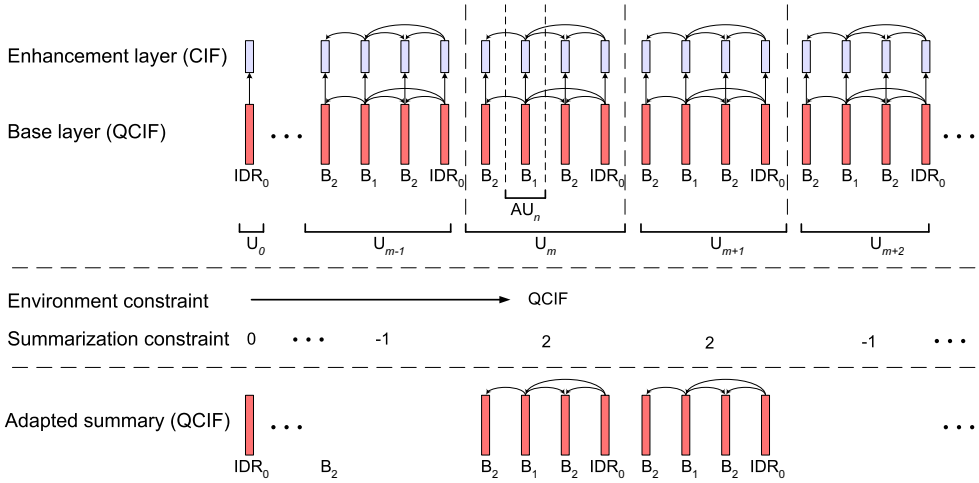


FIGURE 14. Bitstream adaptation guided by summarization and environment constraints.

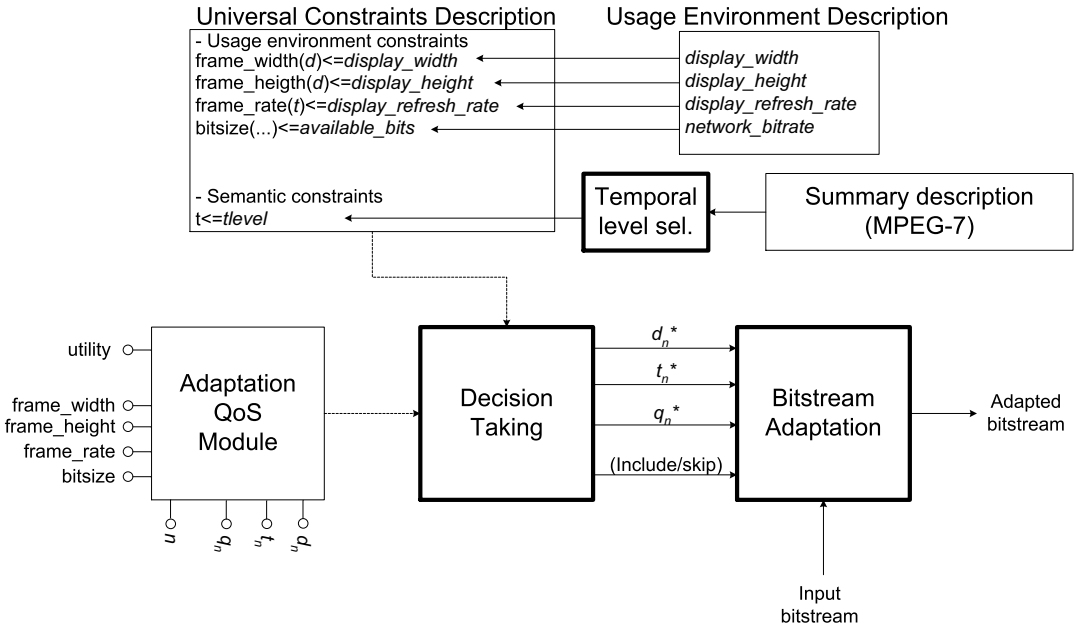


FIGURE 15. Summarization and adaptation engine using MPEG-21 DIA tools.

extractor is a non standard module. Different coding formats need specific BAEs according to their syntax. To solve this dependence on the coding format, MPEG-21 DIA defines a generic framework for bitstream adaptation, independent of the coding format. This framework is based on XML and the concept of Bitstream Syntax Description (BSD)[56]. A BSD is an XML document describing, in a standard way, the high level structure of a bitstream. The adapted bitstream is described by another BSD. The adaptation process consists of the transformation of the first BSD into the second one. Any XML transformation language, such as XSLT or STX, can be used to describe this transformation.

The syntax of a coding format is also described in Bitstream Syntax (BS) schema, and it is used by a specific processor called BintoBSD to generate the BSD from the input bitstream. Once the output BSD is obtained, the ouput bitstream is generated by another processor called BSDtoBin. Along with the BSD, MPEG-21 DIA also defines the generic BSD (gBSD), which is a generic version of BSD. Using bitstream descriptions, if optimized properly, the extractor can be more efficient, as it does not need to parse headers in the bitstream, only the bitstream description.

Scalable formats are organized in such a way that it is easy to obtain adapted versions using few operations such as data truncation and simple header modifications. BSD based adaptation is very suitable for these formats. For instance, SVC can be adapted using BSDs, so the JSVM extractor could be replaced by an extractor based on BSDs[76, 77, 78, 79].

7. AUDIO EXTRACTION AND MULTIPLEXING

Until this section, we have only considered the visual signal contained in an audiovisual stream. However, in practical applications, audio must be also considered. The audio stream must be edited according to the summarization model and multiplexed with the video stream.

The audio extraction process is similar to the video one. Audio streams are coded into packets, which once decoded result in a number of audio samples. The length of the packet depends on the coding format, and other parameters such as the audio rate, the number of channels and the bitrate. We have implemented an audio extractor for MPEG-1 layer III[62, 63, 64]. It can be extended also to other MPEG-1 layers and standards such as MPEG-2 AAC and MPEG-4 AAC, as the coding principles and structures are similar. The duration of each audio frame in MPEG-1 layer III is fixed, and it depends on the number of samples per frame (1152 samples), the sampling rate F_s and the number of channels $N_{channels}$ as

$$t_{audio\_frame} = \frac{1152}{F_s \cdot N_{channels}} \tag{2}$$

For example, considering a frame rate of 22050 Hz and two channels, the duration of each audio frame is approximately 26.12 milliseconds, which corresponds to an audio frame rate of approximately 38.28 frames per second. The frame size is also fixed, depending also on the bitrate R_{audio} and the number of channels, given (in bytes) by

$$s_{audio\_frame} = \frac{144 R_{audio}}{F_s + padding\_bit} \tag{3}$$

where $padding\_bit$ is a parameter (a bit specified in the header of the frame) used, if necessary, to add extra data in order to adjust the bitrate. In our case, a bitstream description is used to specify the exact size of each packet.

The extraction process must be guided by the same summarization constraint $tlevel\,(m)$, which is computed using the video stream as reference. Packets containing video frames (NAL units) have also a fixed duration. For instance, for a video frame rate of 29.97 frames per second, the duration of each video frame would be approximately 33.37 milliseconds. The duration of audio and video frames differs, and the duration of a SU may not match an integer number of audio frames. In that case the extractor must decide either to include or to drop the last audio frame in order to select an integer number of audio frames.

Figure 16 shows an example of mismatch between audio and video frames, and how it results in a non synchronized video when they are multiplexed. The inclusion or dropping of audio frames at the boundaries of a new segment must be decided based on the instantaneous delay between both audio and video streams, in such a way that the delay is compensated. If it is done properly, the maximum delay should not be larger than $t_{audio\_frame}/2$. In the previous example it would be approximately 13.06 milliseconds (26.12 milliseconds in the case of a single audio channel), which is almost imperceptible. However, if a better synchronization is required, a solution may be the dynamic adjustment of time stamps at system level. Thus, whenever a new segment of frames is included, the time stamp of both streams must be adjusted.

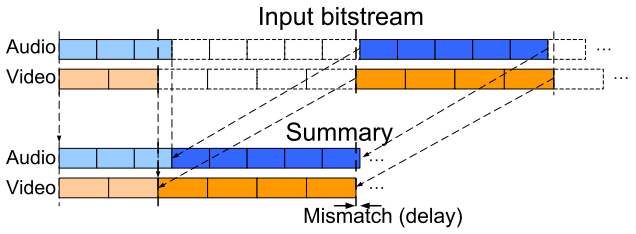


FIGURE 16. Audio extraction and multiplexing.

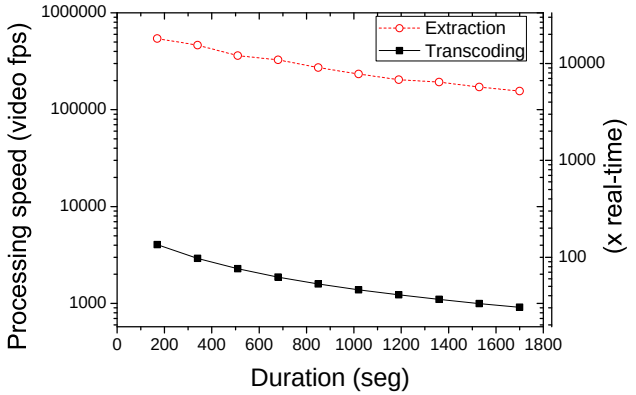


FIGURE 17. Processing speed of audio extraction and transcoding. Video frames per second are shown for easier comparison with other figures.

We evaluated experimentally[6] the efficiency of both transcoding and extraction with an audio file with a frame rate of 22050 Hz, a bitrate of 64 kbps and two channels. As shown in Figure 17 (the processing speed is measured in video frames per second, according to the corresponding video stream), with any of the two approaches, the generation can be perfomed much faster than real time, although extraction is notably faster. The performance degrades slightly as the length of the summary increases, but still being very efficient. However, in an audiovisual stream, the bottleneck for efficient processing is still the generation-adaptation of the video bitstream, as the experiments described in the next section show.

[6] Experiment performed in an Intel Core 2 at 2.83 Ghz (2 GB of RAM)

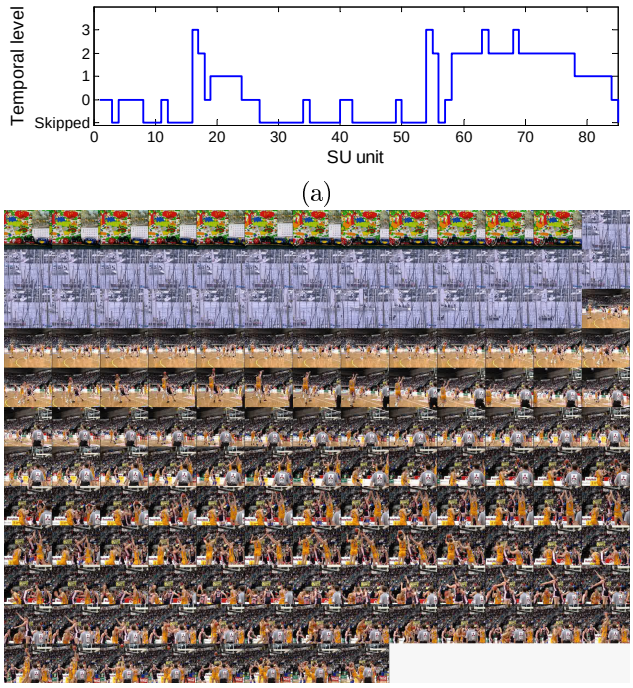


FIGURE 18. Summarization example: a) summarization constraint, and b) resulting summary.

8. EXPERIMENTAL RESULTS

This section shows some experiments to evaluate performance in terms of efficiency and visual quality (rate-distortion performance). For comparison, we also provide experimental results with an alternative approach based on transcoding.

8.1. Example of summarization constraint

First, in order to illustrate how the summarization constraint guides the summarization process, we created a short test sequence by concatenating three sequences used in video coding: *mobile* (low activity), *harbour* (low-medium activity) and *basket* (high activity). Each of these sequences has 300 frames with 4CIF resolution at 30 frames per second. We encoded the resulting sequence with a GOP length of 8 frames and extracted a summarization constraint based on the method proposed in [75], which measures the motion activity and then quantizes the level according to logarithmic thresholds, so the result is a content-based fast forward summary in which low active segments are presented faster than highly active segments. Results are shown in Figure 18.

8.2. Test scenario

For these experiments we assume a test scenario with users accessing content via two types of terminals capable of decoding H.264/AVC and SVC: a terminal with a high resolution display in a broadband network, such as a PC or a TV, and a terminal with a medium resolution display in a medium-low capacity network, such a PDA or mobile phone.

Layer Number	Spatial resolution	Temporal resolution	Quality resolution (QP)
0	CIF	30 Hz	39
1	CIF	30 Hz	30
2	4CIF	30 Hz	40
3	4CIF	30 Hz	29

TABLE 2. Settings of the layers for SVC encoding

The experiments target both efficiency and rate-distortion measures. However, it is difficult to find test sequences suitable for both purposes simultaneously. On the one hand, video summarization itself and the measure of processing time require sequences with a certain length, in order to create meaningful summaries. On the other hand, evaluation of rate-distortion performance in video coding requires test sequences available in uncompressed formats, such as YUV. These sequences are usually very short sequences with a single shot, being not suitable as test sequences for video summarization. For these reasons, we created a longer test sequence using six commonly used YUV sequences (*city*, *crew*, *harbour*, *ice*, *soccer* and *foreman*) concatenated in a single YUV sequence.

We used the reference software JSVM 9.18 in the simulations. The test sequence (1729 frames at 4CIF and 30 frames per second) was encoded in SVC with 2 spatial levels and 2 quality levels, using MGS for quality scalability. The details of these layers are shown in Table 2. Dyadic hierarchical structures were used for temporal scalability with GOP lengths from 1 to 32 frames (1 to 6 temporal levels). In order to compare the approach with a non scalable approach, two additional versions were also encoded in H.264/AVC with the settings of layer 1 (CIF) and layer 3 (4CIF) in Table 2.

Given the test scenario, we considered two target conditions to test the performance of the framework:

- *4CIF@30*. Both spatial and temporal resolutions do not change with respect to the original bitstream. Therefore, neither spatial nor temporal adaptation will be required, and only efficiency in the generation of summaries is studied. This could be the adaptation path for a PC or TV.
- *CIF@15*. In this scenario there is adaptation in both spatial and temporal resolutions. Both generation of the summary and adaptation to the target conditions are studied. This could be the adaptation path for a PDA or mobile phone.

The summaries were generated and adapted to the test conditions with the following methods (see Table 3):

- *AVC transcoding*. The sequence is first decoded to YUV format. The summary is generated and adapted (if required) into another YUV sequence,

which is finally encoded to H.264/AVC. For the case CIF@15 there are two possibilities, depending on which AVC version is used as input bitstream (4CIF or CIF).

- *SVC extraction.* It uses the SVC bitstream extractor to select the required packets and to generate the adapted summary.
- *AVC extraction.* The same bitstream extractor is used in this case (either from 4CIF version or CIF version). This method can be used only when neither spatial nor quality adaptation are required.
- *AVC hybrid.* This method complements the previous one, as the summary is first generated using extraction from the 4CIF AVC bitstream, and then it is transcoded to the adapted version of the summary in CIF. Note that, compared to transcoding from the 4CIF version, only a few frames (depending on the length of the summary) are processed, as most of them were discarded during extraction.

For AVC transcoding and AVC hybrid methods, the settings of the encoder were modified to reduce significantly the computational burden due to encoding. Thus, a fast search method was used with a smaller search range (8 pixels).

For the purpose of this paper and these experiments, the summarization algorithm itself is out of the scope, and it could be any algorithm. Depending on the application, the summarization algorithm should be selected in order to obtain the most appropriate summary. However, the summarization method itself will have no significant impact on the performance of the proposed framework, in terms of visual quality and efficiency. Thus, in this case we used a simple method consisting of sampling the sequence at constant intervals, selecting single frames for storyboards and segments of 64 frames (aproximately 2 seconds) for video skims. Although extremely simple, this method is very suitable for the test sequence used in the experiment, as the shots have similar lengths and they are distributed regularly in the sequence, so it is very likely that the algorithm creates summaries covering most of the shots.

8.3. Efficiency

We compared transcoding and extraction for different SU lengths (GOP length in the experiments). Figure 19 shows the results for a video skim (20% of the total length). We used the processing speed (as the number of frames of the input sequence divided by the processing time), measured in frames per second[7]. As expected, transcoding is much slower than methods based on extraction. Both SVC extraction and AVC extraction have very good performance, over 1000 frames per second. The latter is faster for both 4CIF and CIF as it needs to parse a lower number of NAL units, due to the absence of enhancement layers and the smaller size of the bitstream.

[7]Experiment performed in an Intel Xeon at 2.83 Ghz (24 GB of RAM)

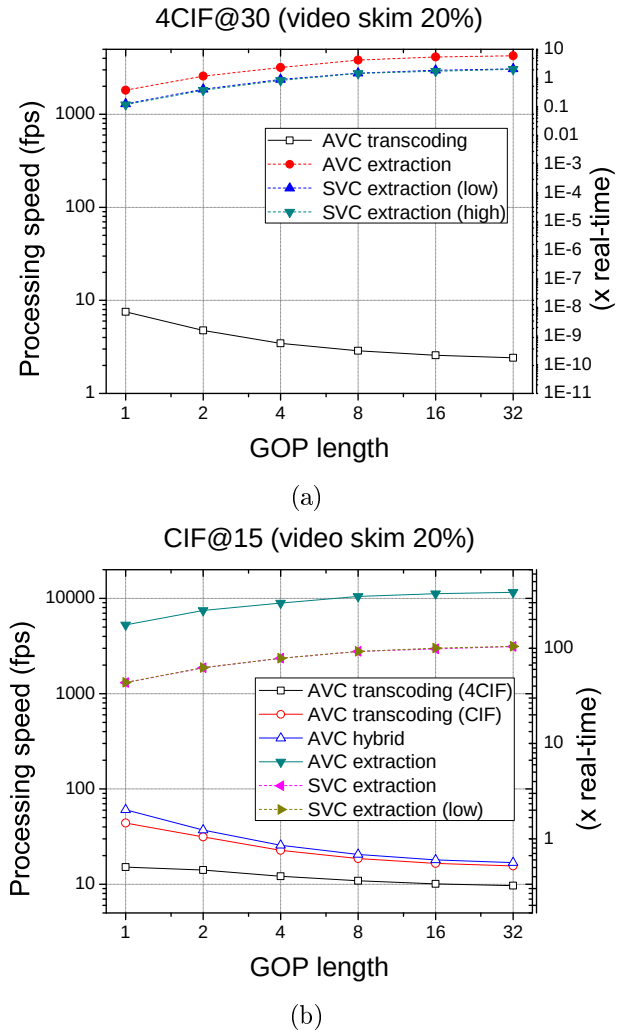


FIGURE 19. Dependency of the processing speed with the GOP length: a) 4CIF 30 Hz and, b) CIF 15 Hz.

The length of the coding unit has different effects on transcoding and extraction. In the case of extraction, longer GOPs result on smaller bitstreams, which are processed faster, as extraction basically is a selective packet forwarding operation. On the contrary, encoding using longer GOPs requires more computational effort on motion estimation. Thus, the efficiency of transcoding decreases as the GOP length increases.

In the case of transcoding to CIF there are three possibilties. Transcoding from the CIF version is faster than transcoding from the 4CIF version, due to the faster decoding of lower resolution sequences. The hybrid method combining extraction and transcoding is also faster than pure transcoding from 4CIF. However, their performance is still quite far from that of extraction approaches.

Method (resolution)	Spatial resolution (input/output)	Temporal resolution (input/output)
Adaptation to 4CIF@30		
AVC transcoding (4CIF)	4CIF/4CIF	30/30 Hz
AVC extraction (4CIF)	4CIF/4CIF	30/30 Hz
SVC extraction (4CIF)	4CIF/4CIF	30/30 Hz
SVC extraction (4CIF low)	4CIF/4CIF	30/30 Hz
Adaptation to CIF@15		
AVC transcoding (4CIF)	4CIF/CIF	30/15 Hz
AVC transcoding (CIF)	CIF/CIF	30/15 Hz
AVC extraction (CIF)	CIF/CIF	30/15 Hz
AVC hybrid	4CIF/CIF	30/15 Hz
SVC extraction (CIF)	4CIF/CIF	30/15 Hz
SVC extraction (CIF low)	4CIF/CIF	30/15 Hz

TABLE 3. Methods and cases used in the experiments.

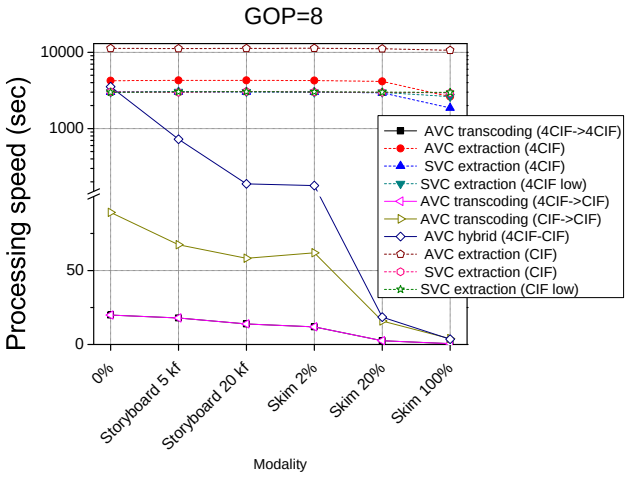


FIGURE 20. Processing speed for different modalities. Note that half of the vertical scale is linear and the rest is logarithmic.

The different methods were also compared for several modalities and summary lengths, ranging from the empty to the whole sequence, using a GOP length of 8 frames (see Figure 20). Methods based on extraction also have an almost constant performance for all the summary lengths, slightly degraded for long summaries. Methods based on transcoding are more sensitive to the length of the summary. For short summaries (e.g. storyboards), most of the processing time in transcoding is due to decoding, as only a few frames are encoded in contrast to the decoding of all frames. However, a significant increment of the processing time can be observed for longer summaries. The hybrid method reduces the number of frames to be decoded, increasing the performance dramatically for short summaries, although it is still degraded for long summaries due to transcoding.

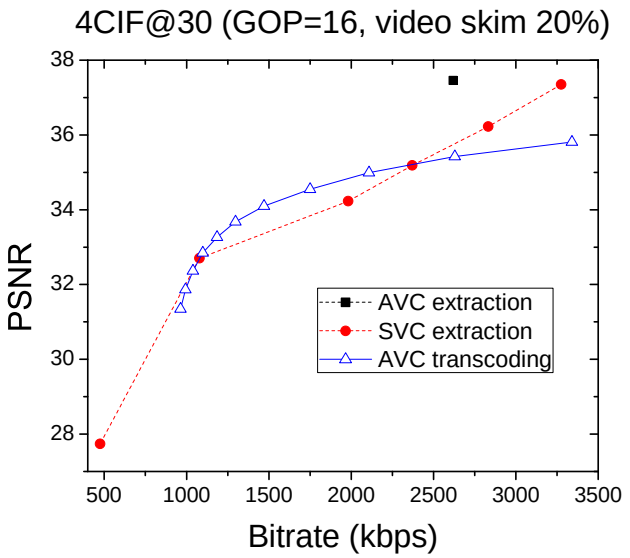
The generation of a summary with the 0% of the frames in the sequence (an empty summary) is very useful to have a reference of the time used in initialitation and other processes independent of the length of the summary. In transcoding, this time is due to the decoding of all frames, and it is the most important contribution to the overall processing time. In the case of extraction, the JSVM extractor performs the extraction in two passes. In the first pass, all the NAL headers are parsed in order to obtain a description of the bitstream, which is then used to perform the actual extraction. As it can be seen in the figure, most of the extraction time is used in this first pass. The use of bitstream descriptions reduces significantly the time required for this first pass, which in that case would consist of parsing the bitstream description instead of parsing the whole bitstream.

As experiments showed, a simple solution based on extraction has better performance than others based on transcoding, for the purpose of video summarization and adaptation. A hybrid solution based on both extraction and transcoding can also be useful when neither spatial nor quality scalability are available, especially for short summaries such as storyboards.

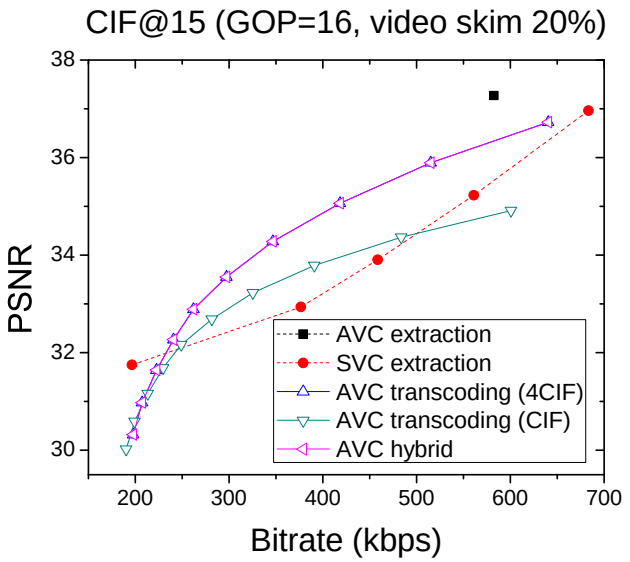
Note that this experiment compares the JSVM reference implementations, which are not optimized for speed. Their efficiency could be greatly improved by using hardware implementations, for both transcoding and extraction. Note also that transcoding is often accelerated by using simplified implementations (e.g. small search windows, reusing motion vectors)[49, 51, 50], leading to a degraded rate-distortion performance.

8.4. Rate-distortion performance

As explained before, a single layer AVC extraction outperforms AVC transcoding in rate-distortion performance, as transcoding implies an additional quantization stage. However the multilayered approach of SVC has a penalty in coding efficiency compared to a single layer version. In the case of the experiment, the 4CIF SVC version of the test sequence is encoded predictively from the other 3 versions while the 4CIF AVC version is encoded directly into a single layer, which is more optimal in terms of rate-distortion performance. Although



(a)



(b)

FIGURE 21. Comparison of average PSNR: a) 4CIF 30 Hz and, b) CIF 15 Hz.

the penalty due to scalability is small in MPEG-4 SVC, each additional layer increments the overall penalty compared to a single layer version. Thus, although SVC extraction avoids the additional quantization stage, its performance was degraded previously compared to the AVC version by the use of multiple scales.

We studied the rate-distortion performance of a video skim (20%) extracted from the original bitstream coded with a GOP length of 8 frames. Figure 21 shows the rate-distortion curves obtained for the two test scenarios. In both cases AVC extraction has the best results, as it is a single layer and does not need to re-encode the sequence.

Transcoding curves were obtained varying the quantization parameter. It outperforms SVC extraction in the middle of the bitrate range. Figure 21b also shows that transcoding from a higher resolution version (4CIF) has a better performance (the AVC hybrid approach has exactly the same rate-distortion performance).

SVC extraction works better at the low and high ends of the bitrate range, as they correspond to the operation points represented in Table 2, while the intermediate points are obtained by discarding transform coefficients. Rate-distortion performance at these points is significantly worse.

However, both AVC transcoding and SVC extraction performances can be improved using different configurations. Transcoding can be improved using a larger search range for the motion estimation algorithm, at the cost of less efficient processing. Using fewer enhancement layers (e.g. removing layers 0 and 2 from Table 2 to remove quality scalability) also improves the rate-distortion performance of the remaining operation points. Alternatively, quality scalability in SVC bitstreams can be slightly optimized using rate-distortion analysis and priority identifiers[73].

9. COMPARISON OF ARCHITECTURES

In this section, based on the results of the experiments, we compare the three basic architectures to provide adapted video and adapted summaries.

The simplest architecture consists of using a file for each of the versions (summaries and adapted versions), which, following the nomenclature of MPEG-7 Multimedia Description Schemes (MDS)[59], are called variations. MPEG-7 MDS defines the *Variation* description scheme to describe each of these elements (e.g. different bitrate, spatial or temporal resolution)[44, 46, 47]. Each variation is created and encoded prior to the interaction with users. A summary is a type of semantic adaptation of the content, and it can be considered as another variation, which in turn, may have different variations (e.g. bitrate, resolution). The advantage of variations is that, once the files are created and stored, the adaptation process is very simple, as the server only has to select the most appropriate variation, according to the request and the usage environment, and deliver it to the user. The most critical and time-consuming task, which is the generation and adaptation of summaries, is completely carried out prior to the interaction with users. However, it requires a significant amount of storage space and the potential adapted versions are limited to those available as variations.

The second possible architecture is on demand transcoding, with flexibility as the main advantage, since it can cope any possible adaptation (only limited by the actual capabilities of the transcoder). In contrast, transcoding video content is usually a very time consuming process.

In some sense, extraction using SVC has features of both variations and transcoding approaches, as each

file has embedded multiple summaries and versions of the same video content (embedded multiple variations), which also can be created on demand (as in transcoding) by selecting the appropriate packets of the bitstream.

The differences between the three approaches are summarized in Table 4.

9.1. Efficiency

The experiments showed that extraction is notably more efficient than transcoding, mainly because of the simplicity of the generation-adaptation process. In a server based on variations, the process is even simpler, i.e. select and forward a suitable pre-stored bitstream, which would be slightly faster than extraction.

Extraction and variations have also the advantage of consuming little computer resources such as memory and CPU usage, in contrast to transcoding, which is extremely demanding, especially for high resolution sequences. Thus, a single computer which could serve tens of clients using extraction and variations, could serve just a few using transcoding.

Closely related to efficiency, the generation-adaptation delay is another important factor which may be decisive in some applications. The delay would depend on the processing load which would also depend on the number of connections. In some applications, such as personalized[41, 80] and scalable summaries[81, 82, 83], a low delay is critical for a satisfactory user experience.

9.2. Rate-distortion performance

Although not critical, in the sense that the content would reach the user even with lower quality, providing the user with the best video quality is important for a good user experience.

For single layer AVC, used when neither quality nor spatial adaptation are required, extraction preserves the original quality, while transcoding has some degradation (due to a second lossy stage, i.e. requantization). This degradation also depends on the configuration of the encoder. Usually, efficiency and rate-distortion performance are traded off, although transcoding has always some quality loss (except for some special cases, such as idempotent coding[84]). Using variations, if they are generated from the original uncompressed sequence, they have the same quality as those versions obtained using extraction. However, if the variations are generated from a previously coded bitstream (i.e. decoding and re-encoding), there would be a second quantization stage that would degrade the quality compared to the original uncompressed sequence. Using a transcoder with a high quality configuration (large motion estimation search window, advanced coding tools) to generate the variation would help to lessen the quality loss, although the encoding process would be very demanding and slow at the preprocessing stage.

For SVC, there is some quality loss compared to the single layer case (i.e. AVC). This quality loss is due to the coding penalty associated with layered coding. Thus, only the base layer does not degrade. However, the other enhanced versions may have worse quality than transcoding, depending on the operation point (see Figure 21). In this case, variations generated from the uncompressed original sequence provide the best quality, as they do not have requantization and each of them is a single layer bitstream.

9.3. Storage requirements

The main drawback of the use of variations is that each one must be stored in a separated file. Thus, systems using a large number of variations (due to a large number of summaries, adapted versions of summaries and/or adapted versions of the main video) may require large storage resources, especially for high resolution sequences. In contrast, transcoding and adaptation only require the storage of one file. Due to the coding penalty of layered coding, given the same quality (i.e. PSNR), extraction could require slightly more space.

A scalable summary is a special case in which the number of potential summaries may be very large. [81] proposes a framework which can generate hundreds of video skims, each of them with a different length. And each of them must be stored separately. While for storyboards may be feasible, for video skims the storage requirements could be unacceptable.

9.4. Coding

Variations and transcoding can provide codec adaptation. In principle, variations can be stored in any coding format supported by the encoder. Thus, the system can deliver different versions with the same characteristics but with different coding formats (e.g. MPEG-1, MPEG-2, H.263, AVC), useful in heterogenous scenarios in which the different terminals have different decoding capabilities (e.g. codecs, profiles). Transcoders may also adapt the content to any coding format, in principle. However, extraction relies on a specific scalable codec, either AVC for temporal scalability, or SVC for extended adaptation. All the terminals must support SVC decoding. There are two special cases in which AVC decoding would be enough. The first one is the case in which only the base layer is required. The second one is the use of SVC-to-AVC rewritting[85, 86], in which the SVC bitstream is converted to an AVC single layer bitstream. However, the latter is closer to lightweight transcoding than to extraction. Adaptation to other codecs is not possible with extraction using SVC.

A similar capability is the adaptation to arbitrary resolutions and bitrates. Variations and transcoding may support any arbitrary spatial and temporal resolution and bitrate, provided that an appropriate

		Variations	Transcoding	Extraction
Efficiency		Very high	Low-very low	Very high (AVC) High (SVC)
Quality		Best (if the source is uncompressed) Good (with high quality transcoding)	Medium-poor (depending on encoding parameters)	Best (AVC) Good-medium (SVC)
Storage requirements		High	Low	Low
Precision		Frame	Frame	SU/GOP
Delay		Very low	High	Low-very low
Coding	Support	Any (but must be available in the stored versions)	Any supported by the transcoder	AVC/SVC or other scalable codec
	New codecs	Yes (requires reencoding and extra storage space)	Yes (requires to include the codec in the transcoder)	No
	Potential adaptations	Any (but must be available in the stored versions)	Any (supported by the transcoder)	Those available in the coded bitstream Usually dyadic decompostions (e.g. 4CIF, CIF, QCIF, 15 Hz, 30 Hz)
Flexibility	Cost of a new version	Encoding and additional storage space	No additional cost	No additional cost if already embedded in the bitstream
	Cost of a new summary	Encoding and additional storage space	No additional cost	No additional cost
	Customized summaries	No	Yes	Yes

TABLE 4. Comparison of summarization-adaptation architectures.

encoder or transcoder is used. The transcoder would perform the adaptation on demand, while the system using variations must have created the variation previously. However extraction only supports those versions embedded in the original bitstream, which are typically encoded using dyadic decompositions in temporal (e.g. 15, 30 frames per second) and spatial dimensions (e.g. 4CIF, CIF), and possibly several bitrates.

9.5. Flexibility

Transcoding is the most flexible of the three approaches, as the inclusion of new summaries or versions not considered in an initial design do not require any additional processing (other than the description of the summary). The new version is generated on demand with the same cost of any other version. Extraction is still a flexible approach, although limited by the versions available in the source bitstream and a lower precision to describe summaries (i.e. the length of the summarization unit, in contrast to frame precision in variations and transcoding). However, in most cases that flexibility is more than enough. In contrast, the use of variations does not provide any flexibility, as any version not considered initially cannot be generated.

9.6. Hybrid architectures

In a practical application, it is not necessary to use strictly only one of the previous architectures. Hybrid architectures, combining pre-stored variations with transcoding or extraction, may be more suitable, and will depend on the specific scenario and requirements.

One example of hybrid architecture would be a transcoding architecture with caching. In that case, summaries and adapted versions are generated by transcoding the source content on demand. However, the server stores all the previously generated files, as variations. Thus, if the user requests any variation that was requested previously, the server just delivers the cached file. Caching trades off efficiency and storage space. As described in a previous section, transcoding and extraction can also be combined using a first extraction stage followed by a transcoder. That reduces significantly part of the cost of transcoding. Another example is the combination of different approaches for images and video files. Image based summaries, such as storyboards, require much less storage space than video based summaries. Thus, image based summaries can be stored as variations, while video based summaries are generated via transcoding or extraction.

10. CONCLUSIONS

We described an application using tools from four MPEG standards in the context of video summarization and adaptation. Summaries are described using MPEG-7 MDS summarization tools, while the information about the usage environment, mainly terminal and network characteristics, are described using the UED tool specified in MPEG-21 DIA. The main advantage of having public and standarized syntax is the possibility of reusing the same descriptions in other applications and systems with MPEG-7 and MPEG-21 compliant devices, capable of parsing them.

The main advantage of the proposed framework is its simplicity, which is suitable for lightweight and

fast processing. Using an integrated approach the generation of the summary has all the advantages of bitstream extraction in terms of efficiency. Another advantage is the adaptation, in the same process, of summaries to a specific usage environment, using the layered approach of SVC.

Based on the work in [3, 4], in this paper we provided further details about the low level implementation of the extraction mechanism and audio/video synchronization, and analyzed the architectures and rate-distortion performance. Additionally, qualitative and quantitative comparisons with other two alternative approaches (transcoding and variations) are included in this paper, discussing the advantages and drawbacks of each of them.

The experimental results show that extraction is much more efficient than transcoding, outperforming transcoding in rate-distortion performance in the case of single layer AVC, and being comparable in the case of SVC (depending on the configuration and number of layers). Thus, certain frameworks may be appropriate in different scenarios. While variations are useful if few adapted versions and summaries are required, transcoding is a much more flexible approach with the drawback of being computationally expensive and slow. The proposed extraction method is a flexible yet simple approach that may be suitable in a broad range of applications, especially in those ones in which both flexibility and efficiency are required, such as customized and scalable summaries.

Future work may focus on extending the framework to other coding formats. Another line of research is the integration with scalable summaries[81], in which summaries themselves can be scaled semantically (i.e. the longer the more informative). In the proposed framework scalability is used only for usage environment adaptation, but it could be integrated to obtain easily adaptable and interactive summaries.

ACKNOWLEDGEMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61150110480 and in part by the Chinese Academy of Sciences Fellowships for Young International Scientists under Grant 2011Y1GB05.

REFERENCES

[1] Yeung, M. and Yeo, B.-L. (1997) Video visualization for compact presentation and fast browsing of pictorial content. *IEEE Transactions on Circuits and Systems for Video Technology*, **7**, 771–785.

[2] Pfeiffer, S., Lienhart, R., Fischer, S., and Effelsberg, W. (1996) Abstracting digital movies automatically. *Journal of Visual Communication And Image Representation*, **7**, 345–353.

[3] Herranz, L. and Martínez, J. M. (2009) An integrated approach to summarization and adaptation using H.264/MPEG-4 SVC. *Signal Processing: Image Communication*, **24**, 499–509. Special Issue on Scalable Coded Media beyond Compression.

[4] Herranz, L. and Martínez, J. M. (2009) On the use of hierarchical prediction structures for efficient summary generation of H.264/AVC bitstreams. *Signal Processing: Image Communication*, **24**, 615 – 629.

[5] Chang, H. S., Sull, S., and Lee, S. U. (1999) Efficient video indexing scheme for content-based retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, **9**, 1269–1279.

[6] Dimitrova, N., Zhang, H.-J., Shahraray, B., Sezan, I., Huang, T., and Zakhor, A. (2002) Applications of video-content analysis and retrieval. *IEEE Multimedia*, **9**, 42–55.

[7] Zhu, X., Elmagarmid, A., Xue, X., Wu, L., and Catlin, A. (2005) InsightVideo: toward hierarchical video content organization for efficient browsing, summarization and retrieval. *IEEE Transactions on Multimedia*, **7**, 648–666.

[8] Li, Z., Schuster, G., Katsaggelos, A., and Gandhi, B. (2005) Rate-distortion optimal video summary generation. *IEEE Transactions on Image Processing*, **14**, 1550–1560.

[9] Truong, B. T. and Venkatesh, S. (2007) Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications and Applications*, **3**, 3.

[10] Money, A. G. and Agius, H. (2008) Video summarisation: A conceptual framework and survey of the state of the art. *Journal of Visual Communication and Image Representation*, **19**, 121–143.

[11] Valdés, V. and Martínez, J. M. (2010) A framework for video abstraction systems analysis and modelling from an operational point of view. *Multimedia Tools and Applications*, **49**, 7–35.

[12] Marchionini, G., Wildemuth, B. M., and Geisler, G. (2006) The Open Video digital library: A Möbius strip of research and practice. *Journal of the American Society for Information Science and Technology*, **57**, 1629–1643.

[13] Ma, Y.-F., Hua, X.-S., Lu, L., and Zhang, H.-J. (2005) A generic framework of user attention model and its application in video summarization. *IEEE Transactions on Multimedia*, **7**, 907–919.

[14] Smith, M. and Kanade, T. (1998) Video skimming and characterization through the combination of image and language understanding. *Proceedings of IEEE International Workshop on Content-Based Access of Image and Video Database*, 3 Jan., pp. 61–70.

[15] Li, Y., Lee, S.-H., Yeh, C.-H., and Kuo, C.-C. (2006) Techniques for movie content analysis and skimming: tutorial and overview on video abstraction techniques. *IEEE Signal Processing Magazine*, **23**, 79–89.

[16] Wildemuth, B., Marchionini, G., Yang, M., Geisler, G., Wilkens, T., Hughes, A., and Gruss, R. (2003) How fast is too fast? evaluating fast forward surrogates for digital video. *Proceedings of Joint Conference on Digital Libraries*, 27-31 May, pp. 221–230.

[17] Peker, K., Divakaran, A., and Sun, H. (2001) Constant pace skimming and temporal sub-sampling of video using motion activity. *Proceedings of International*

Conference on Image Processing, 7-10 Oct., pp. 414–417.

[18] Bescós, J., Martínez, J. M., Herranz, L., and Tiburzi, F. (2007) Content-driven adaptation of on-line video. *Signal Processing: Image Communication*, **22**, 651–668.

[19] Lotfallah, O. A., Reisslein, M., and Panchanathan, S. (2006) Adaptive video transmission schemes using MPEG-7 motion intensity descriptor. *IEEE Transactions on Circuits and Systems for Video Technology*, **16**, 929–946.

[20] Gang, Z., Chia, L.-T., and Zongkai, Y. (2004) MPEG-21 digital item adaptation by applying perceived motion energy to H.264 video. *Proceedings of International Conference on Image Processing*, 24-27 Oct., pp. 2777–2780.

[21] Calic, J., Gibson, D., and Campbell, N. (2007) Efficient layout of comic-like video summaries. *IEEE Transactions on Circuits and Systems for Video Technology*, **17**, 931–936.

[22] Mrak, M., Calic, J., and Kondoz, A. (2009) Fast analysis of scalable video for adaptive browsing interfaces. *Computer Vision and Image Understanding*, **113**, 425—434. Special Issue on Video Analysis.

[23] Ngo, C.-W., Ma, Y.-F., and Zhang, H.-J. (2003) Automatic video summarization by graph modeling. *Proceedings of Ninth IEEE International Conferenceon Computer Vision*, pp. 104–109.

[24] Li, B. and Ibrahim Sezan, M. (2001) Event detection and summarization in sports video. *Proceedings of Workshop on Content-Based Access of Image and Video Libraries*, pp. 132–138.

[25] Ekin, A., Tekalp, A., and Mehrotra, R. (2003) Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, **12**, 796–807.

[26] Tjondronegoro, D., Chen, Y.-P. P., and Pham, B. (2004) Highlights for more complete sports video summarization. *IEEE Multimedia*, **11**, 22–37.

[27] Zhao, Z., Jiang, S., Huang, Q., and Zhu, G. (2006) Highlight summarization in sports video based on replay detection. *Proceedings of International Conference on Multimedia and Expo*, pp. 1613–1616.

[28] Babaguchi, N., Ohara, K., and Ogura, T. (2007) Learning personal preference from viewer's operations for browsing and its application to baseball video retrieval and summarization. *IEEE Transactions on Multimedia*, **9**, 1016–1025.

[29] Maybury, M., Greiff, W., Boykin, S., Ponte, J., McHenry, C., and Ferro, L. (2004) Personalcasting: Tailored broadcast news. *User Modeling and User-Adapted Interaction*, **14**, 119–144.

[30] Damnjanovic, U., Piatrik, T., Djordjevic, D., and Izquierdo, E. (2007) Video summarisation for surveillance and news domain. *Semantic Multimedia*, Lecture Notes in Computer Science, **4816**, pp. 99–112. Springer-Verlag Berlin.

[31] Lie, W.-N. and Lai, C.-M. (2005) News video summarization based on spatial and motion feature analysis. *Advances in Multimedia Information Processing*, Lecture Notes in Computer Science, **3332**, pp. 246–255. Springer-Verlag Berlin.

[32] Zhang, H. J., Wu, J., Zhong, D., and Smoliar, S. W. (1997) An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, **30**, 643 – 658. Image Databases.

[33] Peng, W.-T., Huang, W.-J., Chu, W.-T., Chou, C.-N., Chang, W.-Y., Chang, C.-H., and Hung, Y.-P. (2008) A user experience model for home video summarization. *Semantic Multimedia*, Lecture Notes in Computer Science, **5371**, pp. 484–495. Springer-Verlag Berlin.

[34] Fonseca, P. M. and Pereira, F. (2004) Automatic video summarization based on MPEG-7 descriptions. *Signal Processing: Image Communication*, **19**, 685–699.

[35] Over, P., Smeaton, A. F., and Kelly, P. (2007) The TRECVid 2007 BBC rushes summarization evaluation pilot. *Proceedings of the International Workshop on TRECVid video summarization*, New York, NY, USA, pp. 1–15. ACM.

[36] Over, P., Smeaton, A. F., and Awad, G. (2008) The TRECVid 2008 BBC rushes summarization evaluation. *Proceedings of the TRECVid Video Summarization Workshop*, New York, NY, USA, pp. 1–20. ACM.

[37] Ren, J. and Jiang, J. (2009) Hierarchical modeling and adaptive clustering for real-time summarization of rush videos. *IEEE Transactions on Multimedia*, **11**, 906–917.

[38] Chang, S.-F. and Vetro, A. (2005) Video adaptation: concepts, technologies, and open issues. *Proceedings of the IEEE*, **93**, 148–158.

[39] Vetro, A., Christopoulos, C., and Ebrahimi, T. (2003) From the guest editors - universal multimedia access. *IEEE Signal Processing Magazine*, **20**, 16.

[40] Vetro, A. (2004) MPEG-21 digital item adaptation: Enabling universal multimedia access. *IEEE Multimedia*, **11**, 84–87.

[41] Tseng, B., Lin, C.-Y., and Smith, J. (2004) Using MPEG-7 and MPEG-21 for personalizing video. *IEEE Multimedia*, **11**, 42–52.

[42] Cavallaro, A., Steiger, O., and Ebrahimi, T. (2005) Semantic video analysis for adaptive content delivery and automatic description. *IEEE Transactions on Circuits and Systems for Video Technology*, **15**, 1200–1209.

[43] Cheng, W.-H., Wang, C.-W., and Wu, J.-L. (2007) Video adaptation for small display based on content recomposition. *IEEE Transactions on Circuits and Systems for Video Technology*, **17**, 43–58.

[44] van Beek, P., Smith, J., Ebrahimi, T., Suzuki, T., and Askelof, J. (2003) Metadata-driven multimedia access. *IEEE Signal Processing Magazine*, **20**, 40–52.

[45] Magalhaes, J. and Pereira, F. (2004) Using MPEG standards for multimedia customization. *Signal Processing: Image Communication*, **19**, 437–456.

[46] Böszörményi, L., Hellwagner, H., Kosch, H., Libsie, M., and Podlipnig, S. (2003) Metadata driven adaptation in the ADMITS project. *Signal Processing: Image Communication*, **18**, 749 – 766. Special Issue on Multimedia Adaptation.

[47] Libsie, M. and Kosch, H. (2004) Video adaptation using the variation factory. *Proceedings of IEEE Workshop on Multimedia Signal Processing*, pp. 403–406.

[48] Ahmad, I., Wei, X., Sun, Y., and Zhang, Y.-Q. (2005) Video transcoding: an overview of various techniques

and research issues. *IEEE Transactions on Multimedia*, **7**, 793–804.

[49] Xin, J., Lin, C.-W., and Sun, M.-T. (2005) Digital video transcoding. *Proceedings of the IEEE*, **93**, 84–97.

[50] Lefol, D., Bull, D., Canagarajah, N., and Redmill, D. (2007) An efficient complexity-scalable video transcoder with mode refinement. *Signal Processing: Image Communication*, **22**, 421 – 433.

[51] De Cock, J., Notebaert, S., and Van de Walle, R. (2007) A novel hybrid requantization transcoding scheme for H.264/AVC. *Proc. of International Symposium on Signal Processing and Its Applications*, pp. 1–4.

[52] Adami, N., Signoroni, A., and Leonardi, R. (2007) State-of-the-art and trends in scalable video compression with wavelet-based approaches. *IEEE Transactions on Circuits and Systems for Video Technology*, **17**, 1238–1255.

[53] Ohm, J.-R. (2005) Advances in scalable video coding. *Proceedings of the IEEE*, **93**, 42–56.

[54] Schwarz, H., Marpe, D., and Wiegand, T. (2007) Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, **17**, 1103–1120.

[55] Sullivan, G. J. and Wiegand, T. (2005) Video compression - from concepts to the H.264/AVC standard. *Proceedings of the IEEE*, **93**, 18–31.

[56] Devillers, S., Timmerer, C., Heuer, J., and Hellwagner, H. (2005) Bitstream syntax description-based adaptation in streaming and constrained environments. *IEEE Transactions on Multimedia*, **7**, 463–470.

[57] (2003). ISO/IEC 21000-7,Information Technology - Multimedia Framework (MPEG-21) - Part 7: Digital Item Adaptation.

[58] De Bruyne, S., De Schrijver, D., De Neve, W., Van Deursen, D., and Van de Walle, R. (2007) Enhanced shot-based video adaptation using mpeg-21 generic bitstream syntax schema. *IEEE Symposium on Computational Intelligence in Image and Signal Processing*, 1-5 April, pp. 380–385.

[59] (2001). ISO/IEC 15938-5,Information Technology - Multimedia Content Description Interface - Part 5: Multimedia Description Schemes.

[60] ITU-T and ISO/IEC (2003). Advanced video coding for generic audiovisual services.

[61] Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A. (2003) Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, **13**, 560–576.

[62] ITU-T and ISO/IEC (1992). Coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s - part3: Audio.

[63] Musmann, H. (2006) Genesis of the MP3 audio coding standard. *IEEE Transactions on Consumer Electronics*, **52**, 1043 –1049.

[64] Pan, D. (1995) A tutorial on MPEG/audio compression. *IEEE Multimedia*, **2**, 60 –74.

[65] ITU-T and ISO/IEC (2007). Generic coding of moving pictures and associated audio information - part 7: Advanced audio coding.

[66] Noll, P. (1997) MPEG digital audio coding. *IEEE Signal Processing Magazine*, **14**, 59 –81.

[67] ITU-T and ISO/IEC (2005). Coding of audio-visual objects - part 3: Audio.

[68] Herre, J. and Dietz, M. (2008) MPEG-4 high-efficiency AAC coding [standards in a nutshell]. *IEEE Signal Processing Magazine*, **25**, 137 –142.

[69] Tourapis, A. M., Leontaris, A., Sühring, K., and Sullivan, G. (2007) H.264/MPEG-4 AVC reference software manual. JVT-X072. Int. Telecommun. Union-Telecommun. (ITU-T) and Int. Standards Org./Int. Electrotech. Comm. (ISO/IEC).

[70] ITU-T and ISO/IEC (2001). ISO/IEC 21000-1, Information Technology - Multimedia Framework (MPEG-21) - Part 1: Vision, Technologies and Strategy.

[71] Bormans, J., Gelissen, J., and Perkis, A. (2003) MPEG-21: The 21st century multimedia framework. *IEEE Signal Processing Magazine*, **20**, 53–62.

[72] Vetro, A. and Timmerer, C. (2005) Digital item adaptation: overview of standardization and research activities. *IEEE Transactions on Multimedia*, **7**, 418–426.

[73] Amonou, I., Cammas, N., Kervadec, S., and Pateux, S. (2007) Optimized rate-distortion extraction with quality layers in the scalable extension of H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, **17**, 1186–1193.

[74] Mukherjee, D., Delfosse, E., Kim, J. G., and Wang, Y. (2005) Optimal adaptation decision-taking for terminal and network quality-of-service. *IEEE Transactions on Multimedia*, **7**, 454–462.

[75] Herranz, L. (2007) Integrating semantic analysis and scalable video coding for efficient content-based adaptation. *Multimedia Systems*, **13**, 103–118.

[76] De Schrijver, D., Poppe, C., Lerouge, S., De Neve, W., and Van de Walle, R. (2006) MPEG-21 bitstream syntax descriptions for scalable video codecs. *Multimedia Systems*, **11**, 403–421.

[77] Schrijver, D. D., Neve, W. D., Wolf, K. D., Sutter, R. D., and de Walle, R. V. (2007) An optimized MPEG-21 BSDL framework for the adaptation of scalable bitstreams. *Journal of Visual Communication and Image Representation*, **18**, 217 – 239.

[78] Van Deursen, D., Van Lancker, W., De Bruyne, S., De Neve, W., Mannens, E., and Van de Walle, R. (2010) Format-independent and metadata-driven media resource adaptation using semantic web technologies. *Multimedia Systems*, **16**, 85–104. 10.1007/s00530-009-0178-9.

[79] Arnaiz, L., Menendez, J. M., and Jimenez, D. (2011) Efficient personalized scalable video adaptation decision-taking engine based on MPEG-21. *IEEE Transactions on Consumer Electronics*, **57**, 763–770.

[80] Money, A. G. and Agius, H. (2008) Feasibility of personalized affective video summaries. *Affect and Emotion in Human-Computer Interaction: From Theory to Applications*, Lecture Notes in Computer Science, **4868**, pp. 194–208. Springer-Verlag, Berlin, Heidelberg.

[81] Herranz, L. and Martínez, J. M. (2010) A framework for scalable summarization of video. *IEEE Transactions on Circuits and Systems for Video Technology*, **20**, 1265–1270.

[82] Yuan, Z., Lu, T., Wu, D., Huang, Y., and Yu, H. (2011) Video summarization with semantic concept preservation. *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, pp. 109–112.

[83] Cong, Y., Yuan, J., and Luo, J. (2012) Towards scalable summarization of consumer videos via sparse dictionary selection. *IEEE Transactions on Multimedia*, **14**, 66–75.

[84] Zhu, Z. and Lin, T. (2010) Idempotent H.264 intraframe compression. *Multimedia Tools and Applications*, **46**, 25–45.

[85] Segall, A. and Zhao, J. (2008) Bit stream rewriting for SVC-to-AVC conversion. *Proceedings of International Conference on Image Processing*, 12-15, pp. 2776 –2779.

[86] De Cock, J., Notebaert, S., Lambert, P., and Van de Walle, R. (2008) Advanced bitstream rewriting from H.264/AVC to SVC. *Proceedings of International Conference on Image Processing*, 12-15, pp. 2472 –2475.