Real-Time BSD-driven Adaptation Along the Temporal Axis of H.264/AVC Bitstreams

Wesley De Neve, Davy De Schrijver, Davy Van Deursen, Peter Lambert, and Rik Van de Walle

Ghent University - IBBT
Department of Electronics and Information Systems - Multimedia Lab
Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium
{wesley.deneve,davy.deschrijver,davy.vandeursen,
peter.lambert,rik.vandewalle}@ugent.be
http://multimedialab.elis.ugent.be

Abstract. MPEG-21 BSDL offers a solution for exposing the structure of a binary media resource as an XML description, and for the generation of a tailored media resource using a transformed XML description. The main contribution of this paper is the introduction of a real-time work flow for the XML-driven adaptation of H.264/AVC bitstreams in the temporal domain. This real-time approach, which is in line with the vision of MPEG-21 BSDL, is made possible by two key technologies: BFlavor (BSDL + XFlavor) for the efficient generation of XML descriptions and Streaming Transformations for XML (STX) for the efficient transformation of these descriptions. Our work flow is validated in several applications, all using H.264/AVC bitstreams: the exploitation and emulation of temporal scalability, as well as the creation of video skims using key frame selection. Special attention is paid to the deployment of hierarchical B pictures and to the use of placeholder slices for synchronization purposes. Extensive performance data are also provided.

Key words: BSDL, H.264/AVC, STX, temporal scalability, video skims

1 Introduction

Video adaptation is an active area of interest for the research and standardization community [1]. The major purpose of a framework for video adaptation is to customize video resources such that the resulting bitstreams meet the constraints of a certain usage environment. This makes it possible to optimize the Quality of Experience (QoE) of the end-user. Several adaptation strategies can be identified, either operating at a semantic level (e.g., removal of violent scenes), at a structural level (e.g., picture dropping), or at a signal-processing level (e.g., coefficient dropping). In this paper, we introduce a real-time work flow for the structural adaptation of H.264/AVC bitstreams along their temporal axis, based on describing their high-level syntax in XML. Our approach enables applications such as the exploitation and emulation of temporal scalability in streaming scenarios, as well as the creation of video highlights in off-line use cases.

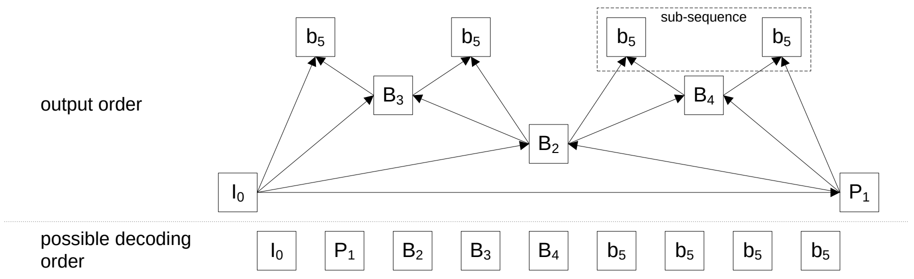


Fig. 1. The IbBbBbBbP coding pattern containing four sub-sequence layers

This paper is organized as follows. Section 2 introduces the two main enabling technologies, while Sect. 3 discusses several methods for the XML-driven extraction of bitstreams of multiple frame rates from a single coded H.264/AVC bitstream. Finally, Sect. 4 concludes this manuscript.

2 Enabling Technologies

2.1 Temporal Scalability in H.264/AVC

In video coding formats prior to H.264/AVC, temporal scalability is typically realized by the disposal of bidirectionally predicted pictures (B pictures). However, H.264/AVC only defines I, P, and B slices, and not I, P, and B pictures. Second, a coded picture can comprise a mixture of different types of slices. Finally, B slices can be used as a reference for the reconstruction of other slices [2].

Therefore, the recommended technique for achieving temporal scalability in H.264/AVC is to rely on the concept of sub-sequences [3] [4]. A sub-sequence represents a number of inter-dependent pictures that can be disposed without affecting the decoding of the remaining bitstream. In practice, these units of content adaptation are typically created by relying on a hierarchical coding pattern. This is a coding structure in which the use of reordering between picture decoding order and picture output order takes the form of building up a coarse-to-fine structuring of temporal dependencies. Nowadays, it is common to implement such a coding pattern using B slice coded pictures [5] (further referred to as hierarchical B pictures). However, hierarchical I slice or P slice coded pictures can be used as well (if coding efficiency is less important than encoding and decoding complexity), or a mix of the different slice and picture types.

An example coding pattern, offering four temporal levels, is shown in Fig. 1. A capital letter denotes a reference picture; a small letter a non-reference picture. Each picture is tagged with the value of `frame_num`. This syntax element acts as a counter that is incremented *after* the decoding of a reference picture, a functionality useful for the purpose of error concealment and content adaptation. Note that a hierarchical coding pattern is typically a good structure in terms of coding efficiency, but not in terms of end-to-end delay.

2.2 BSD-driven Content Adaptation

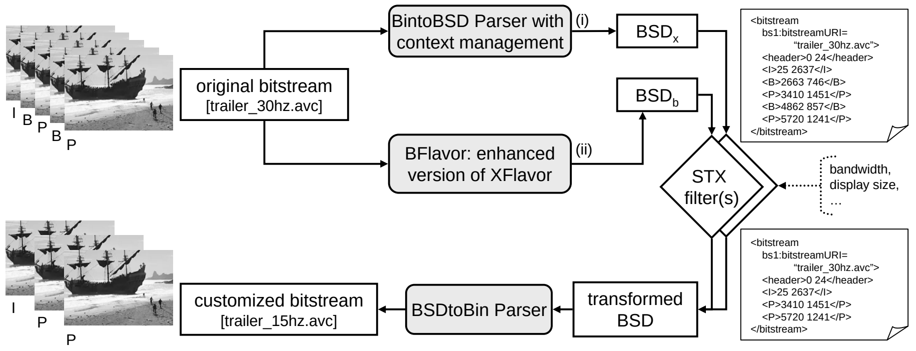
MPEG-21 BSDL The MPEG-21 Digital Item Adaptation (MPEG-21 DIA) standard addresses issues resulting from the desire to access multimedia content anywhere, anytime, and with any device. This concept is better known as Universal Multimedia Access (UMA). The MPEG-21 Bitstream Syntax Description Language (MPEG-21 BSDL) is a description tool that is part of MPEG-21 DIA. The language in question is a modification of W3C XML Schema to describe the (high-level) structure of a particular media format (file format, coding format) [6]. This results in a document called a Bitstream Syntax Schema (BS Schema). It contains the necessary information for exposing the structure of a binary media resource as an XML-based text document, called Bitstream Syntax Description (BSD), and for the creation of a tailored media resource using a transformed BSD. This media resource is then suited for playback in a particular usage environment, for instance constrained in terms of processing power.

In a BSDL-based content adaptation framework, the generation of a BSD is done by a format-neutral software module called BintoBSD Parser, while the adapted bitstream is constructed by a format-agnostic engine called BSDtoBin Parser. How to transform a BSD is not in the scope of MPEG-21 BSDL. In this paper, Streaming Transformations for XML (STX) are used for the transformation of BSDs (see further). The functioning of BintoBSD and BSDtoBin is guided by a BS Schema for a particular media format. As such, the BintoBSD and BSDtoBin Parsers constitute the pillars of a generic software framework for media format-unaware content adaptation, and of which the operation is entirely steered by XML-based technologies (e.g., XML, XML Schema, STX).

Finally, the main advantages of BSD-driven adaptation of binary media resources can be summarized as follows:

- the complexity of the content adaptation step is shifted from the compressed domain to the XML domain, allowing the reuse of standard XML tools (e.g., editors, transformation engines) and an integration with other XML-oriented metadata specifications (e.g., the MPEG-7 standard);
- the high-level nature of the BSDs allows to think about a media resource on how it is organized in terms of headers, packets, and layers of data;
- a *format-agnostic* content adaptation engine can be implemented (i.e., a combination of a BSD adaptation engine and BSDL's BSDtoBin Parser).

BFlavor The first version of the MPEG-21 BSDL specification is characterized by a number of performance issues with respect to the automatic generation of BSDs. Indeed, a format-agnostic BintoBSD Parser has to store the entire BSD in the system memory in order to support the at-run time evaluation of an arbitrary set of XPath 1.0 expressions. These XPath expressions are used to get access to XML-structured information that is already retrieved from a media resource, needed by a BintoBSD Parser for its decision-making while progressively parsing a bitstream. This behavior of BintoBSD results in an increasing memory usage



```
<bitstream
   bs1:bitstreamURI=
            “trailer_30hz.avc”>
   <header>0 24</header>
   <I>25 2637</I>
   <B>2663 746</B>
   <P>3410 1451</P>
   <B>4862 857</B>
   <P>5720 1241</P>
</bitstream>
```

```
<bitstream
   bs1:bitstreamURI=
            “trailer_30hz.avc”>
   <header>0 24</header>
   <I>25 2637</I>
   <P>3410 1451</P>
   <P>5720 1241</P>
</bitstream>
```

Fig. 2. BSD-driven media content adaptation, using BFlavor, STX, and BSDL

and a decreasing processing speed during the generation of an XML description for the high-level structure of a media resource.

Two different solutions were developed by the authors of this paper to address the performance issues of BSDL's BintoBSD process:

1. the first approach adds a number of new attributes to BSDL, allowing a BintoBSD Parser to keep the in-memory tree representation of a BSD minimal while still guaranteeing a correct output for the BintoBSD process [7];
2. the second solution consists of the development of a new description tool for translating the structure of a binary media resource into an XML description, called BFlavor (BSDL + XFlavor) [8] [9].

BFlavor is the result of a modification of XFlavor to efficiently support BSDL features. It allows to describe the structure of a media resource in an object-oriented manner, after which it is possible to automatically create a BS Schema, as well as a code base for a format-specific parser. This automatically generated parser is subsequently able to generate BSDs that are compliant with the automatically generated BS Schema. As such, this implies that the resulting BSDs can be further processed by the upstream tools in a BSDL-based adaptation chain, such as a format-neutral BSDtoBin Parser.

Fig. 2 provides a high-level overview of our XML-based content adaptation chain. It illustrates the two different approaches for creating BSDL-compliant BSDs: (1) by relying on an optimized BintoBSD Parser, using our extensions to BSDL; (2) using a BFlavor-based parser. The transformation of the BSDs is done by relying on STX while the adapted bitstreams are constructed using BSDL's BSDtoBin Parser.

3 Bitstream Extraction in H.264/AVC

In this section, a few experiments are discussed that were set up to evaluate the expressive power and performance of the XML-driven content adaptation chain

Table 1. Bitstream characteristics for The New World movie trailer

ID	coding pattern	frame rate (Hz)	resolution	#slices/ picture	#NALUs[a]	duration (s)	size$_o$ (MB)
TNW$_1$	IbBbBbBbP	23.98	848x352	5	17808	148	42.9
TNW$_2$	IbBbBbBbP	23.98	1280x544	5	17808	148	78.7
TNW$_3$	IbBbBbBbP	23.98	1904x800	5	17808	148	126.0

[a] NALU stands for Network Abstraction Layer Unit; size$_o$ stands for original file size.

Table 2. BSD generation using an optimized BintoBSD Parser and BFlavor

ID	BintoBSD$_m$ Parser				BFlavor			
	throughput (NALU/s)	MC[a] (MB)	BSD (MB)	BSD$_c$ (KB)	throughput (NALU/s)	MC (MB)	BSD (MB)	BSD$_c$ (KB)
TNW$_1$	124	1.7	44.3	326	1164	0.7	28.9	308
TNW$_2$	110	1.7	44.2	335	777	0.7	29.0	317
TNW$_3$	97	1.7	44.3	332	533	0.7	29.0	314

[a] MC stands for peak heap Memory Consumption; BSD$_c$ for compressed BSD size.

as proposed in Fig. 2. The focus is hereby put on the real-time adaptation of H.264/AVC bitstreams along the temporal axis. The media resources involved are three different versions of the same movie trailer, called The New World[1]. The performance analysis was done by breaking up the XML-driven content adaptation chain in its three fundamental building blocks: BSD generation, BSD transformation, and bitstream construction. Real-time means that every building block, typically running in a pipelined fashion on different processing nodes, is able to achieve a throughput that is at least as fast as the playback speed of the original media resource.

The most important properties of the bitstreams used, encoded with the H.264/AVC reference software (JM 10.2), are shown in Table 1. The coding pattern employed is visualized by Fig. 1. The results were obtained on a PC with an Intel Pentium IV 2.61 GHz CPU and 512 MB of memory. All time measurements were done 11 times, after which an average was taken of the last 10 runs in order to take into account the startup latency. BSDs were compressed using WinRAR 3.0's default text compression algorithm. The anatomy of the H.264/AVC bitstreams was described up to and including the syntax elements of the slice headers, once in MPEG-21 BSDL and once in BFlavor.

3.1 BSD Generation

Table 2 summarizes the results obtained during the generation of BSDs for the bitstreams involved. The BFlavor-based parser outperforms our optimized BintoBSD Parser on all metrics applied: the parser is faster than real-time for all bitstreams used (i.e., its throughput is always higher than the playback speed of 23.98 x 5 NALUs/s or 120 NALU/s) and is characterized by a very low memory footprint. The BFlavor-driven parser also produces textual BSDs that are much

[1] Online available at: `http://www.apple.com/trailers/`.

Table 3. BSD transformation using STX and tailored bitstream construction using BSDL's format-neutral BSDtoBin Parser

ID	operation	BSD transformation				bitstream reconstruction		
		throughput (NALUs/s)	MC (MB)	BSD (MB)	BSD_c (KB)	throughput (NALUs/s)	MC (MB)	$size_a$ (MB)
TNW_1	remove EL[a] 3	835	1.2	21.5	159.0	406.2	2.0	98.4
TNW_2	remove EL 2 + 3	980	1.2	11.0	81.0	361.2	2.3	65.2
TNW_3	remove EL 1 + 2 + 3	1098	1.3	5.8	41.0	264.3	2.2	38.6
TNW_1	replace EL 3	537	1.7	36.9	194.0	515.2	2.1	98.5
TNW_2	replace EL 2 + 3	445	1.7	33.5	121.0	554.7	2.3	65.3
TNW_3	replace EL 1 + 2 + 3	447	1.3	33.1	84.6	537.2	2.2	38.8

[a] EL stands for enhancement layer.

smaller than those created by the BintoBSD Parser. This is due to the design of our manually created BS Schema (used by the BintoBSD and BSDtoBin Parser): it is less optimized than BFlavor's automatically generated BS Schema (only used by a BSDtoBin Parser) for the purpose of readability.

3.2 BSD Tranformation and Bitstream Reconstruction

The transformation of the BSDs was done using Streaming Transformations for XML (STX)[2]. This transformation language is intended as a high-speed, low memory consumption alternative to XSLT as it does not require the construction of an in-memory tree. As such, STX is suitable for the transformation of large XML documents with a repetitive structure, which are typical characteristics for BSDs describing the high-level structure of compressed video bitstreams. Indeed, several publications have shown that XSLT, as well as a hybrid combination of STX/XSLT, are unusable in the context of XML-driven video adaptation, due to a respective high memory consumption and high implementation overhead [10].

A number of STX stylesheets were implemented in the context of this research, dependent on the targeted use case. In what follows, the semantics and performance of the different transformation steps are outlined in more detail.

Exploiting Temporal Scalability by Dropping Slices. A first STX stylesheet was written to drop the different temporal enhancement layers as visualized in Fig. 1. The decision-making process was implemented by checking the values of the following syntax elements: `nal_ref_idc`, `slice_type`, and `frame_num`. The value of `gaps_in_frame_num_value_allowed_flag` in the Sequence Parameter Set (SPS) was modified to one, signaling to a decoder that reference pictures were intentionally dropped. As shown in the upper half of Table 3, the implementation of the removal operations, at the level of a BSD, can be done very efficiently in terms of processing time and memory consumption needed. The STX engine used was the Joost STX processor (version 2005-05-21).

[2] Online available at `http://stx.sourceforge.net/`.

Emulation of Temporal Scalability Using Placeholder Slices. In the context of digital video coding, it is important to separate the concept of what is encoded in the bitstream, which is essentially a compact set of instructions to tell a decoder how to decode the video data, from the concept of what is the decision-making process of an encoder. The latter process is not described in a video coding standard, since it is not relevant to achieving interoperability. Consequently, an encoder has a large amount of freedom about how to decide what to tell a decoder to do. This freedom can also be exploited by a content adaptation engine to offer a solution for resynchronization issues that may occur after the adaptation of an elementary bitstream in the temporal domain.

The traditional view of temporal scalability is to remove certain coded pictures from a bitstream while still obtaining a decodable remaining sequence of pictures. This approach is typically applied when using BSD-driven bitstream thinning. However, a major drawback of this method is that it fails when, for instance, the remaining pictures are to be resynchronized with an audio stream.

Elementary bitstreams usually do not convey (absolute) timing information as this responsibility is typically assigned to the systems layer (e.g., file formats, network protocols), and not to the coding layer. Consequently, after having dropped certain pictures in a bitstream, it is often impossible to synchronize the remaining pictures with a corresponding audio stream without an external knowledge, an observation that is especially true when varying coding patterns are in use. Therefore, we propose to exploit temporal scalability in elementary video bitstreams by replacing coded pictures with placeholder pictures, a technique that operates at the same level as BSDL, i.e. at the coding layer [10].

A placeholder or dummy picture is defined as a picture that is identical to a particular reference picture, or that is constructed by relying on a well-defined interpolation process between different reference pictures. Therefore, only a limited amount of information needs to be transmitted to signal placeholder pictures to a decoder. Placeholder pictures are used to fill up the gaps that are created in a bitstream due to the disposal of certain pictures, a technique that is further referred to as the emulation of temporal scalability. This approach makes it straightforward to maintain synchronization with other media streams in a particular container format, especially when a varying coding structure is in use because the total number of pictures remains the same after the adaptation step. As such, from the bitstream's point of view, emulating temporal scalability can be considered a substitution operation, and not a removal operation.

Several STX stylesheets were developed to translate the B slices in the temporal enhancement layers of the H.264/AVC bitstreams to skipped B slices and skipped P slices (see Fig. 3).

– A picture consisting of skipped B slices tells an H.264/AVC decoder to reconstruct the picture by doing an interpolation between the previous picture and the next picture in output order[3].

[3] The interpolated picture is computed based on the relative temporal positions of the list 0 and list 1 (decoded) reference pictures.

```
<stx:group name="BtoskippedP">
   <stx:template match="jvt:coded_slice_of_a_non_IDR_picture" public="no">
      <stx:element name="coded_slice_of_a_skipped_non_IDR_picture" namespace="h264_avc">
         <stx:process-children group="BtoskippedP"/>
      </stx:element>
   </stx:template>
   <stx:template match="jvt:slice_layer_without_partitioning_rbsp" public="no">
      <stx:element name="skipped_slice_layer_without_partitioning_rbsp" namespace="h264_avc">
         <stx:process-children group="BtoskippedP"/>
      </stx:element>
   </stx:template>
   <stx:template match="jvt:slice_type" public="no">
      <stx:element name="slice_type" namespace="h264_avc">0</stx:element>
   </stx:template>
   <stx:template match="jvt:slice_qp_delta" public="no">
      <stx:element name="slice_qp_delta" namespace="h264_avc">0</stx:element>
   </stx:template>
   <stx:template match="jvt:if_slice_type_eq_B" public="no"/>
   <stx:template match="jvt:slice_data" public="no">
      <stx:element name="skipped_slice_data" namespace="h264_avc">
         <stx:element name="mb_skip_run" namespace="h264_avc">234</stx:element>
         <stx:element name="rbsp_trailing_bits" namespace="h264_avc">
            <stx:element name="rbsp_stop_one_bit" namespace="h264_avc">1</stx:element>
            <stx:element name="rbsp_alignment_zero_bit" namespace="h264_avc">0</stx:element>
         </stx:element>
      </stx:element>
   </stx:template>
</stx:group>
```

Fig. 3. Extract of the STX stylesheet for translating B slices to skipped P slices in the XML domain. Similar logic is used for translating I and P slices to skipped P slices

– A picture consisting of skipped P slices instructs a decoder to output the last picture in the decoded picture buffer[4].

Skipped B slices were used as a substitute for the B slices in the third enhancement layer when only this layer is to be removed; the use of skipped P slices would lead to a wrong output order (i.e., $I_0B_4B_3B_4B_2B_4B_4B_4P_1$), due to the fact that B_4 is the last picture in the decoded picture buffer. Skipped P slices were used as a substitute for all B slices when at least two enhancement layers are replaced; a correct output order can be obtained then (e.g., $I_0B_2B_2B_2B_2B_2B_2B_2P_1$ when dropping two enhancement layers).

Performance results are provided in the lower half of Table 3. It is clear that the translation operations, which are entirely expressed in the XML domain, can be executed in real time. The same observation is true for the behaviour of BSDL's BSDtoBin Parser [11]. The overhead of the skipped slices in the resulting bitstreams can be ignored, as one can notice in the column with label size_a.

Video Skims by Key Frame Selection. Finally, our XML-driven content adaptation approach was also used for the production of video skims. These compact abstractions of long video sequences are typically created by filtering out relevant pictures, e.g. key pictures that are located near the beginning of a shot. Therefore, a STX stylesheet was implemented that takes as input the shot detection information as produced by the IBM MPEG-7 Annotation Tool[5] , and that subsequently identifies and marks the I slice coded pictures located near

[4] This is, the first (decoded) reference picture in list 0.

[5] Online available at: `http://www.alphaworks.ibm.com/tech/videoannex`.

```xml
<bitstream xmlns="h264_avc" xmlns:jvt="h264_avc" bitstreamURI="the_new_world_h480p_IbBbBbBb.h264">
   <byte_stream>
      <byte_stream_nal_unit pic_cnt="0" shot="false">
         <!-- Sequence Parameter Set -->
      </byte_stream_nal_unit>
      <byte_stream_nal_unit pic_cnt="0" shot="false">
         <!-- Picture Parameter Set -->
      </byte_stream_nal_unit>
      <byte_stream_nal_unit pic_cnt="1" shot="true">
         <!-- First coded slice of I_0 (an IDR picture) -->
      </byte_stream_nal_unit>
      <byte_stream_nal_unit pic_cnt="2" shot="false">
         <!-- First coded slice of I_1 (a non-IDR picture) -->
      </byte_stream_nal_unit>
      <byte_stream_nal_unit pic_cnt="3" shot="false">
         <!-- First coded slice of B_2 (a non-IDR picture) -->
      </byte_stream_nal_unit>
      <!-- Remaining byte stream NALUs in decoding order -->
   </byte_stream>
</bitstream>
```

Fig. 4. Embedding shot information as additional attributes in a BSD

the start of a shot. More precisely, the information about the different shots is embedded by the STX stylesheet as additional attributes in a BSD (see Fig. 4). The resulting BSD is then provided as input to a next STX stylesheet; it filters out the relevant I slice coded pictures and translates all remaining I and B slices to skipped P slices to maintain synchronization with the original audio stream. Note that the IbBbBbBb coding pattern was used instead of IbBbBbBbP, offering random access at regular picture intervals as every picture in the base layer is encoded as an I slice coded picture. The summary of the video bitstream also results in a significant reduction of its file size: from 44.7 MB to 4.40 MB when TNW_1 is used with the IbBbBbBb pattern. This technique may be of particular interest for the repurposing of content for constrained usage environments.

4 Conclusions

This paper introduced a real-time work flow for the description-driven adaptation of H.264/AVC bitstreams along their temporal axis. The key technologies used were BFlavor for the generation of BSDs, STX for the transformation of BSDs, and BSDL's format-neutral BSDtoBin Parser for the construction of tailored bitstreams. Our approach was validated in several use cases: the exploitation of temporal scalability by dropping certain slices; the emulation of temporal scalability by relying on skipped slices; and the creation of video skims. The use of video skims, new in the context of BSD-based video adaptation, is made possible by enriching a BSD with additional metadata to steer the BSD adaptation process. As an example, an overall pipelined throughput of at least 447 NALUs/s was achieved when emulating temporal scalability in a high-definition H.264/AVC bitstream by substituting all slices in the enhancement layers by skipped P slices, together with a combined memory use of less than 5 MB.

A remaining bottleneck in this content adaptation system is the size of the textual BSDs. Further research will also concentrate on shifting the focus of BSD-driven content adaptation from a structural level to a semantic level.

Acknowledgments. The research activities that have been described in this paper were funded by Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), the Belgian Federal Science Policy Office (BFSPO), and the European Union.

References

1. Chang, S.-F., Vetro, A.: Video Adaptation: Concepts, Technology, and Open Issues. Proc. the IEEE 93 (1) (2005) 145-158
2. Sullivan, G.J., Wiegand, T.: Video Compression - From Concepts to the H.264/AVC Standard. Proc. the IEEE 93 (1) (2005) 18-31
3. Tian, D., Hannuksela, M., Gabbouj, M.: Sub-sequence Video Coding for Improved Temporal Scalability. Proceedings 2005 IEEE International Symposium on Circuits and Systems (ISCAS 2005), pages 6074-6077, Kobe, Japan, May 2005
4. De Neve, W., Van Deursen, D., De Schrijver, D., De Wolf, K., Van de Walle, R.: Using Bitstream Structure Descriptions for the Exploitation of Multi-layered Temporal Scalability in H.264/AVC's Base Specification. Lecture Notes in Computer Science, Volume 3767, pages 641-652, Oct 2005
5. Schwarz, H., Marpe, D., Wiegand, T.: Analysis of Hierarchical B Pictures and MCTF. Proceedings 2006 International Conference on Multimedia & Expo (ICME 2006), Toronto, Canada, July 2006
6. Panis, G., Hutter, A., Heuer, J., Hellwagner, H., Kosch, H., Timmerer, T., Devillers, S., Amielh, M.: Bitstream Syntax Description: A Tool for Multimedia Resource Adaptation within MPEG-21. Signal Processing: Image Communication **18** (2003) 721-747
7. De Schrijver, D., De Neve, W., De Wolf, K., Van de Walle, R.: Generating MPEG-21 BSDL Descriptions Using Context-Related Attributes. Proceedings of the 7th IEEE International Symposium on Multimedia (ISM 2005), pages 79-86, USA, December 2005
8. Van Deursen, D., De Neve, W., De Schrijver, D., Van de Walle, R.: BFlavor: an Optimized XML-based Framework for Multimedia Content Customization. Proceedings of the 25th Picture Coding Symposium (PCS 2006), 6 pp on CD-ROM, Beijing, China, April 2006
9. De Neve, W., Van Deursen, D., De Schrijver, D., De Wolf, K., Lerouge, S., Van de Walle, R.: BFlavor: a harmonized approach to media resource adaptation, inspired by MPEG-21 BSDL and XFlavor. Accepted for publication in EURASIP Signal Processing: Image Communication, Elsevier.
10. De Neve, W., De Schrijver, D., Van de Walle, D., Lambert, P., Van de Walle, R.: Description-Based Substitution Methods for Emulating Temporal Scalability in State-of-the-Art Video Coding Formats. Proceedings of the 7th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2006), pages 83-86, Incheon, Korea, 2006
11. Devillers, S., Timmerer, C., Heuer, J., Hellwagner, H.: Bitstream Syntax Description-Based Adaptation in Streaming and Constrained Environments. IEEE Trans. Multimedia 7 (3) (2005) 463-470