Using Bitstream Structure Descriptions for the Exploitation of Multi-layered Temporal Scalability in H.264/AVC's Base Specification

Wesley De Neve[1], Davy Van Deursen[1], Davy De Schrijver[1], Koen De Wolf[1], and Rik Van de Walle[2]

[1] Ghent University - IBBT, Multimedia Lab, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
Wesley.DeNeve@UGent.be
http://multimedialab.elis.ugent.be

[2] Ghent University - IBBT - IMEC, Multimedia Lab, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium

Abstract. In this paper, attention is paid to the automatic generation of XML-based descriptions containing information about the high-level structure of binary multimedia resources. These structural metadata can then be transformed in order to reflect a desired adaptation of a multimedia resource, and can subsequently be used to create a tailored version of the resource in question. Based on this concept, two technologies are presented: MPEG-21 BSDL and a modified version of XFlavor being able to create BSDL compatible output. Their usage is elaborated in more detail with respect to the valid exploitation of multi-layered temporal scalability in H.264/MPEG-4 AVC's base specification, and in particular with a focus on a combined usage of the sub-sequence coding technique and Supplemental Enhancement Information (SEI) messages. Some performance measurements in terms of file sizes and computational times are presented as well.

1 Introduction

Scalable video coding is a major point of interest in the community of digital video coding. The technology in question is supposed to pave the way for the deployment of several new multimedia architectures. The latter should make it possible to tackle the tremendous diversity in terminals and networks as used in the present-day and future multimedia ecosystem. It is important to be aware of the fact that an efficient solution for this heterogeneity does not only imply the usage of scalable video coding, but also requires the usage of a complementary adaptation decision taking engine and a complementary content adaptation system [1]. Hence, a question that arises is how to optimally customize a scalable bitstream according to a given set of constraints (e.g., device and network characteristics, user preferences, natural environment). A solution for the content adaptation problem, based on the description of the high-level structure of compressed bitstreams in the eXtensible Markup Language (XML), is elaborated in more detail in this paper.

The outline of the paper is as follows: after having given an overview of the temporal scalability features in H.264/AVC's base specification in Section 2, a short description of two bitstream structure description languages is provided in Section 3, as well as a discussion on how they can be used to customize (scalable) bitstreams. Section 4 discusses some performance results as obtained in the context of H.264/AVC while Section 5 concludes this paper.

2 Temporal Scalability in H.264/MPEG-4 AVC

2.1 Context

ITU-T H.264 / MPEG-4 (Part 10) Advanced Video Coding (commonly abbreviated as H.264/AVC) is a new and standardized specification for digital video coding, characterized by a design that targets efficiency, robustness, and usability. Because of its support for a wide range of bit rates, H.264/AVC can even be considered as a universal standard for digital video coding. It is expected that H.264/AVC will have a powerful impact on both consumer and professional video applications in the years to come, especially when taking into account the tools that were added to the standard in the course of 2004 and that are known as the Fidelity Range Extensions (FRExt) [2]. Hence, there is a good chance that H.264/AVC will be used in very diverse usage environments, thus making it relevant to gain an insight into the tools this standard makes available with respect to bitstream customization.

In the first version of the H.264/AVC specification, as approved in the spring of 2003, there are only tools available for enabling multiple representations of the same content and for enabling multi-layered temporal scalability. The latter can be defined as the ability to remove some coded pictures from a bitstream while still obtaining a decodable remaining sequence of pictures (frame dropping or stream thinning). The most important tools in this context are switching or synchronization slices and Supplemental Enhancement Information messages (SEI messages) for signaling the appearance of sub-sequences and sub-sequence layers. Switching slices make it possible to switch between alternate representations of the same video content (simulstore), to recover from data losses or errors, and to apply trick modes such as fast-forward and fast-reverse. Switching slices will not be discussed in further detail in this paper.

The sub-sequence coding technique will be explained in more detail in the next section, as well as some closely related concepts, such as pyramid encoding and explicit Group Of Pictures structures (GOP structures). To conclude this section, it is worth noting that work is currently going on in order to address the need for more advanced scalability tools in H.264/AVC (e.g., to allow spatial, and fine and coarse grain quality scalability). This activity, again conducted by the Joint Video Team (JVT), is known as Scalable Video Coding (SVC) and will most probably result in a second amendment to the H.264/AVC standard.

2.2 Sub-sequences: Background and Signaling

Before providing more details pertaining to the sub-sequence coding technique, it is interesting to have a closer look at some design features of H.264/AVC, especially when taking into account the fact that temporal scalability is often realized in its predecessors by the disposal of bidirectionally predicted pictures.

First, in H.264/AVC there is no such thing as an I picture, a P picture, or a B picture since the recommendation only defines I slices, P slices, and B slices. In addition, it is allowed to construct a coded picture that consists of a mixture of different types of slices. Second, B slices can be used as a reference for the reconstruction of other slices, a concept known as generalized B slices. Hence, it should be clear that for exploiting temporal scalability in H.264/AVC or for enabling fast forward operations, a subtle approach is needed in order to obtain a bitstream that is valid in terms of its syntax and semantics (e.g., correct reference picture management).

The recommended way for achieving temporal scalability in H.264/AVC is to make use of the sub-sequence coding technique. The latter was introduced by Hannuksela as the enhanced concept of a GOP [4]. A sub-sequence represents a number of inter-dependent pictures that can be disposed without affecting the decoding of any other sub-sequence in the same sub-sequence layer or any sub-sequence in any lower sub-sequence layer. It is hereby possible to assign coded pictures in a bitstream to sub-sequences and sub-sequence layers in multiple ways, provided that the structure fulfills the requirements for dependencies between sub-sequences. Typically, each picture will belong to exactly one sub-sequence, and each sub-sequence will belong to exactly one sub-sequence layer in any sub-sequence structure. In short, a sub-sequence layer contains a subset of the coded pictures in a sequence while a sub-sequence is a set of coded pictures within a sub-sequence layer. An example will be provided in Subsection 4.2.

In order to signal the appearance of sub-sequences, sub-sequence layers, and their dependencies, it is possible to insert SEI messages in the compressed video data. SEI messages, introduced for the first time in H.263+, can assist in processes related to decoding, display or other purposes. However, SEI messages are not required for constructing the luma or chroma samples by the decoding process. Three types of SEI messages are defined for sub-sequences. The sub-sequence information SEI message maps a coded picture to a certain sub-sequence and sub-sequence layer. The sub-sequence layer characteristics SEI message and the sub-sequence characteristics SEI message provide statistical information on the indicated sub-sequence layer and sub-sequence respectively (e.g., the number of sub-sequence layers, the average frame and bit rate). Furthermore, the dependencies between sub-sequences are indicated in the sub-sequence characteristics SEI message. Hence, bitstream extraction tools and decoders can use this metadata to scale down a bitstream in the temporal domain or to implement a feature such as a fast-forward mode, without having to delve deeply into the syntax of a compressed bitstream. It is interesting to know that similar adaptation hints are currently standardized in the context of the activities with respect to H.264/AVC SVC.

In practice, sub-sequences will most probably be created by relying on pyramid encoding or an explicit GOP structure. Pyramid encoding organizes the coded pictures of a bitstream in several layers of data dependencies, hereby making use of the following rule of thumb: the layers are ordered hierarchically based on their dependency on each other such that any picture in a layer shall not be predicted from any picture on any higher layer. When making use of this technique, the pictures in the enhancement layers are typically (hierarchically) B slice coded pictures [3]. The latter constraint is weakened when making use of an explicit GOP structure. Such a structure not only makes it possible to specify explicitly the type of slice to use but also makes it possible to specify the layer that will contain the slice. For instance, the explicit GOP structure coding technique allows to add an I slice coded picture to an enhancement layer.

2.3 Syntax Considerations: A View from the Trenches

Several syntax elements can be important when exploiting temporal scalability in the base version of H.264/AVC, such as the `nal_ref_idc`, `frame_num`, `gaps_in_frame_num_value_allowed_flag`, `fixed_frame_rate_flag`, and `slice-_type` syntax elements, as well as the syntax elements in the sub-sequence related SEI messages. For instance, the sub-sequence information SEI message shall not be present unless `gaps_in_frame_num_value_allowed_flag`, as available in the Sequence Parameter Set (SPS) referenced by the picture associated with the sub-sequence SEI message, is equal to one. The latter is necessary in order to allow the intentional disposal of slices that are used for the reconstruction for other slices, a scenario that is likely to occur when dealing with multi-layered temporal scalability. Otherwise, the decoder might invoke error concealment procedures due to the fact that the syntax element `frame_num` is a way to achieve picture loss robustness: a gap in its value indicates a missing reference slice.

The fact that `frame_num` is primarily a loss robustness feature, is reflected in the way that the behavior of `frame_num` depends on whether the picture is a reference picture or not (i.e., on the value of `nal_ref_idc`): `frame_num` acts as a counter that increments each time a reference picture is decoded. As such, it is possible for a decoder to detect that some picture(s) are missing and to conceal the problem without losing track of what is going on. Since the proper decoding of a non-reference picture is not necessary for the proper decoding of other pictures that arrive later, `frame_num` was designed so that a missing non-reference picture would not cause `frame_num` to indicate the presence of a problem when a non-reference picture is missing[1]. Note that the value of `frame_num` is reset to zero whenever a new coded video sequence begins.

The syntax element `sub_seq_frame_num`, as available in a sub-sequence information SEI message, is a variant of the syntax element `frame_num`. The behavior is the same: it acts as a counter that is only incremented when the associated picture is used as a reference. However, the difference lies in the fact that `sub_seq_frame_num` belongs to one sub-sequence and not to a coded video

[1] A better name for `frame_num` would probably have been `ref_pic_num`.

sequence. Hence, the value of `sub_seq_frame_num` is reset to zero whenever a new sub-sequence begins. As such, it is still possible for a decoder to detect the loss of a reference slice when gaps are allowed in the value of `frame_num`. Further, it is also interesting to know that the first syntax element of a sub-sequence information SEI message (i.e., `sub_seq_layer_num`) represents the number of a sub-sequence layer, while the second syntax element (i.e., `sub_seq_id`) identifies a sub-sequence within a particular sub-sequence layer.

With respect to the semantics of the `slice_type` syntax element, it is relevant to be aware of the fact that a value in the range 5–9 specifies, in addition to the coding type of the current slice, that all other slices of the current coded picture shall have the same type.

3 Bitstream Structure Description Languages

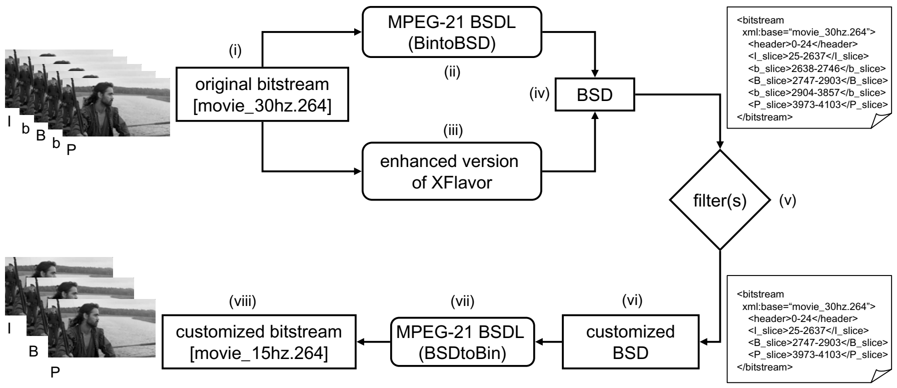
3.1 Context

In order to be able to deliver scalable video in a heterogeneous environment, it is important to be aware of the need of complementary logic that makes it possible to exploit the scalability properties of the parent bitstream. This bitstream extraction process typically involves the removal of certain data blocks and the modification of the value of certain syntax elements.

One way to realize the scenario as just mentioned, is to rely on automatically generated XML-based descriptions that contain information about the high-level structure of scalable bitstreams. These structural metadata can subsequently be transformed (an operation in the semantic domain) in order to reflect a desired adaptation of a scalable bitstream, and can then be used to automatically create an adapted version of the bitstream in question (an operation in the compressed domain). In other words, the Bitstream Structure Descriptions (BSDs[2]) can be used as an intermediate format to customize scalable bitstreams (without requiring a recode of the compressed video data). As such, the BSDs act as an abstraction of the compressed bitstream since their high-level nature only requires a limited knowledge about the bitstream structure: one no longer has to reason about the bitstream in terms of motion vectors or transform coefficients, but one can think in terms of layers and packets. Moreover, the XML-based formalism also allows the usage of many already existing tools for manipulating XML documents, as well as a straightforward integration with other metadata standards such as MPEG-7.

The structural metadata also enable other applications. For instance, one can think of correcting wrong coded syntax elements without the need of a recode or recompilation of the media data: e.g., correcting aspect ratio information, correcting four-character codes in file containers (FourCCs), ... Such operations are sometimes called header hacks. Further possible applications with regard to BSDs are multiplexing and demultiplexing, automatic video summarization, scene selection, and bitstream syntax validation.

[2] In MPEG-21 terminology, a BSD stands for Bitstream Syntax Description.



```xml
<bitstream
  xml:base="movie_30hz.264">
    <header>0-24</header>
    <I_slice>25-2637</I_slice>
    <b_slice>2638-2746</b_slice>
    <B_slice>2747-2903</B_slice>
    <b_slice>2904-3857</b_slice>
    <P_slice>3973-4103</P_slice>
</bitstream>
```

```xml
<bitstream
  xml:base="movie_30hz.264">
    <header>0-24</header>
    <I_slice>25-2637</I_slice>
    <B_slice>2747-2903</B_slice>
    <P_slice>3973-4103</P_slice>
</bitstream>
```

Fig. 1. High-level representation of the joint MPEG-21 BSDL/XFlavor approach

```xml
<xsd:element name="seq_parameter_set_rbsp">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element name="profile_idc" type="xsd:unsignedByte"/>
   <xsd:element name="constraint_set0_flag" type="bt:b1"/>
   <xsd:element name="constraint_set1_flag" type="bt:b1"/>
   <xsd:element name="constraint_set2_flag" type="bt:b1"/>
   <xsd:element name="reserved_zero_5bits" type="bt:b5"
                   fixed="0"/>
   <xsd:element name="level_idc" type="xsd:unsignedByte"/>
   <xsd:element name="seq_parameter_set_id"
                   type="jvt:UnsignedExpGolomb"/>
   <xsd:element name="log2_max_frame_num_minus4"
                   type="jvt:UnsignedExpGolomb"/>
   <xsd:element name="pic_order_cnt_type"
                   type="jvt:UnsignedExpGolomb"/>
   <xsd:element name="if_pic_order_cnt_type_eq_0" minOccurs="0"
                   bs2:if="./jvt:pic_order_cnt_type = 0">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="log2_max_pic_order_cnt_lsb_minus4"
                 type="jvt:UnsignedExpGolomb"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
   <!-- ... -->
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

```
class Seq_parameter_set_rbsp {


  bit(8) profile_idc;
  bit(1) constraint_set0_flag;
  bit(1) constraint_set1_flag;
  bit(1) constraint_set2_flag;
  bit(5) reserved_zero_5bits = 0;

  bit(8) level_idc;
  ue seq_parameter_set_id;

  ue log2_max_frame_num_minus4;

  ue pic_order_cnt_type;

  if ( pic_order_cnt_type.value == 0 )



    ue log2_max_pic_order_cnt_lsb_minus4;



  /* ... */



}
```

Fig. 2. Partial description of the SPS syntax in BSDL (left) and XFlavor (right)

In the following sections, a short overview is given of two languages that provide solutions for discovering the structure of a multimedia resource in order to generate its XML description and for the generation of an adapted multimedia resource using a transformed description. To be more specific, more details will be provided with respect to MPEG-21 BSDL (MPEG-21 Bitstream Syntax Description Language) and a modified version of Flavor/XFlavor (Formal Language for Audio-Visual Object Representation) that is able to output BSDL compatible descriptions. A profound comparison between the languages as just mentioned will be provided in a forthcoming paper, as well as a discussion of the merits of the enhanced version of XFlavor. Figure 1 shows a high-level overview of our harmonized MPEG-21 BSDL/XFlavor approach. The different steps in the operational flow are as follows: (i) the parent H.264/AVC bitstream; (ii) a BSD is created by making use of MPEG-21 BSDL; (iii) a BSD is created by making use of the enhanced version of XFlavor; (iv) MPEG-21 BSDL and the modified version of XFlavor allow to create an equivalent BSD (but not necessarily an identical one); (v) filter(s) for customizing a BSD in order to meet the constraints of a certain usage environment; (vi) the customized BSD; (vii) MPEG-21 BSDL allows to create an adapted bitstream, guided by the customized BSD (bitstream extraction); (viii) the tailored child H.264/AVC bitstream. A partial description of H.264/AVC's Sequence Parameter Set (SPS) datastructure in MPEG-21 BSDL and XFlavor is illustrated by Figure 2. Such descriptions are used by the MPEG-21 BSDL and enhanced XFlavor tool chain in order to automatically create a BSD for an arbitrary H.264/AVC bitstream. Note that MPEG-21 BSDL is built on top of W3C XML Schema (the metadata community's point of view) while XFlavor is built on top of the principles of object oriented languages such as C++ and Java (the developers community's point of view).

3.2 The MPEG-21 Bitstream Syntax Description Language

MPEG-21 BSDL is a language that enables the (partial) description of the syntax of (scalable) bitstreams. The technology, built on top of W3C XML Schema, was created by Philips Research, France [6]. The primary motivation behind its development is to assist in customizing scalable bitstreams. In order to avoid a large overhead and unnecessary computations, the language in question will most often only be used for the description of the high-level structure of a bitstream. BSDL falls under the umbrella of the Bitstream Syntax Description tool of the MPEG-21 Multimedia Framework, just like the gBS Schema language [8]. The MPEG-21 standard attempts to realize the ideal of easily exchanging any type of information without technical barriers.

The generic character of the BSDL technology lies in the format independent nature of the logic responsible for the creation of the BSDs and for the generation of the adapted bitstreams. To be more specific, it is not necessary to update the different pieces of software involved in order to support a new (scalable) video coding format since all information necessary for discovering the structure of the bitstream is available in the BSDL description of (a part of) the syntax of the coding format. As such, BSDL allows to construct a universal adaptation engine.

With respect to the first version of H.264/AVC, a generic BSDL schema was developed that allows to describe its Annex B syntax up to and including the slice header datastructure (independent of the profile@level combination used) [9].

3.3 The Formal Language for Audio-Visual Object Representation

Flavor, developed by Columbia University, was initially designed as a declarative language with a C++-like syntax to describe the bitstream syntax on a bit-per-bit basis. Its aim is to simplify and speed up the development of software that processes audiovisual bitstreams by automatically generating the required

C++ and Java code to parse the data, hence allowing the developer to concentrate on the processing part of the software. Flavor was enhanced to support XML features (XFlavor), resulting in the development of tools for generating an XML description of the bitstream syntax and for regenerating an adapted bitstream [7]. As discussed in [8], there are, however, some fundamental differences to BSDL, stemming mainly from the original focus of the two technologies. For instance, in XFlavor, the complete bitstream data are actually embedded in the BSD, resulting in potentially huge descriptions, while BSDL uses a specific datatype to point to a data range in the original bitstream when it is too verbose to be included in the description. This is why, unlike XFlavor, BSDL is rather a description language than a representation language, and can describe a bitstream at a high syntactical level instead of at a low-level, bit-per-bit basis.

Recently, we developed an extension to XFlavor that allows to create BSDL compatible BSDs when taking into account certain restrictions. As such, an XFlavor-alike description was developed for the first version of the H.264/AVC standard, allowing to discover its Annex B syntax up to and including the slice header datastructure. However, the XFlavor-alike description only allows to process H.264/AVC bitstreams that contain one Sequence Parameter Set (SPS) and one Picture Parameter Set (PPS).

4 Simulations

4.1 Context

In this section, experimental performance measurements are presented in terms of file sizes and computational times. Two use cases are targeted: a download-and-play scenario and a simulstore-based streaming scenario. In both scenarios, the goal is to make use of an H.264/AVC bitstream that has provisions for the exploitation of multi-layered temporal scalability in order to target three different usage environments: a desktop computer able to process video data at 30 Hz, a portable entertainment device able to process video data at 15 Hz, and a cell phone able to process video data at 7.5 Hz. The bitstreams in question (Foreman; CIF resolution; 300 pictures; 30 Hz) are compressed by making use of the pyramid encoding technique and are compliant with H.264/AVC's Main Profile. With respect to the streaming scenario, three slices per picture are used for improved error robustness while the download-and-play scenario only uses one slice per picture. It is also important to know that both scenarios are relying on I slice coded pictures, P slice coded pictures, and B slice coded pictures (i.e., all slices in the same picture share a common value for the `slice_type` syntax element). The resulting bitstreams contain one SPS and one PPS.

To demonstrate the usefulness of the proposed techniques, simulations were carried out for bitstreams having the following GOP structure: $I_0p_2P_1p_2P_0$, $I_0b_2B_1b_2P_0$, $I_0p_3P_2p_3P_1p_3P_2p_3P_0$, and $I_0b_3B_2b_3B_1b_3B_2b_3P_0$. The $I_0b_2B_1b_2P_0$ coding pattern is shown in Figure 3. Every picture (frame) contains three slices. Each slice is tagged with its type and the value of `frame_num`. Each sub-sequence information SEI message is tagged with the value of the `sub_seq_frame_num`

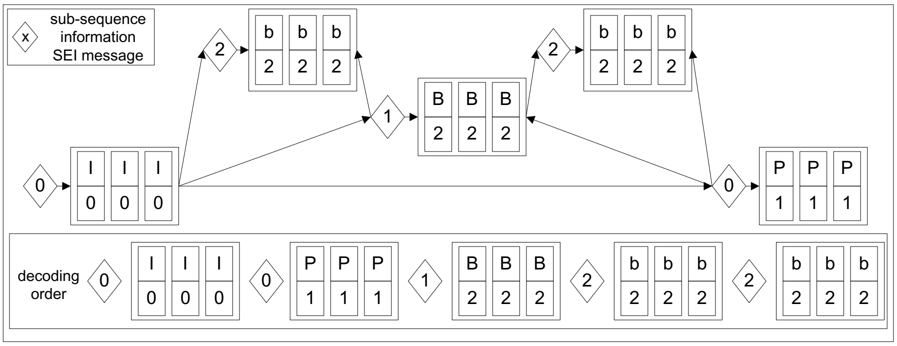


Fig. 3. The $I_0b_2B_1b_2P_0$ coding pattern for the streaming scenario

syntax element, making it possible to identify a sub-sequence layer. It is clear that the $I_0b_2B_1b_2P_0$ coding pattern contains three different sub-sequences and that each sub-sequence is part of a different sub-sequence layer. Hereby, 'P' and 'p' denote a P slice coded reference picture and P slice coded non-reference picture, respectively. As such, the settings used are in line with the ones as applied in the complementary paper by Tian *et al.* [5]. The first two GOP structures offer three-level temporal scalability while the last two GOP structures offer four temporal resolutions. The value of the quantization parameter for the base layer is equal to 28 and is increased by two for every additional layer.

The simulations were done on a PC having an AMD Athlon4 1600+ CPU and 512 MB of RAM at its disposal. The operating system used was Windows Server 2003, running the Java 2 Runtime Environment (SE version 1.4.2_07). SAXON 6.5.2 was used in order to apply Extensible Stylesheet Language Transformations (XSLTs) to BSDs. The H.264/AVC bitstreams were created by relying on the JM 9.4 reference software. Half of them were manually annotated with SEI metadata by relying on the BSDL approach. Version 1.1.3 of the MPEG-21 BSDL reference software was used. In order to make a fair comparison with the extended version of XFlavor, the generic BSDL schema was simplified such that it is only possible to describe bitstreams that contain one SPS and PPS.

4.2 Simulation Results

This section covers a selection of the performance results that were obtained. Due to place constraints, only the results for the most challenging scenario are shown, i.e., the streaming scenario. From Table 1, it is clear that the obtained values for the different metrics are almost all independent from the GOP structure used. The latter only has a clear impact on the amount of data dropped: obviously, the bit rate reduction is higher when exploiting temporal scalability in case of a GOP structure that embeds P slice coded pictures. This is for instance relevant in case one has to pay for the amount of data transfered.

Table 1. Simulation results for the streaming scenario

			MPEG-21 BSDL					XFlavor+	
	coding pattern	fps (Hz)	parse time (s)	transform time (s)	BSD (KB)	cBSD (KB)	BSDtoBin (s)	parse time (s)	bitstream size (%)
no SEI	IpPpP	30	3696.2	-	1895.1	18.5	-	1.2	100.0
		15	-	2.3	1571.5	10.6	5.4	-	76.9
		7.5	-	2.2	924.9	6.2	3.9	-	53.5
	IpPpPpPpP	30	3722.1	-	1975.3	19.2	-	1.1	100.0
		15	-	2.5	1571.5	10.8	5.5	-	78.1
		7.5	-	2.4	791.4	6.4	3.8	-	56.9
	IbBbP	30	3975.4	-	1998.0	17.7	-	1.2	100.0
		15	-	2.4	1713.6	10.4	5.3	-	86.2
		7.5	-	2.3	924.1	6.2	3.9	-	67.5
	IbBbBbBbP	30	4066.6	-	1964.8	17.2	-	1.3	100.0
		15	-	2.5	1655.4	10.6	5.2	-	87.4
		7.5	-	2.4	864.0	6.4	3.8	-	70.1
SEI	IpPpP	30	300.9	-	793.9	12.0	-	0.7	100.0
		15	-	1.8	561.6	7.4	3.1	-	76.8
		7.5	-	1.7	287.1	4.5	2.4	-	53.4
	IpPpPpPpP	30	300.2	-	793.7	11.6	-	0.7	100.0
		15	-	1.8	561.6	7.3	3.0	-	78.0
		7.5	-	1.7	287.1	4.6	2.3	-	56.8
	IbBbP	30	301.0	-	793.7	11.8	-	0.6	100.0
		15	-	1.8	561.6	7.3	2.9	-	85.9
		7.5	-	1.8	287.1	4.5	2.3	-	67.3
	IbBbBbBbP	30	300.4	-	793.5	11.4	-	0.6	100.0
		15	-	1.7	561.5	7.3	2.7	-	87.1
		7.5	-	1.7	287.0	4.5	2.1	-	69.7

With respect to the bitstreams that do not carry SEI, the following observations can be made. The amount of time, as needed by the parser that is available in the MPEG-21 BSDL reference software package, is unacceptably high in order to create a BSD (3865s on the average). This can be explained by the fact that a lot of XPath expressions have to be executed for resolving the necessary dependencies in order to correctly guide the parsing process. Those XPath expressions are needed for retrieving information about the structure of the bitstream as discovered so far. Due to the fact that H.264/AVC's syntax is described with a rather fine granularity (the resulting BSDs contain info up to and including the slice header syntax structure), the parser is forced to delve deeply into the structure of an H.264/AVC bitstream. It is supposed by the authors that this fundamental problem cannot be solved without relying on some non-normative extensions to MPEG-21 BSDL. From that point of view, it is interesting to notice that an equivalent BSD can be created by our modified version of XFlavor in hardly more than one second. XFlavor has much faster access to the information already gathered thanks to its object oriented nature: only simple indexing operations are needed. Note that the BSDs, as created by our extended version of XFlavor, also produce a certain overhead when comparing them with the BSDs as produced by the BSDL software: the average size of a full BSD is 2594.1 KB in case of XFlavor and 1958.3 KB in case of BSDL (not shown in Table 1).

Table 1 also makes clear that the resulting BSDs can be compressed very efficiently. Compression factors of up to 182 are possible when using common compression software (WinRAR). Adapting a BSD can be done very efficient as well. The latter only takes a few seconds. The syntax elements `frame_num` and `nal_ref_idc` are used to guide the adaptation process in the case of bitstreams that do not carry SEI. For bitstreams containing sub-sequence information SEI messages, the sub-sequence layer identification information (stored in the syntax element `sub_seq_layer_num`) and the value of `nal_unit_type` are used to guide the adaptation process. Customized bitstreams can also be created very fast when making use of the MPEG-21 BSDtoBin tool, especially due to the fact that this process does not require the evaluation of expensive XPath-expressions. The only things that have to be taken care off, are the appropriate binarization of certain values by performing look-ups in the BSDL schema and the selection of the appropriate data packets from the parent bitstream by performing look-ups in the BSD. Both types of BSDs can be used in order to create a customized bitstream since BSDL-based and XFlavor-based BSDs are completely equivalent.

When relying on SEI messages, there is no need to delve deeply into the syntax for gathering the necessary information in order to be able to exploit temporal scalability. This can immediately be derived from the cost needed for generating a BSD: the amount of time has dropped significantly in case of the MPEG-21 BSDL parser (from 3865s on the average to 300s on the average) because of the fact that very few XPath expressions have to be evaluated. The uncompressed BSDs are also two to three times smaller, resulting in a positive impact with respect to the time needed for adapting such a lightweight BSD and for customizing the corresponding bitstream. Hence, the sub-sequence information SEI messages make it straightforward to exploit multi-layered temporal scalability by relying on BSDs.

It is also important to note that BSDL is more fitted for dropping sub-sequence layers (static content adaptation) in pre-coded bitstreams, while the disposal of individual sub-sequences is more of interest to streaming servers for achieving short-term, immediate, and accurate bit rate adjustment (dynamic content adaptation). One can also see that the presence of the sub-sequence information SEI messages hardly has an impact on the size of the compressed bitstreams, while those content adaptation hints are well suited for enabling fast and intelligent bitstream customization[3].

5 Conclusions

In this paper, two languages were discussed that provide solutions for discovering the structure of a scalable bitstream in order to generate its XML description and for the generation of an adapted bitstream using the transformed description. Their (combined) usage was developed in more detail with respect to the valid exploitation of multi-layered temporal scalability in H.264/AVC's base specification. Special attention was paid to the usage of the sub-sequence coding technique, enabling the easy and efficient identification of disposable chains of pictures when processing pre-coded bitstreams. Some performance measurements in terms of file sizes and computational times were presented as well, illustrating the feasibility of the presented concepts. Our results show that the

[3] As such, there is a correspondence with gBS Schema's marker concept.

sub-sequence related SEI messages have a positive impact on the efficiency of the bitstream customization process, especially due to the fact that those content adaptation hints assist in abstracting the coding format to be manipulated. As such, the BSDs, together with the sub-sequence coding technique and the sub-sequence related SEI messages, offer an elegant and practical solution for the exploitation of multi-layered temporal scalability in H.264/AVC's base version.

Acknowledgements

The authors would like to thank Gary Sullivan and Miska Hannuksela for providing information about the sub-sequence coding technique and the vision behind the `frame_num` syntax element.

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. Lerouge, S., Lambert, P., Van de Walle, R.: Multi-criteria Optimization for Scalable Bitstreams. Proceedings of the 8th International Workshop on Visual Content Processing and Representation, p. 122-130, Springer, (Madrid), September 2003
2. Sullivan, G., Topiwala, P., Luthra, A.: The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions. Applications of Digital Image Processing XXVII **5558** (2004) 454–474
3. Schwarz, H., Marpe, D., Wiegand, T.: Hierarchical B pictures. Doc. JVT-P014, Poznan, Jul. 2005
4. Hannuksela, M.: Enhanced Concept of a GOP. Doc. JVT-B042, Geneva, Jan. 2002
5. Tian, D., Hannuksela, M., Gabbouj, M.: Sub-sequence Video Coding for Improved Temporal Scalability. Proc. ISCAS 2005, Kobe, Japan, May 23-26, 2005
6. Amielh, M., Devillers, S.: Bitstream Syntax Description Language: Application of XML-Schema to Multimedia Content Adaptation. In WWW2002: The Eleventh International World Wide Web Conference, (Honolulu, Hawaii), May 2002.
7. Hong, D., Eleftheriadis, A.: XFlavor: Bridging Bits and Objects in Media Representation. Proceedings, IEEE Int'l Conference on Multimedia and Expo (ICME), Lausanne, Switzerland, August 2002
8. 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
9. De Neve, W., Lerouge, S., Lambert, P., Van de Walle, R.: A Performance Evaluation of MPEG-21 BSDL in the Context of H.264/AVC. Applications of Digital Image Processing XXVII **5558** (2004) 555–566