XML-driven Bitrate Adaptation of SVC Bitstreams

Tom Paridaens, Davy De Schrijver, Wesley De Neve, and Rik Van de Walle
Department of Electronics and Information Systems – Multimedia Lab
Ghent University - IBBT
Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium
email: {tom.paridaens, davy.deschrijver, wesley.deneve, rik.vandewalle}@ugent.be

***Abstract*— Thanks to technological evolutions, the number of devices capable of playing video bitstreams is growing. The heterogenity in these devices grows in terms of screen resolution, processing power, and available band width. In this paper, we describe an MPEG-21 Bitstream Syntax Description Language-based (BSDL-based) adaptation framework that allows providers to easily adapt scalable bitstreams without having to recode the original bitstream. We describe the steps necessary to adapt the bitstreams through BSDL. The main contribution of this paper is an optimized adaptation framework using a Bitstream Syntax Schema developed to minimize the size of the Bitstream Syntax Descriptions (BSDs). Furthermore, we created a Streaming Transformations for XML Stylesheet (STX-stylesheet) to exploit the advantages of Fine Grain Scalability, this to adapt the bitrate of Scalable Video Coding bitstreams in the most accurate way possible. Our results show that BSDL-based adaptation is able to compete with binary adaptation tools. The target bitrates can be reached within a margin of 2%, which is comparable to the reference software which uses binary adaptation.**

I. Introduction

The evolution of mobile devices (e.g., Personal Digital Assistants (PDAs), mobile phones) in terms of computing power and screen resolution has allowed more and more people to have access to video streams in a mobile context. All these different devices pose different requirements to the video bitstreams offered by content providers. Mobile phones typically have a smaller screen and lower band width (e.g., through Global Packet Radio Services (GPRS)), while PDAs have higher resolution screens and can have access to broadband connections such as Wireless Fidelity (WiFi). In order to serve all these different devices, providers typically have several versions of the same video bitstream (e.g. different band widths, resolutions,...). These multiple versions result in a high storage and encoding costs, consequently providers want to limit the number of different versions.

To allow the adaptation of video bitstreams in order to fit the different requirements these devices pose, a new extension to the H.264/AVC-standard is being developed, H.264/AVC Scalable Video Coding (SVC) [1]. The SVC standard allows to create one "parent"-bitstream from which adapted bitstreams can be extracted in a straightforward way. The resulting bitstreams can have a lower resolution (spatial axis), frame rate (temporal axis) and/or bitrate (quality axis). This allows providers to adapt the stream to every device without the need of coding and storing the stream in several versions. Still, in order to adapt the bitstreams, extra techniques are necessary. These adaptations can be applied in several domains (e.g., Binary, Textual).

In this paper, we will pose an adaptation solution for SVC video bitstreams based on the use of the Bitstream Syntax Description Language (BSDL) [2]. This solution will create a textual description of the video bitstreams, transform this description and finally, create the adapted video bitstream. We focus on adaptation along the quality axis for more accuracy, although adaptation along the spatial and temporal axis are possible too [3].

This paper is organized as follows. Section II describes the fundamental technologies used in this paper, i.e. MPEG-21 BSDL and Joint Scalable Video Model 8 (JSVM8). Furthermore, we describe the workflow of the MPEG-21 BSDL adaptation framework as used in our tests. Section III describes the actual adaptation process, while Section IV discusses the performance results. Finally, Section V concludes.

II. Fundamental Technologies

In this section, an overview is given of the fundamental technologies used in this paper.

A. MPEG-21 BSDL

MPEG-21 BSDL is part of the Digital Item Adaptation standard of the MPEG-21 Multimedia Framework. A BSDL framework allows to generate a (high-level) Extensible Markup Language (XML) description of the structure of a bitstream in a *format-agnostic* manner, called the Bitstream Syntax Description (BSD) [4]. This high-level approach allows an easier adaptation of a bitstream as only a limited knowledge of the bitstream structure is necessary. The adaptation typically happens, in video context, on a picture per picture basis rather than on a bit-per-bit basis. Furthermore, as a BSD is an XML file, existing XML transformation technology can be used for expressing a particular adaptation.

Fig. 1 shows the workflow of a BSDL-based adaptation framework. The workflow consists of three steps and takes place in two different domains (Binary and XML). In the first step, the description of the bitstream structure is generated using the BintoBSD-tool (conversion from Binary to XML domain). To generate this description, a Bitstream Syntax Schema (BS Schema) is needed that describes the structure of an SVC bitstream. This BS Schema makes the BSDL framework format-agnostic. This implies that the proposed framework can adapt bitstreams compliant with other coding

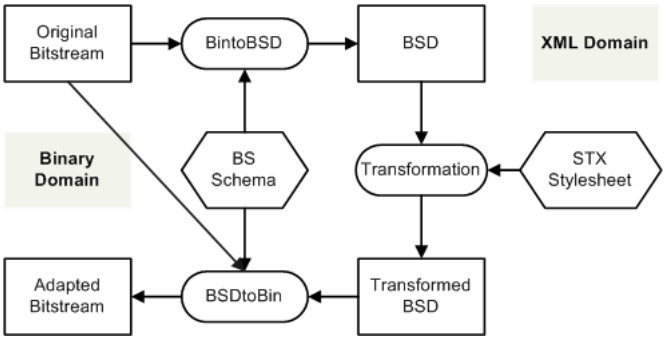

Fig. 1. Workflow of an MPEG-21 BSDL-based adaptation framework.

formats as well (such as JPEG2000 [5] for still images, SVC for video, and MPEG-4 Bit Slice Arithmetic Coding [6] for audio) as long as a BS Schema of the bitstream structure is available.

In the second step the actual adaptation is applied. As the description is an XML file, regular XML transformation techniques such as Streaming Transformations for XML (STX) [7] can be used. Using these techniques, a description is generated that complies with the imposed restrictions.

In the final step, the transformed description is used by the BSDtoBin-tool to generate the adapted bitstream (conversion from XML to Binary domain).

B. H.264/AVC SVC

H.264/AVC SVC [1] is a block-based coding scheme based on the H.264/AVC standard. SVC introduces several types of embedded scalability levels (temporal, spatial, and quality scalability, also called Signal-to-Noise Ratio scalability(SNR scalability)). Thanks to this embedded scalability, an SVC bitstream can be adapted without re-encoding. By leaving out unnecessary Network Abstraction Layer Units (NALUs) or truncating them, the frame rate, resolution, and quality of an SVC bitstream can be lowered. In this paper, we use the JSVM8 specification as implemented in the 7.5 reference software. This software suite contains a reference adaptation utility which works in the binary domain and therefore is format-specific.

In this paper, we make use of Fine Grain Scalability (FGS). FGS is a form of Signal-to-Noise Ration Scalability (SNR scalability). To enable FGS, a bitstream consists of several *quality layers*. The lowest layer is called the *base layer*. The additional layers are called *enhancement layers*. The base layer must not be deleted or adapted as the enhancement layers are based on it. Therefore, the base layer declares the lowest achievable bitrate (using only SNR scalability). In order to control the bitrate of the adapted video bitstreams as fine as possible, we make use of FGS. FGS makes it possible to cut off the bitstream at any byte position in the enhancement layers. Note that an enhancement layer can only exist if all the lower-quality layers are unadapted. As stated in the specification, every layer is placed in its own NALU.

III. Adaptation

In this section, we will discuss the regulation of the bitrate by adaptation along the SNR axis, using FGS. This bitrate is defined by the user environment. After adaptation, the adapted stream can be used in this environment.

Fig. 2 shows a simplified example of a transformation of a BSD for adaptation along the SNR axis. The BSD has been simplified for clarity and describes the 4 NALUs shown in the drawing (corresponding to one frame in this example). The second NALU in this example contains the first enhancement layer (line 8: `quality_level` is 1). As this NALU contains an enhancement layer, it can either be dropped or adapted. In this example, we adapt the value of the `slice_payload` element (line 10). This element describes the byte position of the actual video information in the current NALU (the first number) and the size of that information (the second number) in bytes. By adapting the value of the second number, the size of the NALU is adapted and therefore the bitrate of the video bitstream. When adapting the `slice_payload` element, the NALUs containing the higher enhancement layers have to be dropped (in this case with `quality_level` higher than 1).

In order to define the cut-off point and to transform the BSDs, we developed a STX-stylesheet. STX is chosen for its low memory requirements and its speed [8]. The stylesheet defines the cut-off point using the information embedded inside the video bitstream. Every bitstream generated with the reference software, implementing JSVM8, contains Supplemental Enhancement Information messages (SEI messages). In these messages lies bitrate information for the different quality layers inside the bitstream. In our case, the most important bitrate information is the average bitrate. Using the information of the average bitrates of all the available quality layers, the adaptation engine calculates the point every NALU has to be cut off at. The engine first determines which of the layers contains the target bitrate. The NALUs containing the layers beyond the cut off layer will be removed. Then, if the bitrate is in the range of an enhancement layer, the engine calculates which percentage of this layer should be preserved:

$$Percentage_{preserved} = \frac{(target\_bitrate - avg\_bitrate_{layer\_x})}{(avg\_bitrate_{layer\_x+1} - avg\_bitrate_{layer\_x})} \quad (1)$$

with $layer_x$ the highest, completely reserved layer and $layer_{x+1}$ the layer containing the cut-off point. This percentage is used to define the cut-off point for every NALU containing the highest preserved layer by changing the value of `slice_payload`:

$$slice\_payload = slice\_payload * Percentage_{preserved} \quad (2)$$

If the target bitrate is below the bitrate of the base layer the target bitrate is unreachable and the enhancement layers will be dropped. If the bitrate is higher than the maximum bitrate of the highest layer, the complete bitstream will be preserved.

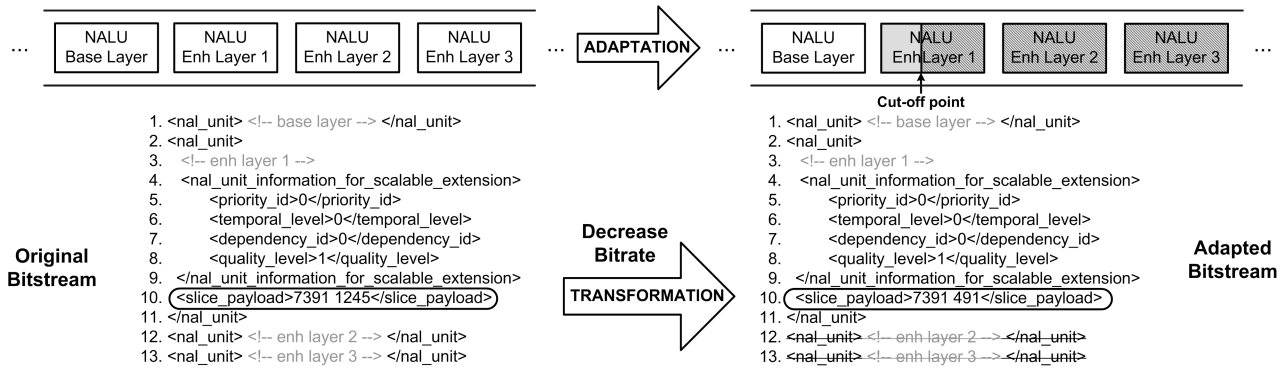

Fig. 2. Adaptation of the bitrate.

IV. Test Results

In this section, we describe the test sequences and discuss the performance of the adaptation using BSDL.

A. Test Sequences

The test sequences are created using the JSVM8 reference software [9] containing four quality layers, one spatial layer, and five temporal layers. Table I shows the bitrates for the different quality layers for each of the test sequences. The bitrates are the sum of the bitrates of the lower layers and the layer itself. The test sequences are:

TABLE I
AVERAGE BITRATES OF THE QUALITY LAYERS OF THE TEST SEQUENCES (KBIT/S).

	Seq. 1	Seq. 2	Seq. 3	Seq. 4	Seq. 5
Base Layer	863	166	459	90	316
Enh. Layer 1	1535	276	771	218	738
Enh. Layer 2	3159	510	1408	426	1796
Enh. Layer 3	7902	1190	2671	888	4177

- Sequence 1: Crew (4CIF, 300 frames, 1507 NALUs)
- Sequence 2: Foreman (CIF, 300 frames, 1507 NALUs)
- Sequence 3: Football (CIF, 250 frames, 1257 NALUs)
- Sequence 4: Stockholm (CIF, 250 frames, 1257 NALUs)
- Sequence 5: Compilation (CIF, 2100 frames, 10507 NALUs): Container, Foreman, Head with glasses, Mobile, Mother Daughter, News and Silent

Note that one frame, in our tests, consists of five NALUs (one NALU for H.264/AVC compatibility, one for the base layer, and three for the enhancement layers). The bitstream also has seven extra NALUs (one NALU for the scalability information, and six NALUs for the Sequence and Picture Parameter Sets).

B. Execution Times

Table II shows, for each sequence, the playback time, the processing time for each step in the adaptation chain, and the total processing time. The measurements were done on a PC with an Intel Pentium D 3.0 GHz CPU, having 2 GB RAM and running Windows XP Professional SP2 and the Sun Microsystems Java Runtime 1.5.0_09. The times are measured for 6 consecutive executions from which the 2 best and the 2 worst results were removed. The average of the 2 remaining times is given. The execution times for the BSDtoBin step are the average of the times to create a 300 kbit/s, a 1000 kbit/s and a 2500 kbit/s stream. The values between brackets are the maximum deviation. When looking at the total processing time, we see two notable results. First of all, we see that the high resolution of Sequence 1 (compared to the other sequences) has a relatively small effect on the total processing time, when compared to the increase in bitrate. This is due to the fact that the processing speed is mostly dependent on the number of NALUs a sequence contains, rather than on the actual bitrate. The low effect of the bitrate on the processing time of bitstreams, is shown by the processing times for Sequences 3 and 4. While the bitrate of Sequence 3 is at least three times higher, the processing time is only 2% higher. When we compare the processing times for Sequences 2 and 3, we also see the result of the time needed to load the bintoBSD-tool and the BS Schema (this takes approx. 0.6s).

TABLE II
PROCESSING TIMES FOR THE ADAPTATION CHAIN (S).

	Seq. 1	Seq. 2	Seq. 3	Seq. 4	Seq. 5
Playback	10	10	8.3	8.3	70
BintoBSD	10.3	6.6	6.3	5.7	46.7
Transform	3.2	3.5	3.0	3.1	16.7
BSDtoBin	1.4 (0.12)	1.5 (0.14)	1.4 (0.04)	1.4 (0.08)	3.5 (1.1)
Total	14.9	11.6	10.7	10.2	66.9

When we look at the separate steps, we see that the generation of the BSD is the most complex. This is not a problem as the generation is still real time (for long sequences) and is done only once.

C. File Sizes

To adapt the original bitstream with BSDL, the BSDs have to be generated. In our tests, we tried to keep these descriptions as small as possible. Therefore we created a BS schema for a high-level description, containing only the Supplemental Enhancement Information messages (SEI), the NALU Headers containing the scalability information, the

Picture and Sequence Parameter Sets and the payload size of the slices inside the NALU. By leaving all information not necessary for the adaption of the bitstreams out of the BSDs, we reach an acceptable generation speed and relatively small XML files (compared to BSDs generated by BS Schemas describing the structure of the bitstreams completely). Table III shows the sizes of all created BSDs. As with the processing time, the file size of the original BSD is mostly dependent on the number of NALUs rather than on the bitrate. This is why Sequences 1 and 2 (and 3 and 4) have a (nearly) equal size description despite their different resolution and size.

TABLE III
SIZE OF VIDEO BITSTREAMS AND GENERATED XML FILES (KB).

	Seq. 1	Seq. 2	Seq. 3	Seq. 4	Seq. 5
Bitstreams	9647	1453	2718	903	35690
BSD	1738	1736	1459	1457	11771
BSD 300 kbit/s	624	1365	840	1148	3972
BSD 1000 kbit/s	624	1736	1150	1457	9169
BSD 2500 kbit/s	1367	1736	1459	1457	11769

The second part of Table III shows the BSD sizes after transformation. We can see that the size of the BSDs grows as the target bitrate raises. This is because a higher bitrate results in more preserved quality layers, and thus more preserved NALUs in the BSD. The equal size of the 300 kbit/s and 1000 kbit/s BSDs for Sequence 1 are the result of the limited bitrate range. These bitrates are outside of the bitrate range of the Sequence so the transformed BSD contains only the base layer, the lowest possible bitrate.

For Sequence 2, we see the BSD size is equal for 1000 kbit/s and 2500 kbit/s. This is because the highest enhancement layer is already used for every picture in the 1000 kbit/s bitstream. The only differences between the 2500 kbit/s BSD and the 1000 kbit/s BSD are the values of the `slice_payload` elements. Therefore, the size of the BSDs are nearly the same.

D. Adaptation Performance

To conclude the performance results, we show the resulting bitrates in Table IV and the corresponding PSNR values in Table V. We also added the resulting bitrates and PSNR values when using the reference software. The Italic numbers show the bitrates and PSNR-values limited by the bitrate ranges of the test sequences.

As can be seen in Table IV and V, our BSDL Framework can compete with the binary, format-specific adaptor of the reference software suite. The resulting bitrates are within 2% of the required bitrate.

TABLE IV
RESULTING BITRATES AFTER ADAPTATION (KBIT/S).

		Seq. 1	Seq. 2	Seq. 3	Seq. 4	Seq. 5
300 kbit/s	BSD	*863*	298	*459*	300	*316*
	Ref	*863*	292	*459*	293	*316*
1000 kbit/s	BSD	983	984	989	*888*	981
	Ref	977	977	977	*888*	976
2500 kbit/s	BSD	2456	*1190*	2463	*888*	2447
	Ref	2442	*1190*	2442	*888*	2441

TABLE V
RESULTING PSNR-VALUES AFTER ADAPTATION (DB).

		Seq. 1	Seq. 2	Seq. 3	Seq. 4	Seq. 5
300 kbit/s	BSD	*33.12*	35.31	*30.25*	36.0	*36.90*
	Ref	*33.12*	35.27	*30.25*	35.9	*36.90*
1000 kbit/s	BSD	33.59	39.74	34.28	*41.00*	39.99
	Ref	33.57	39.71	34.21	*41.00*	39.98
2500 kbit/s	BSD	36.85	*40.83*	40.17	*41.00*	43.67
	Ref	36.83	*40.83*	40.08	*41.00*	43.66

V. CONCLUSION

In this paper, we have shown that a Bitstream Syntax Description Language Framework (BSDL Framework) has many advantages over binary adaptation tools. It is, as it is BSDL-based, *format-agnostic*, high level and allows the use of existing transformation techniques such as Streaming Transformations for XML (STX). The framework creates a high-level description, transforms it conform the required bitrate for the video bitstream, and then creates, using the transformed description, the adapted video bitstream. Our tests show that a BSDL framework, using our simplified Bitstream Syntax Schema (BS Schema) and STX-stylesheet for Fine Grain Scalability (FGS) quality adaptation, can compete with binary adaptation tools such as the reference software. The adapted video bitstreams have a bitrate within 2% of the required bitrate.

ACKNOWLEDGEMENTS

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] ITU-T and ISO/IEC JTC1 *JVT-SVC specifications* http://ftp3.itu.ch/av-arch/jvt-site/2006_10_Hangzhou/JVT-U201.zip
[2] M. Amielh, S. Deviller *Bitstream Syntax Description Language: Application of XML schema to multimedia content adaptation* Proceedings of the 11th International WWW Conference, Honolulu, Hawaii, 2002
[3] D. De Schrijver, W. De Neve, K. De Wolf, R. De Sutter, R. Van de Walle *An Optimized MPEG-21 BSDL Framework for the Adaptation of Scalable Bitstreams* Journal of Visual Communication and Image Representation, http://www.elsevier.com/
[4] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner *Bitstream Syntax Description-Based Adaptation in Streaming and Constrained Environments* IEEE Transactions on Multimedia, Vol. 7, No. 3, June 2005
[5] JPEG committee *JPEG 2000* http://www.jpeg.org/jpeg2000/
[6] ISO/IEC *Audio Lossless Coding (ALS), new audio profiles and BSAC extensions* ISO/IEC 14496-3:2005/Amd 2:2006
[7] P. Cimprich *Streaming transformations for XML version 1.0 working draft* http://stx.sourceforge.net/documents/spec-stx-20040701.html
[8] D. De Schrijver, W. De Neve, D. Van Deursen, J. De Cock, and R. Van de Walle *On an Evaluation of Transformation Languages in a Fully XML-driven Framework for Video Content Adaptation* IEEE ICICIC, 2006
[9] ITU-T and ISO/IEC JTC1 *JVT-SVC reference implementation* http://ftp3.itu.ch/av-arch/jvt-site/2006_07_Klagenfurt/JVT-T203.zip