Analysis of BSDL-Based Content Adaptation for JPEG 2000 and HD Photo (JPEG XR)

W. De Neve[1, 2], S. Yang[1], D. Van Deursen[2], C. Kim[1], Y. M. Ro[1], R. Van de Walle[2]

[1] Image and Video Systems Lab, Multimedia Group, Information and Communications University, South Korea
[2] Multimedia Lab, Department of Electronics and Information Systems, Ghent University - IBBT, Belgium

Keywords: Adaptation, BSDL, HD Photo, JPEG 2000.

Abstract

JPEG 2000 and HD Photo (JPEG XR) enable the coding of images with several adaptivity provisions. These provisions allow taking into account the capabilities and constraints of diverse usage environments. However, the actual adaptation of coded bitstreams requires additional system complexity. This paper investigates how JPEG 2000 and HD Photo can be used in a standardized and XML-based framework for format-independent content adaptation in the compressed domain. An analysis is provided of the cost of offering adaptivity in the compressed domain, in terms of loss of coding efficiency, as well as in terms of the complexity of the format-agnostic adaptation system used. Our results show that editing-style, high-level operations on JPEG 2000 and HD Photo bitstreams can be executed with a feasible complexity. However, low-level operations in the compressed domain cannot be supported by the format-independent adaptation system used, due to the need for entropy coding and data reordering at macroblock level.

1 Introduction

The ever-increasing heterogeneity in terms of coding formats, devices, and networks hampers the straightforward consumption of multimedia content. This observation has triggered several research and standardization efforts in the domain of scalable coding formats, which allow taking into account different device and network characteristics, and format-independent adaptation systems, which allow dealing with an increasing number of scalable coding formats. An extensive overview of a number of these research and standardization activities can for instance be found in [8].

This paper discusses the integration of two formats for the coding of still images into a standardized and format-independent adaptation system. The investigated coding formats are JPEG 2000 [7], which became an international standard in December 2000 (ISO/IEC 15444-1), and HD Photo [4], which is currently under consideration for standardization by the JPEG group. This potentially new standard is tentatively called JPEG Extended Range (JPEG XR). Both JPEG 2000 and HD Photo offer support for the scalable and lossless representation of still images and intra-only video.

Table 1: Lossless coding of 24-bits RGB.

coding tool	JPEG 2000	HD Photo
1. inter component transform	Reversible Component Transform (RCT)	reversible YCoCg (YCoCg-R)
2. intra component transform	tile-based, reversible, integer-valued, discrete, wavelet transform	block-based, hierarchical, lapped, reversible, integer-valued transform
3. prediction (i.e. removal of inter-block redundancy)	no prediction in the spatial or frequency domain	prediction of 4×4 blocks of transform coefficients
4. post-processing of transform coefficients	bit-plane coding	adaptive coefficient scanning
5. entropy coding	context-driven, binary arithmetic coding	context-driven Huffman coding (adaptive VLC table switching)

The adaptation system investigated in this paper makes use of the Bitstream Syntax Description Language (BSDL; [2]), which enables the automatic creation of XML-based descriptions of the high-level structure of coded bitstreams. These descriptions only contain information required for content adaptation purposes, thus concealing the internal complexity of a scalable format. The main merit of BSDL lies in the fact that it allows developing a format-independent adaptation system – the same logic can be applied for the adaptation of scalable audio, video, and still image bitstreams. Format-independence also means that the adaptation system is future proof and suitable for a hardware implementation.

The organization of this paper is as follows. Section 2 provides a technical discussion of JPEG 2000 and HD Photo, putting the emphasis on the emerging HD Photo format. Content adaptation using BSDL is briefly discussed in Section 3, while extensive experimental results are presented in Section 4. Finally, Section 5 concludes this paper.

2 Technical discussion of HD Photo and JPEG 2000

This section briefly reviews the new HD Photo format in terms of its coding tools and adaptivity provisions. A concise comparison is drawn with JPEG 2000, a coding format that has already been extensively described in the scientific literature (see for instance [7]). Consequently, we assume some awareness of the basic design principles of JPEG 2000.

2.1 Coding algorithm and bitstream structure

The operation of an HD Photo encoder for the lossless compression of 24-bits RGB image samples is as follows.

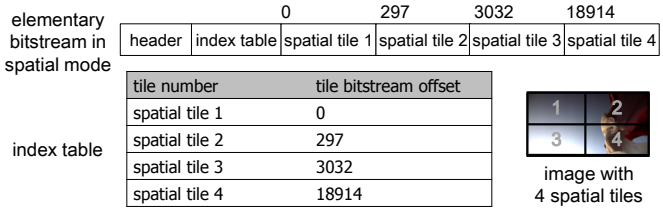

Figure 1: Tile bitstream offsets stored in an index table.

First, for decorrelation purposes, a reversible inter component transform is applied, essentially transforming the color samples from the RGB to the YCoCg color space. A two-stage 4×4 and integer-valued block transform is subsequently applied to the YCoCg color samples, together with optional deblocking and deringing. Next, DC/AC prediction is performed on 4×4 blocks of transform coefficients. Finally, adaptive coefficient scanning is applied to the predicted transform coefficients, followed by adaptive Huffman coding. These steps are also summarized in Table 1, and compared to JPEG 2000.

The resulting elementary bitstream consists of a header, an index table, and a sequence of tiles, as illustrated by Figure 1. The spatial tiles can be stored using two modes of operations: spatial mode or frequency mode. In spatial mode, the compressed bits of each macroblock are located together. In frequency mode, the transform coefficients of all macroblocks in a spatial tile are laid out as a hierarchy of four frequency subbands: a DC , lowpass (LP), highpass (HP), and Flexbits subband. Each frequency subband for a particular spatial tile is stored in a separate frequency tile.

2.2 Adaptivity provisions in the compressed domain

Natively supported by Windows Vista, HD Photo can be considered the first widespread scalable coding format, thus motivating the integration effort described in this paper. Scalable image representations are for instance useful for applications that require online and dynamic interaction with high-resolution images – servers only have to extract and send the parts needed by a particular client.

In frequency mode, a trivial form of quality scalability is supported by simply removing all Flexbits tiles from an HD Photo bitstream. For images coded in a lossless manner, removing the Flexbits tiles comes down to degrading the image quality from a lossless to a lossy representation. Spatial scalability (or coarse-grained quality scalability) is supported by additionally removing the HP and LP subband tiles, each time resulting in a reduction of the spatial resolution by a factor of four along the horizontal and vertical axis. The removal of tiles also requires updating the index table in order to maintain format compliance: offsets belonging to removed tiles need to be eliminated from the index table, while remaining offsets need to be recomputed due to their absolute nature.

Table 2: Adaptivity provisions in JPEG 2000 and HD Photo.

adaptivity provision	JPEG 2000	HD Photo
flipping and rotating	+	+
arbitrary ROI extraction	+	+
tile-aligned ROI extraction	++	++
spatial scalability	++	++
lossless-to-lossy degradation	++	++
fine-grained quality scalability	++	-
color component scalability	++	+
bitstream reordering	++	+
spatial retiling	+	+

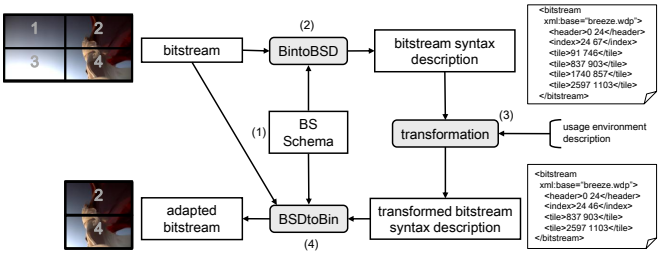

Figure 2: Adaptation with BSDL.

All compressed-domain operations supported by HD Photo are summarized in Table 2, and compared to JPEG 2000: '-' means that the operation is not supported; '+' denotes that (partial) entropy decoding and/or bitstream reordering at macroblock level is required; and '++' indicates that the operation can be executed in an editing-style fashion (e.g., by simply removing particular tiles and updating certain syntax elements).

3 Content adaptation using BSDL

BSDL is a standardized language for describing the high-level syntax of a particular media format in a formal way. Its use is illustrated in Figure 2 for the tile-aligned extraction of a region-of-interest (ROI): (1) a Bitstream Syntax Schema (BS Schema) is written in BSDL; (2) a Bitstream Syntax Description (BSD) is created by the format-independent BintoBSD Parser, taking as input a particular bitstream and a corresponding BS Schema; (3) the BSD is transformed to meet the constraints of a certain usage environment; (4) an adapted bitstream is created using the format-independent BSDtoBin Parser, taking as input the BS Schema, the original bitstream, and the transformed BSD.

BSDL was initially part of the MPEG-21 Digital Item Adaptation standard (MPEG-21 DIA; ISO/IEC 21000-7; [8]). However, to promote its use for purposes beyond adaptation (such as format-independent streaming and multiplexing), the language in question was recently relocated to a standalone standard known as MPEG-B BSDL (ISO/IEC 23001-5). For a more extensive overview of BSD-driven content adaptation, we would like to refer the interested reader to [2].

4 Performance analysis

4.1 Application scenario

This paper targets a static client-server application, where heterogeneous clients may consume images in two different coding formats: JPEG 2000 and HD Photo. The images, pre-coded in a lossless and scalable manner, support a number of common adaptivity provisions that are to be exploited in an on-the-fly fasion (i.e., while handling a user request): lossless-to-lossy degradation of the perceptual image quality, spatial scalability, and interactive and tile-aligned ROI extraction. Interactive means that the location of the ROI is only known after encoding. Consequently, a tile structure with a sufficiently fine granularity has to be used during encoding, a requirement that will impact the coding efficiency (see Section 4.3.1).

A user with a mobile device may for instance communicate with the server as follows. First, after having selected a particular coding format, the user may request a low-resolution image that fits the limited display resolution of the device used. This request can be implemented by extracting a low-resolution version of the selected image (i.e., by exploiting spatial scalability). Next, the user may request a high-quality version of a semantically meaningful region that fits the resolution of the device at a high quality level. This request can be implemented by first extracting an ROI and by subsequently dropping a quality layer in order to go from a lossless to a lossy representation. Determining an ROI can be done either manually, with user intervention, or automatically, by making use of a content-aware image resizing tool.

4.2 Evaluation methodology

Integrating HD Photo and JPEG 2000 in a BSDL-based adaptation system required the design of BS Schemata, which allow exposing the high-level structure of an HD Photo and JPEG 2000 bitstream as a BSD, and a number of stylesheets, which allow transforming the BSDs in a pipelined fashion. The stylesheets were written in Streaming Transformations for XML (STX), because of its low computational complexity, low memory footprint, and streaming capabilities [3][1].

For HD Photo, a BS Schema and two stylesheets were newly designed: one stylesheet for dropping subbands and one stylesheet for interactive and tile-aligned ROI extraction. For JPEG 2000, the BS Schema and the XSLT stylesheets in the MPEG-21 DIA reference software repository were used as a starting point: the BS Schema was extended with support for handling multiple tiles in a JPEG 2000 bitstream, while the Extensible Stylesheet Language Transformations (XSLT) stylesheets for exploiting spatial and quality scalability were rewritten in STX. Also, a new STX stylesheet was written to support interactive and tile-aligned ROI extraction, as well as a new STX stylesheet for changing the packet order in a JPEG 2000 bitstream from spatial to quality progression (given the requirement of spatial and quality scalability in the example application discussed in Section 4.1). BSD-based adaptation for JPEG 2000 has been discussed before by Panis *et al.* in [2].

[1] A demo can be found at `http://multimedialab.elis.ugent.be/bflavor/`.

Table 3: Bitstream characteristics.

name	file size (KB)	textual BSD size (KB)	compressed BSD size (KB)
breeze.wdp	2590 (2590)	4 (5)	1 (1)
kungfu.wdp	2208 (2208)	4 (5)	1 (1)
night.wdp	2002 (2002)	4 (5)	1 (1)
plane.wdp	2571 (2571)	4 (5)	1 (1)
waves.wdp	2278 (2278)	4 (5)	1 (1)
breeze.j2c	2483	15	2
kungfu.j2c	1956	15	2
night.j2c	1451	15	2
plane.j2c	2394	15	2
waves.j2c	2063	15	2

Operations marked with a '+' symbol in Table 2 cannot be supported by BSDL, due to its lack of support for entropy coding and data reordering at macroblock level. Indeed, the vision of the investigated adaptation system is to only support editing-style operations in the compressed domain, facilitated by descriptions of the high-level bitstream syntax that abstract the internal complexity of the format processed.

All experiments were carried out on a PC with an Intel Pentium 4 2.6 GHz processor and 2 GB of system memory, running Windows XP Pro SP2 and Sun Microsystems Java 2 Runtime Environment (Standard Edition version 1.5.0_05). The memory consumption of the Java programs was measured by relying on JProfiler 4.2.1. Version 1.2.1 of the BSDL reference software was used, supporting the context management attributes as introduced in a revision of BSDL by MPEG. These attributes allow keeping the internal memory consumption of the BintoBSD Parser constant when generating BSDs [1]. Compressing the textual BSDs was done using WinRAR 3.70. The STX stylesheets were executed using the Joost STX processor (version 2005-05-21). All time measurements were executed six times – an average was calculated over the last five runs to mitigate startup effects.

Five 24-bits RGB test images with a resolution of 1920×1072 were used: Breeze, Kungfu, Night, Plane, and Waves. Each image respectively maps to the first frame of one of the five VIPER HD video sequences, as distributed by FastVDO during the MPEG meeting in Munich. The images were encoded in a lossless and scalable manner, once by using the Kakadu v5.2.5 JPEG 2000 encoder and once by relying on the encoder in Microsoft's HD Photo Device Porting Kit 1.0. Table 3 provides more information about test bitstreams containing one spatial

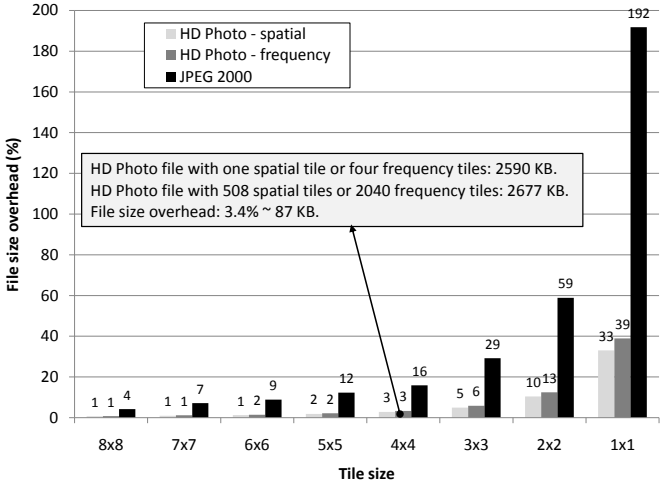

Figure 3: File size overhead (for Breeze).

tile. For HD Photo, the numbers between brackets denote size information for bitstreams in frequency mode.

4.3 Experimental results

Our performance analysis is realized by breaking up the XML-based adaptation system in four elementary steps: scalable bitstream creation, BSD creation, BSD transformation, and adapted bitstream creation. The first step analyzes how offering adaptivity in the compressed domain influences the coding efficiency. The next three steps are analyzed in terms of their memory and computational complexity. The file size overhead of textual and compressed BSDs is also taken into account.

4.3.1 Creation of scalable bitstreams

Figure 3 shows the file size overhead of HD Photo and JPEG 2000 representations of the Breeze test image, for a uniform tile size varying between 8×8 and 1×1 macroblocks[2]. The loss in coding efficiency is measured compared to an HD Photo and JPEG 2000 representation of Breeze that only contains one spatial tile (see Table 3). The following observation can be made: the smaller the tile size used, the finer the interactive ROI selection, but the lower the coding efficiency. For HD Photo, a tile size of 4×4 macroblocks (64×64 pixels) is feasible in terms of file size overhead, which is less than 4% for all five test images used. For JPEG 2000, to achieve an overhead that is strictly less than 4% for all test images used, a tile size of 12×12 macroblocks (192×192 pixels) is required.

The file size overhead for JPEG 2000 is significantly higher than for HD Photo, due to the lack of intra prediction in JPEG 2000 and a broken context for entropy coding. In particular, the smaller the tile size, the more a JPEG 2000 encoder acts as a block-based encoder (instead of acting as a global, tile-based encoder). A JPEG 2000 encoder cannot efficiently deal with this, as shown by the significant decrease in coding efficiency for a tile size of 1×1 macroblocks. Intra prediction in HD Photo is still active when using a tile size of 1×1 macroblocks, thanks to the use of blocks of 4×4 samples for intra prediction.

[2]As the concept of a macroblock does not exist in JPEG 2000, we assume that a macroblock also refers to 16×16 pixels in the context of JPEG 2000.

Table 4: Trade-off tile size, image file size, and BSD file sizes.

overhead for HD Photo (frequency mode)				
tile size	4×4	5×5	6×6	8×8
image file size (%)	[2, 4]	[1, 3]	[1, 2]	[0, 1]
textual BSDs (%)	[25, 34]	[17, 23]	[11, 15]	[6, 9]
compressed BSDs (%)	[1, 2]	[0, 2]	[0, 1]	[0, 1]
overhead for JPEG 2000 (progression by resolution)				
tile size	4×4	12×12	14×14	16×16
image file size (%)	[15, 25]	[2, 4]	[2, 4]	[1, 3]
textual BSDs (%)	[202, 323]	[28, 46]	[20, 35]	[18, 35]
compressed BSDs (%)	[5, 10]	[1, 2]	[0, 2]	[0, 2]

The file size overhead of HD Photo bitstreams in frequency mode compared to HD Photo bitstreams in spatial mode is due to additional tile headers and a larger index table: in frequency mode, the same amount of coded image data is partitioned over a number of tiles that is four times higher than in spatial mode. In short, using the frequency mode only has a limited impact on the coding efficiency, while it enables quality and spatial scalability, features not available in spatial mode. For example, for the Breeze test image in frequency mode, compared to the Breeze test image in spatial mode, the file size overhead can be ignored when the image contains one spatial tile (as shown in Table 3), while the overhead of the tile headers and the index table amounts to 4.5% or 154 KB when the Breeze test image contains 8040 spatial tiles (or 32 160 frequency tiles).

4.3.2 BSD creation

For the five test images used, Table 4 clarifies the trade-off between the tile size granularity, the file size of compressed images, and the file size of textual and compressed BSDs. Knowing this trade-off is important for enabling tile-aligned and interactive ROI extraction in an efficient way, using BSDL. In general, the file size overhead of textual BSDs is significantly high, but can also be significantly reduced when making use of compressed BSDs on the server (which is feasible for our application scenario, as bitstreams and their associated BSDs are created offline). For HD Photo, a tile size of 4×4 macroblocks results in a feasible file size overhead for the compressed images (between 2% and 4%) and the compressed BSDs (between 1% and 2%), while still enabling interactive ROI extraction with a feasible granularity. For JPEG 2000, to achieve a file size overhead for the compressed images and BSDs that is similar to the overhead of HD Photo,

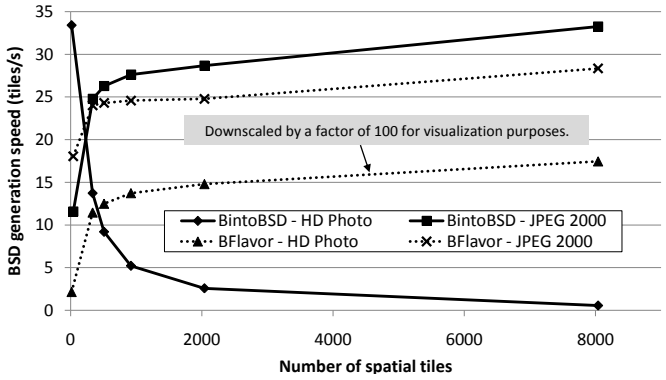

Figure 4: BSD generation speed (for Breeze).

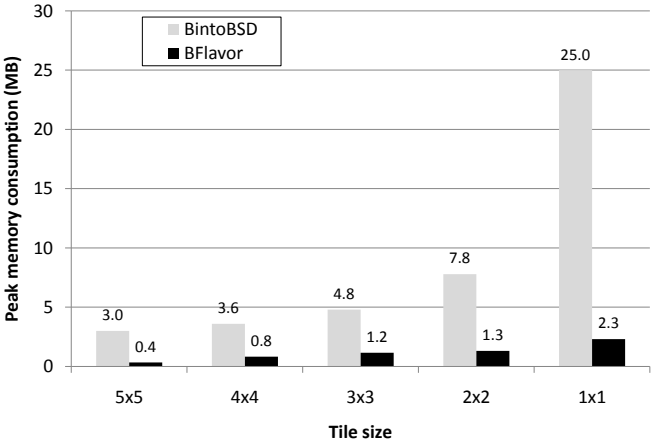

Figure 5: Memory consumption (for Breeze).

a tile size of 12×12 macroblocks needs to be used. The observations regarding the efficient compression of BSDs are also in line with results previously presented in [2].

Figure 4 illustrates the speed at which BSDs can be generated by the format-independent BintoBSD Parser for HD Photo representations of Breeze. The following observation can be made: the BSD generation speed is constant for an HD Photo file with a particular number of tiles, but decreases for HD Photo files with a higher number of tiles. This behavior is due to the higher number of entries in the index table, which is accessed multiple times by the BintoBSD Parser when processing a tile (e.g., to compute the payload length of a tile, to check whether a tile bitstream offset is not equal to zero, and so on). Consequently, the more entries in the index table, which is internally represented by BintoBSD using an XML tree, the more verbose the internal context of BintoBSD, thus the more time needed to access the index table using regular expressions taking the form of rather complicated XPath expressions. As shown in Figure 4, this performance behavior cannot be observed for BintoBSD when parsing JPEG 2000 bitstreams: similar to most other media formats, start codes are used in JPEG 2000 to detect packet (i.e., tile) boundaries, thus not requiring the execution of XPath expressions by BintoBSD[3].

To generate BSDs, we have also used BFlavor [5]. BFlavor (BSDL + XFlavor) is a modification of the Formal Language for Audio-Visual Object Representation, extended with XML features (XFlavor; [6]). This modification allows for the automatic creation of a format-specific parser, based on a manually designed description of the high-level syntax of the format in question using a Java-alike approach[4]. This parser is then able to produce BSDs that are equivalent to the BSDs generated by the format-independent BintoBSD Parser.

Figure 4 shows that BFlavor significantly outperforms BintoBSD in terms of BSD generation speed for HD Photo. This can be explained by the internal data model used: to represent and access the index table of HD Photo, BintoBSD uses an XML tree and XPath, where BFlavor uses arrays and simple indexing operations. On the other hand, the BSD generation speed of BintoBSD and BFlavor is similar for JPEG 2000. This can be attributed to the fact that JPEG 2000 is a low-complexity format in terms of parsing, not requiring the execution of complicated XPath expressions against a verbose context by BintoBSD. As such, I/O (i.e., the size of the textual BSDs) is the main bottleneck in the BSD generation process for JPEG 2000, for both BintoBSD and BFlavor. The latter also explains why BFlavor's BSD generation speed for HD Photo is significantly higher than for JPEG 2000, as BSDs for JPEG 2000 tend to be more verbose than for HD Photo, given a particular tile size (as shown in Table 3 and 4).

The peak memory consumption for BintoBSD and BFlavor is illustrated in Figure 5 for a number of HD Photo representations of Breeze. The following observation can be made: the memory consumption is constrained for an HD Photo file with a fixed number of tiles (thanks to the use of the context management attributes). However, the memory consumption increases as the number of tiles increases in an HD Photo file, which is due to the increasing size of the index table (which is entirely kept in the system memory, as this syntax structure is parsed before tiles are parsed). For JPEG 2000, the peak memory consumption is low and constant for both BintoBSD and BFlavor, and independent of the number of tiles present in a particular file (as parsing is done using start codes). As an example, for the JPEG 2000 representation of Breeze, the peak memory consumption has a constant value of 1.4 MB for BintoBSD and 0.7 MB for BFlavor.

4.3.3 BSD transformation and adapted bitstream creation

Transforming BSDs can be efficiently realized using STX. For example, for the HD Photo and JPEG 2000 bitstreams with

[3]Start codes are also specified by HD Photo, but cannot be reliably used for parsing purposes due to a lack of means for preventing start code emulation.

[4]A parser can be automatically generated from a manually designed description in BFlavor using any Java compiler. For the same media format, different parsers have to be generated for different BSD layouts.

a tile size of respectively 4×4 and 12×12 macroblocks, all transformations (as implemented in STX) can be separately performed in less than 1.8 seconds for HD Photo, and in less than 4.3 seconds for JPEG 2000, using less than 2 MB of memory in both cases. A similar observation can be made for creating the adapted bitstreams using BSDtoBin. For the HD Photo and JPEG 2000 bitstreams with a tile size of respectively 4×4 and 12×12 macroblocks, all adapted bitstreams can be generated by BSDtoBin in less than 1.3 seconds for HD Photo, and in less than 7.0 seconds for JPEG 2000, again using less than 2 MB of memory. The difference in execution times between HD Photo and JPEG 2000 is due to the bitstream reordering in JPEG 2000, which is computationally expensive and I/O intensive. However, in general, BSD transformation and adaptated bitstream creation can be done efficiently. This is an important observation for our use case, as these steps need to be implemented in a pipelined and on-the-fly fashion (in contrast to creating scalable bitstreams and their BSDs).

5 Conclusions and future work

This paper discussed the integration of HD Photo and JPEG 2000 in a format-independent adaptation system using MPEG-B BSDL and STX. As BSDL only supports high-level, editing-style adaptation operations in the compressed domain, only a subset of the adaptivity provisions in HD Photo and JPEG 2000 could be exploited, in particular quality and spatial scalability, as well as interactive and tile-aligned ROI extraction. For JPEG 2000, BSDL also allows exploiting color component scalability and changing the bitstream progression order. However, low-level operations in the compressed domain, such as rotating an image, a feature typically required by image sharing services, cannot be supported by BSDL, due to a lack of support for entropy coding and data reordering at macroblock level. This leaves open the question of how to support all possible compressed and uncompressed domain operations in a unified, BSDL-based adaptation architecture to meet the requirements of present-day and future multimedia applications.

Our research also investigated the loss of coding efficiency when providing adaptivity provisions in the compressed domain, as well as the complexity of the adaptation system used. In general, adaptivity in HD Photo and JPEG 2000 can be provided and exploited in a BSDL-based adaptation system with a feasible loss in coding efficiency and a feasible computational and memory complexity by choosing an appropriate tile size during encoding: for the test images used, a tile size of 4×4 and 12×12 macroblocks was respectively appropriate for HD Photo and JPEG 2000. To efficiently generate BSDs for HD Photo, BFlavor had to be used, instead of BSDL's BintoBSD Parser, due to BintoBSD's inefficient representation and processing of HD Photo's index table.

Future research might investigate the efficiency of XPath variables in the context of HD Photo. XPath variables, recently introduced in BSDL during a revision of the language by MPEG, may allow for a more efficient handling of HD Photo's index table as BintoBSD can then make use of an internal data model that is similar to the data model used by BFlavor. At the time of writing, support for XPath variables was not integrated (yet) in the BSDL reference software by its proponents. Future research may also address alternative architectures for format-independent adaptation, such as architectures in which format-specific bitstream extractors are automatically generated, no longer using BSDs for adaptation purposes. Finally, attention might be paid to the JPEG 2000 XML (JPXML) structure initiative.

Acknowledgements

The research described in this paper was funded by Ghent University (UGent), the Information and Communications University (ICU), the Interdisciplinary Institute for Broadband Technology (IBBT), and the Brain Korea 21 (BK21) program of the Korean Ministry of Education.

References

[1] D. De Schrijver *et al.* An Optimized MPEG-21 BSDL Framework for the Adaptation of Scalable Bitstreams. *Journal of Visual Communication & Image Representation*, 18(3):217–239, June 2006.

[2] G. Panis *et al.* Bitstream syntax description: a tool for multimedia resource adaptation within MPEG-21. *Signal Processing: Image Communication*, 18(8):721–747, September 2003.

[3] P. Cimprich *et al.* Streaming Transformations for XML (STX). Technical report, Available on `http://stx.sourceforge.net/`, 2007.

[4] S. Srinivasan *et al.* HD Photo: a new image coding technology for digital photography. In *Proceedings of the SPIE*, volume 6696, San Diego, USA, August 2007.

[5] W. De Neve *et al.* BFlavor: a harmonized approach to media resource adaptation, inspired by MPEG-21 BSDL and XFlavor. *Signal Processing: Image Communication*, 21(10):862–889, November 2006.

[6] D. Hong and A. Eleftheriadis. XFlavor: Bridging Bits and Objects in Media Representation. In *Proceedings of ICME*, pages 773–776, Switzerland, August 2002.

[7] A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Processing Mag.*, 18(5):36–58, September 2001.

[8] A. Vetro and C. Timmerer. Digital Item Adaptation: Overview of Standardization and Research Activities. *IEEE Trans. Multimedia*, 7(3):418–426, June 2005.